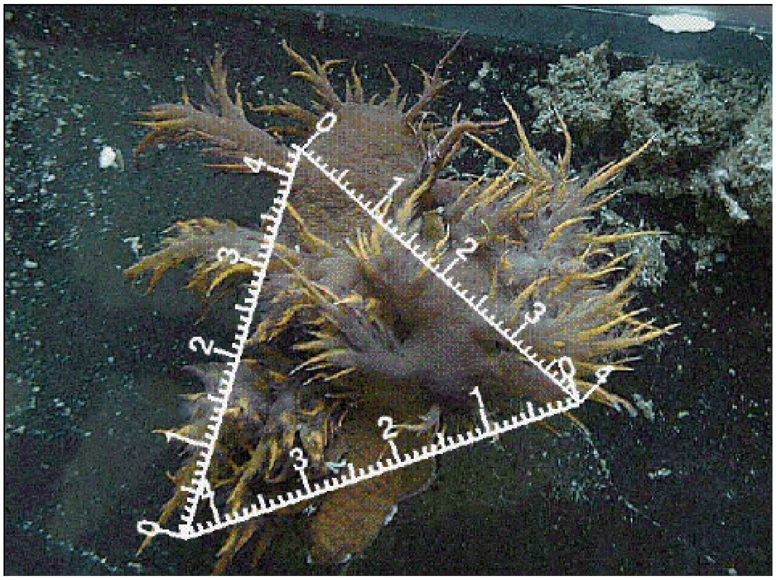


Image Measurement and Analysis Lab (tnimage)

Description of tnimage
Image Analysis Software

T. Nelson (tjnelson at brneurosci dot org)

Latest version: 3.5.1
Latest revision: April 24, 2006

e-mail:
tjnelson at brneurosci dot org

ftp sites:
ftp://entropy.brneurosci.org/pub/unix (all Unix versions)
http://entropy.brneurosci.org/tnimage.html (all versions)
ftp://entropy.brneurosci.org/pub/dos/ (DOS version)
A collection of sample test images is also available at these sites.

Contents

Section 1

Introduction

1.1 Summary

`Tnimage` is an image analysis program oriented toward scientific and technical applications. It has been extensively used by molecular biologists, forensic pathologists, biochemists, physicists, and others to analyze images. It is also useful for general image viewing and editing. The program has an easy to use, menu-driven interface based on Motif. Precompiled binaries are available for several Unix platforms, and an earlier version is available for MS-DOS.

This manual principally describes the Unix version of `tnimage`. Many features of this manual pertain to both the DOS and Unix versions. However, DOS users should consult the file `tnimage.doc` for a more accurate description, since some of the latest features are only available in the Unix version.

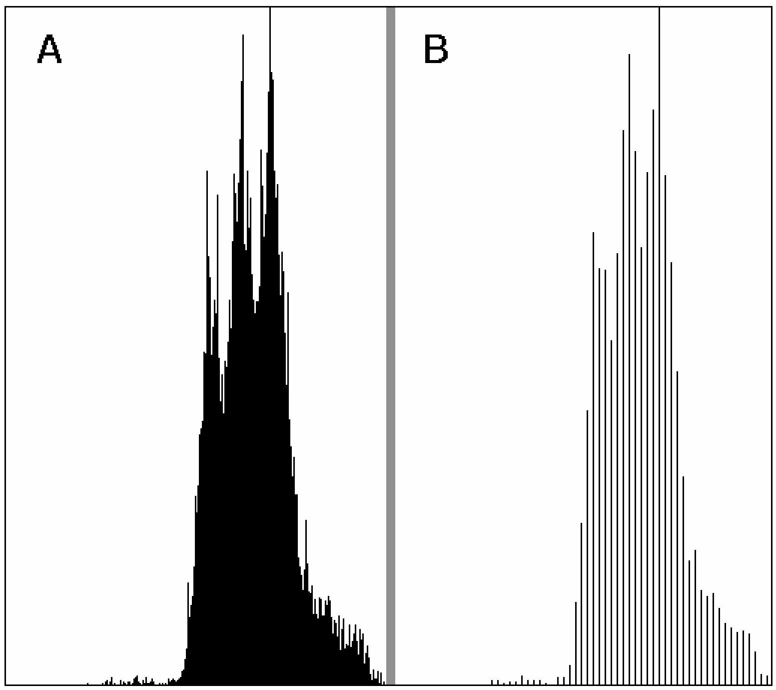
1.2 Features

- Grain counting and size distribution
- Morphological operations (erosion, dilation, segmentation, etc.)
- Wavelet transforms
- Movable text labels; smart selection of objects in images.
- Images can be viewed directly or in an editable spreadsheet using integers, RGB values, hex integers, or (if an image has been FFT'd) floating point numbers. Changes in the image are immediately reflected in the spreadsheet and vice-versa.
- Image registration
- Transparency, chromakey, and paste functions facilitate creation of composite images.
- Scanner interface for scanners with preview scan and interactive image scanning at 8, 10, 12, 24, 30, and 36 bits/pixel (Not available in ConvexOS and MS-DOS versions).
- Create, cut/paste, and add text labels in multiple fonts and graphic elements such as circles, Bezier curves, freehand drawing, etc.
- Handles up to 512 images of any depth simultaneously. Each image can be in a separate window or in a single large window to facilitate creation of composite images. Cut/paste works even if images are of different depths or in different windows.
- Prints to PCL or PostScript printer. CMY, CMYK, or RGB formats.
- Import/export formats: PCX, IMG, TIFF (both Macintosh and PC), JPEG, BMP, GIF, TGA, IMG, Lumisys Xray scanner, XDM, FITS, PDS, PBM, PGM, PPM, PNG and ASCII images, of any depth from 1-32 bits per pixel, color or monochrome, raw binary images, and 3D images (such as PET scan and confocal images). Handles unusual image

depths such as 12- and 17-bit grayscale. HDF format images are readable with the supplied plugin `readhdf`. Import only of BioRad scanner images.
- Users can create new custom image file formats to transparently read and write images in unusual image formats.
- Interconversion of image formats.
- Solid and gradient flood fill.
- Color gradient-filled wide lines with adjustable size arrow heads.
- R, G, and B image planes can be manipulated separately.
- Adjust color, intensity, contrast, and grayscale mapping. Grayscale images deeper than 8 bits/pixel, such as medical grayscale images, can be viewed with a sliding scale to enhance any particular intensity region.
- Rotate, resize, warp, flip, invert or remap colors; crop, paint, spray paint, etc.
- Labels and images can be rotated at any angle.
- Spray-filtering and spray-math to enable interactive image manipulation on local regions of an image as a spray.
- Convolution filters: sharpen, blur, edge enhancement, shadow sharpening, background subtract, background flatten, and noise filter.
- Interactively create arbitrary colormaps or select from 10,000 pre-defined colormaps.
- Macro language and macro editor. Macro programming guide is included. Macro editor also functions as a command line interface.
- Image algebra function allows multiple images to be subtracted or otherwise transformed according to arbitrary user-defined equations.
- RGB and intensity histograms.
- Multi-frame or 3D images can be viewed interactively as a movie, or each frame can be manipulated separately.
- Spot densitometry of rectangular regions or arbitrary-shaped areas. Edges of spots and background can be determined automatically. Results can be easily exported to disk in ASCII format.
- Strip (scanning) densitometry of transepts, fixed-width rectangular regions, and trapezoidal areas. This is particularly useful for densitometry of SDS and agarose gels. Can automatically subtract baseline and integrate peak areas, or interactively measure peaks with the mouse. Peak area table can be saved in ASCII format.
- 2D-Fourier transform. Real, imaginary, or power spectra can be viewed and edited.
- FFT frequency components can be changed using the drawing or image algebra functions and transformed back to the real image. A tutorial is provided for digital filtering using the FFT. FFT arrays can also be exported in ASCII format for manual editing or edited in the spreadsheet.
- Convolution of 2 images and image reconstruction using Fourier deconvolution.
- Distance and angle measurement.
- Curve tracing.
- Calibration of X-Y coordinates using linear, polynomial, or exponential functions. For example, the logarithm of image coordinates in one direction can be expressed as molecular weight. Calibrating the pixel intensity is useful for images in which height is represented as a pixel value, such as atomic force microscope images.
- Users can remove or add custom buttons on left side of the main window. The buttons can execute any command or macro, and can execute a different command when un-clicked.
- Users can create plugins to add new image-processing features.
- Highly configurable in terms of fonts, colors, etc.
- Partitioning of images using fuzzy k-means algorithm.
- Zooming of images by an arbitrary factor.

1.2.1 Differences between tnimage and other programs

The biggest difference between `tnimage` and other imaging software is that `tnimage` is oriented toward scientific and technical applications. This requires a lot of additional programming to ensure that the numeric values in the image are accurate, and that information is not lost for the sake of convenience.



Histogram of image produced by tnimage (A) and a well-known Unix image viewer (B).

For example, the figure above shows a histogram of the same 16-bit image which was opened and then saved in tnimage (left) and another, well-known Unix image viewer (right). In this other program, the "Normal Size" box was checked and every other precaution was taken to ensure that the image was saved correctly. The visual appearance of the two images on the screen was identical. Yet the graph shows that the other program had cut corners and saved only 55 of the original 65,536 different intensity values. The other 99.9% of the information was thrown away. Similar, although not as extreme, situations can happen for color images as well – information not required for displaying the image is discarded without your knowledge for the sake of processing speed, because the program is only designed to display images, not to analyze them numerically.

Another example occurs in densitometry. Because of this concern for maintaining data integrity, results obtained in `tnimage` can be significantly more accurate than those obtained from other programs. Part of this is also due to the algorithms used in `tnimage`, some of which were invented for this express purpose. Of course, for some applications, the accuracy doesn't matter. However, the user is usually not aware that the other software takes shortcuts for the sake of speed. This can often produce inaccurate or unusable results.

For example, I was once asked to analyze an image of a Western blot that had very faint bands compared to the background. The person could not repeat the experiment, since these were irreplaceable autopsy samples, and had first tried to analyze the image with another well-known Macintosh-based image analysis program. Unfortunately, it turned out that the other program was skipping some crucial calculations in its calculation of the background level, resulting in errors of 50% with each measurement. Even after taking multiple measurements and discarding those which gave negative numbers, it was impossible to get credible numbers from the other program. Increasing the image contrast merely increased the error by the same factor. Using `tnimage`, the background could be determined to several decimal points, and clear-cut results were easily obtained.

Because `tnimage` was written by a research biochemist, the examples in this manual tend to be oriented toward biochemistry. However, the algorithms in this program were designed to be flexible enough to accommodate the requirements for technical image analysis in any field. If you have an application that is not handled adequately by `tnimage,` or if you have a suggestion for new features or would like to contribute to the `tnimage` project, we would like to hear from you.

1.3 Usage license

Use and distribution of this software is subject to the conditions in the GNU Public License (GPL), version 2. The text of the GPL is available from the Free Software Foundation.

1.4 Summary of features in different versions

Feature	DOS	Linux	Irix	Solaris	ConvexOS
Max. images	512	512	512	512	512
Image editing	•	•	•	•	•
Image conversion	•	•	•	•	•
Format conversion	•	•	•	•	•
Labeling	•	•	•	•	•
Drawing	•	•	•	•	•
Multiple fonts		•	•	•	•
Image algebra	•	•	•	•	•
Grayscale > 8 bits	•	•	•	•	•
Histograms	•	•	•	•	•
Selectable color planes	•	•	•	•	•
Image calibration	•	•	•	•	•
TIFF,TGA,BMP,IMG	•	•	•	•	•
PCX,GIF,Raw,ASCII	•	•	•	•	•
PPM, PBM, PGM, PNG		•	•	•	•
Custom image formats	•	•	•	•	•
JPEG	•	•	•	•	•
3D/multiple frames		•	•	•	•
Densitometry	•	•	•	•	•
Fourier transform	•	•	•	•	•
Deconvolution	•	•	•	•	•
Selectable colormaps	10000	10000	10000	10000	10000
Filters	13	13	13	13	13
Virtual memory	•	•	•	•	•
Peak Area analysis	•	•	•	•	•
PCL Printing	•	•	•	•	•
PostScript Printing	•	•	•	•	•
Macro language	•	•	•	•	•
Image transparency		•	•	•	•
Chromakeying		•	•	•	•
Scanner interface		•	•	•	
Camera interface		•	•	•	•
Plugins		•	•	•	
User-defined buttons		•	•	•	•
Spreadsheet		•	•	•	
Fuzzy partitioning		•	•	•	
Morphology		•	•	•	

1.5 Submitting bug reports

If tnimage crashes, hangs, draws the screen incorrectly or fails to read or write a valid image, it is probably a bug. If it causes some other undesired effect (such as starting World War III), this may also be a bug (*Note:* this should not happen in normal use, and is not covered under Warranty). I would greatly appreciate receiving bug reports from all users. Bug reports should be sent to the above Internet e-mail address. If you send a bug report, please run the command “tnimage -diag > logfile” first and include this log file with your report. Also indicate your computer type, video card, and monitor type, what resolution you are trying to use, and the operating system and OS version you are using, and any other relevant information. Most importantly, indicate what version of tnimage you are using, the size and type of image involved, and the sequence of actions that cause the problem. If the problem only occurs with a specific image, you may also need to send a copy of the image file. The easiest way to do this is to email it to me. All bug reports will receive immediate attention.

Section 2

Installation instructions

2.0.1 System requirements

- X11R5 or higher and Motif 1.2 or higher. Partially-statically-linked version does not require Motif. Fully-statically-linked version for libc5 or a.out systems.
- Precompiled binaries are provided for the Linux (x86) (elf/glibc), Solaris (UltraSparc), Irix (MIPS), and MS-DOS.
- Source code requires Motif, libjpeg, libtiff, libpng, and Xbae widgets to compile.
- DOS version requires SVGA card and 4 MB of RAM; handles all VESA screen modes including 1600x1200, as well as XGA and XGA-2 video cards.

2.0.2 Installation overview

To install a precompiled version of `tnimage`, type `Install` as root. If this doesn't work, `tnimage` can be installed manually as described here.

Tnimage uses several auxiliary files which by default are installed in the following directories:

tnimage	/usr/local/bin/
tnimage.hlp	/usr/local/lib/tnimage/
tnimage.1.gz	/usr/man/man1
formats/*	/usr/local/lib/tnimage/formats/
wavelets/*	/usr/local/lib/wavelets/
plugins	/usr/local/lib/tnimage/
scanners	/usr/local/lib/tnimage/
umax	/usr/local/lib/tnimage/
cameras	/usr/local/lib/tnimage/
quickcam	/usr/local/lib/tnimage/

In addition, 2 files are generated automatically and placed in the following directory, which is created if necessary:

fonts	$HOME/.tnimage/
tnimage.ini	$HOME/.tnimage/

(or $home/.tnimage/ if you are using csh or tcsh). The user must have read/write and execute permission on this directory.

The tutorial image files which are included with tnimage, and the manual may be installed wherever convenient.

The location for the help file and formats file can be changed if desired by editing the "help_directory" and "format_list" entries, respectively, in tnimage.ini. This permits advanced users to create their own new file formats, while still allowing a system administrator to create formats and make site-specific changes to the help file for regular users. Such changes might include, for example, the locations of the manual and tutorials and instructions for viewing them.

2.1 Compiling

`Tnimage` requires Motif or some Motif clone such as Lesstif to compile. It also requires the JPEG and PNG libraries. Other libraries, including Xbae, and lex and yacc, will be used if available. Otherwise, `tnimage` will still compile, but certain functions (the spreadsheet for Xbae, and image algebra for lex and yacc) will be disabled.

2.1.1 General compiling instructions

- Ensure that a C++ compiler, libjpeg, libtiff, libpng, and optionally libXbae are present on your system (run `ldconfig` if necessary).
- Type `./configure` This creates the file `config.h` appropriate for your system. *Do not skip this step.* The file `config.h.example` is provided in case `configure` doesn't work for some reason, and is not normally needed for compilation.
- To disable the Freetype anti-aliased fonts, type `./configure --disable-xft`. Freetype has many problems because of things that happened during its development. The current state of freetype/truetype fonts in Linux is primitive. There are numerous incompatible versions with little documentation and clunky APIs, and many of the libraries will not compile or run correctly. Because of this, freetype fonts were made an optional feature. See below for details on installing freetype.
- Type `make`. If a previous version exists, type `make clean` first.
- Type `make install`(as root).
- Check the contents of

  ```
  $HOME/.tnimage/scanners
  $HOME/.tnimage/plugins
  $HOME/.tnimage/cameras
  ```

  and edit if necessary to indicate the paths of your scanner drivers, plugins, and camera drivers, respectively.

For most systems, this is all that is necessary.

Important: Do not run `flex` or `lex` on `calculator.l` unless you wish to make changes in the math parsing algorithm. Running flex will overwrite the file `lex.yy.c` . If this happens, it will be necessary to manually edit `lex.yy.c`. See the top of `calculator.y` for instructions.

2.1.2 Installing Xbae

To obtain the spreadsheet functionality, the Xbae widget set (from ftp.x.org) must be installed before running 'configure'. If Xbae does not install correctly by itself, the following setup is known to work, at least in Linux:

```
cd xbae (start in the directory where you compiled libXbae)
cp libXbae.a /usr/X11R6/lib/
cp libXbae-4.6.a /usr/X11R6/lib/
mkdir /usr/X11R6/include/Xbae-4.6
ln -s /usr/X11R6/include/Xbae /usr/X11R6/include/Xbae-4.6
cp *.h /usr/X11R6/include/Xbae-4.6
```

Check to make sure Matrix.h is in /usr/X11R6/include/Xbae-4.6, then delete config.cache and rerun ./configure and make in the tnimage source directory.

2.1.3 Problems with Freetype

If you get the message:

```
Xlib : extension RENDER missing on display ":0.0"
```

this means your X server is too old or is missing the RENDER extension needed to render anti-aliased fonts. You need to either upgrade XFree86 or recompile tnimage without font aliasing as described above.

If you get the message

```
In file included from xmtnimage.h:91,
              from xmtnimage.cc:9:
/usr/X11R6/include/X11/Xft/Xft.h:52: error: syntax error before ‘;’ token
/usr/X11R6/include/X11/Xft/Xft.h:86: error: ’FT_UInt’ is used as a type, but is
   not defined as a type.
/usr/X11R6/include/X11/Xft/Xft.h:93: error: ’FT_UInt’ is used as a type, but is
   not defined as a type.
```

this means there is something wrong with your Xft or Freetype installation. libXft is available from `http://pdx.freedesktop.org/x̃libs/release/` or `fontconfig.org` . Freetype is available from `http://freetype.org/`. To get freetype to work, do the following:

1. Download and install the latest version of libXft and Freetype. For both programs, this includes the standard three steps:

   ```
   tar -xzvf <filename>
   cd <directory>
   ./configure
   make
   make install
   ```

   LibXft will install itself in `/usr/local`. Freetype installs in `/usr/include/freetype2`. Before they will work, it is necessary to move the files to the correct locations.

2. Copy the libXft libraries to /usr/X11R6.

   ```
   cd /usr/local/lib/
   cp libfree* /usr/lib/
   cp libXft*  /usr/X11R6/lib/
   ldconfig
   ```

3. Copy the include files to /usr/include.

```
cd /usr/local/include
ln -s freetype2 freetype
cd /usr/local/include/X11/Xft
cp * /usr/X11R6/include/X11/Xft/
```

4. Make links so that freetype points to freetype2.

```
cd /usr/include
mv freetype freetype.bak
ln -s freetype2 freetype
cd /usr/local/include/freetype
cp -R freetype/* .
```

2.1.4 Problems with libpng

Some versions of libpng bomb out on line 255 of pngconf.h. To fix this, remove the two invalid lines of code from /usr/local/include/pngconf.h.

Change this:

```
#    ifdef _SETJMP_H
#      __png.h__ already includes setjmp.h;
#      __dont__ include it again.;
```

to this:

```
#    ifdef _SETJMP_H
/*      __png.h__ already includes setjmp.h;
       __dont__ include it again.;*/
#    endif
```

2.1.5 If the above procedure doesn't work

`Tnimage` can also be compiled by editing `makefile` and `config.h` and installing manually. LibXbae is available from ftp.x.org. If it is not present, tnimage will still compile, but the spreadsheet function for editing pixel values will be disabled.

Depending on how libXbae is compiled, you might also need libXpm (libXpm is not used in `tnimage` itself). If the compilation gives the following messages

```
undefined reference to 'XpmFreeAttributes'
undefined reference to 'XpmCreateImageFromXpmImage'
undefined reference to 'XpmFreeXpmImage'
undefined reference to 'XpmReadFileToXpmImage'
```

this means that you must install libXpm (or install a different version of libXbae) before continuing. Also, add "`-lXpm`" after "`-lXm`" in the makefile if it is not there already.

In addition to libjpeg.a, jpeglib.h must be present on your system. Make sure it is found by `configure` (it should say "jpeg...found"). In addition, it might be necessary to run `configure` in the jpeg-6a directory in order for jconfig.h to be created. Some Linux distributions do not include this essential file by default.

Libtiff is only required for the plugin "readtif" which reads FAX images, and is not required for reading/writing TIFF format images in tnimage.

Libpng is also required.

Note: There are some compilers out there that cannot handle overloaded functions in C++ correctly. These compilers produce large numbers of spurious warning messages, along with nonsensical error messages like:

```
xmtnimage.cc:2017:  no matching function for call to ‘message (char *&, void *,
const int &, int)’
```

If you get the following message:

```
In file included from /usr/include/errno.h:36,
              from xmtnimage.h:32,
              from xmtnimage45.cc:8:
/usr/include/bits/errno.h:25: linux/errno.h: No such file or directory
In file included from /usr/include/signal.h:294,
              from xmtnimage.h:43,
              from xmtnimage45.cc:8:
/usr/include/bits/sigcontext.h:28: asm/sigcontext.h: No such file or directory
In file included from /usr/include/bits/posix1_lim.h:126,
              from /usr/include/limits.h:30,
              from /usr/include/bits/socket.h:31,
              from /usr/include/sys/socket.h:34,
              from xmtnimage.h:45,
              from xmtnimage45.cc:8:
...
```

This means that the compiler was searching for necessary files in the Linux kernel source tree. In this case, it will be necessary to unpack a Linux kernel in /usr/src before compiling `tnimage`. Complaints should go either to your distribution vendor or the Free Software Foundation, Inc.

2.1.6 If it still won't compile

If all else fails, and for some reason it proves impossible to compile the program using `make`, as a last resort the program can be compiled manually as follows:

1. **Run `configure` to generate `config.h`** . Then check config.h and change it if necessary.
2. **Compile calculator.y:**

```
yacc -d calculator.y
cp y.tab.c y.tab.cc
gcc -c -x c++ -O3 -Wall -Wunused -W -I. -I/usr/include -I/usr/include/X11 \
  -I/usr/X11R6/include -I/usr/X11R6/include/Xbae  -I/usr/include y.tab.cc
```

3. Compile the C++ files. For example:

```
gcc -c -O3 -Wall -Wunused -W -I. -I/usr/include -I/usr/include/X11 \
-I/usr/X11R6/include -I/usr/X11R6/include/Xbae xmtnimage60.cc
```

The entries for Xbae should be omitted if libXbae is not installed. The above command must be executed for each *.cc file.

3. Link:

```
gcc -o tnimage -O3 -Wall -Wunused -W xmtnimage*.o lex.yy.o y.tab.o \
-L/usr/X11R6/lib  -lm -ltiff -lXext /usr/X11R6/lib/libXbae.a  \
-lXm -lXpm -lXext -lXt -lX11 -lpng -lfl /usr/lib/libjpeg.so  -s
```

The `-O3, -lXext`, and `-lXpm` entries, and the entries starting with `-W` are not essential. The `libXbae.a` entry should be omitted if `configure` did not find `libXbae`. Some systems may not accept the `xmtnimage*.o` . If so, it must be replaced by a list of all the `.o` files.

Then install the executable by typing `make install` as root.

2.2 Installation on Solaris 2.5

Compiling on Solaris

The makefile is configured for gcc and may need to be edited if Sun's compiler, which is an extra cost option, is installed. Otherwise, the general procedure above should work.

To install the man page on a Solaris machine, execute the following 3 commands as root:

`cp tnimage.1.gz /usr/man/man1`

`gunzip /usr/man/man1/tnimage.1.gz`

`catman 1`*(may take a long time)*

In Solaris, the Xbae widgets may not get installed in the correct locations by the Xbae makefile and must be installed manually to the following locations:

`libXbae.a = /usr/openwin/lib`

`include files = /usr/include/Xbae.`

Alternatively, they can be placed in a directory named “Xbae” under the xmtnimage source tree.

The gcc, libjpeg, libpng, perl, libtiff and m4 packages must be installed before `tnimage` will compile. Xbae, bison, make, xpm, and flex may also be needed.

To install a package file in Solaris, unzip the file with gunzip and enter the command: `pkgadd -d package_file_name` .

The libtiff package is also needed to compile the `readtiff` plugin. This plugin is not needed for `tnimage`.

All of these packages can be found at `http://sunfreeware.com` . After installing new libraries, it may be necessary to update the `ld.config` file using `crle` as shown:

```
crle -c /var/ld/ld.config -l /local/lib:/usr/lib:/usr/openwin/lib -s /local/lib
```

adding additional paths as necessary.

2.3 Installation on DEC Alpha

1. `./configure`
2. `make`
3. `make install` (as root)

On some systems, it might be necessary to change the first line of the Install script, e.g., to `#!/usr/local/bin/bash`. Please let me know if you have trouble compiling tnimage on DEC Alpha, otherwise the problem will never get fixed, as I don't have access to an Alpha for testing.

If it doesn't compile and you want to try to fix it yourself, keep in mind that the most likely source of compilation errors arises from the fact that the Digital compiler cannot handle certain common C++ idioms that may have sneaked into the source code.

Statements like

```
const int FOO=0;
```

must be changed to

```
#define FOO 0
```

and statements like

```
Widget bar[baz+1];
```

must be changed to

```
Widget bar[1024];
```

i.e., the compiler can't handle variables in array declarations. These must be replaced by suitable constants. Using constants that are too small will cause `tnimage` to crash. 1024 is usually a safe number.

Errors like

```
cxx: Error: xmtnimage.cc, line 999: redefinition of default argument
```

Are caused by statements like

```
void do_stuff(int a, int b=0)
```

that have sneaked into the code, which cause problems for DEC's compiler. The default argument "=0" can be safely removed from the function (but not from the prototype in `xmtnimage.h`).

Please notify me if these problems occur, so they can be fixed.

2.4 Installation on Mac OSX / Darwin

As of version 3.2.20, `tnimage` should also build on Max OSX (Darwin). The procedure below was contributed by a user. Unfortunately, I don't have access to a Mac, so there is no way to know whether the new standard procedure actually works. If the standard procedure (`configure; make; make install`) doesn't work, check the following:

1. A working version of Motif must be installed. Lesstif reportedly works in OSX and is included in the Fink distribution of OSX.
2. Test the `config.guess` script to find out whether it prints "darwin" or something similar. If not, replace `config.guess` with the one in /usr/libexec or /usr/share/libtool/, or with the following one line script:

   ```
   echo darwin
   ```

   Make sure the script is executable.
3. Type './configure'
4. Change 'makefile' to contain the following (if necessary):

   ```
   OBJS = xmtnimage*.o lex.yy.o y.tab.o
   LDFLAGS=
   INCLUDES=-I.  -I/sw/include  -I/sw/include/Xm -I/usr/X11R6/include -I/usr/X11R
   LIBS= -lm /sw/lib/libtiff.a /usr/lib/libl.a /sw/lib/libXm.a \
      /usr/X11R6/lib/libICE.a /usr/X11R6/lib/libSM.a /usr/X11R6/lib/libXpm.a \
      /usr/X11R6/lib/libXext.a /usr/X11R6/lib/libXt.a /sw/lib/libfl.a \
      /usr/X11R6/lib/libX11.a /usr/local/lib/libjpeg.a /usr/local/lib/libpng.a \
      /usr/local/lib/libtiff.a
      /usr/lib/libz.dylib /sw/lib/libjpeg.a
   CC=gcc
   CFLAGS=-O3 -fno-common   -Wall  -no-cpp-precomp
   YACC=yacc -d
   ```

   Note that your libraries may be in different locations than shown here. If this command doesn't work, use the `strings` or `grep` commands to determine which library files are needed for each unresolved symbol.
   For FINK, try the following:

   ```
   OBJS = xmtnimage*.o lex.yy.o y.tab.o
   LDFLAGS=
   INCLUDES=-I.  -I/sw/include -I/sw/include/Xbae -I/sw/include/Xm \
       -I/usr/X11R6/include -I/usr/X11R6/include/X11
   LIBS= -L/sw/lib -ltiff -ljpeg -lpng -lXbae -lXm -L/usr/X11R6/lib -lICE \
       -lXpm -lXt -lSM -lX11 -lXext -lm -fl -ll -lz
   CC=g++
   CFLAGS=-O3 -fno-common   -Wall  -no-cpp-precomp
   YACC=yacc -d
   ```

5. Comment out all the occurrences of fcvt() in config.h, i.e.:

   ```
   /* #undef HAVE_FCVT */
   ```

   Similarly, if gcvt() causes problems, cast it from thee:

   ```
   /* #undef HAVE_GCVT */
   ```

6. The command 'strip readtif' apparently does not work in OSX. This is of no consequence. `readtif`, which requires the libtiff library, also may not compile. Readtif is only a plugin and is not needed for `tnimage`.
7. Note that there is no 'make install'. Installation is performed with the `Install` script. However, even if this doesn't work for some reason, `tnimage` will still run if you copy the binary to someplace in your path, preferably /usr/local/bin.

Thanks to Kurt De Vos for this information and for his persistence in compiling `tnimage` on the Mac. Feedback from other Mac users is encouraged.

2.5 Installation on FreeBSD

Tnimage should compile in FreeBSD and NetBSD. It may be necessary to create symlinks for libtiff and libpng:

```
ln -s /usr/local/lib/libtiff* /usr/lib/
ln -s /usr/local/lib/libpng* /usr/lib/
```

If compilation fails with errors about operators 'new' and 'delete' make sure g++ is specified in the makefile, and not gcc. Tnimage requires a functional C++ compiler to build.

2.6 Installation on SGI Irix

Compiling the Irix version can be tricky because of multiple incompatible versions of gcc, and incompatibilities between gcc and the MIPSpro compiler. On systems in which both gcc and MIPSpro are installed, the situation is even worse due to n32 and n64 library incompatibilities. This is the reason a precompiled Irix version is only intermittently available. You should try to make a 32-bit version of the program if possible.

If gcc is the only compiler on the system, the general procedure above should work. Otherwise, set the CC environment variable to "gcc -mips2" or "cc" before running `./configure`, depending on whether gcc or the MIPSpro compiler is desired (i.e., type `setenv CC "gcc -mips2"`). On some systems, it may be necessary to change the line
`CF="-O2"`
to
`CF="-O2 -mips2"`
in configure.in, and then rerun `autoconf` and `configure`. The reason for this is the wide variety of different compilers on Irix, each of which requires different options, but which can't be distinguished by `configure`.

Note that some versions of the MIPSpro compiler also do not seem to handle C++ templates correctly. If you want to try it anyway do the following:

- Uncomment the section in `configure.in` for `*mips-sgi-irix6.5*` , by removing the "dnl".
- Run autoconf.
- Make sure the libraries required by tnimage (libXbae.a, libjpeg.a, libpng.a, libtiff.a and libXpm.a) are also compiled with the MIPSpro compiler. All files and libraries must be compiled with either with the "n32" or "n64" option.
- Run make clean, make, and Install as described above.

One user reported that only gcc 2.7.x worked with the -mips2 flag. The older gcc 2.7.x can be obtained from `http://reality.sgi.com/ariel_engr/freeware/`. You should also use the 32-bit libjpeg.so which can be obtained from `freeware.sgi.com` .

Libjpeg, the libjpeg include files, libtiff, and libpng must be on your system. If they are not found by `configure`, add the path to `configure.in` and rerun `autoconf` . If Xbae is not installed, and installing in the default locations is not possible, `libXbae.a` and its include files can be placed in a directory named "Xbae" under the xmtnimage source tree. If libXbae.a is not found, the program will still compile, but the spreadsheet will be disabled.

Finally, as if that weren't enough, Irix yacc produces incorrect C++ code. After running yacc, it is necessary to modify the following lines in y.tab.c:

```
#ifndef MIPS
extern int yylex(void);
#endif
```

should be changed to:

```
int yylex(void);
```

also,

```
#ifdef __cplusplus
extern "C" {
#endif
extern char *gettxt(const char *, const char *);
#if !defined(yylex) && !defined(__my_yylex)
        extern int yylex(void);
#endif
#ifdef __cplusplus
}
#endif
```

should be changed to

```
extern char *gettxt(const char *, const char *);
#if !defined(yylex) && !defined(__my_yylex)
        extern int yylex(void);
#endif
```

Flex and other libraries can be downloaded from `http://freeware.sgi.com/index-by-alpha.html`

If you install libflex, edit the makefile as shown below. If problems occur with mixed 32- and 64-bit libraries (for example `crt1.o`), /usr/lib32 should be specified explicitly. The makefile should look something like:

```
OBJS = xmtnimage*.o lex.yy.o y.tab.o
LDFLAGS=-s
INCLUDES=-I. -I/usr/include -I/usr/include/X11  -I/usr/include/X11 -I. \
     -I/usr/include
LIBS=-L/usr/lib32  -lm -ltiff -lXm -lXext -lXt -lX11 -lpng \
     /usr/lib32/libjpeg.so /usr/freeware/lib32/libfl.a
CC=CC
CFLAGS=-O2
YACC=yacc -d
```

If the resulting executable has difficulty reading TIFF files (e.g., "Out of memory" errors), the most likely cause is an incorrect specification of byte order. This can be changed by adding the following line in xmtnimage.h before the line `#include <unistd.h>`

```
#undef LITTLE_ENDIAN
```

2.7 Installation on Cygwin Xfree

Cygwin Xfree is a port of XFree86 to Microsoft Windows. If the `gcc` compiler is installed, it is possible to compile and run `tnimage` under Windows. However, some incompatibilities may exist. I experienced problems caused by incompatibilities with the Lesstif that was installed on Cygwin.

1. Make sure Motif is installed.
2. Copy the file `scsi/sg.h` from a Linux system to the same location in Cygwin.
3. Test the `config.guess` script to find out whether it prints "cygwin". If not, replace `config.guess` with the following one line script:
   ```
   echo cygwin
   ```
   and make it executable.
4. Run `./configure` , `make` and `make install` as usual.

2.8 Compiling with Lesstif

Lesstif is a free version of Motif and should not create any particular problems.

If you are unable to persuade your system administrator to install Motif, or you wish to use a different Motif such as Lesstif, the Motif libraries and include files may be installed under the tnimage source directory in directories named "./include" and "./libs", or "./motif/include/Motif-2.0" and "./motif/lib/Xm-2.0/.libs". The latter is primarily useful with systems in which Lesstif is to coexist with another Motif. In this case *only*, edit the file `makefile-lesstif` as necessary and compile `tnimage` with the command `make -f makefile-lesstif` .

If the program does not compile with Lesstif, one possible reason is that Lesstif Motif-2.0 is installed, which is missing some constants and functions found in other more 2.0-compliant Motifs. This can be fixed by editing `config.h` and commenting out the line `#define MOTIF2 1` by changing it to `/* #define MOTIF2 1 */` and recompiling (without re-running `configure`).

Please notify me if you have problems after compiling with Lesstif. Otherwise, it may be a very long time before the problem gets fixed, since I only occasionally test the program with Lesstif.

2.9 If you have problems compiling the program

Here are some tips that have been found helpful in compiling `tnimage` on some systems. For normal Unix systems, these steps should not be necessary.

- In `makefile`, try substituting "CC=g++" for "CC=gcc".
- If a problem occurs with `xmtnimage16.cc`, make sure `libjpeg.a` is somewhere on your system. In one case it was necessary to copy some of the libjpeg include files (`jconfig.h, jdct.h, jerror.h, jinclude.h, jmemsys.h, jmorecfg.h, jpegint.h, jpeglib.h, and jversion.h`) to the `tnimage` source directory or in `/usr/include/gr`.
- Add `/usr/X11R6/LessTif/Motif1.2/lib` to `/etc/ld.so.conf` and type "`/sbin/ldconfig`". (This solved a problem on one machine that could not load `libXm.so.1`).
- Check to make sure the files in /usr/X11R6/include/Xm are the correct version. Don't mix Motif 1.x files with 2.x libraries or vice versa.
- If problems are encountered with `calculator.l` or `calculator.y`, try substituting `bison` and `flex` for `yacc` and `lex` respectively.

- If you have an unusual CPU and the program compiles successfully but cannot read TIFF files, it may be caused by incorrect determination of the byte order. Each compiler seems to have a different way of indicating the byte order. Posix-compliant systems are supposed to have a file called `bytesex.h` or `endian.h`. If these files are not present, `make` may get the byte order wrong. To fix this, add either `#define LITTLE_ENDIAN` or `#undef LITTLE_ENDIAN` after the line `#define MAXWINDOWS 10` in `xmtnimage.h` to force the correct byte order and run 'make clean' and 'make' again.
- *Don't* edit `y.tab.c` or `y.tab.cc`. These are created automatically from `calculator.y`.

More compiling problems and solutions

- **Problem:**
```
In file included from xmtnimage.cc:11:
xmtnimagec.h:9:  Xbae/Matrix.h:  No such file or directory
```
- **Solution:**
  Xbae installation did not occur correctly. Manually copy libXbae.a to /usr/X11R6/lib and copy *.h to /usr/X11R6/include/Xbae.
- **Problem:**`grep:  /Xm/Xm.h:  No such file or directory`
```
./configure:  test:  -ge:  unary operator expected
```
- **Solution:** Motif is not installed correctly; or paths to Motif are incorrect. This occurs on SuSE linux with Lesstif. Should say: `found` after `looking for`
```
/usr/X11R6/include/Xm/Xm.h
...
looking for /usr/openwin/share/include/Xm/Xm.h
```
- **Problem:** `In file included from xmtnimage.cc:10:  xmtnimage.h:79:  bytesex.h: No such file or directory`
- **Solution:**
  This should not occur. Delete config.cache and rerun ./configure.
- **Problem:**
  gcc: installation problem, cannot exec 'cc1plus': No such file or directory
- **Solution:**
  gpp / g++ must be installed; egcs or gcc alone are not sufficient to compile C++ programs.
- **Problem:**
```
/usr/X11R6/lib/libXbae.a(Matrix.o)(.data+0xb7c):
undefined reference to 'XmeFromHorizontalPixels'
```
- **Solution:** Motif is not installed correctly. This message will also occur during compilation of libXbae. Reinstall Motif and recompile and reinstall libXbae.a.
- **Problem:**
```
tnimage:  loadlocale.c:220:  _nl_load_locale:  Assertion 'idx % __alignof__ (u_int32_t
== 0' failed.
```
- **Solution:** `export LC_ALL=POSIX`

2.10 Running DOS version

Before running the DOS version in DOS or Windows for the first time, it is recommended to select a screen mode. If this is not done, `tnimage` will automatically run at 8 bits/pixel and 800x600 resolution.

The procedure is:

1. Open a DOS box (or boot into DOS mode).
2. type `tnimage ?`
   This will print a table of screen modes `tnimage` can handle. All modern video cards for x86 computers can handle VESA modes. Select the desired VESA mode from the table. For example, to start `tnimage` in 1024 x 768 rexolution and 24 bits/pixel, type:
   `tnimage -mode 118`
   or alternatively, you could type
   `tnimage -xres 1024 -bpp 24`

   ```
   Screen modes for various graphics chips
              |                           No. of colors
              |-------------------------------------------------------------------
              |      VESA      Tseng ET4000/W32  Trident 8800/8900 IBM XGA/XGA2
              |-------------------------------------------------------------------
    Resolution|256 32k 64k 16M   256 32k 64k 16M  256 32k 64k 16M    256    64k
   ------------------------------------------------------------------------------
    600 x 400 |100  -   -   -    -   -   -   -    -   -   -   -       -      -
    640 x 480 |101 110 111 112  2E  2E1 2E2 2E3   5D  -   -   -       4      5
    800 x 600 |103 113 114 115  30  301 302 303   5E  -   -   -       7      8
   1024 x 768 |105 116 117 118  38  381 382 383   62  -   -   -       2      -
   1280 x 1024|107 119 11A 11B   -   -   -   -     -  -   -   -       -      -
   1600 x 1200|120  *   *   *

        *  Numbering for screen modes above 11b may vary from card to card.
           All VESA modes with 8 or more bits/pixel are available using the
           -xres and -bpp options.
   ```

   The program will store this value in its .ini file and remember it the next time tnimage is started. See the file `tnimage.doc` for more details.

Currently three SVGA chips are directly supported:

1. Trident 8800, 8900, or more recent chips.
2. Tseng Labs ET4000, W32, or more recent chips. Note: the older Tseng ET3000 chip is not supported. The ET4000 chip is in many cards made by a variety of manufacturers.
3. IBM XGA and XGA-2.

If these chips are not detected, tnimage will use VESA super VGA functions, which are standard on most modern video cards. For earlier video cards, such as the ATI Mach32, it is necessary to run a small program that came with your video card, named "VVESA.COM" (for Mach32 cards), "VESART.COM" (for Realtek cards), or something similar.

Certain really old video cards interacted with the mouse cursor, making smooth movement of the mouse difficult. Some older mouse drivers also cannot handle SVGA modes smoothly. It should be possible to position the mouse at each point on the screen. If you experience difficulty reaching odd-numbered pixels, try to obtain a newer mouse driver. If this doesn't help, it may be necessary to upgrade to a better SVGA card. This problem is known to occur with some RTG3106-based Realtek cards

ATI video card users:

Computers with Mach 32 ATI cards may have problems running tnimage in true-color modes from within a DOS box. The only known solution is to run tnimage from DOS, or use tnimage in an 8 bit/pixel mode (e.g., mode 103).

If your ATI video card won't go into desired screen mode, even though the manual states that mode is supported, run ATI's "Install" program again to verify the specified video mode is activated. If this doesn't work, it is possible that the computer was turned on before turning on the power to the monitor. ATI cards only examine the monitor type at power-up, and could become confused if the monitor is off during a cold boot-up, and refuse to set high-resolution modes.

The display in tnimage is corrupted when using ATI Mach 64 cards if the UNIVBE TSR is present.

2.11 DOS command-line options

`-n` = Skip introductory screen in 8-bit modes.

`-vesa` = forces tnimage to use VESA BIOS even for Trident or Tseng Labs cards.

`-oldvesa` = forces tnimage to use functions for version 1.0 VESA BIOS (slightly slower, but required on some cards).

`-trident` = Forces tnimage to use Trident-specific calls, even if VESA BIOS is present.

`-tseng` = Forces tnimage to use Tseng-specific calls, even if VESA BIOS is present.

`-diamond` = Forces tnimage to use Tseng-specific calls, even if VESA BIOS is present and sets the program to use Diamond Speedstar mode. This option is experimental and will be subsumed under the "Tseng" option after we obtain a Tseng-based Diamond card for testing. Feedback as to whether this option actually works will be appreciated.

`-w32` = Forces tnimage to use Tseng-specific calls, even if VESA BIOS is present and sets the program to use W32 chip mode. This option is experimental and will be subsumed under the "Tseng" option after we obtain a card with the Tseng ET4000W32 chip for testing. Feedback as to whether this option works will be appreciated.

`-S3, -ATI, -XGA` = Force tnimage to control these chips directly, even if VESA BIOS is present. S3 and ATI are not currently implemented. XGA mode currently only works in 640 x 480 mode.

-diag , -diagnose = Diagnostic mode (Prints status messages at each stage of setting your video mode). Useful if you have difficulty starting tnimage on your computer. At each step, you need to press a key to continue. **Please report this information when requesting technical assistance.**

`-fyu` = This option is for debugging use only.

`-macro` *macro_file* = Automatically runs the specified macro file. See "Batch mode processing"
`-files` *file_list* = Specifies a list of image files to be automatically loaded. Using a text editor, create an ordinary text file containing a list of the files to be read, with one filename on each line, for example:

`t*.tif`

`c:\ images\ abc.bmp`

`hubble.img`

Then start tnimage with the command:

`tnimage -files test.lst` substituting the name of your list for "test.lst". Do not put a dash before the filename of the list file. Wildcard characters such as '*' or '?' are also permissible. Up to 512 images can be loaded from the list file (if you have the Registered version). If you only need to load a few files, it may be easier to use the *filename* option (below) instead. filename *filename ...* = Reads the specified file(s). You can load as many files as you can cram onto the command line. If you have a large number of files, it may be easier to use the "-files" option (above). Do not precede the filename with a dash ("-"). Filenames may contain path specifications and wild cards (* or ?).

Examples:

`tnimage my.gif` = Loads my.gif

`tnimage *.tif` = Loads all TIF files in current directory.

`tnimage c:\images\*.tif b:\abc.img` = Loads all TIF files from the C:\images directory, then file abc.img from drive B.

`tnimage -mode 103 *.tif -files file.lst -vesa`

Any combination of command line options is also acceptable. See *Screen modes supported* (below) for a list of screen modes.

`-mode` xxx = Start up in screen mode xxx, where xxx is a hexadecimal number indicating the desired screen mode (see below). Some monitors cannot handle the highest resolution modes and will fail to 'sync' or display a white screen. If this happens, press Alt-X two or three times to escape from the program. You can determine which screen modes your computer can handle by checking in the manual that came with your video card and monitor. **Read the warning below before using this option.**

Example: `tnimage -mode 111`

`-xres` xxx

= Start up in a VESA screen mode with x resolution of xxx, where xxx is the no. of horizontal pixels to display per scan line. 8 bits/pixel is automatically selected unless specified with the -bpp option. Some monitors cannot handle the highest resolution modes and will fail to 'sync' or display a white screen. If this happens, press Alt-X two or three times to escape from the program. You can determine which screen modes your computer can handle by checking in the manual that came with your video card and monitor. **Read the warning below before using this option.**

`-bpp` xxx = Start up in a VESA screen mode of xxx bits/pixel. (Used in conjunction with the -xres option).

`?` *or* `-?` = List the available screen modes.

The options can be in any order, except that if a parameter is needed, it must follow the option. The arguments must be separated by spaces.

Examples: To run tnimage in VESA 1024x768 mode, type:

`tnimage -vesa -mode 105` *Enter* or

`tnimage -mode 105` (The word "Vesa" is only required if you happen to have a Tseng or Trident chip and you want to bypass direct chip addressing).

If you have an old card, specifying "-vesa" will cause display problems. If problems occur, use:

`tnimage -oldvesa` or

`tnimage -oldvesa -mode 105` to select 1024x768 mode.

To run tnimage on a computer equipped with a Trident card, in 640 x 480 mode, type:

tta tnimage -mode 5d

or

`tnimage -trident -mode 5d` To run tnimage on a computer equipped with a card containing a Tseng Labs ET4000 chip, in 800 x 600 mode, type:

`tnimage -mode 30` or `tnimage -tseng -mode 30`

To run tnimage and automatically load and display the image “test.img”, type

`tnimage test.img` or `tnimage -file test.img`

To run tnimage in a 1600 x 1200 resolution at 8 bits/pixel, type:

`tnimage -xres 1600` or `tnimage -xres 1600 -bpp 8`

To run tnimage in a 1600 x 1200 resolution at 24 bits/pixel (true color), type:

`tnimage -xres 1600 -bpp 24`

The command line options are stored in the file tnimage.INI, so the next time you start tnimage, it is not necessary to specify the screen mode.

2.11.1 Notes on DOS version

1. Running Unix version on DOS or Windows systems

See the file `http://entropy.brneurosci.org/unix-emulator.html` for details.

2. The DOS version is no longer shareware, but is freely available. Development of the DOS version was stopped in 1997. Many image analysis features are only available in the Unix version.

Section 3

Basic operations

3.1 Image Analysis Terms

Image depth: The number of bytes per pixel. An image with a higher image depth has a greater number of possible intensity values, which improves the quality of filtering and measurement.

Pixel value: The raw number representing a pixel, as it occurs in an image file. This number can have any number of bits from 1 to 48, corresponding to 2 to 2.8×10^{14} different colors.

Pixel intensity value: The brightness of a pixel. For grayscale images, this is the same as the pixel value. For measurement purposes, the pixel intensity is scaled to $\{0..1\}$ so that the results are independent of the image depth. The actual shade of gray displayed on the screen may be different, depending on whether gamma correction is used.

Gamma correction: A nonlinear correction factor which is supposed to convert an evenly-spaced grayscale palette to evenly-spaced levels of brightness in the monitor (as measured by a photometer). It is usually in the form of a table representing the function

$$output = input^{gamma}$$

where *gamma* is usually 1.7–2.2.

Pixel density: The grayscale intensity of a pixel expressed as a number in the range 0..1.

Luminosity: The grayscale intensity of a color pixel, computed from

$$luminosity = a \times red + b \times green + c \times blue$$

where a, b, c are in the range 0..1.

Image types

Images fall into 3 distinct categories:

1. Grayscale: Pixel values directly correspond to intensity. There may in addition be a table mapping each intensity to an optical density value. Grayscale images are usually 8, 12, or 16 bits/pixel, for 256, 4096, or 65536 gray levels, respectively. Since the monitor can only display 256 gray levels at a time, these gray levels are reduced to 256 in tnimage by a grayscale intensity map which can be adjusted to highlight any specific intensity range.

   "False-color" images are a subset of grayscale images in which the computer displays an arbitrary color instead of a shade of gray.

2. Indexed-color: Pixel values have no intrinsic meaning and are merely indices into a color map containing 256 colors in a random sequence which is totally different for each image. Indexed-color images are almost always 8 bits/pixel.

3. Color: Pixel values are a composite number containing intensities of red, green, and blue. The number of bits of each color is different for each standard image depth (*i.e.,* 15, 16, 24, 32, and 48 bits/pixel).

Each image type has benefits and drawbacks. For example, cutting and pasting among different indexed-color images requires finding the closest matching colors. If the color maps are too different, the pasted image will not have the same color as the original. This problem can be avoided by converting the images to true color first, then converting back to the original image type afterwards. Similarly, densitometry, wavelets, and FFT only work well on grayscale images. Before performing FFTs, it is necessary to convert color images to grayscale. For indexed-color images, the additional step of sorting the colormap is necessary to ensure that pixel values bear a linear correspondence to intensity. Converting grayscale images to color involves defining or selecting a colormap and converting the image to indexed color, and then converting it to true color. All of these operations are done automatically in tnimage.

When the mouse is moved in tnimage, the information area shows the x and y coordinates and the value of the pixel at the mouse position. This is convenient when manipulating the image, since much image manipulation uses simple mathematical functions to transform the image. Thus, to make the image lighter, one could "add" a value of 50 to each pixel. To increase contrast, one can multiply each pixel value by a constant factor. In contrast to some programs on some other operating systems, these steps are not hidden from the user merely for the sake of ease of use. This ensures that quantitative manipulations on the image remain directly traceable to the original data.

3.2 Menu Bars

In the Unix X-Window version, the menu bar automatically wraps to multiple lines if there is not enough space. The program window can be resized so that only the main information window and the menu bar are visible. Images can then be loaded into separate, positionable windows. On occasion, it is necessary to quit and restart tnimage after resizing the window to obtain a multi-line menu bar.

The Motif menus can be "torn off" and dragged to one side by clicking on the dashed line at the top of the menu. Thus, it is possible, for example, to have one or more of the menus on the screen permanently and then shrink the main window to an inconspicuous size. These torn-off menus act as permanent pushbuttons, saving tedious repetitive mouse clicking.

Clicking on one of the four large arrows in the information area moves the current image in the direction of the arrow. The distance moved is determined by the "step value" which can be changed by the gray + and − keys (KP_add and KP_subtract) or in the "Config" dialog.

The main cancel button is always active and interrupts many repetitive operations, such as flood fill, densitometry, circle-drawing mode, etc.

The main paste button copies all opaque pixels in the currently-selected region onto whatever is beneath it.

Tutorial files Five tutorial files are included with Tnimage:

1. DENSITOM.PCX - an image file demonstrating strip densitometry.

2. DENSITO2.TIF - an image file demonstrating spot densitometry.

3. DECONVOL.TIF - an image file demonstrating convolution and deconvolution using FFT (Part 1).

4. DECONVO2.TIF - an image file demonstrating convolution and deconvolution using FFT (Part 2).
5. FILTER.TIF - an image file demonstrating digital filtering using FFT.
6. Also see Section 13.3, "Macro Programming Guide" in this manual.

3.3 Unix Command-line options

`-macro` *macro_file* = Automatically runs the specified macro file. See *Batch mode processing*

`-diag` = Prints diagnostic information. This information should be included in any bug reports.

`-nosparse` = Some X servers report the incorrect result for the X Window BitmapUnits() call. This causes images to appear as multiple horizontal copies of itself when the X server is in 24 bit/pixel mode. This option forces the correct value.

placing the line

`nosparse 1`

in the `tnimage.ini` file causes this setting to take effect automatically.

`-files` *file_list* Specifies a list of image files to be automatically opened. Using a text editor, create an ordinary text file containing a list of the files to be read, with one filename on each line, for example: `t*.tif`

`~images/abc.bmp`

`hubble.img`

Then start tnimage with the command:

`tnimage -files my.list`

substituting the name of the list file. Wildcard characters such as '*' or '?' are also permissible. Up to 512 images can be opened from the list file. If only a few files are needed, it may be easier to use the *filename* option (below) instead.

filename filename ... = Reads the specified file(s). If there are a large number of files, it may be easier to use the "-files" option (above). Filenames may contain path specifications and wild cards (* or ?).

Examples: `tnimage my.gif` Loads my.gif

`tnimage *.tif` Loads all TIFF files in current directory.

`tnimage ~images/*.tif /mnt/abc.img` = Loads all TIFF files from the /images directory, then file abc.img from /mnt.

Any combination of command line options is also acceptable.

`?` , `-?` List the available options.

`-fg` or `-foreground` Specify foreground color

`-bg` or `-background` Specify background color

Example: `tnimage -fg white -bg MistyRose3 &`

`-geometry` *geometry-string*

Example: `tnimage -geometry 400x500+100+100`

-display *screen-IP-address:screen*

Example: `tnimage -display 192.168.1.1:0.0 &`

See ' man X ' for more details. The options can be in any order, except that if a parameter is needed, it must follow the option. The arguments must be separated by spaces.

To run tnimage and automatically load and display the image "test.img", type

`tnimage test.img`

To run tnimage remotely on a supercomputer and have the display appear on your local PC, type the command:

`xhost +`*supercomputer_host_name*

Then login to the other computer, and type:

`tnimage -display` *your_PC_hostname*`:0.0 &` where *your_PC_hostname* can be a registered host.domain name or an IP number. For example, if you are connected over a ppp link, and `ifconfig` indicates your IP address to be 156.40.65.151, type:

`tnimage -display 156.40.65.151:0.0 &.`

This can also be done from DOS or Windows using X-Window server software such as Xappeal, Exceed, Chameleon NFS/X, etc.

Alternatively, this can be done by setting an environment variable, *e.g.,*

`setenv DISPLAY 156.40.65.151:0.0`

(in tcsh or csh) or

`export DISPLAY=156.40.65.151:0.0`

(in bash, ksh, etc.) instead of using the "-display" option.

3.4 Image windows

Images loaded into the X Window version of tnimage can be placed in separate windows instead of the large main window by checking the "Separate window" box in the "Load image" dialog. With a few minor exceptions, images in separate windows have the same functionality (cut/paste, image math, densitometry, etc) as images on the main window. Cut/paste functions identically regardless of whether the source and destination are on the main window or on separate windows. Additionally, if the Window Manager permits, image windows can be created with or without a frame, depending on the settings in the "Config" dialog.

The advantages of separate windows are:

1. Entire image is visible without scrolling
2. Image can be iconified when not needed
3. Scroll bars can be added (using the menu in the upper left corner. This feature is only available in window managers that support it, such as fvwm.)
4. Images can be positioned anywhere on the screen

Disadvantages are:

1. Images cannot be superimposed, added, or subtracted without using "image math" function.
2. Transparency and chromakey do not work.
3. Composite images cannot be created.

3.5 Basic image manipulation

Most operations, such as filtering, saving images to disk, etc. use a "selected region" which has to be selected before the operation is performed To select a region, move the mouse to one corner of the desired area, click and drag to the other corner, then release the mouse button. A rectangle shows the selected region, which stays in effect until (1) the image is moved, (2) another region is selected, or (3) the mouse is clicked on an image to select an entire image. Selecting an image means that all subsequent operations will be performed on the entirety of that image. If no region or image has been selected, the region defaults to the entire screen. If changes are made a region that is partly on an image and partly on the background, the portion of the changes that falls on the image will remain with the image, and the rest will stick to the background.

The background is maintained independently of the images. Thus, if an image is moved over the background or another image, the background is not disturbed. When many images are simultaneously present, it is often convenient to keep most of them out of the way by stacking them up or moving them out of the visible region, off the edge of the screen. Alternatively, unused images can be changed into icons. If an image gets lost off the edge, it can be located by selecting "Select image" or "About the image".

Images can be moved around by clicking on the small arrows in the information area. The distance by which a single click on the arrow moves the image is adjustable by pressing the gray "+" or "–" keys, or by selecting "Cursor movement rate" from the "Configuration" menu.

When the mouse cursor passes over an edge of an image, it changes from a diagonal arrow to double arrows. Clicking the left mouse button at this point, will "grab" the image. Continue pressing the mouse button, move to a new location, and then release. The image is then moved to the new location. The same is true for message boxes, graphs, dialog boxes, the colormap window, and information windows.

Edge-grabbing of images is prevented if the Left Alt, Caps Lock, or Shift keys are depressed. This makes it easier to select areas near the edge of the image.

If the Shift key is depressed, clicking the mouse sets a point on the image. Shift-clicking on a different point on the same image causes the area between the two points to be selected. Since shift-clicking is unaffected by the position of the image with respect to the screen, this feature makes it possible to select regions that are larger than the size of the screen.

3.5.1 Selecting rectangular areas

A rectangular region can be selected by clicking and dragging with the left mouse button.

Dragging on the middle of one of the four borders of the selected area (when the cursor is showing an "edge" symbol) resizes the selected area. Dragging on the corner (a "fleur" symbol) grabs the selected area and copies it to another location.

The size of the selected area can also be changed in one-pixel increments with Shift+Numeric keypad keys. The mouse cursor must be somewhere in the image area for the keystrokes to be received. This permits precise positioning of the selected area.

Shift + 0 Makes selected area smaller.

Shift + 1 Moves top edge down.

Shift + 2 Moves bottom edge down.

Shift + 3 Moves left edge right.

Shift + 4 Moves left edge left.

Shift + 5 Makes selected area bigger.

Shift + 6 Moves right edge right.

Shift + 7 Moves right edge left.

Shift + 8 Moves top edge up.

Shift + 9 Moves bottom edge up.

A rectangular region can also be selected by selecting two image points. Press the Shift key and left-click to select the first point. Release the mouse and shift-click again to select a second point. The difference between this method and the click-and-drag method is that the points are fixed on the image instead of the screen. This means you can move the image around with the mouse or arrow keys before selecting the second point. This can be useful in selecting areas larger than the screen, which would be impossible to select with click-and-drag.

3.5.2 Automatically selecting objects

If the mouse is double-clicked, `tnimage` tries to identify an object in the image. Since images are not in themselves composite objects, object identification is done by finding a group of contiguous pixels within a certain color range of the point at which the mouse was clicked. For example, if the mouse is double-clicked on a white area, `tnimage` will look for and select a white object centered at that point. If "crawling dots" are not displayed around the edges of the object after clicking on it, it means no object could be identified.

Once an object is selected, the selected portion can be copied, contrast-enhanced, etc. without affecting the surrounding pixels. For example, the white matter in the brain image below (from the Visible Human project) was changed from yellow to blue by simply double-clicking on the yellow area, and then inverting the colors of the selected area.

Changing color in non-rectangular portions of an image.

The "Object threshold" value (in the Config..Configure menu) determines how much surrounding area is included. Any pixels in the object whose red, green, and blue values differ from those at the starting point (*i.e.,* the point on which you double-clicked) by less than the threshold are included. A lower value means the selection is more stringent. For example, increasing the red threshold and lowering the green and blue threshold will make it easier to select an object which is predominantly red.

3.5.3 Using "scissors" to manually select objects or groups of objects

Any object or combination of objects can also be selected manually. Click on the desired image and select "Image..Manually select area", or press F3. Click OK on the message box, and outline the desired objects as described below.

As with any other selected region, clicking anywhere outside the region, or reconfiguring or moving the image window will deselect the area. The selected area(s) can be grabbed and copied or moved (depending on the position of the main "Move" button) as a group by positioning the mouse over the crawling dots and clicking the 1st mouse button.

The method of selecting the outline can be changed in the 'Config..' dialog. The options are:

Single (freehand) - Only one object can be selected. Click the left mouse button and draw an outline around the object. When you unclick the mouse button, a message box appears instructing you to click somewhere in the object to define its inside. After you click, the inside will turn blue for a moment, and then its outline will be replaced by crawling lines, indicating the outline of the selected area.

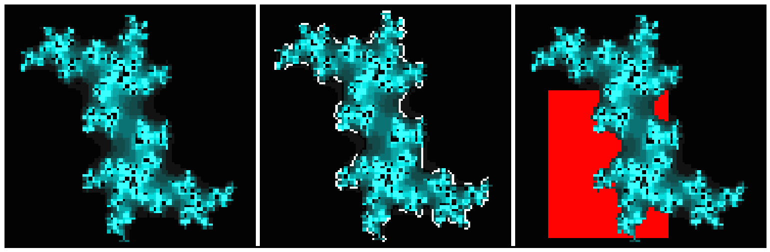
Multiple (freehand) - An arbitrary number of disjoint areas can be selected. Draw a freehand outline around the object with the mouse as before. When you unclick the mouse button, a message box appears instructing you to click somewhere in the object to define its inside. After you click, the inside will turn blue. Continue selecting new objects in the same way. When finished selecting objects, click the main Cancel button. The blue will disappear and be replaced by crawling lines around each of the areas. Even though they are disjoint areas, they will be acted upon as a unit. For example, if you click the mouse on the edge of one object, all the objects will be copied at once.

Polygon - Click with the left mouse button at various points around the outside of the object to define a polygon around the boundary of the object. After each click, a small box indicating the control point will appear. These boxes can be repositioned at any time by dragging them to a new location. When finished, press the space bar (or any other key). A message box will appear instructing you to click somewhere on the object to define the inside. **Note:** a minimum of 3 control points is necessary to define the polygon.

Point-to-point - Click with the left mouse button at various points. Press space bar or click right mouse button when finished. A polygon is drawn in a manner similar to the drawing mode in `xfig`.

Once a non-rectangular area has been selected, it can be reactivated at any time with the 'reselectarea' menu option, without the necessity of re-tracing the previous outline.

One limitation not shared with rectangular selected regions is that the entire irregularly-shaped region must lie within a single image or on the background.



Interaction of painting with non-rectangular selections.

Like rectangular selections, non-rectangular selections also interact with painting and gradient fills. In the above example, the black area around the fractal was selected by double-clicking on the black region. When "Draw..paint region" was used to change a small rectangular portion of it to red, only the selected region was affected. Normally, if no non-rectangular region had been selected, the entire square would be filled with red. If the fractal itself had been selected (by double clicking on the fractal, or by clicking on the black and selecting "Switch selected/unselected"), only the fractal part would be painted red.

Here are some tips on selecting objects in an image:

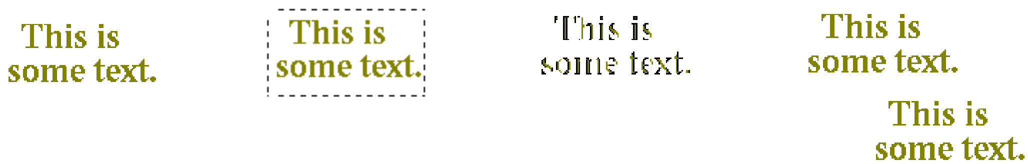
- If the object is near the edge of the image, it may be helpful to put a border around the image.
- If the correct area is not completely identified, try double-clicking on a slightly different point.
- If the object is ill-defined or touches other objects of a similar color, it may be necessary to draw an outline around the object to separate it. This can be done by the "spray math" function which can lighten or darken small regions, or by drawing on the image with the "sketch" function.
- The sensitivity for small variations in color can be adjusted in the "Config..Configure..-Object threshold" menu. This value is the percent difference in each color required to separate objects. A higher value results in larger areas being selected. If difficulty is experienced selecting the correct area, try adjusting the Object Threshold value to match the color of the desired area.
- On slower computers, object identification may take several seconds if the image is large. On really slow computers, if the system load becomes excessive after a complex area is selected, it can be reduced by increasing the SLOW_DELAY constant in `xmtnimage.cc` and recompiling.

3.5.4 Selecting and copying text

Although the above procedure works well for selecting objects in an image or for selecting individual letters, it is often desirable to copy or reposition an entire label. This is easily done by the following procedure:

1. Select a rectangular region around the text with the mouse.
2. Move the mouse pointer over the text, inside the rectangle indicated by "crawling dots".
3. Double-click on the text. The crawling-box rectangle will be replaced by crawling-dots outlining the portion of the text that was inside the box. It is important to click directly on the text, and not on the background near the text; otherwise, everything *except* the text will be selected, creating a stencil.
4. Move the mouse pointer over the crawling-dots. The cursor will change to double arrows.

5. Click and drag the text to its new location. When the mouse button is released, only the selected pixels will be copied to the new location.
6. If the “move” button in the information window is depressed, the text will be moved instead of copied.



Copying a region of text.

Notes:

1. If the mouse is clicked on a point containing a label or graphic element, the alpha channel is used to select the object. Otherwise, the object is selected on the basis of its RGB distance from the threshold value. This allows anti-aliased text and multi-colored graphic elements to be selected.

2. If a rectangular area has been selected (by dragging the mouse), the region selected by a double-click will be confined within that rectangular area. If no rectangular region is selected, the object can be as large as the display size or the image (whichever is smaller).

To summarize:

- To select an entire image, single-click on the image.
- To reposition an image, click and drag on the corner of the image.
- To select a rectangular area, click and drag on the area to create an ‘outline’ of the desired area.
- To select a non-rectangular object (such as an element of text), double-click on the object.
- To manually select a non-rectangular area, select “Image..manually select area” or press F3.
- To confine the automatic area selection to a specific region, select the boundary rectangular region by clicking and dragging with the mouse first.
- To copy a non-rectanglular selected area to a new location, click on the crawling-dot outline along the edge of the selection and drag it to the desired location (this can be in a different image or even a different window).
- To copy a rectanglular selected area to a new location, click on onw of the four corners of the crawling-dot outline of the selection and drag it to the desired location.
- To resize the area selection, click and drag near the middle of one of the ‘crawling-dot’ line segments, or use the Shift+Numeric Keypad keys for fine control.

3.6 Operating under low-memory conditions

There are several ways to increase available memory:

1. In the “Config” menu, turn off the “Automatic undo” option before loading any images. This will double the amount of space available for images, but tnimage will not create an “undo” buffer for each image and it will not be possible to undo any changes to an image.

2. Start tnimage (or the X server) in a lower screen mode. In lower resolution modes, less memory is needed to manipulate images and background. (See under "Command-line options"). Try using 8-bit/pixel modes instead of true color modes. Tnimage reserves a fixed amount of memory for drawing dialog boxes and error messages. The amount of reserved memory is also screen mode-dependent.
3. In the DOS version, memory is limited only by the available hard disk space. More virtual memory can be created by using a disk utility to unfragment your hard disk.
4. If high resolution is not essential, images can be loaded in a small size. Loading an image with "X size=50" and "Y size=50" uses only 1/4 as much memory as a full-size image. (See under "File menu...Open image").
5. Erase FFTs as soon as possible by selecting "File...Erase FFT". Fourier transforms use large amounts of memory.
6. In Linux and other Unix-like OSs, memory is often limited for individual users in `/etc/security/limits.conf` or in `/etc/profile`. These values can be increased on a per-user basis by the system administrator.
7. In Windows, the program will not have enough memory, if there is less than about 2 MB of disk space available. In this case, the program will not start, and Windows will pop up an "Insufficient Memory" error message or may crash. Try closing some applications and/or windows on the screen and try again, or reboot a couple of times.

3.7 Frequently asked questions

Q. How do I add text?

A. Move the mouse cursor to the desired location and begin typing. It is not necessary to click the mouse before typing. Make sure the mouse does not move while typing. The text will stick to the foremost image, or if no image is present, to the background. Unix versions can also select the font, font size, and weight. Alternatively, text can be positioned interactively by selecting "Draw..Label", or by typing on the background and copying it to the image using the "copy" function.

In the Unix version, you can also add labels by selecting some text in a text window with the mouse. Click the middle mouse button to paste it onto an image in the desired location.

Larger amounts of text can be added by opening the text file in the same manner as opening an image. The text file must have the extension ".txt".

Q. Sometimes the Dismiss button doesn't work, or the Cancel button must be clicked twice.

A. Dismiss buttons on a dialog box are temporarily inactivated if a 'child' dialog box or message box is being displayed. Make sure no message boxes or click boxes are visible. In some cases, the message box may become obscured by some other window.

In some situations, for example, when the program is waiting for the user to draw a line, the Cancel button on a dialog box must be clicked twice before the dialog box is dismissed. This is caused by a peculiarity in Motif.

Q. After converting an image to color, any text or graphics added to the image still appears in shades of gray instead of color; or colors are completely wrong.

A. For 8-bit images, only colors on the current colormap associated with the image can be added. If the image previously was grayscale, the colormap will only contain shades of gray. To add different colors, convert the image to 24 bits per pixel and add the colors. When you convert back to 8, a new optimal colormap will be calculated.

Q. How do I select an irregularly-shaped area (i.e., "scissors" command)?

A. Press F3, then click the Cancel button when finished.

Q. When I changed the image depth of a scanned image from 10- or 12-bits to 8 bits/pixel, it turned black.

A. For some images, it is necessary to maximize contrast before changing to a different pixel depth. For example, a 10 bit image has pixel values between 0 to 1024, but it is stored in a 16 bit buffer. The contrast must be expanded to 16 bits, otherwise it will appear black when scaled to 8 bits. This is supposed to be done automatically. Please report the problem.

Q. Why use Motif and not KDE or Gnome?

A. KDE and Gnome are Linux-specific and mutually incompatible (*i.e.,* many distributions only install libraries for one or the other). It is not anticipated that `tnimage` will be ported to either Gnome or KDE until some level of standardization occurs.

Q. It is a pain selecting the menus all the time.

A. All the menus are tear-off menus. Open the menu and click on the dashed line at the top. The menu then tears off and becomes a column of pushbuttons. Alternatively, use the command editor. Or, you can define your own pushbuttons that will automatically be added to the buttons at the left of the main window that will execute any arbitrary command.

Q. How do I undo an operation?

A. Select "Process...Restore". The image will be restored to its state at the time it was last backed up. If the image was never explicitly backed up, it will be restored to its original state – unless the "Auto-Undo" option has been turned off.

Q. How do I turn off box- or line-drawing mode?

A. Click anywhere on the top menu bar or pull down any menu. This automatically turns off everything. Alternatively, click on the small rectangle which displays the current mode on the right side of the menu bar. In the Unix version, click on the main "Cancel" button in the information area.

Q. Is there an easy way to change the color of text, lines, etc.?

A. Yes, click the left mouse button on the color palette to select the foreground color. Click the right mouse button on the palette to select the background color. This will not change the color of any existing text or lines, only new ones.

Q. How do I make the background color white?

A. Select "Color..Set background color" and set the color to the highest value (255 on an 8-bit display). Then select "Image.. erase background" to redraw the background in the new color (**Note:** This will erase everything that was drawn on the background).

Q. When filtering, changing contrast, etc. of color images, why does junk or blackness sometimes briefly appear on the screen?

A. If tnimage is in an 8-bit screen mode, manipulating 16 and 24-bit images temporarily causes the image to be filled with 16 or 24 bit data, which appears as random noise because of the discontinuous colormap. After the operation is finished, the colormap is automatically recalculated, and the image will appear normal again.

Q. How can I combine two or more indexed color images into one image so that they have the same colormap?

A. One way is to start tnimage in a color mode by typing: `tnimage -xres 800 -bpp 24` (if using DOS version). If using the Unix version, it is necessary to restart the X server in a true-color mode by typing: `startx - --bpp 24` or some similar command. Load both images into the same window, then select "Create image" and select a region covering both images, using the mouse. Select "Change image depth" to convert the image to 1 byte (8 bits) per pixel. A new 8-bit colormap will be generated to fit both images.

Alternatively, use the "Colormap...Remap to other colormap" option to map one image onto another's colormap.

Q. Why is xxx image file format not supported?

A. Some image formats (such as GEM IMG) are not commonly used in image analysis. Others (such as lossless JPEG and LZW-compressed TIFF) are not supported because of patent restrictions. However, LZW-compressed TIFF files are readable.

Q. Is a MVS / VMS / Windows-3.1 / Windows-95 (or 97, 98, 0, 2000, etc.) / CE / CP/M / Macintosh/NeXt/Rhapsody / BeOS / Plan9 version of tnimage being planned?

A. There are no current plans.

Q. Is a SunOs 4.x/OpenWindows version of tnimage available?

A. The program requires Motif and does not compile under OpenWindows. If you have Motif somewhere on your disk, the source code version should compile with no problems.

Q. Will lower-resolution screen modes (such as 640 x 480 x 16 colors) ever be supported?

A. No. Reducing the number of colors to 16 would render any quantitative analysis of the image almost meaningless. Users are advised against using 4-bit images for any scientific work, as many such images that have previously appeared in the literature are now being questioned.

Q. How do I scan an 8-bit color image? In the dialog box, all the selections for color images are at least 24 bits/pixel.

A. HP scanners cannot create 8-bit indexed color images. Scan the image at 24 bits and then select "Color.. Change image depth" to change the image to 1 byte (8 bits)/pixel. This will automatically generate an optimal colormap.

Q. Why do controls for red, green, and blue appear when I try to change the contrast on my grayscale image?

A. Some other software incorrectly sets the image identifiers in their image to values which indicate the image is "color" even though it appears as a gray scale. Selecting "Color..Convert to grayscale" will repair the image.

Q. When I set "chromakey" on an image it either turns black or disappears entirely, and cannot be selected by clicking on it with the mouse. Is this a bug?

A. No, if the minimum and maximum chromakey values are too broad, the entire image is transparent. If it becomes invisible, clicking on it will only select the background. Select "Image..image properties" and change the minimum and maximum to more reasonable values.

Q. In densitometry, all the areas are coming out as negative numbers. What is wrong?

A. Either the "background value" or the "maximum signal" is set incorrectly.

Q. Why do you use your own TIFF library? Why not just use libtiff?

A. Although libtiff is great for some types of images, it is not ideal for scientific TIFF files. For instance, with ImageQuant files, you may encounter the following error message:

```
Warning, invalid TIFF directory; tags are not sorted in ascending order.
iq4-05.gel: Warning, TIFF directory is missing required
    "StripByteCounts" field, calculating from imagelength.
iq4-05.gel: Can not handle 16-bit/sample image
```

See sec. 5.9 for more about libtiff.

3.8 Miscellaneous notes

Targa image format Targa (TGA) files must have the extension `.TGA` to be readable.

ASCII files and Text files must have the extension `.ascii` and `.txt`, respectively to be automatically identified. Otherwise, they must be read as "raw ASCII".

LZW compressed TIFF format files As of Version 2.11, creation of compressed TIFF image files is not supported in tnimage. LZW- and PackBits-compressed TIFF files are, however, readable. The reason for this discrepancy is the choice by Aldus Corporation of LZW compression, which is patented by Unisys, for the compression algorithm. The most recent TIFF documentation indicates that LZW is now being phased out of TIFF. This means that LZW-compressed TIFF files will soon no longer be a valid image format.

Section 4

Customizing tnimage

4.1 X Window Resources

Each of the Motif widgets in tnimage has resources that can be individually set, by specifying the widget's name in your .Xdefaults or Tnimage file (in /var/X11R6/lib/app-defaults/). These files also interact with the X Window keymapping which can be changed with `xmodmap`.

As an example, suppose for some bizarre reason it was desired to make the 'Del' key actually delete the character under the cursor. One way of doing this is as follows:

1. Create a keymap file: `xmodmap -pke > keymaps`

2. Edit the line for keycode 91 to read: `keycode 91 = 0x004` . (The keycode can be obtained using `xev`).

3. Load the new key translation table: `xmodmap keymaps` . Pressing the Del key now creates an 0x004. ('Delete' or some valid key name could also be used — these are listed in the file /usr/X11/include/X11/keysym.h).

4. Alternatively, edit .Xdefaults in your home directory and add the line:

```
tnimage*Editor*Translations:  #override \n \
             <Key> 0x4:  delete-next-character() \n
```

(Note this must be entered exactly; in particular, the number must be 0x004 in the `keymap` file but 0x4 in the `.Xdefaults` file.

5. The change will now take effect in the Macro Editor only. On some systems, it may be necessary to restart the window manager for the new key to take effect. Leaving out the "*Editor" portion will cause the change to take effect in all Widgets.

Some of the important widgets in tnimage are:

`Editor` - macro editor (Text)

`MenuBar` - the top menu bar (MenuBar)

`f101 to f199` - the menu items for first menu ("File") (PushButton)

`f201 to f299` - the menu items for second menu ("Image") (PushButton)

`etc`

`drawing_area` - the main drawing window (DrawingArea)

`drawing_area2` - the status area at left (DrawingArea)

`Image` - all the images (DrawingArea)

`List` - list selection boxes (ScrolledList)

Each of these has an extensive set of resources (such as fonts, colors, etc) that can be modified. For instance, Text Widgets have delete-next-word(), cut-clipboard(), page-left(). Refer to the X Window system manuals for details.

Here are additional examples. More specific settings will override more general ones.

`tnimage*drawing_area2*background:green` -Sets the left drawing area to "green".

`tnimage*drawing_area2*background:#2233AA` -Sets the left drawing area to red=22, green = 33, and blue=AA hex (34, 51, and 170 decimal, respectively).

`tnimage*drawing_area*background:#223344`

`tnimage*MenuBar*background:#223344`

To set the main drawing window color, select "colors...set colors", then select "erase background".

`tnimage*Open*fontList: -adobe-courier-medium-r-normal--0-0-0-0-m-0-iso8859-1` Sets the font for the Open button on the information window at left. This can improve the appearance if the default font on your system is too big.

`tnimage*XmToggleButton*fontList: *-courier-medium-r-normal--0-0-0-0*` Sets the font for all toggle buttons

`tnimage*marginHeight: 0` Sets the height of the margin in all widgets

`tnimage*marginWidth: 0` Sets the height of the margin in all widgets

`tnimage*foreground: black` Sets the widget text color

`tnimage*background: gray` Sets the background color (Foreground and background colors of the main drawing area are set separately, from within tnimage).

`tnimage*shadowThickness: 1` Sets the depth of shadows on all buttons

`tnimage*MenuBar*spacing: 3` Sets the separation between items on menus

`tnimage*MenuBar*background: gray` Sets the color of menu bar

`tnimage*MenuBar*fontList: *helvetica-bold-r-normal--12*`
Sets the font for the top menu bar

`tnimage*fontName: *times-bold-r-normal--12*` Sets the default font

`tnimage*information_area.foreground:black` Sets the text color of the information area at the left

`tnimage*information_area.background:gray` Sets the background color of the information area at the left

`tnimage*list*fontList:*times-bold-r-normal--12*` Sets the font in list boxes to times bold (Not recommended, a fixed width font should be used).

These resources control the appearance of the file selection dialog:

`tnimage*FileSelector*background: gray`

`tnimage*FileSelector*foreground: black`

`tnimage*FileSelector*fontList:*helvetica-bold-r-normal--12*`

`tnimage*FileSelector*marginWidth: 5`

`tnimage*FileSelector*marginHeight: 5`

`tnimage*FileSelector*listVisibleItemCount: 30` Sets the number of files to show at a time

`tnimage*FileSelector*borderWidth 0`

Other miscellaneous examples:

`tnimage*Editor*foreground: black` Color of text in text editor

`tnimage*Editor*background: gray` Color of text editor

`tnimage*Editor*fontList:*helvetica-bold-r-normal--12*` Font for text in editor

`tnimage*Warning*background: red` Set warning messages to red

`tnimage*OK*background: red`

`tnimage*Cancel*background: white`

`tnimage*Help*background: blue` Set the OK, Cancel, Help buttons to red, white, and blue, respectively

`tnimage*f101*background: red` set the 1st item in the first menu to red.

`tnimage*borderWidth: 2000000000` Add a border 2 billion pixels wide around each image (Not recommended).

`tnimage*borderColor: papaya whip` Change image border to papaya whip.

The labels and fonts for the dialog boxes may also be individually set. In this case, the resource name is the default string for the label. This also permits non-English labels to be substituted. For example:

`tnimage*DialogForm*fontList: *helvetica-bold-r-normal--12*`

`tnimage*DialogForm*radiobox*fontList: fixed`

`tnimage*DialogForm*boxbutton*fontList: fixed`

`tnimage*DialogForm*boxtext*fontList: fixed`

`tnimage*DialogForm*boxpushbutton*fontList: fixed`

`tnimage*DialogForm*boxfilename*fontList: fixed`

`tnimage*fontList: hanzigb16st` This sets all fonts to Chinese. For fonts encoded by multi-byte strings (such as Chinese fonts), the label string and your locale must also be changed.

`tnimage*Interaction mode.labelString: Information at your fingertips`

`tnimage*DialogForm*File type.labelString: This is a feature`

`tnimage*DialogForm*Read Image.labelString:`
`Where do you want to go today`

These three resources change the text in the menu from “Interaction mode”, “ File type”, or “Read image” to the corresponding commonly-used computer terms. The latter two text labels are changed only if they occur in a DialogForm Widget.

`tnimage*DialogForm*Filename.labelString:`

(An example of a foreign-language font; may not display correctly on all systems)

The time interval (in msec) between double-clicks may also be changed. For example:

`tnimage*XmFileSelectionBox*doubleClickInterval: 1000` sets it to 1 sec. Unfortunately, the requirement for double-clicking itself is a limitation of Motif and cannot be easily removed.

Capitalization and spacing of the original label must be specified exactly. More specific settings override less specific ones. For example, the combination

`tnimage*OK*fontList: *helvetica-bold-r-normal--12*`

`tnimage*DialogForm*OK*fontList: *helvetica-bold-r-normal--24*`

will set a size of 24 for the OK button in dialog boxes and 12 elsewhere.

If the specific resource name is unknown, it may still be set by specifying the general Motif class name beginning with Xm. This is true for any Motif program. For example:

`tnimage*XmMessageBox*marginHeight: 5`

Sets the vertical margin around warning and information boxes

`tnimage*XmMessageBox*marginWidth: 5`

Sets the horizontal margin around warning and information boxes

`tnimage*XmMessageBox*foreground: black`

Sets the text color in warning and information boxes

`tnimage*XmMessageBox*background: gray`

Sets the color of warning and information boxes

`tnimage*XmMessageBox*fontList: *helvetica-bold-r-normal--12*`
Sets the font to use in warning and information boxes

`tnimage*XmDialogShell*background: gray`
Sets the color of dialog boxes, clickboxes, etc.

`tnimage*XmDialogShell*foreground: black` Sets the the text color of dialog boxes, clickboxes, etc.

`tnimage*XmArrowButton.background: gray` Sets the color of the four arrow buttons in the information area.

Note: if you rename tnimage, the .Xdefaults file should contain the actual name specified on the command line.

A number of other generic resources may also be set by specifying the Motif widget class (such as XmPushButton, XmList, etc.). In general, any resources (of which there are many) not explicitly hard-coded in the program can be set two different ways: by specifying the Xm class (such as "XmFileSelectionBox"), or by specifying the name used in the program (such as"FileSelector"). The latter method permits specifying properties for each widget instead of all instantiations of the widget type. So in this spirit, below are listed the names of most of the main widgets used in the program.

Message boxes: Information Prompt Error Question YesNoQuestion Warning

Top menu bar: MenuBar

Arrow buttons at left: Button

OK buttons at bottom of dialog boxes: Slew OK Help Open Save Cancel No

Components of dialog boxes: radiobox label DialogForm

The main windows: Main drawing_area_form drawing_area2 frame information_area drawing_area

Images: Image

List boxes: List list drag_area GraphForm graph

Click box components: FileSelector Chdir Editor ClickboxForm MultiClickboxForm PrintItemForm EditForm Menu3DForm SamplePaletteColorForm SamplePalettePseudoForm

Menu items: f101 to f999.

Menus: fileMenu imageMenu processMenu colorMenu drawMenu aboutMenu configMenu helpMenu.

3D controls: Menu3DForm

More examples:

`tnimage*fileMenu.background:  LavenderBlush1` Sets the color of the file menu to "Lavender Blush 1" (See the file /etc/X11/rgb.txt for a list of valid color names, or use a 6-digit hex number such as "#aa7733").

`tnimage*fileMenu*background:  bisque` Sets the color of the file menu, including the menu items, to "bisque".

`tnimage*DialogForm.fontList:*helvetica-bold-r-normal--12*`

`tnimage*DialogForm*XmToggleButton*fontList:fixed`

`tnimage*DialogForm*XmPushButton*fontList:*times-bold-r-normal--12*`

These 3 lines set the toggle buttons to "fixed" font, the push buttons to a times font, and everything else (i.e., the labels) to helvetica. Use 'xlsfonts' to find what fonts are available on your system.

`tnimage*List.XmLabel.fontList:  *helvetica-bold-r-normal--12*`
sets the labels in all lists to helvetica font, while

`tnimage*List*fontList:  *helvetica-bold-r-normal--12*`
sets all text in all lists to helvetica font.

Notes

1. Be conservative about setting all these widgets to different colors when using an 8-bit display. Each color used by a widget is one less color available for rendering images. In particular, the top menubar and individual menu items should be set to the same color, otherwise if tnimage starts up in "modifiable colormaps" mode, the menus may become unreadable. Modifiable colormaps mode is set automatically if there are fewer than 32 color entries available.

2. Similarly, when running tnimage on an 8-bit display, it is advisable to close programs such as Netscape that allocate colors. Some window managers, such as Solaris' CDE, allocate most of the colors automatically, which can result in poor image quality. In Solaris, it is strongly recommended to set tnimage to use "installed colormaps" (see Sec. 11.4).

3. Be careful not to put spaces after the setting in your .Xdefaults file or after font names in your tnimage.ini file. This will confuse Motif.

4. Some resources, such as "transient", although legal, produce instability in Motif, and must not be put in .Xdefaults.

4.2 Non-Motif Resources

In addition to the Motif resources, a variety of Xlib and program resources are stored in the file `$HOME/.tnimage/tnimage.ini` . Most of these are set from within the program, but can also be edited. However, using inappropriate values will create undesirable behavior.

For example:

`default_font *times-bold-r-normal--12*` Sets the font in the left information window to "times bold 12 pt". Setting this too large will render the information unreadable. If problems occur, the file may be deleted to restore the defaults.

Some of the more useful settings are:

`want_colormaps 0` If set to 1, tnimage will set a colormap instead of allocating colors. The first *save_colors* colors in the colormap are never altered when the colormap is changed. If the value for "save_colors" is too low, colormap flashing will occur when the mouse moves from one window to another. If it is set to 1 or 0, the menus may also become unreadable.

`foreground_color`

`foreground_red`

`foreground_green`

`foreground_blue`

These values specify the color index and RGB colormap values of the foreground color in the main drawing area.

`information_width 120` width of the information area at the left. If set to 0, the information area is not displayed. This setting cannot be changed from within the program.

`default_font fixed` Sets the font in left information window to system default font (may be faster than the default Helvetica font).

`window_handle_size` Sets the size of active area around the edge of each image. Clicking on this area causes the image to be grabbed and moved.

`spacing` Sets the spacing between dots in the line used to indicate an area selection. Setting this to a higher value may improve performance on slower computers, but also makes the selection harder to see.

`help_directory` the directory in which the files `tnimage.hlp` and `formats` are located (default = /usr/local/lib/tnimage and $HOME/.tnimage respectively).

`format_list` Path name of the `formats` file (See Sec.5.6). The `formats` file should be copied into the $HOME/.tnimage directory for each new user.

A few X or X-like servers out there exhibit quirky behavior, particularly on laptops. This can result in colors being inverted or fine vertical stripes in the image. To accommodate such programs, 3 options are available to force `tnimage` to use specific settings.

want_byte_order - If not equal to -1, this sets the byte order for images to the specified value. For instance, `‘‘want_byte_order 0’’` would force `tnimage` to put the bytes least significant byte first, while `‘‘want_byte_order 1’’` forces `tnimage` to use most significant byte first order.

nosparse - If not equal to 0, forces `tnimage` to use packed pixel mode for 32 bit/pixel modes. Some X servers fail to return the correct value when applications check the packing mode. This option will cause problems if used on a display that is not 32 bits/pixel.

sparse - If not equal to 0, forces `tnimage` to use sparse pixel packing mode for 32 bits/pixel. Some X servers fail to return the correct value when applications check the packing mode. This option will cause problems if used on a display that is not 32 bits/pixel.

Locale conventions

In some countries, a comma is used as a decimal point separator. `Tnimage` checks the system locale and automatically substitutes the appropriate character (see 'man setlocale' for details). If you don't want this behavior, or if your locale is not set up correctly, you can set the decimal point separator to any character by changing the `decimal_point` entry in the file `$HOME/.tnimage/tnimage.ini`. Some choices are obviously better than others. Even if ',' is used, this will cause ambiguity parsing commands such as:

```
i = max(12,345);
print("The 3 answers are:",i,12,234);
```

You will need to remember to separate arguments with spaces to prevent getting error messages.

4.3 User-Defined Custom Buttons

The buttons at the left of the screen, above the information area, are completely user-configurable. The existing buttons can be removed, or any additional number of other buttons can be added. To add a new button, edit the configuration file `tnimage.ini` in the directory `$HOME/.tnimage` . Change the entry "nbuttons" to the total number of buttons. Then add a separate line for each button containing the word "button", the button label (which can be anything up to 6 characters, with no spaces or control characters), and the command. The command can be any valid macro command (See Sec.13), including parameters.

Examples:
`button Tran.  transparency`
adds a button labeled "Tran." which will set the transparency level for the current image.

`button Math math`
adds a button labeled "Math" which will activate the Image algebra dialog box.

`button FFT fft`
adds a button labeled "FFT" which will activate the Fourier transform dialog box.

`button Junk load my_junk.tif 100 100 0.75 0.666`
adds a button labeled "Junk" which will read the image file "my_junk.tif" at coordinates 100,100 and shrink it to 75% of its normal width and 66% of its normal height.

`button Gray color→grayscale`

adds a button labeled "Gray" which will convert the current immage to grayscale.

`button Other macro my-macro`
adds a button labeled "Other" which automatically loads and runs the macro from the file "my-macro".

If two commands are placed on the same line, the button automatically becomes a toggle button. Clicking it once causes the button to appear "pushed in", and the first command is executed. Clicking it a second time causes the button to return to its "out" position, and the second command is executed. The commands are separated by a colon (:), for example:

```
button Open open gold.tif 100 100:close
```

This creates a button named "Open" which reads the image "gold.tif" and places it at position 100,100 when clicked. Clicking it a second time causes the image to be closed.

Setting the second command to "null", or omitting it, prevents toggle mode. The button will be automatically unpressed. For commands like "slew", which toggles a state on or off, the same command should be entered for both the "on" and "off" commands. Setting the command to "cancel" causes the main Cancel button (if present) or the command "cancel" in a macro to reset the button,

Here is a sample `tnimage.ini` file for the author's system:

```
version 3.2.1
area_selection_mode 1
auto_undo 1
background_color 9474192
background_image_color 9474192
background_palette 1
background_red 144
background_green 144
background_blue 144
camera
camera_id 0
copy 1
compression /bin/gzip -f
compression_ext .gz
decimal_point .
decompression /bin/gunzip -c
default_font *helvetica-bold-r-normal--12*
double_click_msec 400
image_font *adobe-times-bold-r-normal--24-240-75-75*
information_width 120
foreground_color 0
foreground_red 0
foreground_green 0
foreground_blue 0
format_path /home/tjnelson/.tnimage/formats
help_directory /usr/local/lib/tnimage
jpeg_q_factor 80
luminosity_red 0.299
luminosity_green 0.587
luminosity_blue 0.114
object_threshold_red 50
object_threshold_green 50
object_threshold_blue 50
print_paper_xsize 8.5
print_paper_ysize 11
print_blue_factor 1
print_depletion 0
print_dither 8
print_graybalance 0
print_green_factor 1
print_intensity_factor 1
print_palette 2
print_paper 0
print_language 0
print_positive 1
print_quality 0
print_red_factor 1
print_resolution 300
print_device 3
print_vertical 1
print_hsize 4
print_ratio 1
```

```
print_rotation 0
print_interpolate 0
print_command lpr -P
print_devicename /dev/lp2
print_hpos 0
print_vpos 0
read_cmyk 1
save_cmyk 0
scanner /dev/scanner
scannerid 0
signif_digits 4
spacing 4
split_frame_cols 4
text_spacing 0
total_xsize 1190
total_ysize 862
use_custom_header 1
want_colormaps 0
want_crosshairs 0
want_messages 2
want_quantize 1
want_raise 1
want_shell 0
want_slew 0
want_byte_order -1
want_title 0
wavelet_path /usr/local/lib/wavelets
window_border 1
window_handle_size 2
xlib_cursor 34
nbuttons 20
button Cancel cancel:null
button Slew slew:slew
button Open open:null
button Close close:null
button New new:null
button Save save:null
button Sketch sketch:cancel
button Line line:cancel
button Move move:move
button Erase erase:null
button Scan scan:null
button Print print:null
button Label label:null
button Zoom zoom:repair
button Fcolor foregroundcolor:null
button Bcolor backgroundcolor:null
button Prop attributes:null
button Config configure:null
button Undo undo:null
button Quit quit:null
```

Section 5

File menu

5.1 Open image

5.1.1 Opening images

Reads an image from a disk file.

Vertical/Horizontal - Selects whether the image should be loaded in its normal orientation (vertical) or rotated 90° (horizontal).

Positive/Negative - Selects whether the image is to be loaded normally or as a negative (color-inverted) image.

Filename - Name of the image to be read. In the DOS version, the previous 20 filenames are kept on a stack and can be selected with the up and down arrow keys. Alternatively, pressing *Enter* when no filename is showing activates the menu file selector. In the DOS version, wildcard (* or ?) characters are permissible; thus, "*.pcx" is a valid filename, and will cause all matching files to be loaded.

X position - Horizontal starting pixel position to place the image (in pixels). If the image is in a separate window, the position is relative to the upper left of the screen; otherwise, it is relative to the upper left of the main window.

Y position - Vertical starting position to place the image (in pixels). If the image is in a separate window, the position is relative to the upper left of the screen; otherwise, it is relative to the upper left of the main window.

X size - If x size is 100, the image is loaded in its actual size. If set to some value between 0 and 100, the image is shrunk in the x direction.

Y size - If y size is 100, the image is loaded in its actual size. If set to some value between 0 and 100, the image is shrunk in the y direction.

Auto File Type/ Raw bytes -

Select 'raw bytes' to read the image as a series of bytes and override the automatic image file type detection. (If the image is an unknown format, "raw bytes" mode is selected automatically).

Invert byte order -

Certain other imaging programs save the RGB values in the wrong order. This could also happen with certain video cards. Clicking this option will switch the byte order to the correct value.

CMYK to RGB -

If the TIFF file happens to be a 32-bit CMYK image, tnimage will automatically convert it to RGB format for display purposes. Occasionally, an image file header says the image is a CMYK image but in actuality it is not. Un-clicking this option will cause tnimage to treat the image as if it were an ordinary RGB image. This option has no effect on images less than 32 bits/pixel.

Repair CMYK -

Some software packages create 32-bit images in which the colors are saved in the wrong order. Checking this box will change the order, allowing the files to be read correctly. (TIFF 32 bit files only)

Convert to gray scale

Changes the image to shades of gray instead of its original colors. This is particularly useful for CT or MRI images of greater than 8 bits per pixel, because the gray scale can be remapped as a "sliding scale" to enhance specific regions of the image.

Color reduction - Quantization/Fit current colormap/None

In an 8-bit/pixel (indexed-color) modes, color images must be reduced to 256 colors for display purposes. This can be done by "quantizing" the image or by trying to fit the image to the currently-selected colormap. Even though the quantization algorithm used is new and one of the fastest known, converting the image to 8 bits/pixel often still takes much longer than reading it from the disk.

Quantizing is more general than colormap fitting, and is guaranteed to give a viewable image. Many GIF files were produced with quantizing. However, because it changes the colormap, all other images currently being viewed will become "garbage" in an 8-bit mode. Clicking on an image restores the colormap for that image.

Colormap fitting uses less memory than quantizing. If the error message "Insufficient memory to convert to 8 bits/pixel" appears, try changing the color reduction method to "fit current colormap". This can be quite slow, however, for certain types of images.

Because quantized colormaps are usually not a continuous flow of colors, it may not always be possible to perform all types of quantitative analysis or filtering on a quantized 8-bit image. The best solution is to select "Change color depth" and convert the image to a 24 bits/pixel color image. The colors will then work as expected during filtering.

Fitting to the current colormap does not affect the other images on the screen, and gives a result that is still quantitatively analyzable and filterable. Also, if the original colors in the image are similar to those in the colormap, the result will be a much smoother image. However, if the colormap is substantially different, none of the colors may match up.

WARNING: If "Fit current colormap" is selected, operations that cause the colormap to be regenerated (such as changing the brightness of an indexed-color image) may give unexpected results–*i.e.,* incorrect colors.

In color modes, this option has no effect except when images are saved to disk with "8-bit/pixel colormap" specified as the file type. The colormap is also used in color modes when an image with 8 or less bits/pixel (i.e., a indexed-color image)is loaded.

All 8,16,24, and 32-bit images are kept in memory in their original color depth, regardless of the current screen mode. Other images are converted to the next highest multiple of 8. The conversion makes use of whichever colormap is selected. This means full-color images can be safely edited even if the computer's video card can only handle 256 colors. Editing operations also are handled in a manner appropriate for each image. Thus, for example, an area encompassing both an 8-bit image and a 24-bit image, can be selected and filtered with no problems.

See also under "Color settings" for more details.

If tnimage cannot identify the image type, or finds an invalid file, it asks if it should try to read it anyway as raw bytes. It then makes a guess as to the x and y sizes, and bits/pixel. These should be corrected as needed. In order to read raw bytes, the image should have a constant no. of bytes per scan line, be uncompressed, and have the image rows stored in consecutive order. It may also be necessary to skip a number of bytes to get the image to line up horizontally. This number has to be determined experimentally. If the file is not really an image file, in all likelihood only junk will appear on the screen.

This feature permits importation of other types of files that are not normally thought of as images. For example, to read a database of 16 x 16 Chinese characters from a bitmap font file, use the following parameters:

Filename: ziku (or name of the database)

x pixels: 16

y pixels: *length of file* ÷ *16* , e.g. 15000 or so.

bits/pixel: 1

bytes to skip: 0 The database will be loaded as a long strip of characters, which then can be

cut and pasted as desired. A short section is shown below. Of course, this is not the most efficient way to edit Chinese, but it demonstrates the usefulness of the feature.

Part of a bitmap font file loaded as an image.

See the following section for a detailed procedure for reading raw bytes and creating custom formats.

Procedure for converting a 24-bit image to an 8-bit image with a desired colormap.

A common operation is reducing a color image to 8 bits, with the constraint that the colormap should be the same as that of some other image. For example, a grayscale image might be accidentally converted to 24 bits/pixel and it is desired to reconstruct the original continuous color map.

1. Select "Color...Colormap...Select colormap".
2. Click OK and select the desired colormap.
3. Alternatively, the colormap of some other image can be copied by selecting "File...Create/Resize image" and clicking on "Method...Copy colormap only". The colormap of the image specified under "Image # to copy" will be copied to the currently-selected image.
4. Select "Config...Configure" and change the "Color reduction method" to "Fit current colormap".
5. Select "Color...Change color depth" and convert the image to the new depth (i.e., 1 byte/pixel).

5.1.2 Reading CT, X-ray, or MRI images, or raw bytes

Below is a detailed example for reading images produced by X-ray, CT or nuclear magnetic resonance scanners and other equipment. These machines typically use unusual file formats not recognized by tnimage, but usually the images can easily be read as "raw bytes".

The same procedure is applicable to any non-compressed file of unknown type.

NOTE: Tnimage can create a "custom" image file format, which is a superior method for reading medical image files, because once the custom format is set up, reading the image is transparent to the user, obviating the necessity for reading "raw bytes" as described here (See "Create File Format").

(1) Select "File...Open image" and enter the image file name.

(2) Click on "Raw bytes".

(3) Click on "OK".

(4) Tnimage will make a guess at the image width. If the image width is known, enter the image width and height. Otherwise, if the guess turns out to be wrong, use the auto-increment method described below to determine the width. In this case, the width happens to be 381.

(5) Select the bits/pixel of the image. This is usually 12 or 16 for CT images (MAMOGRAM.LUM is 16 bits).

(6) (Optional): If the length of the image file header is known, enter this value under "skip bytes". Otherwise, leave it as 0.

(7) Click on "OK" to open the image.

(8) Once the image is loaded, if the skip bytes were incorrect in step 6, the image border will be visible as a vertical strip in the image. Use the mouse to locate the X-position of this strip. Close the image and re-open it, using the X-position * bytes/pixel as an estimate for skip bytes.

(10) If the image was color, it may be necessary to add 1,2, or 3 additional skip bytes to get the colors to line up correctly.

(11) If the image is grayscale, the grayscale mapping can be adjusted by selecting "Color...Grayscale map". This the image to be adjusted interactively as a sliding scale, to highlight areas of different intensity (See "Grayscale brightness" below).

(12) The image can now be saved in a conventional file format.

Determining the width of an unknown image:

Tnimage has a unique feature for determining the width of an image whose dimensions are completely unknown. This procedure is substituted for step (4) above:

(4a) Make a conservative guess of the image width. For instance, if the image is known to be square, and it is 16 bits (2 bytes) per pixel, use

$$width = \sqrt{(}filesize/2)$$

as the starting estimate.

(4b) Click on "Auto increment". This will cause the width to be increased with each scan line.

(4c) For "Increment", enter "1" or "2".

(4d) Click on "OK" to load the image. The image should consist of a swirling pattern, with fuzzy lines in a horizontal "U" shape. Usually, the image is slightly clearer at the vertex (marked with a *). If so, find the Y-coordinate of the vertex using the mouse. Subtract 32 (the size of the menu bar) from this value and add the result to the previous estimate of the image width.

(4e) Unload the image.

(4f) Turn off “Auto increment” and load the image again. The image should now be recognizable. If the image slopes to the right (when viewed from top to bottom), the value is still too low. Increase the image width slightly and try again. If it slopes left, the value is too high.

If the image is still not recognizable after these steps, try changing the byte order, or check the “Convert–gray scale” option.

If the wrong number of bits/pixel or y size for a “raw bytes” image was selected, the automatic grayscale detection may estimate the maximum and minimum gray levels incorrectly. This will result in an image that is too dark or too light. This can be easily corrected by remapping the gray levels (See *Grayscale intensity mapping*).

5.1.3 Reading unusual image files

Raw image files

The .IMG files produced by some other software contains a value for the image width that is off by 1. Thus, in order to read the image properly, it is necessary to read the image as “raw bytes”. After manually changing the width from 511 to 512, the file will read normally.

Reading Images from Macintoshes

Files imported from Macintosh via Mac-TCP are corrupted by the addition of an extra header unless they are sent as “Raw Binary”. This often creates the impression that PC-based programs are incapable of reading Macintosh images. Tnimage, however, automatically detects the presence of the header and eliminates it before reading any image.

Reading Raw ASCII images

Raw ASCII images consisting of a series of integers are readable by tnimage. To read an image in ASCII format, select *Raw ASCII* as the File Type in the Open Menu. Then set the remaining parameters the same way as for *Raw Bytes* (Sec. 5.6.1). Some of the options (such as bit packing) do not apply to ASCII files and their setting is ignored.

Reading text files

Text files are a convenient way of adding preformatted labels to an image. The text file must have the extension “.txt” to be identified. The user is prompted to click at the position to place the text, which is then drawn on top of the current image(s) using the current font and foreground color. No new image is created. If the text is too large to fit, it is truncated. The text is also placed in the alpha channel so it can be selected like any other label, but it cannot be edited in the image editor.

Reading labels as a text file is also a simple way of adding text in non-English character sets (such as Japanese).

Reading multi-frame (3D) images (Unix version only)

Select “File type = Raw 3-D”, select a filename, and then enter the following information:

No. of frames = Total number of sub-images in the file.

Frame width(pixels) = The width of each frame in pixels.

Frame height(pixels) = The height of each frame in pixels.

Bits/pixel = The image depth (8,15,16,24, or 32).

Multi-frame images are also supported in custom formats (Sec, 5.6).

Reading images from confocal microscopes and PET scanners

These images, which contain multiple views in 3 dimensions, must sometimes be read as Raw 3D images. Typically, the image size for confocal images is 512 x 512. PET scan images are typically 128 x 128. It is necessary to know the number of frames (slices) in the image. Most PET scan images are only 8 bits/pixel, while confocals often produce 24 bit/pixel images.

Reading images from AFMs (atomic force microscopes)

Images from atomic force microscopes are stored as signed integer values representing measured height. These images can be read by setting File Type to `raw bytes`, and then clicking `Data type = Signed` and `Image Source = Mac` in the Read Raw Bytes dialog. The X and Y sizes in pixels, as well as the total bits/pixel (usually 16) must also be set before reading the image.

Alternatively, the `afm` custom format (included with `tnimage`) can be used to read and write these images transparently. The image must have the extension `.afm` in this case. This can easily be changed by editing the format descriptor file, or by opening the Create Custom Format dialog.

Raw image files

BioRad has two different proprietary image formats: one for their microscopes and one for their gel documentation scanners. Tnimage supports reading of the 16-bit images (.1sc format) from Bio-Rad scanners. Since BioRad does not release their file format specification, the implementation is probably not perfect, and it is not possible to create 1sc format images. Depending on how the image was scanned, the contrast in the image may need to be increased before the image is visible. Occasionally the program may guess wrong about the byte order. This will cause the Bio-Rad image to appear to be composed of grainy black and white pixels. If this happens, click “Swap image bytes” and read the image again.

The BioRad microscope format is not readable directly, but can be converted to the supported PGM format by a utility called bioradtopgm, which is part of the NetPBM package.

Animated and multiframe images

If a GIF or custom format image contains multiple frames, it is automatically animated when read. For other multi-frame images, animation can be started by selecting “Image..Image properties” and checking “Animate”, or by using the “movie” command (see Sec. 6.5). Click the main Cancel button to stop the animation.

Pre-processing

Image files can be pre-processed by any other program before being read by tnimage. This program must be able to read the file and send its output to `stdout` (the `-c` flag in `gunzip` does this). If a file ends in the specified extension, the file is piped through the command shown in the “Read Image” dialog before being read. Typically, this would be a program to decompress the image file, but could also, for example, decrypt it or perform some other action on it. The advantage of using an external program to compress image files is that, should a more efficient compression algorithm come along, the new program can be substituted without invalidating the image file format. Also, file formats such as FITS, which have no compression, can be easily stored in compressed form using this feature.

Note: Because of limitations in the design of certain image file formats (namely TIFF), pre-processing cannot be done using a pipe but must use a temporary intermediate file. Thus, this mechanism should not be relied upon to provide absolute security with an encryption utility.

The default pre-processor is `gunzip`. This means you can read any `gzipped` image directly into `tnimage` without needing to decompress it first.

5.2 Save image

Transfers an image or selected portion of the screen to a disk file.

File format Format in which to save the image. See *Custom format* below for details on creating files in a customized format).

Save entire image/selected region - If "entire image" is selected, the currently-selected image will be saved in its entirety. Otherwise, only the currently-selected screen region (which can beyond an image or contain 2 or more images) will be saved. The actual size of the region saved is displayed after the file is written.

Bits/pixel -

Select a standard depth of 1,8,15,16,24, or 32 bits/pixel, or select "other" to specify any other value between 1 and 32. The resulting file size will be proportional to the bits/pixel. Saving an image in a lower bits/pixel than it was created can cause a loss in color accuracy. For example, if an 8-bit image is saved as a 3 bit/pixel image file, the colors in the image will appear slightly different from the original colors. If it is saved as a monochrome (1 bit/pixel) image, all pixels above 50% of the maximum color will appear white and those below 50% of maximum will appear black.

NOTE - Some file formats (e.g., GIF and PCX) only allow certain values for bits/pixel and number of colors. tnimage displays an error message if an illegal value is selected.

WARNING - Due to limitations in the capabilities of some other programs, saving images using "non-standard" bits/pixel or using "non-standard" numbers of colors can result in image files that are not readable by some other image viewers. For example, one well-known shareware file interconversion program can only handle 4,8, and 24 bit/pixel TIFF images, but not the common 15 and 16 bit images. One commercial Macintosh program can only handle 8,15,24, or 32 bit/pixel images, but not 16 bits/pixel. Very few other programs can handle odd values such as 12 bits/pixel. However, these values can be extremely useful and widely used in handling radiological, electron microscope, or MRI images, which are often of unusual pixel depths.

Tnimage will display an appropriate warning if a "non-standard" file format is selected.

Treat data as Color / Gray scale -

When saving images greater than 8 bits per pixel as TIFF or Custom files, it is sometimes useful to save the image as if the image data represent shades of gray instead of colors. Selecting "gray scale" causes tnimage to treat the image as if each pixel value was already a "luminosity", and skips the conversion from RGB to luminosity. This setting is ignored if the image is saved as 8 or less bits/pixel, or if it is saved as a color image (by selecting "2" or "3" as the number of primary colors). Normally, this setting should be kept on "Color".

Filename - The file name under which the image will be saved (can include drive and path).

Image no. - Indicates the currently-selected image which will be saved if "save entire image" is checked. Has no effect if "selected region" is checked. This option is mainly for informative purposes and does not normally need to be changed.

Extra TIFF Param.(for 'Other') -For TIFF and custom file formats, it is also possible to customize several additional parameters.

No.of primary colors(1-4) - If 1 primary color is selected, the image will be converted to grayscale before saving, and the red, green, blue, and black bits/pixel will be ignored. If more than 1 color is specified, the number of bits for each color, when added together, must equal the total number of bits/pixel specified under "other bits/pixel".

- *Red bits/pixel* No.of bits of red to save.
- *Green bits/pixel* No.of bits of green to save.
- *Blue bits/pixel* No.of bits of blue to save.
- *Black bits/pixel* No.of bits of black to save (CMYK files only).

For example, to save only 4 bits each of the red and blue planes of an image, set the following values:

- 'Other' bits/pixel (on left side of dialog box) = 8
- Number of primary colors = 2
- Red bits/pixel = 4
- Blue bits/pixel = 4
- Green bits/pixel = 0
- Black bits/pixel = 0

NOTE: Tnimage cannot display more than 8 bits/pixel of each color (i.e., a total of 24 bits/pixel). This is due mainly to limitations in the current generation of video cards. Thus, there is no advantage in setting the bits/pixel for any one color higher than "8".

Black bits/pixel can only be specified if the "cmyk" option is checked.

RGB to CMYK -Converts the image from RGB to CMYK (cyan, magenta, yellow & black) format when saving. (CMYK images are automatically converted to RGB for display, since there cannot be a "CMYK computer monitor"). For non-standard bits/pixel modes, it is also necessary to specify the number of red/cyan, green/magenta, blue/yellow, and black bits/pixel. If a non-standard format is selected, tnimage will display a warning to that effect. Selecting CMYK is only allowed for the 32 bit/ pixel standard mode and the "Other" (non-standard) mode. Attempting to save an image with CMYK checked in any other mode will produce an error. This is done to ensure consistency with the TIFF specification which (as of version 6.0) does not recommend using lower color depths for CMYK.

This option is useful for exporting the image to other programs which cannot create their own CMYK images for printing.

JPEG quality factor - An integer between 1 and 99. Values above 90 or below 10 are not recommended. Smaller values result in greater compression and more signal loss. A quality factor of 100 would require arithmetic compression and is not supported because of patent considerations.

NOTE: JPEG is a lossy file format. This means an image saved in JPEG format will lose some information. Images consisting of text and sharp lines are not good candidates for Jpeg compression, because the compression algorithm is optimized for realistic scenes such as images from photographs. Files should only be saved in Jpeg format when all image processing has been completed, and the smallest possible size is needed. JPEG is the most suitable format for images on Web pages.

Some settings, such as red, green, blue, and black bits/pixel, and CMYK conversion, are ignored for JPEG.

Animated and multiframe images

Animated images can be saved in GIF89a or custom formats. Using any other format creates an error message.

Post-processing

Image files can be post-processed by any other program after being written by tnimage. This program must be able to receive data from `stdin` and send its output to a specified file (e.g., `gzip -f`). The extension specified in the "Save Image" dialog will be added to the filename.

Typically, the use of this would be to compress the image file, but it could also, for example, encrypt it or perform some other action on it. The advantage of using an external program to compress image files is that, should a more efficient compression algorithm come along, the new program can be substituted without invalidating the image file format. Also, file formats such as FITS, which have no compression, can be easily stored in compressed form using this feature.

The Post-Processing command is automatically deselected if the original image was not processed by the corresponding decompression program.

5.3 Print

Prints the image or selected portion of the screen on a laser printer. A laser or inkjet printer that uses a page control language similar to Hewlett-Packard's PCL3 or PCL5, or any PostScript printer supporting PostScript Level 2 or above is required.

Printing Black-and-White or gray scale images

Most laser printers are optimized to print text. Often this results in prints that are too dark. To select a lighter print intensity, select 'file...print' from the menu, then click on 'Color/Darkness' and adjust the red, green, or blue intensity factors as desired. 128 is normal (dark) printing. Any other values will be multiplied by the data going to the printer to make it lighter or darker. For grayscale images, the red, green, and blue intensity factors will be averaged.

Currently, only laser and inkjet (bubble jet) printers are supported. No support is planned for dot-matrix printers, because the image quality would be unacceptably low.

Print quality can often be improved by using glossy paper sold for pen plotters instead of regular paper.

In addition, some PostScript printers cannot handle images > 8 bits/pixel. In this case, it is necessary to convert the image to 8 bits before printing.

—Column 1—

Vertical/Horizontal - Selects whether image will be printed in "portrait" (vertical) or "landscape" (horizontal) mode.

Positive/Negative - Print normally (white on the screen = white on the paper) or as a negative.

Print entire image/ print selected area - Select whether to print the entire currently-selected image, or the currently-selected screen area which may span 2 or more images.

Color type

Printer type: PCL / PostScript - Select printer language to use.

—Column 2—

No. of copies - Number of copies of the image to print. Not all printers accept this command.

Printer Device/file name

Color adjustment - Darkness of printed image (255=darkest). The red, green, and blue factors are averaged when *color type* is "B/W".

Vert. offset (inches) - Offsets the image by the specified no. of inches down the page.

Horiz. offset (inches) - Offsets the image by the specified no. of inches to the right across the page.

PCL Printer settings

For PCL printers, such as old HP LaserJets and most inkjet printers, the following options also apply:

Media type - Optimizes the output for the type of paper being used. The following table shows the differences used for different media types on a typical PCL printer:

Media Type	Print Quality	No. of passes[1]	Recommended Depletion[1]
Plain	Draft	1	–
	Normal	2	25% w/gamma
	Presentation	4	50% w/gamma
Bond	Draft	1	–
	Normal	2	25% w/gamma
	Presentation	4	50% w/gamma
Premium	Draft	1	–
	Normal	2	25% w/gamma
	Presentation	4	50% w/gamma
Glossy	Draft	2	25%
	Normal	4	25%
	Presentation	4	0%
Transpar-ency	Draft	2	25%
	Normal	4	25%
	Presentation	4	0%

On certain printers, e.g., H/P 540, the number of passes and the depletion are determined automatically by the printer and may differ from the value in the table. On some other printers, depletion and no. of passes cannot be changed, and setting them to different values has no effect.

[1] Recommended by H/P.

Resolution - Selects the desired dots/inch resolution to print the image. Normally this is set to the highest value the printer can handle (see discussion above).

Typical values are:

150 dpi

300 dpi

600 dpi

1200 dpi

This number must be set **exactly**. For example, a 300 dpi printer may ignore the setting, or it may print nothing at all if the resolution is set at 299 or 301.

Dither size - Determines the number of separate gray or color levels that can be printed:

Dithering size	No. of gray/ color levels
1x1	2
2x2	4
4x4	16
8x8	64
16x16	256

The size of the printed image also increases proportionately with the dithering size. No "error propagation" is performed, in order to maximize resolution for text.

Print Quality: Draft/ Normal/ Presentation - Selects the quality of printout. Presentation quality takes longer, but gives slightly better results.

Not all printers support this option.

Depletion – Improves image quality by removing a certain percentage of the pixels. It is recommended to leave this setting at "printer default" unless unsatisfactory results are obtained. The highest two settings also apply gamma correction to the printed image, if this feature is supported by the printer. Gamma correction produces a smoother transition from one color to the next.

See the table above for recommended depletion levels.

Not all printers support this option.

Printer gray balance - If checked, the internal gray balance adjustment in the printer will be activated if present. This is supposed to adjust the colors so that blacks appear black instead of dark green. Do not check this box if you have changed any of the values under "Printer color adjustment" (see above), since this would result in correcting the gray balance twice. If the printer does not support this option, checking this will have no effect.

PostScript Printer settings

For PostScript printers, the following additional options apply:

Horizontal image size (inches)

Image ratio (y/x) -

The output will be scaled accordingly to shrink or enlarge the image to the specified size.

Rotation (Degrees) The printout will be rotated counterclockwise by the specified angle. The angle is centered at the lower left corner of the paper. Note that a rotation may cause part of the image to be off the page unless 'horizontal position' is also adjusted. The smallest increment is 0.0001 degree.

Interpolate

If checked, and if the output resolution of the printer is high enough, additional dots will be added to smooth the output.

NOTE: *PostScript Level 2 or above is required.*

Color printing Color printing uses the same parameters as B/W printing, except it is necessary to select RGB, CMY, or CMYK color instead of grayscale. Color printing can take a much longer time than B/W printing, and requires more printer memory on a laser printer. Use B/W mode whenever possible. The choice between the color types is subjective; however, CMYK generally gives darker blacks.

NOTE: It is not necessary to convert an image to CMYK format to print it in CMYK mode.

Color type - Selects color printing mode.

Grayscale - Recommended for non-color printers and for monochrome images. The pixel values are sent to the printer without filtering them through the colormap. Thus, if the image was made lighter or darker by dragging the colormap palette, these changes will *not* be reflected in the printout. This setting should not be used for the following types of images:

1. Images that have been converted to a lower bit/pixel depth, e.g. from 24 bits/pixel to 8 bits/pixel.
2. Images containing color.
3. Images such as GIFs which have a discontinuous colormap.

RGB/indexed color Recommended selection for PostScript color printers. Not recommended for PCL inkjet printers, because some PCL printers insist on interpreting this command to mean that all pixels not explicitly defined are to be printed as "black". This causes a wide black stripe to be printed next to the image, and causes printing to take a much longer time.

For PostScript printers, this is a good setting to use for both grayscale and 8-bit color images.

If the image is 8 bits, the colors or gray levels are determined by the colormap. If it is a true-color image, the RGB values are used directly.

Prints using red, green, and blue inks if available (not supported by all printers).

CMY color - Recommended selection for PCL color printers. Not recommended for black & white printers.

CMYK color - For color PCL printers - gives darker blacks and brighter colors than CMY but takes 33% longer. For PostScript printers, this option is the same as CMY.

Printer color adjustment - Adjusts each color to to match the printer output with the screen. For each color, a value of 128 (the default) is "normal". Increasing the value will increase the intensity of that color. If CMY or CMYK is selected, the cyan, magenta, and yellow inks are changed. If RGB is selected, the red, green, and blue inks are changed. If B/W-grayscale is selected, the black value is taken as the average of the three color values.

NOTE: For CMYK printing, it is important to keep the 3 colors balanced so that the average is 128. Otherwise, the black ink, whose value is derived from the 3 pigments, will also become lighter or darker.

NOTE: Some printers do not handle 24-bit PostScript correctly. If problems occur, convert the image to 8 bits/pixel before printing. (This will also greatly reduce the network load).

Printing to a file or another printer

A 'print file' can be created by entering a filename instead of a printer name under *Printer device/file name* in the "Print..." menu. This will create a print file which can be sent to the printer later, such as at night when it is less busy.

For example, in DOS, printing of a file can be done using the command:

```
copy /b print.tmp lpt1
```

However, `copy /b` does not work correctly on some printers. In this case, try `print.exe` , which is a replacement for DOS's `print.exe` . This program is available at the ftp site. Print.exe is also useful for printing large print files, such as those created by Ghostscript.

In Unix, printing of a file could be done using the command:

```
lpr -Pmyprinter print.tmp
```

To schedule the printing at a specific time, use a program such as `tnshell` in DOS, or `cron` or `at` in Unix.

If a printer device is entered, the image will be printed on that device. *Printer device* refers to the printer name as it is known to the operating system, for example:

DOS version: lpt1 lpt2 prn

Unix version: /dev/lp0 /dev/lp1 /dev/lp2 /dev/printer, etc.

Printing under Unix

The Unix version of tnimage has the additional options *Printer command, printer device, and print to file* , for printing to network printers, parallel port printers, or to a file, respectively.

Warning: If the specified "printer device" is not a printer or a file, bad things will happen.

In the box "Printer command", a command can be entered, such as `lpr` − `P`*printer* − `K`*copies*.

Printing in Windows (DOS version only)

In Windows and Windows95, it is possible to print to any network printer provided the printer has been set up correctly with the Print Manager. There are two ways of doing this: "capturing a printer port", and directly printing to a Windows-style shared printer name. If the "Capture Printer Port" in Windows is clicked, printing to DOS printer devices lpt1 to lpt4 is captured and sent to the specified network address. In this case, "`lpt1`" to "`lpt4`" should be entered. Otherwise, enter a Windows address such as "`\\MY_NT_SERVER\ITS_PRINTER`". Printer port capture must be "off" for the specified printer in order for this to work.

5.4 Create/Resize Image

A new image can be created by clicking and dragging with the mouse, setting a fixed size and position, or duplicating an existing image. The image window can also be resized.

Use mouse - Use the mouse to select the region for the new image. The screen contents at that location will be copied into a new image.

Fixed size - Enter the x and y size (in pixels) and the desired number of frames. Then click on the point which you wish to make the upper left corner of the new image. The screen contents at the specified location will be copied into a new image of the specified size. All frames will be identical.

Duplicate image - Enter the x and y position for the upper left corner of the new image.

Resize image window - Changes the size of the image and its window to the specified size.

Change Title

Changes the name of the image. This must be a valid filename. If the title contains illegal characters (such as spaces or control characters in the DOS version), they are automatically removed and an error message is displayed.

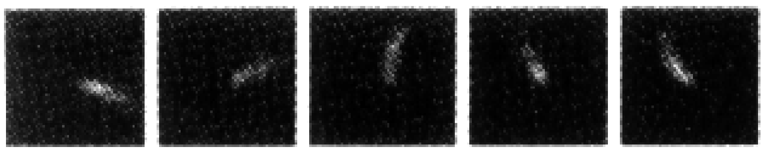
5.5 Multi-frame Images

Animated or multi-frame images can be created from a large panel containing subimages, or from separate individual images. To create an animated image from separate images, start with a series of single-frame images of identical size and color depth. Better results are obtained if the frames are 24 bits/pixel, then converted to 8 bits/pixel afterwards if necessary. When `tnimage` changes the depth of an image, all the frames are converted simultaneously and a common colormap is generated that fits all the frames.

Creating multi-frame images from a panel

Panel→multiframe image takes an image consisting of several sub-images, and puts each subimage into a multi-frame image. If 'Manually find frames' is checked, you must click on the upper left corner of each subimage to indicate its location. Otherwise, the subimages are found automatically, using the specified x and y spacing between subimages and the specified upper left x and y for the first subimage to calculate the location of each subimage.

The program takes a wild guess as to the size of the multiframe image and the number of frames to create from the panel. These values should be corrected as needed.



Example: Part of a typical composite image suitable for building a multiframe image.

Several frames from an image of a rotating F1-ATPase $\alpha_3\beta_3\gamma$ subcomplex. Noji *et al.* [1] coupled the γ subunit of F1-ATPase to a fluorescent actin filament and observed that this protein acts as a naturally-occurring molecular motor. Only 5 of the 105 frames from their article are shown.

If a frame is not aligned correctly with the other frames, use the "Image..shift" menu to adjust its position (see "shift", Sec. 7.5). If this doesn't work, use the frame controls to move to the offending frame and copy the desired region manually onto the image. The region will be pasted onto the frame that is visible.

Combine images→multiframe works similarly, except that instead of a panel, the subimages are taken from a list of individual images. When you click 'Accept', an information box appears allowing you to enter the image numbers to use for each frame. A multi-frame image will then be created with each specified image comprising one frame.

Note: The images to be combined must all be the same pixel depth. If the source images are already multi-frame images, only the first frame is used.

Example: Creating animated GIFs

NOTE: Animated GIFs must be 8 bits/pixel.

1. Convert each source image frame to 24 bits/pixel.
2. Add the desired content to each frame.
3. Click "New/Resize image...Images →multiframe" option to create the animated multi-frame image.
4. Enter the list of image numbers for each frame in the information box that appears. This is a series of numbers or a range, e.g., '`1 4 9-12 22`' would include images 1,4,9,10,11,12, and 22 as frames in the new image.
5. Set the frame rate using the "Image properties" dialog.
6. The new multiframe image will have the same depth as the source frames (i.e., 24 bits/pixel). To save as a GIF, it must be converted to 8 bits ("Color...Change Image Depth").
7. Save final image in GIF89 format (GIF87 images cannot contain animation).

Converting a multiframe image to a panel

Multiframe→panel converts a multi-frame or animated image to a large single image in which each frame is lined up in columns. Set the desired screen position for the new panel image, the desired number of columns, the upper left position in the panel to put the first frame, and the x and y spacing (in pixels) between frames. A new single-frame image of the appropriate size will be created, in which each frame of the original is visible.

Split multiframe →images is similar, except instead of creating a panel, it creates a number of small single-frame images from the multiframe image, and then lines up these images into columns (see below).

Displaying multi-frame MRI images as an image panel

MRI and PET scan images are often more conveniently viewed as a series of individual frames than as animated images. This can be done by selecting "New/Resize image...Multiframe →Images" option. Click on the "Align Images" toggle button and select the desired number of columns. Set the desired x and y position for the upper left of the new display and click "Accept".

This will convert the multi-frame image to separate images, one per frame, aligned in sequence from left to right and top to bottom. The frames can be manipulated or saved individually. The original multiframe image is still intact in its original location.

WARNING Each image created with this option contains only a single frame. If you wish to save all the frames to disk, be sure to save the original multiframe image, or alternatively click "Images →multiframe" to rebuild the multiframe image. (This should be easy because the correct parameters are automatically set when the image is decomposed into separate frames).

If the 'Align Images' toggle button is "off", the new images will be stacked on top of each other instead of lined up in columns.

Multiframe images can be split into an image panel automatically when reading from disk by clicking the "Split frames" toggle button.

Adding a frame

Add frame to image increases the number of frames in the current image by 1. The new frame contains all zeros, i.e., is black.

5.5.1 Creating Composite Images

Composite 2 or more images creates a larger image into which the source images are pasted at specified locations.

Set the x and y size of the new composite image, select "Composite 2 or more images", then click Accept. A message box will be displayed allowing the image numbers of the source images to be specified; for example, `1 2 5-7 19` . All the source images must have the same bits/pixel.

A small spreadsheet will then appear. Type the x and y coordinates in the left and right columns, respectively, for each image. (Note that 0,0 is in the upper left corner of the new composite image). The subimages will be copied to the new composite image, at these coordinates. Make sure the x and y size values for the new image are large enough, otherwise the subimages will be cropped. (X and y size are specified on the Create Image dialog).

Notes:

1. All the images must have the same bits/pixel before creating a composite image.
2. The subimages can be any size.
3. The new image takes on the properties of the first subimage in the list. For example, if the first subimage is grayscale, the composite will also be grayscale.

Creating composite image from a list of coordinates

Copying a large number of rectangular regions from one image to another can be tedious, and it can be difficult to line the areas up manually. This option allows composites to be created using a file from disk.

Select "Spot list→panel" and click Accept to create a new composite image based on a list of coordinates. The list should be a text file in the format created by `tnimage`'s Image Registration function (sec 7.4.3), containing coordinates of regions in the currently-selected image which are to be copied into a new image.

If "Fixed spot size" is selected, the bounding box in the coordinate file will be ignored, and the specified fixed x and y sizes will be used. Otherwise, regions specified by the xmin, ymin, xmax, and ymax columns in the file will be copied into the new image. The spacing between the boxes (in pixels), and an margin of extra pixels around the bounding box which should also be copied, can be specified in the dialog box.

For example, below are two lines from an unwarped spot list from a 2D acrylamide gel, created in Image Registration.

```
#Label orig.x orig.y Size  Signal  xmin ymin xmax ymax  Identity
1      430    75     374   1103    424  74   450  108   1
2      604    91     32    39      603  90   610  99    2
```

If this file were used, regions between (424, 74) and (450, 108) and between (603, 90) and (610, 99) would be copied from the current image and lined up in a new image, providing documentation of the two areas.

Create image from selected area

A new image is created containing the currently-selected area. Areas larger than the size of the screen can be selected by the shift-click method (Sec. 3.5.1), which selects points on the image.

5.6 Create file format

Defines a new image file format. An unlimited number of new formats can be added. Once defined, the new format is automatically recognized by `tnimage` and can be loaded and saved without any special action.

Custom formats are particularly useful if reading medical images from an X-ray scanner, which vary from one manufacturer to the other. Images can be compressed by a number of available methods.

Of course, with this much flexibility it is impossible to test every possible combination of parameters on every destination platform, so it is advisable to make a test image before proceeding, to make sure the file is readable on the target system.

Note: It is necessary have write permission in the directory where the file `formats` is stored in order for this to work. The location of this file can be changed by changing the `format_list` line in tnimage.ini (default location is $HOME/.tnimage/formats).

Defining a custom format is a very flexible and convenient way of storing images, since someone else wishing to view the image only needs the corresponding image descriptor file.

When reading a custom image file, tnimage first looks for the file descriptor in the directory

`$HOME/.tnimage/formats` .

This file descriptor is an ordinary text file, which can be created manually or by using the "Create file format" menu option. The filename of the file descriptor must be the same as the "Identifier" described below.

Here is the procedure for creating a new custom format:

1. Click on "File...Create file format".

2. Select target platform - This option will determine the order in which bytes should be swapped. If the destination computer is not a PC, Mac, or IBM mainframe, pick whichever seems closest. Many UNIX systems use "big-endian"-type processors similar to the Mac. MVS systems use "little-endian" processors similar to the PC.

3. Bit packing - If bits/pixel is not a multiple of 8, the individual image bits can be packed into bytes or allowed to expand to the next largest multiple of 8. This allows for faster reading but results in larger file sizes. Packing bits is performed differently in GIF and TIFF files as shown:

   If mode=GIF

   Assumes bits are packed in the following format (example for n=9 bits pixel, where a,b,c... are pixels) :

```
                           7  6  5  4    3  2  1  0
                          ----------------------------
    Byte 1                a7 a6 a5 a4   a3 a2 a1 a0
    Byte 2                b6 b5 b4 b3   b2 b1 b0 a8
    Byte 3                c5 c4 c3 c2   c1 c0 b8 b7
    Byte 4                d4 c3 d2 d1   d0 c8 c7 c6
    ...
```

   If mode=TIF Assumes bits are packed in the following format (example for n=9) :

```
                           7  6  5  4    3  2  1  0
                          ----------------------------
    Byte 1                a8 a7 a6 a5   a4 a3 a2 a1
    Byte 2                a0 b8 b7 b6   b5 b4 b3 b2
    Byte 3                b1 b0 c8 c7   c6 c5 c4 c3
    Byte 4                c2 c1 c0 d8   d7 d6 d5 d4
```

   In most cases, the bit packing should be left at "none".

4. Format identification - This determines how `tnimage` should recognize an image in the new format. If "by identifier", tnimage will look for the specified identifier string at the specified offset in the file. For instance, if the file always starts with the characters "II", the identifier is II and the identifier offset is 0. If the image is to be identified "by extension", any file having an extension the same as the specified identifier will be identified as a file of that type. This would be appropriate if the file always ended with the extension ".lum" for example.

   Note: Compressed formats can only be identified by extension.

5. Identifier - any string up to 64 characters that should be used by tnimage to identify the file. The final 'null' (0x00) character is not expected to be in the file.

6. File offsets - This opens a list of parameters that may be extracted from the image file header. Each parameter must be present at the specified offset in all image files of this type. The maximum size in bytes for the parameter is shown in parentheses. For example, an entry shown in the listbox such as

   `Y size (2) 8`

means the image contains two bytes starting at a position 8 bytes from the start of the file that indicate the image height (Y size) in pixels. If an offset is set to 65535, it will be ignored.

It is very important that none of the offsets overlap with each other, and that no offsets are included in the file descriptor that do not actually exist in the image file. For example, if an offset for a palette (color LUT) is specified at offset 76, no other parameters can have offsets between 76 and 844, because the palette uses 768 bytes. `tnimage` does not check for this.

The parameters in the table below can be recognized from the image file. Note that, when creating the format descriptor, it is necessary to enter the *position in the file* where the parameter resides, not the actual value. At run time, `tnimage` will fill in the value automatically. The "number of bytes" in the table below refers to the maximum size in bytes that the parameter can attain. Thus, for X size, with only 2 bytes, the maximum width of an image is only 65535. This was done to provide the ability to read certain other (older) image formats as custom format images.

Parameter	Bytes	Meaning
Identifier	64	Format identifier or file extension
X size	2	Width of image in pixels
Y size	2	Height of image in pixels
Bits/pixel	2	Bits per pixel in image
Color type	2	0=Grayscale, 1=Indexed color, 2=true color
Compression	2	0=no compression, 1=compressed
Red bpp	2	Bits/pixel for red
Green bpp	2	Bits/pixel for green
Blue bpp	2	Bits/pixel for blue
Black bpp	2	Bits/pixel for black
X position	2	Position of left edge of image in pixels
Y position	4	Position of top of image in pixels
Frames	4	No. of frames in image
Frames/sec	2	Speed of "movie" for multi-frame images
Chromakey	2	-1=inverse, 0=none, 1=normal chromakey
Chr.gray min	4	Min. pixel value for chromakey if image is grayscale
Chr.gray max	4	Max. pixel value for chromakey if image is grayscale
Chr. RGB min	4	Min. RGB value for chromakey if image is color
Chr. RGB max	4	Max. RGB value for chromakey if image is color
Palette	768	Colormap for image, if image is Indexed color
Image offset	2	Starting offset for image data.

When creating a new image format, it is advantageous to separate the parameters by at least 4 bytes for future compatibility. Also, there is no harm in specifying an offset for a parameter (such as frames/sec) even if it will never be used.

The image data are stored in pixel order, starting at the upper left of the image, as `RGBRGB...` for color images or `IIIIIII..` for indexed or grayscale images. Successive frames are stored without any intervening spaces, except that, if bit packing is used, unused bits at the end of a line or frame are set to 0 and each line and frame starts on a new byte.

Unless the images are always the same size, the image offset should be the highest offset. If the format descriptor file is edited manually, the entries do not need to be in any particular order.

NOTE: If a nonzero value is specified for "Bytes to skip", this value will override any value in the file at position *image offset* for the start of image data.

7. External header file - If the format of the image file is unknown, a custom format can still be created provided that a sample of the image is available. Enter the filename of this sample image in "Header file", and the number of bytes in its header in the next box. Tnimage will then copy the specified number of bytes from the sample image into the start of any new images created in that format. The sample image must always be present. The number of bytes that should be copied to create a valid image file is usually a power of 2 (32, 128, etc). This procedure is obviously risky, since the header being copied may only be valid for an image of a certain size or image depth.

8. Default parameters - these are numbers that will be used as defaults if no offsets are specified for a given parameter (i.e., the offset is 65535). Most of these are self-explanatory.

    (a) Bits/pixel that the destination system expects. This can be calculated from the number of colors on the screen, using the formula

    $$\text{No. of colors} = 2^{(Bits/pixel)}.$$

    Thus, if there are only 2 colors (black and white), there is 1 bit per pixel.

    (b) x size in pixels

    (c) y size in pixels

    (d) RGB bits per pixel

    (e) Black bits per pixel (if image is CMYK)

    (f) No. of frames (must be at least 1)

9. Compression - If "compression" is set to "On", the following three items will be used to execute an external compression program to compress and decompress the image file. This has the advantage that, if new or proprietary compression schemes become available, they can easily be incorporated into custom images. For example, one could use a fractal image compression program if it became available.

    (a) Command - The compression command line. `tnimage` will append the image filename to the end of whatever is entered here. For example, `gzip -f` will cause the image file to be compressed with gzip. The 'f' option is essential to prevent gzip from prompting for a 'yes or no' if the zipped file already exists. `tnimage` already checks for this.

    (b) Extension - The string added by the compression program, if any. If a string is entered, the file must end in that string in order to be correctly identified. (Example: `.gz`).

    (c) Decompress - The decompression command line. The program performing the decompression must send its output to 'stdout' and not to a file. This is accomplished in `gunzip` with the '`-c` ' option. With this feature, it is not necessary to create a temporary unzipped file before reading the image and the original .gz file remains intact.

Warning: Incorrect settings in the format file can crash tnimage.

5.6.1 Custom image format examples

Example 1: Lumisys X-ray scanner - creates 12 bit unpacked grayscale images with a fixed 2048-byte header, and stores data in big-endian format. No sample image file is needed.

1. Select "File...Create file format".

2. Set the following settings in the dialog box:

Target platform	Mac
Bit packing	None
Format Identification	By Extension
Compression	Off
Identifier	lum
Bytes to skip	2048
Use header	(not checked)
Header file	*(leave blank)*
Header bytes	0
Default bits/pixel	16
Default Color type	Grayscale
Default x size	128
Default y size	128
Default RGB bits/pixel	0 0 0
Default black bits/pixel	0
Default no. of frames	1
Compression command	*(leave blank)*
Compression extension	*(leave blank)*
Decompression command	*(leave blank)*

3. Set the following file offsets:

x size	806
y size	808
Bits/pixel	810
All others	65535

4. Then click on “OK”. The new file format will be created.

5. Reading and creating images in the new format will now occur transparently.

“Bytes to skip” must be a multiple of 2 in this case, because there are 2 bytes (16 bits) per pixel. Skipping an odd number of bytes will cause the image will look strange because the light and dark areas are partially reversed.

Similarly, if the wrong target platform is selected, the image could look posterized or grainy, because the bytes are being put into the wrong pixels.

Here is the resulting $HOME/.tnimage/formats/lum file:

```
# Format descriptor for lum image format
# Created by tnimage 2.7.0 on Jun 24 1998
identifier lum
headerfile
compression
decompression
compression_extension
platform 20
packing 0
useheader 0
```

```
useidentifier 0
headerbytes 0
skipbytes 2048
compressflag 0
defaultbpp 8
defaultcolortype 0
defaultxsize 128
defaultysize 128
defaultrbpp 0
defaultgbpp 0
defaultbbpp 0
defaultkbpp 0
defaultframes 1
# Offsets into the uncompressed file:
identifieroff 65535
xoff 806
yoff 808
bppoff 810
colortypeoff 65535
compressflagoff 65535
redbppoff 65535
greenbppoff 65535
bluebppoff 65535
blackbppoff 65535
xposoff 65535
yposoff 65535
framesoff 65535
fpsoff 65535
chromakeyoff 65535
chromakey_graymin_off 65535
chromakey_graymax_off 65535
chromakey_rgbmin_off 65535
chromakey_rgbmax_off 65535
paletteoff 65535
imageoff 65535
```

IMPORTANT: offsets must be set to 65535 unless they are definitely known to be in the image file.

Offsets are not arbitrary locations, and must correspond to the exact position in the file at which the parameter is located. Even when you are creating a new file format, some care must be taken to ensure that offsets do not overlap with each other or with image data. For an existing image in an unknown image format, it is often necessary to find an image of a known size and examine the image file with a hex editor to find out the locations at which the parameters are stored.

If an offset is set to 65535, it will be ignored and a default value will be used instead. Some of these default values are included in the format descriptor file. For example, if `bppoff` is set to 65535, the value in `defaultbpp` will be used. If `bppoff` is any other value, `defaultbpp` will be ignored and the value will be read from the image at the specified offset.

Example 2: Creating an IMDS file. IMDS is a little-known and rarely used monochrome format similar to GEM's IMG format, except that IMDS was used on IBM mainframes running MVS, and there seems to be no documentation for it.

1. Select "File...Create file format".

2. Set the following settings in the dialog box:

Target platform	MVS
Bit packing	TIF-like
Format Identification	By Extension
Extension	IMG
Identifier	(none)
Bytes to skip	0
Use header	$\surd$(checked)
Header file	*(Name of a sample IMDS file)*
Header bytes	32
Default bits/pixel	1
Default No.of colors	1

3. Set the following file offsets:

x size	12
y size	14
All others	65535

4. Then click on "OK". The new file format has been created.

5. Click on an image, or select something on the screen, and save it in the new IMG format (by clicking on "Format" then the last "IMG" item. The sample IMDS file must always be present to create image files in this format.

6. Test to make sure the image is readable. Since tnimage can already read this format, simply select "File...Read" and press *Enter* .

The default bits/pixel and default no.of colors are needed only if the these values are not in the header.

Example 3: Define a file format for multi-frame (3D) images.

- Select "File...Create file format".
- Set the following settings in the dialog box:

Target platform	PC
Bit packing	None
Format Identification	By Identifier
Compression	On
Identifier	3d
Bytes to skip	0
Use header	(unchecked)
Header file	(none)
Header bytes	0
Default bits/pixel	8
Default color type	Indexed
Default x size	128
Default y size	128
Default RGB bpp	0 0 0
Default black bpp	0
Default no. of frames	1
Compression command	gzip -f
Compression extension	.gz
Decompression command	gunzip -c

identifier offset	0
x offset	4
y offset	8
bpp offset	12
colortype offset	16
compressflag offset	20
redbpp offset	24
greenbpp offset	28
bluebpp offset	32
blackbpp offset	36
xpos offset	40
ypos offset	44
frames offset	48
fps offset	52
chromakey offset	56
chr.gray min offset	60
chr.gray max offset	64
chr. RGB min offset	68
chr. RGB max offset	72
palette offset	76
image offset	844

- Then click on “OK”. The new file format has been created.

Here is the resulting $HOME/.tnimage/formats/3d file:

```
# Format descriptor for 3d image format
# Created by ./tnimage 2.7.0 on Jun 24 1998
identifier 3d
headerfile
compression gzip -f
decompression gunzip -c
compression_extension .gz
platform 21
packing 1
useheader 0
useidentifier 1
headerbytes 0
skipbytes 0
compressflag 1
defaultbpp 8
defaultcolortype 0
defaultxsize 128
defaultysize 128
defaultrbpp 0
defaultgbpp 0
defaultbbpp 0
defaultkbpp 0
defaultframes 1
# Offsets into the uncompressed file:
identifieroff 0
xoff 4
yoff 8
bppoff 12
colortypeoff 16
compressflagoff 20
redbppoff 24
greenbppoff 28
bluebppoff 32
blackbppoff 36
xposoff 40
yposoff 44
framesoff 48
fpsoff 52
chromakeyoff 56
chromakey_graymin_off 60
chromakey_graymax_off 64
chromakey_rgbmin_off 68
chromakey_rgbmax_off 72
paletteoff 76
imageoff 844
```

Close image Removes the currently-selected image from the screen and frees up its memory. Also erases the image's backup and its FFT if present.

DOS Command Shell to DOS to execute a command or run a different program. Or, to obtain a DOS command line, enter `Command`. Type `Exit` to return to tnimage. DOS requires a small amount of memory below 1 megabyte to exit commands. Thus an "insufficient memory" error could occur even if there is a lot of high memory.

This function is, of course, not available in the Unix version.

Quit

Return to operating system. Alt-X also can be used to quit.

5.7 Scanner

(Unix version only) Acquires an image from a H/P compatible scanner. HP-compatible SCSI scanners are supported directly. Other scanners are supported with external drivers, using a driver configuration file (See below).

Note: In the past, Hewlett-Packard distributed a Scanner Control Language Toolkit which provided documentation on their scanners to developers. Hewlett-Packard no longer provides this information. This makes it difficult to support HP scanners in Linux, and consequently these devices are no longer recommended.

Setting up a scanner

Here is a brief outline of the procedure for attaching a scanner in Linux on an x86 system. For more details, consult the *Linux SCSI-HOWTO*. For Sun and SGI systems, these steps should not be necessary. Convex does not seem to support a SCSI interface.

1. Install a supported SCSI-2 card and H/P scanner. One combination known to work is an Adaptec AHA-1542CF card and H/P 4C scanner.
2. Throw away the cheap printer-type cable and the SCSI card supplied with the scanner and use a good quality SCSI cable (Centronics-type connector at both ends) and a good SCSI card to connect the scanner.
3. Recompile your kernel if necessary to add "generic SCSI" support and support for the low-level driver for your SCSI card (see the *Linux Kernel-HOWTO*).
4. Shut down the system and make sure the scanner and the SCSI card are both properly terminated. Set the SCSI ID of the scanner to some known value less than 7. Power up the scanner before turning the computer back on. The scanner must be connected and powered on before booting the computer in order to be recognized.
5. The SCSI card should display a message at boot-up indicating its presence. Most Adaptec cards can be configured for IRQ, SCSI ID, etc. at this point by typing Ctrl-A during boot-up. Using the card's DIP switches to turn off the card's BIOS may also help.
6. At boot-up, if the SCSI card is recognized, you will see a message such as:

   ```
   kernel:  scsi0 :  Adaptec 1542
   kernel:  scsi :  1 host.
   ```

   If the scanner is hooked up correctly, you will also see a message such as:

   ```
   kernel:  scsi:  type is processor
   kernel:  Vendor:  HP Model:  C2520A Rev:  3503
   kernel:  Type:  Processor ANSI SCSI revision:  02
   ```

```
kernel:  Detected scsi generic sga at scsi0, channel 0, id 5,
lun 0
kernel:  scsi :  detected total.
```

7. If this message does not appear, the most likely cause is that the scanner is not terminated or an incorrect cable was used. Remember the device name (in this case "sga"). It may be helpful to create a link to this device named "scanner" in /dev.

8. As a last check, " `cat /proc/devices` " should list " `21 sg` " as one of the character devices. This device will be listed only if the scanner device is detected at boot-up. If it is not listed, it may be necessary to create a device file using MAKEDEV and then change the permissions on the device file. It should look something like this:

   `crwxrwxrwx 1 root root 21, 0 Feb 9 17:30 /dev/scanner`

   You can also create links or additional /dev files to describe the same device, for example:

   crwxrwxrwx 1 tjnelson users 21, 0 Oct 3 1998 /dev/scanner
   crwxrwxrwx 1 root disk 21, 0 Jan 19 1999 /dev/sg0
   lrwxrwxrwx 1 root root 3 Oct 3 1998 /dev/sga →sg0

   The important thing is the 21,0. If you change it in the scan dialog, it is remembered next time tnimage is started.

Scanning an image

1. Select "File...scanner.." and change the scanner device name, resolution, brightness, contrast, etc. as desired. Click the "Preview Scan" button then click OK. The scanner should immediately begin to scan a small, temporary, preview image. (The preview image is scanned at 50 dpi).

2. When the preview image is finished scanning, the dialog box will appear again. Change the contrast, brightness, resolution, etc. if necessary and click on "Image scan" then "OK". Although it is possible to enter any number for resolution, some scanners may only accept specific values, such as 50, 75, 100, 150, 300, 600, or 1200 dpi. Entering a resolution inappropriate for a given scanner could result in the parameter being ignored by the scanner.

3. After you click OK, a message box will appear reminding you to select the scan coordinates for the Image Scan. Using the mouse, select a region in the preview image. During this phase, the mouse is constrained within the preview image to prevent scanning of impossible coordinates. The selected region will be outlined with a "crawling box" and, when the mouse button is released, scanned at the selected resolution and placed in a permanent image buffer. Additional regions may be selected and scanned as many times as desired to create more images.

   Note: Occasionally, a crawling box will already be present at this point. It is still necessary to select a region to scan, even if the crawling box appears to be in the correct position.

4. To scan a new original, click "Preview" again to create a new preview image, or click Cancel if finished. The temporary image is automatically erased.

   NOTE: Scanner support has not been tested in the Irix or Solaris versions. Information as to whether this works will be appreciated. The scanner interface is not supported in MS-DOS or ConvexOS.

   WARNING: If tnimage detects an unsupported type scanner, it displays the message:

```
Not an HP scanner
Continuing may cause a lockup
Do you want to continue?
```

Click “OK” to attempt to use the scanner. *Do this at your own risk. This will most likely hang the SCSI bus.*

Notes:

- ADF, transparency adapters, and the ScanJet Button Manager on the HP5c are not currently supported.
- Not all scanners support all scanning modes. For example, the HP4c handles 8 and 10-bit B/W and 24 and 30-bit color, but not 12 bit B/W or 36 bit color.
- No scanners can automatically create an 8-bit color image. To obtain 8-bit color images, first scan the image at 24 bits, then use the “Change Color Depth” feature to reduce the image to 8 bits. This will automatically generate an optimal colormap.
- 30- and 36-bit/pixel images are stored internally as 48 bits/pixel but currently are handled in a pseudo-24 bits/pixel format with the low bits discarded. These images must be converted to 24 bits/pixel before they can be saved. A future version of tnimage will correctly handle 48 bit/pixel images.
- Brightness and contrast settings have no effect on 10 bit/pixel images.
- For images scanned at 10, 12, 30, or 36 bits/pixel, it is necessary to maximize contrast before changing to a different pixel depth. For example, a 10 bit image has pixel values between 0 to 1024, but it is stored in a 16 bit buffer. The contrast must be expanded to 16 bits, otherwise it will appear black when the values are scaled to 8 bits. (Select “Color..Contrast” and click on “Maximize Value”.)

Scanner configuration for non-HP scanners

In addition to HP scanners, any other scanner can be used provided that a suitable driver is installed. Below is the procedure for installing and configuring scanner drivers, using the umax-cli driver written by Oliver Rauch and Michael K. Johnson as an example. These steps should only be necessary for non-HP scanners, as HP SCSI scanners are supported internally by `tnimage`.

WARNING Read the following before creating a scanner driver configuration file. Using an incorrect file could crash `tnimage` or even hang the SCSI bus.

1. Install the scanner driver. (The umax driver is located at

   ftp://tsx-11.mit.edu/pub/linux/ALPHA/scanner).
2. Ensure that the environment variable $HOME (or $home if you are using csh or tcsh) is defined.
3. Create the directory in `$HOME/.tnimage` if necessary and ensure that you have read/write permission in that directory. (These steps should already be done by the Install script).
4. Create a file in `$HOME/.tnimage` named “`scanners`”. This file is a list of configuration files you will create, one filename per line. For example:

   ```
   umax
   mustek
   my_other_scanner
   ```

5. Determine the command line options for the scanner driver you just installed by consulting the documentation or man page. Take note of what options set the dpi, top, left, width, height, and bits/pixel, and determine whether it creates a file or sends its output to `stdout` (i.e., to the screen). Note that the options may be case-sensitive, and may or may not require a space between the option and its value. These options may vary considerably from the example shown below.

6. Once the command line options are known, test the scanner driver to make sure it is working. The term "scanner driver" is used here to mean a command line program that controls the scanner. To be compatible with `tnimage,` the driver must have the following characteristics:

   (a) It must be an executable command-line program in your path or in some known location.
   (b) Its permissions must be such that you and other users can execute it.
   (c) It must take its scanning options from the command line.
   (d) It must be able to create an image file in some valid image format.
   (e) It must either write to `stdout`, or create a file with a specified filename.

   Use the following procedure to test the scanner driver:

   (a) Type a command line appropriate for the scanner driver, e.g., `mydriver {options} > testfile` .
   (b) Check to make sure `tnimage` can open the resulting file `testfile` as a valid image.
   (c) If so, it should be possible to create a driver config file that will work.

7. Using the supplied `umax` file as an example, create a configuration file (named, in this case, `umax`) to indicate to `tnimage` the command-line options needed by the driver, and its location. Here is a minimal example configuration file with the umax parameters added:

```
name  Umax
driver_path $HOME/scanner/umax/umax-0.5.6/umax-cli
dpi_command    -R
left_command   -X
top_command    -Y
width_command  -w
height_command -h
```

   NOTE: Each scanner driver has different command-line options. For example, the "-R" command sets the dpi for `umax-cli` but other scanner drivers may use a completely different command line option.

   The "name" can be any name, which will appear in the menu. Tnimage will automatically supply appropriate numerical values for the other options based on the settings you provide interactively in the scanner dialog. Lines with no option or lines starting with a '#' are ignored. Not all options are meaningful for all drivers.

8. Make sure you have write permission in the directory `$HOME/.tnimage` before starting a scan. This is where `tnimage` will create its named pipe, "fifi".

9. Test the configuration file by starting `tnimage` in diagnostic mode (i.e., `tnimage -diag > outfile`). The outfile will contain a copy of the command line composed by `tnimage` . Check this command line to make sure it is correct.

   For example, here is part of the output from `tnimage` in diagnose mode when using a Umax scanner:

```
...
Scanner command:
$HOME/scanner/umax/umax-0.5.6/umax-cli -R 50 -X 0 -Y 0 -w 2550
    -h 3300 -b 127 -c 127 -r >/home/tjnelson/.tnimage/fifi
Reading image /home/tjnelson/.tnimage/fifi
...
```

10. Once the configuration file is working properly, copy it to `$HOME/.tnimage` (or `$home/.tnimage` if you are using csh or tcsh).

11. If you wish, send me a copy of the config file so it can be included with future versions of `tnimage` for the benefit of other users.

The scanner driver will now function as an integral part of tnimage. The configuration file (shown here with commands for the Umax driver) may also contain other options including:

`brightness_command -b`
`contrast_command -c`
`bpp_command` *Set bits/pixel*
`gray_8_command` *Scan 8 bits/pixel grayscale*
`gray_10_command` *Scan 10 bits/pixel grayscale*
`gray_12_command` *Scan 12 bits/pixel grayscale*
`gray_16_command` *Scan 16 bits/pixel grayscale*
`color_24_command` *Scan 24 bits/pixel color*
`color_30_command` *Scan 30 bits/pixel color*
`color_36_command` *Scan 36 bits/pixel color*
`gray_command -g` *Scan in default grayscale mode*
`color_command -r` *Scan in default color mode*
`preview_command -p`
`extra_commands` Any additional command options entered here will be sent verbatim to the scanner. For example, -s = slow speed, -W = warm up lamp before scanning. If all else fails, this line can be used alone to send a fixed command line to the driver.

Conversion factors may also be added in case scanner driver uses some other units besides pixels at 300 pt/inch. For example, if driver requires all units to be in inches, set each value to 72. If the driver uses 1200 pt/inch instead of 300, set each value to 4. Values do not have to be integers.

```
height_factor 1
width_factor 1
top_factor 4
left_factor 4
```

Numbers may need to be added to the contrast and brightness values supplied by tnimage (which range from 0 to 255) to convert to the range used by the scanner driver. For instance, if the driver uses -127 to +127 instead of 0 to 255, enter -127 here.

```
contrast_add 0
brightness_add 0
```

5.8 Camera Interface

(Unix version only) Acquires an image from a camera, with a user-supplied driver. For example, images can be acquired from the Connectix QuickCam 2 using the program cqcam, available from: `www.cs.virginia.edu/~patrick/quickcam`

A driver is a command-line program that writes a file to standard output in PNM format. Tnimage controls the driver using the driver's command line parameters, which are stored in a configuration file when the camera is given the "initialize" command.

Below is the procedure for configuring a new camera interface, if it has not already been set up automatically by the Install program:

1. Add the name of a configuration file you will create named, for example, "mycamera" to `$HOME/.tnimage/cameras` on a new line. Note that the same driver can appear a number of times, each referring to a different configuration file with different default parameters.

   For example:

   ```
   quickcam
   mycamera
   my_other_camera
   mycamera-at-320x200
   ```

2. Create the "mycamera" configuration file in the directory `$HOME/.tnimage` containing the following lines, which describe the command-line options required by the driver (without numerical values). Below is a typical example, with quickcam options added:

`driver /home/you/camera/cqcam`	Path for driver binary
`defaults /home/you/.cqcrc`	Path for driver's configuration file (see below)
`other -32+`	Any other command line options.
`scale -s`	Driver's command to set scale
`width -x`	Driver's command to set width.
`height -y`	Driver's command to set height.
`brightness -b`	Driver's command to set brightness.
`black -B`	Driver's command to set black level
`white -w`	Driver's command to set white balance
`hue -H`	Driver's command to set hue (blue level).
`saturation -S`	Driver's command to set saturation
`contrast -c`	Driver's command to set contrast.
`initialize -a+ -r`	Driver's command(s) to initialize the camera.
`normal -a-`	Command(s) to acquire an image normally.
`extra`	Any other command line options.

   Anything entered after "`other`" will be placed immediately after the driver name on the command line. In this example, `-32+` was added as a sample "other" command line option (this causes the driver to produce 32-bit/pixel output).

   The "initialize" command should also include any options for automatically saving the driver's new configuration in its config file (`-r` in this case).

   Anything entered after "`extra`" will be placed at the end of the command line. In this example, no extra commands will be sent.

Change the command-line options as needed for your driver. The numerical values for the above parameters will be supplied by tnimage.

Any of the above options may be omitted if desired.

NOTE: Each camera driver has different command-line options. For example, the "-s" command sets the scale for `cqcam` but other camera drivers may use a completely different command line option. Consult the driver documentation before creating a config file.

Next, add the following default parameters to the "mycamera" configuration file, along with the desired numbers. Change the width and height to match the image size produced by the camera:

`#default parameters`	A comment (begins with #)
`default_shell 1`	1= put camera image in a separate, positionable window.
`default_width 640`	Width in pixels of image created by camera.
`default_height 480`	Height in pixels of image created by camera.
`default_scale 1`	Starting scale value for camera.
`default_depth 3`	Bytes/pixel of image produced by camera.
`default_post_command`	Post-processing command

If "default_shell" is 0, the camera image will appear on the main tnimage window; otherwise, it will be placed in its own separate, positionable frame.

The default_post_command can be any `tnimage` command or macro. If the box beside the command is checked, this command will be executed as a macro after every frame. For example, the line

```
default_post_command filter(1,1,10);
```

Will sharpen the image by 10% using a 3x3 sharpening filter after each frame. See Sec. 13 for details on macros.

A common use of post-processing is to adjust the image color. This can be done by entering a mathematical formula (See "image algebra", Sec. 12), for example,

`r+=20;`

to increase the brightness of the red component by 20 (on a scale of -255 to 255). For simple commands such as color or brightness adjustment, it is usually much faster to use the corresponding macro command, i.e.:

`color(0, 20, 0, 0);`

which will do the same thing, nearly instantaneously.

Note that the post-processing command is not executed unless the box next to it is checked.

3. (Optional) If the camera driver does not create its own defaults file, create another text file (named, for example, '.cqcrc') describing the starting default values, containing the following lines:

`contrast 104`	Contrast
`white 100`	White balance
`blacklevel 100`	Black level
`hue 110`	Hue (blue level)

Put the name and path of this .cqcrc file after "defaults" in the first ('mycamera') file. These parameters will be the starting default values for the camera. (**Note:** cqcam produces a .cqcrc file automatically). During initialization, the dialog will automatically be updated to reflect the new settings in the defaults file.

As an example, here is the `$HOME/.tnimage/cameras` file on my system:

```
quickcam
```

Here is the `$HOME/.tnimage/quickcam` file on my system:

```
# tnimage configuration file for connectix quickcam
# order of parameters may be important

# command-line options for camera driver
driver /home/tjnelson/camera/cqcam-0.45a/cqcam
defaults /home/tjnelson/.cqcrc
other -32+
scale -s
width -x
height -y
brightness -b
black -B
white -w
hue -H
saturation -S
contrast -c
initialize -a+ -r
normal -a-

# default parameters
# 0=put camera image on main window, 1=put in separate window
default_shell 1

# width and height of image produced by camera
default_width 640
default_height 480

# scale of image as defined by camera driver
default_scale 1

# depth of camera image, in bytes/pixel
default_depth 3

# default post-processing command or macro name
default_post_command filter 1 1 10
```

Here is the `$HOME/.cqcrc` file on my system, which was created by the driver:

```
brightness      127
```

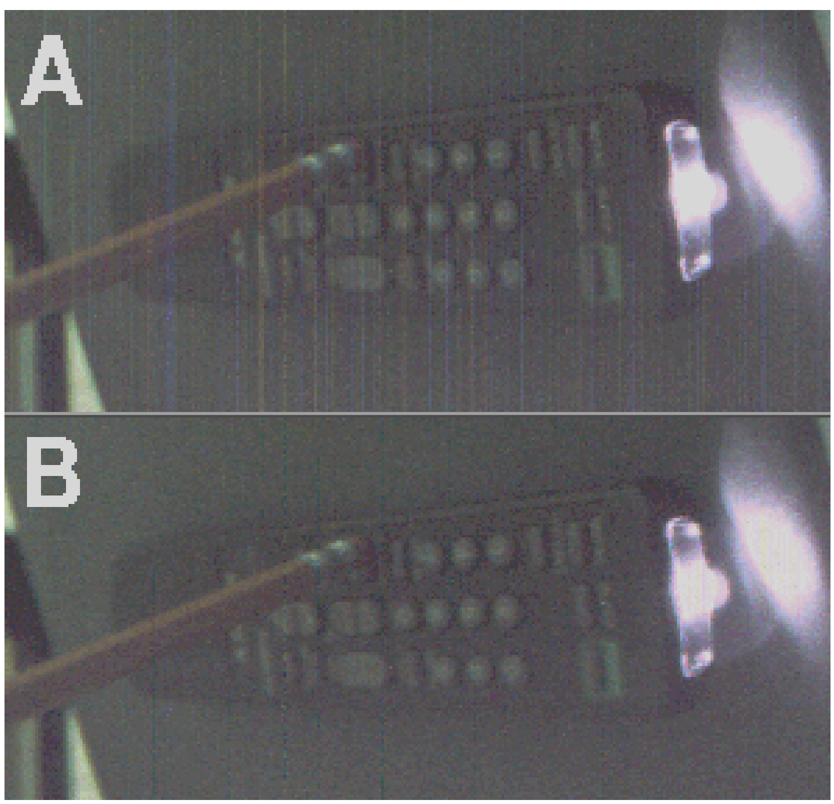
A camera config file and a patch for the bttv driver for the Hauppauge WinTV camera card is included in the file `bttv_kit.tgz`. (This file was contributed by Michael Lapsley (mlapsley@sthelier.sghms.ac.uk)).

Camera Dialog Options

Acquisition mode

1. Normal: Each frame replaces the previous frame.
2. Accumulate frames: Successive camera frames are averaged with all previous frames. This can greatly improve the quality of the image.
3. Add: The new frame is added to the previous frame. This can be used to increase the sensitivity of images collected in low light conditions.
4. Subtract: The new frame is subtracted from the previous frame. Camera images often contain streaks or abnormally bright pixels caused by camera imperfections. The 'subtract' option can be used to subtract these from the image. After acquiring an image, cover the lens (or turn off the light), click on "Acquisition mode.. Subtract", "Camera..normal" and "Frames..Single frame". Then click on "Acquire frame" to collect one frame of dark background pixels. The deviation of each rgb value in the dark image from the average rgb values for each scan line will subtracted from the previous image.

For example, below is a low-resolution image of an infrared TV remote control created using a QuickCam (which is sensitive to infrared as well as visible light). The original top image **A** contains numerous vertical streaks. Subtracting the dark current produced **B**. The vertical streaks are mostly eliminated, although the picture is more grainy. This could have been repaired by accumulating several frames. Darker scenes are more susceptible to dark current artifacts.



Camera

1. Initialize: Sends the "initialization" command to the camera. This can take 10 times as long as acquiring an image. The camera is also initialized automatically before the first

frame. The camera dialog will be automatically updated to reflect the contents of the "defaults" file.

2. Acquire: Obtain frames normally. The menus may become relatively unresponsive while a frame is being acquired.

Frames

1. Continuous: Tnimage continually acquires new images at the specified frame rate. The desired frame rate may not actually be achieved due to limitations in the camera camera driver. The "camera" setting should always be set to "normal" during continuous operation, otherwise the camera will be initialized on each frame which will greatly decrease the frame rate.
2. Single frame: The program waits until the "Acquire frame" button is clicked, then takes a single frame.

When the "OK" button or the main "Cancel" buttons are clicked, image acquisition stops and the dialog box disappears, leaving the camera image that is currently visible as the final image.

Notes:

1. If the camera driver requires root permissions, tnimage must either be set suid or run as root.
2. The image data are sent through a fifo and do not use a temporary file.
3. To stop acquiring frames continuously without dismissing the camera dialog, click on the "Frames..Single frame" button.
4. When acquiring frames continuously, if the dialog buttons become unresponsive, click on the "Cancel" button on the main tnimage window. This will usually stop image acquisition.
5. Do not exit `tnimage` while acquiring an image. This will create a zombie process.
6. With some camera drivers, the driver's config file is not automatically updated by the camera driver. The symptom of this is that, when several frames are collected, the first frame is normal (because initialization of the camera is occurring), but subsequent frames are completely black or gray. The solution is to set the brightness value manually in `tnimage`.

5.9 Creating, executing, and testing plugins

In the event that specialized features not included in tnimage are needed, it is possible to create a "plug-in". Tnimage can send data to an external program and retrieve it after processing.

Running plug-ins

- Create a text file in `$HOME/.tnimage` (or `$home/.tnimage/` if you are using csh or tcsh) named "plugins" containing a list of plugins, each on a separate line. Command-line arguments meaningful to the plugin follow on the same line, and are separated by spaces. For example,

  `plugin`

  `/usr/local/bin/myplugin`

  `./joes_plugin` *[argument1 [argument2]...]*

- Select "File...Execute plugin" and select the desired plugin.

Plugin execution dialog

Creating plug-ins

The plug-in interface is still experimental and may change slightly in future versions.

1. The files `plugin.cc` and `plugin.h` provide a template for creating new plugins. `xmtnimage.h` is also required to compile a plugin. Tnimage passes image and configuration data to the plugin through a stream pipe, then reads the modified data back afterwards. Stream pipes are used because the amount of shared memory in most versions of Unix is quite limited.
2. A message of up to 1024 characters may also be sent back to the parent. No information should be sent to stdout. Error messages can write to stderr if necessary.
3. The new plugin may be compiled with the command line:
   `gcc -o plugin -O3 -Wall plugin.cc` (use `-O2` on Irix).

Plugin programming considerations

1. New images can be created within a plugin by including these 2 lines:

   ```
   newimage(g.image_count,x,y,bitsperpixel,colortype,frames);
   g.image_count++;
   ```

   where `x` × y is the size in pixels for the image, frames is the total number of frames (must be at least 1), bitsperpixel is one of {8,15,16,24,32,48}, g.image_count is the image number to create, and `colortype` is either `GRAY`, `INDEXED`, or `COLOR`. Image numbers must be sequential.
   The image bytes can then be addressed as

```
z[image_no].image[frame][y][x]
```

Pixels are packed in standard format according to image depth:
Indexed color:
8 bits/pixel: One byte = 1 pixel

Grayscale: Rounded up to next highest multiple of 8 bits/pixel, maximum of 32 bits/pixel.

Color:
15 bits/pixel: `0rrrrrgg gggbbbbb`
16 bits/pixel: `rrrrrggg gggbbbbb`
24 bits/pixel: `rrrrrrrr gggggggg bbbbbbbb`
32 bits/pixel: `00000000 rrrrrrrr gggggggg bbbbbbbb`
48 bits/pixel: `00000000 rrrrrrrr 00000000 gggggggg 00000000 bbbbbbbb`

This means that each horizontal line of n pixels in the image can be n, 2× n, 3× n, 4× n, or 6× n bytes long, depending on the image depth. When creating a loop over the pixels, it is necessary to increment by the correct number of bytes in the x direction. For example, the following loop increases the brightness of frame 0 of a 16-bit grayscale image by 100:

```
uchar *address;
int x, y, value;
int bpp = z[ci].bpp;          // bpp is bits per pixel
int step = g.off[z[ci].bpp]; // step is bytes per pixel

for(y=0; y < z[ci].ysize; y++)
for(x=0; x < step*z[ci].xsize; x+=step)
{    address = z[ci].image[0][y][x];

     // Read pixel value in depth-independent manner
     value = 100 + pixelat(address, bpp);

     // Make sure value doesn't exceed maximum permitted value
     // for the given image depth.
     value = min(value, (int)g.maxvalue[bpp]);

     // Put pixel back into image
     putpixelbytes(address, value, 1, bpp, 1);

}
```

For color images, the red, green, and blue components must be extracted from the pixel and processed separately. To determine whether the image is color, check `z[ci].colortype` which can be one of `GRAY`, `INDEXED`, `COLOR`, or `GRAPH`.
The function `uint RGBvalue(int red, int green, int blue, int bpp)` takes the red, green, and blue components and converts them into a composite value that can be stored into the array.
The function `void valuetoRGB(uint pix, int &rr, int &gg, int &bb, int bpp )` does the reverse, putting values for red, green, and blue in the parameters rr, gg, and bb that are appropriate for the given bits/pixel. The parameters rr, gg, and bb are passed by reference. If you don't like this, you can easily change it to pointers in your plugin.
The functions `pixelat()`, `RGBvalue()`, `valuetoRGB`, and `putpixelbytes()` can be copied to your plugin from `xmtnimage42.cc` .
There are a number of other parameters in the Image struct that may prove useful in plugins – see `xmtnimage.h` for a description. Some of these, such as `xsize` and

`ysize`, must correspond to the dimensions of the image array and should not be changed arbitrarily.
The parent program will take care of scaling and redrawing the image when the plugin finishes, provided that `g.changed[ci]` is set to 1.
All plugins must be recompiled if you change to a new version of `tnimage`.

2. For new 8-bit images, it is recommended to also set an appropriate colormap, by the following:

```
for(k=0;k<256;k++)
{   z[ino].palette[k].red = k/4;
    z[ino].palette[k].green = k/4;
    z[ino].palette[k].blue = k/4;
}
memcpy(z[ino].opalette, z[ino].palette,768);
memcpy(z[ino].spalette, z[ino].palette,768);
```

(where `ino` is the image number). Each image has 3 copies of the colormap, to facilitate undo operations.

3. The following variables should be set to ensure correct handling of the new image once control returns to tnimage:

```
g.changed[ino] = 1;
z[ino].touched = 1;
```

and the image should be given a default title:
`strcpy(z[ino].name, "New_title");` *(or other title).*

4. The user can enter up to 20 arguments to each plugin. Four additional command-line arguments defining user-selected upper left and lower right coordinates are then passed to the plugin, in the following format:
   (a) `arg[0]` plugin name
   (b) `arg[1]` to `arg[n]` user-supplied command-line arguments entered in the "Execute plugin" dialog. These arguments are automatically updated in the file `$HOME/.tnimage/plugins`, whenever they are changed.
   (c) Left coordinate of current image or currently-selected area.
   (d) Top coordinate of current image or currently-selected area.
   (e) Right coordinate of current image or currently-selected area.
   (f) Bottom coordinate of current image or currently-selected area.

   This is followed by a null character and the remaining configuration information, followed by the image data.
   This arrangement allows any program that takes command-line arguments and writes to `stdout` to be easily repackaged as a plugin, by modifying the skeleton program `plugin.cc`. This could be used, for example, in plugins that can read new file formats or for controlling a scanner. The stream pipe mechanism used in tnimage is very fast, and does not require an intricate interface with shared memory or error-prone function calls to internal routines in tnimage.
   Another advantage is that an unlimited amount of data can be sent to the plugin or returned to tnimage. If additional parameters, such as user-selected filenames using the graphical file selector, are desired, the plugin can be incorporated into a tnimage macro.

5. A message can be displayed upon completion by setting `have_message` to 1 and copying the message string into `display_message`, *e.g.*

```
strcpy(display_message, "Your error message\n\
up to 1024 characters here");
```

6. Some parameters of existing images, including size and bits/pixel, must not be altered by a plugin.
7. The plugin also must not write to `stdout`, but can open files, write to `stderr`, and make system calls as needed.
8. It is also possible to add a custom button on the left information area for your new plugin. This is easier than selecting it from the menu each time (See Sec.4.3).

Warning: Incorrectly setting parameters in the Image struct can cause your plugin to crash, or crash tnimage.

Notes on plugins:

1. Writing to stdout in a plugin can cause the plugin and/or `tnimage` to hang.
2. Plugins must be recompiled and reinstalled whenever a new point version of `tnimage` is installed.
3. Check the file `plugin.cc` for any last-minute changes to the interface.
4. In ver. 3.1.0, the interface was changed to accommodate creating Fourier-transforms in a plugin. All plugins should change the `write_data()` function as indicated in `plugin.cc` .
5. Users are encouraged to send new plugins to the `tnimage` site so they can be incorporated into the `tnimage` distribution and benefit other users.

Example - scanner plugin

A typical use for a plugin would be to control a new scanner. This example demonstrates the preferred method for setting up a scanner plugin for interactive scanning in preview and image scan mode.

Assume the plugin is named "`myscan`" and that the plugin was written to take command-line arguments as follows:

1. 0 = preview mode, 1 = image mode
2. Scan resolution in DPI
3. Left x coordinate to begin scan
4. Top y coordinate to begin scan
5. Right x coordinate to end scan
6. Lower y coordinate to end scan

The scanner plugin could be incorporated into tnimage by the following procedure:

1. Edit `$HOME/.tnimage/tnimage.ini` and change `nbuttons 16` to `nbuttons 18`. This will add 2 new buttons on the left side of the tnimage screen.

2. After the last `button` entry in `tnimage.ini`, add the line `button Preview executeplugin myplugin 0 75 0 0 640 825` . This button will scan a preview image and place the result on the main window, or, if tnimage is configured to use separate windows for each image, it will open a new window of 640x825 pixels and place the preview scan in it. (Ideally, when executed in preview mode, the plugin would send back the message, "Preview completed, now select area to scan with mouse" or some similar message. See *Plugin programming considerations* above.)

It is also possible to instead use the line `button Preview macro` *pluginmacro*. This macro could contain the lines

`windows(1);` *(= Sets separate-window mode on)*
`executeplugin(plugin,0,75,0,0,640,825);` *(= executes plugin)*
`windows(0);` *(= Sets separate-window mode off)*

3. After the Preview button in `tnimage.ini`, add the line `button Myscan executeplugin myplugin 1 300`. This new button will execute the plugin in image mode. The scan start and end coordinates are left blank. Tnimage automatically sends the coordinates of the currently-selected area immediately after the specified command-line arguments. Thus, if the user has selected the region (100,100) to (200,200), when the "Myscan" button is clicked the plugin will automatically be executed with these arguments: `myplugin,1,300,100,100,200,200;`
4. Add the line "`myplugin`" to the file `$HOME/.tnimage/plugins`, which is a simple list of plugins (along with any command-line arguments).
5. Finally, put the plugin in the same directory as `tnimage` . If this is not possible, then change the occurrences of "myplugin" above to include the actual path.

The functionality of the scanner plugin will now be transparent to the user. If the user clicks the `Preview` button, the plugin is automatically executed, scans an image at 75 dpi, and the preview image appears in the main tnimage window. The user then uses the mouse to select the desired scan region and clicks the `Myscan` button. This executes the plugin again, which scans the desired area at 300 dpi.

Here is a concrete example of a minimal plugin that takes the current image (`ci`), and subtracts its pixel values from 255. If the image was an 8 bit grayscale image, this would create a photographic negative of the image.

```
int f,i,j, bytesperline;
//// Calculate number of bytes in a scan line based on bits/pixel
bytesperline = z[ci].xsize * g->off[z[ci].bpp];
for(f=0; f<z[ci].frames; f++)
for(j=0; j<z[ci].ysize; j++)
for(i=0; i<bytesperline; i++)
     z[ci].image[f][j][i] = 255 - z[ci].image[f][j][i];
//// Set g->changed so parent redraws the image
g->changed[ci] = 1;
//// Set 'touched' so user is prompted to save image before quitting
z[ci].touched = 1;
//// Change the title of the image
strcpy(z[ci].name, "Modified");
```

The actual mechanics of transferring the data to and from the parent are handled by the skeleton program, `plugin.cc`, in which this code would be placed.

Testing plug-ins

To debug a plug-in, create a macro containing the following line:

`executeplugin(` *plugin_name* `, 0)`

or equivalently,

`testplugin(` *plugin_name*)

The '0' causes the plugin to be executed in 'debug mode', i.e., its stdout is not redirected, whereas '1' causes it to be executed normally.

Note: It is advisable not to move the mouse, open other menus, or execute other commands while the plugin is executing.

Supplied plugins

'Plugin'

This is an example plugin that simply creates a small test image.

'Readtif'

This plugin uses the Leffler libtiff library from `ftp.sgi.com` . It was not feasible to use the libtiff routines in `tnimage`, because the libtiff library does not support many of the TIFF subtypes commonly found in the laboratory, including 16 bit grayscale, 15 and 16 bit color, and cannot read Image Quant files. Some well-known image viewers incorporating libtiff will even crash when trying to read such files. On the other hand, some TIFF subtypes not supported by `tnimage` (such as Fax images) may be readable by this plugin. Also note that `readtif` expands all images to 32 bits/pixel. The user should down-convert the image to the desired pixel depth before saving it. Enterprising users who strongly prefer `libtiff` (you know who you are) can even uncomment the libtiff code in `xmtnimage11.cc`. The include file (`tiffio.h`) must be present on your system.

'Readhdf'

Readhdf reads images in NCSA's HDF format. The HDF format is extremely complex and can contain a variety of data types along with or instead of images. HDF files not containing images are not readable by `readhdf`. Users are encouraged to adapt the `readhdf` code to read other data types. Feel free to send any working plugins to the tnimage site (by email only - the `incoming` directory has been closed). If your plugin is useful, it may be of great benefit to some other user.

Compiling plugins

1. Make sure `config.h` and `tnimage.h` are present. If you did not compile `tnimage` yourself, copy `config.h.example` to `config.h` and edit as needed for your system.
2. Compile the plugin using the command:

   `gcc -O2 -o plugin plugin.cc`
3. Copy the plugin to `/usr/local/bin` .
4. Add the plugin command line to `/usr/local/lib/tnimage/plugins` .

Compiling and installing the HDF image format plugin

1. Install the HDF software as instructed in the INSTALL file in the HDF distribution. The most recent version of the distribution can be obtained from the NCSA ftp archive site at:

   `ftp://ftp.ncsa.uiuc.edu/HDF/HDF_Current`

   A Fortran compiler is not necessary to compile this library provided that the defaults in the `Makefile.in` and `Makefile` files in the distribution are changed to `FC = NONE`.
2. Copy the files `complex.h, xmtnimage.h, xmtnimageb.h, plugin.h,` and `readhdf.cc` to a directory containing the HDF libraries `libdf.a, libjpeg.a,` and `libmfhdf.a`.
3. Edit the readhdf.cc plugin if desired and compile it with the command:

   `gcc -o readhdf readhdf.cc libdf.a libmfhdf.a libjpeg.a libz.a -I../src` changing the last parameter if necessary to reflect the location of the HDF include files (hdf.h *et al.*).

4. Move the plugin `readhdf` to a convenient location and edit the file `/usr/local/lib/tnimage/plugins` (which is a simple list of available plugins, one entry per line) to indicate the location of the plugin.

Section 6

Image menu

6.0.1 Delete region

Erase part or all of an image or background. After selecting 'delete region', move to one corner of the region to erase, and click and drag to the other corner. This rectangular region will be set to the background color. (If there is another image behind the topmost image, the obscured part will be unaffected.)

Pressing the `Del` key after selecting a region has the same effect.

6.0.2 Crop

`Crop` surrounds the image with a crawling dashed line. Move the mouse cursor to the middle of the dashed line. The cursor will change to an edge-arrow shape. Click and drag on the dashed line to select the desired region of the image, then press the space bar or any other key. The image will be cropped and the region outside the box will be removed. Pressing the Escape key will abort cropping.

If part of the image is off the edge of the screen, use Shift+arrow (hold down the left or right Shift key and press the arrows on the numeric keypad).

- Shift-0 (Ins) - Makes selected region smaller
- Shift-1 (End) - Moves top border down
- Shift-2 (Down) - Moves bottom border down
- Shift-3 (PgDn) - Moves left border to the right
- Shift-4 (Left) - Moves left border to the left
- Shift-5 - Makes selected region bigger
- Shift-6 (Right) - Moves right border to the right
- Shift-7 (Home) - Moves right border to the left
- Shift-8 (Up) - Moves top border up
- Shift-9 (PgUp) - Moves bottom border up

The amount of movement is determined by the Step size. This can be adjusted by pressing the gray (+) or (-) keys. The default step size is 8 pixels.

6.0.3 Erase background

Removes everything on the screen that is not a part of an image, and sets it to the background color.

6.0.4 Copy/Move

Copies or moves part of an image or drawing to another region. Before selecting 'copy' or 'move', click and drag on the image to select a region to copy; or double-click to make tnimage automatically find the edges of the object to copy.

The region is copied and pasted in the new location as soon as the mouse button is released. If the pasted portion falls partly on an image and partly on the background, the portion that falls on the image sticks to the image and the part that falls on the background sticks to the background.

If the "move" button in the information window is depressed, the pixels will be moved instead of copied.

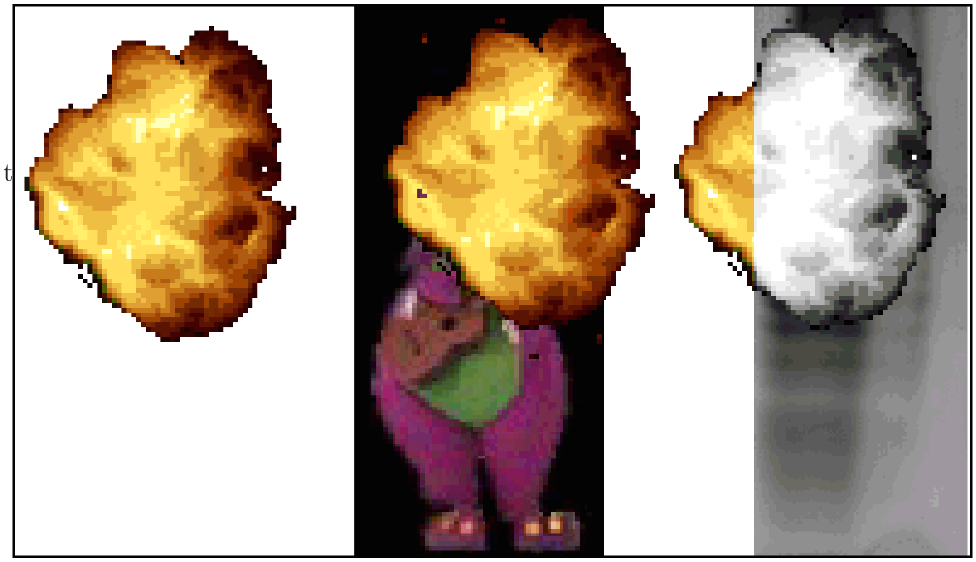
The mode of interaction of the moved region with the background or other images can be changed by changing the "Pixel interaction mode" from the "Config" menu. For example, if the pixel interaction mode is "subtract" the moved region will be subtracted from whatever is there. (See *Pixel interaction mode* below).

The source and destination do not have to be in the same window. To copy or move an area from the main window to an image in another window, continue pressing the mouse button and move the mouse cursor to the new window. The copied area will be invisible while the cursor is outside one of tnimage's windows and will reappear when the cursor moves to the new window. Copying to other programs is not yet supported.

The colormap of the destination image is considered inviolate during a copy. The reason for this is that changing the destination colormap would require remapping the preexisting image, and after several copy operations it could eventually get corrupted, that is, the original correspondence between intensity and pixel value would be lost.

Therefore, the copied region is converted to the image depth and color type of the destination. Thus, a region moved from a 32-bit color image and pasted onto a grayscale image will be converted to grayscale values. Pixels copied onto indexed-color images are converted to the pixel value with the closest matching color so as not to disturb the existing colormap. The tradeoff of this is that, for indexed-color images, the copied region may appear slightly different from the original; thus it is recommended that indexed-color images be converted to 24 bits/pixel before cut/paste operations.

This is illustrated in the image below, where copies of the explosion at left were pasted onto an image of a purple reptile and onto a grayscale image. The pixels falling onto the grayscale image were automatically converted to grayscale, while pixels falling on the background or the dinosaur retained their original colors.



6.0.5 Paste

Selecting "Paste" or clicking on the main "Paste" button activates a small dialog box that selects what will be pasted onto what. This provides a range of options useful for stenciling, adding labels, and creating mattes and composite images. Important: Both images must be the same depth (bits/pixel).

1. Background to transparent part of image — copies pixels from the background into the image, only in regions of the image that are currently transparent. If transparency is "off", it does nothing.
2. Background to opaque part of image — copies pixels from the background into the image, only in regions of the image that are currently not transparent. Note that if the chromakey bit of the image is "on", the image will suddenly blend in with the background and seem to disappear until chromakey is turned "off". Conversely, if chromakey is "off", it copies everything behind it, obliterating the original image.
3. Opaque part of image to background — copies all pixels from the image onto the background unless they are transparent. This effect is identical to the main "Paste" button in the information area.
4. Transparent part of image to background — copies the original, hidden pixels from the part of the image that is transparent onto the background. If chromakey is "off", it does nothing.
5. Stencil transparent part of image to background — same as option 4, except that the pixels are all set in the current drawing color to create a stencil.
6. Entire image to background — copies the entire, original image onto the background, regardless of any chromakey settings.
7. Add RGB to background – Sets the background red, green, and blue color values to be the sum of the foreground + background. A typical use would be in adding shadows. The foreground image would be thresholded, then reduced in contrast to create a gray figure on a black background. Color-invert the background, then add the foreground RGB values and color-invert the background again to create a dark shadow superimposed on the background image.
8. Paste opaque pixels if > background – If the sum of red, green, and blue pixel values of the foreground image is greater than that of the background image, the pixels are pasted; otherwise the background is untouched.
9. Paste opaque pixels if < background – If the sum of red, green, and blue pixel values of the foreground image is less than or equal to that of the background image, the pixels are pasted; otherwise the background is untouched.

Notes

- These operations do not create any new images. If the result is to be saved, it is necessary to use "File..Create/Resize object" to create a new image.

- The “paste” feature can also be used to paste between parts of images that are in separate windows, provided that they are properly superimposed.
- “Paste” and “Copy/move” do not utilize the X Window clipboard and cannot be used to copy between applications.
- If the source image is zoomed, the pasted portion of the image is converted to its original size before being pasted onto the destination. In order for pasting to work as expected, both foreground and background images must be zoomed by the same amount.

6.1 Changing size

There are four different ways of changing the size of images for easier viewing. They differ in whether they change the image buffer or the screen buffer, and their memory requirements, features, and speed differ accordingly.

Change size (reduplicate) Creates a new, permanent image that is an identical copy of the original except that the pixels are reduplicated or decimated as needed. In other words, if you select an X enlargement factor of ‘2.0’, each pixel will be repeated twice in the X direction. No pixel values will be present in the new image that were not present in the old one. This is the old algorithm for changing size and may be dropped in future versions.

Change size (interpolate) Creates a new, permanent image that is an identical copy of the original except for its size. The new pixels are calculated as weighted averages of pixel values in the original. This gives a smoother, less chunky result, but also can add new pixel values to the image that were not present in the original, changing the image’s histogram. Both of these methods use a lot of memory, because complete image and backup buffers are created. The original image is not affected.

Zoom Increases or decreases the size of the image on the screen by a constant factor by resizing the screen buffer. The entire image is enlarged, but the image and backup buffers are not affected. Not every feature in tnimage works on the zoomed image. You should convert the image to the same depth as the display for best results.

Floating magnifier Creates a new, small image that is continuously updated with the pixels in the area around the mouse cursor. The image region can only be enlarged, and only by 4, 9, or 16x. The new image is useful for precise positioning of the cursor during densitometry, cut-and-paste, or calibration. This method is also useful in low-memory situations because the size of the new image buffer is independent of the size of any images on the screen.

These resizing options are described in more detail below.

6.1.1 Change size

Permanently resizes the currently-selected image. Each dimension can be changed by a different size factor. However, both x and y factors must be either greater than or less than 1.0. For example, setting the x factor to 1.1 and y factor to 4.78 creates a vertically elongated image.

If the image does not contain fine detail, it is often convenient to shrink the image to x and y sizes of 0.5 or less before saving it. Setting the x and y factors to 0.5 decreases the file size by 75%. Conversely, enlarging the image allows each pixel value to be fine-tuned before shrinking it back to normal size. (Note that in this case, 3 images will exist in memory: the original, the enlarged image, and the new shrunken modified image. It is recommended to delete the enlarged images as soon as they are not needed, to avoid running out of memory.)

6.1.2 Zoom

Temporarily resizes the screen buffer of the currently-selected image so that small details can be viewed. To zoom an image, click the "zoom" button or select "zoom" from the menu, then click on the image to be zoomed. Clicking with the left button makes it larger, clicking on the right button makes it smaller, and clicking with the center button restores the original size. Unclick the "zoom" button or click the main Cancel button to return to normal mode. The original image is not affected by zooming. Most image operations such as filtering, labels, sketch, etc. are still functional on the zoomed image. However, menus and area selection are disabled until you exit 'zoom' mode by unclicking the Zoom button or clicking the main Cancel button. To restore to normal size, click the middle mouse button or select "unzoom" or "undo" from the menu. The zooming size factor can be changed in the "Config" dialog.

When executed in a macro or from the command editor, the amount of magnification can be specified. The "unzoom" command restores the original size.

6.1.3 Floating Magnifier

Creates a special image whose contents are continuously updated with an enlarged copy of the region near the mouse cursor. The communication is unidirectional, so it is not possible to change the original image by making changes in the magnified image.

6.2 Rotating

Rotate image

Creates a new image identical to the currently-selected image, except that it is rotated by the specified number of degrees. The antialiasing normally performed during rotation can be disabled by clicking the "Off" button. Disabling antialiasing could be useful if it was desired for some reason to keep all pixel values close to the original.

Flip horizontally

Converts the currently-selected image or region into a left-right mirror image of itself. For images $>$ 8 bits/pixel, only the selected color planes are flipped. Color planes can be selected in the "Config" dialog box.

Flip vertically

Converts the currently-selected image or region into a vertical mirror image of itself. For images $>$ 8 bits/pixel, only the selected color planes are flipped. Color planes can be selected in the "Config" dialog box.

6.3 Image properties

Sets a variety of modifiable properties for each screen object or image.

Copy tables - Copies the colormap, gamma table, or calibration parameters from some other image.

Color type - Sets the 'colortype' flag for the image to Grayscale, Indexed, or Color. **Warning:** Changing color types is not always a reversible process.

Image Number - Sets the source and destination image numbers for the 'copy tables' option.

Image attributes - Sets the title, position, image depth, and transparency (see Sec. 6.3 for the image.

Transparency

Checking the "Transparent" button in the "Image properties" dialog causes all the pixels in the image to be treated as transparent. This means that images behind the transparent image or on the background will partially show through.

Note. The transparency feature still has some minor limitations:

(1). Best results are obtained by starting the X server in a color mode (16 or 24 bits/pixel). With XFree86, this can be done with the command "`startx -- -bpp 16`".

(2). If the image does not appear transparent after performing some operation, moving the image by clicking on the arrow buttons should restore the transparent effect.

(3). No conversion between pixel types or image depths is currently performed by the "Paste" button.

(4). Saving images as transparent GIFs is not yet supported.

(5). Transparency can be slow with color images, because calculating the new pixel value requires fetching and performing calculations on the RGB values for both the foreground and background. This can take up to several seconds for a large image. If possible, convert the image to grayscale first.

(6). All images must be the same bits/pixel as the display. Use "change image depth" if necessary to convert the images.

(7). Multiple transparent images may be superimposed in as many layers as desired. Bear in mind that if the background is not black, the background will also show through a transparent image.

(9). Before using the "Create/Resize Image" function to create a new composite image from a region of the screen containing one or more transparent images, or performing some other operation such as image warping, the images should be converted to the same bits/pixel as the display in order to obtain satisfactory results.

To change the transparency settings, select "Image..image properties.." (or click the "Prop" button at left), and set the "Transparency" value to the desired value (0=completely opaque, 100=completely transparent).

To deactivate transparency, simply set the transparency value to 0.

6.4 Chromakey

Chromakeying is the process of making a specific range of colors transparent. In contrast to transparency, this makes the pixel either completely opaque or completely transparent. A common use for this is in creating composite images, such as the image below of sea slugs in the Martian landscape. This image was created by making an image of the sea slug surrounded by a uniform dark color, then setting "chromakey" for that image to make that color transparent, creating a cutout. The cutout can then be positioned, rotated, or resized as desired, and then pasted onto the image behind it by clicking the 'Paste' button in the information area at the left. This causes the opaque pixels in the currently-selected region to be pasted onto whatever is behind it (*i.e.,* onto the background or the image behind the transparent image).

Left: Composite image created using chromakey feature
Right: Chromakey dialog

To change the chromakey settings, select "Image..image properties.." (or click the "Prop" button at left), check the "Chromakeyed" button, and click OK. A small dialog box will appear. If the image is color, select the RGB values for the minimum and maximum colors which should appear opaque. For grayscale images, select the minimum and maximum opaque pixel values. If "Invert chromakey" is checked, the opacity will be reversed. For example, if the minima and maxima RGB values were 20 30 40 and 123 123 123, normal chromakey would make a pixel opaque if its red is between 20 and 123, its green is between 30 and 123, **or** its blue is between 40 and 123, and transparent otherwise. Inverse chromakey would make the pixel transparent under those conditions and opaque otherwise.

To deactivate chromakey, uncheck the "Chromakeyed" button. All operations on chromakeyed images are significantly slower than with normal images. Also, chromakeying is performed recursively, so that the background will still be visible regardless of the number of chromakeyed images superimposed on it.

6.5 Frame Controls

Activates the frame control for multi-frame or 3D images. Each frame control permits selection of the visible frame of one image and rapid switching of frames to create a "movie". If the image only has a single frame, it has no effect. The frame may be selected manually, or the time interval between frames can be adjusted for a movie. (*Note: Sound is not supported.*)

Click on “Start” to automatically change frames and “Pause” to stop. The interval between frames in milliseconds can be adjusted. On slower computers or with large images, the actual interval may be larger. If frame rates are too low, it may be helpful if `tnimage` is started in a screen mode that has the same depth as the image.

Movies The `movie` function can also be given in the command editor. The syntax is:

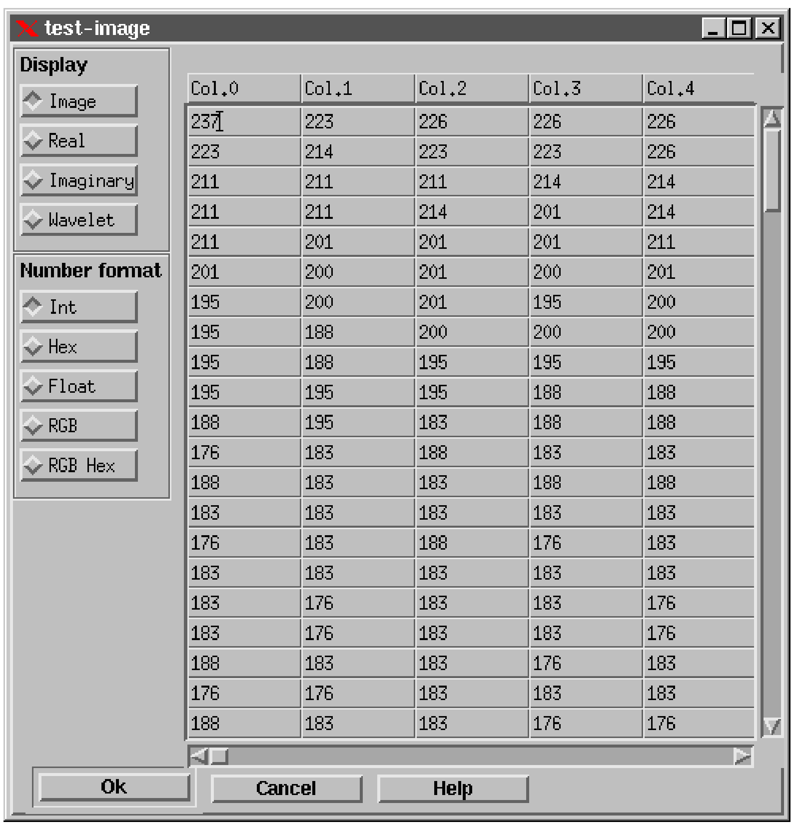
`movie` *delay*

where *delay* is the time between frames in msec. Using the command editor gives higher frame rates than the multiframe control. Be careful not to specify too short of a delay, as this will cause the computer to become unresponsive. Clicking on “OK”, “Cancel”, or the main Cancel button will stop the movie.

6.6 Spreadsheet

Opens a spreadsheet window for the current image. Pixels can be displayed as decimal integers, hexadecimal integers, RGB values, or hexadecimal RGB values. If a FFT is being displayed, the real or imaginary components can also be displayed as floating point numbers. Changes of pixels on the image (for example, by adding a label or changing the contrast), are immediately reflected in the spreadsheet. Similarly, typing a number in the spreadsheet changes the value of the corresponding pixel (if the number is a valid pixel value). Also, selecting an area on the image automatically causes the corresponding part of the spreadsheet to be highlighted. This gives precise control over the image as a numerical array. A separate spreadsheet can be activated for each image.

Since tnimage can read and write 1 bit/pixel images as raw bytes, the spreadsheet can also be used as a bitmap editor.



Col.0	Col.1	Col.2	Col.3	Col.4
237	223	226	226	226
223	214	223	223	226
211	211	211	214	214
211	211	214	201	214
211	201	201	201	211
201	200	201	200	201
195	200	201	195	200
195	188	200	200	200
195	188	195	195	195
195	195	195	188	188
188	195	183	188	188
176	183	188	183	183
188	183	183	188	188
183	183	183	183	183
176	183	188	176	183
183	183	183	183	183
183	176	183	183	176
183	176	183	183	176
188	183	183	176	183
176	176	183	183	183
188	183	183	176	176

Spreadsheet display of an image

6.7 Repair, Backup and Restore

Repair

Re-builds and remaps the screen display for the image in case it is incorrect. (Alt-R has the same effect).

Reinitialize

Resets all program data to initial values. This can sometimes restore functionality in case user closed a dialog window using the Window Manager instead of in tnimage, or if the program gets screwed up for some reason.

Show Alpha Channel

Temporarily displays contents of the alpha channel. Pressing Alt-1 has the same effect. Pressing Alt-R or selecting "Repair" restores the display.

Backup

Creates a backup copy of the current image in memory. This is recommended before manipulating the image or adding text, in case you don't like the results. Selecting "Restore" will retrieve the screen back onto the screen. See also sec. 11.4.

Restore Replaces the current image with the backed-up copy if one exists.

Note: The FFT is not backed up. If the image has been Fourier-transformed, only the original image is restored. If the display is set to show the real, imaginary or power spectrum components, restoring the image will not affect the display. However, the next time the image is transformed, the restored data will be used.

Note: if the depth of an image is changed, the original backup is no longer valid and is automatically discarded.

Clear alpha channel

Removes all labels in current image. Image backups must be turned on for this to work.

Manually select area (F3)

Starts non-rectangular area selection mode. Click the 1st mouse button and drag to outline the desired area(s). Then click the main Cancel button when finished. The areas need not be contiguous. Function key F3 has the same effect.

Switch selected/unselected

Changes a non-rectangular selected region so that the complement of what was selected is now selected. For example, if you double-clicked on an object, the selection now becomes a stencil. If the selected region extends over 2 or more images, only the part in the currently-selected region is inverted.

Section 7

Process menu

7.1 Filter

Performs a convolution filter on the currently-selected image, or the selected screen region. The original image should be backed up before filtering it, in case you don't like the results.

For images greater than 8 bits/pixel, only the selected color planes are filtered. Color planes can be selected in the "Config" dialog box. Indexed-color images are temporarily converted to 24 bits/pixel before filtering, then converted back using the quantization method selected in the "Configure..." dialog.

In general, the kernel size should be close to the size of the features of interest. In particular, with Laplace edge enhancement, if the features are larger than 3 pixels across, a 3x3 or 9x9 kernel may actually make the image appear fuzzier, while a 5x5 kernel might dramatically sharpen the image. The amount of filtering can be adjusted from 1 to 100.

High-pass filter (Sharpening)

Extracts only the highest spatial frequencies in the image. More low frequency components are saved if a larger 'kernel' is selected. If the image contains few high-frequency components, and too much sharpening was selected, the result may appear dark or black. This can be corrected by increasing the contrast of the image after filtering it.

Low-pass filtering

Low-pass filtering eliminates sharp edges from the image, causing a blurring effect. The total intensity is unchanged, but is spread throughout the neighboring pixels.

Background subtract

This is similar to high-pass filtering except it removes only the lowest frequencies. This has the effect of removing uneven background in the image. As with high-pass filtering, it can make the image appear darker. Selecting a larger kernel removes less of the low frequencies, but will require much more processing time. A preferred method is to use the "kernel multiplier". Selecting a multiplier of 5 with a 3x3 kernel produces results similar to using a multiplier of 1 with a 15x15 kernel, but is many times faster. For this reason, it is recommended to set the kernel to 3x3 for background subtract.

It is possible to specify whether "black" or "white" pixels should be considered as the background. If it is set to "white", the image will generally get lighter in regions of unvarying pixel intensity. If the background value is "black", the image will get darker wherever the pixel intensity does not vary. This can be compensated for by increasing the contrast.

Background subtraction will also trash any text in the image.

See also *Background flatten*, *remove low frequencies*, and *Rank leveling*.

Background Flatten (de-trending)

This filter removes the largest-scale gradients from the image or selected region by measuring the average pixel value in each corner and then adding or subtracting a value to equalize the overall intensity. This could also be done manually, by creating a gradient region on the screen and adding the image to it (see *Gradient fill*). However, the manual method is not 2-dimensional.

In contrast to Background Subtraction, this filter does not trash text. It differs from *remove low frequencies* in that it only removes large-scale gradients from the image.

See also *Background subtract*, *remove low frequencies*, *Force background*, *Rank leveling*, and *unsharp mask*.

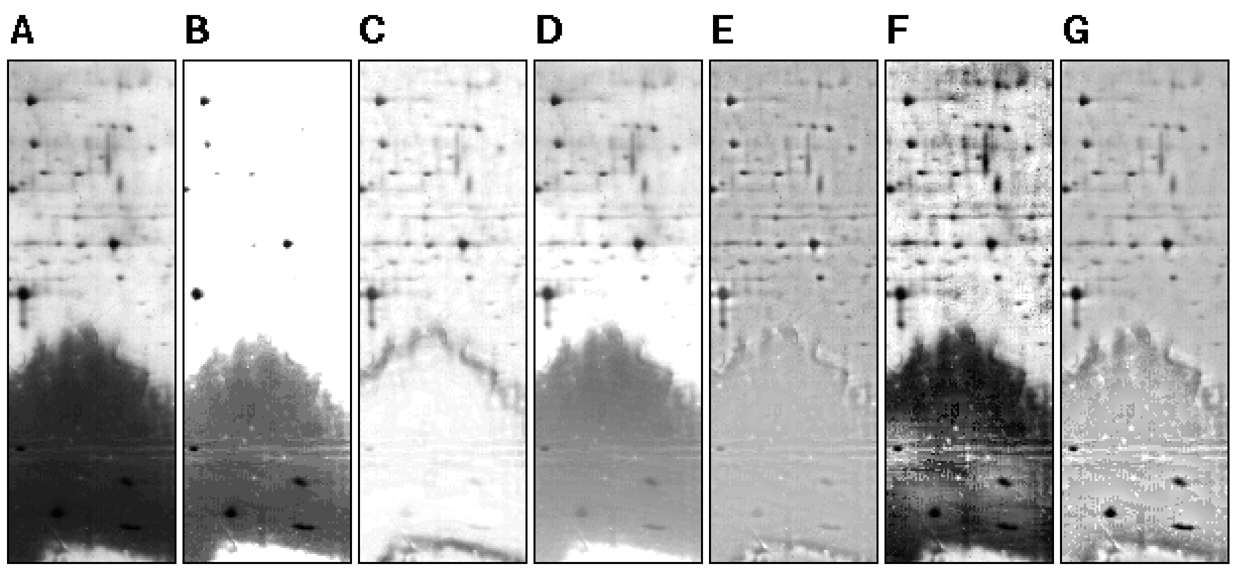
Force background to fixed value

This filter is similar to the "Remove low frequencies" filter, except that instead of adding and subtracting values to remove gradients in the image, it multiplies dark and light areas by some factor to make the low-frequency background constant. This constant value can be adjusted by clicking on the "Background value" box.

The difference between this filter and removing low frequencies is that, with this filter, the intensity of any information hidden in a dark area will be more accurately preserved. This filter can be thought of as a cross between the "maximize local contrast" and "Remove low frequencies" filters.

Forcing the background to the 50% intensity value (e.g., 128 for 8-bit images) gives the best retention of the original intensity variation. Setting it to 100% (i.e., 255) would be useful if the high-frequency features (such as grains or spots) were of interest.

See also *Background subtract*, *remove low frequencies*, and *Rank leveling*.



Comparison of different methods of compensating for uneven background.
A Original image
B Conventional contrast increase.
C Background subtract (bkg=white).
D Background flattening
E Remove low frequencies
F Maximize local contrast
G Force background to fixed value (191).

Noise filtering (median filtering)

Median filtering removes extraneous pixels from the image. This is useful if the image contains noise which consists of little dots that are clearly outside the range of the other pixels. If a given pixel varies by more than a certain amount from the pixels around it, median filtering substitutes the median of the neighboring pixels, thereby eliminating noise from the image. The image is otherwise unaffected. The range in ± pixel value units, outside of which a value is to be considered noise, can be changed. A higher value results in less noise removal. Excessive noise filtering can make an image appear posterized. A setting of 0 will create a smoothing effect.

Laplace edge enhancement

This filter finds any edges in the image whose length is equal or greater than the kernel size. The edges are then increased in intensity while non-edge regions are eliminated. The effect on text is to create an outline of the text.

Sobel edge enhancement

This filter is similar to Laplace edge enhancement except that the effects are gentler. A Sobel filter enhances the color or intensity gradient in a symmetrical fashion, so that areas of constant color become black. **Note:**The Sobel function here is slightly different from some other implementations, hence the term 'Sobel' may not be strictly accurate.

Edge detect — , –

Detects and enhances edges in the image preferentially in the horizontal or vertical direction.

**Sharpen — / – **

Sharpens the selected region or image preferentially in the indicated direction, creating a 'freeze-fracture' 3-D effect. This is particularly useful for images of biological specimens that have low contrast, since it highlights the overall shape of the cells.

Remove low frequencies

This filter uses a combination of operations to subtract a blurred copy of the image from itself, thereby making the background perfectly flat. The effect is about halfway between Background Subtract, which only removes the very low spatial frequencies, and High Pass, which enhances the high frequencies. Selecting a 3x3 kernel removes all but the highest frequencies, while a 15x15 kernel has a less harsh effect. The effect can be made even more gentle by specifying a kernel multiplier of 2 or more.

See also *Background flatten*, *Background subtract*, *Force background*, and *Rank leveling*.

Background subtraction by rank leveling

Rank leveling subtracts background gradients rapidly by assuming that the features of interest are small in size (less than 10×10 pixels). Subtracting nearby background pixels from the feature pixels achieves a very localized background flattening. Features larger than 10x10 will also be attenuated.

"Background level" must be set correctly, otherwise the result will be garbage. For example, if the features are dark on a light background, background level must be set to "white".

This filter only works for grayscale images. Other settings including kernel size, range, and amount of filtering, are ignored.

As with "background subtract", it is usually necessary to adjust the contrast after filtering with rank level filter, because the image will get lighter or darker depending on the setting of "background level".

See also *Background flatten*, *Background subtract*, *Force background*, and *Remove Low Frequencies*.

Unsharp mask

This filter is similar to the old-style unsharp mask filter used in photography. A temporary copy of the image is blurred and subtracted from the original. A constant value is added to each pixel to prevent pixel values from going to zero or becoming negative. This can be adjusted by changing the “Background value” setting. The amount of blurring done to the copy is set by the “Kernel” and the “Kernel multiplier” settings. Setting the kernel multiplier causes the blurring filter to be applied to the copy more than once. The percentage of the blurred copy that is subtracted can be adjusted by changing the “Amount of filtering”.

Unsharp mask filtering flattens the background and also sharpens the image. However, it is less efficient for sharpening than the convolution sharpening filter. For sharpening images, use the “sharpen” convolution filter instead.

User-defined

Performs a convolution filtering using a user-defined filter. This filter should be a text file with the folowing characteristics:

1. First line is a single odd number indicating the number (n) of rows.
2. The no. of rows must equal the no. of columns.
3. The first line is followed by n rows of n numbers separated by whitespace. The numbers can be positive or negative, and need not be integers.
4. The total of all elements should equal 1 (=100%), otherwise an error message is printed. If the total is not 1, the filter will make the image lighter or darker. The center element, which represents the pixel under consideration by the filter, should be increased or decreased to make the total equal to 1.
5. The “Kernel” setting is ignored for user-defined filters.
6. Size of filter is not limited.
7. No scaling is done on the filter elements. For example, a value of 5.43 will add a weighting of 543 if the pixel has a value of 100.

Example This 5×5 filter will enhance vertical edges.

```
5
-1  -1  0  1  1
-1  -1  0  1  1
-1  -1  1  1  1
-1  -1  0  1  1
-1  -1  0  1  1
```

Engrave

Filters the image by converting edges to shadows so that the image resembles an engraved outline. Uses a threshold value to determine the pixel intensity value at which the outline is to be drawn.

Spatial difference

Filters image using a spatial differencing filter. This has the effect of separating closely-spaced objects so that there is a clear boundary between them. This is useful in preprocessing images before performing densitometry or grain counting. Uses a ‘size’ parameter which determines the area over which the differencing will be performed for each pixel. A larger size will accentuate larger objects, and will have more noise immunity, but takes more processing time. Currently only works on monochrome images.

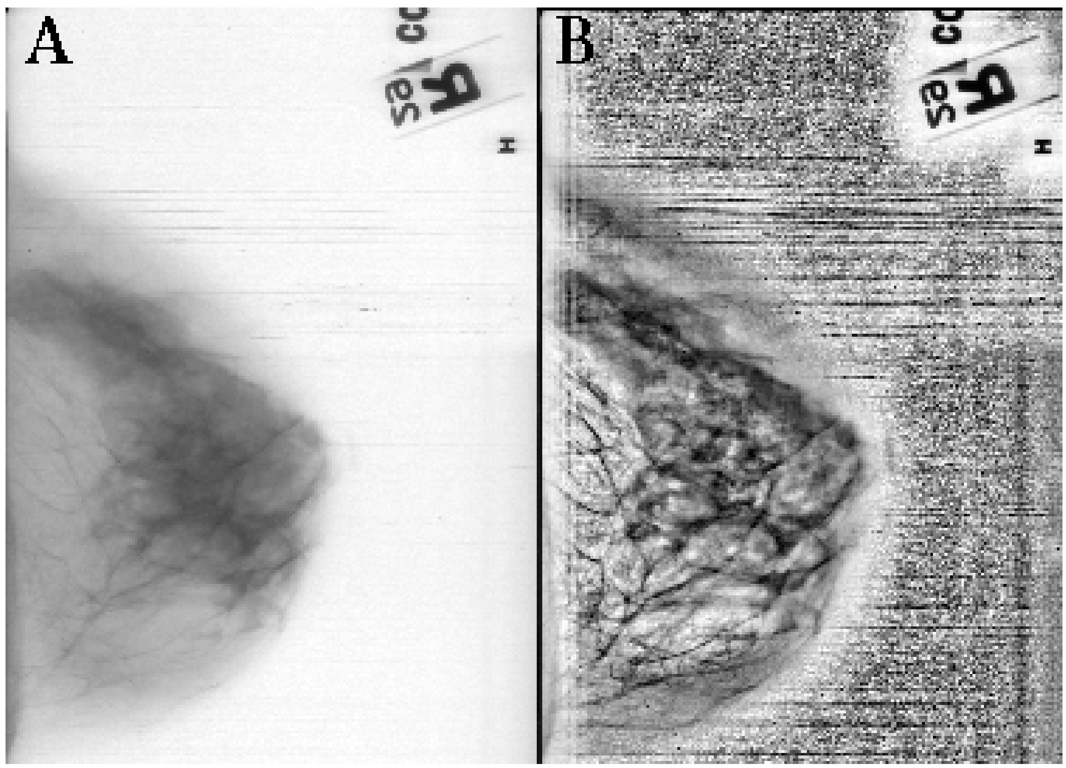
Multiplicative Sobel

This experimental filter is similar to the Sobel filter (Sec.7.1) except that the differences are multiplied to each pixel rather than added. It produces a strongly non-linear effect of enhancing details in the image. Only works on monochrome images.

Maximize local contrast

This filter increases the contrast at each point in the image to the maximum possible value with respect to its neighbors. This is useful for images in which details are obscured in large dark or light areas. Increasing the contrast in the overall image would make these faint features visible, but at the expense of washing out all detail in other areas. Maximizing the contrast locally allows details in both light and dark areas to be viewed simultaneously. See Sec.

For example, in the mammogram shown below, in the left panel (**A**), some of the details in the mammogram are visible, while others are obscured in a dark area. After maximizing the local contrast, the details of the vascularization are easily discernable, and even the subtle gradations in the white background area are also clearly visible (**B**).



Maximizing local contrast.

The degree of contrast enhancement can be selected in the clickbox labeled "Local contrast scale". This value is the area over which the contrast will be measured for each pixel. A smaller value gives greater enhancement of fine details, while a large value only removes gentle background gradients. An excessively large value, however, can create artifacts. See sec. 8.12 for another example.

Adaptive maximize local contrast

This filter is similar to the "Maximize local contrast" filter, except that the contrast factor is calculated "adaptively". This makes it much faster. However, the results are sometimes not as good. The adaptation or "decay factor" can be adjusted.

Adjustable parameters

Kernel

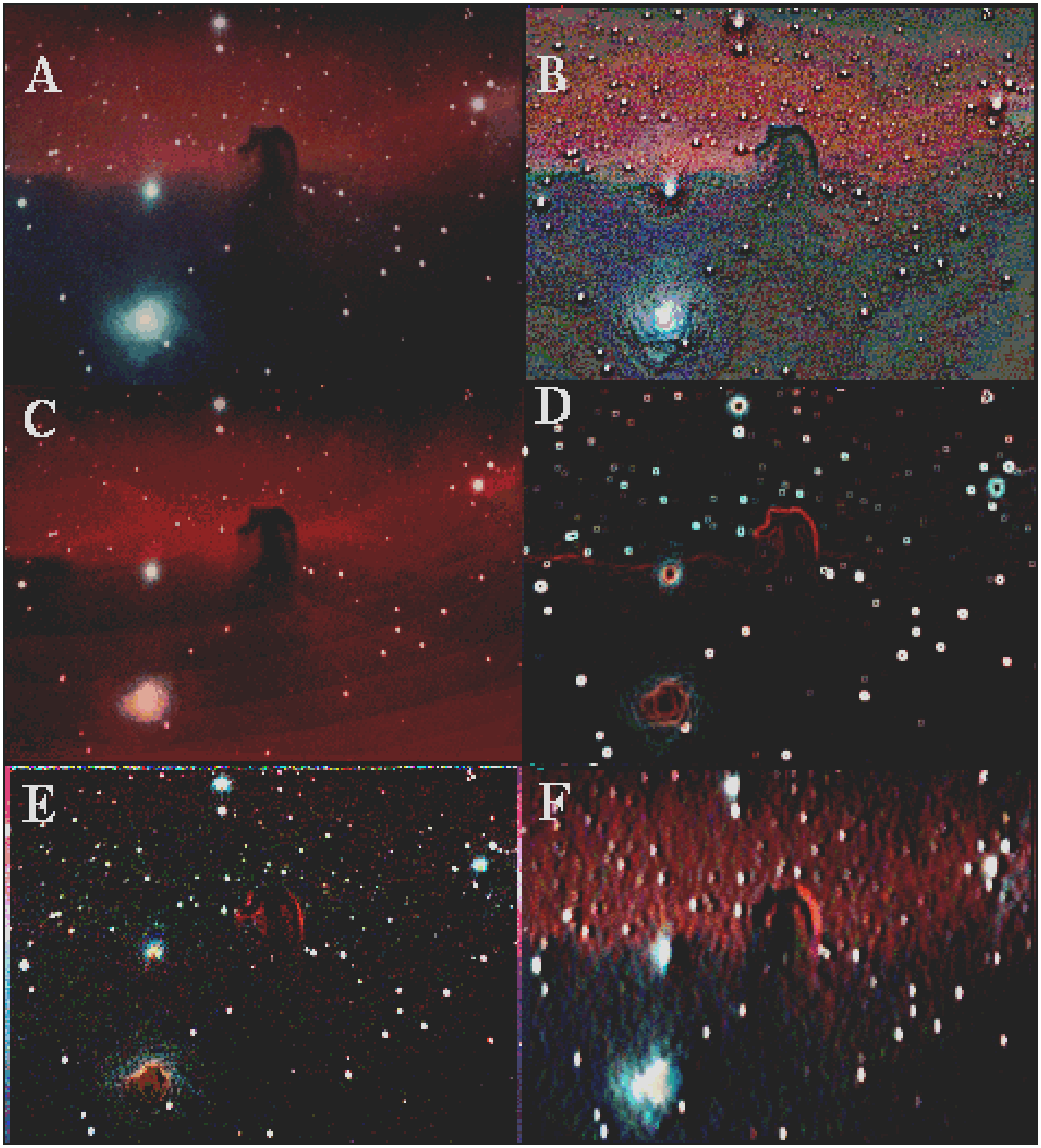
The kernel is the number of pixels used in calculating a pixel in the new image. The processing time needed increases with the square of the kernel size.

Kernel multiply factor

A *kernel multiply factor* permits arbitrarily large kernel without an increase in computation time. This is possible because usually only a small sample of the surrounding pixels are really needed to calculate the new pixel value. For most types of filters, a 3x3 kernel with a kernel multiplier of 3 gives the same results as a 9x9 kernel, but is 3 times faster. For sharpening and blurring, this doesn't work, and the factor is automatically set to 1.

Amount of filtering

The amount of filtering applied to the image, from 1 to 100 (maximal filtering).



Effect of some different filters. The original is an image of the Horsehead Nebula.
A. Original
B. Background subtract, background level=white
C. Background flatten
D. Sobel edge detection
E. Background subtract, background level=black
F. Sharpen |

7.2 Mask

Sets the pixels of a target image to a certain value depending on the pixels in some other image. This allows several types of special effects, such as combining parts of two images. The term "mask" is from photography, where originally a black cutout was placed over the image, and the film was then double-exposed, to accomplish this.

1. **Mask (1 &=2)** Each pixel in image 1 is set to 0 if the corresponding pixel in image 2 is 0, otherwise it is untouched. "Corresponding pixels" are pixels the same x and y distance from the upper left corner of each image.
2. **Inverse Mask (1 &=˜ 2)** Each pixel in image 1 is untouched if the corresponding pixel in image 2 is 0, otherwise it is set to 0.
3. **Add (1 + 2)** The value of each pixel in image 2 is added to the value of the corresponding pixel in image 1. This was used, for example, in overlaying the watershed demarcation lines onto the original image in Sec. 7.16.
4. **Subtract (1 - 2)** The value of each pixel in image 2 is subtracted from the value of the corresponding pixel in image 1.
5. **Multiply (1 * 2)** The value of each pixel in image 1 is multiplied by the value of the corresponding pixel in image 2.

7.3 Warp

Corrects (or adds) distortion in the image.

7.3.1 1-D Warping

1. Select a series of control points to define the distortion to be corrected. For example, if you have an SDS gel in which the bands are U-shaped, trace out a U-shaped curve somewhere on the image that follows the U shape.
2. Press any key. This fixes the curve in place (it becomes invisible).
3. (Vertical warping only) With the mouse, select a y-value somewhere below the lowest point in the curve which you traced out.
4. Click the left mouse button at the desired y-value.
5. Each vertical line on the screen will be shifted up or down so that each point on the curve you traced out is lined up with the selected y-value.

Horizontal warping is similar, except that the pixels are shifted left and right to create a ripple distortion effect similar to a reflection in water. The height of the curve you trace with the mouse determines the distance by which the pixels at the corresponding x coordinate are shifted. (Up→left, down=→right).

7.3.2 2-D Warping

Clicking on "2-D warp" creates a new copy of the image with a square grid of blue lines. The grid vertices are the control points. Move the control points as desired to distort the image in 2 dimensions. When finished, pressing Esc or clicking the main Cancel button will erase the grid lines and return to normal mode.

There are two ways to move the control points:

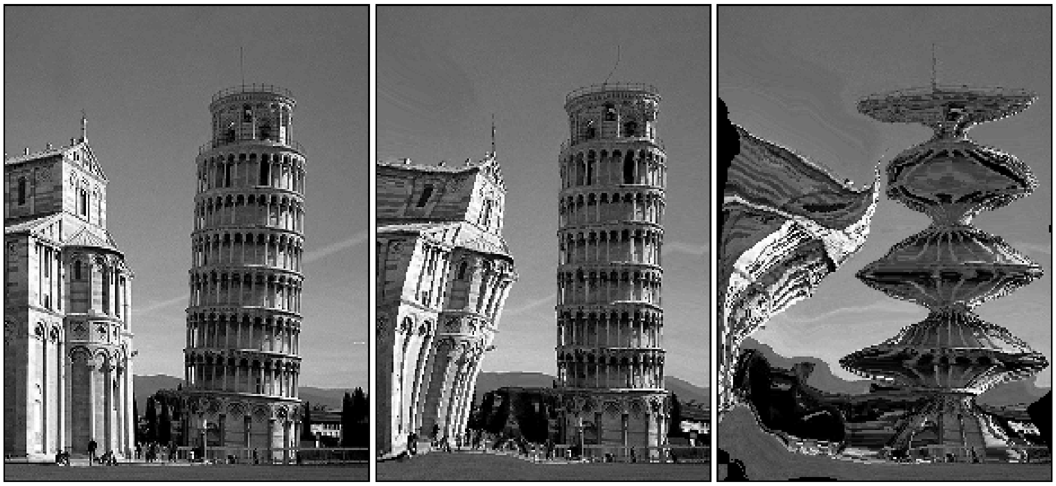
1. Click near the control point with the mouse and drag the control point in the desired direction.
2. Click near the control point to select it and release the mouse button. Then use the arrow keys on your keyboard to move the control point. This has the advantage of moving the point by a fixed number of pixels each time.

Pixels/grid point: selects the spacing between grid points.

Cursor movement: selects the distance a grid point is moved when arrow key is pressed.

Commands available while warping is active:

- **c** Makes a copy of the warped image. Warping then continues, making it easy to create multi-frame images with different amounts of warping (see example below).
- **g** Toggles the grid lines on/off. Warping is still possible.
- **r** Causes the grid to revert back to its original square shape.
- **Esc** Press Esc or click the main cancel button when finished.



Example of several frames created by pressing 'c' during 2D warping, showing increasing degrees of distortion to the original image.

Tips on 2D warping

1. Grid lines should not cross each other. This can cause discontinuities in the warped image. If a large displacement is needed, move the adjoining grid points as well to prevent this.
2. It is okay if the control points are off the edge of the image.
3. Speed can be increased by turning grid lines off. Even though the grid lines are not visible, warping is still active. You should make sure the individual displacements are small enough to prevent the control points from crossing. Also, extreme displacements can cause an unnaturally smooth appearance due to replication of adjacent source pixels.
4. Individual grids should be kept as rectangular as possible to preserve spatial relationships within features.
5. Images should not be deleted during warping. If the original image or warped image are deleted during warping, an error message is displayed.
6. Future versions of `tnimage` will be able to move sets of grid points simultaneously instead of one at a time.
7. As with all functions in `tnimage`, only the active color planes will be warped. For example, you can warp the red component of your image by clicking "Config..configure" and un-checking the Green and Blue Color Planes checkboxes. This may be useful in removing chromatic aberration (see also Chromatic aberration in Sec.7.6).

7.3.3 Removing barrel or pincushion distortion

Removing barrel or pincushion distortion is a special case of 2-D warping. The image is automatically divided into a 4×4 region marked by a blue grid. The corners and the points where the blue lines intersect are control points. To remove barrel distortion, move the corner control points away from the center of the image. To remove pincushion distortion, move the corner control points closer the center of the image.

There are two ways to move the control points:

1. Click near the control point with the mouse and drag the control point in the desired direction.
2. Click near the control point to select it and release the mouse button. Then use the arrow keys on your keyboard to move the control point. This has the advantage of moving the point by a fixed number of pixels each time.

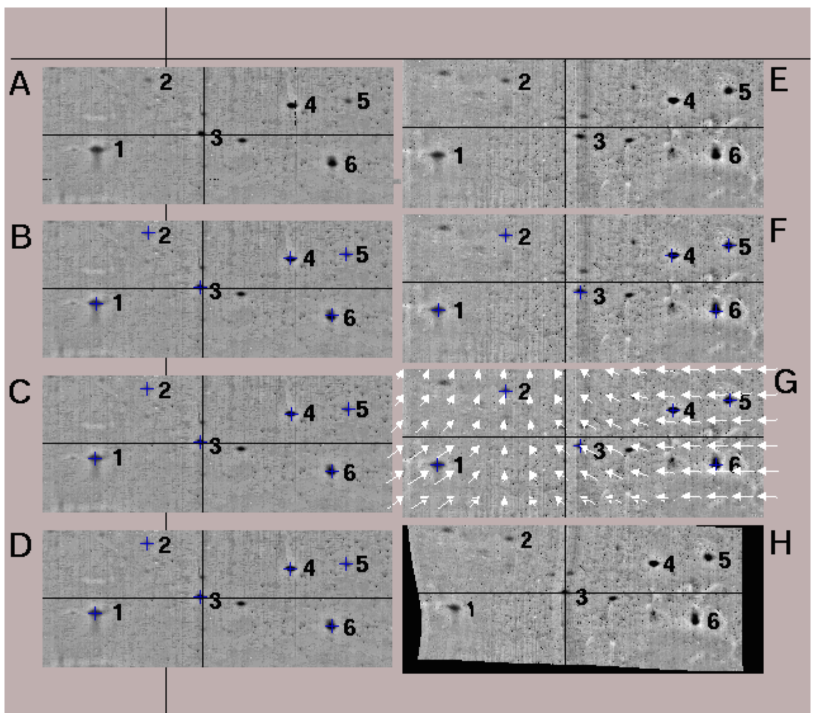
Next, move the intermediate control points along the edge of the image so that the image is warped in a smooth manner. It is okay if the control points are off the edge of the image.

When finished, press the Esc key. The blue guide lines will disappear, indicating that warping is finished.

7.4 Image registration

In image registration, the pixels in an image are remapped to superimpose an arbitrary number of fixed control points. Image registration is thus a generalized form of affine transformation; however, in image registration, the parameters (translation, rotation, scaling, etc.) are not constant but vary with x and y.

Image registration is ideal for repairing gradual or "differentiable" distortions, but less suitable for correcting abrupt discontinuities in the image, such as those caused when an object moves while being scanned in a scanner. These can be removed by manual image warping (Sec. 7.3).



Registration of an image of sections of two protein 2D gels.
A,E. Original images with control data points labeled 1..6
B,F. Same images after obtaining data points. The points are marked with a small cross.
C,G. Vector display superimposed on image
D,H. Result of warping the right image. Program is in multiple cursor mode with cursor on spot #3. All spots in H are now aligned with their corresponding spots in D.

Typical uses for image registration would be in analyzing terrain maps, and 2D polyacrylamide gels. In a 2D gel, each protein spot is identified by its x and y coordinates, so it is necessary to know the exact position of each spot in order to accurately identify it. The figure above illustrates the steps in registering two protein 2D gel images.

The image registration algorithm in `tnimage` consists of two parts. First each part of the image is rotated to remove any rotations or twist in the image. Then, a large array of vectors or *vector map* is created that allows each part of the image to be elastically warped in the x,y direction to remove any residual differences in the pixel coordinates. This two-step procedure is used because rotation usually is much simpler than warping (i.e., one rarely finds images with regions of local twisting), and therefore rotation needs fewer control points. For example, if there is no local twisting, a single control point is sufficient to correct image rotation.

Once calculated, the same warping map can be used either to warp an image, or to locate the new positions of a set of data points. This approach is more flexible than calculating the new coordinates on an 'as needed' basis because the same map can be used to warp multiple images very rapidly, to remap arbitrary lists of coordinates that may not be related to any image, and to create lists of coordinates for automatic densitometry without actually warping the image. This latter feature is important since warping an image can introduce slight variations in the size of features on an image.

Unlike other programs, `tnimage` does not need a predefined set of landmark points from the two images. `Tnimage` uses a powerful pattern-matching method that can usually find the correspondences between two sets of points without user intervention. Once the correspondences are found, the user can make corrections as needed before applying the map to an image.

`Tnimage` matches discrete control points instead of matching all grayscale features in an image. The reason for this is that grayscale features are irrelevant for many types of images (such as 2D gels). The control points can be obtained manually, or automatically by using the grain counting function in `tnimage`, or any other suitable means. For terrain maps, edges and corners can be used.

After the images are registered, the cursor can be set to "multiple crosshairs", which draws a set of crosshairs on each image. This facilitates identification of corresponding features. You can place the cursor at a spot in one image and observe the corresponding location of the second cursor on the other image. You can also run macros to perform densitometry (for example) on specified areas, without having to change the macro for each image.

The program uses two sets of data points:

- Data points - an arbitrarily large set of coordinates indicating the x and y locations of coordinates whose positions are to be identified. These can be read individually by clicking "Read points" or read from a combined file by clicking "Read match table".
- Landmark points - A smaller set of known points whose corresponding positions on both images are either already known or should be calculated. There is a practical maximum of 100 landmark points. The landmarks are stored in one single file.

The landmark points are used to calculate a vector map which can then be applied either to the data points or to an image. The landmark points do not need to be a subset of the data points.

Neither image has to be present unless you wish to obtain control points interactively or perform warping.

Clicking "Correlate points" automatically sorts and cross-correlates the two sets of landmark coordinates to determine the which landmark points correspond to each other, and eliminates orphan points. Correlating points is unnecessary if both landmark sets already contain corresponding points, as when the points have been obtained interactively.

Options

- Image no. - selects which image to use as a reference and as an unknown image. Only needed when obtaining points or warping images.
- Obtain landmarks - Manually obtain landmark points from reference and unknown image.
- Translate landmarks - Add an x and y value to the landmark points. This can be useful in improving the accuracy of the calculations.
- Read data points - reads a full set of data points to be remapped.
- Save data points - saves the data points after remapping.
- Restore data points - restores the original data points in case something went wrong with the remapping.
- Read landmarks - reads a set of landmark coordinates from disk.

- Edit landmarks - permits viewing and modification of the landmark coordinates.
- Save landmarks - save the modified landmarks list.
- Read match table - Read the worksheet containing the data points and other information from disk.
- Edit data/match table - Edit and modify the data points
- Save match table - Write the modified data points to disk.
- Correlate points - Calculate the correspondences between the possible landmark points and generate a landmark table (Not needed if the landmark points are already correlated).
- Calculate rotation map - Create a large rotation map suitable for rotating an image or remapping data points.
- Rotate image - rotate an image using the most recently calculated rotation map (which is derived from the landmark points).
- Smoothing - Sets the amount of smoothing to be applied to the vector map or rotation map (1=3, 2=5, 3=7, 4=9, 5=11, 6=15, 7=19, or 8=21 point smoothing). Changing this parameter can help in fine-tuning the construction of an optimal vector map.
- Calculate vector map - Create a large vector map suitable for warping an image or remapping data points.
- Warp image - warp an image using the most recently calculated warp vector map (which is derived from the landmark points).
- Show data vectors - Draws arrows indicating where each data point will move if the image is warped.
- Create unwarped spot list - Calculates the new identities of the current list of data points from the "unknown" data set along with their original coordinates, and displays them in an editor. This option should be selected if you did not warp the image.
- Create warped spot list - Calculates the new identities of the current list of data points from the "unknown" data set along with their new coordinates after warping, and displays them in an editor. This option should be selected if you warped the image.

7.4.1 Registering two images manually

This procedure will warp an image to match a reference image, using manually-obtained landmark points.

1. Identify a number of points that are landmarks visible in both images. In the example above, 6 protein spots were selected and labeled 1 to 6. The greater number of points that are selected, the more complicated the warping that can be done.
2. Click the "Image no." settings to ensure the desired image numbers are shown.
3. Click on "Obtain landmarks" for the reference image and click on each spot. Press Space when finished. Keep a mental note of the order in which the points were clicked.
4. Repeat for the unknown image, making sure to click on the point in the second image that corresponds to a point in the first image.
5. Click "Edit landmarks" if desired to check the selected coordinates and edit if necessary.
6. If the unknown image needs to be rotated, click "Rotate image". This should be done before any warping. The landmark points will also be rotated, so that the same landmarks can be used for warping.
7. Click "Calculate vector map". This will create a new image illustrating the vector map using a color-coding scheme.
8. Click on the image to be unknown to select it.
9. Click "Warp image" to perform the warping alignment using the vector map.
10. (optional) Select "Config..Configure..Main cursor" and change the cursor to "multiple crosshairs". The corresponding spots will now be indicated by crosshair cursors on both images.

Notes

1. If the image is both rotated and warped, rotation should be performed first. Better results are occasionally obtained if new control points are obtained after rotating.
2. Only one point is needed to rotate or warp an image. If a single point is used, the rotation or translation will be constant over the entire image. More points allow correction for more complicated twists and warps.
3. Don't click "Correlate points" when using manually-selected control points. Manually selected points are already correlated.
4. Rotation and warping are done independently for greater flexibility. If the image is known not to be rotated, you can click on "Warp image" to warp without rotating (and vice versa).
5. The control points are automatically moved to new locations after rotation and warping.
6. Avoid selecting control points for rotation near the exact center of the image, since the image is rotated around the center point. This might cause excessive twisting near the center of the image.

7.4.2 2D Gel Analysis

Although image registration will work with any type of image, it has been optimized for automatically aligning 2D protein gels. Aligning and measuring the hundreds of spots on a typical 2D gel can be tedious and time-consuming. Here is a procedure for performing densitometry on two 2D gels rapidly.

2D Gel Analysis - part 1: Obtaining spot coordinates

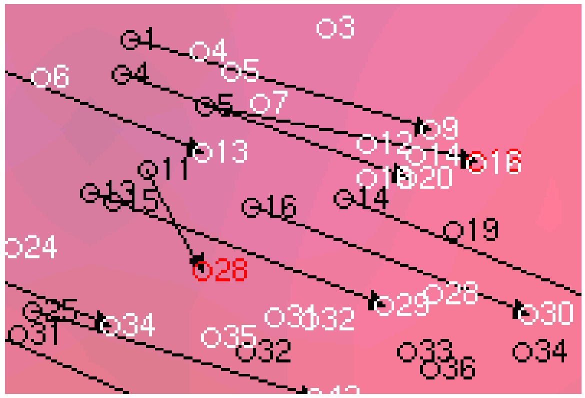
1. Open the "reference" image. This should be the best 2D gel available, showing as many spots as possible.
2. Clean up the 2D gel image to eliminate streaks, edges, molecular weight markers, labels, borders, dirt, etc. The algorithm can match spots correctly if as few as half the points correspond to actual spots, but matching will be more accurate if no noise is present.
3. Filter the image (using "Force background to fixed value" or "Remove low frequencies" if necessary to ensure that the background is constant.
4. If some of the spots are faint, make them darker by increasing the contrast or by drawing/painting on the spot. This will not affect the results, as the spot signals are ignored at this step. (Don't save the modified image, however. The original image will be needed for densitometry).
5. Using the "draw" function, draw white lines on the image to separate any spots that are overlapping.
6. If the 'unknown' gel is rotated with respect to the reference gel, better results will be obtained if you rotate it manually before starting. This can be done in Image Registration or by clicking "Image...Rotate".
7. Click "Grain/Pattern counting"
8. Perform Grain Counting on both images, using appropriate threshold and minimum size values so that all the spots are counted.
9. Click "Save Grain Results" to write the spot locations to disk.
10. Make a backup copy of the grain counting data. This will be used during densitometry.
11. Save a copy of the modified, marked image for future reference. (Don't overwrite the original image).
12. Repeat for the other "unknown" images.

2D Gel Analysis - part 2: Generating spot list

1. Click "Image..Image registration".
2. Read the data points for reference image. This is a list of all the spots that were found, obtained from Grain Counting in part 1.

3. Repeat for unknown image.
4. Save match table - if something goes wrong or you wish to reanalyze, you can just click on "Read match table" (optional).
5. Create a landmarks file by taking a subset of the most salient points from the reference and 'unknown' data. (See below for format). These can either be a list of manually aligned spots, or an uncorrelated list of the most significant spots. The landmarks file should be a list of at least 6 of the most reproducible spots on each image, or at least 6 aligned spots.
6. Read the landmarks file. If the landmarks file is a list of manually obtained landmarks, proceed to step 8.
7. Correlate points - If the landmarks file is a list of the most significant spots, but the correspondence between the spots is unknown, they must be correlated before generating a vector map. `tnimage` will attempt to calculate the correspondence using pattern-matching. The spots from the unknown image will be reordered so that the best statistical matching is shown. As the number of landmark spots increases, the difficulty in determining which unknown spot corresponds to which reference spot increases by n^4. Thus, best results are obtained if the total number of landmarks to be determined from each image is kept below 100.
8. Calculate vector map - uses landmark spots from the previous steps to create a mapping of the 'unknown' image onto the reference image.
9. The calculated shifts will be shown on a new image, with changes in red indicating horizontal shifts and changes in blue indicating vertical shifts. Thus, the color code would be:
   - Green = large left shift
   - Red = large right shift
   - Yellow = large up shift
   - Blue = large down shift

   Mixed vertical and horizontal shifts are indicated by intermediate colors. The landmark points and their vectors are superimposed on the map.
10. Referring to the displayed vector map, click "Edit matching table" and edit the match table to remove any incorrect links (see figure below). The vector map should be a smooth gradient of colors. All the displayed vectors should be roughly parallel to the vectors nearby. Any points containing two or more vectors should be inspected and the incorrect link removed in the editor. Any vectors wildly different from their neighbors should be inspected and removed if necessary.
11. Recalculate vector map and repeat the previous step until an accurate and reasonable vector map is obtained.
12. Once the correct matching has been obtained, click "Save" in the matching table editor to save the matches. This file can also be reloaded later if desired.
13. Create unwarped spot list - Changes the x,y coordinates of the data points from the unknown image to match the reference image. (Note that this does not warp the image.)
14. In the Spot List Editor, click 'Save' to save the corrected spot coordinates for part 3.
15. The spot list can be used to create a composite image displaying the spots lined up in a rectangular array (sec.5.5.1).



Part of a vector map showing an incorrect link that should be removed. The vectors point from landmark points on the unknown image to landmark points on the reference image. The numbers are user-specified point labels; black labels are from the unknown image, and white labels are from the reference image. In this example, the data files were created by grain/spot counting, so the labels are all numbers. The link from 11 to 28 is incorrect because it has a markedly different size and angle than its neighbors. The "28" is also shown in red because its coordinates were found in the landmarks file but not the data file. The link from 5 to 16 may also be incorrect.

2D Gel Analysis - part 3: Measuring the signals in each spot

1. Reload the original, unmodified image.
2. Click on "Spot densitometry"
3. Set pixel calibration, maximum signal, calibration factor, and background value or automatic background as desired (see Sec 7.9 for details).
4. Select data source:
   (a) Spot list from image registration (kept in memory)
   (b) Disk file in 'spot list' format (see below for details)
5. Click 'Accept'. `tnimage` will automatically perform densitometry on each spot and create a list.
6. Click "Save" on the editor box to save the results.

NOTES

- The data points should be loaded before creating the vector map from the landmark points in order to estimate the correct size to make the vector map. Otherwise, the vector map may be too small. If this happens, load the data points and click "Calculate vector map" again.
- The distance metric used in creating the vector map is appropriate for images of objects that have undergone elastic deformation or some equivalent shape change, such as 2D polyacrylamide gels being warped by temperature gradients. Some other types of images may require a different method for calculating the vector map. Contact author for assistance.
- Deleting links: If the numbers of points in the two data sets are unequal, some of the data points in one set will be closest to two or more points in the other set. Although most of these extra links are automatically pruned, tnimage is very conservative about pruning links. This means that occasionally, a data point will have 2 or more vectors pointing to it. It is usually obvious by inspection which one is incorrect. Deleting the entire line from the match table editor will remove the incorrect link and the incorrect data point. If you wish to keep the incorrect data point but remove only the link, change

the x and y coordinates of its matching point to zeros. On the next recalculation, the point will still be plotted, but will be listed as an “Unmatched point”.

- Currently only rectangular region spot densitometry is supported. It is recommended to set ‘Auto bkg/fuzzy k means’ if there is an uneven background.
- The editor has a maximum capacity of 10,000 spots and 384K characters. These limits can be increased by changing RPOINTS and EDITSIZE, respectively, in xmtnimage.h and recompiling.

7.4.3 File formats for landmark and data point lists

The data points are stored in plain text files in the same format as the grain counting files produced by `tnimage`. The five-line grain counting header is optional, and is ignored if present. Lines starting with a # are ignored. Data points are stored one per line as shown below, in the order: label <tab> x-coordinate <tab> y-coordinate <tab> value. The label can be any string, but cannot contain white space. The x and y coordinates are distances in pixels from the upper left corner of the image. The data points do not have to be in any particular sequence.

All lines containing non-data must start with a #.

File format for single sets of data points

There are two data files: one for the reference image, containing the coordinates and labels of all the known spots, and one for the unknown image, containing all the data points to be remapped. The first 9 columns of each file are used. Columns are separated by white space. If a column is empty, values are set to 0. Columns cannot be skipped. A complete table is shown below. The columns for `minx, miny, maxx,` and `maxy` are used for densitometry. If the `minx, miny, maxx,` and `maxy` columns are omitted, `tnimage` asks for a spot size. Enter the width in pixels for the spots. In this case, all the spots will be assumed to be the same size.

Example data file with 5 columns

```
# Size analysis of /home/tjnelson/data/30a/30a38a-filtered5.tif
# Largest signal     2793
# Most frequent size 23 (count=7)
# Largest object     457
# Label       x pos   y pos   Size    Signal
# All the above lines are ignored
calcineurin  566     149     149     422
trypsin      618     176     325     1018
S6_kinase    575     194     86      139
36           698     209     373     1015
49           694     238     413     1468
```

Example data file with 9 columns The additional columns are produced by `tnimage`’s grain counting and are essential if the results are to be used for densitometry (The minima and maxima are the bounding box of the spot). Any calibration data is included in columns 10-12.

```
# Obj.  x pos   y pos   Size    Signal   Min.x   Min.y   Max.x   Max.y
1       158     6       18.02   20.10    155     5       162     12
2       152     13      5.12    5.4      151     12      156     15
3       262     18      11      11.5     260     17      265     23
```

In the above example, spot “1” is centered at (158, 6), and fits in a bounding box between (155,5) and (162, 12).

File format for 2 sets of data points (a.k.a. match table) (Created by “Save match table”) These files contain the two original data sets. If you clicked “correlate points”, the Unknown data set will be permuted so that identified corresponding points are on the same line. Any unmatched data points are listed at the end.

```
#Ref Ref.x Ref.y  Unk Unk.x Unk.y xmin1 ymin1 xmax1 ymax1 xmin2
1    158   6      1   158   6     155   5     162   12    155
2    152   13     2   152   13    151   12    156   15    151
3    262   18     3   262   18    260   17    265   23    260
4    276   18     4   276   18    275   17    278   21    275
5    238   22     5   238   22    234   21    243   30    234
#Unmatched reference points
11   272  133  None   0     0     259  132    297  158      0
# Unmatched unknown points
None   0   0      3   1170  101     0    0      0    0   1167
```

(continued)

```
ymin2 xmax2 ymax2 Score
5     162   12    20
12    156   15    5
17    265   23    11
17    278   21    3
21    243   30    35

 0     0     0     0

100  1178  114    0
```

Unwarped Spot list and **Warped spot list** - contains spot name, spot x, spot y, xmin, ymin, xmax, and ymax of each spot after remapping (Created by “Create warped spot list” or “Create unwarped spot list”). This format is the same as the original data files with the addition of an extra column for Identity. This means files of both types can also be used for spot densitometry. The unwarped spot list shows the spots in their original locations, along with their identification. This would be the appropriate file to create if you did not warp the image. The warped spot list shows the spots in their new locations after warping. This would be the appropriate file if you warped the image.

```
# Unwarped spot list
#Label   orig.x  orig.y  Size  Signal  xmin  ymin  xmax  ymax  Identity
1          398    97      374   1103    395    96   402   103   9
spot_21  1905    117       32     39   1888    98   1905  117   cytochrome_C
spot_22  1509    153       70    112   1491   134   1509  153   actin
25       1817    157      123    177   1802   139   1817  157   protein-44
27        679    185      133    269    659   165    679  185   1234
```

This means spot “spot_21” is centered at (1905, 117), is bounded by a rectangle from (1888, 98) to (1905, 117), and was calculated to be the same as the cytochrome_C spot in the reference image.

Landmark list containing corresponding points and locations This file shows the label and x and y coordinates of each known landmark and of its counterpart in the unknown image. The file has to be created manually from the data point files. If the point correspondences are unknown, clicking "Correlate points" will rearrange the right 3 columns to put matching points on the same line.

```
#ref    ref x   ref y   unk     unk x  unk y
20      494     145     40      461     116
28      503     185     49      468     152
29      486     189     53      452     163
46      430     227     65      376     196
42      462     221     67      418     198
49      694     238     68      653     200
```

In the above example, spot "20" on the reference image is centered at (494, 145), and corresponds to spot "40", which is centered at (461, 116) on the unknown image.

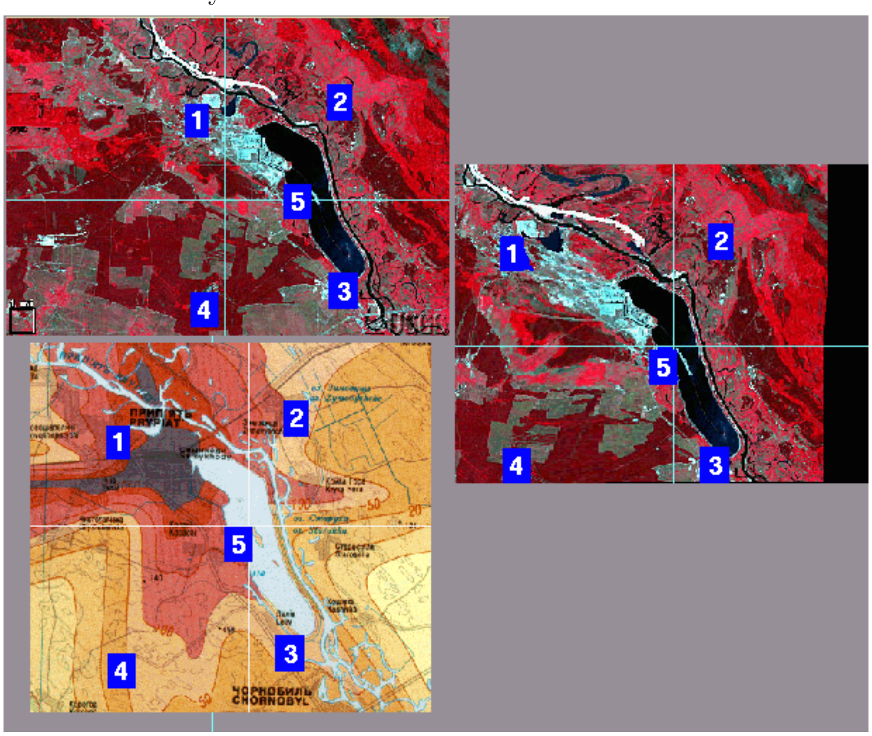
7.4.4 Tutorial on image registration

This tutorial will demonstrate how to register two images by aligning a map of Chernobyl to a Landsat image.

1. Open the files chernobyl1.tif.gz (a map of the Chernobyl area) and chernobyl2.tif.gz (a small version of a Landsat image of the same region from the U.S. Geological Survey).
2. Click on the edge of chernobyl2.tif to select it and move it so both images are visible.
3. Open the Image Registration Dialog and click on “Img 1 Obtain landmks”. Then click on the Chernobyl map at the points labeled 1 to 5 (the two small lakes, the south end of the large island, end of the road, and the island). Press Space when finished.
4. Click on “Img 1 Obtain landmks” and click on the Landsat image at the points labeled 1 to 5. Press Space when finished.
5. Click on “Edit Landmarks” to view the data points. It should resemble the table below:

# Ref.	Ref.x	Ref.y	Unk.	Unk.x	Unk.y	xmin1	ymin1	xmax1	ymax1
	96	86	-	206	93	0	0	0	0
	280	59	-	350	73	0	0	0	0
	275	295	-	349	266	0	0	0	0
	101	314	-	209	282	0	0	0	0
	223	185	-	305	172	0	0	0	0

6. Click on ‘Calculate warp vector map’. A blue-green image will be created.
7. Click ‘Warp image’ to warp the Landsat image.
8. Click ‘Config..Configure’ and set the Main Cursor to ‘multiple crosshairs’ and verify that the registration points are at similar locations. If the crosshair cursor is too dark, click ‘Fcolor’ and change the main drawing color.
9. After warping, the landmark points are automatically remapped. Click on ‘Calculate warp vector map’ and ‘Warp image’ again to perform a closer approximation. More repetitions can be performed until the two sets of registration points become identical. This usually occurs after two to three iterations.



Warping of Landsat image (upper left) to a map (lower left). The five control points are labeled. In the final image at right, the landsat image has been warped so image coordinates of every feature are the same as the image coordinates of every feature on the map.

7.5 Measure

Distance measurement

Measures distance between two points. Click the left mouse button, move to the end point, and release the mouse. Click on the top menu bar or press *ESC* when finished.

Before making distance measurements, the image can be calibrated as described in Sec.7.7. This will cause the results to be in known units (such as centimeters) rather than pixels.

If the image has been calibrated in a "2D" mode, calibrated values are presented for both the x and y dimensions.

Clicking the "Clear" button removes the results from the list. The list items can be edited by clicking on the item.

Angle between 2 lines

Measure the angle between 2 lines. Same as distance measurement except it is necessary to click and drag twice. The angle is always given as the most acute of the two angles between the two lines.

Angle of 1 line

Calculates the angle of a line in degrees from horizontal.

Length of segmented curve

Measures total distance along a curve comprised of short segments. After selecting this option, create a segmented curve by clicking at several points along the path to be measured. The program will place a small box at each point, and will connect the boxes with line segments. These boxes can be moved to different positions by dragging them with the mouse at any time.

When you are finished creating and positioning a curve, press the space bar. This will erase the boxes and line segments, and present the total length in a dialog box. The Information box will then reappear, allowing you to select another curve. When finished, click the Cancel button to end measurement mode. A message box is then displayed asking whether to save the list contents in an ASCII file.

Before making distance measurements, the image can be calibrated as described in Sec.7.7. This will cause the results to be in known units (such as centimeters) rather than pixels.

If the image has been calibrated in a "2D" mode, total lengths are presented for both the x and y dimensions (this is not terribly useful).

Points

Creates a list of points. This consists of 7 columns:

1. Measurement number
2. Uncalibrated x value
3. Uncalibrated y value
4. Uncalibrated pixel value
5. Calibrated x value
6. Calibrated y value
7. Calibrated pixel value (z value)

Shift

The shift function moves an image left/right or up/down within its image buffer. This is useful for multiframe images, to align individual frames that may have been pasted inaccurately.

7.6 Chromatic Aberration

Chromatic aberration is caused by different refractive indices in the lens for different wavelengths. This is encountered frequently in refractive telescopes and telephoto lenses and in microscopes not equipped with achromatic lenses. There is no simple, general method for correcting for chromatic aberration. However, we can come close by making certain assumptions about the lens.

Chromatic aberration can be manifested in three different ways:

1. One color channel (frequently blue) may be blurred, because it focuses in front of or behind the detector. This can be repaired to some extent by sharpening the blurred color.
2. One color channel may be shifted in the x or y direction. This can result from inaccurate registration of multichannel images. This would cause red fringes near one edge of the image and blue color fringes near the other edge.
3. One color channel may radially distorted, so that in effect it is magnified slightly. This would also cause red or blue color fringes near the edges of the image, but in a symmetrical pattern. For example, the outer edges of all objects might have red fringe.

To correct this, `tnimage` can remove four types of distortion: shifting, linear stretching, radial stretching, and blurring. Each of these uses a parameter that must be determimed empirically for each image.

Shifting is done by moving all the pixels of one or more colors in the x or y direction. Negative x values correspond to "left", and negative y values correspond to "up". If zero is specified, no change is made. Typical starting value: -1.

Linear stretching is done by enlarging the R, G, or B channel by a specified factor, keeping the center point constant. A factor of 1.0 means no change for that color. The x and y stretch factors for each color can be specified independently. Typical starting value: 1.01.

Radial stretching is done by increasing the distance of the R, G, or B pixel from the center point by a specified factor. A factor of 1.0 means no change for that color. Typical starting value: 1.01.

Second-order radial stretching is not yet implemented.

Sharpening is done by applying a high-pass filter to the R, G, or B component. A value of 0 means no sharpening is done, while a value of 100 means maximal filtering. Typical starting value: 2.

Erosion makes objects of the specified color smaller. This is experimental. 0=no erosion 100=maximum erosion. Typical starting value: 1.

Notes

Images must be at least 16 bits/pixel for chromatic aberration correction to work.

Repeated stretching and shrinking of an image will eventually cause loss of resolution. Use the "undo" function to restore the image while searching for the correct parameters.

Excessive sharpening of one color may affect the overall color balance of the image.

Excessive erosion will make small objects disappear entirely.

7.7 Calibrate

Calibrates the image x and y coordinates and pixel values.

'x units' and 'y units' both calibrate x and y coordinates to user-specified calibration values. The coordinates are fitted to a straight line or a polynomial curve depending on the setting of "No. of terms":

No. of terms	Type of curve
1	Straight line
2	Quadratic curve
3	Cubic curve

The coordinates are also fitted to the calibration values that you enter using the same type of curve.

The calibration units are labeled 'x' and 'y' for convenience, but in fact they are independent of each other and can be in any direction, including any diagonal direction, or can even be parallel to each other. The 'x' and 'y' calibration is useful if the position of a point on the image corresponds to some parameter, such as molecular weight, longitude, etc.

'z units' calibrates pixel values only, and is independent of the pixel's location. This is useful when pixel value represents some position-independent parameter such as altitude, height, temperature, etc. Densitometry results can also be calculated in terms of calibrated z values.

If you change a calibration point by clicking "Edit data", or if you change one of the fitted parameters in the boxes (such as 'y intercept'), a new calibration curve will be calculated when you click "Recalculate". The new calibration values take effect immediately.

Each image can have a different x, y, and z calibration curve.

Procedure for calibrating an image:

1. Click on the image to select it.
2. Select "Calibration" from the menu.
3. Set Non-directional or Directional calibration type.
4. For each dimension to calibrate, select the equation which will be used to fit the data (linear, logarithmic, polynomial, or distance from 0).
5. Click on "Obtain data" to select calibration points from the image, or enter the desired parameters if already known.
6. When calibrating from calibration points, click at the points on the image where the x, y, or z value is known. A small box will appear indicating the control point. This box can be dragged to a different location.
7. Continue clicking until all the desired control points have been acquired, then press any key or space bar to finish.
8. A small spreadsheet will appear allowing you to edit the x and y coordinates and to enter the calibration value for each point. When calibrating the z dimension, the spreadsheet has two columns instead of 3. In this case, enter the calibration value that corresponds to the indicated pixel value and edit the pixel value if necessary.
9. Click OK. The image is automatically calibrated. The f value and correlation coefficient for the curve fitting are shown.
10. Repeat with other dimensions if desired.
11. The calibration points can be modified by clicking on "Edit data".
12. The calibration parameters can also manually altered. Any changes in these boxes take place immediately. Clicking on "Recalc from data" will recalculate the calibration parameters from the data points.
13. The data points and calibration parameters can also be read from and saved to a disk file by clicking on the corresponding button.

Each image, as well as the background, can be calibrated separately and can have a different title. The calibrated value is shown in the left information panel below the Relative Coordinates display.

Several types of calibration are possible:

1. Non-directional linear calibration: The distance between the starting and ending points is multiplied by a conversion factor to give the calibrated value. This calibration is useful for morphometric measurements on images, where the units are independent of the angle.
2. Directional linear calibration: Measures the distance from a line perpendicular to the best-fit line passing through your calibration points. This calibration is useful for images of isoelectric focusing gels, or other situations where the distance in one direction is the parameter of interest.
3. Logarithmic calibration: Same as (2) except distances are logarithmic (For example, if 10 pixels corresponds to 1 unit of distance, 20 pixels would be 10 units, and 30 pixels would be 100 units). This calibration is suitable for acrylamide and agarose gel electrophoresis images.
4. 2nd-order polynomial calibration: Same as (1) except distances are fitted to a quadratic equation. Could be useful.
5. Distance from 0: Distances are calculated as the distance from a (0,0) point somewhere in the image. This calibration type is suitable for calibrating to objects of known size on the image, for example if a ruler was included in the image.
6. Exponential: pixel values raised to some exponent. This equation is often used as 'gamma correction'.

For example, to calibrate a 2D SDS acrylamide gel, set the calibration type to "Directional", calibrate the first dimension from the molecular weight standards (using a logarithmic function), and then calibrate the isoelectric point from the IEF standards using a linear function. Set the Units to "MW" and "pI" respectively.

Example 1: Calibrate an SDS-PAGE gel for molecular weight standards.

1. Select "Calibration".
2. Select "logarithmic" and "Directional" calibration from the dialog box.
3. Check to ensure the correct image number is shown.
4. Change units to "Molecular weight" (optional).
5. Click on "Obtain data" to start calibration.
6. Click on each size standard in the image. Each point will be shown by a small box. In case of a mistake, you can go back and drag any of the boxes to a new location. A best-fit line is automatically drawn connecting the boxes. Make sure that this line is in the same orientation as the lanes in your gel (i.e., vertical or near vertical if your lanes are straight).
7. When finished, press a key. The Data Entry dialog will appear. Enter the corresponding molecular weights in each box.
8. When finished, press Enter or click on OK. The dialog box will disappear.
9. The molecular weight is now displayed continuously.

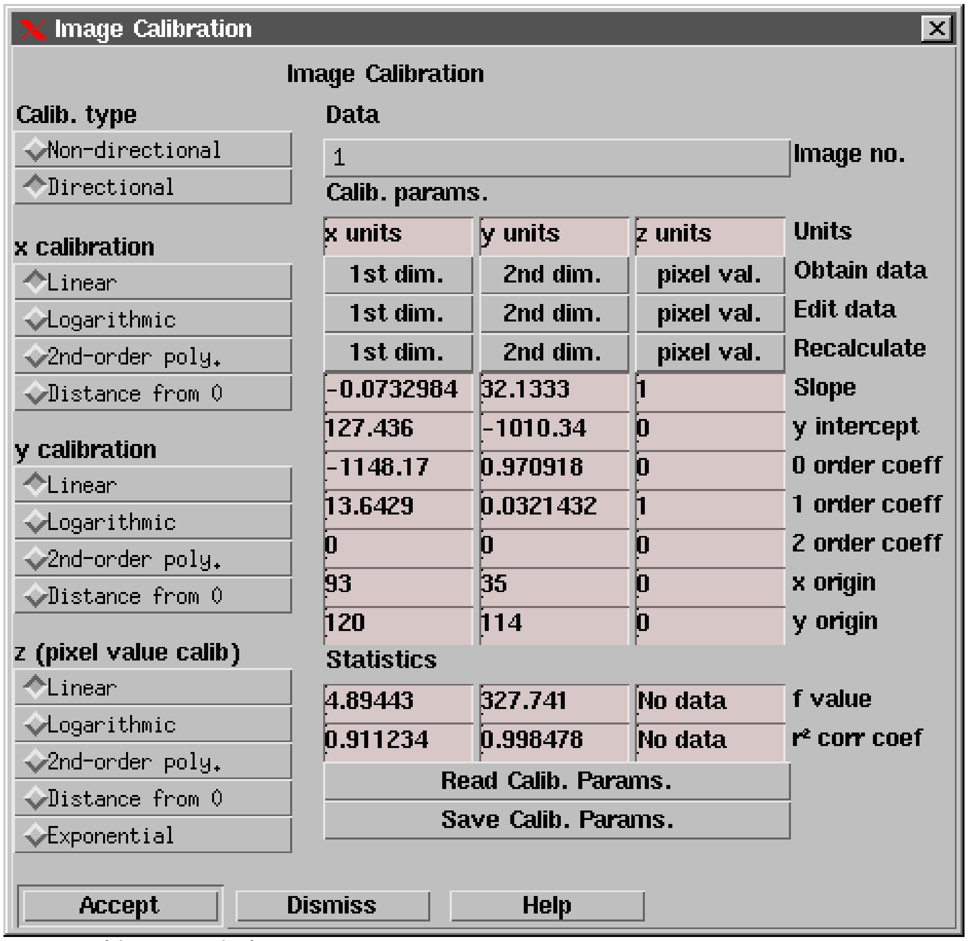


Image calibration dialog

Example 2: Calibrate coordinates on the image to centimeters using a ruler in the image.

1. Select "Calibration".
2. Select "Distance from 0" and "Non-directional" calibration from the dialog box.
3. Check to ensure the correct image number is shown.
4. Click on "Obtain data" to start calibration.
5. Click on the ruler markings on the strip in the image. A small box indicates the calibration point. Move the box by clicking-and-dragging if necessary.
6. Press a key when finished, then enter the centimeter value.
7. When finished, press *Enter* .
8. The image is now calibrated. Each image, as well as the background, can have a different calibration. The distance is measured as the distance from 0,0 in any direction as defined by the calibration points. Calibrated distance measurements can now be performed.

Notes:

1. If the image is calibrated from data, at least 3 points are required. Values less than or equal to 0 are not permitted for logarithmic calibration.
2. If the parameters are already known, they can be entered directly (see below). It is not necessary to acquire calibration points in this case. The 'statistics' fields will remain blank.
3. The control points do not need to be obtained in any particular order.

4. The x, y, and z calibrations are calculated independently, using separate parameters. This means the calibration axes do not need to be orthogonal with respect to each other or to the screen axes. They also do not need to be oriented in any particular direction. This means, for example, that you could calibrate both the 1st and 2nd dimensions to different scalings in the same direction to indicate inches and centimeters in the same direction.
5. The z calibration is different from the x and y calibrations in that only the pixel values are considered, and is independent of the x and y coordinates.
6. The calibrated distances are also used for distance and angle measurements and strip densitometry area calculations. Z calibration can also be used to calibrate densitometry density measurements.

Entering calibration coefficients directly

If the calibration coefficients are already known, they may be entered directly into the dialog box. For calibrating pixel values, `tnimage` interprets these parameters as follows:

Calibration type	Equation
Exponential	answer = q_0 + pixel^{q_1}
Logarithmic	answer = q_0 + 10^{pixel}
Linear	answer = q_0 + pixel$\times q_1$
Polynomial	answer = q_0 + pixel$\times q_1$ + pixel^2 $\times q_2$

where q_0 to q_2 are the 0, 1st, and 2nd order polynomial coefficients of the best fit line and 'pixel' is the pixel value. For calibrating x and y coordinates, `tnimage` interprets the parameters as follows:

Calibration type	Equation
Exponential	answer = q_0 + x^{q_1}
Logarithmic	answer = q_0 + 10^{x}
Linear	answer = q_0 + x$\times q_1$
Polynomial	answer = q_0 + x$\times q_1$ + x^2 $\times q_2$

where x is the distance from the origin point in pixel units. The x and y origin points are calculated from

$$x_o = first\_x\_data\_point$$

$$y_o = q_0 + first\_x\_data\_point \times q_1$$

The slope and intercept of a line perpendicular to the best-fit line are also shown but have no effect on the calibration if they are changed.

For example, if the 1st order coefficient in z is set to 1.3 and the z calibration is set to 'exponential', the program will use (pixel value)$^{1.3}$ instead of the pixel value in density calculations.

Calibrating images based on a reference image

Often it is necessary to calibrate a series of images to a known standard, such as a scanned image of a ruler. This can be done as follows:

1. Calibrate the standard image as described above. In this example, the calibration type (i.e., "Calibration for x") would be "1-D Non-Directional".
2. Select "Image Properties...Copy tables" and select "Copy calibration". Set the "Image to change" to the image number of the new image, and "Copy from" to the image number of the standard image.
3. Click OK. The new image now has the same calibration as the standard.

7.8 Area Measurement

Area measurement is done using spot densitometry (Sec. 7.9). The region to be measured can be selected manually or automatically using a variety of methods. Considerations for measuring areas are identical to those used in spot densitometry. Area measurement results are displayed in the same window as densitometry results.

7.9 Spot Densitometry

Performs densitometric analysis on parts of the image. See also *Strip densitometry* . For a tutorial, view the image DENSITOM.PCX.

Area selection

Which area selection method is most appropriate? There are several ways of selecting the area to measure. All of these methods can quantitate arbitrary irregular shapes such as spots, bands, or anatomical features in an image, but they differ in the details of how this is accomplished and in the degree of automation.

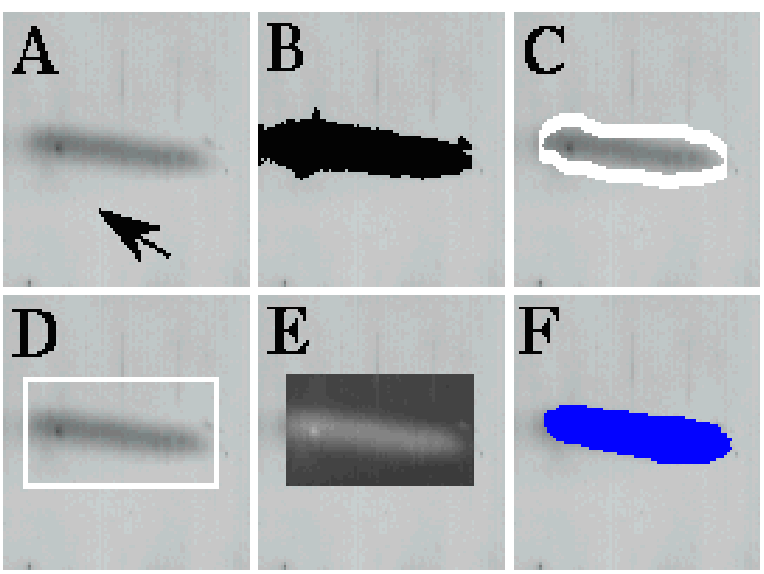
7.9.1 Automatic

In automatic area selection, the computer decides where the boundaries of the object to quantitate are, based on the background value that you provide. The background value is a number between 0 (black) and 1 (white). This number is found by moving the mouse over the background near the feature to be measured and observing the "d" (density) value shown in the left information window.

For example, in panel **A** below, the intensity of the background (pointed to by the arrow) ranged from 0.702 to 0.765 while the intensity inside the band was around 0.482. Thus, the best background value would be the background value that is closest to the signal, i.e., 0.702. So in the Spot Densitometry dialog, you would set the following:

Maximum signal	`Black` (since the band is black).
Area selection	`Automatic`
Pixel density calib.	`None`
Bkgd.value	`0.702`
Calib. factor	`1`
Diameter	*(ignored)*
Leave area marked	*(unchecked)*
Auto bkgd. (fuzzy k means)	*(unchecked)*

Keep Auto bkgd. unchecked, since you wish to use a constant background value.



Then click OK, and click in the center of the band. The boundaries of the band are automatically calculated, and the band is temporarily highlighted (**B**), indicating precisely what is being measured. If this is too large, the Bkgd. value should be changed. The algorithm is designed so that the computer should find the same region, and the same area, density, and signal, regardless of the exact spot you click on within the object. If you click on the background, the program will report an area of 0. As long as you continue clicking, new areas will be selected and analyzed. At any time, clicking on Cancel or pressing Esc will stop the densitometry.

Sometimes the computer may not find the correct edge of the object. It is important that there is no unbroken trail of pixels the same color as the object that extends out of the object. If there is no clear boundary around the region of interest, it may be necessary to create one before starting densitometry by sketching a black or white border around it (for example, **C**).

7.9.2 Manual

Manual rectangular:

In this mode, for each measurement, select (by clicking and dragging) a rectangular region around the object (**D** above). When you un-click the mouse, the entire rectangle will be highlighted (**E**) and the entire area within the rectangle will be measured. The reported area will be the area of the rectangle. However, since the background is still set to 0.702, only the pixels in the spot, where the image is different from the background, will contribute to the signal and density results. Thus, if the background is set correctly, the reported signal and density should be the same regardless of the exact size of the rectangle.

In Manual Rectangular mode, it is sometimes helpful to select several background areas first, and adjust the background level, until selecting a rectangle on the background gives a density close to zero.

Alternatively, click on "Auto bkgd" to allow the program to automatically calculate the background value each time (see below). Make sure to include a small portion of the background with each measurement if you do this.

Densitometry will continue as long as rectangles are selected. It is not necessary to click OK on the Result Window (this actually has no effect). To stop, click Cancel.

Fixed size rectangular:

This mode is similar to Manual Rectangular mode, except that all measurements are guaranteed to be the same area. Before the first measurement, select a rectangular region. An information box will appear saying "Ready to begin densitometry on (*size*) regions". From then on, as long as densitometry is active, clicking on a single point will cause a region of size and shape identical to the first region to be be analyzed, centered at the point where the mouse was clicked.

As with Manual Rectangular, if Auto Bkgd. is checked, the background level will be calculated automatically each time. In this case, it is important to make the rectangle large enough to include a small amount of the background, so that the calculation is accurate (ideally, the area of background and the area of signal should be the same).

Manual Irregular:

This mode gives the maximal amount of control to the user. In this mode, you must manually draw an outline around the feature to measure. When you un-click the mouse, an Information Box appears saying "Click on any point inside selection". This means the program needs to know which part of the area you sketched is the inside. Clicking on the inside causes the inside region to be temporarily colored blue (**F** in the figure above). When you are finished selecting the area, the blue will disappear, and the outlined area will be automatically measured.

This mode would typically be used for morphometry and area measurements, in which case it is desirable to avoid including background pixels in the selection.

If you want to measure multiple disjoint areas simultaneously, change the `SELECT_MODE` const in `xmtnimage20.cc` from `SINGLE` to `MULTIPLE` and recompile `tnimage`. The only difference will be that, instead of one blue area (**F** above), you can have any number of them. When you are done selecting blue areas, click the main Cancel button. The densitometry results will be reported for the union of all the sets of points.

Color images should be converted to grayscale before performing densitometry. Color densitometry is not yet supported.

If "Auto calculate background" is selected, you must include an approximately equal number of foreground and background pixels so the fuzzy k-means algorithm can calculate the correct background. It is recommended to keep "auto background" off and use a fixed background value when measuring irregular areas.

During this mode, pressing Ctrl-V will permanently mark the selected area.

Fixed Shape Irregular:

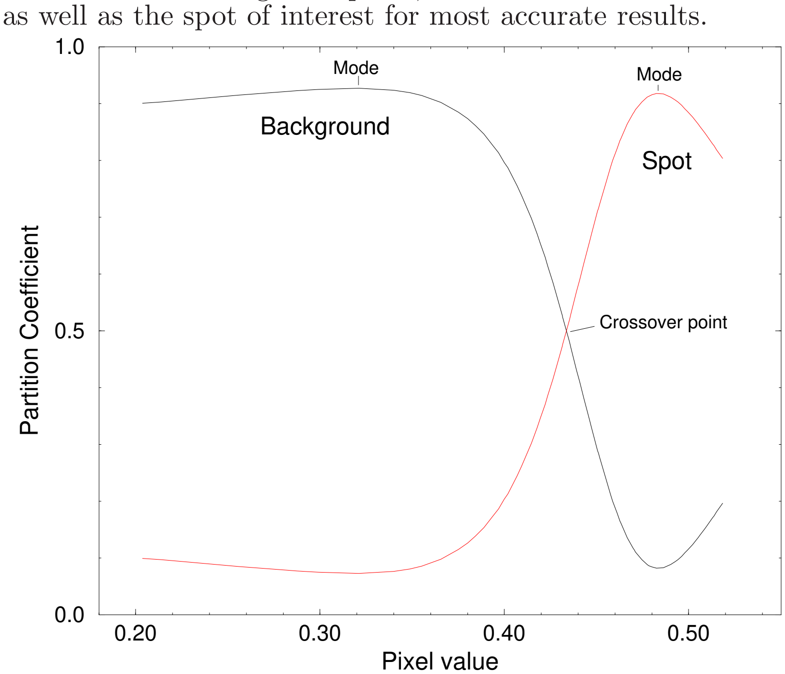
Same as `Manual Irregular`, except the shape needs to be defined only once. Subsequent mouse clicks will measure the same shape centered at the new location.

Note: The irregular shape is automatically prevented from being placed too close to the edge of the image.

During this mode, pressing Ctrl-V will permanently mark the selected area.

7.9.3 Automatic Background Calculation

If checked, the background value in the vicinity of the area being measured is calculated automatically. This obviates the need to adjust the background manually in images having uneven backgrounds. The algorithm performs a fuzzy k-means analysis on the area being measured to calculate the background density (*see below*). This gives sub-pixel-value accuracy, which is much greater than could be achieved by manually specifying (or allowing the computer to estimate) a single integer background value. It is therefore recommended to keep this option checked. When using this option, the area selected should contain a portion of the background as well as the spot of interest for most accurate results.



Typical distribution of fuzzy partition coefficients for a region of an image containing a spot and a portion of background. The centroid values and crossover point are shown.

If "auto calculate background" is checked, the foreground and background pixel densities are computed from the selected region by the fuzzy k-means algorithm [2–5]. As shown in the above figure, this algorithm estimates the contribution to each pixel from the foreground and background clusters. The centroid (modal value) of the resulting population of partition coefficients is the most probable density of the pixels in each cluster. If the algorithm converges, a table containing statistical information about the analysis is also displayed. The most useful parameters are:

- *Demarcation point:* Pixel density at border between the two clusters
- *Centroid (mode) density:* Pixel density maximum of each cluster
- *Probability of 2 or more clusters:* Approximate measure of confidence that two distinct clusters of pixel intensities have been detected. A low value (near 0) indicates a faint, indistinct spot, while a value near 1 indicates that two or more populations of pixel intensities (not necessarily in any given shape) were observed.
- *Net difference in density maxima:* A measure of the difference in density of the spot vs. the background. A spot that is sharply focused will have a higher density difference than a fuzzy one, even if the total signal in both spots is the same.
- *Pixels in cluster 0:* No. of pixels classified as part of the background.
- *Pixels in cluster 1:* No. of pixels classified as part of the spot.

The densitometry results are shown in a list, which can be saved into a file by clicking the "save" button. The list displays:

- *No.:* Measurement number
- *x:* X value at center of region, or at point clicked. Corrected for calibration.
- *y:* Y value at center of region, or at point clicked. Corrected for calibration.
- *Area:* Total number of pixels in the selected region.
- *Total signal:* The sum of all pixel values in the signal. That is, if there are 1000 pixels, each of which had a value of 0.5, the total signal would be 500.
- *Spot density:* Average density of all pixels classified as part of the spot.
- *Total background:* Calculated (or specified) background pixel density multiplied by the no. of pixels classified as part of the background.
- *Bkg. density:* Calculated (or specified) background pixel density.
- *Net density:* Pixel density of spot minus pixel density of background.
- *Signal − bkg.:* Integrated signal in the entire spot minus background in the spot. (i.e., net density × no. of pixels in spot). This is the number to use when analyzing images of SDS or agarose gels, in which the entire spot represents a quantity of some substance.

Because the background density is calculated from all the pixels, background values can have sub-pixel accuracy. For this reason, results obtained with this method are significantly more accurate than those obtained from programs that use a median value or some other integer as the estimate of background density. For example, I was once asked to analyze an 8-bit image of a Western blot that had bands with pixel values averaging about 211 (on a scale of 0...255), while the background was in the range of 206-209. The person did not want to repeat the experiment, since these were autopsy samples, and had first tried to analyze the image with another well-known image analysis program, which selected integer values for the background. When the other program happened to use 209 instead of 208, it resulted in an error of 50% because of the inaccuracy in the background estimate. Even after taking multiple measurements and discarding those which gave negative numbers, it was impossible to get an accurate measurement from the other program. Increasing the image contrast merely increased the error by the same factor. Using the fuzzy k-means method here, the background could be determined to several decimal points, and clear-cut results were easily obtained.

Maximum signal

Selects whether pixel value corresponding to 'black' or 'white' (0 or 255 in 8 bpp mode) is to be considered the strongest signal. If you have an image of a protein gel, for instance, the bands appear black and maximum signal should be set to "black". For DNA gels, the bands appear white and maximum signal should be set to "white".

Note: Selecting "Black" as the maximum signal will cause the reported values to be calculated as 1 – (measured value). Thus, if the average uncorrected pixel intensity of a spot is 0.25, the density (with "pixel compensation" off) will be 0.75.

Background

Select the fractional pixel value (between 0 and 1) which is to be considered the 'background level'. The background level is important in automatic area selection as described above. The background value will be automatically subtracted from the results.

Pixel density calibration

Making density measurements on an image requires some way of relating pixel values to the real quantity being measured. Ordinary scanners do not produce a linear relationship between pixel value and the quantity being measured. Even expensive densitometry imaging equipment and analysis systems, such as MolecularDynamics' PhosphorImager, may not yield linear relationships between band "density" and concentrations. This is particularly true for 8-bit images, in which a wide signal has been compressed into only 256 different values. Some software packages use a gamma correction function, which assumes an exponential relationship between signal and pixel value. However, general-purpose software should not attempt such corrections, because many people scan images from nonlinear sources such as film autoradiograms or HRP-stained Westerns, all of which have different inherent response characteristics. When the physical nonlinearities are combined with the nonlinearities in the image acquisition process, the relationship is too complex for accurate results to be obtained using a simple exponential relationship.

For example, exposing a grain on X-ray film or emulsion requires two photons striking a grain within a short period of time, which means that at low light levels, the signal on the film is related to some power of the photon flux. However, at high levels, the unexposed grains become depleted, making the signal independent of light intensity. These factors are highly dependent on numerous factors including temperature, film speed, etc. For Western blots, the production of a colored signal by the immobilized enzyme follows time, temperature, and concentration-dependent hyperbolic saturation kinetics. Clearly, the only accurate way to estimate the true signal under such conditions is by using standards.

Options:

OD value If checked, this will cause `tnimage` to use the value in the optical density table in its calculations instead of the raw pixel value. This information is sometimes provided by digital scanners in the form of a 'gray response curve' or 'gamma curve' that is embedded in the TIFF file. For most images, this option will have no effect since no gray response curve is present. However, you can easily create a curve by clicking "Config..Show OD table" and modifying the graph to create a curve of any desired shape. Since this curve has only 256 elements, it is only applicable for 8-bit grayscale images.

z units If checked, this causes `tnimage` to use the calibrated pixel value in its calculations instead of the raw pixel value. Pixel value calibration is performed before starting densitometry, by clicking "Process..Calibration", checking the desired equation to which the 'z value' is to be fitted, and then clicking on each calibration standard in the image (See Sec. 7.7) and entering the known value in the spreadsheet. `Tnimage` performs a linear regression on the data and uses these coefficients to calibrate pixels in the image.

No. of terms Sets the number of polynomial terms used in the linear regression for fitting coordinates to calibrated values. If '1' does not give a satisfactory fit, up to 3 terms can be used. Using higher terms will result in lower accuracy in areas extrapolated beyond your calibration points.

NOTE: For this option, the "Maximum Signal Black/White" setting is ignored, because the user calibration automatically defines what pixel value is the maximum signal.

None In this option, the signal and density are given as raw densities (expressed as a number between 0 and 1). (signal = density × number of pixels.)

Calibration factor If a value is entered here, the result is multiplied by that number. This is useful in converting to micrograms of protein, for example, on an image of an SDS gel.

After each measurement, a list box appears with the results. This contains: (1) The area (total number of pixels) analyzed; (2) The total signal measured (either in total pixel values or total O.D. units, depending on whether pixel compensation is active); (3) The average signal, i.e., the quotient of (2)/(1); (4) The corrected total signal minus background. For most purposes, including gel analysis, the number of interest is (2) or (4), because it is desired to measure the total amount of absorbing (or fluorescing) material in the entire band. For other purposes, the signal density or concentration per unit area on the image may be desired. The total area is of interest in morphometry.

NOTE: See the warning under *Contrast*.

NOTE: Densitometry may not work correctly on zoomed images. Use "Change size" first.

WARNING: Pixels beyond the edges of the screen are not included in the area or density calculations.

WARNING: If a monochrome or 8-bit image is converted to color, the original correspondence of pixel values to optical density (if one existed), may be lost. This will result in inaccurate densitometry results.

7.9.4 Spot densitometry using lists of spots

Instead of selecting each spot interactively, you can use a list of spots from a disk file to make up to 10,000 densitometric measurements automatically. These are plain text files that can be produced by `tnimage` or created manually. It is often convenient to identify the spots automatically, save the results in a file, then perform densitometry on the spots all at once.

Below is the file format for spot lists:

```
#Unwarped spot list
#Label orig.x  orig.y  Size  Signal xmin  ymin  xmax  ymax  Identity
1      430     75      374   1103    424  74    450   108   spot1
2      604     91      32    39      603  90    610   99    2nd_spot
3      466     93      70    112     463  92    471   109   aldolase_A
4      422     101     66    94      411  100   427   114   4
```

Any line not containing data must start with a '#'. The file must have at least 10 columns, separated by white space. The first column is a text label identifying the spot. This label cannot contain white space. The other columns are the center x and y position of the spot (measured in pixels from the upper left corner of the image), the spot's size and total signal, and the bounding box (xmin, ymin) to (xmax, ymax) of the spot. The Size and Signal columns are ignored in densitometry and may be set to 0. The last column is a second optional label containing the 'identity' of the spot.

This file format is produced by `tnimage`'s grain counting and image registration functions.

When a warped or unwarped spot list is created during image registration, the list is stored in memory and can also be accessed in spot densitometry by clicking "Data Source.. Spot list".

It is recommended to use "Auto background (fuzzy k means)" when performing densitometry on lists in case the background is uneven. Currently only densitometry of rectangular areas is supported for lists.

7.9.5 Obtaining the most accurate densitometry measurements

Below are some tips for making the most accurate spot densitometry measurements:

1. Scan the image at the highest depth and resolution possible. Ten bits/pixel and 300 dpi should be considered minimum values.
2. If using "Auto background", select an area approximately twice the size of the spot. Ideally the number of background pixels and spot pixels should be the same. The "Area Partitioning Statistics" window gives this information.
3. Make several repeated measurements and take the average.
4. Although it is more tedious, the most accurate technique is to use strip densitometry instead of spot densitometry. This gives a density profile from which the optimal background value can be subtracted manually. (See figure below in "Strip Densitometry", Sec.7.10).

Notes:

- If all density values come out as negative numbers, either the "background value" or the "maximum signal" is set incorrectly.
- If all density values come out as identical numbers, the most likely cause is low contrast in the original image. This will not necessarily be visually apparent, because an optimal grayscale map is calculated automatically before displaying the image. Try clicking on "Contrast..Maximize Value".

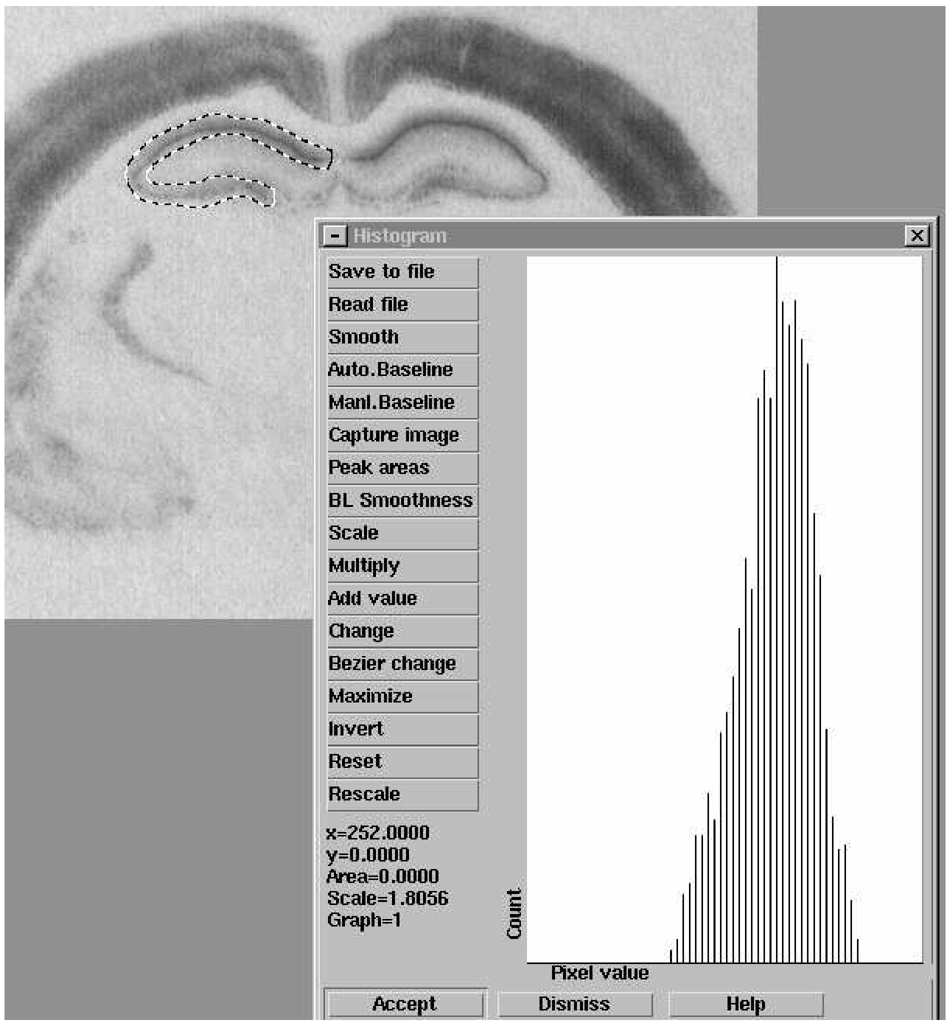
Area measurements

Procedure for making area measurements:

1. Before starting, obtain a histogram on the image to find the desired background value for use with automatic spot detection (select "Color...Histogram"). Alternatively Use "Draw...Sketch" before starting, and completely encircle the objects to be measured with a white line. Outlining the objects manually will obviate the need to select an appropriate background value.
2. Select "Image...Spot densitometry". Leave all the settings at their defaults except for the Background Value.
3. To identify the spot automatically, set the background value to the pixel value that corresponds to the brightest pixel in the background. Alternatively, the regions of interest can be demarcated by an outline beforehand, and the background value set to any large number (for example, if the areas were outlined with white (=255), set the background to 254).
4. Click on OK, then click again to dismiss the message box.
5. Click anywhere inside the object to measure. The area being measured will temporarily be painted in black, then a list box will appear displaying the area.
6. Click on the background (or the upper left corner of the list box) to hide the list box. Then click on the next object to measure.
7. Press *Esc* when done.

7.9.6 Densitometry Measurements in Autoradiograms

The simplest (and wrong) way of measuring the density of a region in autoradiograms is shown in the figure below, which is a film autoradiogram of a rat brain slice incubated with S-35 labeled RNA probe.



1. Press F3 to enter area selection mode.
2. Select the region of interest using the mouse. The area selection mode can be set in the Configuration Dialog as: ‘Single’, ‘Multiple’, ‘Polygon’, or ’Point to point’. The default mode is ‘Single’, in which a single area is selected by freehand drawing of an outline around the region with the mouse. In Polygon mode, the area is selected by dragging small boxes to form a polygon around the area. This has the advantage that you can interactively change the area selection. Multiple mode is similar to Single mode, except that you can select any number of unconnected areas for each measurement. Point-to-point mode is similar to the drawing mode in `xfig`.
3. Release the mouse button and then click inside the area to tell `tnimage` which region is the inside of the area of interest. The selected region will be surrounded by a crawling dashed line.
4. Select “Color...Histogram” to create a histogram plot. The graph will contain at least

one maximum corresponding to the most probable pixel value, as shown in the figure above.

This is a crude way of estimating the density of a region and is often employed by users of Photoshop and similar programs. The result can be made more accurate by clicking on "Smooth" to convert the histogram into a smoother curve, or dragging the mouse on the histogram to produce a summation of all the pixel values, which is shown on the graph as "Area". However, all of these methods are much less accurate than using spot densitometry, because it is necessary to manually exclude part of the histogram from the calculation to exclude the contribution from the lighter background. In a good image, there may be two distinct peaks separated by a clearly-defined minimum; but, as shown in the above example, most often there is only a single large peak and an indistinct shoulder.

It is strongly recommended to use "Manual irregular" mode in "Spot densitometry" (Sec. 7.9.2) to analyze images of this type, rather than this histogram method.

A disadvantage of film autoradiograms is that it is often necessary to expend considerable effort in setting up 12- or 16-bit CCD cameras and expensive light boxes to acquire good quality images. If you use film, a Northern Light light box, costing about $3000, which uses an electronic servo to maintain precisely-regulated light intensity, is recommended to obtain reproducible results. The images can be converted to colorful pseudo-color pictures that give the impression of smooth gradations of labeling. However, this perception is often misleading because it misrepresents the physical process, which is actually a discrete, binary exposure of individual photographic grains in the film or autoradiographic emulsion.

A second source of inaccuracy in film autoradiograms is that using the most probable value introduces errors if the illumination behind the autoradiogram changes. Also, unlike spot densitometry, this method cannot provide sub-pixel-value accuracy.

A better approach is to use a microscope with a 1× objective, attached to a microscope camera. Under the microscope, what originally appears to be smooth gray areas can be seen to be collections of black grains surrounded by unexposed regions. This sort of image is easily analyzed by the histogram technique just described, and gives a large, sharp peak corresponding to the black exposed grains. This obviates the need for acquiring 12- or 16-bit images and using precisely controlled illumination.

The difference in quality between a photomicrograph and an image acquired from a CCD camera with a zoom lens makes it worth the additional effort. Not only is pixel counting of exposed grains more accurate, it is more direct inasmuch as it is closer to a measurement of the actual physical process of exposing silver grains in the film.

Of course, the ideal method for imaging radiolabeled sections is to forget about film and use a phosphorimager. Nowadays, these are available for $10,000 or less and far surpass film autoradiography in terms of speed and resolution.

7.10 Strip Densitometry

See also *Spot densitometry*. For a tutorial, view the image file DENSITO2.TIF. Strip densitometry is not as exciting as it sounds. It is similar to spot densitometry, but is easier to perform and interpret because it is not necessary for tnimage to find the edges of the object.

To obtain a densitometric tracing of a trapezoidal or rectangular region, select “Strip densitometry” from the “image” menu. The options are:

Coordinates

If ‘select coordinates’ is checked, it is necessary to select the region to scan each time.

Repeat prev. scan

If ‘repeat previous scan’ is checked, the same region as the last time will be re-scanned without the necessity of selecting it. This is useful if you modified the image in some way, such as by filtering it, in order to see the effects on scanning.

Maximum signal

Selects whether a “black” or “white” pixel is to be regarded as the most intense value.

Scan type

This option determines what type of region to select for scanning. Selecting ‘Rhomboid, 90° (4 pts)’ causes the starting and ending edges of the scan to “snap” to either vertical or horizontal, whichever is closest. This is fastest because it does not require any of the anti-aliasing calculations which are needed when scanning diagonally. However, if the image contains objects that are oriented diagonally, some resolution would be lost in the scan because the objects will be scanned at an angle to their true orientation.

The first two points always determine the orientation of scanning. Scanning is always done along a line parallel to these 2 points. The 3rd and 4th points determine the direction and shape of the region to scan. The area being scanned is indicated by a sweeping wave of temporarily inverted pixels.

Selecting *Rhomboid (4 points)* allows selection of any trapezoidal area with no restrictions on the starting angle. However, the angle of the far side of the trapezoid is always adjusted to make it parallel to the starting side. This avoids confusion that would otherwise be caused by having a small ‘tail’ at the end if the starting and ending sides were not parallel.

Because of the grid nature of the screen, this option is slower because it is necessary to correct for ‘aliasing’ caused by pixels being at unpredictable distances from the starting point. To compensate for the slower speed, the pixels are no longer inverted as they were in earlier versions of tnimage.

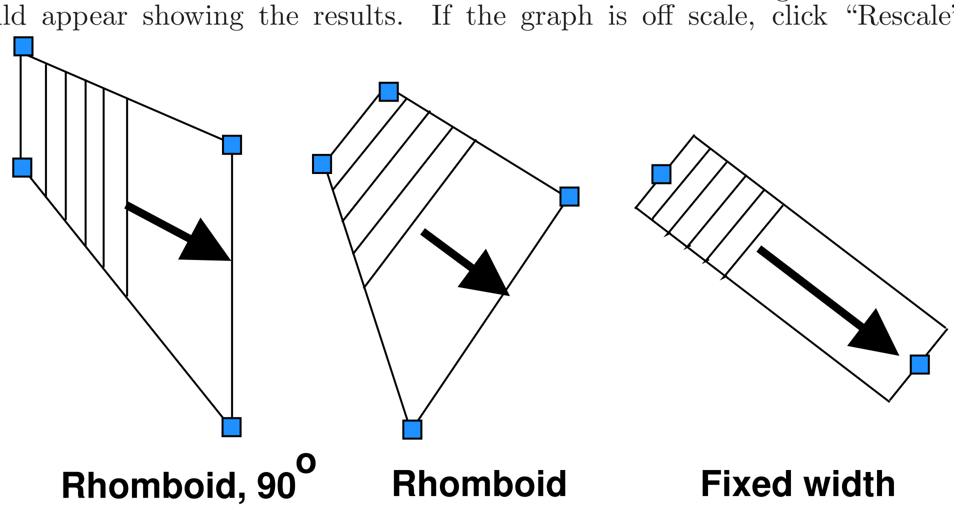
Fixed width only requires two end points instead of 4. Densitometry is performed in a rectangular region between the two points using the specified width in pixels (which can be 1 to 200). If the region is not perfectly vertical or horizontal, anti-aliasing is performed.

This method is the most useful for obtaining results which need to be compared to each other, or when the density of a very narrow region (such as a line) is desired. Of course, a narrower region will tend to be noisier than a wide one.

Fixed width densitometry is slower than 90° rhomboid scanning because more floating point calculations and memory accesses are needed. However, it is more accurate. Using fixed-width densitometry with a width of 1 gives a “transept line”.

7.10.1 Strip densitometry procedure

1. Make sure the image is grayscale before starting. Pixel values on color images may or may not have any correlation with brightness. Select "Color...Color→gray scale" to convert an image to grayscale.
2. Click "Process..Strip densitometry". The Strip densitometry dialog should appear.
3. Change the settings in the dialog box as desired. The most important is to specify whether the largest signal in your image is "Black" or "White". Click on "Maximum signal=black" or "Maximum signal=white" depending on whether the features of interest are darker or lighter than the background.
4. Note that there is no box for background subtraction in the dialog box. Background can be subtracted interactively or automatically in the graph. The graph is created automatically if "Plot result" is checked, and is automatically updated when you measure a new area.
5. Set the Width as desired if you are using "Fixed width" scanning. This value must be in pixels, because the algorithm is concentrating only on raw pixels at this point.
6. Click 'Accept' to begin densitometry.
7. Use the mouse to select points on the image that define the boundaries of the region to measure. For Fixed with scanning, select one point at the beginning and one at the end of the region so that 2 boxes appear on the screen.. This region need not be vertical or horizontal, but can be at any angle.
8. For Rhomboid-type scanning, select two points at the start and two points at the end of the strip to scan. This will define two lines. Click once on each corner of the rectangular area to be scanned, in a clockwise direction, so that 4 boxes appear on the screen.
9. Drag the 2 or 4 control points using the mouse to the desired positions. If any of the boxes are in the wrong position, or the lines connecting them are crossed, click and drag the boxes to the correct position. (NOTE: if the lines are crossed, the algorithm automatically swaps the coordinates to un-cross them).
10. Press the space bar to begin densitometric measurement.
11. The image will be scanned between the two lines as shown in the figure below. A graph should appear showing the results. If the graph is off scale, click "Rescale".



Direction of pixel scanning in various types of strip densitometry.

12. In the graph, you can subtract background (baseline) values manually or let the computer select a best fit line automatically. Once the feature of interest has been isolated as a single peak, drag on the graph with the mouse to highlight the peak. The portion of the graph that is selected will be calculated and the measured area will be shown in a small list box. This "area" represents the total signal minus background of the feature in the image, multiplied by the Calibration factor.

13. In the graph window, click on "Save to disk" to save the densitometry tracing in an ASCII file.
14. Select additional control points and press the space bar again to make additional measurements. The graph will automatically be updated and resized if necessary to accommodate the new region. It is not necessary to click 'Accept' before taking a new measurement. You can continue selecting additional areas indefinitely.
15. When you are finished performing densitometric measurements, click the main Cancel button or press Esc. This will unmap the densitometry dialog box. You will be prompted to save the graph and the list box containing the results.

Options:

Pixel density calibration

OD table If checked, this will cause `tnimage` to use the value in the optical density table in its calculations instead of the raw pixel value. This information is sometimes provided by digital scanners in the form of a 'gray response curve' or 'gamma curve' that is embedded in the TIFF file. For most images, this option will have no effect since no gray response curve is present. However, you can easily create a curve by clicking "Config..Show OD table" and modifying the graph to create a curve of any desired shape. Since this curve has only 256 elements, it is only applicable for 8-bit grayscale images. If you create a curve manually, it must be monotonically increasing or decreasing (i.e., no peaks).

z units If checked, this causes `tnimage` to use the calibrated pixel value in its calculations instead of the raw pixel value. Pixel value calibration is performed before starting densitometry, by clicking "Process..Calibration", checking the desired equation to which the 'z value' is to be fitted, and then clicking on each calibration standard in the image (See Sec. 7.7) and entering the known value in the spreadsheet. `Tnimage` performs a linear regression on the data and uses these coefficients to calibrate pixels in the image.

NOTE: For this option, the "Maximum Signal Black/White" setting is ignored, because the user calibration automatically defines what pixel value is the maximum signal.

None In this option, the signal and density are given as raw densities (expressed as a number between 0 and 1). (signal = density $\times$ number of pixels.)

Automatically save scan

If this option is checked, each scan will automatically be saved in an ASCII file.

Filename for scan

A default filename of "1.scn" is provided. If the 1st 8 characters of the filename consist entirely of digits, the program will automatically increment the filename after each scan (For example, if the starting filename was "1000.dat", subsequent scans would be saved under 1001.dat, 1002.dat, etc.). If the filename contains letters, the filename must be typed in each time.

Plot results

If checked, the scan results will automatically be plotted on the screen after each scan. Clicking the "OK" button on the graph makes the graph disappear upon the return to scanning mode. The graph is always automatically scaled in the Y direction to fill the entire box. When the graph is visible, the data can be saved to disk, a background curve can be subtracted from the data, or the graph can be captured into a new image (See "Plotting densitometry results and other data").

If the image has been calibrated, moving the cursor over the plot area of the graph causes two different x-values are printed. The upper x value is the data point number, and the lower x value is the calibrated value for the center pixel of the line along which the strip densitometry was performed. If the image has not been calibrated, the two x values will be identical. Clicking-and- dragging within the plot selects a portion of the graph for area calculation. The selected area is highlighted in reverse color. If the image has been calibrated, this area (printed at the lower right of the graph) is also calculated from the calibrated values.

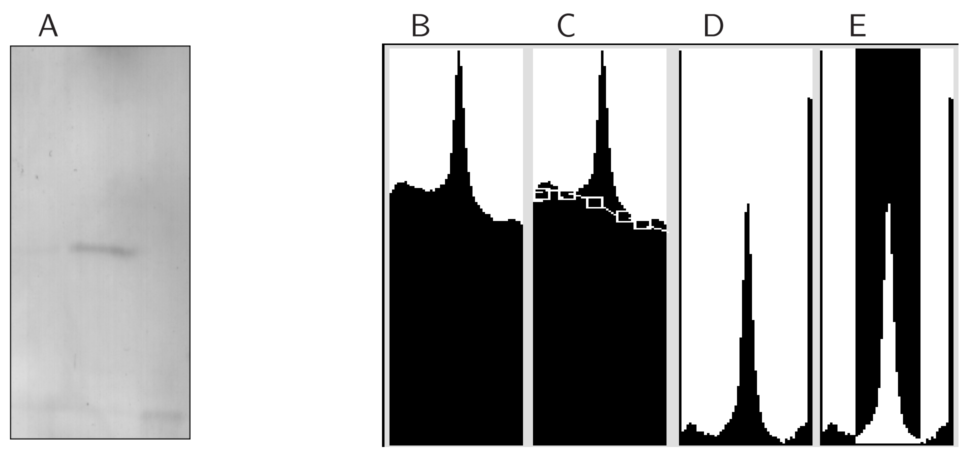
Pause to show region

If checked, the program pauses after drawing the 4 boxes, allowing you to verify that you selected the correct region. Press a key to continue.

Fixed width Width to be used when "Scan Type" is set to "Fixed width".

Panel **A** in the figure below is a typical 8-bit image showing part of a nitrocellulose blot stained for proteins with colloidal gold. The dark area is the protein calexcitin. It is necessary to know the total amount of the protein in this band.

- The band was first measured using "fixed-width scan" in "Strip densitometry". The rectangle was oriented so that the band was perpendicular to the start and end points of the rectangle. This produced the trace shown in **B**.
- A manual baseline was subtracted, using the points shown in **C** (control points are shown by small squares). The program generated a Bezier curve and subtracted it from the data, producing **D**.
- The peak area can now be easily determined, by dragging the mouse to highlight the peak (**E**).



Densitometric analysis of a protein band using strip densitometry.

Despite the complicated statistical analysis available in spot densitometry (above), this method of visually identifying the band is still the most accurate and reproducible densitometric technique.

NOTE: Densitometry may not work correctly on zoomed images. Use "Change size" first.

7.10.2 Plotting densitometry results and other data

There are several useful options available in the plot mode:

Save to Disk– Saves the currently-displayed graph into an ASCII file.

Smooth – Performs Gaussian smoothing on the data being displayed.

Auto Baseline – Automatically calculates an optimal curved baseline and subtracts it from the data. The smoothness of the calculated baseline can be changed by clicking on "BL smoothness". The calculated baseline is displayed for 1 second before subtracting it.

Manual Baseline– Allows you to select a baseline manually. Click at several locations on the graph where you want the baseline to go. Small squares will appear to indicate where you clicked. You can click-and-drag these squares to move them. When finished, press a key or click "Finished" to subtract the baseline. The baseline curve is constructed using a B-spline (see 'B-spline curve'). The squares can be anywhere on the screen, and do not have to be inside the plot. Also, it is possible to subtract a baseline from only a portion of the data by placing boxes only under the portion you wish to change. However, any negative numbers created by baseline subtraction are truncated to 0's. Up to 200 boxes ('control points') can be used.

Capture Image– Clicking 'capture' allows you to capture the graph into a new image. After clicking on 'capture', a new image which includes the graph is automatically created. The new image is put into the background, behind the image you are currently scanning, to continue with the next scan.

Change– Toggles between "change" and "measure" modes.

Bezier change– Changes data in current graph using a Bezier curve. If graph has multiple panels, click on the desired panel before clicking "Bezier change" to select which graph to change. After clicking "Bezier change", click on the graph to set the control points. Each point is shown by a small square and may be re-positioned by clicking within the square. Clicking "Finished" will cause the curve to be substituted for the data in the graph.

Help– Calls a help screen.

OK – Ends plotting, returns to densitometry mode.

7.10.3 Curve Densitometry

Performs densitometry on a Bezier curve one pixel thick. Click on the desired control points. These are indicated by small boxes which can be repositioned by dragging to different locations. Pressing 'Space' will create a graph of pixel intensities along the Bezier curve. Click on Cancel when finished.

Because the line is one pixel thick, the graph will be noisier than other types of densitometry. Future versions of `tnimage` will be able to perform curve densitometry on arbitrarily thick lines. Evaluating thick curve densities is not trivial however, because pixels will be oversampled in concave parts of the curve and undersampled in convex parts.

7.10.4 Peak Areas

Finds the x and y coordinates of each peak and calculates the peak area. The integration limits are shown as short black lines on either side of each peak. No baseline subtraction is carried out before calculating peak areas. Thus, best results are usually obtained if a baseline was first subtracted by clicking on 'Auto.baseline' or 'Manual baseline'.

When the list of peaks is displayed, most of the click buttons on the left side of the plot are still active, but now apply to the peak list. This includes:

Save to disk

Capture image

Help

OK

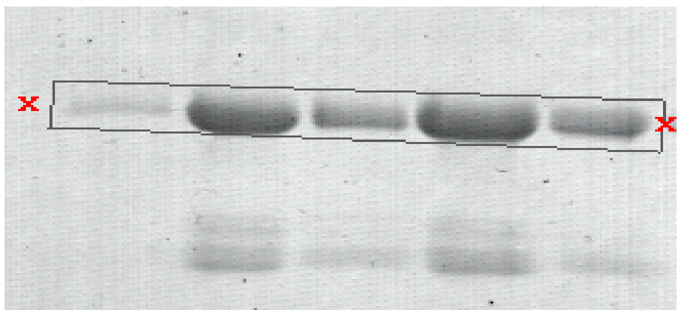
The other buttons ('peak areas','smooth', 'manual baseline', and 'auto baseline') are inactivated while displaying the list.

In the DOS version, the peak peak area list can also be closed by clicking on the '-' symbol in the upper left corner.

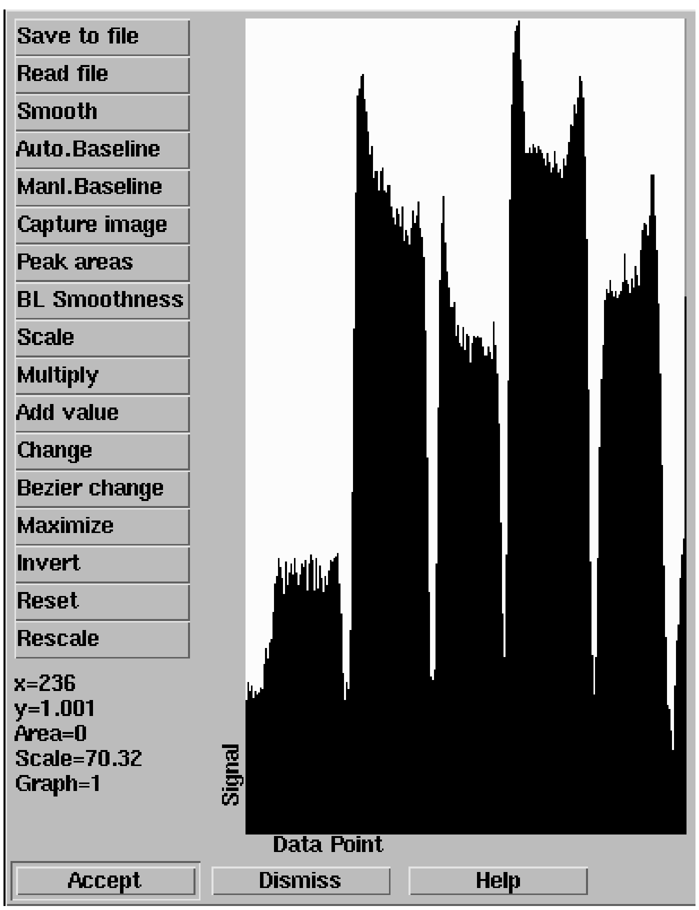
Area calculations: While the graph is displayed, it is also possible to manually measure peak areas by selecting the desired region of the graph. To select a region, click with the left mouse button and drag horizontally within the graph area. When the mouse button is released, the total area of the selected region is automatically displayed.

7.11 Western Blots and Stained Gels

Another application of strip densitometry is measurement of band densities on Western blots or protein/DNA gels. The image below is a polyacrylamide gel stained with Coomassie blue. Instead of measuring each lane as a separate strip, it is often faster to make the strip horizontal, so that the protein band of interest shows up as a series of rectangular peaks. This method is also very accurate.



In order to measure the amount of protein in this gel, click on "Strip Densitometry" and select two endpoints (shown with 'x') on either side of the gel, perpendicular to the migration direction as shown. The result is a simultaneous measurement of multiple bands.



In this graph, each of the five peaks represents the cross-sectional area of the corresponding band in the gel. After subtracting the background (manually using "Manl.Baseline" or automatically using “Auto Baseline”), the area of each spot can be measured by dragging on the peak using the mouse. The measured band intensities are automatically placed in a list box which can be saved into a text file.

7.12 Grain counting and pattern recognition

`Tnimage` can use the quick segmentation algorithm, a neural network algorithm, or a spatial differencing algorithm to identify and count objects in an image. The choice of which algorithm to use will depend on the pattern to be counted. Neural networks are preferable for complex patterns, since they take the internal structure of the pattern into account. However, neural networks require the adjustment of 3 parameters (threshold, match weight, and mismatch weight) instead of one, so it can be tricky to find the best combination of parameters to optimally discriminate between similar patterns.

Quick segmentation is extremely efficient and fast, but is only suitable for counting black grains on a light background. The grains need to be already separated from each other.

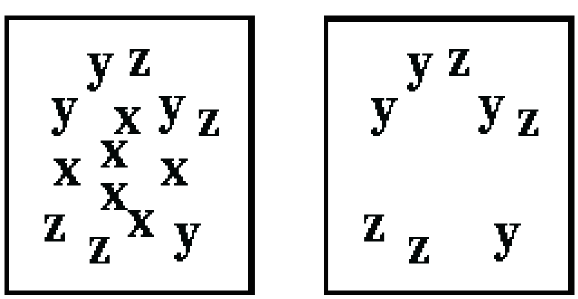
The differencing algorithm is also suitable mainly for small dark objects like grains, but is better at separating objects that touch each other.

7.12.1 Comparison of Grain Counting Methods

Feature	Neural Network	Differencing	Quick Segmentation
Speed	slow	medium	fast
No. of params	3	2	1
Works with color images	yes	no	no
Ability to separate closely-spaced patterns	good	medium	poor
Requires selecting representative pattern	yes	no	no
Provides graphs of signal distribution and size distribution	yes / no	yes / yes	yes /yes
Type of pattern	Any	Grains/spheres	Grains

The neural network algorithm is a modified and greatly simplified version of a hierarchical neural network [6]. With appropriate threshold values, it is possible to count grains or any other pattern such as cells, faces, etc., in the presence of other potentially-interfering objects. To illustrate, we will count occurrences of the letter "x" in a simplified image.

The left panel below shows the original image, a combination of x's, y's and z's. The pattern recognition process subtracts pixels that are part of the pattern (in this case the 'x's), while tye 'y' and 'z' are unaffected. This subtraction also allows any patterns obscured by the pattern to be identified on a subsequent pass.



Pattern recognition in tnimage.

7.12.2 Counting patterns with neural network method

1. Select "Process..Grain/pattern counting".
2. Click on "Select pattern"
3. With the mouse, click and drag on the image such that the crawling box circumscribes an "x".
4. Set Threshold to 0.5.
5. Set Match weight to 1 and Mismatch weight to -0.1.
6. Use the mouse to select a region to count, or do nothing to count the entire image.
7. Click on "Count patterns".
8. The x's will be indicated by a red cross as they are counted, leaving only the y's and z's (right panel). If the threshold is too low, the y's and z's will also be counted.
9. The count is displayed in a message box.

Options:

- Select pattern: defines the pattern to search for.
- Read pattern from disk: Define a pattern based on a disk file. (not currently implemented)
- Save pattern to disk: Save current pattern to a file. (not currently implemented)
- Count patterns: Count the occurrences of current pattern in currently-selected region or image, making no assumptions about the prototype pattern. It is necessary to select a pattern before using this option.
- Count grains: Counts occurrences of current pattern, making the assumption that the pattern is supposed to be a dark grain on a light background. It is necessary to select a pattern before using this option.
- Count std. size grains: Uses a pre-defined "grain" pattern to count grains. It is not necessary to select a pattern before using this option.
- Enhance grains: Performs a Sobel filtering to improve contrast for grain counting.
- Filename: file name for pattern if saving the pattern to disk file.
- Threshold: Determines the sensitivity to variations in the candidate patterns. A higher threshold means greater selectivity. Too low a threshold will result in extraneous items being counted as patterns, while too high a threshold will cause some patterns to be missed.
- Match Weight: Adjusts the relative weight given to a pixel which is part of a pattern.
- Mismatch Weight: Adjusts the relative weight given to a pixel which is not part of a pattern. This parameter is the penalty for an mismatched pixel, thus it is usually between 0 and -1.

7.12.3 Grain counting using neural network method

- After you click "Count patterns", the program will pause for several seconds while it searches for candidate patterns in the image or selected region. This may be a long time if the template pattern or the image is large. Color images (including indexed-color images) take considerably longer than grayscale images. It is recommended to test the threshold values on a small part of the image before counting a large image.
- After counting, each occurrence of the pattern that is found is indicated by a red crosshair. If desired, the original image can be restored by clicking "Restore original". The program will repeatedly make passes through the image, until no additional matching patterns are found.
- In most cases, the Match Weight should be kept at 1.0. Setting the value lower would cause the area around the pattern (i.e., the mismatched pixels) to have proportionately more influence on whether the pattern is identified.

- Setting the Mismatch Weight to a more negative value may improve the selectivity in some cases, by increasing the penalty for a mismatched pixel. More negative values are needed for complex patterns such as faces.
- When counting grains, the grains must be black on a light background. Use "Color..Invert colors" if necessary to achieve this.
- When counting grains, especially on color images, it is recommended to click "Enhance grains" before starting. Otherwise, extraneous dark areas (such as cells) may be misidentified as large clumps of grains.
- During the counting, the patterns the most closely match the prototype pattern are recognized first. The counting can be interrupted at any time by clicking on "Cancel", pressing Esc, or clicking the main Cancel button.
- When selecting the prototype pattern, select an area of average size and intensity, surrounded by a small region of background. Omitting the background region, or selecting a pattern that is too small, will increase the likelihood that a large dark clump will be identified as several grains that are overlapping.

7.12.4 Grain counting using segmentation and differencing methods

- When counting grains, the grains must be black on a light background. Use "Color..Invert colors" if necessary to achieve this.
- Maximize contrast in image before starting. Ideally, all the grain pixels should be below 127 and all background pixels should be above 127 (assuming an 8-bit image and a threshold of 0.5).
- Any large dark objects in the image, such as labels or borders, must be removed before starting. Large areas of uneven background can be removed using "Filter..Remove Low Frequencies" and setting kernel to 3×3 or 15×15 and 'Amount of filtering' to 100.
- If extraneous items are misidentified as occurrences of the pattern, the threshold should be increased to improve specificity.
- If noise pixels are counted as grains, filter the image with a low-pass filter or noise removal filter before starting.
- "Min size" discards any spots containing fewer than the specified total number of pixels. "Min size" and "threshold" can be adjusted to give a more accurate grain count without counting stray dots.
- If the 'labels' box is set, the following marks will be added to the image:
  - A number indicating the measured size of the spot. Clicking on "Show size" opens a menu allowing you to specify what is printed on the labels. This can be size, spot number, signal, or no label.
  - An arrow pointing to the center of the spot
  - A rectangle indicating the bounding box of the spot. **Note:** the box only indicates the maximum x and y range of the spot. Calculations are done on the subset of pixels within this range that exceed the threshold value.

  Be sure to back up the original image (Ctrl-B) before using this feature. The marks can be removed by clicking "Undo" (or Ctrl-U). If 'labels' is not checked, the spots will be marked with a small cross which will disappear when the mouse is clicked on the image.
- 'Label color' sets the color for the labels.
- If any single grain contains more than 10,000 pixels, the counting is automatically aborted.
- If the image contains more than 10,000 grains, `tnimage` will need to be recompiled with a larger value for MAXGRAINS.
- The size distribution data can be saved to disk by clicking on the "Save" button in the Size Distribution graph.
- The differencing algorithm uses the "Size" parameter which determines the area over which differencing is to be performed. A larger value will difference over a larger area, which would be suitable for larger grains, but will take longer. Using too small a value will increase the chance that noise pixels will be counted as grains. The differencing method is also more sensitive than quick segmentation, so the threshold should be increased to compensate.

- If “Sz Graph” is checked, a graph of size distributions will be created.
- Checking “Dens Graph” creates a graph of signal distributions, *i.e.,* the number of spots *vs.* densitometrically measured signal per spot. This is useful, for example, in analyzing spot sizes on 2D gels.
- If “Save Data” is checked, a list of all spots, together with their x and y coordinates, sizes, and signal per spot, will be saved into a disk file. This will be followed by the size distribution and the signal distribution data.
- On large images, if counting grains using the differencing method is too slow, the same effect can be achieved by filtering the image, setting “Difference filter size” to 10 in the Filter dialog, then counting grains using the Quick Segmentation method. This allows different thresholds and filter sizes to be more rapidly tested.

7.12.5 Notes on Pattern counting

- It is usually necessary to clean up the image before counting grains, for example to remove dark areas around the edge and other artifacts in the image, and to maximize the contrast before starting. You should also back up the image (Ctrl-B) before starting.
- Both the threshold and mismatch weight should be increased when counting patterns, compared to counting grains. For example, if the threshold is 0.9, a pixel will match if the average of the red, green, and blue values in a pixel deviate from the pattern by less than (1-0.9)$\times$ 255, or 25.5. If the sum of the signals from each pixel is higher than a minimum signal (defined as pattern area $\times$ threshold), the neuron will register a match.
- Increased speed can be obtained by converting color images to grayscale before starting. Note that just because an image appears to contain only shades of gray, this does not necessarily mean it is a grayscale image.
- Image contrast should be maximized before starting (“Color..Contrast.. Maximize Value”). Increasing the image contrast has an effect similar to increasing the threshold.
- Clicking on “Enhance grains” often improves the accuracy of pattern counting as well as the accuracy of grain counting.
- The pattern detection algorithm is currently unable to recognize patterns that are rotated or of different sizes. Often, patterns of different sizes can be recognized by selecting only one edge as the model pattern, e.g. if the desired pattern is a symmetrical round object, drawing the box around the left half of the object instead of the entire object will cause the algorithm to recognize objects of a broad range of sizes.
- The Size Graph is a histogram of the number of objects found that were counted because they were larger than the minimum size, plotted for each possible size (i.e., area of each object in pixels). The Density Graph (or “Signal Distribution Graph’) is a histogram of the number of objects for each possible total signal. The density for each pixel is a number between 0 and 1.0. The total signal is the sum of the densities for all the pixels in a given object. For example, if all the pixels in a grain had a density of 0.5, and the grain was a 10$\times$10 pixel square, the total signal would be 50. This could be different if you calibrated the pixel values (see “Image Calibration”). The data shown in both of these graphs is compressed to fit onto the screen. To see the complete data set, click on “Save grain results” to save the data into a file.

7.12.6 Possible problems with Grain and Pattern counting

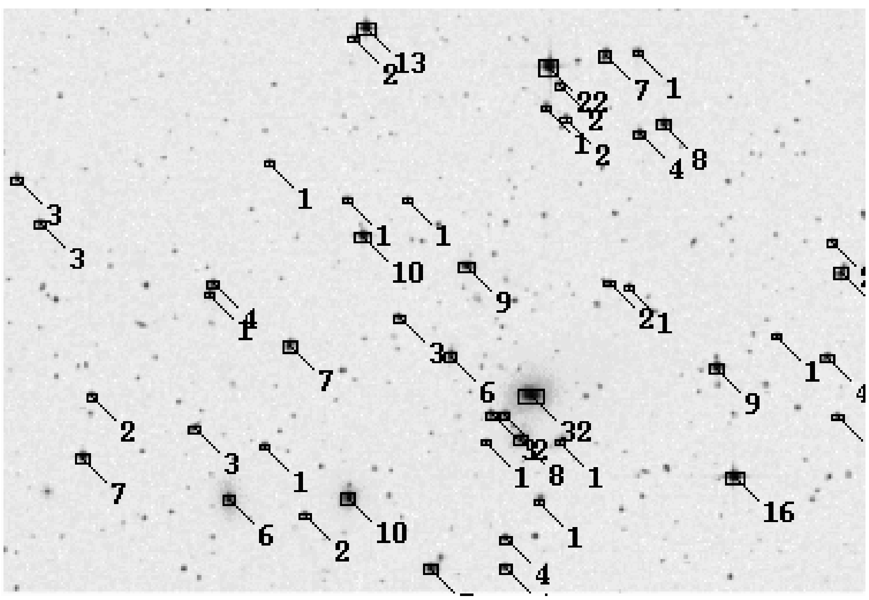
- **No patterns detected** Threshold is too high, mismatch weight is too negative, or match weight is too low; or, contrast is too low in image.
- **Too many patterns detected** Threshold is too low, mismatch weight is not negative enough, or match weight is too high. The optimum threshold is different for each type of image. The ideal threshold can be estimated visually by watching the numbers in the information box, and determining the signal at the point at which non-patterns begin to be detected. Interrupt the counting at this point and set the threshold to the value of the signal shown in the information box.

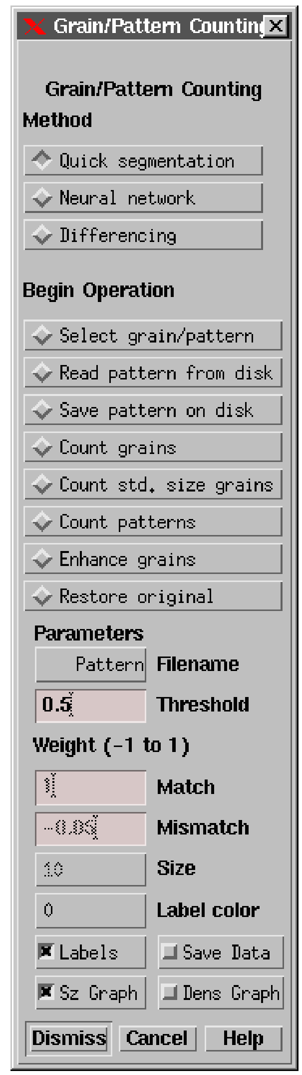
- **Same pattern counted more than once; or, algorithm gets stuck on one pattern (coordinates don't change).** This is a bug; please report it.
- **Processing is too slow.** Convert image to grayscale. This will increase speed at least 2.8 fold. Converting image to 8 bits will also help.
- **Message box says "0 Patterns found", even though it found patterns on the same image earlier.** You may have accidentally clicked on the background or menu bar, deselecting the image. Click on image and repeat.
- **Message box says "Neuron count mismatch".** This message occurs when you click on a different image while the algorithm is counting. The buffers are incorrect for the new image, so the algorithm aborts.
- **Small red box indicating location of pattern is off-center.** This occurs when defining the pattern, if the region selected for the pattern is not centered on the pattern (i.e., too much background is included in the pattern on one side).
- **Large areas with shading similar to the pattern are identified as numerous instances of the pattern.** This is caused by selecting an insufficient amount of background around the prototype pattern; or the mismatch weight is too low.

Examples

Size distribution using quick segmentation

Here is an example of an astronomical image after analysis by the quick segmentation algorithm. The threshold was set to 0.8 so that only the larger stars are counted. Each star that was counted is automatically labeled with its relative size. The size of a given object depends on the threshold and will be larger if the threshold is smaller, because more pixels are present in the distance matrix which is used to find grains. The grain counting dialog box is shown at right.





Left: Image with objects labeled with relative size using quick segmentation method
Right: Grain counting dialog

The size distribution graph (not shown) usually follows a Poisson distribution.
NOTE: Measured grain sizes will be different depending on the method and parameters used. Users should verify that the results obtained are consistent with their expectations.

7.12.7 Data file format

When you click on “Save results”, tnimage creates a file containing information about the grains.

The first part of the file is a list of data about each grain. The columns are: (1) the spot number; (2-3) the x and y coordinates of the center of each grain, (4) the size in pixels, (5) the total signal in the spot, and (6-9) the coordinates of the corners of the spot. If the image has been calibrated, the corners are also shown as their calibrated values (columns 10-13).

The second part of the file is a histogram of spot sizes. Each spot size, from 1 to whatever is the largest spot, is listed in the first column, followed by the number of spots of that particular size in column 2.

The third part of the file is a histogram of spot signals, combined into 256 bins. This would be the same data as in the Densitometry graph.

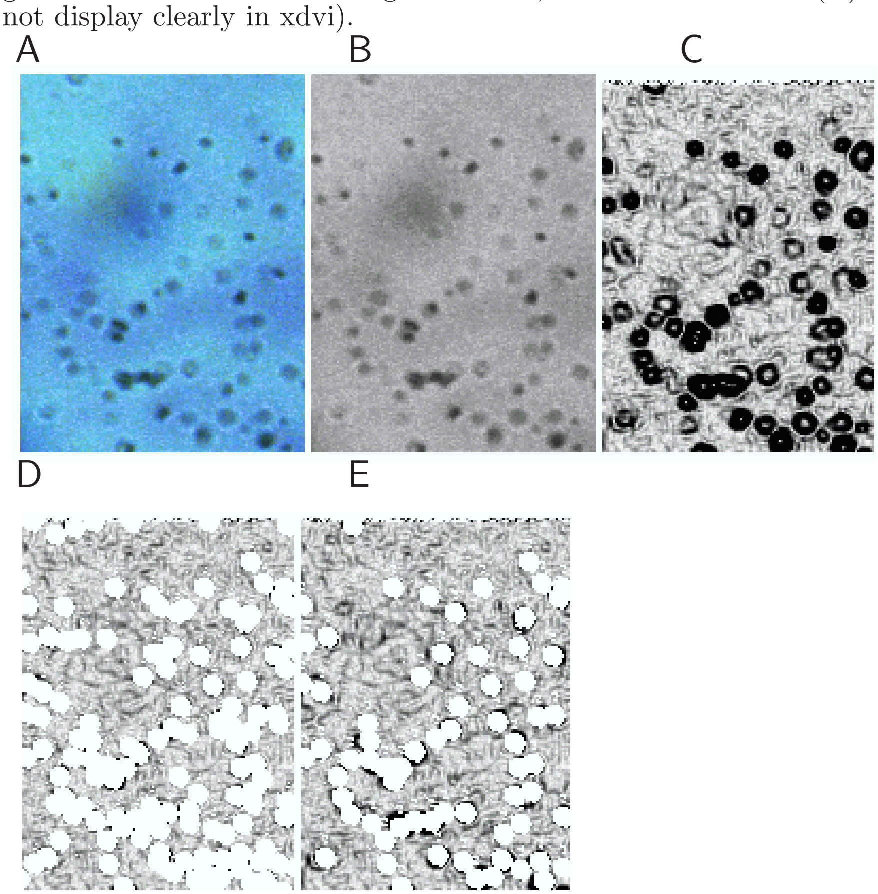
Recommended Threshold and Weight values for neural network method

Type of pattern	Threshold	Match Wt.	Mismatch Wt.
Grains (monochrome)	0.50	1.0	-0.05
Grains (color)	0.45	1.0	-0.10
Grains (enhanced)	0.55	1.0	-0.15
Faces	0.60	1.0	-0.15
Faces (color)	0.95	1.0	-1.00
Patterns	0.90	1.0	-0.50

Note: These values should be considered as starting points only. Different images will need different thresholds and mismatch weights.

Example of neural network grain counting:

Below is an example of neural network grain-counting of a moderately-difficult image illustrating some of the steps that may be necessary to obtain the most accurate counts. Panel **A** shows the original image, which contains several grains that are out of focus or clumped together, as well as some grains that are sharply focused, and a darker, out-of-focus blue cell in the center, which could be difficult to distinguish from the two faint grains on top of it. Under some conditions, the cell could even be misidentified as a large clump of grains. The image was converted to grayscale (**B**) and the grains were enhanced (by clicking on "Enhance grains"), producing **C**, which is more easily analyzed. Using the default threshold value of 0.5 and Match and Mismatch weights of 1 and -0.05, however, caused a number of extraneous points to be counted as grains (**D**). This occurred because the filtering process also increased the noise slightly. Changing the mismatch weight to -0.15 solved the problem, and the program counted the number of grains as 64, the correct number (**E**). (**Note:** This figure may not display clearly in xdvi).



7.12.8 Fuzzy Partitioning

This method uses the fuzzy k-means method to classify each pixel in the image or selected region into clusters. The pixel is then set to black or white depending on which category it falls into.

This feature is still under development. Currently only the two cluster case is considered, and classification is only performed on the basis of pixel intensity. This is primarily useful in spot densitometry. Future versions of `tnimage` will use fuzzy cluster analysis to identify image features.

7.13 Morphological analysis

Morphological filtering simplifies the appearance of an image to facilitate measurement of features. Combinations of these basic operations can also produce desired effects. These operations are non-linear and irreversible.

7.14 Thresholding, Dilation, Erosion, Opening, and Closing

Thresholding converts a grayscale image to a binary black and white image.

Erosion removes pixels around the periphery of features in the image, making lines thinner. Because small features disappear during erosion, counting the features that have disappeared can give an approximate estimate of the size distribution.

Dilation is the converse of erosion, and adds pixels around edges of features. This is commonly performed after an erosion or skeletonization to prevent pixels from disappearing.

Skeletonization locates the center of features, converting a figure into something resembling a line drawing, where the lines depict centers or 'skeletons' of the figures. This is useful, for instance, in metallurgic analysis. Usually must be followed by dilations to obtain solid lines.

Opening is an erosion followed by a dilation, and opens up spaces between just-touching figures.

Closing is an dilation followed by an erosion, and closes up breaks in features. Frequently, several repetitions of openings and closings are performed to achieve a desired effect.

7.15 Quick Segmentation

Quick segmentation Uses a highly efficient segmentation algorithm to separate features, converting an image into a line drawing in which the lines depict outlines of objects in the image. Its purpose is to reduce the complexity of the image for pattern counting or further analysis. The algorithm is rapid but does not always give perfect segmentation.

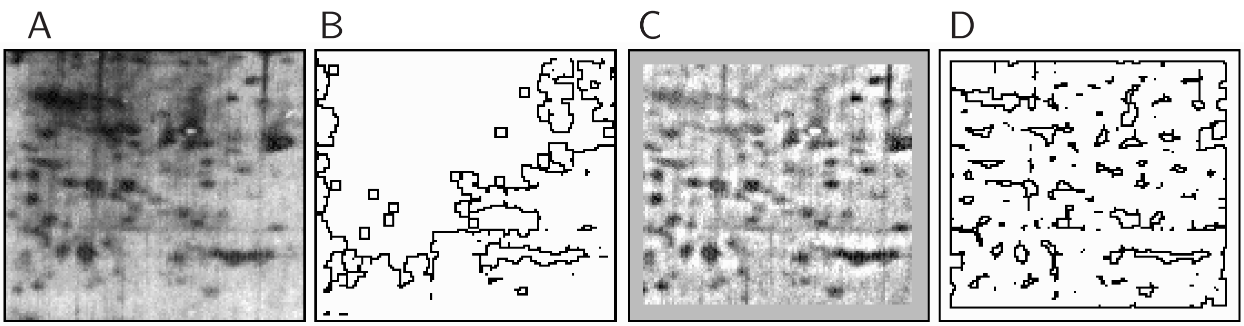
The image to be segmented should be converted to grayscale and 8 bits/pixel for best results. If the segmentation appears incorrect, try one or more of the following:

- Select "Color..Color→grayscale" to make sure `tnimage` is handling the image as grayscale. (Image may appear to be shades of gray and still be some other colortype).
- Invert the colors and segment the negative of the image. This often works if areas in the background were outlined.
- Try changing Maximum Signal from Black to White.
- Try changing Structuring element from N8 to N4.
- Change the brightness or contrast of the image.
- Filter the image to flatten the background using the following parameters:
  - Filter type: Remove low frequencies
  - Kernel: 15×15
  - Kernel multiplier: 2
  - Amount of filtering: 100

The **structuring element** is the morphological kernel that determines how pixels are added or removed. Two structuring elements can be selected in `tnimage`:

- N4 consists of the 4 closest pixels, directly above, below, left, and right of each pixel.
- N8 consists of all 8 of the nearest neighbors to the pixel. N8 generally produces a smoother result, but takes longer.

This parameter is not used by watershed segmentation, which uses the "Watershed kernel size" parameter instead.



Quick segmentation of an image of a 2D protein gel.
A Original image
B Quick segmentation classifies much of upper left of image as a big feature.
C Same image filtered to remove low frequencies (kernel = 15x15, multiplier = 1).
D Quick segmentation of filtered image (threshold = 180).

Binary vs Graylevel - Binary filtering produces a solid white or black result, while graylevel filtering retains the original shades of gray. This is also not used by segmentation.

Maximum signal - If the features of interest in an image are black, this should be set to black. For grayscale erosion, switching the maximum signal is the same as switching from erosion to dilation.

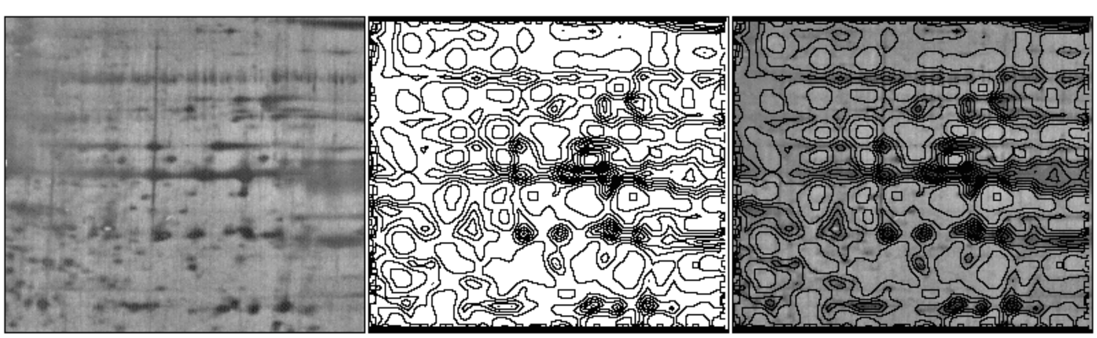
Threshold - Determines the cutoff value for deciding whether a pixel is a feature or the background. A higher threshold causes more pixels to be considered features and produces more lines in the segmentation.

Watershed kernel - Used for watershed segmentation instead of structuring element. Determines distance from pixel to be considered as part of a neighborhood group. Large values will take a long time.

Notes

1. For quick segmentation, a threshold that is too high or too low will produce an all-white image.
2. For watershed segmentation, excessive low-pass smoothing can result in no watersheds being found, while no smoothing can result in over-segmentation.

Contour Map Uses the quick segmentation algorithm to create a new image consisting of contours following areas of steepest gradient. The image should be low-pass filtered first to avoid drawing circles around small noise features. The number of contours is determined by the "Contour sep" setting. A larger value results in fewer lines but is faster. The actual number of contours also depends on the image. The product is still an image and does not contain vectorized data. Note that it is not a true iso-intensity contour map since the number of lines is determined by the gradient.



Contour map Image was low-pass filtered with a 9×9 kernel before creating the map using a spacing of 10. **Left:** Original, **Center:**Contours, **Right:** Contours superimposed on original image.

7.16 Watershed Segmentation

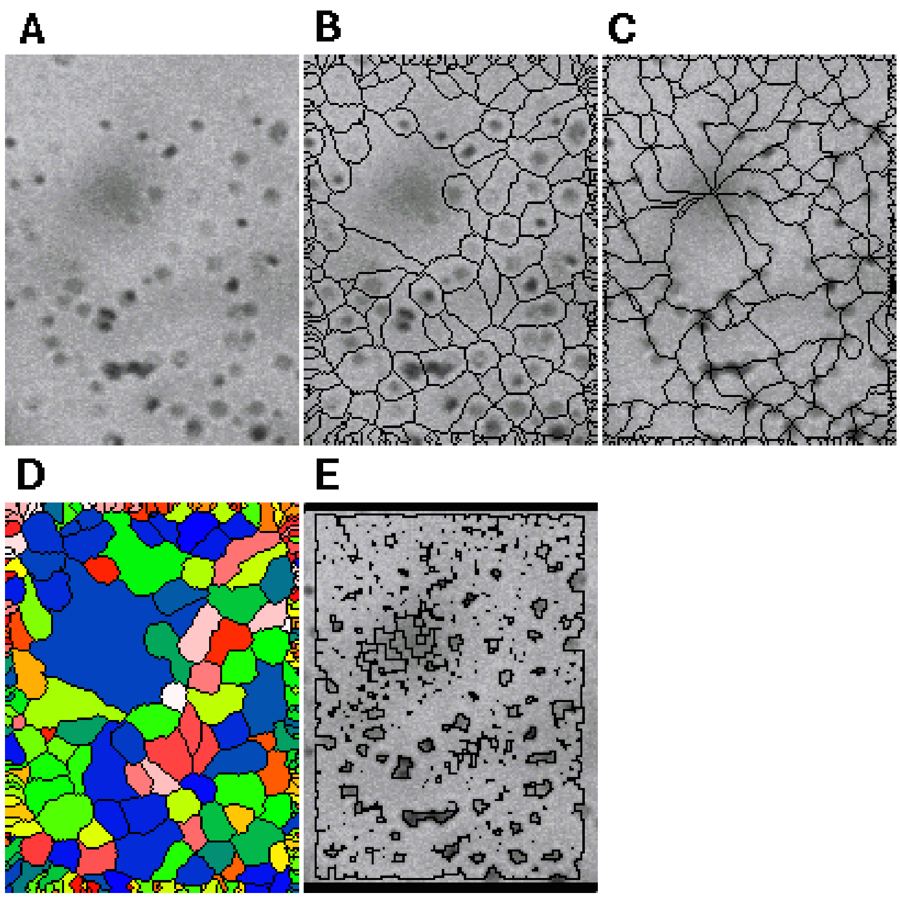
Watershed segmentation [7] finds pixels at the midpoints between features by treating the pixel values as topographic elevations. The areas between features become *catchment basins* which depict the influence zones in the image (see figure below). Images should be low-pass filtered first to avoid over-segmentation.

The output is a new image in which each catchment basin is given a different gray scale value. The actual watershed points are those points on the new image with a value of 0. These points can be selected by inverting and increasing the contrast, or by applying the formula

$$if(i > 0)i = 255;$$

to the image. These edges form a tessellation of the image while the constant color regions indicate the catchment basins.

If the gray scale of the image is inverted first, the result is that lines are drawn connecting the highest elevation routes through the features instead of around them (see figure).



Watershed segmentation of an emulsion autoradiogram.
A Original image

B Watershed segmentation superimposed on the image. The image was low-pass filtered twice using a 5×5 kernel before segmentation. The segmentation edges (i=0 points) were pasted over the original using the "Mask..Add" function.
C Watershed segmentation of grayscale-inverted image, created as in **A** except the image colors were inverted before segmentation. The watersheds now track the closest distance between the grains instead of the greatest distance between them.
D Actual watershed segmentation obtained in **B**, with a spectrum colormap added.
E Quick segmentation result superimposed on the same image in **A** for comparison. Image was filtered once with a 15×15 low frequency removal filter, using a kernel multiplier of 2 before quick segmentation. Line around outside is an edge artifact caused by the filtering.

7.17 Trace curve

`Tnimage` can convert an image of a graph, chromatogram, etc., into an ASCII file. Currently, tnimage only traces in 1 dimension, and from left to right. Contour tracings will be implemented later.

Procedure for tracing curves

1. Convert the image to grayscale. Pseudocolor images cannot be traced accurately.
2. Clean up the image to remove any stray specks, labels, or other points that could be confused with the graph. The entire image must be clean, including the edges. The trace will jump to the darkest y-value point for each x value, regardless of where it is located on the image, even if it is not visible on the screen. Use "Paint region" if necessary to eliminate nonwhite areas from the image.
3. Select "Image...Trace curve".
4. To cause the trace to be saved automatically, check "Save trace".
5. If "Plot trace" is checked, the data will be displayed. Clicking on the "Save" button will save it. While the graph is visible, it is possible to smooth it, subtract a background, measure areas, etc. (See "Plotting densitometry results").
6. Find the pixel value of the curve to trace by moving the mouse onto the desired area. The pixel value is displayed in the information box on the left after "i=". Set "Color to track" to this value.
7. Click on "OK", then move the mouse cursor to the left end of the curve and click the left mouse button.

A cross-hair cursor indicates what is being traced out. Click on "Cancel" when finished.

Sometimes, sharpening or thresholding the image can improve the tracing. If there are extremely sharp corners in the image, it may be necessary to manually create a channel between the ascending and descending parts of the peak to force the trace to follow to the top. This can be done by selecting "Draw" from the menu, or pressing *F2* (=manual draw), and carefully moving around with the cursor keys (Be sure to back up your image before starting).

Checking "Debug" waits for a keypress after each x increment (in the DOS version) and prints the x, y, and the pixel value being tracked. This is helpful if stray noise pixels are accidentally getting included in the scan. In Unix, this option prints the values to `stdout` without pausing. It is possible to get a file containing this debug information by starting `tnimage` with the command `tnimage > debug.file` .

Only pixels on the selected image are included in the trace. Thus, in order to trace something in the background, it must be converted to an image first (use “File...Create/resize image”). The tracing algorithm tracks whatever pixel is closest to the selected tracking color for each x value. Thus, if the image contains a pixel closer to the tracking color than any pixels on the curve, the cursor will jump to this area instead of tracing the curve. If the pixel is beyond the edge of the screen, the cursor will jump to the edge of the screen. If this occurs, it will be necessary to delete the offending area before tracing anything (The “backspace” key is the most convenient way to do this).

7.18 Fourier Transform

Performs a Fast Fourier Transform on the image or selected region. This can use a lot of memory because the frequencies are stored as double-precision complex numbers, which require 16 bytes for each pixel. Thus, an FFT is 4 times as expensive as a 32-bit image. Additionally, the mathematics of the FFT requires that the image be first enlarged so that each dimension is the next higher power of two. For example, if your image is 129 x 67 pixels, `tnimage` has to create a new buffer of 256 x 128 pixels (524,288 bytes) to carry out the calculations. Your image is automatically enlarged to that size. Moreover, the sharp discontinuity between the edge of the image and the new black pixels in the buffer can create undesirable artifacts in the Fourier transform.

The FFT result is a matrix which is displayed as a new 128 bit/pixel image. This new image can be saved, unloaded, annotated, filtered, and manipulated like any other image. Any changes made to this image (such as adding text) are also converted to the appropriate floating point numbers and inserted at the appropriate position into the displayed component (real or imaginary) of the FFT matrix. Therefore, changing pixels by typing on the displayed image could change the real component of the FFT, the imaginary component, or both. After editing the image to enhance or eliminate specific spatial frequencies, perform a reverse FFT to obtain the filtered result.

Forward/Reverse/Change display only

Selects whether to carry out a forward or reverse FFT, or to merely change which component (original image, imaginary, real, or power spectrum) is being displayed. No actual change is made to the FFT data. The display can also be changed by typing `fftdisplay` *mode* in the command editor, where mode is 0=original, 1=real, 2=imaginary, and 3=power spectrum.

Real/Imaginary/Power spectrum

Selects whether to display the real, imaginary, or power spectrum component of the transformed image, or only the original image.

NOTE: If you select “imaginary” or “power spectrum”, the displayed image will appear black if you perform a FFT followed by a reverse FFT. This is because the original data do not have an imaginary component. The original data are not lost!

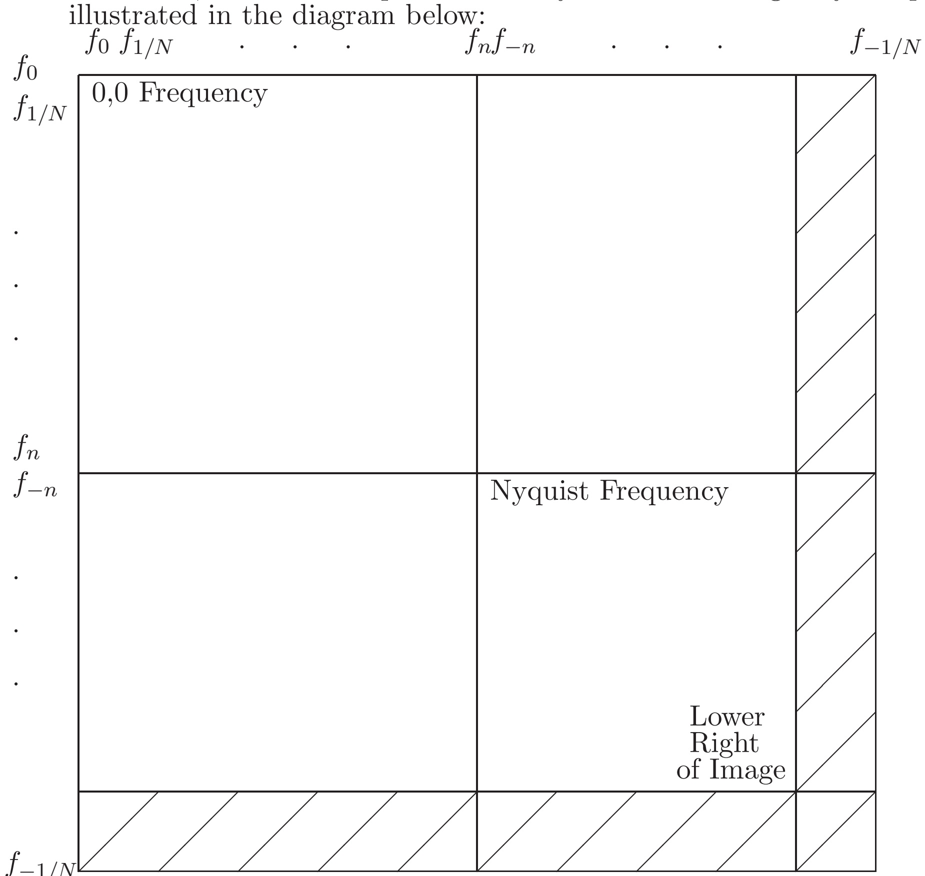
Although a complete description of the many applications of FFTs and deconvolution is beyond the scope of the manual, there are a few `tnimage`- specific points worth remembering:

1. Regardless of the screen mode or color depth of the image, the FFT and deconvolution algorithms treat all pixels as monochrome. Thus, a 24-bit/pixel image is treated as a 24 bit deep grayscale image. In other words, the colors are not deconvoluted separately in the current version of `tnimage`.
2. The grayscale mapping algorithm sets the most negative FFT result to black and the most positive FFT result to white. Thus, zero will be some shade of gray.
3. Editing power spectra will lead to unpredictable results in your image.
4. Select “About...About the image” to view some of the FFT parameters. These include:

- Minimum FFT value (which is mapped to black)
- Maximum FFT value (which is mapped to white)
- FFT=0 color value (the color to which FFT values of zero are mapped)
- FFT=0 RGB values (the corresponding red, green, and blue components of the FFT=0 color value).

These numbers can be used to calculate the color values needed to edit the image in the frequency domain to remove or accentuate specific frequencies. The filtered image can then be reverse- transformed to obtain the result.

5. When an image is set to display the real or imaginary frequency components of an image, operations that only affect the original image (such as "flip image", "change contrast", etc. will appear to have no effect. Before performing these operations, select "Process..FFT...Change display only" to switch the display to the original image.
6. FFTs are displayed in the conventional manner, i.e. with lowest positive frequencies at the top and left, lowest negative frequencies on the right and bottom, and Nyquist in the center, with the exception that only the real or imaginary components are shown, as illustrated in the diagram below:



In this diagram, the shaded area marks portions of the FFT extending beyond the lower right corner of the original image. The 0,0 frequency is in the upper left corner, and the f=0 frequencies run vertically along the left and top edge, from f=0, f=1/N, ... f=n, f=-n, ... f= -1/N.

If the x or y size of the image is not a power of 2, the image is enlarged to the next power of 2 in each dimension.

7. FFTs should be done with as few other images present as possible. Because an FFT is extremely memory-intensive, if `tnimage` is forced to use virtual memory, the FFT may take a very, very long time.
8. When deconvoluting or convoluting, the two images should be approximately the same size for optimal results.

9. Avoid image sizes that are just larger than a power of 2, such as 129 x 129 pixels. This will cause the complex array for FFT to be allocated as the next higher power of 2 (i.e., 256 x 256). The additional pixels will also be set to solid black, which will create artifacts in the FFT.
10. The screen mappings of FFT intensity values are automatically rescaled whenever the FFT image is redrawn. Thus, changing the contrast of an FFT'd image will have no effect on the appearance of the real or imaginary components. To change the brightness of a FFT'd image, use "Color...Grayscale map".

Example: Removing periodic noise from an image.

Periodic noise, such as the circular dots and moiré patterns that appear when a newspaper photo is scanned in a scanner, are ideal candidates for removal by Fourier transform.

1. Make sure the image is close to a power of two in both x and y dimensions.
2. Select "Process..FFT" and click "Operation = Forward" and "Display = Real FFT". (The "Data" and "Display scale factors" are used only for deconvolution and can be ignored).
3. Click OK. The FFT'd image should show white dots or lines near the middle of the image. These represent the spatial frequencies of the periodic noise.
4. Select the areas to be erased with the mouse and press Del, or erase them by setting the Fcolor to 0 and drawing over them. Do not erase along the edges or in the upper left corner.
5. Select "Process..FFT" and click "Display = Imaginary FFT", and repeat the erasing process on the imaginary components.
6. Select "Process..FFT" and click "Operation = Reverse". The new image should have most of the noise removed.
7. Repeat if necessary.

Convolution

View the image DECONVOL.TIF for a quick tutorial on convolution and deconvolution.

Convolution - Convolutes two images. Convolution is the same as multiplication in the frequency domain. Therefore, the resultant image will have the characteristics of both images. In the simplest possible example, a sharp image convolved with an image of a blurred point will become blurry. An image convolved with a single point is unchanged.

7.18.1 Deconvolution and image reconstruction

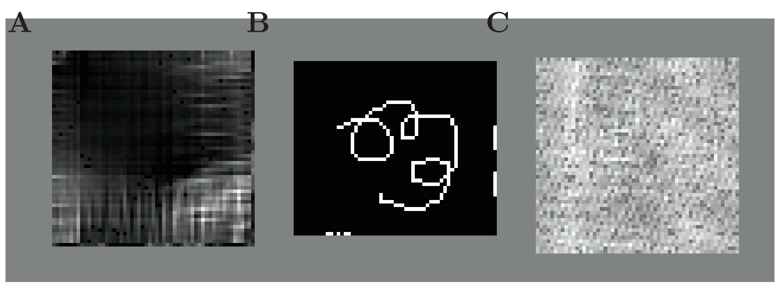
Deconvolution is a form of image reconstruction which was made famous by successfully being used on the distorted images from the Hubble space telescope. As with convolution, two images are required. Typically, one of the two images is a *point spread function*, or psf, which represents the effect of distortion on a single pixel, and the other is the image from which this distortion is to be removed. The two images must be approximately the same size. For example, if your image is blurred by spherical distortion from a bad lens, you would create an image the same size as your blurry test image, but consisting of a single point in the upper left of the image on a black background, subjected to the same blurring effect. If the point spread function is known accurately, after deconvolution the blur will be removed. In effect, *a priori* knowledge from outside the image is used to remove the distortion.

In principle, any degraded image can be reconstructed by deconvolution if the point spread function is known. If a photo is blurred due to movement in one direction, the point spread function would be a line in the direction of motion whose length depends on the amount of movement. In practice, deconvolution will not work for every image.

This method is far more powerful than using a “sharpening” filter, because any arbitrary type of degradation of the image can be eliminated. In practice, noise (especially noise in the psf) or low precision in the FFT will cause inaccuracies in the recovered image. For this reason, FFTs in `tnimage` are maintained as double-precision numbers. Also, if the spatial frequency of the psf at any point happens to be zero, it will also be impossible to reconstruct that point (since it would require dividing by zero). Adding a little noise to the psf will ensure that all frequencies are non-zero. Thus, sometimes it is necessary to tolerate some noise in the psf to get good image recovery.

Important: The point spread function must be centered at the upper left corner of the image to be used as a psf. Otherwise, a translation effect will also be convoluted into or out of the image.

Exercise: Try deconvoluting fft1.tif and fft2.tif. It should be possible to reconstruct the original image in fft1.tif, which was a small square, despite the complicated point spread function in fft2.tif which renders the distorted image unrecognizable. The more complicated the point spread function, the more difficult it is to recover a noise-free image.



Example of Fourier deconvolution using excessively complicated psf: Deconvolution of image A using psf B produces image C. Although noisy, the original small white square has been partially recovered.

Procedure:

1. Open the two files (fft1.tif and fft2.tif).
2. Select “Process..FFT”.
3. Set the following:

Operation =	“Deconvolute 2 images”
Display =	“Real FFT”
Image #1	0
Image #2	1

4. Click OK.

The first image will be replaced by the original, unconvoluted image, in this case, a small white square.

7.18.2 Digital Filtering of images using the FFT

Digital filtering can be easily done while the FFT spectrum is being displayed, by changing the pixels which correspond to the frequencies you wish to change. This is most conveniently done by setting the background color to “black” and using *delete* or *backspace* to erase the frequencies you wish to remove; or by selecting “sketch” (*F2*) and adding points at the desired frequencies by drawing with the mouse. It is necessary to change both the real and imaginary components as well as the positive and negative frequencies to achieve good filtering.

Example: High-pass filtering.

1. Make sure the original image size is a power of 2.
2. Set background color to black by right-clicking on “black” in the colormap palette.
3. Forward-FFT the image.
4. Delete the low-frequency regions (as shown with a ’L’) with the ’delete’ key.
5. Alternatively, use the mouse to select these regions and use “paint region”, “change contrast” or “color/intensity” to make them darker.
6. Optionally, the higher frequencies can be enhanced by using “enhance contrast” in the high-frequency region (shown by ’H’.
7. Drawing on a FFT’d image only affects the component being displayed (“real” by default). Apply the same filtering procedure to the imaginary component by selecting “FFT... change display...imaginary” and repeat the changes.

```
0LLLL---------------------LLLLL
LLLLL          |          LLLLL
|              |              |
|          HHHHHHHHH          |
|----------HHHHHHHHH----------|
|          HHHHHHHHH          |
|              |              |
LLLLL          |          LLLLL
LLLLL---------------------LLLLL
```

8. Reverse-FFT to obtain the filtered image. *Note:* Changing the zero-frequency pixel (at the upper left corner of the FFT) will have drastic effects on the image.

Open FFT

Reads a Fourier-transformed image stored in ASCII format. The file format is as follows:

- FFT of (image name)
- xsize (x size in pixels)
- ysize (y size in pixels)
- Real
- (Real frequencies from row 1)
- ...
- (Real frequencies from row n)
- Imaginary
- (Imag frequencies from row 1)
- ...
- (Imag frequencies from row n)

This format differs from the format used in `tnimage` prior to version 2.18. The header in these earlier files must be changed before they can be opened.

Save FFT

Saves the currently-selected FFT matrix to disk in ASCII format. This makes it possible to precisely change any desired frequencies with a text editor, then reload the FFT and reverse-transform. **Warning:** *These files can be quite large.*

Erase FFT

Erases the matrix used in storing the FFT, if a Fourier transform has been performed on the image, thus freeing up a considerable amount of memory. It will not erase the Fourier transformed image that appears on the screen. If no FFTs have been performed, this option has no effect.

Copying FFT into image

Occasionally, it is desired to copy the Fourier frequencies into an image (for example, to illustrate the appearance of a FFT). This can be done by clicking on “Copy FFT into image buff”. A copy of the whatever is being displayed will be put into the image. It is a good idea to immediately press Ctrl-B or select “Image..Backup” to prevent accidentally overwriting the image buffer on subsequent FFT operations. The image now contains a representation of the FFT display which can be saved on disk.

Notes:

1. If the FFT coefficients are not copied, saving the image will save the original untransformed image data regardless of what is being displayed.
2. Because this is a copy of the screen and does not contain any Fourier coefficients, it cannot be used to recover the original untransformed image.
3. To save the Fourier coefficients, use “Other functions..Save FFT”.

7.19 Wavelet transform

Performs a Discrete Wavelet transform on an image. This section assumes the reader is familiar with the theory and uses of wavelet transforms. Good introductions to wavelets can be found in Starck *et al.* [8] and Prasad and Iyengar [9].

Procedure

1. Select a wavelet. This is a file such as `b4_0.0.wavelet` . A variety of wavelets are installed in /usr/local/lib/wavelets when `tnimage` is installed. This step is not necessary for Laplacian wavelet transforms.

2. Check that the image number is correct.

3. Checking "Show Grid" will draw lines demarcating the boundaries of each wavelet resolution.

4. Select algorithm. Currently only "pyramidal" and "Laplacian" are supported.

5. Click "Forward" or "Reverse" to perform the wavelet transform. The image is automatically converted to grayscale and resized to the next higher power of two in each dimension if necessary.

7.19.1 Reconstituting an image from wavelet coefficients

The following settings are used only during reconstitution (i.e., so-called "reverse" wavelet transform).

1. Index range to use - can be `all, none`, or a range such as `0 to 256, 0 - 256,` or `0 256.`The range can be from 0 to the size in pixels of each dimension in the image. Values with indices outside the specified range in each dimension are set to 0. If the default number of levels is used, indices start at 0-3 for the low-detail wavelets, and each successive detail level encompasses the next power of 2 (e.g., 4-7, 8-15, etc).

2. Value range to use - can be `all, none`, or a range such as `-100 to 100`, etc. Values outside the specified range are set to 0. The value of each wavelet coefficient is displayed in the left information window as the mouse passes over it.

3. Value range to ignore - can be `all, none`, or a range such as `-100 to 100`, etc. Values within the specified range are set to 0. The three conditions (index range, value range, and ignore range) are ANDed together.

4. Gray value offset - If checked, the specified value will be added to the resulting pixel value, and the image contrast will be rescaled appropriately. This is useful if the selected wavelet produces negative values, as often happens if the low-detail coefficients are changed. Reconstituted pixel values are always coerced into the range of $[0, 2^{bits/pixel}]$; thus, a negative value would produce a large region of black unless an offset was given.

7.19.2 Other options

1. Save coefficients - writes the selected wavelet-transformed image to a file.

2. Read coefficients - reads a file containing a wavelet-transformed image (see below for file format).

3. Restore original - Replaces the image with its backup copy if present, to facilitate experimentation with different wavelets.

4. Copy into image - Creates a new image and copies the transformed image into the new image.

7.19.3 Selecting a wavelet

A number of wavelets with different properties, including several "Bathlet" orthonormal wavelets [10], biorthogonal wavelets [11], "Coiflets" [12], Daubechie's wavelets [13], UCLA wavelets [14], and the FBI fingerprint wavelet used for compressing fingerprint images [15] are automatically installed in `/usr/local/lib/wavelets`. The default search path can be customized for each user by editing the `.tnimage` file.

If the Laplacian algorithm is used, selecting a wavelet is not necessary. The Laplacian algorithm uses the specified binomial filter length.

7.19.4 Wavelet file format

The format for `.wavelet` files is described here, in case it is desired to create new wavelet functions.

- All data are stored in ASCII format. Lines are separated by newline or return characters which may be platform-dependent.
- Lines beginning with '#' are comments. This should include, at minimum, the wavelet's name and a literature reference, and possibly its properties.
- Blank lines are permitted anywhere, and are ignored.
- The minimum and maximum indices immediately precede the filter coefficients. The minimum index need not be 0. Example: `0 3` indicates that 4 coefficients will follow.
- The decomposition filter coefficients are next, one per line.
- The minimum and maximum indices for the reconstitution filter coefficients, if present, are next. The minimum index need not be 0. If no reconstitution filter is present, it is assumed to be identical with the decomposition filter.
- The reconstitution filter coefficients are next, one per line.

7.19.5 File format for wavelet-transformed images

Wavelet-transformed images are saved in the following format. All values are in ASCII format. Lines are separated by newline or return characters which may be platform-dependent. The data are stored in a rectangular array with 0's added if necessary to maintain an equal number of coefficients per line.

1. Line 1: `Wavelet transform of` followed by the title of the original image.
2. Line 2: `xsize` followed by the number of coefficients per line.
3. Line 3: `ysize` followed by the number lines of coefficients.
4. Line 4: `bits/pixel` followed by the bits/pixel of the original image.
5. Line 5: `format` followed by the name of the format used by the transform. This could be "mallat", "Laplace", "pyramidal", "multiresolution", etc. The format determines the meaning of the coefficients (see *filtering* below).
6. Line 6: `levels` followed by the number of levels used by the transform.
7. Line 7: `xminres` followed by the x size in pixels of the low-pass component.
8. Line 8: `yminres` followed by the y size in pixels of the low-pass component.

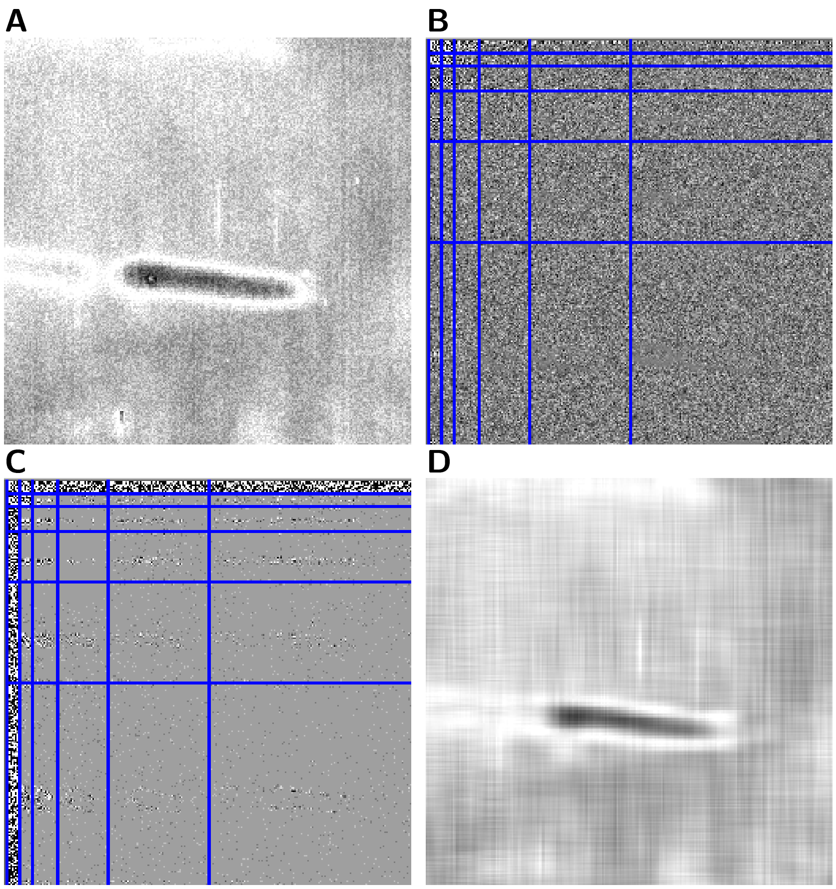
9. Line 9: `wavelet` followed by a path or filename indicating which wavelet function was used.
10. Subsequent lines contain *ysize* lines of data with *xsize* coefficients per line, with coefficients separated by spaces.

7.19.6 Filtering and convolution using wavelets

The coefficients of the wavelet-transformed image can be modified by typing or drawing on the transformed image, or a rectangular or irregular-shaped region can be selected with the mouse and modified by the usual operations for modifying pixels in an image, such as delete (by pressing the delete key, which sets the selected coefficients to 0 if the background is black), contrast, subtract pixel value, image algebra, etc. The transformed image can also be zoomed in or out, or edited in the spreadsheet like a regular image. This gives great flexibility in manipulating the data in wavelet space.

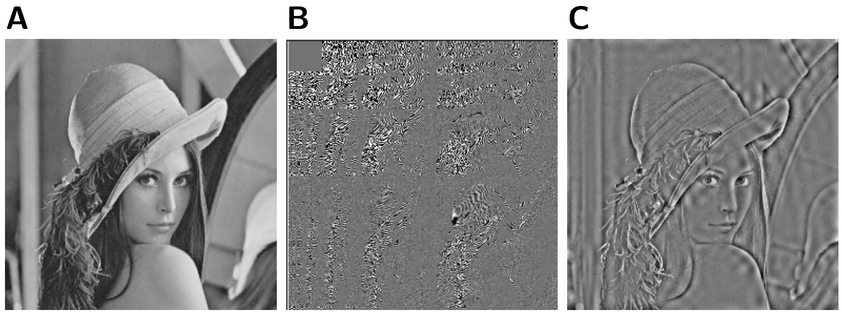
Filtering consists simply of altering the wavelet coefficients and then reconstituting the image. For example, suppose it is desired to remove the high-frequency noise from the image below (**A**). Somewhat arbitrarily selecting a Daubechie's-16 wavelet (`dau16.wavelet`) and transforming produces **B**. This is a pyramidal representation of the wavelet coefficients. Each level of detail is demarcated by blue lines, with the highest detail in the lower right; the residual smoothed image (i.e., the lowest detail) is the tiny box in the upper left. The off-diagonal boxes are diagonal coefficients.

To remove the noise, the background color was set to 0 and the areas shown in **C** were selected with the mouse and set to 0 with the delete key, thus removing most of the detail coefficients. Reverse-transforming produced the de-noised image **D**.



Noise removal using wavelets.

If the low-detail coefficients are removed instead, an outline or edge-enhanced image is produced instead:



Wavelet filtering.

In this image, all the lowest-detail coefficients were removed, creating the small gray box in upper left of **B**. This produced the image in **C**. When removing low-detail coefficients, it is usually necessary to use a gray-scale offset (128 was used in this case) to ensure positivity of the resulting pixel values. Afterwards, the intensity contrast and/or grayscale mapping may need to be adjusted to restore the image balance.

It is recommended to use the "Change size" command to resize the image to a power of 2 in each direction before performing wavelet transformations, instead of letting the algorithm enlarge it, to avoid artifacts caused by the additional background pixels. The artifacts can arise because the new pixels are set to black, which can create a sharp edge in the image.

Convolution: Convolution is carried out by adding the coefficients of two transformed images. Deconvolution is also possible. These can be done using image algebra (see Sec. 12).

7.19.7 Copying Wavelets into an image

Occasionally, it is desired to copy the wavelet coefficients into an image (for example, to illustrate the appearance of a wavelet transform). This can be done by clicking on "Copy wavelets→image". A copy of the whatever is being displayed on the screen will be put into the image. It is a good idea to immediately press Ctrl-B or select "Image..Backup" to prevent accidentally overwriting the image buffer on subsequent operations. The image now contains a representation of the wavelet display which can be saved on disk.

Notes:

1. If the wavelets are not copied, saving the image will save the original untransformed image data regardless of what is being displayed.
2. Because this is a copy of the screen and does not contain any wavelet coefficients, it cannot be used to recover the original untransformed image.
3. To save the wavelet coefficients, use "Other functions..Save coefficients".

Notes

- For grayscale images having depths that are not a multiple of 8 (e.g., 12-bit images), maximize the contrast ("Color..Contrast..Maximize Int.") before performing a wavelet transformation.
- Best results are obtained if the x and y dimensions are the same size.
- Contrast and brightness adjustments made while the transformed image is being displayed are automatically scaled to wavelet coefficients. Thus, increasing the contrast of some pixels by a factor of 2 will multiply the corresponding wavelet coefficients by 2.
- Negative wavelet coefficients can only be entered using image algebra or by entering the negative value in the spreadsheet.
- The image buffer is automatically enlarged if necessary to display the wavelet coefficients.
- Images cannot be simultaneously transformed by FFT and wavelets.

7.20 3D Plot

Creates a wireframe representation of the currently-selected image. A dialog box allows interactive adjustment of various graph parameters.

1. Img.# - The image to be graphed (should not need to be changed).
2. x° - Angle around x axis to rotate the graph.
3. y° - Angle around y axis to rotate the graph.
4. z° - Angle around z axis to rotate the graph.
5. x scale - Magnification in x dimension (0.01 – 10x).
6. y scale - Magnification in y dimension.
7. z scale - Magnification in z dimension.
8. x ctr - x coordinate of image to use as x axis of rotation. If this value is set appropriately, the graph should remain centered when rotating around x axis.
9. y ctr - y coordinate of image to use as y axis of rotation. If this value is set appropriately, the graph should remain centered when rotating around y axis.
10. z ctr - z coordinate of image to use as z axis of rotation. If this value is set appropriately, the graph should remain centered when the z scale is changed.
11. x trans - x translation for positioning of graph.
12. y trans - y translation for positioning of graph.
13. z trans - z translation for positioning of graph.
14. granularity - the degree of fineness of the graph. Lower values give more detail, but take longer to render.
15. color - Selects method by which graph should be colored.
    (a) 0 = all white (not recommended for surface plots)
    (b) 1 = grayscale, intensity is proportional to z
    (c) 2 = original colors
16. Invert Z - Multiply image pixel values by -1 to swap white and black coordinates.
17. Rotate X - Continuous animated rotation about x axis.
18. Rotate Y - Continuous animated rotation about y axis.
19. Rotate Z - Continuous animated rotation about z axis.
20. Surface - Toggle between wireframe and surface plot.

Notes:

1. If the image on which the graph is based is unloaded, the graph still is visible but becomes unmodifiable. Unloading the image while the graph is being adjusted produces unpredictable results.

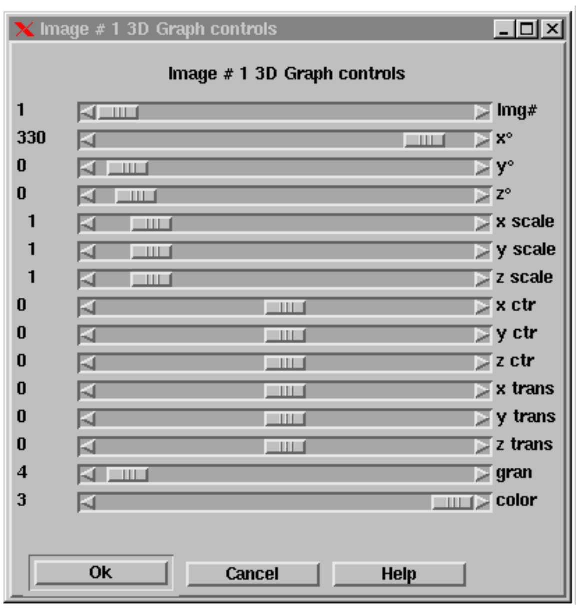
2. The x,y, and z scale magnify the graph in the original xyz coordinates, which will usually differ from the screen x-y coordinates.

3. The graph is handled like any other image, and can be saved and printed. Currently there is no provision for saving in a vector format; graphs are saved as bitmap images. Future versions will have vector printing in PostScript format.

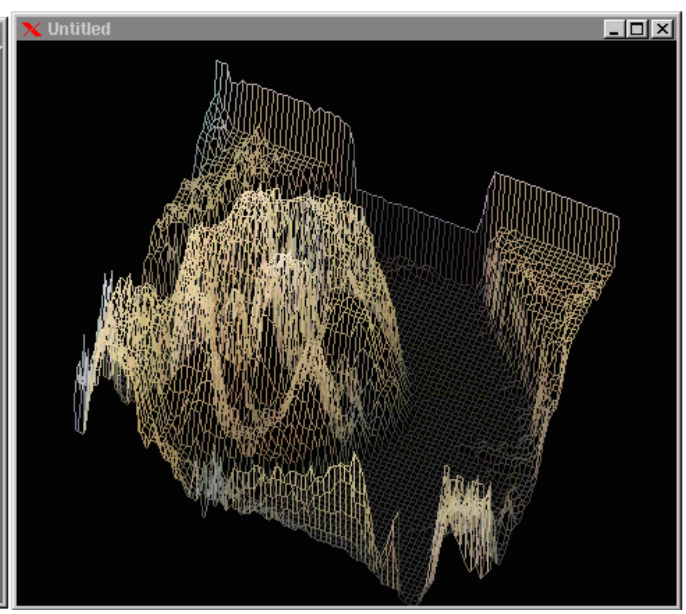
4. Small graphs ($< 200 \times 200$) are redrawn continuously as the scale indicator is dragged. Larger graphs are only redrawn when the mouse is unclicked after changing a value. If you have a fast computer, this behavior can be modified by editing the section after the line "`Callbacks for x,y,z rotation sliders`" in `xmtnimage41.cc` and recompiling.

5. For images larger than the width of the screen, it is recommended to use "Image..Change Size" to make a smaller copy of the image before creating a graph.

6. If the original image is still present, the dialog box can be reopened at any time to adjust the graph, by clicking on the desired graph and selecting "Process..3d graph" again. If you select the image first, a new graph will be created instead.
7. Not all image operations are functional on graphs.
8. Future versions of `tnimage` will allow interactive rotating of the graph by dragging on parts of the graph.
9. For color images, better results are obtained if the image is converted to screen bits/pixel first.
10. For color mode 1 (grayscale), the intensity varies with the plotted z value. If the graph is too dark, it can be lightened by translating on the z axis. Similarly, changing the z scale adjusts the grayscale gradient of the plot.
11. Grayscale images can be plotted in color (e.g., a color spectrum with high z values encoded by red and low encoded by blue) by the following procedure:
    (a) Click on the graph to select it.
    (b) Convert the image to color ("Color...Gray scale→color").
    (c) Select desired colormap ("Color..Colormap/False color..Select colormap").
    (d) Click on one of the 3d controls to rebuild the graph. If this has no effect, click on the graph again and press Ctrl-R to rebuild the graph image.
    (e) Spectrum and multicolor 1 give the best results.
12. If the Z scale is too small, this indicates the contrast in the image is too low. This can occur with 10- and 12-bit images, which are stored in memory as 16 bits/pixel. To correct this, select "Color..Contrast" and click "Maximize Value".



3d Plot dialog



Wireframe plot

Section 8

Color menu

In color images, each pixel value is a number that represents the color of that pixel. In indexed-color (8-bit/pixel) images, on the other hand, the pixel value has no fixed meaning. The image uses a colormap which defines the relationship between pixel value and color. If an 8-bit (256-color) image which lacks a colormap of its own is loaded and converted to color, the most recently-selected colormap will be used to perform the conversion.

There are several different ways of changing the colors of an image:

Change colors The brightness of each red, green, or blue value is adjusted by dragging control points that define a brightness curve. This is the most powerful method, but it is also more difficult to change multiple images in precisely the same way because it is interactive.

Brightness Changes the red, green, blue, hue, saturation, or value component of each pixel by adding or subtracting some user-specified number. The change is easier to reproduce and can also be done in 'image math'.

Gamma Changes the gamma value of an image. This makes dark areas lighter without affecting normal areas. Ideal for repairing underexposed photographs.

Contrast Similar to 'Brightness', except that instead of adding a number, the pixels are decomposed into their RGB or HSV values and the R,G,B,H,S, or V components are multiplied by separate, constant factors. For example, you could increase the 'contrast' of the red pixels while decreasing the contrast of blue.

Colormap/False color Allows selection of colormaps for indexed-color or grayscale images. User-defined colormaps are also supported.

Remap colors Switches colors. For example, if some other program created an image in which 'red' appeared 'blue', this option could repair the image.

Change pixel value The most basic operation, adds or subtracts a raw number from the image. This is useful for grayscale images, which can be made lighter or darker by a precisely-defined amount.

Each of these options is described in greater detail below.

8.1 Change colors

This method allows the greatest flexibility in setting the brightness. A 3-part graph is displayed showing the relationship between pixel value and intensity for red, green, and blue.

Tnimage differs from some other programs in that color editing is done in full 24-bit mode for 24 bit/pixel images, rather than simply converting the image to an 8-bit indexed image. In no case does tnimage simply take the shortcut of altering the colormap for true color iamges, which would cause tremendous loss of image color detail.

For grayscale images, use "Change Gray Values" instead; or use "change gray values" instead or select 'Color — Convert to color' before proceeding.

Procedure

1. Click "Image..Backup" or press Ctrl-B to backup the image before starting.
2. Click the left mouse button in the graph to create a curve which defines the desired mapping of pixel value to color. This will create small boxes (control points) which can be repositioned within the graph by dragging them. Each time you click, a new point is added.
3. Click the 'Finished' button on the lower left of the graph window when you are satisfied with the new curve. The graph will then be redrawn and the new color settings will be applied to the image. If you click "Accept" without clicking "Finished", the image will not be changed.
4. Repeat with other colors as desired. The graph can also be smoothed, saved to disk, etc. by clicking on the other buttons at the left of the graph. The Automatic Baseline button and a few other buttons are disabled. See 7.10.2 for more information about the buttons at left.

For 8 bit/pixel modes, the colormap (palette) of the image is changed and the pixel values in the original image are unaffected. It is not possible to change the color of only part of the image for 8 bit images with this method. To change only a portion of an image, you must convert the image to 24 bits/pixel first.

8.2 Brightness

Adds or subtracts a value from red, green, blue, hue, saturation, or value in the image or selected region. For grayscale images, only "value" can be changed.

8.3 Contrast

Multiplies red, green, blue, hue, saturation, or value in the image or selected region by some value to change the contrast. For grayscale images, only "value" can be changed.

NOTE*WARNING**
Adjusting the brightness or contrast may invalidate the correspondence between pixel values and the original optical density in the image, making numerical results obtained by subsequent densitometry scientifically invalid.

Changing brightness using the colormap palette bar

This method is very fast, but only works on 8-bit images that have a continuous colormap or with grayscale images.

Click on the top part of the bar and drag down to "squash" the colors downward, or click on the bottom part of the bar and drag up to squash the colors upward. This increases the contrast in all images of the screen.

Then, click in the middle of the bar and drag up or down to adjust the brightness as desired. The currently-selected image will be set to contain the new colormap.

If you change the colormap from the menu, or select "restore original colormap", the brightness and contrast settings return to normal.

Maximizing contrast

Clicking on the "Maximize" buttons in the Contrast dialog will automatically maximize the red, green, blue, or grayscale intensity contrast in the current image or selected region. In most cases, it is desired to maximize the intensity contrast; however, the red, green, and blue components can be maximized independently by clicking on the corresponding "Maximize" button. This may be useful, for instance, on images from confocal microscopes. For example, in a 24-bit image, if the red pixel values range from 50–100, maximizing the red will stretch the red range to 0–255.

For grayscale images, the display may not appear to change after maximizing contrast, because the grayscale mapping already maximizes the grayscale intensity range that is mapped to the screen.

On occasion, the image may not appear to become fully maximized. Usually this is the result of stray black or white pixels which must be removed manually or using image math.

In some cases, the display may temporarily appear incorrect until the OK button is clicked.

8.4 Change colormap / false-color

In 8-bit/pixel modes, each pixel value is translated through a "colormap", a look-up table which you can easily modify in `tnimage`. The new table is saved along with the image (for TIF, BMP, and TGA files) so the next time you load the image, the display changes to the modified colormap. Changing the colormap has two advantages over other methods of changing brightness or color: (1) it is much faster, and (2) it is always reversible since it never affects the original image data. The disadvantages are that changing the colormap only works in indexed color screen modes, and it always changes the entire screen. If you need to brighten just a portion of the screen, use the "color intensity/brightness" menu option.

In color modes, the current colormap table is used as a guide to determine how an monochrome image is converted to color when it is loaded from disk, but does not otherwise affect the display.

It is usually convenient to select "Show colormap" while selecting a false-color palette, as the colormap display is updated as soon as the palette has changed. (Be sure to click on an 8-bit image before doing this, so the colormap display shows the 8-bit colormap and not the true-color 'colormap').

Invert colormap

Switches the colormap so that all images become a negative. Changes the appearance only, and has no effect on the actual pixel values in the image.

Select colormap

You can select from a variety of different color colormaps. Changing the colormap has no effect on the image when it is stored or densitometrically analyzed. However, it can be used to enhance the visibility of details in the image. Each image can have a different colormap.

The following pre-set colormaps are available:

Gray scale - continuous shading of 256 levels of gray.

Multi-color 1 - a continuous blend of colors useful for enhancing low-contrast images.

Multi-color 2 - a discontinuous sequence of short color gradients

Multi-color 3 - a discontinuous sequence of color gradients arranged oppositely from multi-color 2.

RGBI - 4 continuous gradients, 0-63 = red, 64-127 = green, 128-192 = blue and 193-255 = gray scale.

Spectrum - A single continuous gradient from blue through the colors of the spectrum to red and then white.

Black to green - Same as gray scale except black to green instead of black to white.

Zebra - Alternating black and white - useful for finding very subtle patterns in the image. Creates a contour-line effect.

Brightness - Lightens or darkens colormap, increasing or decreasing color saturation.

Other - User-selected colormap. Select the desired number with the mouse to select from a wide range of different computer-generated colormaps.

1-100 Various gray scales

101-1000 randomly-generated gradients. The amount of color gradually increases in intervals of 100.

1000-10000 continuous, smooth gradients without sharp breaks in the colors. The number of light and dark cycles varies from 1 upwards. Some are a single color, others are gray scales, and others are multiple colors.

Example: 2345 = shades of 'gold'

Read colormap

Reads a colormap file from disk. Colormap files are ASCII text files which can be created with a text editor, or created graphically within `tnimage` (see *Create colormap*).

If you create a colormap file manually, your colormap file should consist of 4 columns of up to 256 rows. The first column is the color number or 'index', an integer between 0 and 255, indicating which color. The other 3 columns are integers between 0 and 63, which define the amount of red, green, and blue respectively for each color. For example:

```
  0    0    0   63
  1   63   63   63
255    7    7    7
```

This file would set color 0 to light blue, color 1 to bright white, and color 255 to dark gray. All other colors would be unchanged (Normally, a colormap file would have 256 rows instead of only 3).

Save colormap

Saves the current colormap into a disk file.

Create colormap

Graphically creates a custom colormap. There are 3 boxes, one for red, green, and blue, which graphically represent the intensities of each color in the colormap. To change the colormap, click on the graph corresponding to the desired color and trace out the desired new graph while holding the left mouse button down. The graph and colormap will both adjust in real time to reflect the new values.

Clicking 'smooth' smoothes out the newly-drawn curve to create a more even dispersal of colors in the colormap.

Rotate colormap

Rotates the colormap table through all 255 colors. This effectively increases the number of predefined colormaps to 2.5 million. Useful in achieving fine control of the appearance of the image and enhancing subtle details.

If you find a number that gives a useful effect, write the number down for future reference or save it using "Save colormap". The actual pattern of colors in the colormap is not easily predictable from the color number. (Saving the currently-displayed screen or a portion of it as a TIFF or PCX file, or selecting 'Save colormap' will also save the colormap.)

Restore original colormap

Restores the colormap the image had when it was originally loaded from disk. Useful when you accidentally rotate the colormap of a GIF file.

Sort colormap

Sorts the colormap from lowest to highest luminosity and remaps the image accordingly. This is often an essential step before performing densitometry on an image, or before setting chromakey for an image, as these functions rely on a correspondence between the raw pixel value and intensity.

Remap to other colormap

Sets each color in the currently-selected image, equal to the closest color in a second user-specified image, and then copies the other image's colormap. This permits creating two 8-bit images that have the same colormap. This is useful in creating images for Web pages, where it is desirable to use a minimum number of separate colors. If someone is viewing your Web page with an 8-bit display, a large number of images with different colormaps can result in poor overall quality because the browser will run out of colors.

False color map for grayscale

Grayscale images can be displayed in false colors instead of shades of gray by setting the "false color" map. This allows a large range of gradations to be distinguished at the same time, or a narrow range to be highlighted. The colormaps numbered 0–199 are particularly useful for this, as they contain several repeated grayscale gradients.

The false-color map is global and not attached to the image. This has several implications:

- All grayscale images will be changed to the new false colors.
- The false color map is not saved with the image when the image is written to disk.
- The false colors are not printed with the image.
- False color does not affect densitometry, calibration, or FFT results.

(To attach a colormap to an image, the image would have to be converted to "indexed-color" or "color" first.)

Remember that a false-color image is, despite appearances, still a grayscale image as far as the computer is concerned. Thus, selecting "invert colors" will merely invert the false color map, instead of changing red to green, yellow to blue, etc. To change the false-color mapping, you must still select "Grayscale mapping", in the same manner as a regular grayscale image; dragging the palette bar, selecting "Color/contrast", or other operations that act on color images will not work.

Note: If for some reason the image does not get redrawn properly after changing the false-color map, try pressing Alt-R to repair the image.

Reset grayscale to default

Sets the false-color map back to its default setting, i.e. a continuous gradient from black to white.

8.5 Grayscale intensity mapping

Grayscale images of greater than 8 bits/pixel often contain subtle details which would not ordinarily be visible. This option permits interactively changing the way the pixels are mapped to the screen, creating, in effect, a "sliding scale".

This sliding scale uses 4 parameters: the maximum and minimum values in the image to display, and the maximum and minimum values on the screen to which these are mapped (these latter 2 do not normally need to be changed). When a monochrome image is first loaded, `tnimage` automatically determines the maximum and minimum values in the image to allow the entire image to be viewed. By selecting "Grayscale brightness", you can use the mouse to select the actual range of values to be displayed.

For example, if the image has 12 bits of grayscale depth, it will be possible to set the maximum and minimum to any number between 0 and 4096. Setting the numbers close together enhances the contrast in that intensity range. Similarly, making the numbers small enhances the contrast in the darker areas of the image.

The maximum can also be lower than the minimum, creating an inverse image. Each image on the screen can have a different brightness/contrast level.

Note: Color images must be converted to monochrome before this feature can be used.

If a "false color" colormap is set, this option will change the mapping of false colors for the image (See Sec. 8.4).

8.6 Remap colors

Changes the colors in an image by substituting from a map file. It acts differently on indexed color and color images.

8-bit/pixel images:

Changes the pixels in the image or screen region by substituting from a set of values specified in an ASCII file. This file should consist of a list of 2 columns of numbers. The 1st column is the pixel value to change, the 2nd column should be the desired new values. For example:

```
  1      255
  2      200
  3      180
  4      175
255        0
```

This file would change all pixels with a value of 1 to pixels with a value of 255, those with values of 2 to 200, 3 to 180, 4 to 175, and 255 to 0. Any other pixels would be unaffected. Typically in a remap file, you would remap all 255 colors, but this is not essential. Be sure colors 0 and 255 do not accidentally get remapped to the same value, otherwise it will be impossible to use the menus (this applies only to DOS version).

The most common use of this is to reduce the number of separate intensity values in an image, by eliminating those values in regions that are not of interest. This can greatly improve the compressibility of an image.

Another use is to stretch the contrast. Since the new intensity values can be known in advance, contrast enhancement can be done on multiple images in an exactly reproducible manner.

The Macro Editor can be used to create remap files without leaving `tnimage` .

Color images:

Interchanges r,g, and b color values. This could be used, for example, to convert an image to a single color or to correct for color errors in a corrupted image file.

8.7 Invert colors

Changes the image into a negative. This differs from "invert colormap" in that the pixel values themselves are changed.

8.8 Change color depth

Converts the selected image to a different number of bits per pixel. This option only works on entire images. When working with indexed-color images, it is advisable to use this option to convert them to color images before filtering them (see "Filtering color images") or using other operations that assume the image is color. Images can be converted from any depth (8, 16, 24, or 32 bits per pixel) to any other depth.

If the starting image is an 8-bit image with a colormap, the colormap colors are used to select the appropriate colors for the new image. These can be easily changed (see *Change colormap*).

If the starting image is a color image, and it is being converted to 8 bits/pixel, the currently-selected palettization method is used to calculate the colors. The default method is "quantization". This can be changed (see *Color Settings*). 3D images are always converted using the closest-fit method.

Once converted, writing the image to disk will cause it to be saved in its new depth.

Warning: For grayscale images, it is essential to maximize the contrast ("Color .. Contrast .. Maximize Value.") before changing image depths, converting to color, or most other operations. Otherwise, a 10-bit image converted to an 8-bit grayscale image may appear completely dark, while copying a 10-bit image into a 32-bit image buffer (for example) may cause it to appear with excessive contrast.

8.9 Color and Gray scale

Color →Gray scale

Converts the image from color or indexed-color to gray scale (luminosity). The relative contributions to luminosity from the red, green, and blue components are determined by the "Luminosity factors" which can be changed by selecting "Color...Color settings".

The default is:

$$value = .299 * r + .587 * g + .114 * b$$

The results for 8-bit images may be different from the results of merely changing the colormap to gray scale, because for 8-bit images, the conversion creates a new colormap to match the luminosities in the image as closely as possible.

Gray scale →color

Deactivates grayscale mapping, reactivates color manipulation features, and tries to restore previous color information if possible.

WARNING: Converting color to grayscale and back is not always a reversible process.

8.10 Separate colors

Creates 3 8-bit monochrome images containing the red, green, and blue components of the original color image. This is useful, for example, in images of silver-stained gels, in which the signal is primarily in the red channel. The gel can be scanned in 24-bit color and the green and blue channels thrown away, for a crude form of color filtering.

Future versions of `tnimage` will split 36- and 48-bit color images into 12- and 16-bit monochrome images. Contact the author for availability.

8.10.1 Composite image

Combines 3 8-bit monochrome images containing the red, green, and blue components into a 24-bit color image. All 3 images must be the same size.

8.11 Histogram

Plots a histogram of the no. of pixels with a given value within the selected area vs. the pixel value. Quite useful when trying to optimize the contrast or brightness of an image.

Clicking *capture* converts the graph into a new image.

Clicking *subtract baseline* allows you to subtract a baseline from the histogram. Select the desired baseline by clicking as many times as desired to create the baseline curve. The baseline curve is constructed using a B-spline (see under *B-spline curve*). You can modify the curve as desired by clicking and dragging the small boxes (which represent control points for the curve). Up to 200 control points can be used. The control points do not have to be within the graph region; however, any negative numbers created by baseline subtraction are truncated to 0's.

See *strip densitometry* above for more details on the plot functions.

Note: When a graph is being displayed, smoothing, subtracting baseline, etc., affect the displayed data only, and have no effect on the image.

8.12 Histogram equalization

Histogram equalization is a process of adjusting the image so that each intensity level contains an equal number of pixels. In many cases, this can improve the appearance of the image by balancing light and dark areas. Other times, it degrades the image's appearance by accentuating image defects. This is particularly true with JPEGs.

`Tnimage` uses a new histogram equalization algorithm that redistributes pixels into equal-size bins much more equitably than other algorithms. This produces a noticeably smoother result than other programs.

One problem that arises when equalizing color images is that if the red, green, and blue are equalized independently, as `tnimage` does, this can alter the color balance of the image. If you want to remove this feature, change the line

```
#define COLORS_EQUAL 0
```

in `xmtnimage4.cc` to

```
#define COLORS_EQUAL 1
```

and recompile `tnimage`. This will reduce the amount of equalization that is performed, and more closely mimic other software. Alternatively, you can readjust the color balance with the Contrast or Intensity controls in `tnimage`.

8.13 Change pixel value

Changes the raw pixel values in an image or selected region. "Change pixel value" is particularly useful for making grayscale images lighter or darker. It allows precise changes to be made to the brightness level. The effect is consistent across all image types and screen modes but behaves differently for color and grayscale images.

For grayscale images:

This adds a value to all pixels in the currently-selected image or screen region, making it darker or lighter. Works in conjunction with 'Grayscale map'.

For color images:

This adds or subtracts a value for the red, green, and blue components of the currently-selected image or screen region, making it darker or lighter or altering its color balance.

8.14 Change Gray Values

This method allows the greatest flexibility in setting the brightness of grayscale images. A graph is displayed showing the relationship between pixel value and intensity for the selected image.

For color images, use "Change Colors" instead.

Procedure

1. Click "Image..Backup" or press Ctrl-B to backup the image before starting.
2. Click the left mouse button in the graph to create a curve which defines the desired mapping of pixel value to change. This will create small boxes (control points) which can be repositioned within the graph by dragging them. Each time you click, a new point is added.
3. Click the 'Finished' button on the lower left of the graph window when you are satisfied with the new curve. The graph will then be redrawn and the new brightness settings will be applied to the image. If you click "Accept" without clicking "Finished", the image will not be changed.
4. The graph can also be smoothed, saved to disk, etc. by clicking on the other buttons at the left of the graph. The Automatic Baseline button and a few other buttons are disabled. See 7.10.2 for more information about the buttons at left.

Section 9

Draw menu

Most of the drawing operations listed below put `tnimage` in drawing mode, in which each mouse click creates another graphic element. To return to normal mode, click anywhere on the top menu bar or the *Cancel* button.

9.1 Text

9.1.1 Ordinary labels

Text labels can be added to an image by simply moving the mouse cursor to the desired location and typing. The text is added in the currently-selected bitmap font.

9.1.2 Labels

Label

Adds a label.

9.1.3 Drawing text at an angle

It is possible to select any angle for the text. The text is automatically anti-aliased as it is rotated so that it appears smooth regardless of the angle.

Text may be placed at any angle by selecting "Draw...Label" and typing the label. Once the label is typed, a clickbox will appear that permits selection of the angle in degrees measured counterclockwise from horizontal. Normal text is 0 degrees.

In the Unix version, you can also add labels by selecting some text in a text window with the mouse. Click the middle mouse button to paste it onto an image in the desired location.

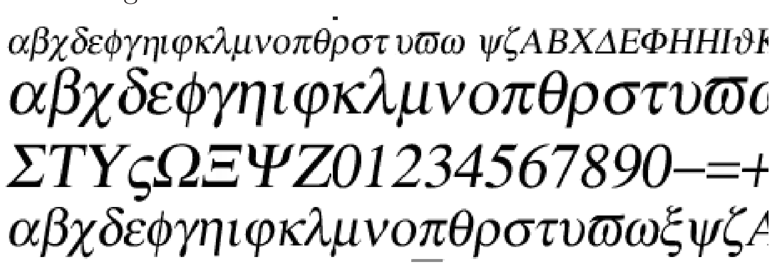
Larger amounts of text can be added by opening the text file in the same manner as opening an image. The text file must have the extension ".txt". Open the text file with "Open image" and click at the position where the text should be placed.

9.1.4 Freetype labels

Adds a label using Freetype font. Freetype is anti-aliased so that it appears smooth. The label must be contained within a single image. Because support for anti-aliased fonts is new in Linux, there may be problems with Freetype labels.

9.1.5 Special characters

Tnimage can import any image file and use it as a source of new anti-aliased fonts. By default, tnimage looks in /.tnimage for font image files. When the font image file is opened, it is treated differently from a normal image. To add a special character, click on the desired symbol in the font image and drag it to its new location. The symbol will be pasted onto the topmost image using the currently-selected foreground color. If the font image contains anti-aliased characters, the symbol will be treated as an anti-aliased character. If the image on which the symbol is pasted is grayscale, the symbol will be converted to grayscale before being applied to the image.



Sample font image for Greek letters.

This feature allows users to create their own font files for special characters, commonly-used text strings, or icons.

Notes

1. To use an image as a font file, be sure to use "Draw...Special Character", not "Open image".

2. Fine positioning of the symbol can be achieved by using the arrow keys. Drag the symbol to the desired location with the mouse. Then, continue holding the mouse button and press the keypad arrows (1-9). The mouse cursor and symbol will move a fixed distance in the corresponding direction. When the symbol is in the correct location, release the mouse button. The distance by which the cursor moves can be increased or decreased by pressing the gray (+) or (-) keys on the numeric keypad.

3. To create a label consisting of more than one character, set the background pixel value of the image containing the text to 253 or 254. This will cause the entire string to be selected.

4. Up to 510 different font images can be displayed at any given time. This number can be increased by changing MAXIMAGES in xmtnimage.h and recompiling tnimage.

5. Since tnimage is an image editing program and not a drawing program, the only way to edit the text after it has been added to the image is with the "undo" or "erase" functions.

9.1.6 Creating new fonts

To create a new font for tnimage, do the following:

1. Using your favorite word processor, create a text file containing the desired characters. Make sure the characters are far enough apart so they don't touch each other.

2. Convert the text file to TIFF or some other image format. This can be done by displaying the text file and capturing a screen image with xwd or tnimage, or by creating a PDF or PostScript file and converting it to an image format. For example, here is the script I use to convert .dvi files from LaTeX to PDF format:

```
dvips -Pamz -Pcmz -o $1.ps $1.dvi
ps2pdf -dMaxSubsetPct=100 -dCompatibilityLevel=1.2  \
    -dSubsetFonts=true -dEmbedAllFonts=true \
    -dAutoFilterColorImages=false -dAutoFilterGrayImages=false \
    -dColorImageFilter=/FlateEncode -dGrayImageFilter=/FlateEncode \
    -dModoImageFilter=/FlateEncode  $1.ps
```

   Alternatively, you could use ps2gif to create a .GIF file.

3. Use tnimage to convert the font image file to 8 bits/pixel and grayscale.

4. If any symbols in the font image are touching each other, separate them using the "sketch" function in tnimage to place pixels with a value of 255 (white) between them. All symbols must be completely surrounded by white pixels.

   (a) Click on the font image.
   (b) Use "About the image" to make sure it is 8-bit grayscale. If not, change it to grayscale first, then change the image depth to 1 byte/pixel (=8 bits/pixel).
   (c) Check the color of the background between the symbols. If it is less than 255, increase the contrast so the background is uniformly 255.
   (d) Click on the font image and set the foreground (drawing) color to 255.
   (e) Open the floating magnifier.
   (f) Move to the point that needs to be fixed, and press F2 to go into sketch mode.
   (g) Use the arrow keys to set pixels to 255 between the overlapping symbols.

5. If any of the symbols consist of two or more parts, connect the two parts together by adding pixels of a value of 253 or less between the parts, using the procedure described above.

6. Save the font image file in /.tnimage or wherever convenient.

The font image file acts as a menu from which any number of symbols can be selected.

9.1.7 Subscripts and fonts: Embedded LaTeX-like commands

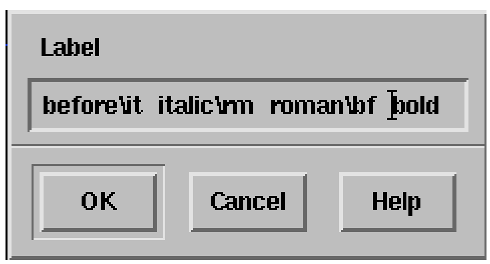
In the label clickbox, a limited subset of rudimentary LaTeX-like commands may be embedded in the string. These include:

- \it (italic font)
- \rm (roman font)
- \gr (greek font)
- \sf (sans serif font)
- \tt (typewriter (courier) font)
- \bf (boldface weight)
- \med (medium weight)
- \font *fontname* (specify font name, e.g. helvetica, times, etc.)

- \size *size* (specify desired font size (`xlsfonts` provides a list))
- \up *distance* (superscript)(moves remainder of string up by specified no. of pixels)
- \down *distance* (subscript)(moves remainder of string down by specified no. of pixels)
- \left *distance* (moves remainder of string left by specified no. of pixels)
- \right *distance* (moves remainder of string right by specified no. of pixels)
- \\A backslash character
- \r (same as "right")
- \l (same as "left")
- \u (same as "up")
- \d (same as "down")
- \s (same as "size")

For example, the string

```
before\it italic\rm roman\bf bold\rm
```



Embedded LATEXcommands in a label.

would produce:

When specifying a font, a font name should be specified that corresponds to the second part of an X-window fontname, for example:

```
before \font courier  more text\font helvetica after
```

will print the label "before more text after" in 3 different fonts.

Greek letters can also be printed, for example:

```
abc\size 14 \up 6 \gr abc
```

will print something like

$$\mathrm{abc}^{\alpha\beta\chi}\ .$$

In making superscripts, it may be helpful to preface the label with a "down" command. For example, the string:

```
\gr\d14 2p\rm e\s14 \u9 -x\u5 2
```

would be rendered as: 

Note: Font commands are cumulative, so a command like

```
\tt a \rm b \gr c \it d
```

would probably give an error because on most systems there is no Italic Greek font. The correct command would be:

```
\tt a \rm b \gr c \rm\it d
```

This would change the typeface back to Roman before changing it to Greek symbols.

Note: LaTeX-like commands only work with the “label” command, and do not work when simply typing text on the image.

9.1.8 Text Font

Currently, fonts are only available on the Unix version. You can select from any font available in X Windows. See the X man pages for additional information. If you add new fonts, delete the file “fonts” so that the font list will be regenerated.

The ‘fonts’ file can be edited to remove undesired fonts, or rearranged as desired.

9.1.9 Pasting text

Instead of typing the text manually, you can paste arbitrarily large quantities of text onto an image. This is an easy way to create slides for presentation or to add preformatted labels. See Section 5.1.3.

Procedure:

1. Select text in some text window by clicking and dragging with the mouse.
2. Position the mouse cursor in tnimage at the upper left corner where the text should be placed.
3. Click the middle mouse button. The text will be converted to the current font and foreground color and drawn on top of the image. The text is also placed in the alpha channel and can be manipulated as usual for any other graphic element.
4. Currently, any text that falls outside the tnimage window is truncated.

Text files can also be loaded in the “File..Open Image” menu as if they were an ordinary image. If the text file ends in the extension ‘.txt’, it will be treated as text and pasted on top of the current image.

9.2 Set foreground color

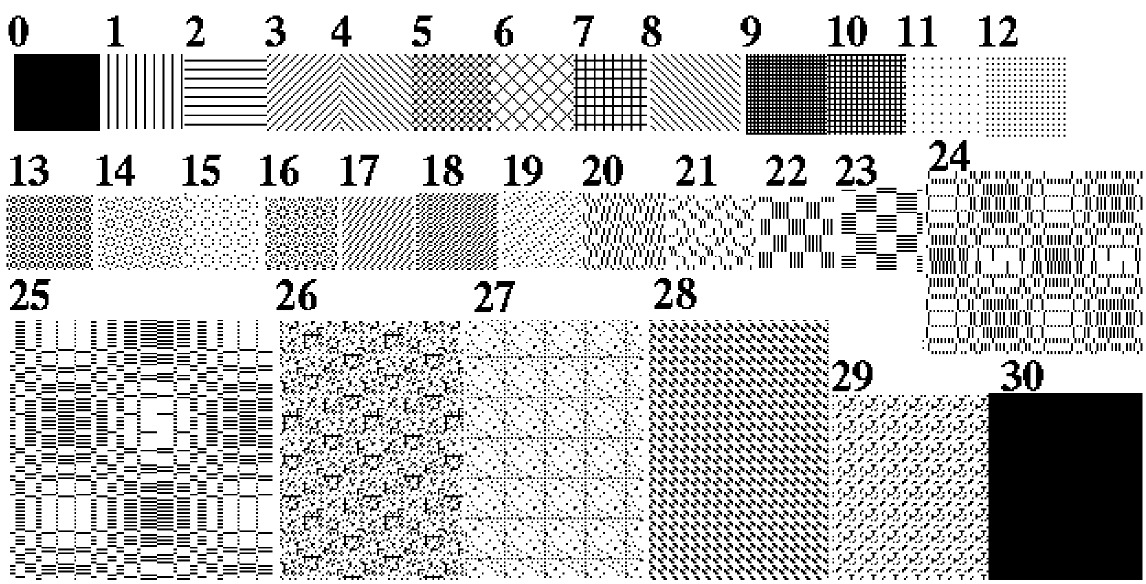
Changes the foreground (drawing) color. The foreground color is used for all text, lines, circles, spray, arrows, etc. The foreground color can also be changed by clicking the left mouse button when the mouse cursor is pointing to the desired color in the 'colormap display' (the long vertical bar or square at the right of the screen).

9.3 Set background color

Changes the background color. The background color is used for the backspace and delete keys, and when erasing an image from the screen. The background color can also be changed by clicking the right mouse button on the colormap display. Because it is possible to have images and text on the background, changing the background color does not immediately redraw the background in the new color. This can be done by selecting "Erase background".

9.4 Set foreground pattern

Sets the pattern for drawing operations. All operations that set pixels, including paint, spray, text, erase, and drawing operations, will be ANDed with the pattern. The pattern may be one of the following:



9.5 Graphics

9.6 Line/Arrow

Draws lines, arrows, or rulers. Click at the starting position, drag to the end position, and then release to draw a straight line. Click the "Cancel" button or anywhere on the top menu bar to return to normal mode.

Once set, the line drawing parameters affect all subsequent lines, boxes, sketch operations, and image borders but not curves and circles.

Transverse gradient: Changes the color across the width of a line according to the specified gradient and center color. A gradient of RGB =128,128,128 is the same as a solid color. A gradient of RGB =127,126,129 decreases the red by 1, the green by 2, and increases the blue by 1 from the top to bottom of the line. If "Double gradient" is selected, the starting color is the center of the line. Of course, the line width must be greater than 1 for this to work.

Arrowhead: Draws an arrow on the line with the tip at the starting coordinate. If the line width is changed, the arrow sizes change automatically to keep the arrow size proportionately constant. It is also possible to customize the arrow:

Width: the maximum width of the arrowhead

Outer length: the length of the arrowhead measured parallel to the line, from the tip to the ends of the two outer points.

Inner length: the length of the arrowhead from the tip to the point at which the arrow touches the shaft.

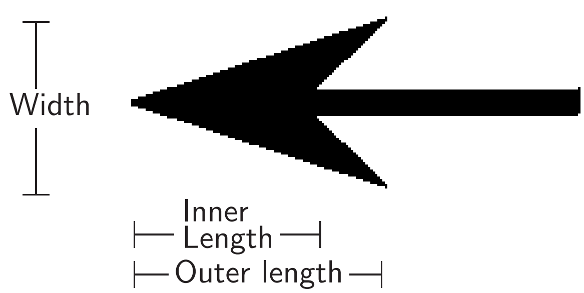
Skew: The number of pixels in the x direction by which the line is shifted for each unit of thickness. This is useful in sketching gradient lines.

Ruler: Adds tic marks perpendicular to line.

Ruler + numbers: Adds tic marks and graduations perpendicular to line. The graduations start at 0 and increment in the direction in which the line is created. The currently-selected font and foreground color are used. The text portion is anti-aliased so it appears smooth at any angle.

Ruler scale: Enter a factor to change the spacing between tic marks in the ruler.

Ruler tic length: Length of the small perpendicular tic marks in the ruler, in pixel units.



Note: On 8-bit displays, the colors produced by the gradient are constrained by the current colormap. If the gradient calculation needs a color that is not contained in the colormap, the closest color is used. This can result in inaccurate color gradients.

9.7 Flood Fill

Fills an area with a single color or a continuous gradient. Does not work on grayscale images.

Solid fills a region with a constant color

Vert.Gradient - fills a region with colors increasing or decreasing top to bottom

Horiz.Gradient - fills a region with colors increasing or decreasing left to right

Normal - Pre-select an area beforehand by dragging a rectangular region with the mouse, or double-clicking on an object. The entire selected region or object will be filled.

Interactive - Click inside each object to fill. The objects must be completely surrounded by a boundary which is between the "lower boundary color" and the "upper boundary color". If there are gaps in the object's border, the color will spill out onto the area surrounding the object. When finished, click on the main "Cancel" button, or press Esc.

Note: "Flood fill" and "Add/Remove Gradient" both can create color gradients, but work in different ways. Flood fill adds color to all pixels in a region bounded by user-specified upper and lower RGB values and therefore determines the area to be colored automatically. "Add/Remove Gradient" adds the gradient to the entire selected area. It is therefore easier to control precisely where the color is added by selecting the desired area beforehand.

9.8 Add/Remove Gradient

Fills the entire selected area with a continuous gradation of color. Does not work on grayscale images.

Method - Whether the gradient is added, subtracted, or multiplied by the pixel values in the selected image.

Upper left gradient value

Upper right gradient value

Lower left gradient value

Lower right gradient value

Values of gradient - The red, green, and blue values to be added or subtracted.

Upper left gradient value

Upper right gradient value

Lower left gradient value

Lower right gradient value

Coordinates scope

Entire image - The specified gradient colors are calculated as a fraction of the entire image, even though only the selected area is affected. For example, if the upper left red value is set to 25, only the pixel in the upper left corner of the image will change by 25. This allows you to add gradients at different parts of the image, while keeping the colors consistent with each other.

Selected area - The specified gradient colors are calculated fresh each time for the selected region. If no region is selected, this option has the same result as "Entire image". For example, if the upper left red value is set to 25, the pixel in the upper left corner of the selected region will always change by 25.

Note that the "starting" color can be larger or smaller than the "ending" color. This gives greater flexibility for creating a blend of one color into another.

Sometimes, as when filling the inside of letters in text, it is desirable to have the appearance of the same gradient extending out of sight between the letters. This can be done easily in `tnimage`, by reusing the same starting and ending gradient colors and setting "Coordinates scope" to "Entire image". Clicking inside each area to be filled sequentially will cause the same colors to be used as if the entire region had been filled with the same gradient.

Special care should be given to starting and ending r,g, and b values when working in 15- and 16-color modes. Slightly different starting and ending values for each color should be specified in order to avoid a "stepped" appearance of the gradient. This prevents all 3 colors from being incremented at the same place on the screen, due to the fact that each color only has 31 or 63 discrete values. Because of this, it is recommended to convert all images to 24 bits/pixel before performing a flood fill.

If a non-rectangular region has been selected, only this region will be filled (see sec. 3.5.3).

Notes:

1. Best results with color gradient fills are obtained with X servers running in 24-bit mode. In 8- 15- and 16-bit display modes, not enough separate colors can be created to produce a high-quality gradient.

2. It is necessary to double-click on the desired region or use the mouse to select a region to fill before performing a flood fill.

3. The gradient fill dialog box behaves differently depending on the pixel depth of this center point. If the center point is an 8-bit pixel, only numbers between 0 and 255 can be entered for fill colors and border colors. If the center point is higher than 8 bits, the individual RGB values can be specified.

4. On 8-bit displays, the colormap is recalculated after the "cancel" button is clicked. Colors may appear incorrect until then.

5. If the image to be filled is grayscale or indexed color, the dialog boxes for selecting the fill color will only have one slider, which selects the grayscale value or color index respectively. Otherwise, the R, G, and B values can be set independently.

6. **8-bit indexed color images must be converted to 24 bits/pixel for satisfactory results. This is not done automatically.**

Example 1. Creating a rectangular area with a gradient increasing in brightness from left to right.

1. Select the region to be filled by double-clicking or by selecting with the mouse.
2. Click on the colormap palette to select a light color, e.g., 250.
3. Select "Draw...Box" and draw a box. Note that the box is drawn in color 250.
4. Make a note of the starting and ending x coordinates of the box (Shown on the menu bar).
5. Select "Draw...Fill region".
6. Click on "Horizontal gradient".
7. Set the starting and ending values of the gradient as desired.
8. Click on "OK".
9. The box should fill up with the gradient.

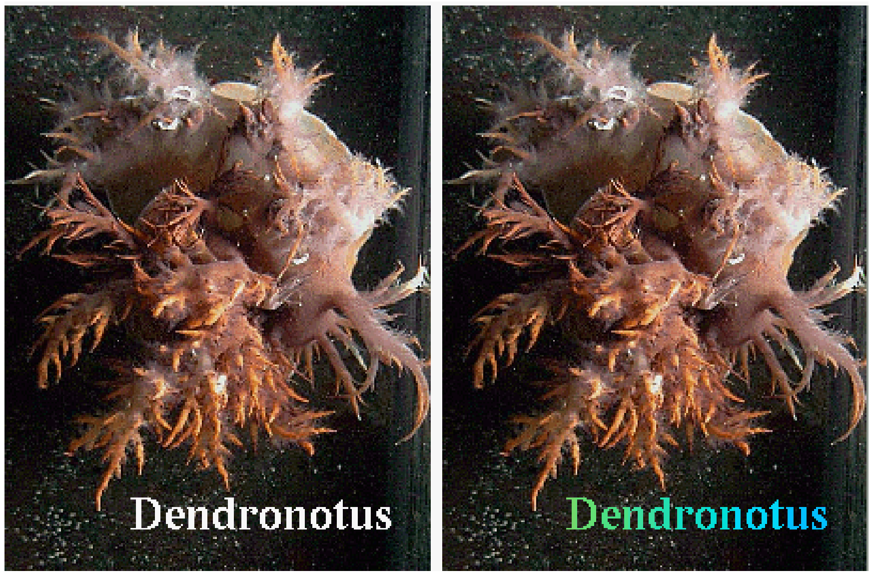
Example 2. Removing a gradient background from an image (flattening the background).

If an image is too light on one side and too dark on the other, one way to fix it is by adding or subtracting an artificial gradient. This can be done automatically, by selecting "Filter...Background Flatten".

1. Move all images off to one side to create a blank area on the screen.
2. Select "Color...Set colors" and change the background color to black (0) and the foreground color to white (e.g.,255).
3. Draw a box as in Example 1, slightly larger than the image that has the uneven background.
4. Set the Grad Start and Grad End Coordinates to match the position of the box.
5. Set the Start and End Gradient Colors to values which, when added to the corresponding parts of the image, will equalize the intensity. For instance, if the image colors are mostly 10-20 on the left and 90-100 on the right, set the Start and End Gradient Colors to 80 and 0, respectively. The image should be dark enough so that adding these colors will saturate the image. If the image is too light already, you should set the Start and End Gradient Colors to 0 and 80, respectively, and use "Subtract" instead of "Add" below.
6. Fill the box with the gradient.
7. Move your image back onto the screen (don't cover the gradient).
8. Select "Configure...Set pixel mode" and click on "Add".
9. Select "Image...Copy" and click-and-drag to copy a region from the image. Paste it on top of the gradient. The image will be "added" to the gradient.
10. Select "Configure...Set pixel mode" and set the pixel mode back to "Overwrite".
11. Select "File...Create Image" and select the area you just pasted. This will put the modified area into an image buffer.
12. Select "Image...Erase Background" to clean up the desktop.

Example 2: Changing the color of a text label.

In the image below, after the label "Dendronotus" was pasted in the foreground color (white) to the image, it was decided to change it to a gradient. This was done as follows:



1. Add a label to the image. Adding a label automatically writes to the alpha channel as well as the image.
2. Select a rectangular area around the label using the mouse.
3. Double-click on the white text. This causes the entire label to be selected.
4. Select Draw..Add/Remove Gradient and set Method to "Subtraction" (since color value cannot be added to white).
5. Set the RGB values to be subtracted for each corner.
6. Alternatively, press Alt-V to make the label black, and set the Method to "Addition". In this case, the Values of Gradient will be the final color values.
7. Click 'Accept'. Only the label, and not the image, was changed.
8. Before repeating with a different label, be sure to select "Draw... Clear Alpha Channel" first. Otherwise, if the new label is at the same location as the old one, the changes may be applied to both the new and old labels.

Note that alpha channels are not saved when the image is written to disk as an image file. If the label was added prior to starting `tnimage`, the program will attempt to identify the label in one of two ways:

If no item is selected it will look for a single continuous object similar in color to the point where you clicked.

If a rectangular area has been selected it will assume every point inside the rectangular area is part of the label if and only if it is exactly the same color as the point where you clicked.

9.9 Spray

Creates a spray-paint effect. Click the left mouse button at the center of the region to be sprayed. The area covered is determined by the “spray factor” in the “configuration” menu. **NOTE: Be sure to click the main “Cancel” button at left when finished to return to normal mode.**

Fine spray

Creates a solid spray painting effect. Click the left mouse button at the center of the region to be sprayed. The area covered is determined by the “spray factor” in the “configuration” menu.

Diffuse spray

Diffuse spray sets the pixels around the cursor to the foreground (drawing) color, but in a diffuse pattern.

Math spray

Math spray allows you to enter any mathematical formula to be applied to all the pixels in a fixed area around the cursor. This can be used to interactively copy small parts of another image or a different part of the image. For example, you could enter the formula

`i=image[1][0][x+100][y+100];` .

This would copy pixels from image no. 1, frame 0, 100 pixels to the right and below the mouse cursor, for a pattern retouching or texture mapping effect. The image number can be the same image or a different one (see *image algebra*).

If the image is color, the red, green, and blue components can have independent equations.

If the image is wavelet-transformed, the wavelet components can be sprayed with equations such as

`w*=10;` .

Note: Because rebuilding the screen display is a relatively slow process, screen rebuilding is not done if the image depth differs from the screen depth until the mouse is released. This can cause colors to temporarily appear incorrect.

Filter spray

Filter spray applies the selected convolution filter to the pixels in a fixed area around the cursor. This can be used to interactively remove noise or blur small parts of the image. The spray area must be larger than the filter kernel size.

‘Amount of Filtering’ should be set to some small value (e.g. 1-2%) to avoid sharp discontinuities around the edge of the spray-filtered region.

Erase Spray

Restores area under the mouse cursor to its original state. This has the same effect as clicking the “Erase” button (see Sec. 14.1).

Line Spray

Extends the pixels using a user-selected line as a template, creating a drawing.

1. Select ”Spray...Line pattern Spray”
2. Click ”Accept”
3. Define the linear region to use as the template by click-and-dragging. If you click the mouse without dragging, the previous template, if any, will be used.
4. Move to the desired area to change and click and drag to draw. Pixels from the template will be copied to the new location. The pixels will be rotated in the direction of drawing to create a wide line.

5. If you release the mouse button and continue drawing somewhere else, the previous angle is remembered. This can cause undesirable artifacts in the drawing. If you want the new drawings to start at a different angle, click the main "cancel" button, and click "Accept" to start again and click without dragging to continue using the previous template.
6. Click the main cancel button in the information window when finished.

9.10 Circles, ellipses, and rounded rectangles

Draws circles, ellipses, or rounded rectangles. Click on main “Cancel” button when finished. They can be filled with a solid color or up to two linear or radial gradients to create a 3-dimensional appearance. The following options are available:

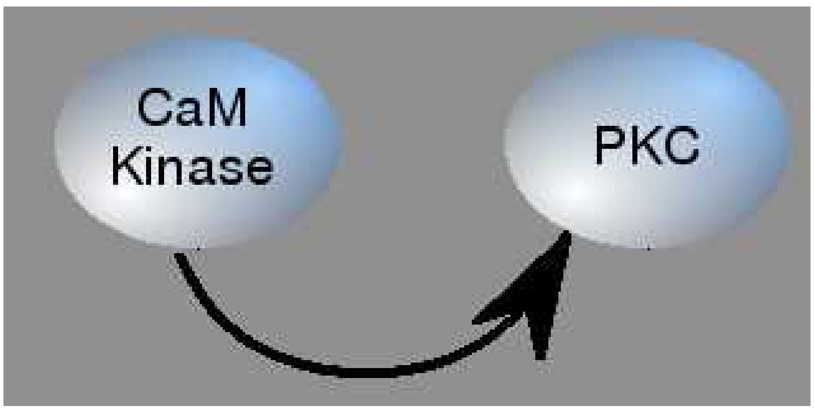
1. **Shape** Selects between fixed-size circles, variable-size circles or ellipses, ovals, or rounded rectangles.
2. **Diameter** For fixed-size circles, specifies the diameter. For rounded rectangles, specifies the diameter of the curve at each corner.
3. **Line width** Specifies the width of the line used to draw the outline of the object. The outline is drawn in the current foreground color.
4. **Filled** If checked, the object is filled with a solid color or a gradient depending on the setting of ”Gradient type”.
5. **Outline** If checked, draws the outline of the shape. This is only useful if ”Filled” is checked. If both ”Filled” and ”Outline” are unchecked, nothing will be drawn.
6. **Fill 1 type** Specifies whether the object is filled with a solid color, a radial gradient, or a linear gradient. See below for an example of a radial gradient.
7. **Fill 2 type** If set, specifies how to fill with the second set of gradient colors. This permits creation of shapes with two apparent illumination sources. See below for an example of a radial gradient.
8. **Grad. 1, 2 x angle** Specifies the horizontal angle of the “illumination” of the object if the ”Radial gradient” box is checked. The numbers range from 0 (= left) to 1 (right).
9. **Grad. 1, 2 y angle** Specifies the vertical angle of the “illumination” of the object if the ”Radial gradient” box is checked. The numbers range from 0 (= below) to 1 (= above).
10. **Outer color 1** If “Gradient type” is “solid color”, specifies the color. If “Gradient type” is “radial”, specifies the color for the “unilluminated” region of the object.
11. **Inner color 1** If “Gradient type” is “radial”, specifies the color for the “illuminated” region of the object.
12. **Outer color 2** If “Gradient type” is “radial”, specifies the color for the “unilluminated” region of the object for the second gradient.
13. **Inner color 2** If “Gradient type” is “radial”, specifies the color for the “illuminated” region of the object for the second gradient.

Although tnimage is not a drawing program, it can be used to create some types of drawings. The principle used by tnimage is different from that used by drawing programs such as Corel Draw. In tnimage, all graphical elements are treated solely as pixels on an image. This means that once something is drawn, it can only be removed by replacing the pixels with other pixels. The easiest way to create a drawing in tnimage is to hit Ctrl-B (or select Image..Backup) to backup the image, then add the graphic element. Hit Ctrl-U (or select Image..Restore) to undo the changes. To move an object, select a rectangle around the object and click and drag the corner of the rectangle to a new location. Objects such as text can also be selected by (1) double-clicking on the text area (to select a single letter), or (2) selecting a rectangle around a group of objects (such as a word) and double-clicking on the text area to select more than one letter. Be sure to click on the colored portion of the text and not the background color surrounding the text, or the program will select a “stencil” of the text instead.

Below is an example of a simple diagram created by drawing two ellipses and a Bezier curve using a double radial gradient. The gradient colors were:

	Red	Green	Blue
Outer color 1	0	0	0
Inner color 1	127	189	255
Outer color 2	0	0	0
Inner color 2	255	255	255

The labels were added using the Freetype font option.



9.11 Other Graphic Elements

9.11.1 Add border

Puts a border around the current image using the current foreground color.

9.11.2 Paint Region

Sets a rectangular region to the foreground color. After selecting 'paint region', move the mouse cursor to one corner and click and drag to the other corner. The selected area will be painted.

Click the "cancel" button when finished.

If a non-rectangular region has been selected, only this region will be painted (see sec. 3.5.3).

9.11.3 Box

Draws boxes by the click-and-drag method.

9.11.4 Curve

Draws a curve or line defined by user-selected control points.

9.11.5 Bezier curve

Draws a smooth curve. Click on the desired locations for the control points. At least 3 points are required to draw a curve. The position of each control point is indicated by a small box. The control points can be moved interactively by clicking on an existing box and dragging it to a new location. Clicking outside a box creates a new control point. When you are satisfied with the shape of the curve, pressing any key erases the small boxes and makes the Bezier curve permanent. You can have a maximum of 199 control points.

9.11.6 B-spline curve

Similar to Bezier curve, except has a maximum of 196 control points. B-spline curves differ from Bezier curves in that the curve tends to be closer to the control points and sharper in appearance.

9.11.7 Least-squares line

Fits a straight line between the control points using a least-squares algorithm.

9.11.8 Polygon

Draws a polygon through an arbitrary number of control points.

9.11.9 Trapezoid

Draws a trapezoid through 4 control points. The control points should be selected in a clockwise direction.

9.11.10 Snap trapezoid

Draws a trapezoid through 4 control points. The control points should be selected in a clockwise direction. The first and third line segments are forced parallel to each other (so it is really a parallelogram).

9.11.11 Fixed-width rectangle

Draws a rectangle with the specified width (in pixels) through two control points.

9.11.12 Open polygon

Similar to polygon but without automatically connecting the starting and ending points.

9.11.13 Sketch

In sketch mode, whenever the left mouse button is pressed, a continuous line is drawn on the screen. Sketch mode is also toggled on/off by the F2 key and can be deactivated by the main "Cancel" button.

9.11.14 Point to point polygon

Similar to polygon, but the control boxes are not drawn. Drawing is similar to the line drawing mode in `xfig`.

Section 10

About menu

10.1 About the program

Displays the version number, operating system, amount of free memory, the number of images currently loaded, and other information.

10.2 About the file

Displays a variety of technical information about an image file. This is useful in diagnosing problems with reading a possibly corrupted file or in getting parameters from an unknown file format such as that produced by certain specialized frame grabbers.

10.3 About the image

Displays a variety of information on the current image. This window is updated dynamically whenever a change is made to the image, or when a different image is selected.

10.4 Select image

Displays information on the size, position, title, and type of all images Selecting one desired image from the list with the arrow keys or mouse, and pressing *Enter* or clicking on *OK* , selects the image and raises its window. This window is updated dynamically whenever a change is made to the image, or when a new image is opened.

Section 11

Configure menu

11.1 Show colormap

Click the left mouse button to indicate where the new colormap strip should go. The colormap strip is redrawn in a window of its own, separate from the images. The colormap can be used to select the foreground and background colors (by clicking the left or right mouse buttons, respectively, on the desired color).

11.2 Show O.D. table

Draws a graph showing the correspondence between pixel values (0 to 255) and optical density, as calculated by the scanner. Typically, only TIFF files from scanners have an O.D. table (referred to in TIFF files as a 'gray response curve'. If the image does not provide one, `tnimage` will create a linear O.D. table. (**Note:** most Macintosh TIFF files do not have O.D. tables.)

The O.D. table (also known as a "gamma correction table") is only meaningful in monochrome-/indexed-color modes (8 bits/pixel). Each image can have a separate O.D. table. The table is a simple way of mapping pixel values to OD in a non-linear manner.

11.2.1 Changing the O.D. table

The table can be modified by clicking the "→change" button and dragging with the mouse, or a pre-established gamma table can be read from disk by clicking on the "Read file" button.

This file should consist of two columns of numbers in ASCII format, the first column being the pixel value and the second column being the optical density for that pixel. Pixel values not found in the file are left unchanged. For example:

```
  0       1.23
  1       2.34
  3       8.88
 17     99.0000
255      123.345
```

The above table would modify the pixel value →optical density mapping for pixels with intensities of 0,1,3,17, and 255.

Notes:

1. Numbers in the first column must be between 0 and 255, otherwise the entry will be ignored.
2. Gamma tables are only useful for 8-bit grayscale images. For grayscale images higher than 8 bits/pixel, it is impractical to use a table. Non-linear conversions can still be performed using the "image algebra" function.
3. `tnimage` checks the gamma table to ensure it is a monotonically increasing or decreasing function of pixel value. If not, an error message is displayed.
4. The new table can be multiplied, smoothed, captured to an image, etc. using the other buttons.

11.3 Pixel interact mode

Selects how 2 pixels interact with each other when you copy and paste parts of an image, read an image from disk, or add graphical elements.

Overwrite - Default mode. The new pixel erases the old pixel. This is by far the fastest mode.

Maximum - The larger of the 2 values is used.

Minimum - The smaller of the 2 values is used.

Add - The 2 values are added (up to the maximum value).

Subtract - The new value is subtracted from the old value (with a minimum result of 0).

XOR - The 2 values are XOR'd with each other.

Average - The 2 values are averaged.

Superimpose - The new pixel replaces the old pixel unless the new pixel's value is 0, in which case the old pixel is unaffected (allows superimposing images with parts "masked out").

Example 1. Finding subtle differences between two images (also creates a silhouette effect).

1. Load an image into `tnimage`.
2. Select "Pixel interaction mode...Subtract".
3. Select "Open image" and change "x position" and "y position" to 2.
4. Click on "OK".
5. The images have now been subtracted. (It may be necessary to make it lighter or increase the contrast.)

Example 2. Creating contour maps of an image.

1. Load an image into `tnimage`.
2. Select "Pixel interaction mode...XOR".
3. Select 'copy', and click-and-drag to select a portion of the image.
4. Release the mouse button.
5. Move the mouse to a location 1 pixel to the right and 1 pixel below the original position and click the mouse button to put the copy in place.
6. You now have a crude contour plot of your image. The gradient sensitivity of the contour plot can be decreased by using an offset of 2 or more pixels in step (5). You can also "threshold" the plot by subtracting a value ("make darker") and then adding it back("make lighter").

Example 3. Reducing chunkiness in image.

1. Read an image into `tnimage`.
2. Select "Pixel interaction mode...Average".
3. Select "Open image" and change "x position" and "y position" to 2.
4. Click on "OK".
5. You now have an image that is smoother without becoming blurred.

Don't forget to change "Pixel interact mode" back to "Overwrite" afterwards.

NOTE: Don't forget that, when loading an image, unless the interact mode is "overwrite", the image will interact not only with other images but also with the background in areas where no image is present. Thus it may be helpful to set the background color to black first, to avoid unexpected results.

11.4 Configure...

Configures the following options:

Color reduction method

See "Color reduction" (Sec. 5.1.1).

Quantization - (Default) Calculates the optimal set of 256 colors to match the image as closely as possible.

Fit current colormap - Calculates the closest match to the currently-selected colormap. This method can give a smoother result than quantization or it can give garbage, depending on how closely the colors in the colormap match the image.

If there are several images that need to be converted to the same colormap, one procedure is to convert the first one using quantization then convert the rest by fitting to the current colormap.

This method is used automatically for 3D images.

Update undo buffer

Automatically: Backs up the image after every operation that transforms the image.

Manually: The image is backed up when it is first created. Thereafter, selecting "Image..Backup" (or pressing Ctrl-B) will back it up.

"Image...Restore" will restore the image to its state when it was last backed up. If you make a mistake when filtering or adding text to the image, you can restore the most recent backup from the undo buffer by selecting "Image...Restore". See also Sec. 6.7.

Never: No backup buffer is created. This is useful in low-memory situations, or for extremely large images. Changes made to the image cannot be undone.

Colormap usage

The X Window system allows applications to setting colors in two ways: by allocating individual colors as needed, or by installing a colormap. When one program allocates a color, it cannot be changed by other programs. In many situations, if other programs or the window manager have used most of the color cells, the image quality can be severely degraded. This is common, for example, on Sun workstations. In this case it is recommended to select "install colormap". Because this can cause screen flashing if the mouse moves away from the `tnimage` window, the `tnimage` window should also be enlarged to full screen in this situation.

This option is set automatically if less than 32 free color cells are available.

Message boxes

None: no warning or information boxes are ever shown. This option is dangerous because you will not be notified before overwriting a file.

Minimal: only warning message boxes are shown. This may make repetitive operations more convenient, but you will not be prompted as to what you are supposed to do next.

Normal: all message boxes are shown.

Area Selection Mode

Single (freehand): Areas are selected by clicking and dragging the mouse cursor around the desired region.

Multiple (freehand): Same as above, except multiple discontinuous regions can be selected.

Polygon (Adjustable): Areas are selected by clicking at points around the region. These points, designated by small boxes, are automatically converted to a polygon. The points can be re-positioned by clicking on the small boxes and dragging them to a new location.

Point-to-Point: Same as above, except that the control points are not adjustable. This mode is similar to the line-drawing mode in `xfig`.

Luminosity factor

Used when converting an image from color or indexed-color to gray scale. The relative contributions to luminosity from the red, green, and blue components are determined by the "Luminosity factors". The default is:

$$pixelvalue = .299 * r + .587 * g + .114 * b$$

Significant digits

Changes the number of significant digits to display for floating point numbers in all dialog boxes.

Cursor movement rate

Controls (a) distance moved by the mouse cursor when the arrow keys are pressed, and (b) the distance which an image moves when you click one of the 4 arrows at the top of the screen.

Spray factor

Controls the area affected by "fine spray", "math spray" , etc (in "Draw...Spray" menu).

Active color planes

Selects which colors (red, green, or blue) are capable of being modified. For example, if "red" and "green" are un-checked, smoothing the image would only smooth the blue component, while the red and green components were unaffected. Using this method to sharpen one color, while leaving the other colors unaffected, or warping a single color, can create an unusual artistic effect.

This feature only works for color images greater than 8 bytes/pixel. To use color planes on an 8 bit/pixel image, select "Color...Change color depth" and convert the image to 24 bits/pixel.

Main Cursor

Sets shape of the normal cursor. If "full screen crosshairs" is selected, the Xlib cursor is turned off and full window vertical and horizontal lines are drawn in the foreground color.

Raise image on focus

If checked, clicking on an image will bring the image window to the foreground.

Separate windows

If checked, new images will be placed in separate, positionable windows.

Window border

If checked, new images will have a complete X-window border and decoration. This option is only effective if “Separate windows” is also checked. The exact appearance will depend on the window manager.

Show image title

If checked, the base file name of each image is printed in the upper left corner of the image. The label is only temporary and is not saved when the image is saved to disk.

Text spacing Adds extra space between characters (units=pixels).

Split frame cols.

The default number of columns to use when converting a multi-frame image into a panel.

Object threshold

See “Selecting Objects” (Sec. 3.5.2). Selects the RGB distance to use in determining the edge of an object.

Zoom factor

The amount by which the image is enlarged or reduced during zooming. Click the “zoom” button to enter zoom mode. Then click the left mouse button to zoom in or the right mouse button to zoom out.

Capture root win.

If checked, the floating magnifier will enlarge pixels under the mouse cursor from anywhere on the screen. If unchecked, the floating magnifier is only updated when the mouse cursor is over the `tnimage` main window.

Section 12

Mathematical pixel operations (Image Algebra)

Although it is possible to add or multiply pixel values by any factor by using the "brightness" or "contrast" menu items, `tnimage` also has a more powerful tool – mathematical pixel operations.

Using the "Macro/image math" menu, mathematical formulas can be entered to transform each pixel value in one pass. These formulas could be any legal C-style mathematical equation consisting of constants, variables, and operators from the following list:

Constants - any real number (floating point or integer)

Any of several pre-defined constants (see below)

Variables - one of 6 pre-defined variables:

i = intensity (total pixel value)

v = pixel density (pixel value, corrected for image depth and scaled to 0 to 1, but not corrected for calibration)

d = calibrated pixel density (see Image Calibration)*

r = red component of the pixel

g = green component of the pixel

b = blue component of the pixel

x = x coordinate of the pixel (0,0=upper left of image)

y = y coordinate of the pixel

re = FFT real coefficient

im = FFT imaginary coefficient

w = Wavelet coefficient

* variable is read only (any changes to this variable are ignored).

Operators -

`, , =, +=, -=, *=, /=, ==, !=, < , <=, >, >=,`

`+, - , * , / , ++, --, [ , ] , {, }, (, ), if, else, for, while`

Integer operators -

NOTE: The operands are truncated to integers before integer operators are applied.

`|=, &=, ^ =, ||, &&, | , ^ , & , ~ , !`

Single parameter functions

sin cos tan

sinh cosh tanh

asin acos atan

sqrt abs

log *(natural logarithm)*

ln *(natural logarithm)*

log10 *(base 10 logarithm)*

exp

pi

rand *(random number based on specified seed, scaled 0-1.0)*

The following 4 functions are only available on Linux:

asinh

acosh

atanh

cbrt *(cube root)*

Two-parameter functions

max

min

pow *(pow(x,y) = x raised to the power y)*

Image names, if used in quotes, will evaluate to the image number and can be used anywhere an image number is needed. See Sec. 12.2.

Miscellaneous functions Any macro function (Sec. 13) can be included along with image math.

User-defined variables

Variable names must start with a letter. It is not necessary to declare variables before using them. If they are referenced without being initialized, the value is 0.

For example, if you entered the following equations:

increase=1.2345;

r=r+3;

g=(r/4)+increase;

b=g–pow(b,2);

`tnimage` would create a variable named `increase` and set it to 1.2345. Each pixel would have its red component increased by 3, its green component would be set equal to red/4 + 1.2345, and blue would be set equal to green minus (blue squared). Since pixels can only have integral values of red, green, and blue, the numbers are converted back to integers before being stored in the pixel.

The formula

i = abs(w);

Copies the absolute value of the wavelet coefficients into the image. **Warning:** Wavelet-transformed images may have a different size from the original image.

These image algebra formulas are automatically iterated over every pixel in the current image or selected region. You can select a region with the 'select' command, or iterate over some other variable by explicitly creating a 'for' or 'while' loop. See 'Macro Programming Guide' for details.

All variables are double-precision floating point numbers and are remembered from one call to the next. Thus, it is possible to declare and initialize variables on one pass and change them on the next. For example, to initialize the variable "increment", select a small region and execute the equation

`increment = 0;` .

Then one can select the image and execute a set of equations such as:

```
if(r>80 && g>80 && b>80){ r=increment; g=66; } else { g=77; b=88; }
increment+=0.01;
```

This would create a gradient effect in the image because `increment` would be increased by 0.01 after every pixel was processed.

After editing the equations, there are several options:

- Pressing *Enter* inserts a new line in the editor.
- Pressing *Shift-Enter* evaluates a single line.
- Clicking *Execute* evaluates all the equations sequentially, **starting at the current cursor position**. Numbers are not checked or put back into the image until all the equations have been evaluated for that pixel. Thus, it is possible for intermediate values for intensity in an 8-bit image, for example, to exceed 255 (normally the maximum intensity for an 8 bit image). Intermediate values are stored as double precision floating point numbers.
- Clicking *Cancel* stores the changes and exits the editor.
- Clicking *Load* reads a text file into the editor.
- Clicking *Save* saves the editor contents into a text file.

Example: To perform "gamma correction" to an image, enter the following equations:

r = pow(r,1.8);

g = pow(g,1.8);

b = pow(b,1.8); ,

where 1.8 is the gamma correction factor. For monochrome, a higher value (e.g., 2.35) is often used.

Example: Conditional expressions:

if(r==32) g=r+x+y;

Sets green equal to red plus the x and y coordinates (measured in pixels from the upper left corner of the image), if red is equal to 32.

Example: Multiple nested if..then..else statements:

```
if(r>=23){ g=34; b=45; }else{ g=99; if(r>56) b=56; else b=76;}

if(r>=23)
```

```
{   g=34;
    b=45;
}else
{   g=99;
    if(r>56)
        b=56;
    else
        b=76;
}
```

These two equations are equivalent. White space and line breaks are ignored. Syntax is similar to “C” programming. If two or more statements are on the same line, C-style semicolons are required between statements. Portions of the statement could also be executed as a single line, by pressing Shift-Enter instead of clicking OK. If a single line is executed, all lines before and after it are ignored, and the line must have valid syntax when taken by itself out of context. **Note:** The line will be iterated over all pixels in the image or selected area, even if the statements before and after it define a loop over some other variable.

Example: To eliminate all pixels below 100 and above 200:

i = min(200,max(i,100));

Using data from other images

Mathematical operations can also be performed using data from other images, other frames, or other regions in the same image using the [][][][] operator:

Unix version:

image[image_number][frame][x][y]

red[image_number][frame][x][y]

green[image_number][frame][x][y]

blue[image_number][frame][x][y]

real[image_number][x][y]

imag[image_number][x][y]

wave[image_number][x][y]

density[image_number][frame][x][y]

DOS version:

image[image_number][x][y]

red[image_number][x][y]

green[image_number][x][y]

blue[image_number][x][y]

where:

image_number is the number assigned to the source image by `tnimage` (This number may change unexpectedly when images are unloaded). You can also use variables or functions for each of these indices, for example:

```
re = imag["myimage.gif"][sqrt(a)][cbrt(b+c)];
```

See Sec. 12.2. for more details.

As in C, brackets denote array notation. Thus, the x, y, and frame coordinates in a bracket are in pixel units relative to the upper left corner of the image, which will be different from the screen coordinates if the image is not located at the upper left corner of the main window.

frame is the frame number of the source image (3D images only). For non-3D images, the frame number must be 0. It may be convenient to create an image with 2 or more frames to hold intermediate calculations. By default, no intermediate values are stored; therefore, an equation such as

```
if(x>0&&y>0) i = image[0][0][x-1][y-1];
```

will overwrite its own source data. (This would have given an 'out of bounds' error without the 'if' statement.)

Note that, if all operations are on the same pixel and same image, then i,r,g,b,im,re, and w are the same as their longer versions (image, red, green, blue, imag, real, and wave). For example,

i = i+10;

would be the same as

i = image[1][0][x][y];

However, the long versions are technically functions and so they may not appear on the left side of an equation. Thus, the following is illegal:

image[1][0][x][y] = 6;   // Illegal

y and *x* are the coordinates of the source pixel. The coordinates are relative to the upper left corner of the source image.

Note that wavelet coefficients and the real and imaginary components of FFTs have only 3 indices, since there is only a single frame for FFT'd images. Referring to a real or imaginary component of an image that has not been transformed, or referring to coordinates or frames not present in a given image, will produce an error.

'density[][][][]' is similar to 'image[][][][]' except that the value is corrected for bits/pixel and calibration (if the image has been calibrated), and the value is returned as a number between 0 (black) and 1.0 (white). See also 'rdensity' and 'cdensity'.

'd' gives the same result as 'density[][][][]', but gives the calibrated density for the current image and current x and y value.

NOTE: The value returned by 'density' will be different depending on the Pixel Density Calibration setting in the Densitometry dialog. This can also be changed in a macro by changing the COMPENSATE constant[2]. If 'z units' was selected, the image must be calibrated first. See the sections on Image Calibration and Densitometry for details.

Before using the functions *real* and *imag* to copy frequency components from one FFT'd image to another, it may be desirable to increase the size of the destination window to match the size of the source image. This can be done by the command

`resizewindow(` *ino, xsize, ysize);* .

For example:

```
resizewindow(1, 128, 128);
```

will increase the window for image no. 1 to 128x128, leaving the original image in the upper left corner.

All numbering of frames, images, and x and y coordinates starts with 0. An invalid coordinate, frame, or image causes an error and returns a value of 0.0. The formula is iterated over every (x,y) in the selected destination image or region.

If this form is used, all 4 bracketed parameters must be specified even if only one frame exists.

[2]So yes, in fact that means that `COMPENSATE` is not really a constant but a variable.

Since 3D images are not supported in the DOS version, only the image number and x and y coordinates need to be specified.

Example: Subtracting 2 images with a small offset and increase the brightness of the result by a factor of 2.5.

1. Click on destination image. (Assume source image is #1).
2. Select "image math".
3. i=2.5*(i-image[1][0][x-2][y-2]);

Example: Making all the pixel values in 8-bit image #0 an even number:

```
i=image[0][0][x][y]&254;
```

Example: Increasing the green in pixels whose red and blue are both greater than 128:

```
if(r>128 && b>128) g*=2;
```

Example: Multiplying the real frequencies of Fourier-transformed image no. 7 by 2:

```
re=2*real[7][x][y];
```

Exercises:

Practice running the sample macros supplied with `tnimage`.

Notes:

1. Conditional expressions must consist of a single line unless the predicate statements are enclosed between braces. For example, the lines:
   ```
   if(r==g)
       b+=20;
       g=0;
   ```
   will increase b only if r is equal to g, but will set all g's to 0.
2. Multiple expressions can appear on the same line if separated by semicolons (;).
3. Equations which do not make sense to apply to the image are ignored. For example, it is not possible to alter the RGB components of a grayscale image separately. While is is possible to directly alter the intensity components of indexed-color images, this is not recommended.
4. 8-bit indexed-color images are temporarily converted to 24 bits before applying the formulas. The RGB values for 8 bit images therefore range from 0 to 255 instead of their normal range of 0 to 63. When the image is converted back to 8 bits/pixel, its colormap is regenerated to make the actual colors correct. This in turn causes the actual pixel values for each color to change in an unpredictable manner. Moreover, although the colors are correct, the ultimate RGB values will be lower by a factor of 4 than the RGB values specified in the equation.
5. It is strongly recommended that 8-bit indexed-color images be converted to 24 bits before proceeding. This provides greater accuracy and prevents problems from equations like `i=image[1][0][x][y];` . Since the image being operated on is temporarily at 24 bits/pixel, copying parts of another 8-bit image will result in very dark colors, unless the source image is also 24 bits/pixel. This is not done automatically for performance reasons.
6. Syntax and operator precedence is similar to C.

7. The calculations may be interrupted at any time by pressing the Escape key (Keyboard focus must be on the editor or image window for this to work).
8. `i` should be used with grayscale images, and `r, g,` and `b` should be used with color images.
9. "i" has precedence over r,g, and b. That is, if the equation `i=r=g=b=2444444;` is entered, the selected region will be set to 2444444 (a light blue), not white (r,g,b = 255,255,255, where 255 is the highest permitted red, green, or blue value).
10. The predefined variables i,r,g,b,x,y,temp, and user-defined variables may appear on either side of an equation, but functions such as sin(), max(), and image[][][][] may only appear on the right side.
11. The predefined variables i,r,g,b,w,x,y, temp, etc., may be redefined or accessed within a formula, e.g., `x=23;` ; however, their values will be automatically reset for the next pixel. Other variables retain their previous values as long as `tnimage` is running.
12. All calculations are carried out with double-precision floating point numbers. When the result is placed back into a pixel, it is converted to the closest integer. Thus, intermediate calculations such as multiplying by π give the correct result, but you should not expect to see floating point numbers in the image.
13. The 'math' command is no longer applicable, and has been removed since macro statements and algebra statements can now be freely intermixed.

12.1 Predefined Variables

The following variables are set for each pixel and may be changed in equations:

1. `r` red value of pixel
2. `g` green value of pixel
3. `b` blue value of pixel
4. `i` intensity value of pixel (integer)
5. `x` x coordinate
6. `y` y coordinate
7. `v` density value of pixel (0..1)
8. `re` FFT real (if image has been Fourier transformed)
9. `im` FFT imaginary (if image has been Fourier transformed)
10. `w` wavelet coefficient (if image has been Wavelet transformed)

The following variables are set for each pixel, but any changes are ignored, because their value is calculated from equations or data points entered by the user.

1. `d` calibrated value of pixel (based on user calibration)

The following constants are set once, before executing the math command. They can be changed if desired, but in some cases this would not make sense.

1. `IMAGES` The total no. of images present
2. `XRES` No. of horizontal pixels in the display
3. `YRES` No. of vertical pixels in the display
4. `ULX` Upper left x coordinate of selected area
5. `ULY` Upper left y coordinate of selected area
6. `LRX` Lower right x coordinate of selected area
7. `LRY` Lower right y coordinate of selected area
8. `CI` Current image number
9. `BPP` Bits/pixel of current image
10. `FRAMES` Total frames of current image

11. XSIZE Width in pixels of current image
12. YSIZE Height in pixels of current image
13. XPOS Position of current image (0,0=upper left)
14. YPOS Position of current image (0,0=upper left)
15. CF Currently-visible frame no. of current image
16. CALLOG0 Calibration of 1st (x) dimension, 0 = linear, 1 = logarithmic, 2 = polynomial, 3 = distance from 0,0.
17. CALLOG1 Calibration of 2nd (y) dimension, 0 = linear, 1 = logarithmic, 2 = polynomial, 3 = distance from 0,0.
18. CALDIMS No. of dimensions that are calibrated (0-2)
19. FFTSTATE Current Fourier-transform state of image (-1, 0, or 1)
20. FMIN Lowest FFT value or high-resolution wavelet coeff
21. FMAX Highest FFT value or high-resolution wavelet coeff
22. FFAC Conversion factor for FFT value or high-resolution wavelet coeff to pixel value
23. WAVELETSTATE Current Wavelet-transformed state of image (-1, 0, or 1)
24. WMIN Lowest low-resolution wavelet coefficient
25. WMAX Highest low-resolution wavelet coefficient
26. WFAC Conversion factor for low-resolution wavelet coefficient value to pixel value
27. Q00 to Q09 Image calibration coefficients for first dimension
28. Q10 to Q19 Image calibration coefficients for 2nd dimension dimension
29. GRAINS The total no. of grains found in the most recent instance of grain counting.
30. SPOTS The total no. of spots found in the most recent instance of grain counting (same as GRAINS).
31. PIXELS The total no. of pixels in the current image.
32. COMPENSATE The current setting of Pixel Density Calib. in the Spot or Strip Densitometry dialog. This determines the calibration method used when density values are accessed.
33. INVERT The current setting of Maximum Signal in the Spot or Strip Densitometry dialog. This determines whether the maximum signal is considered to be black or white when density values are accessed.

Example:

```
i=image[1][0][XSIZE-x-1][YSIZE-y-1];
```

12.2 Using image names as variables

In most situations, an image can be referred to either by its filename or by its number. The only exception is in image algebra formulas, where the filename must be enclosed in double quotes to avoid possible confusion with any variable names. In macro commands, this is not necessary, but quotes are allowed. The advantage of using names is that you need not worry about an image number changing when some other image is deleted.

For example, if the image "myimage" is image no. 2, all the following are valid:

```
convert(myimage, 24);              (changes image to 24 bits/pixel)
convert(2,24);                     (changes image to 24 bits/pixel)
switchto(myimage);                 (select "myimage")
switchto("myimage");               (select "myimage")
switchto(2);                       (select "myimage")
r = green[2][0][x][y];             (sets green equal to red)
r = green["myimage"][0][x][y];     (sets green equal to red)
fft(2,1);                          (forward Fourier transform of myimage)
fft(myimage, 1);                   (forward Fourier transform of myimage)
```

```
im = real["myimage"][x][y];       (sets imaginary equal to real component)
im = real[2][x][y];               (same)
evaluate(imag[2][12][12]);        (display imaginary component at 12,12)
evaluate(imag["myimage"][12][12]);(display imaginary component at 12,12)
```

If more than one image has the same name, all commands will be applied to the image with the lowest image number.

12.3 Evaluating expressions

The expression evaluator can be accessed directly with the `evaluate` command. This has several uses:

1. Checking the current value of variables, for example:
   `evaluate(abc);`or
   `evaluate(aaa==12);`
2. Performing calculations and checking syntax: `evaluate(sin(erf(2.34)));`
3. Setting variables: `evaluate(a=4);` (*set()* is the preferred way of setting variables.)

The `evaluate` command prints the answer in a small information box.

The command `whatis` is synonymous with `evaluate`. Another use of `whatis` is in conjunction with `print`. Normally, `print` would iterate over all the pixels, creating a long printout. However, if you were to enter

```
whatis(print("The pixel is ", image[0][2][200][200]));
```

the value of a single pixel would be printed.

`Evaluate` and `whatis` should not be used in loops.

Comments and multiline equations

Any line that starts with the # character is a comment and is ignored. Similarly, blank lines and carriage returns in equations are ignored. Thus you could have:

```
i = image[0][0][x][y] +
    image[2][0][x+4][y+4];
```

However, comments cannot appear in the middle of an expression.

Relationship between image math and macros

Although pixel math equations and macros use a 'C'-like syntax, the main difference is that, by default, all math commands iterate over all the pixels in the currently-selected image or selected area. This makes it consistent with the fact that macro commands such as 'filter' and 'fft' also work on entire images, even though they are only executed once. Some other commands, such as `fopen` and `fclose` are also only executed once. Thus if you have:

```
fopen("test");
write("test","the answers are i=",i," x=",x," y=",y,"\n");
fclose("test");
```

the write statement is iterated over all the pixels, while the fopen and fclose are only executed once.

A number of sample macros (`*.macro`) are included with `tnimage`.

Mixing image math and macros

The fact that `tnimage` commands and image math can appear in the same macro is a tremendous convenience in writing macros. For example, here is a sample macro:

```
# 1.macro - Simple macro
open(gray);
if(x>50) i+=27;
if(y>x) i+=66;
```

This macro opens the image file `gray`, and applies two equations to it. The variable “i” is used instead of r,g, and b because `gray` happens to be a grayscale image. However, `tnimage` has to check each statement to determine whether it is a macro or a math equation. Slightly greater speed can be obtained by enclosing the math equations in a “loop”:

```
# 1.macro - Simple macro
open(gray);
loop
{
   if(x>0) i+=27;
   if(y>x) i+=66;
}
```

In this macro, the “loop” keyword tells `tnimage` that the statements enclosed in braces are mathematical equations and not macro statements, so no checking is done against macro names. This has several advantages:

1. It is slightly faster.
2. Sometimes it is confusing as to whether a statement is looped or not. The ‘loop’ notation makes it clearer that these equations will be looped over all the pixels.
3. Variable names that are the same as macro commands can be used.

12.4 Input and output

Printing to stdout

You can print a value to the standard output with the command:

`print(`*stuff, more stuff, numbers, etc*`);`

For example:

```
print("The answers were: ", x, " ", y+z," yep, those were the answers\n");
```

The syntax is similar to C’s printf() statement, including the usual ‘backslash’ characters (tab, newline, etc), except that no formatting characters (e.g., %d, %g, etc.) are needed, and all strings (including single characters) must be enclosed in double quotes. There can be any number of parameters in the print statement. The output is flushed after each print. The output can be directed to a file by starting tnimage with

```
tnimage > filename .
```

If your locale uses commas as decimal separators, it may be necessary to add a space between two numbers to avoid ambiguities like:

```
print("The 2 numbers were: ", 42,567, "\n");
```

Printing messages

A simple message can be displayed with the command

```
message("your message here");
```

This will open a small information box that will display your message. Note that it will *not* wait for the user to click OK before continuing.

File output

File syntax is slightly different from C. Files are always accessed by their filename, not by pointers. There can be any number of files open simultaneously. A file must be opened before executing a 'write' statement. The filename is the first parameter, and literal filenames should be in quotes. For example:

```
fopen("data");
write("data", "The answer was", wavelet[x][y] "\n");
fclose("data");
```

As with `print`, there can be any number of parameters in the write statement. The output is flushed after each write and the file will be closed when `tnimage` exits. Even so, it is good practice to close files. A string variable can also be used as the filename (see below).

Input

The `input` command gets a number from the user and returns it to some numeric variable:

```
a = input("prompt string");
```

Similarly, `getstring` gets a string (such as filename or a function name) from the user and returns it to some string variable:

```
a = getstring("prompt string");
```

For both cases, the value can be examined with `whatis()`:

```
whatis(a);
```

12.5 Loops and control structures

There are 5 types of control loops in `tnimage`: `for` loops, `while` loops, implicit loops, explicit loops, and the ever-popular `goto`. The syntax is simple and C-like:

for loops initialize the first argument at the beginning, execute as long as the second argument is true, and execute the 3rd argument at the end of each loop. The entire `for` loop is only executed once. The loop body must be surrounded by braces. Unlimited nested loops are permitted.

```
for(k=0; k<GRAINS; k++)
{
   print("grain was at ", spotx(k), spotx(y), "\n");
}
```

while loops execute indefinitely as long as the argument is true. The loop body must be surrounded by braces.

```
while(i<96)
{  x+=100;
   y+=100;
   i++;
   d = cdensity(x,y,34);
   print("Density of well no. ", i, " is ", d, "\n");
}
```

implicit loops Any math statement not in a for or while loop will automatically be executed for every pixel in the current image or currently-selected area. This makes the notation extremely compact:

```
i*=2;
```

multiplies all pixels in the currently-selected region by 2.

explicit loops As described earlier, enclosing a group of statements in the "loop" structure makes certain that they will be evaluated as image algebra statements. For example:

```
loop
{
   ... statements ...
}
```

Statements inside explicit loops must be enclosed in braces. Any macro commands inside an explicit loop will cause an error.

goto Jumps to specified line number. Statement labels are not supported yet.

```
fft(1,2);
k++;
if(k<5) goto(1);
```

Section 13

Summary of macro commands

`Tnimage` can execute a series of predefined commands. This is particularly useful when a large number of images need to be processed in an identical manner. Technically, the macros in `tnimage` are not macros but small programs. The macro editor can also edit small text files (up to 255 lines), making it useful as a clipboard for making notes.

Macro commands consist of the text of one of the menu items, followed by a series of parameters which are usually numeric, in parentheses, separated by commas. For example, "Open Image..." becomes "openimage". If a sufficient number of arguments are given, the dialog box is bypassed and the command is executed. Image algebra (Sec. 12) commands can be freely interspersed with macro commands. Some of the more recently-added menu items may not be macro-ready, and using them will simply call up the dialog box.

The numbering of command parameters corresponds to the order in which the item would appear in the menu or dialog box. It is not necessary to give all the parameters. If a parameter is not given, it will be unchanged from the last time the command was executed.

If `tnimage` should crash due to an error in a macro, it is a bug. Please report it.

Here is a list of macro commands:

`3d`

`3dplot`

`aboutthefile`

`abouttheimage`

`abouttheprogram`

`abs`

`acos`

`acosh`

`asin`

`atan`

`acquire`

`addborder`

`arrow`

`attributes`

`automaticundo`

`backgroundcolor`

`backgroundvalue`

`backup`

`bcolor`

`beep`

`between`

`border`

`box`

`brightness`

`calibration`

`camera`

`cancel`

`cbrt`

`cdensity`

`changecolordepth`

`changecolormap`

`changecontrast`

`changeimagedepth`

`changepalette` *Redraw current image using specified palette number.*

`changepixelvalues`

`changesize`

`changetitle`

`chromakey` *Toggles chromakey on or off for current image*

`circle`

`clearalpha`

`clearalphachannel`

`close` *Same as unload*

`color`

`color->grayscale`

`colorgrayscale`

`colorbrightness`

`colormap`

`compositergb` *combine 3 specified grayscale images into a single RGB color image.*

`configure`

`contrast` *See below*

`convert`

`copy`

`cos`

`cosh`

`countgrains`

`createfileformat`

`createimage` *see new*

`createresizeimage` *see new*

`crop`

`curve`

`curvedensitometry`

`darken` *decreases brightness by a fixed amount*

`deleteregion`

`densitometry`

`density`

`down` *Moves image down by the distance specified in 'step'.*

`drawbox`

`drawcircle`

`drawfilledbox`

`drawfilledrectangle`

`drawlabel`

`drawline`

`drawrectangle`

`duplicate`

`else`

`erasebackground`

`erasefft`

`eraseimage`

`erf`

`evaluate` *Evaluate an expression (sec 12)*

`executeplugin` *executes specified plugin*

`exit`

`exp`

`fclose`

`fcolor`

`fft`

`fftdisplay`

`findspots`

`fileformat`

`fill`

`fillregion`

`filter` *filter current image or selected region (see below for details)*

`findspots`

`fliphoriz`

`fliphorizontally`

`flipvertically`

`font`

`fopen`

`for`

`foregroundcolor`

`foregroundvalue`

`frame` *selects frame of current multiframe image to display*

`freegraincountingarrays`

`getstring`

`goto`

`grains`

`grayscale→color`

`grayscalecolor`

`grayscalemap`

`help`

`histogram`

`histogramequalize`

`if`

`image` *selects an image*

`imagemath`

`input`

`intensity` *changes contrast of image*

`invertcolors`

`invertregion`

`label`

`left` *Moves image left by the distance specified in 'step'.*

`lighten` *increases brightness by a fixed amount*

`line` *Opens 'line' dialog*

`linestyle`

`loadfft`

`loadimage`

`localcontrast`

`log`

`log10`

`loop`

`macro`

`manuallyselectarea`

`mask`

`math`

`max`

`maximize`

`maximizecontrast`

`measure`

`message`

`messages`

`min`

`morphfilter` *Morphological filtering*

`move` *Moves a portion of an image or area*

`movie`

`moveto`

`new` *Creates new image*

`null` *Unpushes null buttons*

`open` *Same as load*

`paint`

`paintregion`

`palette`

`partition`

`paste` *Paste current contents of region being copied.*

`partition`

`patternrecognition`

`pixelinteractmode`

`pixels`

`plugin` *executes specified plugin*

`pow`

`printimage`

`print` *Print an image if no parameters are given; otherwise, prints data to stdout.*

`quit`

`rand`

`rdensity`

`readfft`

`registration`

`remapcolors`

`repair`

`repeat`

`reselectarea` *Reactivates previously-selected non-rectangular selected area*

`resize`

`resizewindow`

`resizeimage` *creates a new image of different size*

`restore`

`right` *Moves image right by the distance specified in 'step'.*

`root` *Puts image in root window*

`rootoff`*Puts image back in* `tnimage` window

`rootpermanent` *Sets X11 display background to the image*

`rotate`

`rotateimage`

`save`

`savefft`

`saveimage`

`savescandata`

`scan`

`scanner`

`scissors`

`select` *selects a region*

`selectarea` *start manually selecting an non-rectangular area*

`selectaregion`

`selectregion`

`selectimage` *switches focus to specified image*

`selectirregulararea`

`separatergb`

`setbcolor`

`setbackgroundcolor`

`set`

`setcolor` *Sets foreground drawing color*

`setfcolor`

`setforegroundcolor`

`setpixel`

`shift` *shift frame up/down or left/right*

`shiftframe`

`showalpha` *Temporarily display alpha channel on an image*

`showalphachannel` *Temporarily display alpha channel on an image*

`showcolormap`

`showselected` *Temporarily display selected region on an image*

`scan` *Same as "acquire"*

`showodtable`

`showpalette`

`sin`

`sinh`

`size`

`slew`

`sketch`

`smooth`

`spotdensitometry`

`spotdensity`

`spotsignal`

`spotsize`

`spotx`

`spoty`

`spray`

`spreadsheet`

`sqrt`

`step`

`stop`

`stripdensitometry`

`switchselected` *If a non-rectangular region has been selected, selects the complement of this region instead.*

`switchto` *switches focus to image specified by name or image number.*

`tan`

`tanh`

`testplugin`

`textdirection`

`title`

`tracecurve`

`transparency` *Sets amount of transparency (0-100) for current image*

`undo` *Same as restore*

`unloadimage, close, erase, unload` *remove image specified by name or image number from memory.*

`unloadall` *remove all images*

`unsetpixel`

`unzoom`

`up` *Moves image up by the distance specified in 'step'.*

`wallpaper` *Sets X11 display background to the image, tiling it if necessary to fill the screen*

`wantimagetitle Selects whether to superimpose image title on each image`

`warp`

`wavelets`

`whatis`

`while`

`windows` *Toggles separate window mode for new images*

`write` *Puts data into an open file*

`zoom`

Of course, interactive commands (such as `sketch` and `help`) are not too useful in a macro. But they can be quite useful as commands for user-defined buttons (Sec. 4.3).

Notes

1. Any underline characters or spaces are stripped from the command before it is executed.
2. Capitalization is not significant.
3. Some commands work on the currently-selected image or region. Other commands, usually those that only could be applied to the entire image (such as the image name), take the image name as the first argument.
4. The program is very liberal about syntax errors. This may change in the future.

13.1 Variable Types

There are only three data types: numbers, strings, and function variables. Variable names are changed from one type to the other if an expression of the appropriate type is assigned to it. Since the data type is determined automatically, there is no need to declare variables or distinguish them with special characters (such as '$' or '@').

Numeric variables

All numeric expressions are evaluated to double-precision floating point numbers. There is no need to specify a data type. For example, the following macro sets 'a' and 'b' to 123.456 and 42, respectively, checks their values, then adds 81.456 to each pixel between (100,100) and (200,200).

```
set(a=123.456);
set(b=sqrt(1764));
whatis(a);
whatis(b);
select(100,100,200,200);
i+=a-b;
```

If you want to change a variable outside of a loop, use `set()` to prevent automatic looping over the pixels:

```
set(a=123);
set(a+=exp(2.34));
set(a=a-sqrt(8350.5));
whatis(a);
```

For large macros, it may be necessary to increase the number of symbols in `calculator.h` (default=1000).

String variables

Strings are useful for file names, image titles, and for output formatting. To set a string variable, put the string in quotes:

```
set(bimage="gray.tif");
whatis(bimage);
open(bimage);
```

You cannot have macros like

```
set(bimage="gray.tif");
open(bimage);
set(a=123);
bimage += a;          /* Wrong */
```

The last line in the above macro would simply set all the pixels to 0 because a string always has a value of 0[1]. Use `i+=a` instead. **Caution:** Don't use predefined variables such as i,r,g,b,re,im, or w as string variables. This would cause your variable to be changed back to a numeric variable when a math command is executed.

Assignment of string variables (e.g., `a=bimage`) is not yet supported.

`Set` should not be used in loops.

Function variables

Another useful feature is the ability to interchange names of functions with string variables. For example, suppose you defined string `a` with the name of macro command `left`:

```
set(a="left");
a(100);
```

From then on, executing `a(100);` would execute the function "left", which moves the image to the left by the specified number of pixels. Now if `a` is redefined to be "right", executing `a(100);` would move the image to the right by 100 pixels. There are many situations where this makes the macro more compact.

In future versions, this will be used to permit the creation of new user-defined functions in macros.

[1]Actually, some string variables, such as names of open files, carry the value of their file pointer. But this doesn't count since you can't access it in a macro.

13.2 Macro Command Reference

The next section contains a partial listing of commands which require parameters. Commands not using parameters are not listed. Parameters must be specified in the indicated order, enclosed in parentheses, and separated by commas. All parameters must be on the same line as the command. Each command must be on a separate line.

Anything that appears in any of the menus may also be used as a command. Those commands that do not require parameters are not listed here. Some, such as 'automatic_undo' toggle a feature on or off. For these commands, using the same command a second time turns the feature back on.

Unless noted otherwise, all parameters can be omitted. This causes the current default setting to be used. However, parameters cannot be skipped. Thus, if you wish to set parameter #3, for example, you also must set parameters #1 and #2.

If a fatal error occurs, the macro is automatically terminated and an error message is displayed showing what line caused the error.

Commands with no parameters are entered as a single word, e.g. `erase` . Semicolons are optional if there is only one statement per line.

If images are in separate windows, the desired image should be selected before selecting coordinates. Otherwise, the coordinates may be calculated relative to some other image. These coordinates could be well off screen and result in unknowingly corrupting some other image. For example:

```
image(2);
select(100, 100, 200, 200);
```

This does not apply to functions like `image[][][][]`, `red[][][][]`, etc. in which the image number is specified.

abs

Returns absolute value of a number

Command	Parameter#	Refers to
acosh	1	value

Returns inverse hyperbolic cosine of a value.

Command	Parameter#	Refers to
asinh	1	value

Returns inverse hyperbolic sine of a value.

Command	Parameter#	Refers to
atanh	1	value

Returns inverse hyperbolic tangent of a value.

Command	Parameter#	Value	Refers to
automaticundo	1	0	never backup image, free backup buffer
		1	manually backup image when user hits Alt-B
		2	Backup image after every major command

Controls undo behavior

b Predefined variable for reading and setting blue component of a pixel in current image.

Ex.: b = r*0.8 + g*0.2 + b*0.1;

Command	Parameter#	Refers to
backgroundcolor	1	red
	2	green
	3	blue

Sets the background color to the color containing the specified red, green, and blue components.

Ex:

```
backgroundcolor(255,255,255);
```

backgroundvalue see bcolor

Command	Parameter#	Refers to
bcolor	1	pixel value

Sets the background color to the specified pixel value, bypassing the conversion from red, green, and blue.

Ex:

```
bcolor(255);
```

between between(a,b,c);
Returns 1 if a≥b and a≤c, otherwise returns 0.

blue Predefined variable for reading blue component of pixel of an image. Can only appear on right side of an equation.

Ex.: b = blue[1][0][x+5][y-4];

brightness see color

Command	Parameter#	Refers to
cbrt	1	value

Returns cube root of a value.

Command	Parameter#	Refers to
cdensity	1	x coordinate
	2	y coordinate
	3	diameter

Measures the density of a circular region centered at (x,y) with the specified diameter. The result will depend on whether the image pixel values have been calibrated, and on the settings of the variables COMPENSATE and INVERT, which are automatically set in the densitometry dialogs. These variables reflect whether the pixel calibration is to be used and whether the maximum signal is black or white (Sec.12). They can also be set in a macro. 'cdensity' returns the sum of the calibrated density value for each pixel inside the circle, corrected for image depth. Useful for creating macros to measure dot blots and 96-well plates. See also 'rdensity'

changebrightness see color

changecolordepth see convert

changecolormap see palette

changepalette see palette

Command	Parameter#	Refers to
changepixelvalues	1	intensity change
	2	red change
	3	green change
	4	blue change

Adds the specified intensity, red, green, and blue to the image or selected region.
Ex.: `changepixelvalues(-10,20,30,40);`
Adds 20, 30, and 40 to red, green, and blue and also subtracts 10 from intensity.

Command	Parameter#	Refers to
changesize	1	x factor
	2	y factor

Changes size of current image. Both x and y factors must be > or < 1.
Ex.:`changesize(0.5, 0.4);`
`changesize(2.5, 6.2);`

changetitle see title

close Unloads currently selected image.
Use 'select_image' command to select an image.)
Don't confuse with fclose.

Command	Parameter#	Refers to
color	1	change in red for color portion
	2	change in green for color portion
	3	change in blue for color portion
	4	change in hue for color portion
	5	change in saturation for color portion
	6	change in value for color portion
		If the entire area is monochrome, parameters 1-5 are ignored.

Changes RGB or HSV color in current image or selected region.

Examples: To increase the red in a color image by 10:
`color(10);` *or* `color(10,0,0);`
To increase the blue in a color image by 10: `color(0,0,10);`

To increase the brightness in a color or monochrome image by 23:
`color(0, 0, 0, 0, 0, 23);` or `lighten(23);`
The color, brightness, and color_brightness commands are synonymous.
`color(0,20,0,-5);` increases the red by 20 and decreases the blue by 5
Don't confuse with `setcolor()`.

colorbrightness	see color

Command	Parameter#	Refers to
compositergb	1	Image number for red (must be an 8-bit image)
	2	Image number for green (must be an 8-bit image)
	3	Image number for blue (must be an 8-bit image)

Creates a composite 24-bit color image from 3 8-bit images representing red, green, and blue intensities. All three images must be of identical size. The *color* or *contrast* commands may be used to fine-tune the color balance of the composite image.

Example

`compositergb(4, 5, 6);` creates a new color image no. 7
`compositergb("myred", "mygreen", "myblue");`

Command	Parameter#	Refers to
contrast	1	change in red contrast for color portion
	2	change in green contrast for color portion
	3	change in blue contrast for color portion
	4	multiplies hue by a factor (100 = no change). Note that hue is expressed in degrees, so a hue of 360 is the same as 0.
	5	changes saturation
	6	changes value (intensity). This parameter also affects grayscale images.

Changes contrast

Note:
To change contrast of a grayscale image, use "intensity".
If no parameters are given, the "contrast" command activates the graphical Color/Contrast control.

Examples:
To increase blue contrast in a color image by a factor of 1.5:
`contrast( 100, 100, 150);`

The following two commands have an identical effect:
```
contrast( 100, 100, 100, 100, 100, 234);
intensity( 234 );
```

Command	Parameter#	Refers to
convert	1	bits/pixel to convert image to (must be 8, 16,24, or 32).

Changes image depth

Command	Parameter#	Refers to
cos	1	radians

Returns cosine of a value.

Command	Parameter#	Refers to
cosh	1	value

Returns hyperbolic cosine of a value, i.e. (exp(x) + exp(-x)) / 2

countgrains - same as findspots

createimage - see new

createresizeimage - see new

d Predefined variable for reading calibrated value of a pixel. This is the pixel value scaled according to user-supplied calibration values.

Ex.: print("Elevation in topographical map is ",d);

Command	Parameter#	Refers to
crop	1	upper left x
	2	upper left y
	3	lower right x
	4	lower right y

Crops the current image, using specified boundary.

Ex.: crop(100,100,200,200);

Command	Parameter#	Refers to
darken	1	pixel value to subtract

Makes image darker by subtracting specified value from each pixel. Suitable for grayscale but not indexed-color images. See 'lighten'

Command	Parameter#	Refers to
densitometry	1	x1 1st x coordinate
	2	y1 1st y coordinate
	3	x2 2nd x coordinate
	4	y2 2nd y coordinate
	5	x3 3rd x coordinate
	6	y3 3rd y coordinate
	7	x4 4th x coordinate
	8	y2 4th y coordinate
	9	pixel compensation
	10	calibration factor

Performs strip densitometry on a rhomboidal region, starting from the line defined by (x1,y1) - (x2,y2). Subsequent samples are taken at 1-pixel intervals parallel to this line, in the direction toward (x3,y3) - (x4,y4). See Sec. 7.10. The first 8 parameters must be present or an error message is displayed. The x and y values must be provided in raw pixel coordinates, where (0,0) represents the upper left corner of the screen. Future versions of tnimage will allow x and y values to be specified in user-calibrated units.

If argument 9 is a 0 or omitted, no "pixel compensation" is performed. A 1 causes the pixel values to be translated through the image's OD table (gamma table), and a 2 converts the pixel values to their Z calibration value. This will change the measured signal.

If argument 10 is provided, the densitometry results are multiplied by this value.

Example: densitometry(20,40,160,160,340,700,350,800);

The parameters need not be numbers; as with all macros, any expression entered where a number is expected will be evaluated. Thus, for example, if the image has been calibrated, you can use the calibration coefficients that are displayed in the Image Calibration dialog to define a rhomboidal region in terms of calibrated units such as inches. These numbers are also available as the predefined variables Q00 to Q09 for the first dimension calibration coefficients, Q10 to Q19 for the 2nd dimension coefficients, and Q20 to Q29 for the 3rd dimension coefficients. For example, instead of a raw x screen coordinate of 20, you could enter the value in inches, multiplied and added to the appropriate Q00 to Q29 factors that would give a value of 20.

Since the formulas can be saved on disk, it is only necessary to figure out which coefficients to use once.

density Predefined variable for reading calibrated value of a pixel in an image. Can only appear on right side of an equation.

Ex.: d = density[1][0][x][y];

Command	Parameter#	Refers to
down	1	Image no.
	2	distance

Moves specified image down by the specified number of pixels. If no image is specified, the current image is moded. If no distance is specified, distance is the current shift value (change with the gray +/- keys)

drawbox - see drawrectangle

Command	Parameter#	Refers to
drawcircle	1	starting x
	2	starting y
	3	diameter in pixels
	4	color
	5	linewidth
	6	antialiasing (0 or 1)
	7	pixel drawing mode (SET, XOR, etc.)

Draws a circle at the specified screen coordinates. 0,0 is upper left of the display. Do not confuse with 'circle' which opens the interactive circle dialog. The color is specified in the pixel depth of the image.

Ex.: drawcircle(100,100,8);

drawfilledbox see drawfilledrectangle

Command	Parameter#	Refers to
drawfilledrectangle	1	starting x
	2	starting y
	3	ending x
	4	ending y
	5	color
	6	pixel interaction mode

Draws a rectangle between the specified screen coordinates and fills it with the current foreground drawing color. 0,0 is upper left of the display.

Ex.: `drawfilledrectangle(100,100,200,200);`

Command	Parameter#	Refers to
drawlabel	1	string to draw
	2	starting x
	3	starting y
	4	0=horizontal, 1=vertical
	5	0=no background, 1=draw background
	6	foreground color
	7	background color

Draws specified string at designated screen coordinates. Do not confuse with 'label' which opens the interactive label dialog. If param. 5 is 0, only the foreground pixels are set. If it is 1, the label is drawn in a filled rectangular box. 0,0 is upper left of the display.

Ex.: `drawlabel("Elevation 2100 ft",100,100,200,200);`

Command	Parameter#	Refers to
drawline	1	starting x
	2	starting y
	3	ending x
	4	ending y
	5	color
	6	pixel interaction mode

Draws a line at the specified screen coordinates. 0,0 is upper left of the display. Do not confuse with 'line' which opens the interactive line dialog. Drawing is done in the current drawing color.

Ex.: `drawline(100,100,200,200);`

Command	Parameter#	Refers to
drawrectangle	1	starting x
	2	starting y
	3	ending x
	4	ending y
	5	color
	6	pixel interaction mode

Draws a rectangle between the specified screen coordinates. 0,0 is upper left of the display.

Ex.: drawrectangle(100,100,200,200);

duplicate	Makes a copy of current image.

Makes an exact copy of image
Ex.: `duplicate("myimage");`

`duplicate(2);`

else

Use with 'if'

Ex.: if(i¿234) i=255; else i=0;

erase

- Begins spray erase mode. Click over areas to restore from image backup. Size of erased area is determined by spraysize in Config dialog. Click Cancel to stop.

exit	Quits the program *without* asking if you want to save your changes. Useful when running `tnimage` macros in batch mode (See "Batch-mode processing").

exp

Returns exponential of a value (e raised to power x)

evaluate	evaluates expression once, putting answer in message box.

Ex: `evaluate(sqrt(k));`

executeplugin

see plugin

Command	Parameter#	Refers to
fclose	1	filename

Closes the previously-opened file. Unlike C, does not use a file pointer. Do not confuse with 'close'.

Ex.: fclose("myfile.dat");

Command	Param.#	Refers to	Value
fft	1	Direction	1 Forward 2 Reverse 3 Convolute 2 images 4 Deconvolute 2 images 5 Change display only
	2	Display	1 Real 2 Imaginary 3 Power spectrum
	3	Image #2 (only required for deconvolution)	

Performs Fourier transform on image
Examples:
`fft(1);` (forward fft of current image)
`fft(4, 1, 8);` (deconvolute current image with image 8 and show real values)

Command	Param.#	Refers to	Value
fftdisplay	1	Display mode	0 Original image 1 Real 2 Imaginary 3 Power Spectrum

Controls whether to display original image, or real, imaginary, or power spectrum components of FFT'd image.

Examples: `fftdisplay(1);`

Command	Param#	Refers to	Value
file_format	1	Format	 -1 (automatically detect) 0 NONE 1 TIF 2 PCX 3 IMA 4 IMG (Frame grabber) 5 GIF (GIF87a) 6 IMM 7 GEM 8 IMDS 9 RAW 11 JPG 12 TARGA (Targa RLE) 16 BMP (Windows bitmap) ...etc. 23 ASCII 100 first custom format 101 2nd custom format ...etc. This will supercede the image's existing default file format for the duration of the macro.

Sets the default file format for loading and saving images.
Example: fileformat(5); = sets default format to GIF

These numbers correspond to the `const` definitions in the file `xmtnimage.h`. Check this file in case a new file format has been added.

Command	Param#	Refers to	Value	Meaning
filter	1	filter type	0	low pass
			1	high pass
			2	laplace
			3	background subtract
			4	background flatten
			5	noise (median)
			6	sharpen /
			7	sharpen |
			8	sharpen –
			9	sharpen \
			10	edge detect –
			11	edge detect |
			12	sobel
			13	remove low freq.
			14	rank leveling
			15	engrave
			16	multiplicative sobel
			17	spatial difference
			18	maximize local contrast
			19	user-defined
	2	kernel size:	1	3x3
			2	5x5
			3	9x9
			4	15x15
	3	amount of		
		filtering	1-100	
	4	range (median filter)		
	5	kernel multip.	any integer	
	6	max background	0	black
			1	white

Filters current image or selected area.

Examples:

`filter(5, 1, 3, 10, 1, 1);` Filters the currently-selected image or area using a 3x3 median filter, with a filtering level of 3, a range of ±10, kernel multiplier of 1, and white declared as the background.

`filter(12, 2);` Filters the currently-selected image or area using a 5x5 Sobel filter.

Command	Param#	Refers to	Value	Meaning
findspots	1	method		Which method to use
			1	1=quick segmentation
			2	2=neural networks
			3	3=differencing
	2	threshold	0-1	Threshold (0 to 1)
	3	size	0-20	Size (for differencing method only)
	4	match weight	0-1	Weight for match (0-1, used for neural networks only)
	5	mismatch weight	0-1	Weight for mismatch (0-1, used for neural networks only)
	6	labels	0	no labels
			1	print labels
	7	label color	0-∞	color for labels
	8	graph	0	no graph
			1	graph of size distribution
	9	signal distrib.	0	no graph
		graph	1	graph of signal distribution
	10	rectangles	0	draw bounding rectangle
			1	don't draw rectangle
	11	minimum size	0-any integer	Minimum size (for quick segmentation only)
	12	maximum size	0-any integer	Maximum size (for quick segmentation only)
	13	save data	0	don't save
			1	save in file
	14	filename		file to save results

Finds grains or spots in image. For simplicity, all 3 types of grain counting use the same parameters. This means not all parameters are used by all methods, but saves retyping if you decide to change methods.

Example: `findspots(1, 0.7, 0, 0, 0, 1, 255, 1, 0, 1, 2, 1000, 1, test.data)`

Counts grains using quick segmentation method, using a threshold of 0.7, printing labels in color 255, draws a size distribution graph, puts a rectangle around each spot, ignoring spots smaller in size than 2 or larger than 1000, and saves the result in file "test.data". If test.data already exists, the program will ask whether you wish to overwrite it. If the filename is omitted, you will be prompted for a filename.

This macro also sets the constants SPOTS and GRAINS, which can be used in mathematical expressions.

Command	Parameter#	Refers to
fopen	1	filename

Opens the designated file for writing. Unlike C, does not use a file pointer. Do not confuse with `open`, which loads an image from disk.

Command	Parameter#	Refers to
for	1	initializer
	2	condition
	3	increment

Executes following statement list as long as condition 2 is true. Statement list must be enclosed in braces. Parameters must be separated by semicolons (;).

Ex:

```
for(k=0; k<5; k++)
{  print("k is ",k,"\n"); }
```

Command	Parameter#	Refers to
foregroundcolor	1	red
	2	green
	3	blue

Sets the foreground color to the color containing the specified red, green, and blue components.

Ex:

```
foregroundcolor(255,255,255);
```

foregroundvalue see fcolor

Command	Parameter#	Refers to
fcolor	1	pixel value

Sets the foreground color to the specified pixel value, bypassing the conversion from red, green, and blue.

Ex:

```
fcolor(255);
```

Command	Parameter#	Refers to
frame	1	frame to display

Changes current frame of multi-frame image.

Example: `frame(172);`
Jumps to frame #172 of the current multi-frame image.

freegraincountingarrays - After grain counting, the grain data arrays remain allocated permanently, in case user decides to reanalyze it. This command will free about 500K of memory. Don't use spotsize(), spotx(), spoty(), or spotarea() after executing this command.

g Predefined variable for reading and setting green component of a pixel in current image.

Ex.: g = r*0.8 + g*0.2 + b*0.1;

Command	Param#	Refers to
getstring	1	Prompt string

Displays a box displaying the specified prompt string, which requests a string. Returns the string to your macro. (See also input())

Ex.: f = getstring("Enter a filename");

Command	Param#	Refers to
goto	1	line number in macro to execute next. A jump to a lower line number causes an infinite loop. Press *ESC* to stop the macro.

Jumps to specified line in macro

Ex: if(k<5) goto(44);

green Predefined variable for reading green component of pixel of an image. Can only appear on right side of an equation.

Ex.: g = green[1][0][x+5][y-4];

i Predefined variable for reading and setting pixel intensity value of a pixel in current image.

Ex.: i+=50;

if

Ex.: if(i>234) i=255; else i=0;

im Predefined variable for reading and setting imaginary component of Fourier transform of current image.

Ex.: im = im/1000 + imag[1][x+5][y-4];

imag Predefined variable for reading imaginary component of Fourier transform of an image. Can only appear on right side of an equation.

Ex.: im = imag[1][x+5][y-4];

image Predefined variable for reading pixel value of an image. Can only appear on right side of an equation.

Ex.: i = image[1][0][x+5][y-4];

imageregistration See registration

Command	Param#	Refers to
input	1	Prompt string

Displays a box displaying the specified prompt string, which requests a number. Returns the number. (See also getstring())

Ex.: a=input("Please enter a number");

Command	Parameter#	Refers to
intensity	1	changes value (intensity) of color or grayscale images by some factor (100=no change).

Note:
To change the RGB components of a color image separately, use "contrast".

Example:
To increase intensity contrast by a factor of 1.5: `intensity(150);`

The following two commands have an identical effect:

```
contrast( 100, 100, 100, 100, 100, 234);
intensity(234);
```

Command	Parameter#	Refers to
invertcolors	1	upper left x
	2	upper left y
	3	lower right x
	4	lower right y

Inverts colors between specified screen coordinates. If no arguments are given, it inverts the entire current image.
Ex;: `invertcolors(100,100,200,200);`

invertregion see invertcolors

Command	Parameter#	Refers to
left	2	distance

Moves specified image left by the specified number of pixels. If no image is specified, the current image is moded. If no distance is specified, distance is the current shift value (change with the gray +/- keys)

Command	Parameter#	Refers to
lighten	1	pixel value to add

Makes image lighter by adding specified value to each pixel. Suitable for grayscale but not indexed-color images. See 'darken'

ln

Returns natural logarithm of a value

load - see open

Command	Param#	Refers to
loadfft	1	filename

Opens the specified Fourier-transform file (which is in ASCII format) See sec. 7.18.2 for details on the format.

Command	Parameter#	Refers to
localcontrast	1	step

Performs local contrast maximization using the specified step value.

log

Returns natural logarithm of a value

log10

Returns base 10 logarithm of a value

Command	Param#	Refers to	Value
macro	1 Macro file name	Control is transferred to the first line of the new macro. After the new macro is finished, the old macro continues. There is no limit on the depth of nested macros except as imposed by available memory. Do not create recursive macros (macros which call themselves directly or indirectly).	

If an argument is given, it executes the current macro; otherwise, it opens the macro editor. Not useful in macro editor.

Command	Param#	Refers to
mask	1	image no. to change
	2	image no. for mask
	3	mode
		1 Mask (1&=2)
		2 Inverse Mask (1&= 2)
		3 Add (1+=2)
		4 Subtract (1-=2)
		5 Multiply (1*=2)
	4	x offset in pixels
	5	y offset in pixels

Performs image masking.
Ex: `mask(1,2,3,0,0);` adds image 2 to image 1 with no offset
`mask(1,2,4,2,2);` subrract image 2 from image 1, with image 2 shifted down and right by 2 pixels.

Command	Parameter#	Refers to
max	1	expression
	2	expression

Returns value of the larger of two expressions

Ex.: i=max(i-100, 0);

Command	Parameter#	Refers to
maximize, maximizecontrast	1	0=maximize red
		1=maximize green
		2=maximize blue
		3=maximize intensity

Maximizes contrast in current image or selected region.

Example: `maximize 2;`
Maximizes blue contrast.

If no parameter is given, (i.e., `maximize;`) it maximizes the intensity.

maximizecontrast

see maximize

Command	Param#	Refers to
message	1	Text to put in message box

Displays an information box containing the specified text.

Command	Param#	Refers to	Value
messages	1	Message state	0 off
			1 on

Controls message box behavior. Setting messages to 0 disables messages, including all requests for confirmation and requests for filenames. Messages are automatically re-enabled after the macro ends. Useful in batch mode. Turning messages off is dangerous and should only be used in fully-debugged macros. If messages are turned off, a macro could overwrite files, unload images, create a file in a non-standard format, or exit without asking for confirmation.

Ex: messages(0);

Command	Parameter#	Refers to
min	1	expression
	2	expression

Returns value of the smaller of two expressions

Ex.: i=min(i*2, 255);

| move | Toggles state of 'move/copy' button, which determines whether dragging on a selected area erases the original when it moves the object.

Command	Param#	Refers to
moveto		Reposition an image
	1	x coordinate
	2	y coordinate

Command	Parameter#	Refers to
movie	1	msec between frames

Starts a movie. **Ex.:** `movie(1000);`
Starts a 1 frame/sec animation of current image. The command version may be faster than using the dialog box because those Motif widgets can be a tad slow.

Command new	Param#	Refers to	Value
	1	Method for obtaining image coordinates.	1 Use mouse 2 Use specified coords. 3 Copy another image 4 Resize image using specified coordinates
	2	Border	0 No border 1 Add border
	3	x size (in pixel units)	
	4	y size (in pixel units)	
	5	x position (0,0 = upper left)	
	6	y position (0,0 = upper left)	
	7	Number of frames for image	
	8	0=put on main window, 1=separate window	
	9	0=Don't search for edge when creating multiframe images 1=search	
	10	Should subimages be aligned in panel (0 or 1)	
	11	No. of columns to create in panel	
	12	Left x coordinate	
	13	Upper y coordinate	
	14	Right x coordinate	
	15	Lower y coordinate	
	16	X Spacing between subimages in panel	
	17	Y Spacing between subimages in panel	

Creates a new image. Not all options are used for each method.

null - unclicks null buttons (see "Custom buttons").

Command	Parameter#	Refers to
open,	1	filename (can include wildcards)
load	2	x position for upper left corner
	3	y position for upper left corner
	4	x size (0 to 1) (1=full size)
	5	y size (0 to 1)

The remaining arguments (6 to 15) are used only for reading raw images.

Command	Parameter#	Refers to	Value	Meaning
open,	6	platform	0	PC
load			1	Mac
			2	IBM
	7	color type	0	RGB color
			1	grayscale/indexed color
			2	CMYK color
	8	bit packing	0	TIF-like
			1	GIF-like
			2	none
	9	x size in pixels		
	10	y size in pixels		
	11	No.of bytes to skip at start of file		
	12	image bits/pixel		
	13	red bits/pixel		
	14	green bits/pixel		
	15	blue bits/pixel		
	16	black bits/pixel		

Loads an image from disk

Examples:
`load(myimage.tif.gz, 100, 100, 1, 1);` Decompresses and reads myimage.tif.gz from disk, positions it at (100,100), at full size. Quotation marks are optional, since the argument must be a string.

The most recently loaded image becomes the currently-selected image for other operations.

Example: Load an image of raw bytes.

```
fileformat(9);
load(raw.img, 0, 0, 1, 1, 0, 1, 0, 382, 362, 10, 8);
```

The first command is necessary to override the automatic file detection, which would otherwise display a message “Invalid file type”.

The second command loads the image at x=0, y=0, x size=100%, y size=100%, platform=PC,color type=indexed color, x size=382 pixels, y size = 362 pixels, skip 10 bytes, and interpret data as 8 bits/pixel.

If a large number of raw byte images having identical parameters are to be read, it is easier to create a custom file type and define these values as the defaults. This will eliminate repetitive retyping of the arguments.

Do not confuse with `fopen`, which opens a disk file for writing in a macro.

| openimage | see open

Command	Param#	Refers to	Value
palette	1	Palette number	1 - gray scale 2 - spectrum 3 - multi colors 1 4 - multi colors 2 5 - multi colors 3 6 - RGBI 7 - black to green 8 - zebra (useful for testing image authenticity) >9 = palette no. 9-10000 -1 - Other (random) -2 = original image palette -3 = original image palette -4 = original background palette

Sets colormap of current image

Command	Param#	Refers to	Value
pasteimage	2	Mode	1 Background →transparent part of region 2 Background →visible part of region 3 Visible part of region →background 4 Transparent part of region →background 5 Stencil transparent part of region →background 6 Entire region →background

Sets paste mode for subsequent "paste" commands and starts interactive pasting.

pi

Returns 3.1416..

Command	Param#	Refers to	Value
pixel_interact_mode		Pixel interaction mode	1 Overwrite 2 Maximum 3 Minimum 4 Add 5 Subtract new - old 6 Subtract old - new 7 XOR 8 Average 9 Superimpose

Sets global method for putting pixels on screen

Command	Parameter#	Refers to
pixels	1	image no.

Returns total no. of pixels in specified image. The predefined variable PIXELS gives the no.

of pixels in the current image.

Command	Parameter#	Refers to
pow	1	expression 1
	2	expression 2

Returns expression 1 raised to the power of expression 2

Ex.: i=pow(i, 1.23);

Command	Parameter#	Refers to
print	1,2,...	a variable name, expression, or string const

Prints the specified items to stdout and flushes the output stream.
Ex.:

```
print("And the answer is...:\t", 21*sqrt(4), "!!!\n");
```

Note: Char consts such as (newline) are placed in double quotes, same as string constants. String variables are not yet supported. There can be any number of parameters. No formatting strings or type specifiers are needed. Any numeric expressions are evaluated before being printed.

Executing 'print' with no arguments calls the Print Image dialog.

plugin	1	Plugin pathname
	2...20	Arguments to pass to plugin

Ex: `plugin(plugin, 100, 200, 300, "stuff", 500);`

| quit | Quits the program and asks if you want to save your changes.

| r | Predefined variable for reading and setting red component of a pixel.

Ex.: r = r*0.8 + g*0.2 + b*0.1;

Command	Parameter#	Refers to
rand	1	random seed

Returns a random number between 0 and 1.

Ex.: i += rand(2)*4;

Command	Parameter#	Refers to
rdensity	1	x1 upper left x coordinate
	2	y1 upper left y coordinate
	3	x2 lower right x coordinate
	4	y2 lower right y coordinate

Measures the density of a rectangular region centered between (x1,y1) and (x2,y2). The result will depend on whether the image pixel values have been calibrated, and on the settings of the variables COMPENSATE and INVERT, which are automatically set in the densitometry dialogs. These variables reflect whether the pixel calibration is to be used and whether the maximum signal is black or white (sec.12). They can also be set in a macro. 'rdensity' returns the sum of the calibrated density value for each pixel, corrected for image depth. See also 'cdensity'

re Predefined variable for reading and setting real component of Fourier transform of current image.

Ex.: re = re/1000 + real[1][x+5][y-4];

read see open

readfft see loadfft

real Predefined variable for reading real component of Fourier transform of an image. Can only appear on right side of an equation.

Ex.: re = real[1][x+5][y-4];

red Predefined variable for reading red component of pixel of an image. Can only appear on right side of an equation.

Ex.: r = red[1][0][x+5][y-4];

registration

Warps an image to align an arbitrary number of user-defined fixed points in both images. Currently can only be done interactively.

Command	Parameter#	Refers to
resize	1	new x size in pixels
	2	new y size in pixels

Resizes window and image buffers for image. Image data remain unchanged in upper left corner of newly resized window. Useful for resizing to power of 2 for FFT, or in creating composite images. Use "changesize" to rescale the image data.

resizeimage see new

resizewindow see resize

restore restores image from backup

restoreimage see restore

Command	Parameter#	Refers to
right	2	distance

Moves specified image right by the specified number of pixels. If no image is specified, the current image is moded. If no distance is specified, distance is the current shift value (change with the gray +/- keys)

root Puts current image on root window.

| rootoff | Removes current image from root window.

| rootpermanent | see wallpaper

Command	Parameter#	Refers to
rotate	1	degrees
	2	0=no alias, 1=antialias

Rotates current image. Default is to use antialiasing.
Ex.: rotate(45.32);
rotate(264.1, 1);

Command	Parameter#	Refers to	Value
save	1	filename	
	2	file format	see under "file_format" for permissible values
	3	bits/pixel	(Any value between 1-32).
	4	compress	0=no compression 1=compress file

Saves image to disk file

The bits/pixel be a legal value for the specified format otherwise it is a fatal error. If omitted, it uses the image's current bpp. **Example:** `save(test.tif, 0);` Creates a TIFF file named "test.tif" using the currently-selected area or image.

If no arguments are given, the image's original filename and format will be used. Use 'file_format' to override the default image format. It may be necessary to change the filename extension in case the file is to be read by file viewers, which may require a specific file extension. If a macro tries to save a file without specifying a filename, a fatal error occurs, terminating the macro. If 'compress' is set, the current external compression programd will be used. This will be in addition to any compression imposed by the file format itself, so it is not very useful for GIF or JPEG formats.

| saveimage | see save

Command	Parameter#	Refers to
select	1	upper left x coord.
	2	upper left y coord.
	3	lower right x coord.
	4	lower right y coord.

Sets boundary for other operations such as contrast. The coordinate values are relative to the upper left corner of the screen (= 0,0). Select_region is cancelled by select_image and vice versa. All 4 parameters must be given.

Example: `select_region(100, 100, 200, 200);`
This has the same effect as clicking-and-dragging the mouse from (100,100) to (200,200).

Command	Parameter#	Refers to
selectimage	1	image no.

Sets current image
NOTE: Image numbering changes unpredictably when an image is unloaded.

Example: `selectimage(5);`

Has the same effect as double-clicking on image #5.

select_region,

selectaregion see select

separatergb Separates current image into 3 8-bit grayscale images representing red, green, and blue color planes. See "compositergb".

set Sets a variable.
Ex.: set(k=5);
Iteration is prevented for the command in parentheses (which need not be an assignment). If you typed `k=5;` by itself, the program would iterate through every pixel in the image, setting k to 5 repeatedly.

Command	Parameter#	Refers to
setbackgroundcolor	1	color
or	1	red
	2	green
	3	blue

Sets background color used by backspace and delete keys. Arguments should be appropriate for the screen depth, e.g., on a 24-bit display use `setbackgroundcolor(127,127,127)` or `setbackgroundcolor(8355711)`.

setbcolor see setbackgroundcolor

Command	Parameter#	Refers to
setcolor	1	color
or	1	red
	2	green
	3	blue

Sets foreground drawing color. Arguments should be appropriate for the screen depth, e.g., on a 24-bit display use `setcolor(127,127,127)` or `setcolor(8355711)` .

setfcolor see setcolor

setforegroundcolor see setcolor

Command	Parameter#	Refers to
setpixel	1	x coord.
	2	y coord.
	3	color
	4	pixel mode

Sets a single pixel at specified screen coordinates. If specified, color is a number appropriate for the bits/pixel of the screen; otherwise it uses the current drawing color. Pixel mode - see "pixel interact mode". After setting a number of pixels, use the command "repair" to rebuild the display.

shiftframe see shift

Command	Parameter#	Refers to
shift	1	x pixel distance
	2	y pixel distance

Shifts a single frame of the current image in its image buffer for alignment with other frames.

Example: `shift(-20, 34);`
Moves current frame left 20 and down 34 pixels.

shift-enter Pressing Shift+Enter simultaneously executes a single line in the macro. Use with caution, because the command may iterate over all the pixels in the image, even if the overall macro restricts it to a small loop. Clicking on the Execute button executes the macro starting at the current cursor position.

shrink see changesize

Command	Parameter#	Refers to
sin	1	radians

Returns sine of a value.

Command	Parameter#	Refers to
sinh	1	value

Returns hyperbolic sine of a value, i.e. (exp(x) - exp(-x)) / 2.

size see changesize

Command	Parameter#	Refers to
spray	1	Spray mode
		1 Fine spray
		2 Diffuse spray
		3 Math spray
		4 Filter spray
		5 Erase spray

Begins spraying things when left mouse button is clicked. Lick the main “Cancel” button to stop.

Command	Parameter#	Refers to
spotsignal	1	spot number

Returns total signal in a spot or grain

After performing a grain counting, the constants SPOTS and GRAINS are automatically set to the number of grains found. The total signal in each spot can be accessed using spotsignal().

Ex:

```
print("Total amount of stuff in spot 44 was ", spotsignal(44), "\n");
```

Command	Parameter#	Refers to
spotsize	1	spot number

Returns size of a spot or grain

After performing a grain counting, the constants SPOTS and GRAINS are automatically set to the number of grains found. The size of each spot can be accessed using spotsize().

Ex:

```
print("spot size was ", spotsize(44), "\n");
```

Command	Parameter#	Refers to
spotx	1	spot number

x coordinate of a spot or grain

After performing a grain counting, the constants SPOTS and GRAINS are automatically set to the number of grains found. The x coordinate of each spot can be accessed using spotx().

Ex:

```
print("spot was found at ", spotx(44), spoty(44), "\n");
```

Command	Parameter#	Refers to
spoty	1	spot number

y coordinate of a spot or grain

After performing a grain counting, the constants SPOTS and GRAINS are automatically set to the number of grains found. The y coordinate of each spot can be accessed using spoty().

Ex:

```
print("spot was found at ", spotx(44), spoty(44), "\n");
```

sqrt

Returns square root of a value

Command	Parameter#	Refers to
step	1	step size

Sets the step size by which an image is moved when you click the large arrow buttons on the information window.

| stop | Breaks out of currently executing macro. Unlike 'break', it terminates execution.
Ex: if(k>2000) stop

| switchto | see selectimage

Command	Parameter#	Refers to
tan	1	radians

Returns tangent of a value.

Command	Parameter#	Refers to
tanh	1	value

Returns hyperbolic tangent of a value, i.e. sinh(x)/cosh(x)

| testplugin | Same as executeplugin, except plugin prints debugging information to stdout.

| textdirection | sets direction for labels - 0=horizontal, 1=vertical

Command	Parameter#	Refers to
threshold	1	threshold value (0-1)

Thresholds an image, setting all pixels with density less than the threshold value to 0, all others to 1.

Command	Parameter#	Refers to
title	1	title

Changes title of current image.
Ex.: title("Newtitle");

Command	Parameter#	Refers to
transparency	1	transparency (0-100)

Sets the transparency value for an image.

| undo | restores image from backup

| unload | see close

| unloadimage | see close

| unzoom | restores zoomed image to normal size

| unloadall | Unloads all images. If 'messages' is 'on', you are prompted to save any image that has been modified.

Command	Parameter#	Refers to
unsetpixel	1	x coord.
	2	y coord.

Restores a single pixel at specified screen coordinates from the image backup, if it exists. After unsetting a number of pixels, use the command "repair" to rebuild the display.

Command	Parameter#	Refers to
up	1	Image no.
	2	distance

Moves specified image up by the specified number of pixels. If no image is specified, the current image is moded. If no distance is specified, distance is the current shift value (change with the gray +/- keys)

v Predefined variable for reading and setting density value of a pixel in current image. This is the pixel value corrected for image depth and scaled 0..1.

Ex.: v*=1.1;

w Predefined variable for reading and setting wavelet coefficient of current image

Ex.: w = w/100 + wave[1][x+5][y-4];

wallpaper Puts current image on root window, tiling it if necessary to create wallpaper. If image is animated, animation only continues for the first upper left copy. The root image remains after tnimage exits, but animation will stop.

Command	Parameter#	Refers to
wantimagetitle	1	0=no titles displayed 1=titles displayed

Sets config option of whether to superimpose image title on upper left of image.

wave Predefined variable for reading wavelet coefficient of a pixel in an image. Can only appear on right side of an equation.

Ex.: w = w/100 + wave[1][x+5][y-4];

whatis see evaluate
Ex: `whatis(k);`

Command	Parameter#	Refers to
while	1	condition

Executes following statement list as long as condition is true. Statement list must be enclosed in braces.

Command	Parameter#	Refers to
windowborder	1	0=minimal border 1=images are given a full border

If windowborder is 0, images are given a minimal border and positioned according to the coordinates specified as “x position” and “y position” in the “open” command. If windowborder is 1, images are positioned interactively and are given a title bar. This parameter only has an effect when windows=1. The actual behavior depends on the window manager.

Command	Parameter#	Refers to
windows	1	0=images are placed on main window 1=images are placed in separate windows

Controls whether images are put in separate windows or on main window.

Command	Parameter#	Refers to
write	1	file name to write data to
	2,3,...	a variable name, expression, or string const

Writes the specified items to specified file using fprintf, then flushes the output. The file must already be opened with fopen. Close file with fclose after writing.
Ex.:

```
write("myfile", "And the answer is...:\t", cbrt(74088), "\n");
```

Note: Char consts such as (newline) are placed in double quotes, same as string constants. String variables are not yet supported. There can be any number of parameters. No formatting strings or type specifiers are needed. Any numeric expressions are evaluated before being written.

x Predefined variable for reading x pixel coordinate.

Ex.: i = image[1][0][x+5][y-4];

y Predefined variable for reading y pixel coordinate.

Ex.: i = image[1][0][x+5][y-4];

zoom zooms image by selected factor

Command	Parameter#	Refers to
zoom	1	Image number
	2	Zoom factor

Ex.:
`zoom(1, 2.3);` Zooms image no. 1 to 2.3 times its original size.

13.3 Macro Programming Guide

Here is a brief tutorial on some of the finer points on creating useful macros. In general a macro should reset the program to its original state. If a macro does something different the second time it is executed, the cause is almost always a failure to reset the original conditions. In a real macro, you should also always execute "select_image" or "open" first to make sure the operation is performed on the correct image.

Example macro #1. Nested macros
Macro "beep.macro"

Line #	Command
1	beep;
2	beep;
3	beep;

Note: the words "Line #" and "Command" and the numbers 1,2, and 3 are not part of the macro. In this example, the macro would consist of only 3 words ("beep"), one on each line.

Macro "test.macro"

Line #	Command
1	macro(beep.macro);
2	beep;

Executing "test.macro" should cause a total of 4 beeps. A macro must never call itself - this would cause an infinite loop.

Example macro #2. Converting GIF image to TIFF format interactively

Line #	Command
1	a=getstring("What is the filename?");
2	messages(0);
3	fileformat(0);
4	save(a);
5	messages(1);
6	goto(1);

Line 1 - Prompts you for a filename each time

Line 2 - Resets default file format to 0 (TIF). If this line is omitted, the file format would default to the original format of each image.

Line 4 - Saves the image in TIFF format. Don't forget to rename the file extension to TIFF later.

Line 5 - Resume error messages

Line 6 - Unconditional loop back to line 1. Press *ESC* to stop.

Example macro #3. Converting GIF image to TIFF format in batch mode

Line #	Command
1	messages(0);
2	load(*.gif);
3	file_format(0);
4	save;

In the DOS version, this macro could be executed at 3 a.m. by setting the following timed events in `tnshell` or other command scheduler:

Line #	Command
Event 1:	cd c:\ tnimage 3:00am
Event 2:	tnimage -macro convert.macro 3:01am
Event 3:	ren *.gif *.tif 4:00am
Event 4:	cd c:\ 4:01am

Event 3 could also be replaced by a new line in the macro:

```
dos ren *.gif *.tif
```

Make sure there is enough free disk space before starting this macro.

Example macro #4. Subtracting two images

Line #	Command
1	pixel_interact_mode(1);
2	load cells.tif(0,0,1,1);
3	pixel_interact_mode(5);
4	load cells.tif(1,1,1,1);
5	pixel_interact_mode(1);
6	intensity(185);

This macro loads an image, then subtracts the same image after offsetting it by (1,1), then increases the intensity. All 6 lines are essential for a good macro.

Line 1: The pixel interaction mode is set to 1 (overwrite) in case a previous operation changed it.

Line 2: All 4 parameters (x offset, y offset, x size, and y size) must be given. Otherwise, if the macro was executed again, the default parameters would be 1,1,1,1 because of line 4. This would cause the image to be placed at (1,1) and then subtracted from itself, giving solid black.

Line 3: The pixel interaction mode is set to 5 (subtract). It is usually helpful to change the screen background color to 0 (black) before doing this.

Line 4: Load the image to subtract.

Line 5: Reset the pixel interaction mode to overwrite.

Line 6: Increase the intensity by a factor of 1.85.

Be careful using "DOS command" in combination with "messages 0". If there is insufficient low DOS memory to execute the command, "messages 0" may cause you to miss the fact that the command was not executed.

Example macro #5. Multiple image math formulas

Line #	Command
1	selectregion(50,50,100,100);
2	i=i+5;
3	i=i+image[2][0][x][y];
4	r=r*1.23;
5	i=i-5;

Line 1: Select a region to modify. If this line is omitted, it would modify the entire image.

Line 2: Lighten the image by 5 intensity units.

Line 3: Add the corresponding region from image #2.

Line 4: Increase the red contrast by 1.23 fold.

Line 5: Subtract 5 intensity units from the selected region.

Section 14

Miscellaneous features

Help menu

Displays the Help screen. Context-sensitive help is also available for several topics by clicking on the "Help" button. The help file, `tnimage.hlp`, can be edited to include site-specific information. Pressing F1 when the cursor is not on an image also calls the help screen. For more detailed help, see this manual, which is available in PostScript, dvi, and HTML formats.

Image notes

Pressing F1 when the cursor is on an image displays a small text editor containing the comments for the image. You can enter any text (up to 65536 characters) in this editor. Clicking the "Save" button in the editor's window will save the notes. This feature is useful for documenting changes made to the image, patient information, etc.

The image notes are saved in the file $HOME/.tnimage/notes/*filename* where *filename* is the same as the image filename. This has several implications:

1. Since the notes are not saved in the image file itself, they will not be present if the file is copied to another computer and will not be visible if a different imaging program is used.
2. The notes can be edited with any text editor.
3. The notes will be different for each user. This is unavoidable because of security considerations. `tnimage` is typically not run by root and could not access any common area without giving any user on the system the ability to read, change or delete any image note.

Frame rate test

Alt-T - Test video - Tests the maximum frame rate of your video system.

Frame rates will be much lower in Unix X11 versions and in Windows.

Emergency image repair

Alt-R - Rebuild screen display If for some reason the display is incorrect, Alt-R will cause it to be rebuilt.

Repeat previous command

Shift-Enter - Repeats the previous command.

Erase image

Alt-E - Erase image Erases the currently-selected image (after asking you). Useful if memory somehow gets filled with images and there is insufficient memory to open a window. This can sometimes happen in a Windows DOS box but should not happen in DOS or Unix.

Sketch mode

F2 - Toggles Sketch mode on/off

Area selection mode

F3 - Starts manual area selection mode (click Cancel to stop)

Setting Wallpaper

The image can be placed in the root window using the "root" command in the Command Editor. Most operations, including animation, are still functional; however, it is not possible to copy and paste regions of the image on the root window.

The "root-off" command restores the image back into its normal window.

The "root-permanent" and "wallpaper" command puts multiple copies of the image in the X11 display background. This becomes permanent wallpaper for the display; however, animation ceases when you exit `tnimage`.

14.1 Erasing

Labels, graphic elements, and any other changes made to an image can be removed by clicking the "Erase" button on the left, and dragging the mouse over the area to restore while pressing the left mouse button. The area near the mouse cursor will be restored with the image's backup if it exists (*see Backups,* Sec. 6.7 and 11.4). To return to normal mode, click the "cancel" button. The size of the restored area can be changed by selecting "Config..Configure..Spray factor".

Batch mode processing

Tnimage can also be run in "batch" mode for unattended operation. This is useful for operations that are time-consuming or need to be done every day. A cron-like utility or (for the DOS version) a DOS command scheduler such as `tnshell` (available from the author) can be used to automatically carry out any sequence of operations at (for example) 3 a.m. on Thursdays or to automatically load a series of images before you come to work.

Procedure: 1. Create a macro in tnimage and save it to disk. The last command in the macro should be "exit" to ensure tnimage quits when finished. The command "messages 0" can also be used to ensure that any error messages are ignored instead of causing the program to wait for an acknowledgment (Make sure the macro works before using this).

2. Using `tnshell` (or other program), set the following DOS command to be executed at the desired time (e.g., 3 a.m. every Saturday): `tnimage -macro` *macro_file* where *macro_file* is the name of the macro (without brackets). See *Macro programming guide* (Sec. 13.3) for examples of useful batch macros.

Editing text files

The Macro Editor in tnimage can also be used as a text editor for small (up to 65536 characters) text files. This can be used to take notes on image analysis, create color-remapping files, etc. as well as writing macros. The Unix version also supports clipboard cut & paste.

Command line interface

Some users find it more convenient to type commands than select items from the mouse. The macro editor in tnimage doubles as a command-line interface. Moving the cursor to the end of a line and hitting <Shift-Enter> or the <Ctrl-M> key causes a single line to be executed. This has the additional advantage that a list of commands can be saved in a file, providing documentation and enabling the same commands to be executed later. The main window can also be iconified and images loaded into individual windows, so that complete functionality is obtained without use of a mouse if desired. For a complete list of commands, see *Macro Commands* (Sec. 13).

Reading mail attachments with tnimage (Unix version)

To use tnimage as an image viewer for MIME mail attachments, add the following line to /etc/mailcap or ~ /.mailcap and remove all other lines containing "image":

`image/*; showpicture -viewer /usr/local/bin/tnimage %s`

or `image/*; /usr/local/bin/tnimage/tnimage %s` .

This will cause your mail reader (e.g., `pine`) to automatically run tnimage when an image attachment is to be viewed.

Section 15

Windows and OS/2 compatibility

This section applies to the DOS version only.

It is recommended to use the included PIF file (TNIMAGE.PIF) when running `tnimage` under Windows 3.x.

The suggested procedure is:

1. Copy and unzip `tnimage` in the desired location, e.g., C:\ TNIMAGE.
2. In Windows, Select "File...New program item"
3. Set "Description" to "Tnimage".
4. Set "Command line" to C:\ TNIMAGE\ TNIMAGE.PIF
5. Set "Working directory" to C:\ TNIMAGE
6. Set "Shortcut Key" to "none"
7. Leave "Run minimized" unchecked.
8. Select "Change icon" and enter "C:\ TNIMAGE\ TNIMAGE.ICO"
9. Click on "OK".

Virtual Memory

Because Windows takes over virtual memory, running `tnimage` under Windows will affect virtual memory usage. Before starting `tnimage`, make sure the amount of Virtual Memory selected in the Windows Control Panel is greater than the amount of RAM in your computer. Also, check to make sure there are several MB of free disk space and that all other applications and windows are closed. Failure to do this can result in frequent "Out of memory" errors. Windows' virtual memory is much slower than the VM built into `tnimage`. Hence, there is no particular advantage to running `tnimage` under Windows. Also, as with all applications, speed under Windows will be markedly slower than in DOS.

A peculiarity of Windows is that if you have the wrong mouse driver installed in Windows, you can lose control of the mouse when you run a mouse-based DOS program such as `tnimage`. If this happens, run the "setup" program in Windows to ensure that Windows is set up to use the mouse driver that is appropriate for your mouse. For example, if you have a Logitech mouse, using the Microsoft mouse driver may cause the mouse to become immobilized when you return from `tnimage`. Because Windows uses a different mouse driver from DOS, the mouse may work under Windows but not in a DOS box. If the mouse doesn't work right when running under Windows, you almost certainly need a new (or different) mouse driver. An alternative approach which sometimes works for some reason is to change Windows from its default VGA screen mode to a 256 color mode. Many of these problems have been fixed in Windows 95.

Windows 3.1 also has an "unexpected feature" of sometimes not redrawing its own screen when returning from DOS graphics programs such as `tnimage`. One solution is to "minimize" the

Windows screen after returning from `tnimage` and then immediately "Restore" it. This forces Windows to redraw its screen.

Computers with ATI Mach 32 cards have problems running `tnimage` in true-color mode from within a DOS box. In this case, the only solution is to run `tnimage` from DOS, or use `tnimage` in an 8 bit/pixel mode (e.g., mode 103). (See "Command-line options" above). Mach 64 cards are a little better (they can handle 16 bit/pixel modes, but still can't handle 24 bit modes in a DOS box). Most other cards have no problems.

For both OS/2 and Windows, `tnimage` must be run in "full screen" mode. For OS/2, it may be necessary to change the DOS box parameters to get `tnimage` to run properly. `tnimage` may not run on all OS/2 configurations, because it needs to access the hardware.

Note: No technical support is available for `tnimage` running under OS/2.

NOTE: In a Windows DOS box, the virtual memory manager in `tnimage` sometimes gets confused if Windows is set up for a small amount of virtual memory. This causes the "Free Memory" display to be incorrect, and more importantly, allows you to completely fill up memory with images, making it impossible to use the menus. If this happens, you can still erase an image by clicking on the image you want to erase and pressing Alt-E. It is therefore recommended that Windows' virtual memory be set to a size greater than the total amount of real memory.

NOTE: When running `tnimage` in Windows, if Windows' display resolution is changed, it must also be changed the next time `tnimage` is run in order for the new resolution to take effect. For example, if the resolution is changed to 1600x1200, start `tnimage` by typing

```
tnimage -xres 1600
```

This only has to be done once. Note that `tnimage` must be run in full-screen mode.

Changing Computers

If problems occur after moving `tnimage` to a different computer, after changing video cards, or upgrading to a new version, erase the file TNIMAGE.INI. This file contains `tnimage`'s video mode selection, and may inhibit automatic detection of the video chip in your new computer. `Tnimage` will revise the file automatically if it detects an incompatibility.

This is also important in the Unix versions.

Section 16

Problems

Problem: Message box popping up on screen repeatedly during densitometry.

Solution: This is caused by clicking the "Accept" button too many times.

Problem: When typing on the screen, the cursor moves but no text is visible; or after adding a label, no label is visible.

Solution: Change the foreground color to make it different from the background color.

Problem: A 16-bit image known to be grayscale is displayed in psychedelic colors instead of shades of gray.

Solution: The image has somehow been mis-identified as a color image. Select "Image Properties" and set the Color Type to Grayscale.

Problem: A 32-bit image appears completely black after being loaded.

Solution: Check the setting "CMYK→RGB" in the Read Image dialog. This box should stay checked for normal images.

Problem: When drawing lines or boxes, the "rubber band" lines are not visible.

Solution: The "rubber band" lines are XOR'd against the background color. If this color is near the middle of the color range, the lines will not be visible. Change the background color to some other color.

Problem: Program crashes on start-up or refuses to start.

Solution: Create a log file by typing `tnimage -diag > logfile`. Send a bug report by e-mail, and attach a copy of the log file so the problem can be fixed.

Problem: Strange colors on screen or `tnimage` only using top 1/12 or 1/7 of the screen (DOS version).

Solution: Some other Super VGA programs and communication programs (such as some versions of Carbon Copy) contain an "unexpected feature" that results in the computer being left in a state that prevents proper functioning of `tnimage` with certain video cards. The DOS shell program 1Dirplus and one HPLC data-acquisition program also have a similar feature which can interfere with the colormap registers. If `tnimage` suddenly begins to run incorrectly after running one of these programs, reboot the computer and start `tnimage` again. This can also occur when running `tnimage` as a DOS box in OS/2. Change the DOS box parameters to correct the problem.

Problem: Entire screen compressed into upper 1/7 of the screen (DOS version).

Solution: (1) Try starting `tnimage` with the command line:

```
TNIMAGE -OLDVESA
```

(2) If that doesn't work, you probably are using an outdated VESA VBE TSR (Terminate-and-stay-resident program). Obtain a new version from the video card manufacturer. Note that most new video cards already have VESA BIOS built in, and don't need a TSR. Try running `tnimage` without it. If nothing works, contact the author for assistance.

Problem: Lower part of screen flickering (upper 2/3 of screen is okay, lower 1/3 is "snow" or "static" (DOS version).

Solution: This occurs when the video card reports to `tnimage` that it is OK to set the specified high-resolution screen mode, but in fact the card has insufficient video RAM to handle it. This will occur on certain Cirrus-based cards. To solve the problem, it is necessary to install additional memory in your video card. Using `tnimage` when this problem occurs could result in a system crash.

Problem: Screen is completely white (DOS version).

Solution: This occurs when using one of the "experimental" Tseng modes, if the video card is unable to handle the specified screen mode. Press Alt-X two or three times to return to DOS, then use the "mode" command line option to run `tnimage` in a different resolution.

Problem: Multiple copies of Menu Bar (DOS version).

Solution: This is usually caused by an old video card with VESA 1.0 BIOS. Start `tnimage` again using the command line:

```
TNIMAGE -OLDVESA
```

This can also happen if the `UNIVBE` TSR is present. `UNIVBE` must be removed in this case.

Problem: Multiple horizontal copies of image on laptop computer in 24 bit/pixel mode, while 8 and 16 bit/pixel modes are normal.

Solution: Start `tnimage` with the command line

`tnimage -nosparse` .

If this solves the problem, add the line

```
nosparse 1
```

to the `tnimage.ini` file. This is a bug in the X server.

Problem: Printer prints part of image, then gives an error message or goes into a "form feeding frenzy".

Solution: This can occur on laser printers if your printer has insufficient internal memory to handle the image. You will have to install additional printer memory, print a smaller image or print at a lower resolution or dither size. Generally, 4 MB is required to print a 8x10 inch page in black-and-white at 600 dpi. Color printing requires 3 or 4 times as much. Unfortunately, there is no way for `tnimage` to test how much memory the printer has. For some reason, printer manufacturers have also made it impossible to print an image in small segments. This problem does not occur on inkjet printers.

This will also occur if printing is attempted on a dot-matrix printer.

Problem: Mouse gets "lost" when running `tnimage` from a Windows DOS box.

Solution: This occurs with some Microsoft and possibly other mouse drivers. Exit Windows, and replace your mouse driver with a newer version or with a Logitech mouse driver. Then use Windows' Setup program to change to the new mouse. Alternatively, use Windows' Setup program to change Windows' screen display to 256 colors (It's not known why this helps). These problems have not been observed in Windows 95.

Problem: Screen is not restored correctly after running `tnimage` from a Windows DOS box.

Solution: Install a different video driver in Windows using Windows' Setup utility.

Problem: Mouse jumps around on the screen, or exhibits jerky movement as if trapped in a grid.

Solution: Obtain a new mouse driver. This can usually be obtained from a bulletin board or ftp site operated by the mouse manufacturer. Alternatively, Logitech mouse drivers can work on many mice. Older mouse drivers often have trouble in SVGA modes. Microsoft mouse drivers are particularly prone to these sorts of troubles. Some old SVGA cards, such as Realtek video cards, create difficulties with the mouse, making it possible to reach only x and y coordinates that are divisible by 4. In this case, the video card must be replaced.

Problem: Screen is garbage when running `tnimage` from a Windows DOS box, even though there is lot of video RAM.

Solution: This is a problem in computers with ATI video cards in Windows for Work Groups for some screen resolution modes. Run `tnimage` at a lower color depth by changing the "Command Line" entry in Windows' "Properties" menu to:

```
C:\TNIMAGE\TNIMAGE.PIF -mode 103
```

Alternatively, run tnimage from DOS.

Problem: Unusual behavior or display colors immediately after upgrading from a previous version of tnimage.

Solution: Delete the configuration file "tnimage.ini".

Problem: Colors appear posterized.

Solution: This will happen on 8-bit displays if other applications (such as Web browsers or certain window managers) allocate a lot of colors, This can also happen after using certain other image viewing programs, particularly if they crash and fail to release their allocated colors. If closing other applications or restarting the X server doesn't help, try starting the X server in 16- or 24-bit mode. If this is not possible, it may be necessary to switch to a different window manager.

Problem: Operations such as inverting colors, etc. on the main window give a solid color instead of the expected inverted text.

Solution: This occurs when foreground and background colors are adjacent numerically (e.g., 40 and 41) but their actual colors (determined by the colormap) are different. Inverting the colors makes the color values 215 and 214, which map to similar colors. Change the foreground or background color to something else.

Problem: Message "Invalid image file" appears after transferring an image file from a Unix or mainframe system.

Solution: The most common cause of this problem is the failure to use "binary" mode when sending the image file over the network. If you are using Kermit, give the command

```
set file type binary
```

before getting the image. If you are using ftp, give the command

```
binary
```

before getting the image. It is very difficult to fix an image file that has been corrupted by sending it in text mode.

Problem: Opening or processing of images is slow.

Solutions: (1) Speed can be greatly increased by starting tnimage in a screen mode that has the same color depth as the images. Analyzing 24-bit color images in 8-bit mode, for example, will be slow because the entire image projected to the screen has to be re-palettized every time any operation changes one or more pixels.

(2). If your hard disk light goes on during an image processing operation, this means tnimage is paging out to disk. This can be reduced by installing more memory in your computer.

(3). Speed can be approximately doubled by running tnimage from DOS instead of in Windows, or by removing any expanded memory managers such as 386 Max or (especially) EMM386, which slows down math operations by over 50%.

Problem: Colormap rotation in startup screen is slow or jerky.

Solution: If you are running the Unix version, this can happen if another copy of tnimage is running, or if the mouse is moved suddenly during startup. In the DOS version, this is usually caused by a slow video card.

Problem: "Insufficient memory" message occurs even though "About the program..." indicates there should be enough memory.

Solution: Turn "Automatic undo" option off. This will cut the storage requirements in half. Also, a certain amount of memory is also reserved by tnimage for dialog boxes, etc. and is unavailable. If this happens in Windows, you must increase the "Virtual Memory" setting in Windows ("Control panel..386 Enhanced..Virtual memory.)".

Problem: Video card won't go into desired screen mode, even though video card manual states that mode is supported.

Solution: (1) If ATI card: Run ATI's "Install" program again to verify the specified video mode is activated. (2) Run tnimage's diagnostics by typing: tnimage -diag to determine the problem, which is usually caused by insufficient RAM or a monitor limitation. (3) Another possibility is that the computer was turned on before turning on the power to the monitor. Some video cards only examine the monitor type at power-up, will become confused if the monitor is off during a cold boot-up, and refuse to set high-resolution modes.

Problem: Monitor makes a buzzing sound and fails to sync on startup (causing jagged diagonal lines on the screen).

Solution: Turn the monitor off immediately and press Alt-X several times to stop the program. Restart tnimage using a lower screen mode. This problem could occur if your video card reports that it is safe to set a given screen mode when in fact it is not safe. Only one card has been found so far that does this, when setting a 1280x1024 resolution mode. Prolonged operation in this condition could harm your monitor.

Problem: A grayscale image appears completely black or white.

Solution: This is usually caused by the automatic grayscale mapping feature misinterpreting the upper and lower gray levels of your image. If the image is being read as “raw bytes”, make sure you have selected the correct number of bits/pixel, x size and y size. If the y size you entered is too large, tnimage may read beyond the end of the image and incorrectly estimate the maximum and minimum levels. To correct this, select “Color...Set colormap...Grayscale mapping” and change the maximum and minimum values to correspond to those in the image. (For example, if the image is known to be 12 bits deep, the maximum should be approximately 2^{12} or 4096.)

Problem: “Mouse required” message occurs when starting tnimage, even though Windows runs OK with the mouse.

Solution: Windows has its own separate mouse driver. It is still necessary to load a mouse driver for DOS applications. Run the program “mouse.com” that came with your mouse.

Problem: When printing a grayscale image in PostScript mode, changes in the contrast, brightness, or colormap are not reflected in the printout.

Solution: Set “Color type” to RGB/indexed color instead of BW/grayscale.

Problem: When typing text on the screen, it comes out in the wrong color.

Solution: The available colors are determined by the color depth, the gray scale mapping parameters and colormap (if present) of the image you are typing on. For example, if an 8-bit image with a particular colormap is loaded, you can only add colors that exist in that colormap, even if tnimage is in a 24-bit color mode. This prevents you from accidentally adding a color which is impossible to save with the image. If all else fails, press Alt-R to rebuild the screen display.

Problem: The message “Converting from 24 to 8 bits/pixel” stays on the screen for a long time.

Solution: Select “Color...Color settings” and make sure “Color reduction method” is set to “Quantization” and not “Fit current colormap” which can be slow.

Problem: When reading color images, the colors are wrong, or it takes a long time.

Solution: Select “Color...Color settings” and make sure “Color reduction method” is set to “Quantization” and not “Fit current colormap” which may give incorrect colors.

Problem: After reading a GIF file or a color image, subsequent images in other formats appear to be “garbage”.

Solution: This only occurs in 8-bit/pixel screen modes. When an image that does not have its own colormap is loaded, the colormap is not automatically changed. If the current colormap is discontinuous, the new image may appear to contain garbage (of course, it is not really garbage). Simply click anywhere on the background, or use the “Change colormap” menu option to change to a continuous colormap.

Problem: Medical grayscale images appear grainy or posterized; or appear as strange shades of blue instead of gray.

Solution: The wrong target platform may have been selected. Select “File...Create custom format” and change the “Target platform” and/or “Bytes to skip” for the specified format until a smooth image is obtained.

Problem: Image manipulation creates a distorted image (e.g., 3 copies of the original, or colors wrong).

Solution: This or other strange problems can happen if the X server is running out of resources, for example, after a program has crashed. Try restarting the X server. If this doesn't help, send a bug report (see Sec. 1.5).

Problem: Deconvolution result is completely black, completely white, or garbage.

Solution: This can be caused by (a) using an inappropriate point spread function, (b) using a point spread function that has zero or near-zero intensities at one or more frequencies, or (c) wrap-around effects. Sometimes, adding noise to the point spread function will increase the intensity at the missing frequencies and solve the problem. Alternatively, try a less ambitious point spread function. Try dumping the FFT data to disk and reading it with a text editor. If the numbers are all "-nan"s, this is a sign that the intensities were too low and a different psf must be used. Some images simply don't work well with deconvolution.

Problem: Spots floating across screen, or jagged flickering lines.

Solution: You are working too hard, go lie down.

Problem: "Fill Region" was selected, but nothing happened when mouse was clicked on the region to fill.

Solution: Make sure the area being filled is darker (lower intensity value) than *both* the Max. Border Color and the Min. Border Color. Filling will stop whenever any color between these two values is encountered.

Problem: Mouse cursor moving by itself.

Solution: Make sure mouse is on a level surface.

Problem: "out-of-memory" problems running tnimage in a Windows DOS box.

Solution: This is usually caused by an incorrect setting in your virtual memory. Try the following procedure to fix it:

1. Click on Windows Control Panel
2. Click on "386 Enhanced"
3. Click on "Virtual Memory"
4. If the size of the Virtual memory is less than the amount of RAM in your computer, problems may occur. Change the amount of Virtual Memory to the maximum recommended by Windows; or exit Windows and run tnimage from DOS.

If this doesn't work, check the free disk space with `chkdsk.`If insufficient disk space is available for paging, Windows may decide there is not enough memory to run the program. It may also help to close some other applications.

Problem: M-x psychiatrist doesn't work.

Solution: You must be thinking of some other program.

Problem: Display is completely blue, with funny little hexadecimal numbers in the middle.

Solution: Your computer may be still traumatized from having had Some Other Operating System (TM) on it in the past, and is compensating by attempting to reenact the traumatic event and its accompanying "Blue screens of death" (BSOD). Try rebooting a few times to assuage its separation anxiety. Computer counseling may also be of benefit. Running `emacs` in Psychiatrist mode once a week for a year or so will help; however, it is computationally expensive.

16.0.1 Limitations and known bugs

In the Unix version, the width or height of the viewable area cannot be made larger than the width or height of the screen.

Not all functions work correctly on zoomed images.

Chinese, Japanese, Arabic, and Hebrew fonts are still not handled correctly.

It is not possible for an image to be transparent and chromakeyed at the same time.

It is possible to specify a print command (such as “lpr -P ” followed by a space) that causes lpr to hang. This is a problem with lprng.

It is not possible to perform wavelet transforms and Fourier transforms on the same image at the same time. This limitation was deliberately introduced to conserve memory.

In mwm (Motif Window Manager) and olwm on Solaris (but not Linux), the arrow keys do not move the image the correct distance for independent windows.

If the image is off screen, moving it with the Slew button moves the image faster than when the image is visible.

Display of animated GIFs is slightly faster than in Netscape for some reason. Also, all images written in GIF89 format are positioned at 0,0 instead of their normal screen coordinates because it was discovered that including the position causes Netscape to crash when the GIF is incorporated into a Web page.

It is necessary to quit and restart the program to get the menu bar to wrap around onto multiple lines when the main window is made smaller than the menu bar width in fvwm and olwm.

In fvwm, if an image in a separate window has scrollbars, the “copy” and “move” functions don’t work on the main window.

Densitometry and image rotation do not work correctly on a Fourier-transformed image.

Plugins do not work in Solaris or ConvexOS. Feedback on this problem from Solaris programmers will be appreciated.

It is impossible to select an area while a 3D image is being played as a “movie”.

Several features, including the scanner interface and image rendering on 16- and 24-bit displays have not been tested on SGI or Solaris X servers. Information as to whether these features work correctly will be appreciated.

If an image in a separate window is behind the main window, labels placed on the main window will actually appear on the image instead.

In 16-bit screen modes, changing contrast or multiplying red, green, and blue by a constant factor has a disproportionate effect on green. This is an unavoidable consequence of the fact that in 16 bpp mode, green has 6 bits while red and blue have only 5.

Animated chromakey images have the incorrect background color when placed in the root window.

3D graphs occasionally draw pixels on images in the background.

Section 17

Error messages

17.1 All versions

Insufficient conventional memory to run program

Insufficient extended memory to run program

These errors usually mean an extended memory manager is required but not present. The simplest solution is to find the file HIMEM.SYS (which comes with DOS) and install it in your CONFIG.SYS file, e.g., add the line

```
C:\DOS\HIMEM.SYS
```

to CONFIG.SYS (make sure HIMEM.SYS is in the DOS directory before doing this).

Also, check to see if you have an expanded memory manager installed. These programs sometimes block access to portions of video memory or high memory.

Not enough memory

Insufficient memory to perform the requested action (see "Operating under low-memory conditions" above). This message should not occur in the DOS version of tnimage, unless there is insufficient disk space.

Insufficient memory to convert to 8 bits/pixel

A color map table could not be generated for the image. When this happens, the image cannot be converted to the screen resolution, leaving "garbage" on the screen. Try changing 'color reduction method' to "fit current colormap". This method uses less memory. Alternatively, start tnimage in a color mode and try again. This message should not occur in the DOS version of tnimage, unless there is insufficient disk space.

Internal error [...]

Contact author for assistance, specifying the exact message and the circumstances under which it occurred.

Can't compact free space

This can occur when you try to unload an extremely large image. The only consequence is that the maximum amount of free memory could not be recovered. Usually, erasing the image again will free up the space.

Too many images

The limit of 512 images was exceeded while attempting to read an image from disk or create a new image. Don't forget that a new image is also created when you try to rotate or resize an image or carry out an FFT.

Image was not backed up

This only occurs when you select "Restore" when the "Auto undo buffer" option was unchecked, so that new images were not automatically backed up. Although un-checking 'auto undo buffer' creates more free memory, it means that it will be impossible to undo any changes by selecting "restore".

Function not available

The function has not yet been implemented. Contact the author for availability dates.

Error - mixed expand and shrink operations

You cannot shrink an image in one dimension and enlarge it in another simultaneously, it must be done in two passes.

Error: Bad OD table, Turn pixel compensation off

Densitometry could not be performed because part or all of the selected region was on an image whose O.D. table mapped two or more pixel intensity values to the same O.D. You must either repair the OD table, or select "No pixel compensation".

Bad scan parameters

This occurs if the starting and ending points of your densitometry scan are the same, or if the region could not be scanned for some reason. Select the region again.

Can't create file!

An invalid file name was specified, or the filename was the same as some existing file that DOS has marked as "read-only", or a "directory".

Can't find file!

The specified filename does not exist, or is marked by the operating system as "hidden". Use a utility such as NSHELL to un-hide the file.

Only the 1st image in your file will be displayed

Your TIFF file contains more than one image.

Sorry, can't read xxx compressed TIFF images

TIFF files are sometimes compressed using a weird form of compression not supported by tnimage.

Not a valid TIFF file

The TIFF file is corrupted and unreadable, or is an unusual non-standard TIFF format.

Warning: File transfer aborted

You aborted the operation of saving the image to a file by pressing *ESC* or clicking on "Cancel"; or, saving of the file was aborted by the program for some reason.

An error occurred

A "Critical error" occurred in connection with a file transfer. This normally leads to the message "Abort, Retry, Ignore?". This occurs if you try to save a file to a floppy drive that does not exist, does not have a disk in it, has an open door, etc. It can also happen if a problem is found with your disk.

No images to save

You selected ”Save image”, but no images were currently loaded.

Printer not responding!

Printer I/O error!

Printer out of paper!

lp0 on fire (Unix only)

The printer is off-line, powered off, on fire, or something.

Unable to read disk

The specified drive letter referred to a non-existent disk; or there was a disk drive failure. Occasionally, CD ROM drives will spuriously give this error.

Invalid GIF file

Error decoding GIF file

Unable to read GIF file

The GIF file is corrupted and unreadable. The most common reason for this is that, when you downloaded the file, you forgot to set the communications package to “binary” mode. For example, in ftp, before getting the file, type:

```
binary
```

In Kermit, before telling the remote Kermit to send the file, type:

```
set file type binary
```

You may also need to make sure your local computer is set to “binary” mode. The command for this differs for each communications package.

Number of colors doesn’t add up

Change the bits/pixel on a color or number of primary colors

You have selected an incompatible combination of options. The number of primary colors must correspond to the number of colors which have non-zero bits/pixel. For example, if you select “1” as the number of primary colors, only one of the 4 entries (red, green, blue, or black) can have a non-zero number of bits. The rest must be 0.

RGB bits/pixel don’t add up to total

Change total bits/pixel or number of colors

You have selected an incompatible combination of options. The sum of the red, green, blue, and black bits to be used to save an image must be equal to the total bits/pixel selected. For example, if you select to save the image as 17 bits/pixel, the red + blue + green + black bits must also total up to 17. If you are reading a raw 8-bit monochrome image, the “color type” must be set to “Gray Scale/Indexed”.

You must have only 1 primary color to save data as grayscale

You have selected an incompatible combination of options. If you want to save the image as “Gray scale” image data, you must also change the “no. of primary colors” to 1.

Non-standard image format - is this ok?

This means the file format selected is not standard, and some other programs may have difficulty reading the file that is created.

Not enough bits for CMYK

You must either select 32 bits/pixel mode, or “other bpp” mode, in order to create a CMYK file. If you select “other bpp”, you need to specify the number of bits to use for black as well as red, green, and blue.

You must select CMYK to use black bits

Setting a non-zero value for “black bits/pixel” is only allowed if you have also selected “CMYK”.

Must have 4 colors for CMYK

If you selected “other bpp”, you need to also select “4” as the number of primary colors in order to create a CMYK file.

PCX files must be 8 bits/pixel

Tnimage can only create 8 bit/pixel PCX files. The image must be converted to 8 bits/pixel before saving it as a PCX file.

IMG files must be 1 or 8 bits/pixel

Tnimage can only create 1 or 8 bit/pixel IMG files. The image must be converted to 8 bits/pixel before saving it as an IMG file, or “monochrome” should be selected.

File is an unsupported type of PCX file.

The PCX file had too many bit planes, or was not 1 or 8 bits/pixel.

No header file specified

This only occurs when “Create custom file” is selected. If you specify a non-zero number of header bytes to copy from a file, you must also specify a file name.

Specified header bytes exceeds length of header file

This only occurs when “Create custom file” is selected. You selected a number of bytes to copy into the header of a custom file format which exceeded the file size of the specified file.

Bad gradient parameters

One or more of the coordinates specified for a gradient fill, or one of the boundary colors, was bad. Reselect another similar area and try again.

Invalid parameters, Setting kernel multiplier to 1

For some types of filtering, the kernel multiplier must be set to 1. Tnimage has done this automatically.

Kernel size is larger than selected area

The size of the kernel to be used for for filtering, multiplied by the kernel multiplier, is larger than the image. This would result in no filtering, so the filtering was stopped. You may have accidentally click-and-dragged a 1x1 area. Try re-selecting the area to filter again.

Bad scan parameters

A trapezoidal region selected for scanning happened to have an illegal combination of values. Select a slightly different region and try again.

Error converting image

Contact the author for assistance.

Too many peaks

The number of peaks exceeded the size of the list tnimage allocated for them. Contact the author to obtain an upgrade which can handle more peaks.

Can't find help file

The help file, TNIMAGE.HLP, was not in the starting directory.

Bad regression order

This is usually caused by trying to draw a curve or calibrate an image using only one calibration or control point. At least 3 control points must be selected.

No unique solution

Division by zero

These messages indicate that it was impossible to calibrate the image using the existing calibration points. Calibrate the image again using slightly different control points.

Error: negative value in logarithm

When calibrating an image in logarithm mode, all calibration values must be positive.

You must select "Other" or "Custom" to save an image with non-standard parameters

Tnimage detected a non-standard value entered for bits/pixel or number of colors. To protect against accidentally creating non-standard files, it is necessary to specifically check the "Custom" file format or the "Other" bits/pixel selection.

Error: need two images to convolute

Convolution or deconvolution of images was selected, but less than 2 images were present.

nn Divisions by zero detected

During deconvolution of two images, if the frequency of the 2nd image is zero at any point, division by zero occurs. If there are a large number of these, it usually means the 2nd image is not entirely appropriate to use for deconvolution.

Can't open list file

A file containing a list of images was expected, because of the "-FILES" option, but either no file list name was given or it was an invalid file name.

Extension must be TGA for Targa format

Since there is no foolproof way to determine whether a file is really in TGA format, tnimage will classify any file with a TGA extension as a Targa file. If you specified some other extension, the file would be unreadable. Thus, tnimage does not allow a file to be saved with any extension other than "TGA".

Unknown error - nothing was saved!

This message should not occur in normal use. Contact the author for assistance.

Selected bit/pixel values will be ignored

Images are automatically converted to 24 bits/pixel before saving in JPEG format. Other values are illegal and are ignored.

Color information will be lost!

You should convert image to 8 bits/pixel first

You are about to save a color image in a file format that does not have color information. When loaded back, the new image might not have the same colors as the original. Select “Change image depth” and quantize the image first by converting it to 1 byte per pixel.

Error: overlapping offsets

While creating a custom image format, you cannot set two or more file offsets to overlap the same byte position. Each offset occupies 2 bytes.

Error: offset exceeds 1024

While creating a custom image format, you cannot set a file offset to a number greater than 1024.

Macro terminated at line xxx

A fatal error occurred while executing a macro. The macro was terminated at the specified line. Commands following the offending line were not executed. A fatal error can include the following:

Out of memory

File not found

No images available for the operation

Can’t create file

Critical error (e.g., disk drive door open)

File was zero length (after creating an image file, the resulting file was checked and found to be empty).

17.2 DOS/Windows-specific error messages

Fatal error, DPMI host does not support 32 bit applications

Fatal error, 80386 processor is required

Fatal error Previously installed software is neither VCPI nor DPMI compatible

Fatal error allocating DOS memory

16 bit code and data are too large

Fatal error, insufficient conventional memory

Cannot enable the A20 line, XMS memory manager required

FATAL error, XMS memory corrupted

16 bit code is too large

DPMI failed to enter protected mode

DPMI operating system error

One or more of these messages will occur if you do not have an 80386 or higher processor, or if some other program is providing DPMI services in a way that is incompatible with tnimage. This might occur if you are running some unusual type of memory manager or have an incompatible type of CPU chip. Certain hardware problems can also cause these messages. Try booting from a floppy or removing memory managers from AUTOEXEC.BAT.

Fatal error reading disk

FATAL error during virtual memory disk IO

These messages mean that something bad happened to your disk or to the swap file used for virtual memory. Try to free up more disk space and try again. This can also happen if you run tnimage DOS version from a floppy, and then remove the floppy.

Mouse required

No mouse driver was found. Run the file "MOUSE.COM" or "MOUSE.EXE" (a file that came with your mouse) and try again. This is necessary even if the mouse works in Windows.

VESA not present

VESA BIOS not found

These messages mean that no VESA driver was found. This driver is a file that should come with your super VGA card, with a name like "VESA.COM". Install the VESA driver and try again. This message will also occur if you do not have a super VGA card.

Can't set VESA mode

Unsupported VESA mode

These messages mean that even if you have a VESA-compatible card, it cannot handle the selected screen mode. See under "Command line options" above for instructions on setting other screen modes.

Unsafe/unable to set VESA mode

Your monitor reported that it is unable to handle the scan rate for the specified resolution, or the video card refused to set the video mode for some other reason.

Trident card not found

This only occurs if you specified "Trident" on the command line and no Trident card was present.

Unsafe to test for Tseng chip

This occurs if "Tseng" was specified and no Tseng Labs chip was found. Certain non-Tseng chips cause a system lock-up if tnimage tests for a Tseng chip, thus the program did not try to set the Tseng mode.

VGA mode set OK - Unrecognized SVGA chip

Tnimage found a VGA card, but it is not VESA-compatible and not a Tseng or Trident card. Probably it is not capable of super VGA modes.

Can't open graphics device

No graphics card was found. This is kind of a bad sign.

Unable to set video mode

You specified a video mode on the command line that is not supported. See "Supported video modes" above.

You either have insufficient video memory or an ET3000 chip

You need 1MB of RAM and an ET4000 chip

A Tseng Labs card was detected, but it is either too old (ET3000 chips are not supported) or does not have enough video RAM. On most cards, it is a simple matter to purchase and install an additional video RAM chip.

Error: Monochrome chip

A monochrome Tseng Labs chip was found on your video card.

You need to upgrade to a newer card.

Error: ET3000 chip

An ET3000 Tseng Labs chip was found on your video card. You need to upgrade to a newer card.

Error: S3 not currently supported, use VESA

Error: ATI chip not supported, use VESA

The “-S3” and “-ATI” options for directly programming these chips are not available yet.

****Warning: unusual value****

An unexpected value was obtained from the VESA BIOS in the video card. Tnimage may not function correctly.

Unsupported Tseng mode

A Tseng mode was specified which could not be set by the video card or is not supported by tnimage.

Unsupported Trident mode

A Trident mode was specified which could not be set by the video card or is not supported by tnimage.

Wrong switch setting on Trident video card

Check your manual to ensure your card is configured correctly

A Trident card was detected, but was configured incorrectly. Please consult your video card manual or computer dealer.

Insufficient video memory in Trident card

1MB video ram required

A Trident card was detected, but was found to have less than 1 MB of video RAM. Take the card to your dealer to have additional RAM installed.

Unable to set video mode on Trident card

(Error code=%x)

A Trident card was detected, but the video mode could not be set for an unknown reason. Consult the author or your computer dealer.

Sorry, your type of SVGA card is not supported

Tnimage does not support this type of SVGA card; or, VESA BIOS was not loaded before running tnimage. This is a file that should come with your super VGA card, with a name like “VESA.COM”. Run this program and try again. This message may also occur if the computer does not have a super VGA card.

DPMI host can’t lock error handling code!

The DOS protected mode interface supplier (i.e., Windows, 386ˆMax, etc) made an error. The simplest solution is to find the file HIMEM.SYS (which comes with DOS) and install it in your CONFIG.SYS file, e.g., add the line

`C:\DOS\HIMEM.SYS .`

Then, run the program from the DOS command line instead of within Windows.

17.3 UNIX-specific error messages

There are a huge number of these. Below are the most common.

Can't open display

X11 must be running. If running tnimage remotely, xdm must be running on the host machine. It is also necessary to have the remote computer's name in your xhosts by typing the following local command:

`xhost +` *Remote-system-name*

fcolor 0 is already used by another application

fcolor is now set to 39

bcolor 35 is already used by another application

bcolor is now set to 40

This message is harmless and means that the foreground and background drawing colors have been taken by some other program and can't be used. They have been automatically remapped.

Can't create icon

Not allowed to create a 32x32 icon!

These messages are harmless and means that your Window manager has some limitation that prevents the creation of an icon of the proper size. Iconized images may look funny.

Less than 32 free color cells available

Using modifiable colormaps

Some other application(s), or the Window manager itself, has grabbed most or all of the colors. Tnimage has automatically switched to "colormap mode" to obtain good image quality. Flashing may occur if the mouse is moved from one window to another. It is recommended to maximize the tnimage window to prevent this. Alternatively, exit tnimage and close Netscape or other imaging programs and restart tnimage.

This message occurs frequently in Solaris, because of the Window Manager.

No visuals found

No appropriate visual

X11 is running in an unsupported screen mode.

Error allocating size hints

Error allocating class hint

Error creating XTextProperty

Error allocating Window manager hints

XCreateImage failed

error in dialogbox

Segmentation fault

Bus Error

BadWindow

Please send a complete bug report (see Sec. 1.5) to the author if any of these messages or similar messages occur.

Can't allocate colors

Another application may be using all the available colors; or another copy of tnimage may be running. Try closing some applications.

virtual memory exceeded in new()

The image appeared to be a valid image file but was actually junk and sent impossible coordinates to `tnimage`, causing it to crash. Please send me a copy of the offending image file so future versions can check for it.

Can't resolve symbol "__register_frame_info"

The version of tnimage you are using was dynamically linked to libc5, but your system has glibc. Obtain version 3.0.0 or higher of tnimage, use the statically-linked version, or recompile tnimage on your own system (Note: to recompile tnimage, you will also need a glibc version of Motif).

8-bit TrueColor mode, yuck! `tnimage` does not run well in 8-bit TrueColor mode, which is the default for VNC. It is recommended to start `vncserver` with the "`-depth 24` " option.

17.4 Trademark disclaimers and acknowledgments

Graphics Interchange format and GIF (SM) are service marks of Compuserve Incorporated.

The JPEG reading/writing routines in this program utilize the work of the Independent JPEG Group, which includes Tom Lane, Philip Gladstone, Luis Ortiz, Jim Boucher, Lee Crocker, Julian Minguillon, George Phillips, Davide Rossi, Ge' Weijers, and others.

The HDF plugin was compiled using HDF libraries developed at the National Center for Superconducting Applications at the University of Illinois at Urbana-Champaign.

Windows and UNIX are trademarks of Microsoft Corporation and X/Open Company, respectively.

TIFF reading/writing routines are based on information provided by Aldus Corporation [16].

PostScript is a trademark of Adobe Systems, Inc. The PostScript writing routine is based on information provided in the PostScript Language Manual [17].

Targa and Windows BMP reading/writing routines were adapted from the public domain work of James D. Murray, Anaheim, CA, USA.

Thanks to Joe Huffman of FlashTek, Inc. for valuable advice in executing real-mode calls in the x32VM Dos Extender used in the MS-DOS version of this program.

Thanks to Dr. C. Segebade of the Bundesanstalt für Materialforschung und -prüfing, Berlin, Germany for numerous criticisms and suggestions for new features.

Thanks to J.M. Kendall of JPL for supplying the sample image used in the FFT filtering tutorial.

The scanner interface was based on general information from the Linux SCSI Programming HOWTO written by Heiko Eissfeldt heiko@colossus.escape.de v1.4, 14 June 1995, and on information generously provided by Hewlett-Packard Corp [18].

The GIF reading/writing routines are loosely based on 'savegif.c' by Roger T. Stevens (12-31-91).

Thanks to Umberto D'Ortona (umberto@ensic.u-nancy.fr) for contributing a fix and a sample image to allow reading of grayscale JPEGs and John Walker for contributing a sample ImageQuant file for testing.

Thanks to Hans Schwengeler (schweng@astro.unibas.ch) for porting the program to the Digital Tru64 Unix (OSF1) V4.0E with the native Digital C++ compiler. (He reports an 11.5× speedup over gcc).

Thanks to Michael Lapsley for contributing a camera config file and patch for the bttv driver for the Hauppauge WinTV camera card.

Thanks to Levente Novák (novak@jaguar.dote.hu) for supplying a sample AFM image.

Thanks to Dietmar Kunz (dkunz@server2.fo.FH-Koeln.DE) for writing the Laplacian wavelet transform, improving the pyramidal wavelet transform, and making several helpful bugfixes and suggestions.

Thanks to Kurt De Vos (kd269@columbia.edu) for information on compiling `tnimage` in Mac OSX.

Many other users, too numerous to list here, contributed helpful suggestions and bug reports.

This software was developed as an independent, unfunded project using privately-owned equipment, and has no association with any government or private organization.

Bibliography

[1] H. Noji, R. Yasuda, M. Yoshida, and K. Kinosita, Jr. Direct observation of the rotation of F1-ATPase. *Nature*, 386(6622):299–302, 1997.

[2] L. Kaufman and P.J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis.* Wiley-Interscience, New York, 1990.

[3] J.C. Bezdek. *Pattern recognition with fuzzy objective function algorithms.* Plenum, New York, 1981.

[4] J.C. Bezdek. Cluster validity with fuzzy sets. *J. Cybern.*, 3:58, 1971.

[5] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *J. Cybern.*, 3:32, 1974.

[6] T.J. Nelson. A neural network model for cognitive activity. *Biol. Cybern.*, 49(2):79–88, 1983.

[7] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[8] J. L. Starck, F. Murtagh, and A. Bijaoui. *Image processing and data analysis.* Cambridge Univ. Press, Cambridge, 1998.

[9] L. Prasad and S. S. Iyengar. *Wavelet analysis with applications to image processing.* CRC Press, Boca Raton, 1997.

[10] D.M. Monro, B. E. Bassil, and G. J. Dickson. Orthonormal wavelets with balanced uncertainty. *IEEE International Conference on Image Processing*, 2:581–584, 1996.

[11] D.M. Monro and B. G. Sherlock. Space-frequency balance in biorthogonal wavelets. *IEEE International Conference on Image Processing*, 1:624–627, 1997.

[12] Y. Meyer, S. Quake, and V. Wickerhauser. Entropy based algorithms for best basis selections. *IEEE Trans. on Information Theory*, 38(2):713–718, 1992.

[13] I. Daubechies. *Ten lectures on wavelets (Cbms-Nsf Regional Conference Series in Applied Mathematics, No 61).* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1992.

[14] J.D. Villasenor, B. Belzer, and J. Liao. Wavelet filter evaluation for image compression. *IEEE Trans. on Image Proc.*, 1995.

[15] J. N. Bradley, C. M. Brislawn, and T. Hopper. The FBI wavelet/scalar quantization standard for gray-scale fingerprint image compression. *SPIE Proceedings, Visual Information Processing II*, 1961:293–304, 1993.

[16] Aldus Developers Desk. *TIFF Revision 6.0 Final Q.* Aldus Corporation, Seattle, WA, 1992.

[17] Adobe Systems Inc. *PostScript Language Reference Manual, 2nd Edition.* Addison Wesley, Reading, MA, 1990.

[18] Hewlett-Packard Corp. *Scanjet scanner control language toolkit version 7.0.* Hewlett-Packard Co., Greeley, CO, 1996.

Index