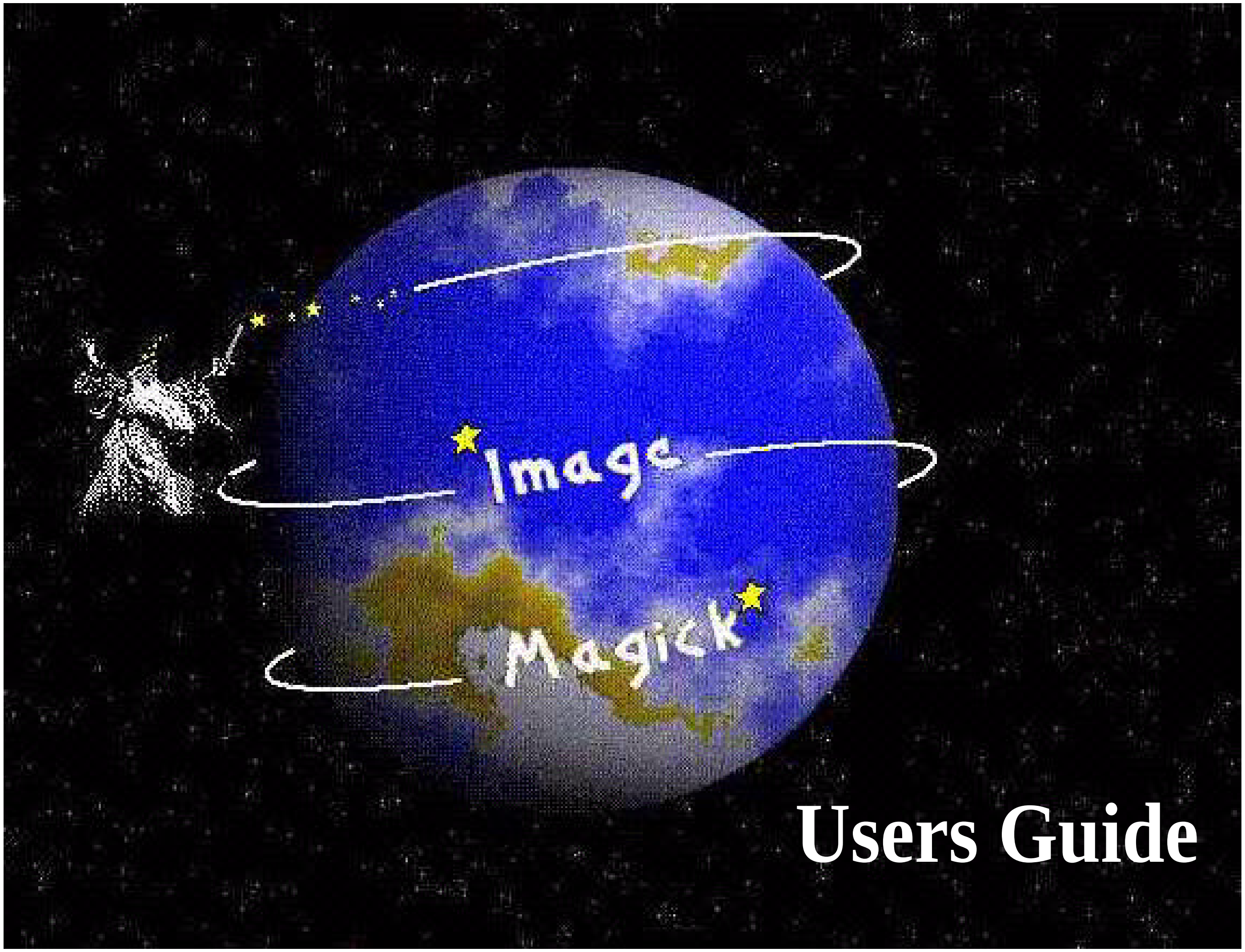


Image
Magick
Users Guide

Table of Contents

Chapter 1

Welcome to ImageMagick

Overview

ImageMagick is a robust collection of tools and libraries to read, write, and manipulate an image in any of the more popular image formats including GIF, JPEG, PNG, PDF, and Photo CD. With ImageMagick you can create GIFs dynamically making it suitable for Web applications.

ImageMagick can read and write over sixty of the more popular image formats including JPEG, TIFF, PNM, GIF, Photo CD, and PostScript. ImageMagick lets you interactively resize, rotate, sharpen, color reduce, and add special effects to an image, and save your completed work in the same or a different image format.

While ImageMagick has a simple point-and-click interface, its power lies in its command line abilities. Today's popular image manipulation software packages require you to work with individual images. With ImageMagick, you can manipulate entire directories of images with one simple script. For example, on Unix, you can convert all your JPEG images to GIF with this C-shell script:

```
foreach file (*.jpg)
    convert $file $file:r.gif
end
```

ImageMagick lets you perform any of the following functions:

- convert from one image format to another (e.g. TIFF to JPEG)
- resize, rotate, sharpen, color reduce, and add special effects to an image

- create a framed thumbnail of an image
- create a transparent image for use on the World Wide Web
- create a GIF animation sequence from a group of images
- combine several images to create a composite image
- draw shapes or text on an image
- describe the format and characteristics of an image
- decorate an image with a border or frame

ImageMagick is written in the portable C programming language and interfaces with the X11 Window library. It will compile with any modern C compiler—no proprietary toolkits are required!

ImageMagick’s Core Features

ImageMagick’s core features include the following.

Display. Display is a machine architecture-independent image and display program. It can display an image on any workstation display running an X server.

For detailed information, see Chapter4, Display.

Import. Import reads an image from any visible window on an X server and outputs it as an image file. You can capture a single window, the entire screen, or any rectangular portion of the screen. You can use Display for redisplay, printing, editing, formatting, archiving, and image processing of the captured image.

For detailed information, see Chapter5, Import.

Animate. Animate displays a sequence of images on any workstation display running an X server. Animate first determines the hardware capabilities of the workstation. If the number of unique colors in an image is fewer than or equal to the number the workstation can support, the image is displayed in an X window. Otherwise the number of colors in the image is first reduced to match the color resolution of the workstation.

In other words, a continuous-tone 24-bit image can display on an 8-bit pseudo-color device or monochrome device. In most instances the reduced color image closely resembles the original. In turn, a monochrome or pseudo-color image sequence can display on a continuous-tone 24-bit device.

For detailed information, see Chapter6, Animate.

Montage. Montage creates a composite image by combining several separate images. The images are tiled on the composite image with the name of the image optionally appearing just below the individual tile.

For detailed information, see Chapter7, Montage.

Convert. Convert converts an input file in one format to an output file in another format. By default, the image format is determined by its magic number. To specify a particular image format, you can precede the filename with an image format name and a colon (e.g., ps:image) or specify the image type as the filename suffix (e.g., image.ps). For detailed information, see Chapter8, Convert.

Mogrify. Mogrify transforms an image or a sequence of images. These transformations include image scaling, image rotation, color reduction, and others. The transmogrified image overwrites the original image.

For detailed information, see Chapter9, Mogrify.

Identify. Identify describes the format and characteristics of one or more image files. It also reports if an image is incomplete or corrupt. The information displayed includes the scene number, file name, width and height of the image, whether the image is colormapped, the number of colors in the image, the number of bytes in the image, its format (i.e., jpeg, pnm, etc.), and finally the number of seconds it takes to read and process the image.

For detailed information, see Chapter10, Identify.

Combine. Combine combines images to create new images.

For detailed information, see Chapter11, Combine.

PerlMagick. PerlMagick is an objected-oriented Perl interface to ImageMagick. You can use it to read, manipulate, or write an image or image sequence from within a Perl script. This makes it very suitable for web CGI scripts. For examples of what you can do with PerlMagick, see http://www.sympatico.org/cristy/MogrifyMagick/scripts/MogrifyMagick.cgi.

For detailed information, see Chapter12, PerlMagick.

ImageMagick Studio

You can visit the ImageMagick Studio web site at http://www.sympatico.org/cristy/MogrifyMagick/scripts/MogrifyMagick.cgi/ to try out any of the ImageMagick functions. A sample image is just a click away.

It's Free

ImageMagick is free! You can do anything with the software you want, including selling it. The software *is* copyrighted, however, you can redistribute it without fee. For detailed information, see the copyright notice at the beginning of the guide.

Chapter 2

Installing ImageMagick

web page
www.wizards.dupont.com

Getting ImageMagick

You can download ImageMagick from ftp://ftp.wizards.dupont.com/pub/ImageMagick. ImageMagick client exectuables are available for some platforms. Macintosh, NT, VMS, and Linux source and binaries are also available.

External Image Viewer

To use *Display* as your external image viewer, edit the global mailcap file or your personal mailcap file—.mailcap located in your home directory—and add this entry:

```
image/*; display %s
```

Mailing List

There is a mailing list for discussions and bug reports about ImageMagick. To subscribe send the message

```
subscribe magick
```

to *majordomo@wizards.dupont.com*. You'll receive a welcome message telling you how to post messages to the list *magick@wizards.dupont.com*.

Memory Requirements

You should allocate sufficient swap space on your system before running ImageMagick; otherwise, you may experience random server or application crashes. Anything less than 80 MB of swap space is likely to cause random crashes.

On many systems, you will find that 80 MB is insufficient and you'll have to allocate more swap space. You should also have at least 32 MB of real memory although 64 MB or more is recommended.

Unix Compilation

Type

```
gunzip ImageMagick-5.1.0.tar.gz
tar xvf ImageMagick-5.1.0.tar
cd ImageMagick
```

Note: If you don't have gunzip, you can download it from ftp://ftp.gnu.org/pub/gnu.

Creating makefiles

There are currently two mechanisms for creating makefiles to build ImageMagick: GNU Configure (see GNU Configure) and X11 Imake (see X11 Imake).

GNU Configure

GNU Configure is easiest to use and is recommended when you want to install ImageMagick outside of the X11 distribution or working imake configuration files are not available. Using *configure* enables automated configuration, building, and installation of PerlMagick. If you're willing to accept configure's default options, type

```
./configure
```

Watch the *configure* script output to verify that it finds everything you think it should. If it doesn't, adjust your environment so it does.

If you're unhappy with *configure's* choice of compiler, compilation flags, or libraries, you can give *configure* initial values for variables by setting them in the environment. Using a Bourne-compatible shell, you can do that on the command line like this

```
CC=c89 CFLAGS=-O2 LIBS=-lposix ./configure
```

Or on systems that have the env program, you can do it like this

```
env CPPFLAGS=-I/usr/local/include LDFLAGS=-s ./configure
```

The configure variables you should be aware of are

Configure Environment Variables

Variable	Definition
CC	Name of C compiler (e.g., `'cc -Xa'`) to use

Configure Environment Variables

Variable (Cont.)	Definition
CFLAGS	Compiler flags (e.g., `'-g -O2'`) to compile with
CPPFLAGS	Include paths (`-I/somedir)` to look for header files
LDFLAGS	Library paths (`-L/somedir`) to look for libraries **Note:** Systems that support the notion of a library run-path may additionally require `-R/somedir` or `'-rpath /somedir'` in order to find shared libraries at run time.
LIBS	Extra libraries (`-lsomelib`) required to link

You must specify an absolute path rather than a relative path for any variable that requires a directory path (e.g., CPPFLAGS or LDFLAGS).

By default, *make install* will install the package's files in /usr/local/bin, /usr/local/man, etc. You can specify an installation prefix other than /usr/local by giving *configure* the option `--prefix=PATH`.

Configure can usually find the X include and library files automatically, but if it doesn't, you can use the configure options `--x-includes=DIR` and `--x-libraries=DIR` to specify their locations.

The *configure* script provides a number of ImageMagick-specific options. When you disable an option,

- `--disable-something` is the same as `--enable-something=no`

- `--without-something` is the same as `--with-something=no`

The *configure* options are as follows (execute `configure --help` to see all options).

Configure Options

This…	Does this
`--enable-16bit-pixel`	enables 16 bit pixels (default is no)
`--enable-gcov`	enables gcov source profiling support (default is no)
`--enable-gprof`	enables gprof source profiling support (default is no)
`--enable-lzw`	enables LZW support (default is no)
`--enable-prof`	enables prof source profiling support (default is no)
`--enable-sfio`	enable sfio-based stdio support (default is no)
`--enable-shared`	builds shared libraries (default is no)
`--enable-static`	builds static libraries (default is yes)
`--enable-socks`	enables use of SOCKS v5 library and 'rftp'
`--enable-socks`	enables SOCKS v5 proxy support (default is no)
`--with-bzlib`	enables BZLIB (default is yes)
`--with-dmalloc`	use dmalloc, as in ftp://ftp.letters.com/src/dmalloc/dmalloc.tar.gz

Configure Options

This… (Cont.)	Does this
`--with-dps`	enables Display Postscript (default is yes)
`--with-fpx`	enables FlashPIX (default is yes)
`--with-frozenpaths`	*enables frozen delegate paths (default is yes)*
`--with-hdf`	enables HDF (default is yes)
`--with-jbig`	enables JBIG (default is yes)
`--with-jpeg`	enables JPEG (default is yes)
`--with-perl`	enables build/install of PerlMagick (default is no)
`--with-png`	enables PNG (default is yes)
`--with-tiff`	enables TIFF (default is yes)
`--with-ttf`	enables TrueType (default is yes)
`--with-x`	uses the X Window System
`--with-zlib`	enables Zlib (default is yes)

ImageMagick options represent one of the following:

- features to be enabled

- packages to be included in the build

When you enable a feature (via `--enable-something`), it enables code already present in ImageMagick. When you enable a package (via `--with-something`), the *configure* script will search for it. If it's properly installed and ready to use (i.e., headers and built libraries are found by the compiler) it will be included in the build.

Note: The *configure* script is delivered with all features disabled and all packages enabled. In general, the only reason to disable a package is if a package exists but it is unsuitable for the build—perhaps it's an old version or it's compiled with the wrong compilation flags.

Special Configure Options Considerations

`--disable-shared`

- The shared libraries are not built. Shared libraries are valuable because they are shared across more than one invocation of an ImageMagick or PerlMagick client. In addition, the clients take much less disk space and shared libraries are required in order for PERL to dynamically load the PerlMagick extension.
- ImageMagick built with plug-ins (see Delegates below) can pose the following additional challenges:
    - You can build all the plug-ins statically and link them into the ImageMagick shared library (i.e., libMagick.so) or
    - you can build the plug-ins as shared libraries. (**Note:** Some systems already have plug-ins installed as shared libraries.)

- Shared library's compilation flags differ from vendor to vendor (gcc's is -fPIC). However, you must compile all shared library source with the same flag. (**Note:** For gcc use `-fPIC` rather than `-fpic`.)

`--disable-static`

- Static archive libraries (with extension .a) are not built. If you are building shared libraries, there is little value to building static libraries. Reasons to build static libraries include:
  - they can be easier to debug
  - the clients do not have external dependencies (i.e., libMagick.so)
  - building PIC versions of the plug-in libraries may take additional expertise and effort
  - you are unable to build shared libraries

`--with-perl`

- Conveniently compile and install PerlMagick in one step. Without this option you must first install ImageMagick, change to the PerlMagick subdirectory, build, and finally, install PerlMagick.

  Note: PerlMagick is configured even if you don't specify `--with-perl`. If you don't specify `--enable-shared`, a new PERL interpreter (i.e., PerlMagick) is built and statically linked against the PerlMagick extension. This new interpreter is installed alongside your existing PERL interpreter. If you specify `--enable-shared`, the PerlMagick extension is built as a dynamically loadable object that's loaded into your current PERL interpreter at run-time. Use of dynamically-loaded extensions is preferable over statically linked extensions so `--enable-shared` should be specified if possible. If the argument `--with-perl=/path/to/perl` is supplied, then */path/to/perl* is taken as the PERL interpreter to use.

`--with-x=no`

- Build and use the X11 stubs library (i.e., ImageMagick/xlib) instead of the core X11 libraries. This may be necessary on systems where X11 is not installed (e.g., a web server).

  Note: *Display*, *animate*, and *import* will not work with this library. The remaining programs have reduced functionality.

Dealing with Configuration Failures

While configure is designed to ease the installation of ImageMagick, it often discovers problems that would otherwise be encountered later when you compile ImageMagick. The *configure* script tests for headers and libraries by executing the compiler (CC) with the specified compilation flags (CFLAGS), pre-processor flags (CPPFLAGS), and linker flags (LDFLAGS). Any errors are logged to the file config.log. If configure fails to discover a header or library, review the log file to determine why. After you correct the problem, be sure to remove the 'config.cache' file before you run *configure* so it will re-inspect the environment rather than using the cached values.

Common causes of configuration failures are

- a plug-in header is not in the header include path (`CPPFLAGS -I` option)
- a plug-in library is not in the linker search/run path (`LDFLAGS -L/-R`option)

- a plug-in library is missing a function (old version?)
- the compilation environment is faulty

Reporting Bugs

If you've tried all reasonable corrective actions and the problem appears to be due to a flaw in the *configure* script, email a bug report to the *configure* script maintainer at *bfriesen@simple.dallas.tx.us*.

Bug reports should contain the following:

- operating system type (as reported by `'uname -a'`)
- the compiler/compiler-version

A copy of the *configure* script output and/or the config.log file may be valuable in order to find the problem.

X11 Imake

Use this option if working imake configuration files are available and you don't mind editing a configuration file. Install the package using the imake default installation directory (i.e., usually the X11 distribution directory). Use of this scheme requires a separate step to install PerlMagick. See the ReadMe file in the PerlMagick subdirectory.

Review the defines in magick/magick.h and magick/delegates.h and make sure they meet the requirements of your local system.

Edit magick.tmpl and set the variables to suit your local environment.

Now type

```
xmkmf
make Makefiles
```

or just

```
xmkmf -a
```

Using X11R6 Imake

ImageMagick requires an ANSI compiler. If the compile fails, first check to ensure your compile is ANSI compatible. If it fails for some other reason, try

```
cd magick
make -k
cd ..
make -k
```

To confirm your build of the ImageMagick distribution was successful, type

```
display
```

If the program faults, verify you didn't inadvertently link to an older version of the libMagick library. In this case type

```
cd ImageMagick/magick
make install
```

```
cd ..
make
```

If the image colors are not correct use

```
display -visual default
```

You can find other sample images in the images directory.

For additional information, see the following ImageMagick chapters.

- Chapter4, Display
- Chapter8, Convert
- Chapter7, Montage
- Chapter10, Identify
- Chapter6, Animate
- Chapter5, Import
- Chapter9, Mogrify
- Chapter11, Combine

Also read the ImageMagick Frequently Asked Questions web page at http://www.wizards.dupont.com/cristy/www/Magick.html. This is “required reading.” Most ImageMagick questions received via email are answered in this document.

Place display X application defaults in /usr/lib/X11/app-defaults/Display. Use the appropriate name for other clients (e.g., Animate, Montage, etc). To execute display as a menu item of any window manager (e.g., olwm, mwm, twm, etc), use

```
display logo:Untitled
```

Delegates

To further enhance the capabilities of ImageMagick, you may want to get the following programs or libraries. Many of these delegates can be found at ftp://ftp.wizards.dupont.com/pub/ImageMagick/delegates.

Background Texture

ImageMagick requires a background texture for the *Tile* format and for the `-texture` option of *Montage*. You can use your own or get samples from KPT.

RALCGM

ImageMagick requires ralcgm to read Computer Graphic Metafile images (may not compile under linux). You also need Ghostscript (see below).

TransFig

ImageMagick requires fig2dev to read TransFig images.

GET

ImageMagick requires Get to read images specified with a world wide web (WWW) uniform resource locator (URL). Get must be in /usr/local/bin. See WWW command in magick/magick.h to change its location.

Note: Don't confuse this Get program with the SCCS Get program. If you don't have an http server, you can use xtp, available in the ImageMagick distribution, for URLs whose protocol is ftp.

FPX

ImageMagick requires the FlashPix SDK to read and write the FPX image format.

FreeType

ImageMagick requires the FreeType software, version 1.1 or later, to annotate with TrueType fonts.

HDF

ImageMagick requires the NCSA HDF library to read and write the HDF image format.

HTML2PS

ImageMagick requires HTML2PS to read HyperText Markup Language (HTML) documents.

JBIG

ImageMagick requires the JBIG-Kit software to read and write the JBIG image format.

JPEG

ImageMagick requires the Independent JPEG Group’s software to read and write the JPEG image format.

Iterative JPEG Compression

See Kinoshita and Yamamuro, *Journal of Imaging Science and Technology, Image Quality with Reiterative JPEG Compression*, Volume 39, Number 4, July 1995, 306–312, who claim that

- the iterative factor of the repetitive JPEG operation had no influence on image quality, and
- the first compression determined base image quality.

MPEG

ImageMagick requires the MPEG encoder/decoder to read or write the MPEG image format.

PNG

ImageMagick requires the PNG library to read the PNG image format.

PostScript

ImageMagick requires Ghostscript software to read PostScript (PS) and Portable Document Format (PDF) images. It is used to annotate an image when an X server is not available. See FreeType, above for another means to annotate an image.

Note: Ghostscript must support the ppmraw device (type `gs -h` to verify). If Ghostscript is unavailable, the Display Postscript extension is used to rasterize a Postscript document (assuming you define HasDPS). The DPS extension is less robust than Ghostscript in that it will only rasterize one page of a multi-page document.

RA_PPM

ImageMagick requires ra_ppm from Greg Ward's Radiance software to read the Radiance image format (which may not compile under Linux).

RAWTORLE

ImageMagick requires rawtorle from the Utah Raster Toolkit to write the RLE image format (which may not compile under Linux).

SANE

ImageMagick requires scanimage to import images from a scanner device.

TIFF

ImageMagick requires Sam Leffler's TIFF software to read and write the TIFF image format. It optionally requires the JPEG and ZLIB libraries.

WMF

ImageMagick requires wmftogif to read Windows Meta File images.

ZLIB

ImageMagick requires the ZLIB library to read the PNG image format or read or write ZLIB compressed MIFF images.

Compiling ImageMagick

The following procedure describes how to build ImageMagick extension libraries in subdirectories of the ImageMagick directory. An alternative to these procedures is to install one or more of these under your system's regular include/lib directory (e.g., the directory specified by `--prefix` to *configure* or /usr/local). This allows the libraries to be shared by other packages. When you use the *configure* script, the two schemes may be mixed.

Also, please note that when the *configure* option `--enable-shared` is enabled, these procedures must be supplemented with the compilation flags that are required on your system to generate PIC code. In the case of gcc, this usually means that -fPIC must be added to the compiler options (i.e., CFLAGS) when you build each plug-in library.

To display images in the HDF, JPEG, MPEG, PNG, TIFF or TTF format, get the appropriate archives and build ImageMagick as follows:

HDF

```
cd ImageMagick
unzip -c HDF4.2r2.tar.gz | tar xvf -
mv HDF4.2r2 hdf
cd hdf
configure
make -k hdf-libnofortran
cd ..
```

JBIG

```
cd ImageMagick
unzip -c jbigkit-1.0.tar.gz | tar xvof -
mv jbig-kit jbig
cd jbig
make
cd ..
```

JPEG

```
cd ImageMagick
gunzip -c jpegsrc.v6b.tar.gz | tar xvof -
mv jpeg-6b jpeg
cd jpeg
configure
make
cd ..
```

PNG

```
cd ImageMagick
unzip -c libpng-1.0.3.tar.gz | tar xvf -
mv libpng-1.0.3 png
cd png
make
cd ..
```

TIFF

```
cd ImageMagick
gunzip -c tiff-v3.4beta037.tar.Z | tar xvof -
mv tiff-v3.4beta037 tiff
cd tiff
./configure
make
cd ..
```

TTF

```
cd ImageMagick
gunzip -c freetype-1.2.tar.gz | tar xvof -
mv freetype-1.2 ttf
cd ttf
./configure -disable-shared
make
cd ..
```

ZLIB

```
cd ImageMagick
gunzip -c zlib-1.1.3.tar.gz | tar xvf -
mv zlib-1.1.3.tar.gz zlib
cd zlib
make
cd ..
```

Support for Shared Libraries

If your computer system supports shared libraries you must type

```
make install
```

Finally, perform the following:

```
cd ImageMagick
```

```
edit Magick.tmpl and define Has???? as instructed
xmkmf
make Makefiles
make clean
make
```

If you prefer to use GNU Configure rather than Imake, type

```
configure
make clean
make -k
```

If the compile fails due to a function redefinition it may be that either jpeg/jconfig.h or mpeg/mpeg.h is redefining const. Fix this problem and try again.

You can now convert or display images in the JPEG, TIFF, PNG, etc. image formats.

If you have HDF, JBIG, JPEG, MPEG, PNG, and TIFF sources installed as directed above, you can also type

```
Install sun
```

Substitute the appropriate machine type (i.e., aix, hpux, sgi, etc.).

VMS Compilation

You might want to check the values of certain program definitions before you compile. Verify the defininitions in delegates.mgk suit your local requirements. Next, type

```
@make
set display/create/node=node_name::
```

where node_name is the DECNET X server to contact.

Finally type

```
display
```

Alternatively, download a zipped distribution (with JPEG, PNG, TIFF, and TTF) from ftp://ftp.wizards.dupont.com/pub/ImageMagick/vms.

The VMS JPEG, PNG, and TIFF source libraries are available from axp.psl.ku.dk in [anonymous.decwindows.lib].

Thanks to *pmoreau@cenaath.cena.dgac.fr* for supplying invaluable help as well as the VMS versions of the JPEG, PNG, TIFF, and TTF libraries.

NT Compilation

The Visual C++ distribution targeted at Windows NT or Windows 95 is provided in the "VisualMagick" subdirectory of the distribution. There are two workspaces (DSW files) that can be used to do the complete build:

```
VisualMagick.dsw

VisualMagickStatic.dsw
```

Use the first to build DLL's otherwise it builds a static version. A complete build can be accomplished by simply doing:

```
Build: Batch Build: Clean

Build: Batch Build: Build
```

The *Clean* step is needed in order to make sure that all of the target support libraries are updated with any patches needed to get them to compile properly on Windows.

All of the required files that are needed to run any of the command line tools will be found in the "bin" subdirectory of the VisualMagick subdirectory. This includes EXE, and DLL files. You should be able to test the build directly from this directory without having to move anything to any of the global SYSTEM or SYSTEM32 areas in the operating system installation.

NOTE: The two utilities *display* and *animate* will compile and link but not function in the default build environment. This is due to the fact that the default build environment uses the X11 stubs to supply non-functional stubs for X-Window functionality. This is due to the lack of a high need for this on a Windows NT or Win95 only system. Work is underway to add X11 libraries to the standard distribution in the neat future.

To view any image in a Microsoft window, type

```
convert image.ext win:
```

Import works if you have at least one X window open. Alternatively, type

```
convert x:root image.gif
```

Make sure gswin32 (Ghostscript) is in your execution path (see Autoexec.bat), otherwise, you will be unable to convert or view a Postscript document.

Make sure iexplore (Internet Explorer) is in your execution path (see Autoexec.bat), otherwise, you will be unable to browse the ImageMagick documentation.

To compile the source with Codewarrior, start with Magick/Magick.mcp, then animate.mcp, convert.mcp, etc. The Visual C++ workspace is ImageMagick.dsw.

Tip! The NT executables will work under Windows 95 and Windows 98.

Macintosh Compilation

The Macintosh Macintosh distribution contains MetroWerks Codewarrior Professional projects for compilation. For those who do not have access to CodeWarrior, the binaries for the command line utilities are enclosed.

Note: The inline intrinsic functions are commented in math.h in order to compile.

Note: *Display*, *animate*, and *import* currently do not work on the Macintosh.

Animation

To prevent color flashing on visuals that have colormaps, *animate* creates a single colormap from the image sequence. This can be rather time consuming. You can speed up this operation by reducing the colors in an image before you animate it. Use *mogrify* to color reduce images.

```
mogrify +map -colors 256 scenes/dna.[0-9]*
```

Alternatively, you can use a Standard Colormap, or a static, direct, or true color visual. You can define a Standard Colormap with *xstdcmap*. For example, to use the “best” Standard Colormap, type

```
xstdcmap -best
animate -map best scenes/dna.[0-9]*
```

or to use a true color visual

```
animate -visual truecolor scenes/dna.[0-9]*
```

Image filenames can appear in any order on the command line if the scene keyword is specified in the MIFF image. Otherwise the images display in the order they appear on the command line. A scene is specified when converting from another image format to MIFF by using the “scene” option with any filter. Be sure to choose a scene number other than zero. For example, to convert a TIFF image to a MIFF image as scene #2, type

```
convert -scene 2 image.tiff image.miff
```

16-bit Imaging

By default, ImageMagick uses a color depth of 8 bits (e.g., [0..255] for each of red, green, blue, and transparency components). Any 16-bit image is scaled immediately to 8-bits before any image viewing or processing occurs. If you want to work directly with 16-bit images (e.g., [0..65535]), edit Magick.tmpl and define *QuantumLeap* or use

```
-enable-16bit
```

with configure.

Next, type

```
make clean
make
```

In 16-bit mode expect to use about 33% more memory on the average. Also expect some processing to be slower than in 8-bit mode (e.g., Oil Painting, Segment, etc.).

In general, 16-bit mode is useful only if you have 16-bit images you want to manipulate, then save the transformed image back to a 16-bit image format (e.g., PNG, VIFF).

64-bit Machines

Each pixel, within ImageMagick, is represented by the PixelPacket structure found in magick/image.h. Only 8 bits are required for each color component and 16 bits for the colormap index for a total of 6 bytes. If *QuantumLeap* is defined (see 16-bit Imaging), the color component size increases to 16 bits for a total of 10 bytes. Some 64-bit machines pad the structure, which can cause a significant waste of memory. For the cray, change the RunlengthPacket structure to the following before you compile.

```
typedef struct _PixelPacket
{
  unsigned char
    red : QuantumDepth,
    green : QuantumDepth,
    blue : QuantumDepth,
    opacity : QuantumDepth;

  unsigned short
    index : 16;
} PixelPacket;
```

Note: This may not work on other 64-bit machines that pad. The Dec Alpha, Solaris, and Irix apparently do not pad the structure so ImageMagick should be fine on this particular 64-bit machine.

MIFF Image Format

MIFF is an image format that

- is machine independent. It can be read on virtually any computer. No byte swapping is necessary.
- has a text header. Most image formats are coded in binary and you cannot easily tell attributes about the image. Use *more* on MIFF image files and the attributes are displayed in text form.
- can handle runlength-encoded images. Although most scanned images do not benefit from runlength-encoding, most computer-generated images do. Images of mostly uniform colors have a high compression ratio and therefore take up less memory and disk space.
- allows a scene number to be specified. This allows you to specify an animation sequence out-of-order on the command line. The correct order is determined by the scene number of each image.
- computes a digital signature for images. This is useful for comparing images. If two image files have the same signature, they are identical images.

There is a *montage* keyword that allows an image to act as a visual image directory. See Chapter4, Display for details.

To get an image into MIFF format, use *convert* or read it from an X window using the import program.

Alternatively, type the necessary header information in a file with a text editor. Next, dump the binary bytes into another file. Finally, type

```
cat header binary_image | display -write image.miff -
```

For example, suppose you have a raw red, green, blue image file on disk that is 640 by 480. The header file would look like this

```
id=ImageMagick columns=640 rows=480 :
```

The image file would have red, green, blue tuples (rgbrgbrgb...). See Chapter4, Display for details.

Chapter 3

The ImageMagick Interface

Overview

Several components—use of options, the Command Widget, using the mouse, and the ImageMagick environment—are common to all areas of ImageMagick. They're described in this chapter.

Using Options

Options are processed in command-line order. Any option you specify on the command line remains in effect until you change it.

By default, the image format is determined by its magic number. To specify a particular image format, precede the filename with an image format name and a colon, for example,

```
ps:image
```

or specify the image type as the filename suffix

```
image.ps
```

See Appendix A, Supported Image Formats for a list of valid image formats.

When you specify X as your image type, the filename has special meaning. It specifies an X window by ID, name, or root. If you specify no filename, you can select the window by clicking the mouse in it.

Specify the image filename as **-** for standard input or standard output. If the filename has the extension .Z or .gz, the file is uncompressed with uncompress or gunzip, respectively. If it has the extension .Z or .gz, the file size is compressed using with compress or gzip, respectively. Finally, precede the image file name with **|** to pipe to or from a system command.

Use an optional index enclosed in brackets after a file name to specify a desired subimage of a multiresolution image format like Photo CD, for example,

```
img0001.pcd[4]
```

or a range for MPEG images, for example,

```
video.mpg[50-75]
```

A subimage specification can be disjoint, for example,

```
image.tiff[2,7,4]
```

For raw images, specify a subimage with a geometry, for example

```
-size 640x512 image.rgb[320x256+50+50]
```

Using Filenames

Single images are read with the filename you specify. Alternatively, you can affect an image sequence with a single filename. Define the range of the image sequence with `-scene`. Each image in the range is read with the filename followed by a period (.) and the scene number. You can change this behavior by embedding a printf format specification in the filename. For example,

```
-scene 0-9 image%02d.miff
```

animates the files image00.miff, image01.miff, through image09.miff.

Image filenames may appear in any order on the command line if the image format is MIFF and the `-scene` keyword is specified in the image. Otherwise the images will be affected in the order you enter them on the command line. See Appendix C, MIFF.

Mouse Buttons

ImageMagick requires a three-button mouse. The effects of each mouse button for the display program are described below.

Tip! If you have a two-button mouse, the left button corresponds to button 1 and the right button corresponds to button 3. To simulate button 2, hold down the Alt key on your keyboard and press the right mouse button.

Mouse Button 1

Press button 1 to map or unmap the Command Widget. See the next section for more information about the Command Widget.

Mouse Button 2

Press button 2 and drag the mouse to define a region of an image to magnify.

Mouse Button 3

Press button 3 and drag the mouse to choose from a select set of Display commands. This button behaves differently if the image is a visual image directory. Choose a directory tile, press this button and drag the mouse to select a command from a popup menu.

Popup Menu Options

This menu item…	Does this…
Open	Displays the image represented by the tile.
Next	Returns from an image to the visual image directory, or moves to the next image.
Former	Moves to the previous image.

Popup Menu Options

This menu item… (Cont.)	Does this…
Delete	Deletes an image tile.
Update	Synchronizes all image tiles with their respective images.

Command Widget

The Command Widget has a number of menu commands. Those menu commands followed by a right-pointing triangle have submenu commands. The animate program, for example, has this menu of commands:

Note: Menu commands are indicated in the following list with a bullet (n). Submenu commands are indicated with a > character.

- Animate

  > Open

  > Play

  > Step

  > Repeat

  > Auto Reverse

- Speed
  > Faster
  > Slower
- Direction
  > Forward
  > Reverse
- Image Info
- Help
- Quit

Selecting a Submenu Command

1 To select a submenu command, move the pointer to the appropriate menu.

2 Press the mouse button and hold it down as you drag through the menu to a command, then its submenu command.

3 Release the mouse button to execute the submenu command under the pointer.

Note: If you decide not to execute a command, drag the pointer away from the menu.

Keyboard Short Cuts

The following table shows keyboard short cuts you can use with the animate program.

Keyboard Short Cuts

Press this…	to do this…
Ctl+o	load an image from a file
space	display the next image in the sequence
<	speed up the display of the images (See -delay for more information.)
>	slow the display of the images (See -delay for more information.)
?	display information about the image; press any key or button to erase the information; the following information is printed: image name, image size, the total number of unique colors in the image
F1	display helpful information about an ImageMagick tool
Ctl+q	discard all images and exit ImageMagick

Environment

DISPLAY

Lets you get the default host, display number, and screen.

Chapter 4

Display

Overview

Display is an image processing and display program. It can display an image on any workstation screen running an X server. Display can read and write many of the more popular image formats—JPEG, TIFF, PNM, Photo CD, to name a few.

With display you can do the following with an image:

- load an image from a file
- display the next or previous image
- display a sequence of images as a slide show
- write an image to a file
- print an image to a PostScript printer
- delete an image file
- create a visual image directory
- select an image to display by its thumbnail rather than its name
- undo last image transformation
- copy and paste a region of an image

- refresh an image
- restore an image to its original size
- decrease an image's size by half
- double an image's size
- resize an image
- crop an image
- cut an image
- flop an image in the horizontal direction
- flip an image in the vertical direction
- rotate an image 90 degrees clockwise
- rotate an image 90 degrees counter-clockwise
- rotate an image
- shear an image
- roll an image
- trim an image's edges

Overview

- invert the colors of an image
- vary an image's color brightness
- vary and image's color saturation
- vary an image's hue
- gamma correct an image
- sharpen an image's contrast
- dull an image's contrast
- perform histogram equalization on an image
- perform histogram normalization on an image
- negate an image's colors
- convert an image to grayscale
- set the maximum number of unique colors in an image
- reduce the speckles within an image
- eliminate peak noise from an image
- detect edges within an image

- emboss an image
- segment an image by color
- simulate an oil painting
- simulate a charcoal drawing
- annotate an image with text
- draw on an image
- edit an image pixel color
- edit an image's matte information
- composite an image with another
- add a border to an image
- add a border to an image
- surround image with an ornamental border
- apply image processing techniques to a region of interest
- display information about an image
- zoom a portion of an image

- show a histogram of an image
- display image to background of a window
- set user preferences
- display information about this program
- discard all images and exit program
- change the level of magnification
- display images specified by a World Wide Web (WWW) uniform resource locator (URL)

Syntax

```
display [ options ...] file [ options ...] file
```

Examples

- To scale an image of a cockatoo to exactly 640 pixels in width and 480 pixels in height and position the window at location (200,200), use

  ```
  display -geometry 640x480+200+200! cockatoo.miff
  ```

- To display an image of a cockatoo without a border centered on a backdrop, use

```
display +borderwidth -backdrop cockatoo.miff
```

- To tile a slate texture onto the root window, use

```
display -size 1280x1024 -window root slate.png
```

- To display a visual image directory of all your JPEG images, use

```
display 'vid:*.jpg'
```

- To display a MAP image that is 640 pixels in width and 480 pixels in height with 256 colors, use

```
display -size 640x480+256 cockatoo.map
```

- To display an image of a cockatoo specified with a World Wide Web (WWW) uniform resource locator (URL), use

```
display ftp://wizards.dupont.com/images/cockatoo.jpg
```

- To display histogram of an image, use

```
convert file.jpg HISTOGRAM:- | display -
```

Display Options

-backdrop

Lets you center an image on a backdrop.

This backdrop covers the entire workstation screen and is useful for hiding other X window activity while viewing the image. The color of the backdrop is specified as the background color. See Appendix B, X Resources for details.

-border *<width>x<height>*

Lets you surround an image with a colored border.

The color of the border is obtained from the X server and is defined as *borderColor* (class *BorderColor*). See the X Windows system manual at http://www.x.org for details about the specification.

-cache_threshold *value*

number of megabytes available to the pixel cache.

Image pixels are stored in memory until 80 megabytes of memory have been consumed. Subsequent pixel operations are cached on disk. Operations to memory are significantly faster but if your computer does not have a sufficient amount of free memory you may want to adjust this threshold value.

-colormap *type*

Lets you specify a type of colormap:

- Shared
- Private

This option applies only when the default X server visual is PseudoColor or GrayScale. See -visual for more details.

By default, a *Shared* colormap is allocated. The image shares colors with other X clients. Some image colors may be approximated and your image may not look the way you intended.

Choose *Private* and the image colors appear exactly as they are defined. However, other clients may go technicolor when the image colormap is installed.

-colormap *type*

Lets you specify a type of colormap:

- Shared
- Private

This option applies only when the default X server visual is PseudoColor or GrayScale. See -visual for more details.

By default, a *Shared* colormap is allocated. The image shares colors with other X clients. Some image colors may be approximated and your image may not look the way you intended.

Choose *Private* and the image colors appear exactly as they are defined. However, other clients may go technicolor when the image colormap is installed.

-colors *value*

Lets you specify the preferred number of colors in an image.

The actual number of colors in the image may be fewer than you specify, but will never be more.

Note: This is a color reduction option. Duplicate and unused colors will be removed if an image has fewer unique colors than you specify. See Appendix D, Quantize for more details. The options `-dither`, `-colorspace`, and `-treedepth` affect the color reduction algorithm.

-colorspace *value*

Lets you specify the type of colorspace.

- GRAY
- OHTA
- RGB
- Transparent

- XYZ
- YCbCr
- YIQ
- YPbPr
- YUV
- CMYK

Color reduction by default, takes place in the RGB color space. Empirical evidence suggests that distances in color spaces such as YUV or YIQ correspond to perceptual color differences more closely than distances in RGB space. These color spaces may give better results when color reducing an image. See Appendix D, Quantize for details.

Note: The transparent colorspace is unique. It preserves the matte channel of the image if it exists.

Tip! The `-colors` or `-monochrome` option is required for the transparent option to take effect.

-comment *string*

Lets you annotate an image with a comment.

By default, each image is commented with its file name. Use this option to assign a specific comment to the image.

Optionally you can include the image filename, type, width, height, or scene number in the label by embedding special format characters. The following table shows these characters and their values.

Special Format Characters

Special Character	Value
%b	file size
%d	directory
%e	filename extention
%f	filename
%h	height
%i	input filename
%l	label
%m	magick
%n	number of scenes
%o	output filename
%p	page number
%q	quantum depth

Special Format Characters

Special Character (Cont.)	Value
%s	scene number
%t	top of filename
%u	unique temporary filename
%w	width
%x	x resolution
%y	y resolution
\n	newline
\r	carriage return

For example,

```
-comment “%m:%f %wx%h”
```

produces for an image—titled bird.miff whose width is 512 and height is 480—the comment

```
MIFF:bird.miff 512x480
```

Note: If the first character of *string* is `@`, the image comment is read from a file titled by the remaining characters in the string.

-compress *type*

Lets you specify one of the following types of image compression:

- None
- Bip
- Fax
- Group 4
- JPEG
- LZW
- RunlengthEncoded
- Zip

Specify

```
+compress
```

to store the binary image in an uncompressed format. The default is the compression type of the specified image file.

-contrast

Lets you enhance or reduce the intensity differences between the lighter and darker elements of an image.

Use

```
-contrast
```

to enhance the image or

```
+contrast
```

to reduce the image contrast.

-crop *<width>x<height>{+-}<x offset>{+-}<y offset>{%}*

Lets you specify the size and location of a cropped image. See the X Windows system manual at http://www.x.org for details about the geometry specification.

To specify the width or height as a percentage, append %. For example to crop an image by 10% on all sides, use

```
-crop 10%
```

Use cropping to apply image processing options to, or display, a particular area of an image. Omit the *x offset* and *y offset* to generate one or more subimages of a uniform size.

Use cropping to crop an area of an image. Use

```
-crop 0x0
```

to trim edges that are the background color. Add an *x offset* and *y offset* to leave a portion of the trimmed edges with the image. The equivalent X resource for this option is *cropGeometry* (class *CropGeometry*). See Appendix B, X Resources for details.

-delay *<1/100ths of a second>x<seconds>*

Displays the next image after pausing.

This option is useful for regulating the display of the sequence of GIF images in Netscape. 1/100ths of a second must pass before the image sequence can be displayed again.

The default is no delay between each showing of the image sequence. The maximum delay is 65535.

The *seconds* value is optional. It lets you specify the number of seconds to pause before repeating the animation sequence.

-density *<width>x<height>*

Lets you specify in pixels the vertical and horizontal resolution of an image.

This option lets you specify an image density when decoding a PostScript or Portable Document page. The default is 72 pixels per inch in the horizontal and vertical direction.

-despeckle

-display *host:display[.screen]*

Specifies the X server to contact. See the X Windows system manual at http://www.x.org for details about the specification.

-dispose

Lets you specify one of the following GIF disposal methods:

GIF Disposal Methods

This method…	Specifies…
0	no disposal specified
1	do not dispose between frames
2	overwrite frame with background color from header
3	overwrite with previous frame

-dither

Lets you apply Floyd/Steinberg error diffusion to an image.

Dithering trades intensity resolution for spatial resolution by averaging the intensities of several neighboring pixels. You can use this option to improve images that suffer from severe contouring when reducing colors.

Note: The `-colors` or `-monochrome` option is required for dithering to take effect.

Tip! Use `+dither` to render PostScript without text or graphic aliasing.

-edge *factor*

Lets you detect edges within an image. Specify *factor* as a percentage of the enhancement from 0.0–99.9%.

-enhance

Lets you apply a digital filter to enhance a noisy image.

-filter *type*

Lets you specify one of the following filters to use when you resize an image:

- Point
- Box

Examples

- Triangle
- Hermite
- Hanning
- Hamming
- Blackman
- Gaussian
- Quadratic
- Cubic
- Catrom
- Mitchell (default)
- Lanczos
- Bessel
- Sinc

See -geometry.

-flip

Lets you create a mirror image by reflecting the scanlines in the vertical direction.

-flop

Lets you create a mirror image by reflecting the image scanlines in the horizontal direction.

-frame *<width>x<height>+<outer bevel width>+<inner bevel width>*

Lets you surround an image with an ornamental border. See the X Windows system manual at http://www.x.org for details about the specification.

Note: The color of the border is specified with the `-mattecolor` command line option.

-gamma *value*

Lets you specify the level of gamma correction for an image.

The same color image displayed on different workstations may look different because of differences in the display monitor. Use gamma correction to adjust for this color difference. Reasonable values range from 0.8–2.3.

You can apply separate gamma values to the red, green, and blue channels of an image with a gamma value list delineated with slashes, for example,

```
1.7/2.3/1.2
```

Use `+gamma` to set the image gamma level without actually adjusting the image pixels. This option is useful if the imagehas a known gamma that isn't set as an image attribute, such as PNG images.

-geometry *<width>x<height>{!}{<}{>}{%}*

Lets you specify the size and location of an image window. See the X Windows system manual at http://www.x.org for details about the geometry specification. By default, the window size is the image size. You specify its location when you map it.

The width and height, by default, are maximum values. That is, the image is expanded or contracted to fit the width and height value while maintaining the aspect ratio of the image.

Append an exclamation mark to the geometry to force the image size to exactly the size you specify. For example,

```
640x480!
```

sets the image width to 640 pixels and height to 480. If you specify one factor only, both the width and height assume that value.

To specify a percentage width or height instead, append `%`. The image size is multiplied by the width and height percentages to obtain the final image dimensions. To increase the size of an image, use a value greater than 100 (e.g., 125%). To decrease an image's size, use a percentage less than 100.

Use `>` to change the dimensions of the image only if its size exceeds the geometry specification. If the image dimension is smaller than the geometry you specify, `<` resizes the image. For example, if you specify

```
640x480>
```

and the image size is 512x512, the image size does not change. However, if the image is 1024x1024, it's resized to 640x480.

Tip! There are 72 pixels per inch in PostScript coordinates.

The equivalent X resource for this option is geometry (class Geometry). See Appendix B, X Resources **for details.**

-interlace *type*

Lets you specify one of the following interlacing schemes:

- none (default)
- line
- plane
- partition

Interlace also lets you specify the *type* of interlacing scheme for raw image formats such as RGB or YUV.

Interlace Types

Scheme	Description
none	does not interlace (e.g., RGBRGBRGBRGBRGBRGB...)

Interlace Types

Scheme (Cont.)	Description
line	uses scanline interlacing (e.g., RRR...GGG...BBB...RRR...GGG...BBB...)
plane	uses plane interlacing (e.g., RRRRRR...GGGGGG...BBBBBB...)
partition	similar to plane except that different planes are saved to individual files (e.g., image.R, image.G, and image.B)

Tip! Use `line`, or `plane` to create an interlaced GIF or progressive JPEG image.

-immutable

Lets you indicate the displayed image cannot be modified.

-label *name*

Lets you assign a label to an image.

-map *type*

Lets you display an image using one of the following standard colormap types:

- best

- default
- gray
- red
- green
- blue

The X server must support the colormap you choose, otherwise an error occurs. For *type* specify `list` and display searches the list of colormap types in top-to-bottom order until one is located. For one way of creating standard colormaps see *xstdcmap*, an X11 client program that's available with an X11 distribution.

-matte

Lets you store the matte channel (i.e., the transparent channel) if an image has one.

-monochrome

Lets you transform an image to black and white.

-negate

Lets you apply color inversion to an image.

The red, green, and blue intensities of an image are negated. Use `+negate` to negate only the grayscale pixels of the image.

-page *<width>x<height>{+-}<x offset>{+-}<y offset>{!}{<}{>}{%}*

Lets you set the size and location of an image canvas. Use this option to specify the dimensions of a

- PostScript page in dots per inch (dpi) or a
- TEXT page in pixels

This option is used in concert with `-density`.

The choices for a PostScript page are

Postscript Page Sizes

Media	Size (pixel width by pixel height)
11x17	792 1224
Ledger	1224 792
Legal	612 1008
Letter	612 792
LetterSmall	612 792

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
ArchE	2592 3456
ArchD	1728 2592
ArchC	1296 1728
ArchB	864 1296
ArchA	648 864
A0	2380 3368
A1	1684 2380
A2	1190 1684
A3	842 1190
A4	595 842
A4Small	595 842
A5	421 595
A6	297 421
A7	210 297

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
A8	148 210
A9	105 148
A10	74 105
B0	2836 4008
B1	2004 2836
B2	1418 2004
B3	1002 1418
B4	709 1002
B5	501 709
C0	2600 3677
C1	1837 2600
C2	1298 1837
C3	918 1298
C4	649 918

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
C5	459 649
C6	323 459
Flsa	612 936
Flse	612 936
HalfLetter	396 612

You can specify the page size by media (e.g. , A4, Ledger, etc.). Otherwise, `-page` behaves much like `-geometry` (e.g., `-page letter+43+43>`).

- To position a GIF image, use

  ```
  -page {+-}<x offset>{+-}<y offset>
  ```

  for example,

  ```
  -page +100+200
  ```

For a PostScript page, the image is sized as in `-geometry` and positioned relative to the lower-left hand corner of the page by *{+-}<x offset>{+-}<y offset>*. The default page dimension for a TEXT image is 612x792.

- To position a TEXT page, use

```
-page 612x792>
```

to center the image within the page.

Tip! If the image size exceeds the PostScript page, it’s reduced to fit the page.

-quality *value*

Lets you specify one of the following compression levels:

- JPEG with a *value* from 0–100 (i.e., worst to best); the default is 75
- MIFF with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)
- PNG with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)

The following are valid filter types:

- 0 for none; used for all scanlines
- 1 for sub; used for all scanlines
- 2 for up; used for all scanlines
- 3 for average; used for all scanlines

- 4 for Paeth; used for all scanlines
- 5 for adaptive filter; used when quality is greater than 50 and the image doesn't have a colormap; otherwise no filtering is used
- 6 or higher for adaptive filtering; used with minimum-sum-of-absolute-values

Note: The default is quality is 75—nearly the best compression with adaptive filtering.

For more information, see the PNG specification (RFC 2083) at http://www.w3.org/pub/WWW/TR.

-raise *<width>x<height>*

Lets you lighten or darken image edges to create a 3-D effect. See the X Windows system manual at http://www.x.org for details about the *geometry* specification.

Use `-raise` to create a raised effect; otherwise use `+raise`.

-remote *string*

Lets you execute a command in a remote display process.

Note: The only command recognized at this time is the name of an image file to load.

-roll *{+-}<x offset>{+-}<y offset>*

Lets you roll an image vertically or horizontally. See the X Windows system manual at http://www.x.org for details about the geometry specification.

A negative x offset rolls the image left to right. A negative y offset rolls the image top to bottom.

-rotate *degrees{<}{>}*

Applies Paeth image rotation to the image.

Use **>** to rotate the image only if its width exceeds the height. If the image width is less than its height, **<** rotates the image.

For example, if you have an image size of 480x640 and you specify

```
-90>
```

the image is not rotated by the specified angle. However, if the image is 640x480, it’s rotated by -90 degrees.

Note: Empty triangles left over from rotating the image are filled with the color defined as bordercolor (class BorderColor). See the X Windows system manual at http://www.x.org for details.

-sample *geometry*

-geometry

-scene *value*

Lets you specify the image scene number.

-segment *value*

Lets you eliminate insignificant clusters.

The number of pixels in each cluster must exceed the cluster threshold to be considered valid.

-sharpen *factor*

Lets you sharpen an image. Specify *factor* as a percentage of enhancement from 0.0–99.9%.

-size *<width>x<height>{+offset}{!}{%}*

Lets you specify the width and height of a raw image whose dimensions are unknown, such as GRAY, RGB, or CMYK.

In addition to *width* and *height*, use `-size` to skip any header information in the image or tell the number of colors in a MAP image file, for example,

```
-size 640x512+256
```

-texture *filename*

Lets you specify a file, which contains a texture, to tile onto an image’s background.

-title *string*

Lets you assign a title to the displayed image. The title is typically displayed in the window title bar.

-treedepth *value*

Lets you choose an optimal tree depth for the color reduction algorithm. Normally, *value* is 0 or 1.

An optimal depth generally provides the best representation of the source image with the fastest computational speed and the least amount of memory. However, the default depth is inappropriate for some images. To assure the best representation try values between 2 and 8. See Appendix D, Quantize for details.

Note: The `-colors` or `-monochrome` option is required for treedepth to take effect.

-update *seconds*

Lets you specify how often to determin an image has been updated and redisplay it.

For example, if an image you are displaying is overwritten, display will automatically detect the input file has been changed and update the displayed image accordingly.

-verbose

Lets you print the following detailed information about an image:

- image name

- image size
- image depth
- image format
- image comment
- image scene number
- image class (DirectClass or PseudoClass)
- total unique colors
- number of seconds to read and transform the image
- whether a matte is associated with the image
- the number of runlength packets

-visual *type*

Lets you display an image using one of the following visual types:

- StaticGray
- GrayScale

- StaticColor
- PseudoColor
- TrueColor
- DirectColor
- default
- visual ID

Note: The X server *must* support the visual you choose, otherwise an error occurs. If you don't specify a visual, the visual class that can display the most simultaneous colors on the default X server screen is used.

-window *ID*

Lets you set the background pixmap of this window to the image.

ID can be a window ID or name. Specify `root` to select X's root window as the target window. By default the image is tiled onto the background of the target window. If `-backdrop` or `-geometry` is specified, the image is surrounded by the background color. See Appendix B, X Resources for details.

Note: The image will not display on the root window if the image has more unique colors than the target window colormap allows.

Use `-colors` to reduce the number of colors. You can also specify the following standard X resources as command line options:

- -background
- -bordercolor
- -borderwidth
- -font
- -foreground
- -iconGeometry
- -iconic
- -mattecolor
- -name
- -title

-window_group *ID*

Lets you exit the program when this window ID is destroyed.

ID can be a window ID or name.

Working with Images

The following sections provide procedures for displaying images using the Command Widget. For details about using the Command Widget, see Chapter 3, The ImageMagick Interface.

- Loading Images
- Creating a Visual Image Directory
- Cutting Images
- Copying Images
- Pasting Images
- Cropping Images
- Chopping Images
- Rotating Images
- Segmenting Images
- Annotating Images
- Creating Composite Images
- Editing Color Images

- Editing Matte Images
- Drawing Images
- Transforming a Region of Interest
- Panning Images

Loading Images

1 To select an image to display, choose File/Open in the Command Widget.

A file browser is displayed.

2 To choose an image file, move the pointer to the filename click.

3 Click Open or press the Return key.

- o Alternatively, you can type the image file name directly into the Filename box.

4 To descend directories, double-click a directory name.

A scrollbar lets you move through a list of filenames that exceeds the size of the list area.

5 To shorten the list of file names, use shell globbing characters. For example, to list only files that end with .jpg, type

```
*.jpg
```

6 To select your image from the X server screen instead of from a file, choose Grab in the Open Widget.

Creating a Visual Image Directory

1 To create a visual image directory, choose File/Visual Directory in the Command Widget.

A file browser is displayed.

2 To create a visual image directory from all the images in the current directory, click Directory or press the Return key.

- o Alternatively, you can select a set of image names by using shell globbing characters. For example, to list only files that end with .jpg, type

```
*.jpg
```

3 To descend directories, dobule-click a directory name.

A scrollbar lets you move through a list of filenames that exceeds the size of the list area.

After you select a set of files, they are turned into thumbnails and tiled onto a single image.

4 Move the pointer to a thumbnail, press button 3, and drag.

5 Select Open.

The image represented by the thumbnail is displayed at its full size.

6 Choose File/Next in the Command Widget to return to the visual image directory.

Cutting Images

Note: Cut information for an image window is not retained for colormapped X server visuals (e.g., StaticColor, GrayScale, PseudoColor). Correct cutting behavior may require a TrueColor or DirectColor visual or a Standard Colormap.

1 To begin, choose Edit/Cut in the Command Widget.

- o Alternatively, press F3 in the image window.

A small window appears showing the location of the cursor in the image window. You are now in Cut mode.

2 To define a cut region, press button 1 and drag.

The cut region is defined by a highlighted rectangle that expands or contracts as it follows the pointer.

3 Once you are satisfied with the cut region, release the button.

You are now in Rectify mode.

4 To make adjustments, move the pointer to one of the cut rectangle corners, press a button, and drag.

5 Click Cut to commit your copy region.

- To exit without cutting the image, click Dismiss.

Copying Images

1 To begin, choose Edit/Copy in the Command Widget.

- Alternatively, press F4 in the image window.

A small window appears showing the location of the cursor in the image window. You are now in Copy mode.

2 To define a copy region, press button 1 and drag.

The copy region is defined by a highlighted rectangle that expands or contracts as it follows the pointer.

3 Once you are satisfied with the copy region, release the button.

You are now in Rectify mode.

4 To make adjustments, move the pointer to one of the copy rectangle corners, press a button, and drag.

5 Click Copy to commit your copy region.

- To exit without copying the image, click Dismiss.

Pasting Images

1 To begin, choose Edit/Paste in the Command Widget.

- o Alternatively, press F5 in the image window.

A small window appears showing the location of the cursor in the image window. You are now in Paste mode.

- o To exit immediately, press Dismiss.

2 Choose a composite operation from the Operators submenu.

- o Optionally choose a composite operator. The default operator is replace.

3 Choose a location to composite your image and press button 1.

- o Press and hold the button before releasing and an outline of the image will appear to help you identify your location.
- o To force a PseudoClass image to remain PseudoClass, use `-colors`.

The actual colors of the pasted image are saved. However, the color that appears in the image window may be different. For example, on a monochrome screen, the image window will appear black or white even though your pasted image may have many colors. If you save the image to a file, it is written with the correct colors. To assure the correct colors are saved in the final image, any PseudoClass image is promoted to DirectClass.

Composite Operator Behavior

The following describe how each operator behaves. *Image Window* is the image currently displayed on your X server and *image* is the image obtained with the File Browser Widget.

over. The result is the union of the two image shapes, with *image* obscuring *image window* in the region of overlap.

in. The result is simply *image* cut by the shape of *image window.* None of the image data of *image window* is in the result.

out. The resulting image is *image* with the shape of *image window* cut out.

atop. The result is the same shape as *image window,* with *image* obscuring *image window* where the image shapes overlap. Note this differs from **over** because the portion of *image* outside *image window's* shape does not appear in the result.

xor. The result is the image data from both *image* and *image window* that is outside the overlap region. The overlap region is blank.

plus. The result is just the sum of the image data. Output values are cropped to 255 (no overflow). This operation is independent of the matte channels.

minus. The result of *image - image window,* with underflow cropped to zero. The matte channel is ignored (set to 255, full coverage).

add. The result of *image + image window,* with overflow wrapping around (mod 256).

subtract. The result of *image - image window,* with underflow wrapping around (mod 256). The add and subtract operators can be used to perform reversible transformations.

difference. The result of abs(*image* - *image window*). This is useful for comparing two very similar images.

bumpmap. The result of *image window* shaded by *image.*

replace. The resulting image is *image window* replaced with *image.* Here the matte information is ignored.

The image compositor requires a matte, or alpha channel in the image for some operations. This extra channel usually defines a mask that represents a cookie-cutter for the image. This is the case when matte is 255 (full coverage) for pixels inside the shape, zero outside, and between zero and 255 on the boundary. If *image* does not have a matte channel, it is initialized with 0 for any pixel matching in color to pixel location (0,0), otherwise 255. See Editing Matte Images for a method of defining a matte channel.

Note: Matte information for *image window* is not retained for colormapped X server visuals (e.g., StaticColor, GrayScale, PseudoColor). Correct compositing behavior may require a TrueColor or DirectColor visual or a Standard Colormap.

Cropping Images

1 To begin, press choose Transform/Crop in the Command Widget.

- o Alternatively, press the [key in the image window.

A small window appears showing the location of the cursor in the image window. You are now in Crop mode.

2 To define a cropping region, press button 1 and drag.

The cropping region is defined by a highlighted rectangle that expands or contracts as it follows the pointer.

3 Once you are satisfied with the cropping region, release the button.

You are now in Rectify mode.

4 To make adjustments, move the pointer to one of the cropping rectangle corners, press a button, and drag.

5 Click Crop to commit your cropping region.

- To exit without cropping the image, click Dismiss.

Chopping Images

You can chop an image interactively—there is no command line argument to chop an image.

1 To begin, choose Transform/Chop in the Command Widget.

- Alternatively, press the] key in the Image window.

You are now in Chop mode.

- To exit immediately, click Dismiss

2 Select a location in the image window to begin your chop, and press and hold any button.

3 Move the pointer to another location in the image.

As you move a line will connect the initial location and the pointer.

4 Release the button.

- o To cancel the image chopping, move the pointer back to the starting point of the line and release the button.

5 The area within the image that's chopped is determined by the direction you choose from the Command Widget.

- o To chop the image between the two horizontal endpoints of the chop line, choose Direction/Horizontal. (This is the default.)
- o To chop the image between the two vertical endpoints of the chop line, choose Direction/Vertical.

Rotating Images

1 Press the / key to rotate the image 90 degrees or \ to rotate -90 degrees.

2 To interactively choose the degree of rotation, choose Transform/Rotate.

- o Alternatively, press the * key in the image window.

A small horizontal line is drawn next to the pointer. You are now in Rotate mode.

- o To exit immediately, click Dismiss.

3 Choose a background color from the Pixel Color submenu.

- o Choose Browser to specify additional background colors and set the X resources pen1 thorough pen9 to change the menu colors.

- o To select the background color using a color on the screen, choose Browser and click Grab. Move the pointer to the desired color on the screen and press any button.

4 Choose a point in the image window, and press and hold this button.

5 Move the pointer to another location in the image and release the button.

As you move a line connects the initial location and the pointer. When you release the button, the degree of image rotation is determined by the slope of the line you just drew.

- o To cancel the image rotation, move the pointer back to the starting point of the line and release the button.

6 From the Direction submenu of the Command Widget, choose Horizontal or Vertical.

The slope of the line you just drew is relative to the direction you choose.

Segmenting Images

Choose Effects/Segment to segment an image by analyzing the histograms of the color components and identifying units that are homogeneous with the fuzzy c-means technique. The scale-space filter analyzes the histograms of the three color components of the image and identifies a set of classes. The extents of each class is used to coarsely segment the image with thresholding. The color associated with each class is determined by the mean color of all pixels within the extents of a particular class. Finally, any unclassified pixels are assigned to the closest class with the fuzzy c-means technique.

The fuzzy c-Means algorithm can be summarized as follows:

- Build a histogram, one for each color component of the image.
- For each histogram, successively apply the scale-space filter and build an interval tree of zero crossings in the second derivative at each scale. Analyze this scale-space “fingerprint” to determine which peaks or valleys in the histogram are most predominant.
- The fingerprint defines intervals on the axis of the histogram. Each interval contains either a minima or a maxima in the original signal. If each color component lies within the maxima interval, that pixel is considered “classified” and is assigned a unique class number.
- Any pixel that fails to be classified in the above thresholding pass is classified using the fuzzy c-Means technique. It is assigned to one of the classes discovered in the histogram analysis phase.

The fuzzy c-Means technique attempts to cluster a pixel by finding the local minima of the generalized within group sum of squared error objective function. A pixel is assigned to the closest class of which the fuzzy membership has a maximum value.

For additional information see Young Won Lim, Sang Uk Lee, “On The Color Image Segmentation Algorithm Based on the Thresholding and the Fuzzy c-Means Techniques,” *Pattern Recognition*, Volume 23, Number 9, pages 935-952, 1990.

Annotating Images

You can annotate an image interactively—there is no command line argument to annotate an image.

1 To begin, choose Image Edit/Annotate in the Command Widget.

- o Alternatively, press the a key in the image window.

A small window appears showing the location of the cursor in the image window. You are now in Annotate mode.

- o To exit immediately, click Dismiss.

2 Optionally choose a font name from the Font Name submenu. The default is fixed.

- o Choose Browser from the Font Name submenu to specify additional font names. You can change the menu names by setting the X resources font1 through font9.

3 Optionally choose a font color from the Font Color submenu. The default is black.

- o Choose Browser from the Font Color submenu to specify additional font colors. You can change the menu colors by setting the X resources pen1 through pen9. If you select the color browser and press Grab, you can choose the font color by moving the pointer to a color on the screen and pressing any button.

4 To rotate text, choose Rotate Text from the menu and select an angle.

Tip: Typically you will only want to rotate one line of text at a time. Depending on the angle you choose, subsequent lines may end up overwriting each other.

5 Choose a location to begin entering text and press a button.

An underscore character will appear at the location of the pointer. The pointer changes to a pencil to indicate you are in Text mode.

- o To exit immediately, click Dismiss.

In Text mode, any key you press will display the character at the location of the underscore and advance the underscore cursor.

6 Enter your text.

7 When you’re finished, click Apply to finish your image annotation.

- o To correct errors, press Backspace.
- o To delete an entire line of text, press Delete.
- o Any text that exceeds the boundaries of the image window is automatically wrapped to the next line.

The actual color you request for the font is saved in the image. However, the color that appears in your Image window may be different. For example, on a monochrome screen the text will appear black or white even if you choose the color red as the font color. However, the image saved to a file with `-write` is written with red lettering. To assure the correct color text in the final image, any PseudoClass image is promoted to DirectClass (Appendix C, MIFF). To force a PseudoClass image to remain PseudoClass, use `-colors`.

Creating Composite Images

You can create an image composite interactively—there is no command line argument to composite an image.

1 To begin, choose Image Edit/Composite in the Command Widget.

- o Alternatively, press x in the Image window.

2 In the popup window that appears, enter an image name, do one of the following:

- o Type a file name.

Click Composite. If the composite image has no matte information, you are informed and the file browser is displayed again. Enter the name of a mask image. The image is typically grayscale and the same size as the composite image. If the image is not grayscale, it is converted to grayscale and the resulting intensities are used as matte information.

- o Click Grab and move the pointer to an image window and press any button.
- o Click Cancel if you choose not to create a composite image.

A small window appears showing the location of the cursor in the image window. You are now in Composite mode.

- o To exit immediately, click Dismiss.

3 Choose a composite operation from the Operators submenu of the Command Widget.

- o Optionally choose a compsite operator. The default operator is replace. (See Composite Operator Behavior for details about composite operators.)

4 Choose a location to composite your image an press button 1. Press andhold thebutton before releasing and an outline of the image appears to help you identify your location.

The actual colors of the composite image are saved. However, the color that appears in the image window may be different. For example, on a monochrome screen, *image window* will appear black or white even though your pasted image may have many colors. If you save the image to a file, it is written with the correct colors. To assure the correct colors are saved in the final image, any PseudoClass image is promoted to DirectClass.

5 Optionally choose Blend. The composite operator becomes over.

The image matte channel percent transparency is intialized to *factor*. The *image window* is intialized to *(100-factor)* where factor is the value you specify in the Dialog Widget.

6 To optionally shift the image pixels as defined by a displacement map, choose Displace.

With this option, *image* is used as a displacement map.

- o Black, within the displacement map, is a maximum positive displacement.
- o White is a maximum negative displacment and middle gray is neutral. the displacement is scaled to determine the pixel shift. By default the displacement applies in both the horizontal and vertical directions. However, if you specify a *mask*, *image* is the horizontal X displacement and *mask* is the vertical Y displacement.

Composite Operator Behavior

The following describe how each operator behaves. *Image Window* is the image currently displayed on your X server and *image* is the image obtained with the File Browser Widget.

over. The result is the union of the two image shapes, with *image* obscuring *image window* in the region of overlap.

in. The result is simply *image* cut by the shape of *image window.* None of the image data of *image window* is in the result.

out. The resulting image is *image* with the shape of *image window* cut out.

atop. The result is the same shape as *image window,* with *image* obscuring *image window* where the image shapes overlap. Note this differs from **over** because the portion of *image* outside *image window's* shape does not appear in the result.

xor. The result is the image data from both *image* and *image window* that is outside the overlap region. The overlap region is blank.

plus. The result is just the sum of the image data. Output values are cropped to 255 (no overflow). This operation is independent of the matte channels.

minus. The result of *image - image window,* with underflow cropped to zero. The matte channel is ignored (set to 255, full coverage).

add. The result of *image + image window,* with overflow wrapping around (mod 256).

subtract. The result of *image - image window,* with underflow wrapping around (mod 256). The add and subtract operators can be used to perform reversible transformations.

difference. The result of abs(*image* - *image window*). This is useful for comparing two very similar images.

bumpmap. The result of *image window* shaded by *image.*

replace. The resulting image is *image window* replaced with *image.* Here the matte information is ignored.

The image compositor requires a matte, or alpha channel in the image for some operations. This extra channel usually defines a mask that represents a cookie-cutter for the image. This is the case when matte is 255 (full coverage) for pixels inside the shape, zero outside, and between zero and 255 on the boundary. If *image* does not have a matte channel, it is initialized with 0 for any pixel matching in color to pixel location (0,0), otherwise 255. See Editing Matte Images for a method of defining a matte channel.

Note: Matte information for *image window* is not retained for colormapped X server visuals (e.g., StaticColor, GrayScale, PseudoColor). Correct compositing behavior may require a TrueColor or DirectColor visual or a Standard Colormap.

Editing Color Images

Changing the the color of a set of pixels is performed interactively. There is no command line argument to edit a pixel.

1 To begin, choose Image Image Edit/Color in the Command Widget.

o Alternatively, press c in the image window.

A small window appears showing the location of the cursor in the image window. You are now in Color Edit mode.

- o To exit immediately, press Dismiss.

2 Choose a color editing method from the Method submenu in the Command Widget.

- o The point method recolors any pixel selected with the pointer unless the button is released.
- o The replace method recolors any pixel that matches the color of the pixel you select with a button press. Floodfill recolors any pixel that matches the color of the pixel you select with a button press and is a neighbor.
- o Filltoborder changes the matte valueof any neighbor pixel that is not the border color.
- o Reset changes the entire image to the designated color.

3 Choose a pixel color from the Pixel Color submenu.

- o Additional pixel colors can be specified with the color browser by setting the X resources pen1 through pen9. (See Appendix B, X Resources.)

4 Press button 1 to select a pixel within the Image window to change its color.

- o You can recolor additional pixels as prescribed by the method you choose. you can recolor additional pixels by increasing the Delta value.
- o If the Magnify Widget is mapped, it can be helpful in positioning your pointer within the image (see Mouse Button 2).

- o Alternatively you can select a pixel to recolor from within the Magnify Widget. Move the pointer to the Magnify Widget and position the pixel with the cursor control keys. Finally, press a button to recolor the selected pixel (or pixels).

Note: The actual color you request for the pixels is saved in the image. However, the color that appears in your Image window may be different. For example, on a monochrome screen the pixel will appear black or white even if you choose the color red as the pixel color. However, the image saved to a file with `-write` is written with red pixels. To assure the correct color text in the final image, any PseudoClass image is promoted to DirectClass. To force a PseudoClass image to remain PseudoClass, use `-colors`.

Editing Matte Images

Matte information within an image is useful for some operations such as image compositing. This extra channel usually defines a mask that represents a sort of a cookie-cutter for the image. This is the case when matte is 255 (full coverage) for pixels inside the shape, zero outside, and between zero and 255 on the boundary.

Setting the matte information in an image is done interactively. There is no command line argument to edit a pixel.

1 To begin, choose Image Edit/Matte in the Command Widget.

- o Alternatively, click m in the image window.

A small window appears showing the location of the cursor in the image window. You are now in Matte Edit mode.

- o To exit immediately, press Dismiss.

2 Choose a matte editing method from the Method submenu of the Command Widget.

- o The point method changes the matte value of the any pixel selected with the pointer until the button is released.
- o The replace method changes the matte value of any pixel that matches the color of the pixel you select with a button press.
- o Floodfill changes the matte value of any pixel that matches the color of the pixel you select with a button press and is a neighbor.
- o Filltoborder recolors any neighbor pixel that is not the border color.
- o Reset changes the entire image to the designated matte value.

3 Choose Matte Value. A dialog prompts you for a matte value.

4 Enter a value between 0 and 255. This value is assigned as the matte value of the selected pixel or pixels.

5 Press any button to select a pixel within the Image window to change its matte value.

- o Optionally, you can change the matte value of additional pixels by increasing the Delta value. The Delta value is first added then subtracted from the red, green, and blue of the target color. Any pixels within the range also have their matte value updated. If the Magnify Widget is mapped, it can be helpful in positioning your pointer within the image (see Mouse Button 2).
- o Alternatively you can select a pixel to change the matte value from within the Magnify Widget. Move the pointer to the Magnify Widget and position the pixel with the cursor control keys.

6 Press a button to change the matte value of the selected pixel (or pixels).

Note: Matte information is only valid in a DirectClass image. Therefore, any PseudoClass image is promoted to DirectClass. Note that matte information for PseudoClass is not retained for colormapped X server visuals (e.g., StaticColor, StaticColor, GrayScale, PseudoColor) unless you immediately save your image to a file (refer to Write). Correct matte editing behavior may require a TrueColor or DirectColor visual or a Standard Colormap.

Drawing Images

You can interactively draw on an image—there is no command line argument to draw on an image.

1 To begin, choose Image Edit/Draw in the Command Widget.

- o Alternatively, press d in the image window.

The cursor changes to a crosshair to indicate you're in Draw mode.

- o To exit immediately, press Dismiss.

2 Choose a drawing primitive from the Primitive submenu.

3 Choose a color from the Color submenu.

- o To specify additional colors, choose Browser and set the X resources pen1 through pen9. (See Appendix B, X Resources for details.)
- o Choose Transparent to update the image matte channel, which is useful for image compositing.

- o If you Choose Browser and click Grab, you can select a primitive color by moving the pointer to the desired color on the screen and press any button.

4 Optionally choose a stipple from the Stipple submenu.

- o Choose Browser to specify additional stipples. Stipples obtained from the file browser must be on disk in the X11 bitmap format.

5 Optionally choose a line width from the Width submenu.

- o To choose a specific width select the Dialog Widget.

6 Choose a point in the image window and press and hold button 1.

7 Move the pointer to another location in the image.

As you move, a line connects the initial location and the pointer.

- o To cancel image drawing, move the pointer back to the starting point of the line and release the button.

8 Release the button.

The image is updated with the primitive you just drew.

Note: For polygons, the image is updated when you press and release the button without moving the pointer.

Transforming a Region of Interest

1. To begin, choose Pixel Transform/Region of Interest in the Command Widget.

   - Alternatively, press R in the image window.

   A small window appears showing the location of the cursor in the image window. You are now in Region of Interest mode.

2. To define a region of interest, press button 1 and drag.

   The region of interest is defined by a highlighted rectangle that expands or contracts as it follows the pointer.

3. Once you are satisfied with the region of interest, release the button. You are now in Apply mode.

4. You can make adjustments to the region of interest by moving the pointer to one of the rectangle corners, pressing a button, and dragging.

5. Choose an image processing technique from the Command Widget.

   Tip: You can choose more than one image processing technique to apply to an area. Alternatively, you can move the region of interest before applying another image processing technique.

   - To exit, press Dismiss.

Panning Images

When an image exceeds the width or height of the X server screen, Display maps a small panning icon. The rectangle within the panning icon shows the area that is currently displayed in the the image window.

1 To pan about the image, press any button and drag the pointer within the panning icon. The pan rectangle moves with the pointer and the image window is updated to reflect the location of the rectangle within the panning icon.

   o Use the arrow keys to pan the image one pixel at a time in any direction within the image window.

2 When you have selected the area of the image you want to view, release the button.

   Note: The panning icon is withdrawn if the image becomes smaller than the dimensions of the X server screen.

User Preferences

Preferences affect the default behavior of Display. Preferences can be either true or false and are stored in your home directory as .displayrc.

display image centered on a backdrop. This backdrop covers the entire workstation screen and is useful for hiding other X window activity while you view an image. The color of the backdrop is specified as the background color. (See Appendix B, X Resources for details.)

confirm on program exit. Prompts for a confirmation before exiting the Display.

correct image for display gamma. If the image has a known gamma, the gamma is corrected to match that of the X server. (See the X resource displayGamma (class DisplayGamma)).

apply Floyd/Steinberg error diffusion to image. The basic strategy of dithering is to trade intensity resolution for spatial resolution by averaging the intensities of several neighboring pixels. Images that suffer from severe contouring when you reduce colors can be improved with this perference.

use a shared colormap for colormapped X visuals. This option applies only when the default X server visual is PseudoColor or GrayScale. See -visual for more details. By default, a shared colormap is allocated. The image shares colors with other X clients. Some image colors could be approximated, therefore your image may look very different fromwhat you expect. Otherwise, the image colors appear exactly as they are defined. However, other clients may go technicolor when the image colormap is installed.

display images as an X server pixmap. Images are maintained as an XImage by default. Set this resource to True to utilize a server Pixmap instead. This option is useful if your image exceeds the dimensions of your server screen and you intend to pan the image. Panning is much faster with Pixmaps than with an XImage. Pixmaps are considered a precious resource, use them with discretion.

Chapter 5

Import

Overview

Import reads an image from any visible window on an X server and outputs it as an image file. You can capture a single window, the entire screen, or any rectangular portion of the screen. Use display for redisplay, printing, editing, formatting, archiving, image processing, etc. of the captured image.

The target window can be specified by id, name, or may be selected by clicking the mouse in the desired window. If you press a button and then drag, a rectangle will form which expands and contracts as the mouse moves. To save the portion of the screen defined by the rectangle, just release the button. The keyboard bell is rung once at the beginning of the screen capture and twice when it completes.

Syntax

```
import [ options ... ] file
```

Examples

- To select an X window with the mouse and save it in the MIFF image format to a file titled window.miff, use:

  ```
  import window.miff
  ```

- To select an X window and save it in the Encapsulated Postscript format to include in another document, use:

  ```
  import figure.eps
  ```

- To capture the entire X server screen in the JPEG image format in a file titled root.jpeg, use:

  ```
  import -window root root.jpeg
  ```

Import Options

Import options can appear on the command line or in your X resources file. See the X Windows system manual at http://www.x.org for details about the specification.

Options on the command line supersede values specified in your X resources file.

-adjoin

Lets you join images into a single multi-image file.

Note: By default, all images in an image sequence are stored in the same file. However, some formats, such as JPEG, do not support more than one image and are saved to separate files. Use `+adjoin` to force this behavior.

-border *<width>x<height>*

Lets you surround an image with a colored border.

The color of the border is obtained from the X server and is defined as *borderColor* (class *BorderColor*). See the X Windows system manual at http://www.x.org for details about the specification.

-cache_threshold *value*

number of megabytes available to the pixel cache.

Image pixels are stored in memory until 80 megabytes of memory have been consumed. Subsequent pixel operations are cached on disk. Operations to memory are significantly faster but if your computer does not have a sufficient amount of free memory you may want to adjust this threshold value.

-colors *value*

Lets you specify the preferred number of colors in an image.

The actual number of colors in the image may be fewer than you specify, but will never be more.

Note: This is a color reduction option. Duplicate and unused colors will be removed if an image has fewer unique colors than you specify. See Appendix D, Quantize for more details. The options `-dither`, `-colorspace`, and `-treedepth` affect the color reduction algorithm.

-colorspace *value*

Lets you specify the type of colorspace.

- GRAY

- OHTA
- RGB
- Transparent
- XYZ
- YCbCr
- YIQ
- YPbPr
- YUV
- CMYK

Color reduction by default, takes place in the RGB color space. Empirical evidence suggests that distances in color spaces such as YUV or YIQ correspond to perceptual color differences more closely than distances in RGB space. These color spaces may give better results when color reducing an image. See Appendix D, Quantize for details.

Note: The transparent colorspace is unique. It preserves the matte channel of the image if it exists.

Tip! The `-colors` or `-monochrome` option is required for the transparent option to take effect.

-comment *string*

Lets you annotate an image with a comment.

By default, each image is commented with its file name. Use this option to assign a specific comment to the image.

Optionally you can include the image filename, type, width, height, or scene number in the label by embedding special format characters. The following table shows these characters and their values.

Special Format Characters

Special Character	Value
%b	file size
%d	directory
%e	filename extention
%f	filename
%h	height
%i	input filename
%l	label
%m	magick

Special Format Characters

Special Character (Cont.)	Value
%n	number of scenes
%o	output filename
%p	page number
%q	quantum depth
%s	scene number
%t	top of filename
%u	unique temporary filename
%w	width
%x	x resolution
%y	y resolution
\n	newline
\r	carriage return

For example,

```
-comment "%m:%f %wx%h"
```

produces for an image—titled bird.miff whose width is 512 and height is 480—the comment

```
MIFF:bird.miff 512x480
```

Note: If the first character of *string* is `@`, the image comment is read from a file titled by the remaining characters in the string.

-compress *type*

Lets you specify one of the following types of image compression:

- None
- Bip
- Fax
- Group 4
- JPEG
- LZW
- RunlengthEncoded
- Zip

Specify

```
+compress
```

to store the binary image in an uncompressed format. The default is the compression type of the specified image file.

-crop *<width>x<height>{+-}<x offset>{+-}<y offset>{%}*

Lets you specify the size and location of a cropped image. See the X Windows system manual at http://www.x.org for details about the geometry specification.

To specify the width or height as a percentage, append %. For example to crop an image by 10% on all sides, use

```
-crop 10%
```

Use cropping to apply image processing options to, or display, a particular area of an image. Omit the *x offset* and *y offset* to generate one or more subimages of a uniform size.

Use cropping to crop an area of an image. Use

```
-crop 0x0
```

to trim edges that are the background color. Add an *x offset* and *y offset* to leave a portion of the trimmed edges with the image. The equivalent X resource for this option is *cropGeometry* (class *CropGeometry*). See Appendix B, X Resources for details.

-delay *<1/100ths of a second>x<seconds>*

Displays the next image after pausing.

This option is useful for regulating the display of the sequence of GIF images in Netscape. 1/100ths of a second must pass before the image sequence can be displayed again.

The default is no delay between each showing of the image sequence. The maximum delay is 65535.

The *seconds* value is optional. It lets you specify the number of seconds to pause before repeating the animation sequence.

-density *<width>x<height>*

Lets you specify in pixels the vertical and horizontal resolution of an image.

This option lets you specify an image density when decoding a PostScript or Portable Document page. The default is 72 pixels per inch in the horizontal and vertical direction.

-descend

Lets you obtain an image by descending window hierarchy.

-display *host:display[.screen]*

Specifies the X server to contact. See the X Windows system manual at http://www.x.org for details about the specification.

-dispose

Lets you specify one of the following GIF disposal methods:

GIF Disposal Methods

This method…	Specifies…
0	no disposal specified
1	do not dispose between frames
2	overwrite frame with background color from header
3	overwrite with previous frame

-dither

Lets you apply Floyd/Steinberg error diffusion to an image.

Dithering trades intensity resolution for spatial resolution by averaging the intensities of several neighboring pixels. You can use this option to improve images that suffer from severe contouring when reducing colors.

Note: The `-colors` or `-monochrome` option is required for dithering to take effect.

Tip! Use `+dither` to render PostScript without text or graphic aliasing.

-frame *<width>x<height>+<outer bevel width>+<inner bevel width>*

Lets you surround an image with an ornamental border. See the X Windows system manual at http://www.x.org for details about the specification.

Note: The color of the border is specified with the `-mattecolor` command line option.

-geometry *<width>x<height>{!}{<}{>}{%}*

Lets you specify the size and location of an image window. See the X Windows system manual at http://www.x.org for details about the geometry specification. By default, the window size is the image size. You specify its location when you map it.

The width and height, by default, are maximum values. That is, the image is expanded or contracted to fit the width and height value while maintaining the aspect ratio of the image.

Append an exclamation mark to the geometry to force the image size to exactly the size you specify. For example,

```
640x480!
```

sets the image width to 640 pixels and height to 480. If you specify one factor only, both the width and height assume that value.

To specify a percentage width or height instead, append %. The image size is multiplied by the width and height percentages to obtain the final image dimensions. To increase the size of an image, use a value greater than 100 (e.g., 125%). To decrease an image's size, use a percentage less than 100.

Use > to change the dimensions of the image only if its size exceeds the geometry specification. If the image dimension is smaller than the geometry you specify, < resizes the image. For example, if you specify

```
640x480>
```

and the image size is 512x512, the image size does not change. However, if the image is 1024x1024, it's resized to 640x480.

Tip! There are 72 pixels per inch in PostScript coordinates.

-interlace *type*

Lets you specify one of the following interlacing schemes:

- none (default)
- line
- plane
- partition

Interlace also lets you specify the *type* of interlacing scheme for raw image formats such as RGB or YUV.

Interlace Types

Scheme	Description
none	does not interlace (e.g., RGBRGBRGBRGBRGBRGB...)
line	uses scanline interlacing (e.g., RRR...GGG...BBB...RRR...GGG...BBB...)
plane	uses plane interlacing (e.g., RRRRRR...GGGGGG...BBBBBB...)
partition	similar to plane except that different planes are saved to individual files (e.g., image.R, image.G, and image.B)

Tip! Use `line`, or `plane` to create an interlaced GIF or progressive JPEG image.

-label *name*

Lets you assign a label to an image.

-monochrome

Lets you transform an image to black and white.

-negate

Lets you apply color inversion to an image.

The red, green, and blue intensities of an image are negated. Use `+negate` to negate only the grayscale pixels of the image.

-page *<width>x<height>{+-}<x offset>{+-}<y offset>{!}{<}{>}{%}*

Lets you set the size and location of an image canvas. Use this option to specify the dimensions of a

- PostScript page in dots per inch (dpi) or a
- TEXT page in pixels

This option is used in concert with `-density`.

The choices for a PostScript page are

Postscript Page Sizes

Media	Size (pixel width by pixel height)
11x17	792 1224
Ledger	1224 792

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
Legal	612 1008
Letter	612 792
LetterSmall	612 792
ArchE	2592 3456
ArchD	1728 2592
ArchC	1296 1728
ArchB	864 1296
ArchA	648 864
A0	2380 3368
A1	1684 2380
A2	1190 1684
A3	842 1190
A4	595 842
A4Small	595 842

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
A5	421 595
A6	297 421
A7	210 297
A8	148 210
A9	105 148
A10	74 105
B0	2836 4008
B1	2004 2836
B2	1418 2004
B3	1002 1418
B4	709 1002
B5	501 709
C0	2600 3677
C1	1837 2600

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
C2	1298 1837
C3	918 1298
C4	649 918
C5	459 649
C6	323 459
Flsa	612 936
Flse	612 936
HalfLetter	396 612

You can specify the page size by media (e.g. , A4, Ledger, etc.). Otherwise, `-page` behaves much like `-geometry` (e.g., `-page letter+43+43>`).

- To position a GIF image, use

```
-page {+-}<x offset>{+-}<y offset>
```

for example,

```
-page +100+200
```

For a PostScript page, the image is sized as in `-geometry` and positioned relative to the lower-left hand corner of the page by *{+-}<x offset>{+-}<y offset>*. The default page dimension for a TEXT image is 612x792.

- To position a TEXT page, use

  ```
  -page 612x792>
  ```

  to center the image within the page.

Tip! If the image size exceeds the PostScript page, it's reduced to fit the page.

-pointsize *value*

Lets you specify the point size of a PostScript font.

-quality *value*

Lets you specify one of the following compression levels:

- JPEG with a *value* from 0–100 (i.e., worst to best); the default is 75
- MIFF with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)
- PNG with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)

The following are valid filter types:

- 0 for none; used for all scanlines
- 1 for sub; used for all scanlines
- 2 for up; used for all scanlines
- 3 for average; used for all scanlines
- 4 for Paeth; used for all scanlines
- 5 for adaptive filter; used when quality is greater than 50 and the image doesn't have a colormap; otherwise no filtering is used
- 6 or higher for adaptive filtering; used with minimum-sum-of-absolute-values

Note: The default is quality is 75—nearly the best compression with adaptive filtering.

For more information, see the PNG specification (RFC 2083) at http://www.w3.org/pub/WWW/TR.

-rotate *degrees{<}{>}*

Applies Paeth image rotation to the image.

Use **>** to rotate the image only if its width exceeds the height. If the image width is less than its height, **<** rotates the image.

For example, if you have an image size of 480x640 and you specify

```
-90>
```

the image is not rotated by the specified angle. However, if the image is 640x480, it's rotated by -90 degrees.

Note: Empty triangles left over from rotating the image are filled with the color defined as bordercolor (class BorderColor). See the X Windows system manual at http://www.x.org for details.

-scene *value*

Lets you specify the image scene number.

-screen

Lets you indicate that the GetImage request used to obtain an image should be done on the root window, rather than directly on the specified window. In this way, you can obtain pieces of other windows that overlap the specified window and more importantly, you can capture menus or other popups that are independent windows, which appear over the specified window.

-silent

Lets you operate silently, i.e., without any bells.

-transparency *color*

Lets you make a specified color in an image transparent.

-treedepth *value*

Lets you choose an optimal tree depth for the color reduction algorithm. Normally, *value* is 0 or 1.

An optimal depth generally provides the best representation of the source image with the fastest computational speed and the least amount of memory. However, the default depth is inappropriate for some images. To assure the best representation try values between 2 and 8. See Appendix D, Quantize for details.

Note: The `-colors` or `-monochrome` option is required for treedepth to take effect.

-verbose

Lets you print the following detailed information about an image:

- image name
- image size
- image depth
- image format
- image comment
- image scene number
- image class (DirectClass or PseudoClass)

- total unique colors
- number of seconds to read and transform the image
- whether a matte is associated with the image
- the number of runlength packets

-window *ID*

Lets you set the background pixmap of this window to the image.

ID can be a window ID or name. Specify `root` to select X's root window as the target window. By default the image is tiled onto the background of the target window. If `-backdrop` or `-geometry` is specified, the image is surrounded by the background color. See Appendix B, X Resources for details.

Note: The image will not display on the root window if the image has more unique colors than the target window colormap allows.

Use `-colors` to reduce the number of colors. You can also specify the following standard X resources as command line options:

- -background
- -bordercolor
- -borderwidth

- -font
- -foreground
- -iconGeometry
- -iconic
- -mattecolor
- -name
- -title

Chapter 6

Animate

Overview

Animate displays a sequence of images on any workstation running an X server. Animate first determines the hardware capabilities of the workstation. If the number of unique colors in an image is fewer than or equal to the number the workstation can support, the image is displayed in an X window. Otherwise the number of colors in the image is first reduced to match the color resolution of the workstation.

For example, a continuous-tone 24 bits/pixel image candisplay on an 8-bit pseudo-color device or a monochrome device. In most cases the reduced color image closely resembles the original. Alternatively, a monochrome or pseudo-color image sequence can display on a continuous-tone 24 bits/pixels device.

To prevent color flashing on X server visuals that have colormaps, animate creates a single colormap from the image sequence, which can be time consuming. You can speed up this operation by reducing the colors in the image *before* you animate them.

- Use mogrify to color reduce the images to a single colormap. See Chapter9, Mogrify for details.
- Alternatively, you can use a standard colormap, or a static, direct, or true color visual. You can define a standard colormap with xstdcmap. See *xstdcmap*, an X11 client program that's available with an X11 distribution.

This method is recommended for colormapped X server because it eliminates the need to compute a global colormap.

Syntax

```
animate [options ...] file [ [options ...] file ...]
```

Examples

- To animate a set of images of a cockatoo, use

  ```
  animate cockatoo.*
  ```

- To animate a cockatoo image sequence using the Standard Colormap best, use

  ```
  xstdcmap -best
  animate -map best cockatoo.*
  ```

- To animate an image of a cockatoo without a border centered on a backdrop, use

  ```
  animate +borderwidth -backdrop cockatoo.*
  ```

Animate Options

-backdrop

Lets you center an image on a backdrop.

This backdrop covers the entire workstation screen and is useful for hiding other X window activity while viewing the image. The color of the backdrop is specified as the background color. See Appendix B, X Resources for details.

-cache_threshold *value*

number of megabytes available to the pixel cache.

Image pixels are stored in memory until 80 megabytes of memory have been consumed. Subsequent pixel operations are cached on disk. Operations to memory are significantly faster but if your computer does not have a sufficient amount of free memory you may want to adjust this threshold value.

-colormap *type*

Lets you specify a type of colormap:

- Shared
- Private

This option applies only when the default X server visual is PseudoColor or GrayScale. See -visual for more details.

By default, a *Shared* colormap is allocated. The image shares colors with other X clients. Some image colors may be approximated and your image may not look the way you intended.

Choose *Private* and the image colors appear exactly as they are defined. However, other clients may go technicolor when the image colormap is installed.

-colors *value*

Lets you specify the preferred number of colors in an image.

The actual number of colors in the image may be fewer than you specify, but will never be more.

Note: This is a color reduction option. Duplicate and unused colors will be removed if an image has fewer unique colors than you specify. See Appendix D, Quantize for more details. The options `-dither`, `-colorspace`, and `-treedepth` affect the color reduction algorithm.

-colorspace *value*

Lets you specify the type of colorspace.

- GRAY
- OHTA

- RGB
- Transparent
- XYZ
- YCbCr
- YIQ
- YPbPr
- YUV
- CMYK

Color reduction by default, takes place in the RGB color space. Empirical evidence suggests that distances in color spaces such as YUV or YIQ correspond to perceptual color differences more closely than distances in RGB space. These color spaces may give better results when color reducing an image. See Appendix D, Quantize for details.

Note: The transparent colorspace is unique. It preserves the matte channel of the image if it exists.

Tip! The `-colors` or `-monochrome` option is required for the transparent option to take effect.

-crop *<width>x<height>{+-}<x offset>{+-}<y offset>{%}*

Lets you specify the size and location of a cropped image. See the X Windows system manual at http://www.x.org for details about the geometry specification.

To specify the width or height as a percentage, append %. For example to crop an image by 10% on all sides, use

```
-crop 10%
```

Use cropping to apply image processing options to, or display, a particular area of an image. Omit the *x offset* and *y offset* to generate one or more subimages of a uniform size.

Use cropping to crop an area of an image. Use

```
-crop 0x0
```

to trim edges that are the background color. Add an *x offset* and *y offset* to leave a portion of the trimmed edges with the image. The equivalent X resource for this option is *cropGeometry* (class *CropGeometry*). See Appendix B, X Resources for details.

-delay *<1/100ths of a second>x<seconds>*

Displays the next image after pausing.

This option is useful for regulating the display of the sequence of GIF images in Netscape. 1/100ths of a second must pass before the image sequence can be displayed again.

The default is no delay between each showing of the image sequence. The maximum delay is 65535.

The *seconds* value is optional. It lets you specify the number of seconds to pause before repeating the animation sequence.

-density *<width>x<height>*

Lets you specify in pixels the vertical and horizontal resolution of an image.

This option lets you specify an image density when decoding a PostScript or Portable Document page. The default is 72 pixels per inch in the horizontal and vertical direction.

-display *host:display[.screen]*

Specifies the X server to contact. See the X Windows system manual at http://www.x.org for details about the specification.

-dither

Lets you apply Floyd/Steinberg error diffusion to an image.

Dithering trades intensity resolution for spatial resolution by averaging the intensities of several neighboring pixels. You can use this option to improve images that suffer from severe contouring when reducing colors.

Note: The `-colors` or `-monochrome` option is required for dithering to take effect.

Tip! Use `+dither` to render PostScript without text or graphic aliasing.

-gamma *value*

Lets you specify the level of gamma correction for an image.

The same color image displayed on different workstations may look different because of differences in the display monitor. Use gamma correction to adjust for this color difference. Reasonable values range from 0.8–2.3.

You can apply separate gamma values to the red, green, and blue channels of an image with a gamma value list delineated with slashes, for example,

```
1.7/2.3/1.2
```

Use `+gamma` to set the image gamma level without actually adjusting the image pixels. This option is useful if the imagehas a known gamma that isn't set as an image attribute, such as PNG images.

-geometry *<width>x<height>{!}{<}{>}{%}*

Lets you specify the size and location of an image window. See the X Windows system manual at http://www.x.org for details about the geometry specification. By default, the window size is the image size. You specify its location when you map it.

The width and height, by default, are maximum values. That is, the image is expanded or contracted to fit the width and height value while maintaining the aspect ratio of the image.

Append an exclamation mark to the geometry to force the image size to exactly the size you specify. For example,

```
640x480!
```

sets the image width to 640 pixels and height to 480. If you specify one factor only, both the width and height assume that value.

To specify a percentage width or height instead, append `%`. The image size is multiplied by the width and height percentages to obtain the final image dimensions. To increase the size of an image, use a value greater than 100 (e.g., 125%). To decrease an image's size, use a percentage less than 100.

Use **>** to change the dimensions of the image only if its size exceeds the geometry specification. If the image dimension is smaller than the geometry you specify, **<** resizes the image. For example, if you specify

```
640x480>
```

and the image size is 512x512, the image size does not change. However, if the image is 1024x1024, it's resized to 640x480.

Tip! There are 72 pixels per inch in PostScript coordinates.

-map *type*

Lets you display an image using one of the following standard colormap types:

- best
- default
- gray
- red
- green

- blue

The X server must support the colormap you choose, otherwise an error occurs. For *type* specify `list` and display searches the list of colormap types in top-to-bottom order until one is located. For one way of creating standard colormaps see *xstdcmap*, an X11 client program that's available with an X11 distribution.

-monochrome

Lets you transform an image to black and white.

-remote *string*

Lets you execute a command in a remote display process.

Note: The only command recognized at this time is the name of an image file to load.

-rotate *degrees{<}{>}*

Applies Paeth image rotation to the image.

Use **>** to rotate the image only if its width exceeds the height. If the image width is less than its height, **<** rotates the image.

For example, if you have an image size of 480x640 and you specify

```
-90>
```

the image is not rotated by the specified angle. However, if the image is 640x480, it's rotated by -90 degrees.

Note: Empty triangles left over from rotating the image are filled with the color defined as bordercolor (class BorderColor). See the X Windows system manual at http://www.x.org for details.

-scene *value*

Lets you specify the image scene number.

-size *<width>x<height>{+offset}{!}{%}*

Lets you specify the width and height of a raw image whose dimensions are unknown, such as GRAY, RGB, or CMYK.

In addition to *width* and *height*, use `-size` to skip any header information in the image or tell the number of colors in a MAP image file, for example,

```
-size 640x512+256
```

-title *string*

Lets you assign a title to the displayed image. The title is typically displayed in the window title bar.

-treedepth *value*

Lets you choose an optimal tree depth for the color reduction algorithm. Normally, *value* is 0 or 1.

An optimal depth generally provides the best representation of the source image with the fastest computational speed and the least amount of memory. However, the default depth is inappropriate for some images. To assure the best representation try values between 2 and 8. See Appendix D, Quantize for details.

Note: The `-colors` or `-monochrome` option is required for treedepth to take effect.

-verbose

Lets you print the following detailed information about an image:

- image name
- image size
- image depth
- image format
- image comment
- image scene number
- image class (DirectClass or PseudoClass)
- total unique colors
- number of seconds to read and transform the image

- whether a matte is associated with the image
- the number of runlength packets

-visual *type*

Lets you display an image using one of the following visual types:

- StaticGray
- GrayScale
- StaticColor
- PseudoColor
- TrueColor
- DirectColor
- default
- visual ID

Note: The X server *must* support the visual you choose, otherwise an error occurs. If you don't specify a visual, the visual class that can display the most simultaneous colors on the default X server screen is used.

-window *ID*

Lets you set the background pixmap of this window to the image.

ID can be a window ID or name. Specify `root` to select X's root window as the target window. By default the image is tiled onto the background of the target window. If `-backdrop` or `-geometry` is specified, the image is surrounded by the background color. See Appendix B, X Resources for details.

Note: The image will not display on the root window if the image has more unique colors than the target window colormap allows.

Use `-colors` to reduce the number of colors. You can also specify the following standard X resources as command line options:

- -background
- -bordercolor
- -borderwidth
- -font
- -foreground
- -iconGeometry
- -iconic

- -mattecolor
- -name
- -title

X Resources for Animate

Animate options can appear on the command line or in your X resource file. Options on the command line supersede values specified in your X resource file. See the X Windows system manual at http://www.x.org for details about the specification.

All animate options have a corresponding X resource. In addition, the animate program uses the following X resources:

- borderColor (class BorderColor)
- borderWidth (class BorderWidth)
- font (class Font or FontList)
- foreground (class Foreground)
- geometry (class geometry)
- iconGeometry (class IconGeometry)
- iconic (class Iconic)

- matteColor (class MatteColor)
- name (class Name)
- sharedMemory (class SharedMemory)
- text_font (class textFont)
- title (class Title)

For detailed information about these X Resources, see Appendix B, X Resources.

Chapter 7

Montage

Overview

Montage creates a composite by combining several separate images. The images are tiled on the composite image. The name of each image can be displayed below its tile.

Syntax

```
montage [ options ...] file [ [ options ...] file ...] output_file
```

Examples

- To create a montage of a cockatoo, a parrot, and a hummingbird and write it to a file called birds, use

  ```
  montage cockatoo.miff parrot.miff hummingbird.miff birds.miff
  ```

- To tile several bird images so that they are at most 256 pixels in width and 192 pixels in height, surrounded by a red border, and separated by 10 pixels of background color, use

  ```
  montage -geometry 256x192+10+10 -bordercolor red birds.*montage.miff
  ```

- To create an unlabeled parrot image, 640 by 480 pixels, and surrounded by a border of black, use

  ```
  montage -geometry 640x480 -bordercolor black -label "" parrot.miff bird.miff
  ```

- To create an image of an eagle with a textured background, use

  ```
  montage -texture bumps.jpg eagle.jpg eagle.png
  ```

- To join several GIF images together without any extraneous graphics (e.g. no label, no shadowing, no surrounding tile frame), use

  ```
  montage +frame +shadow +label -tile 5x1 -geometry 50x50+0+0 *.gif joined.gif
  ```

Montage Options

-adjoin

Lets you join images into a single multi-image file.

Note: By default, all images in an image sequence are stored in the same file. However, some formats, such as JPEG, do not support more than one image and are saved to separate files. Use `+adjoin` to force this behavior.

-blur *factor*

Lets you blur an image. Specify *factor* as a percentage of enhancement from 0.0–99.9%.

-cache_threshold *value*

number of megabytes available to the pixel cache.

Image pixels are stored in memory until 80 megabytes of memory have been consumed. Subsequent pixel operations are cached on disk. Operations to memory are significantly faster but if your computer does not have a sufficient amount of free memory you may want to adjust this threshold value.

-colors *value*

Lets you specify the preferred number of colors in an image.

The actual number of colors in the image may be fewer than you specify, but will never be more.

Note: This is a color reduction option. Duplicate and unused colors will be removed if an image has fewer unique colors than you specify. See Appendix D, Quantize for more details. The options `-dither`, `-colorspace`, and `-treedepth` affect the color reduction algorithm.

-colorspace *value*

Lets you specify the type of colorspace.

- GRAY
- OHTA
- RGB
- Transparent
- XYZ
- YCbCr
- YIQ
- YPbPr
- YUV

- CMYK

Color reduction by default, takes place in the RGB color space. Empirical evidence suggests that distances in color spaces such as YUV or YIQ correspond to perceptual color differences more closely than distances in RGB space. These color spaces may give better results when color reducing an image. See Appendix D, Quantize for details.

Note: The transparent colorspace is unique. It preserves the matte channel of the image if it exists.

Tip! The `-colors` or `-monochrome` option is required for the transparent option to take effect.

-comment *string*

Lets you annotate an image with a comment.

By default, each image is commented with its file name. Use this option to assign a specific comment to the image.

Optionally you can include the image filename, type, width, height, or scene number in the label by embedding special format characters. The following table shows these characters and their values.

Special Format Characters

Special Character	Value
%b	file size
%d	directory

Special Format Characters

Special Character (Cont.)	Value
%e	filename extention
%f	filename
%h	height
%i	input filename
%l	label
%m	magick
%n	number of scenes
%o	output filename
%p	page number
%q	quantum depth
%s	scene number
%t	top of filename
%u	unique temporary filename
%w	width

Special Format Characters

Special Character (Cont.)	Value
%x	x resolution
%y	y resolution
\n	newline
\r	carriage return

For example,

```
-comment "%m:%f %wx%h"
```

produces for an image—titled bird.miff whose width is 512 and height is 480—the comment

```
MIFF:bird.miff 512x480
```

Note: If the first character of *string* is `@`, the image comment is read from a file titled by the remaining characters in the string.

-compose *operator*

Lets you specify the type of image composition.

By default, each of the composite image pixels are replaced by the corresponding image tile pixel. You can choose an alternate composite operation. Each operator's behavior is described below.

Composition Operators

This operator…	Results in…
over	the union of the two image shapes, with the *composite image* obscuring the *image* in the region of overlap
in	*composite image* cut by the shape of the *image*; none of the image data of image will be in the result
out	*composite image* with the shape of the *image* cut out
atop	the same shape as image *image*, with *composite image* obscuring *image* where the image shapes overlap; (**Note:** This differs from *over* because the portion of *composite image* outside *image's* shape does not appear in the result.)
xor	the image data from both *composite image* and *image* that is outside the overlap region; the overlap region will be blank
plus	just the sum of the image data; output values are cropped to 255 (no overflow); this operation is independent of the matte channels
minus	*composite image* minus *image*, with underflow cropped to 0; the matte channel is ignored (set to 255, full coverage)
add	*composite image* plus *image*, with overflow wrapping around (mod 256)

Composition Operators

This operator… (Cont.)	Results in…
subtract	*composite image* minus *image*, with underflow wrapping around (mod 256); the add and subtract operators can be used to perform reversible transformations
difference	The result of abs (*composite image* minus *image*); this is useful for comparing two very similar images
bumpmap	*image* shaded by *composite image*
replace	*image* replaced with *composite image*; here the matte information is ignored

The image compositor requires a matte or alpha channel in the image for some operations. This extra channel usually defines a mask that represents a sort of a cookie-cutter for the image.

This is the case when matte is 255 (full coverage) for pixels inside the shape, 0 outside, and between 0 and 255 on the boundary. For certain operations, if *image* does not have a matte channel, it's initialized with 0 for any pixel matching in color to pixel location (0,0). Otherwise it's 255.

Note: To work properly, *borderwidth* must be 0.

-compress *type*

Lets you specify one of the following types of image compression:

- None
- Bip
- Fax
- Group 4
- JPEG
- LZW
- RunlengthEncoded
- Zip

Specify

```
+compress
```

to store the binary image in an uncompressed format. The default is the compression type of the specified image file.

-crop *<width>x<height>{+-}<x offset>{+-}<y offset>{%}*

Lets you specify the size and location of a cropped image. See the X Windows system manual at http://www.x.org for details about the geometry specification.

To specify the width or height as a percentage, append %. For example to crop an image by 10% on all sides, use

```
-crop 10%
```

Use cropping to apply image processing options to, or display, a particular area of an image. Omit the *x offset* and *y offset* to generate one or more subimages of a uniform size.

Use cropping to crop an area of an image. Use

```
-crop 0x0
```

to trim edges that are the background color. Add an *x offset* and *y offset* to leave a portion of the trimmed edges with the image. The equivalent X resource for this option is *cropGeometry* (class *CropGeometry*). See Appendix B, X Resources for details.

-density *<width>x<height>*

Lets you specify in pixels the vertical and horizontal resolution of an image.

This option lets you specify an image density when decoding a PostScript or Portable Document page. The default is 72 pixels per inch in the horizontal and vertical direction.

-dispose

Lets you specify one of the following GIF disposal methods:

GIF Disposal Methods

This method…	Specifies…
0	no disposal specified
1	do not dispose between frames
2	overwrite frame with background color from header
3	overwrite with previous frame

-dither

Lets you apply Floyd/Steinberg error diffusion to an image.

Dithering trades intensity resolution for spatial resolution by averaging the intensities of several neighboring pixels. You can use this option to improve images that suffer from severe contouring when reducing colors.

Note: The `-colors` or `-monochrome` option is required for dithering to take effect.

Tip! Use `+dither` to render PostScript without text or graphic aliasing.

-draw *string*

Lets you annotate an image with one or more of the followinggraphic primitives:

Graphic Primitives

This…	Requires…
point	a single coordinate
line	a single coordinate, start and end coordinates,
rectangle	*upper-left and lower-right coordinates*
fillRectangle	*upper-left and lower-right coordinates*
circle	center and an outer edge coordinates
fillCircle	center and an outer edge coordinates
polygon	three or more coordinates to define its boundaries
fillPolygon	three or more coordinates to define its boundaries
color	a single coordinate
matte	a single coordinate
text	a single coordinate
image	a single coordinate

Coordinates are integers separated by an optional comma. For example, to define a circle centered at 100,100 that extends to 150,150 use

```
-draw 'circle 100,100 150,150'
```

Consider the target pixel as that specified by your coordinate. Use *color* to change the color of a pixel. Follow the pixel coordinate with one of the following methods:

- *point* recolors the target pixel
- *replace* recolors any pixel that matches the color of the target pixel
- *floodfill* recolors any pixel that matches the color of the target pixel and its neighbor pixel
- *reset* recolors all pixels

Use *matte* to the change the pixel matte value to transparent. Follow the pixel coordinate with one of the following methods:

- *point* changes the matte value of the target pixel
- *replace* changes the matte value of any pixel that matches the color of the target pixel
- *floodfill* changes the matte value of any pixel that matches the color of the target pixel and its neighbor.
- *reset* changes the matte value of all pixels

Use *text* to annotate an image with text. Follow the text coordinates with a string.

Tip! If the string has embedded spaces, enclose it in double quotes.

Optionally you can include the image filename, type, width, height, or scene number in the label by embedding special format characters. The following table shows these characters and their values.

Special Format Characters

Special Character	Value
%b	file size
%d	directory
%e	filename extention
%f	filename
%h	height
%i	input filename
%l	label
%m	magick
%n	number of scenes
%o	output filename
%p	page number

Special Format Characters

Special Character (Cont.)	**Value**
%q	quantum depth
%s	scene number
%t	top of filename
%u	unique temporary filename
%w	width
%x	x resolution
%y	y resolution
\n	newline
\r	carriage return

For example,

```
-draw 'text 100,100 "%m:%f %wx%h"'
```

annotates an image—titled bird.miff whose width is 512 and height is 480—with

```
MIFF:bird.miff 512x480
```

To generate a Unicode character (TrueType fonts only), embed the code as an escaped hex string, for example,

```
\\0x30a3
```

Use `-image` to composite an image with another image. Follow the image coordinates with the filename of an image. If the first character of the string is `@`, the text is read from a file titled by the remaining characters in the string.

You can set the primitive color, font color, and font bounding box color with `-pen`, `-font`, and `-box`, respectively. Options are processed in command-line order so be sure to use `-pen` *before* the `-draw` option.

-font *name*

Font lets you specify the font to use when annotating an image with text.

If the font is a fully-qualified X server font name, the font is obtained from an X server, for example,

```
-*-helvetica-medium-r-*-*-12-*-*-*-*-*-iso8859-*
```

To use a TrueType font, precede the TrueType filename with @, for example,

```
@times.ttf
```

Otherwise, specify a PostScript font, for example,

```
helvetica
```

-frame *<width>x<height>+<outer bevel width>+<inner bevel width>*

Lets you surround an image with an ornamental border. See the X Windows system manual at http://www.x.org for details about the specification.

Note: The color of the border is specified with the `-mattecolor` command line option.

-gamma *value*

Lets you specify the level of gamma correction for an image.

The same color image displayed on different workstations may look different because of differences in the display monitor. Use gamma correction to adjust for this color difference. Reasonable values range from 0.8–2.3.

You can apply separate gamma values to the red, green, and blue channels of an image with a gamma value list delineated with slashes, for example,

```
1.7/2.3/1.2
```

Use `+gamma` to set the image gamma level without actually adjusting the image pixels. This option is useful if the imagehas a known gamma that isn't set as an image attribute, such as PNG images.

-geometry *<width>x<height>{!}{<}{>}{%}*

Lets you specify the size and location of an image window. See the X Windows system manual at http://www.x.org for details about the geometry specification. By default, the window size is the image size. You specify its location when you map it.

The width and height, by default, are maximum values. That is, the image is expanded or contracted to fit the width and height value while maintaining the aspect ratio of the image.

Append an exclamation mark to the geometry to force the image size to exactly the size you specify. For example,

```
640x480!
```

sets the image width to 640 pixels and height to 480. If you specify one factor only, both the width and height assume that value.

To specify a percentage width or height instead, append **%**. The image size is multiplied by the width and height percentages to obtain the final image dimensions. To increase the size of an image, use a value greater than 100 (e.g., 125%). To decrease an image's size, use a percentage less than 100.

Use **>** to change the dimensions of the image only if its size exceeds the geometry specification. If the image dimension is smaller than the geometry you specify, **<** resizes the image. For example, if you specify

```
640x480>
```

and the image size is 512x512, the image size does not change. However, if the image is 1024x1024, it's resized to 640x480.

Tip! There are 72 pixels per inch in PostScript coordinates.

-gravity *direction*

Lets you specify the direction an image gravitates within a tile. See the X Windows system manual at http://www.x.org for details about the gravity specification.

-geometry

direction

-interlace *type*

Lets you specify one of the following interlacing schemes:

- none (default)
- line
- plane
- partition

Interlace also lets you specify the *type* of interlacing scheme for raw image formats such as RGB or YUV.

Interlace Types

Scheme	Description
none	does not interlace (e.g., RGBRGBRGBRGBRGBRGB...)
line	uses scanline interlacing (e.g., RRR...GGG...BBB...RRR...GGG...BBB...)
plane	uses plane interlacing (e.g., RRRRRR...GGGGGG...BBBBBB...)
partition	similar to plane except that different planes are saved to individual files (e.g., image.R, image.G, and image.B)

Tip! Use `line`, or `plane` to create an interlaced GIF or progressive JPEG image.

-label *name*

Lets you assign a label to an image.

-matte

Lets you store the matte channel (i.e., the transparent channel) if an image has one.

-mode *type*

- frame
- unframe (default)
- concatenate

unframe

```
+frame +shadow +borderwidth
```

-monochrome

Lets you transform an image to black and white.

-page *<width>x<height>{+-}<x offset>{+-}<y offset>{!}{<}{>}{%}*

Lets you set the size and location of an image canvas. Use this option to specify the dimensions of a

- PostScript page in dots per inch (dpi) or a
- TEXT page in pixels

This option is used in concert with `-density`.

The choices for a PostScript page are

Postscript Page Sizes

Media	Size (pixel width by pixel height)
11x17	792 1224
Ledger	1224 792
Legal	612 1008
Letter	612 792
LetterSmall	612 792
ArchE	2592 3456
ArchD	1728 2592
ArchC	1296 1728
ArchB	864 1296
ArchA	648 864
A0	2380 3368
A1	1684 2380
A2	1190 1684

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
A3	842 1190
A4	595 842
A4Small	595 842
A5	421 595
A6	297 421
A7	210 297
A8	148 210
A9	105 148
A10	74 105
B0	2836 4008
B1	2004 2836
B2	1418 2004
B3	1002 1418
B4	709 1002

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
B5	501 709
C0	2600 3677
C1	1837 2600
C2	1298 1837
C3	918 1298
C4	649 918
C5	459 649
C6	323 459
Flsa	612 936
Flse	612 936
HalfLetter	396 612

You can specify the page size by media (e.g. , A4, Ledger, etc.). Otherwise, `-page` behaves much like `-geometry` (e.g., `-page letter+43+43>`).

- To position a GIF image, use

```
-page {+-}<x offset>{+-}<y offset>
```

for example,

```
-page +100+200
```

For a PostScript page, the image is sized as in `-geometry` and positioned relative to the lower-left hand corner of the page by *{+-}<x offset>{+-}<y offset>*. The default page dimension for a TEXT image is 612x792.

- To position a TEXT page, use

  ```
  -page 612x792>
  ```

  to center the image within the page.

Tip! If the image size exceeds the PostScript page, it's reduced to fit the page.

-pen *color*

Lets you set the color of the font or opaque color. See -draw for details. See the X Windows system manual at http://www.x.org for details about the *color* specification.

-pointsize *value*

Lets you specify the point size of a PostScript font.

-quality *value*

Lets you specify one of the following compression levels:

- JPEG with a *value* from 0–100 (i.e., worst to best); the default is 75
- MIFF with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)
- PNG with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)

The following are valid filter types:

- 0 for none; used for all scanlines
- 1 for sub; used for all scanlines
- 2 for up; used for all scanlines
- 3 for average; used for all scanlines
- 4 for Paeth; used for all scanlines
- 5 for adaptive filter; used when quality is greater than 50 and the image doesn't have a colormap; otherwise no filtering is used

- 6 or higher for adaptive filtering; used with minimum-sum-of-absolute-values

Note: The default is quality is 75—nearly the best compression with adaptive filtering.

For more information, see the PNG specification (RFC 2083) at http://www.w3.org/pub/WWW/TR.

-rotate *degrees{<}{>}*

Applies Paeth image rotation to the image.

Use **>** to rotate the image only if its width exceeds the height. If the image width is less than its height, **<** rotates the image.

For example, if you have an image size of 480x640 and you specify

```
-90>
```

the image is not rotated by the specified angle. However, if the image is 640x480, it's rotated by -90 degrees.

Note: Empty triangles left over from rotating the image are filled with the color defined as bordercolor (class BorderColor). See the X Windows system manual at http://www.x.org for details.

-scene *value*

Lets you specify the image scene number.

-shadow

Lets you add a shadow to a tile to simulate depth.

-sharpen *factor*

Lets you sharpen an image. Specify *factor* as a percentage of enhancement from 0.0–99.9%.

-size *<width>x<height>{+offset}{!}{%}*

Lets you specify the width and height of a raw image whose dimensions are unknown, such as GRAY, RGB, or CMYK.

In addition to *width* and *height*, use `-size` to skip any header information in the image or tell the number of colors in a MAP image file, for example,

```
-size 640x512+256
```

-texture *filename*

Lets you specify a file, which contains a texture, to tile onto an image’s background.

-tile *<width>x<height>*

Lets you specify the number of tiles to appear in each row and column of a composite image.

Specify the numbr of tiles per row with *width* and the number of tiles per column with *height*. For example, if you want one tile in each row and up to 10 tiles in the composite image, use

```
-tile 1x10
```

The default is five tiles in each row and four tiles in each column of the composite.

-transparency *color*

Lets you make a specified color in an image transparent.

-treedepth *value*

Lets you choose an optimal tree depth for the color reduction algorithm. Normally, *value* is 0 or 1.

An optimal depth generally provides the best representation of the source image with the fastest computational speed and the least amount of memory. However, the default depth is inappropriate for some images. To assure the best representation try values between 2 and 8. See Appendix D, Quantize for details.

Note: The `-colors` or `-monochrome` option is required for treedepth to take effect.

-verbose

Lets you print the following detailed information about an image:

- image name

- image size
- image depth
- image format
- image comment
- image scene number
- image class (DirectClass or PseudoClass)
- total unique colors
- number of seconds to read and transform the image
- whether a matte is associated with the image
- the number of runlength packets

Additional Montage Options

- -background
- -bordercolor

- -borderwidth
- -font
- -foreground
- -mattecolor
- -title

Appendix B, X Resources

Chapter 8

Convert

Overview

Convert changes an input file of one image format to an output file of a different image format. In addition, various types of image processing can be performed on the converted image during the conversion process.

For a comprehensive list of the formats Convert recognizes, see Appendix A, Supported Image Formats. Support for some of these formats require additional programs or libraries; this informationis also provided in the appendix. See the Readme file for informaton about where to find the additional software.

Note: A format delineated with + means that if more than one image is specified, they are combined into a single multi-image file. Use `+adjoin` if you want to produce a single image for each frame.

Raw images are expected to have one byte per pixel unless ImageMagick is compiled in 16-bit mode. Here, the raw data is expected to be stored two bytes per pixel in most-significant-byte-first order.

Syntax

```
convert [ options … ] file [file …] file
```

Examples

- To convert a MIFF image of a cockatoo to a SUN raster image, use

```
convert cockatoo.miff sun:cockatoo.ras
```

- To convert a multi-page PostScript document to individual FAX pages, use

```
convert -monochrome document.ps fax:page
```

- To convert a TIFF image to a PostScript A4 page with the image in the lower left-hand corner, use

```
convert -page A4+0+0 image.tiff document.ps
```

- To convert a raw Gray image with a 128 byte header to a portable graymap, use

```
convert -size 768x512+128 gray:raw image.pgm
```

- To convert a Photo CD image to a TIFF image, use

```
convert -size 1536x1024 img0009.pcd image.tiff convert img0009.pcd[4] image.tiff
```

- To create a visual image directory of all your JPEG images, use

```
convert ‘vid:*.jpg’ directory.miff
```

- To annotate an image with blue text using font 12x24 at position (100,100), use

```
convert -font helvetica -pen blue -draw “text 100,100 Cockatoo” bird.jpg bird.miff
```

- To tile a 640x480 image with a JPEG texture with bumps use

```
convert -size 640x480 tile:bumps.jpg tiled.png
```

- To surround an icon with an ornamental border to use with Mosaic(1), use

```
convert -mattecolor #697B8F -frame 6x6 bird.jpg icon.png
```

- To create a GIF animation from a DNA molecule sequence, use

```
convert -delay 20 dna.* dna.gif
```

Convert Options

-adjoin

Lets you join images into a single multi-image file.

Note: By default, all images in an image sequence are stored in the same file. However, some formats, such as JPEG, do not support more than one image and are saved to separate files. Use `+adjoin` to force this behavior.

-align *type*

Lets you specify how to align text.

- Left (default)

- Center
- Right

See -draw for details.

-average

Lets you average a set of images.

-blur *factor*

Lets you blur an image. Specify *factor* as a percentage of enhancement from 0.0–99.9%.

-border <*width*>*x*<*height*>

Lets you surround an image with a colored border.

The color of the border is obtained from the X server and is defined as *borderColor* (class *BorderColor*). See the X Windows system manual at http://www.x.org for details about the specification.

-box *color*

-draw

-cache_threshold *value*

number of megabytes available to the pixel cache.

Image pixels are stored in memory until 80 megabytes of memory have been consumed. Subsequent pixel operations are cached on disk. Operations to memory are significantly faster but if your computer does not have a sufficient amount of free memory you may want to adjust this threshold value.

-charcoal *factor*

Lets you simulate a charcoal drawing. See the X Windows system manual at http://www.x.org for details about the specification.

-coalesce

Lets you merge a sequence of images.

-colorize *value*

value

`0/0/50`

-colors *value*

Lets you specify the preferred number of colors in an image.

The actual number of colors in the image may be fewer than you specify, but will never be more.

Note: This is a color reduction option. Duplicate and unused colors will be removed if an image has fewer unique colors than you specify. See Appendix D, Quantize for more details. The options `-dither`, `-colorspace`, and `-treedepth` affect the color reduction algorithm.

-colorspace *value*

Lets you specify the type of colorspace.

- GRAY
- OHTA
- RGB
- Transparent
- XYZ
- YCbCr
- YIQ

- YPbPr
- YUV
- CMYK

Color reduction by default, takes place in the RGB color space. Empirical evidence suggests that distances in color spaces such as YUV or YIQ correspond to perceptual color differences more closely than distances in RGB space. These color spaces may give better results when color reducing an image. See Appendix D, Quantize for details.

Note: The transparent colorspace is unique. It preserves the matte channel of the image if it exists.

Tip! The `-colors` or `-monochrome` option is required for the transparent option to take effect.

-comment *string*

Lets you annotate an image with a comment.

By default, each image is commented with its file name. Use this option to assign a specific comment to the image.

Optionally you can include the image filename, type, width, height, or scene number in the label by embedding special format characters. The following table shows these characters and their values.

Special Format Characters

Special Character	Value
%b	file size
%d	directory
%e	filename extention
%f	filename
%h	height
%i	input filename
%l	label
%m	magick
%n	number of scenes
%o	output filename
%p	page number
%q	quantum depth

Special Format Characters

Special Character (Cont.)	Value
%s	scene number
%t	top of filename
%u	unique temporary filename
%w	width
%x	x resolution
%y	y resolution
\n	newline
\r	carriage return

For example,

```
-comment "%m:%f %wx%h"
```

produces for an image—titled bird.miff whose width is 512 and height is 480—the comment

```
MIFF:bird.miff 512x480
```

Note: If the first character of *string* is `@`, the image comment is read from a file titled by the remaining characters in the string.

-compress *type*

Lets you specify one of the following types of image compression:

- None
- Bip
- Fax
- Group 4
- JPEG
- LZW
- RunlengthEncoded
- Zip

Specify

```
+compress
```

to store the binary image in an uncompressed format. The default is the compression type of the specified image file.

-contrast

Lets you enhance or reduce the intensity differences between the lighter and darker elements of an image.

Use

```
-contrast
```

to enhance the image or

```
+contrast
```

to reduce the image contrast.

-cycle *amount*

Amount

-deconstruct

Break down an image sequence into constituent parts.

-delay *<1/100ths of a second>x<seconds>*

Displays the next image after pausing.

This option is useful for regulating the display of the sequence of GIF images in Netscape. 1/100ths of a second must pass before the image sequence can be displayed again.

The default is no delay between each showing of the image sequence. The maximum delay is 65535.

The *seconds* value is optional. It lets you specify the number of seconds to pause before repeating the animation sequence.

-density *<width>x<height>*

Lets you specify in pixels the vertical and horizontal resolution of an image.

This option lets you specify an image density when decoding a PostScript or Portable Document page. The default is 72 pixels per inch in the horizontal and vertical direction.

-despeckle

-display *host:display[.screen]*

Specifies the X server to contact. See the X Windows system manual at http://www.x.org for details about the specification.

-dispose

Lets you specify one of the following GIF disposal methods:

GIF Disposal Methods

This method…	Specifies…
0	no disposal specified
1	do not dispose between frames
2	overwrite frame with background color from header
3	overwrite with previous frame

-dither

Lets you apply Floyd/Steinberg error diffusion to an image.

Dithering trades intensity resolution for spatial resolution by averaging the intensities of several neighboring pixels. You can use this option to improve images that suffer from severe contouring when reducing colors.

Note: The `-colors` or `-monochrome` option is required for dithering to take effect.

Tip! Use `+dither` to render PostScript without text or graphic aliasing.

-draw *string*

Lets you annotate an image with one or more of the followinggraphic primitives:

Graphic Primitives

This…	Requires…
point	a single coordinate
line	a single coordinate, start and end coordinates,
rectangle	*upper-left and lower-right coordinates*
fillRectangle	*upper-left and lower-right coordinates*
circle	center and an outer edge coordinates
fillCircle	center and an outer edge coordinates
polygon	three or more coordinates to define its boundaries

Graphic Primitives

This… (Cont.)	Requires…
fillPolygon	three or more coordinates to define its boundaries
color	a single coordinate
matte	a single coordinate
text	a single coordinate
image	a single coordinate

Coordinates are integers separated by an optional comma. For example, to define a circle centered at 100,100 that extends to 150,150 use

```
-draw 'circle 100,100 150,150'
```

Consider the target pixel as that specified by your coordinate. Use *color* to change the color of a pixel. Follow the pixel coordinate with one of the following methods:

- *point* recolors the target pixel
- *replace* recolors any pixel that matches the color of the target pixel
- *floodfill* recolors any pixel that matches the color of the target pixel and its neighbor pixel

- *reset* recolors all pixels

Use *matte* to the change the pixel matte value to transparent. Follow the pixel coordinate with one of the following methods:

- *point* changes the matte value of the target pixel
- *replace* changes the matte value of any pixel that matches the color of the target pixel
- *floodfill* changes the matte value of any pixel that matches the color of the target pixel and its neighbor.
- *reset* changes the matte value of all pixels

Use *text* to annotate an image with text. Follow the text coordinates with a string.

Tip! If the string has embedded spaces, enclose it in double quotes.

Optionally you can include the image filename, type, width, height, or scene number in the label by embedding special format characters. The following table shows these characters and their values.

Special Format Characters

Special Character	Value
%b	file size
%d	directory

Special Format Characters

Special Character (Cont.)	Value
%e	filename extention
%f	filename
%h	height
%i	input filename
%l	label
%m	magick
%n	number of scenes
%o	output filename
%p	page number
%q	quantum depth
%s	scene number
%t	top of filename
%u	unique temporary filename
%w	width

Special Format Characters

Special Character (Cont.)	Value
%x	x resolution
%y	y resolution
\n	newline
\r	carriage return

For example,

```
-draw 'text 100,100 "%m:%f %wx%h"'
```

annotates an image—titled bird.miff whose width is 512 and height is 480—with

```
MIFF:bird.miff 512x480
```

To generate a Unicode character (TrueType fonts only), embed the code as an escaped hex string, for example,

```
\\0x30a3
```

Use `-image` to composite an image with another image. Follow the image coordinates with the filename of an image. If the first character of the string is `@`, the text is read from a file titled by the remaining characters in the string.

You can set the primitive color, font color, and font bounding box color with `-pen`, `-font`, and `-box,` respectively. Options are processed in command-line order so be sure to use `-pen` *before* the `-draw` option.

-edge *factor*

Lets you detect edges within an image. Specify *factor* as a percentage of the enhancement from 0.0–99.9%.

-enhance

Lets you apply a digital filter to enhance a noisy image.

-equalize

Lets you perform histogram equalization on an image.

-filter *type*

Lets you specify one of the following filters to use when you resize an image:

- Point
- Box
- Triangle
- Hermite

- Hanning
- Hamming
- Blackman
- Gaussian
- Quadratic
- Cubic
- Catrom
- Mitchell (default)
- Lanczos
- Bessel
- Sinc

See -geometry.

-flip

Lets you create a mirror image by reflecting the scanlines in the vertical direction.

-flop

Lets you create a mirror image by reflecting the image scanlines in the horizontal direction.

-font *name*

Font lets you specify the font to use when annotating an image with text.

If the font is a fully-qualified X server font name, the font is obtained from an X server, for example,

```
-*-helvetica-medium-r-*-*-12-*-*-*-*-iso8859-*
```

To use a TrueType font, precede the TrueType filename with @, for example,

```
@times.ttf
```

Otherwise, specify a PostScript font, for example,

```
helvetica
```

-frame *<width>x<height>+<outer bevel width>+<inner bevel width>*

Lets you surround an image with an ornamental border. See the X Windows system manual at http://www.x.org for details about the specification.

Note: The color of the border is specified with the `-mattecolor` command line option.

-gamma *value*

Lets you specify the level of gamma correction for an image.

The same color image displayed on different workstations may look different because of differences in the display monitor. Use gamma correction to adjust for this color difference. Reasonable values range from 0.8–2.3.

You can apply separate gamma values to the red, green, and blue channels of an image with a gamma value list delineated with slashes, for example,

```
1.7/2.3/1.2
```

Use `+gamma` to set the image gamma level without actually adjusting the image pixels. This option is useful if the imagehas a known gamma that isn't set as an image attribute, such as PNG images.

-geometry *<width>x<height>{!}{<}{>}{%}*

Lets you specify the size and location of an image window. See the X Windows system manual at http://www.x.org for details about the geometry specification. By default, the window size is the image size. You specify its location when you map it.

The width and height, by default, are maximum values. That is, the image is expanded or contracted to fit the width and height value while maintaining the aspect ratio of the image.

Append an exclamation mark to the geometry to force the image size to exactly the size you specify. For example,

```
640x480!
```

sets the image width to 640 pixels and height to 480. If you specify one factor only, both the width and height assume that value.

To specify a percentage width or height instead, append **%**. The image size is multiplied by the width and height percentages to obtain the final image dimensions. To increase the size of an image, use a value greater than 100 (e.g., 125%). To decrease an image's size, use a percentage less than 100.

Use **>** to change the dimensions of the image only if its size exceeds the geometry specification. If the image dimension is smaller than the geometry you specify, **<** resizes the image. For example, if you specify

```
640x480>
```

and the image size is 512x512, the image size does not change. However, if the image is 1024x1024, it's resized to 640x480.

Tip! There are 72 pixels per inch in PostScript coordinates.

-implode *amount*

amount

-interlace *type*

Lets you specify one of the following interlacing schemes:

- none (default)

- line
- plane
- partition

Interlace also lets you specify the *type* of interlacing scheme for raw image formats such as RGB or YUV.

Interlace Types

Scheme	Description
none	does not interlace (e.g., RGBRGBRGBRGBRGBRGB...)
line	uses scanline interlacing (e.g., RRR...GGG...BBB...RRR...GGG...BBB...)
plane	uses plane interlacing (e.g., RRRRRR...GGGGGG...BBBBBB...)
partition	similar to plane except that different planes are saved to individual files (e.g., image.R, image.G, and image.B)

Tip! Use `line`, or `plane` to create an interlaced GIF or progressive JPEG image.

-label *name*

Lets you assign a label to an image.

-layer *type*

- red
- green
- blue
- matte

-linewidth *value*

Lets you set the width of a line. See -draw for details.

-loop *iterations*

iterations

-map *type*

Lets you display an image using one of the following standard colormap types:

- best
- default
- gray
- red
- green
- blue

The X server must support the colormap you choose, otherwise an error occurs. For *type* specify `list` and display searches the list of colormap types in top-to-bottom order until one is located. For one way of creating standard colormaps see *xstdcmap*, an X11 client program that's available with an X11 distribution.

-matte

Lets you store the matte channel (i.e., the transparent channel) if an image has one.

-median *radius*

radius .

-modulate *value*

```
-modulate 20/-10
```

-monochrome

Lets you transform an image to black and white.

-negate

Lets you apply color inversion to an image.

The red, green, and blue intensities of an image are negated. Use `+negate` to negate only the grayscale pixels of the image.

-noise

Lets you add noise to or reduce noise in an image.

The principal function of the noise peak elimination filter is to smooth the objects within an image without losing edge information and without creating undesired structures.

The algorithm replaces a pixel with its next neighbor in value within a 3 x 3 window, if this pixel is noise. A pixel is defined as noise if and only if the pixel is a maximum or minimum within the 3 x 3 window.

Use `+noise` followed by a noise type to add noise to an image. Choose from the following noise types:

- Uniform
- Gaussian
- Multiplicative
- Impulse
- Laplacian
- Poisson

-normalize

Lets you transform an image to span the full range of color values using this contrast enhancement technique.

-opaque *color*

-pen

-page *<width>x<height>{+-}<x offset>{+-}<y offset>{!}{<}{>}{%}*

Lets you set the size and location of an image canvas. Use this option to specify the dimensions of a

- PostScript page in dots per inch (dpi) or a
- TEXT page in pixels

This option is used in concert with `-density`.

The choices for a PostScript page are

Postscript Page Sizes

Media	Size (pixel width by pixel height)
11x17	792 1224
Ledger	1224 792
Legal	612 1008
Letter	612 792
LetterSmall	612 792
ArchE	2592 3456
ArchD	1728 2592

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
ArchC	1296 1728
ArchB	864 1296
ArchA	648 864
A0	2380 3368
A1	1684 2380
A2	1190 1684
A3	842 1190
A4	595 842
A4Small	595 842
A5	421 595
A6	297 421
A7	210 297
A8	148 210
A9	105 148

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
A10	74 105
B0	2836 4008
B1	2004 2836
B2	1418 2004
B3	1002 1418
B4	709 1002
B5	501 709
C0	2600 3677
C1	1837 2600
C2	1298 1837
C3	918 1298
C4	649 918
C5	459 649
C6	323 459

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
Flsa	612 936
Flse	612 936
HalfLetter	396 612

You can specify the page size by media (e.g. , A4, Ledger, etc.). Otherwise, `-page` behaves much like `-geometry` (e.g., `-page letter+43+43>`).

- To position a GIF image, use

  ```
  -page {+-}<x offset>{+-}<y offset>
  ```

  for example,

  ```
  -page +100+200
  ```

For a PostScript page, the image is sized as in `-geometry` and positioned relative to the lower-left hand corner of the page by *{+-}<x offset>{+-}<y offset>*. The default page dimension for a TEXT image is 612x792.

- To position a TEXT page, use

  ```
  -page 612x792>
  ```

  to center the image within the page.

Tip! If the image size exceeds the PostScript page, it's reduced to fit the page.

-paint *radius*

radius .

-pen *color*

Lets you set the color of the font or opaque color. See -draw for details. See the X Windows system manual at http://www.x.org for details about the *color* specification.

-pointsize *value*

Lets you specify the point size of a PostScript font.

-quality *value*

Lets you specify one of the following compression levels:

- JPEG with a *value* from 0–100 (i.e., worst to best); the default is 75
- MIFF with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)

- PNG with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)

The following are valid filter types:

- 0 for none; used for all scanlines
- 1 for sub; used for all scanlines
- 2 for up; used for all scanlines
- 3 for average; used for all scanlines
- 4 for Paeth; used for all scanlines
- 5 for adaptive filter; used when quality is greater than 50 and the image doesn't have a colormap; otherwise no filtering is used
- 6 or higher for adaptive filtering; used with minimum-sum-of-absolute-values

Note: The default is quality is 75—nearly the best compression with adaptive filtering.

For more information, see the PNG specification (RFC 2083) at http://www.w3.org/pub/WWW/TR.

-raise *<width>x<height>*

Lets you lighten or darken image edges to create a 3-D effect. See the X Windows system manual at http://www.x.org for details about the *geometry* specification.

Use `-raise` to create a raised effect; otherwise use `+raise`.

-region *<width>x<height>{+-}<x offset>{+-}<y offset>*

`-region`

-roll *{+-}<x offset>{+-}<y offset>*

Lets you roll an image vertically or horizontally. See the X Windows system manual at http://www.x.org for details about the geometry specification.

A negative x offset rolls the image left to right. A negative y offset rolls the image top to bottom.

-rotate *degrees{<}{>}*

Applies Paeth image rotation to the image.

Use > to rotate the image only if its width exceeds the height. If the image width is less than its height, < rotates the image.

For example, if you have an image size of 480x640 and you specify

```
-90>
```

the image is not rotated by the specified angle. However, if the image is 640x480, it's rotated by -90 degrees.

Note: Empty triangles left over from rotating the image are filled with the color defined as bordercolor (class BorderColor). See the X Windows system manual at http://www.x.org for details.

-sample *geometry*

-geometry

-scene *value*

Lets you specify the image scene number.

-seed *value*

-segment *value*

Lets you eliminate insignificant clusters.

The number of pixels in each cluster must exceed the cluster threshold to be considered valid.

-shade *<azimuth>x<elevation>*

azimuth *elevation* `+shade`

-sharpen *factor*

Lets you sharpen an image. Specify *factor* as a percentage of enhancement from 0.0–99.9%.

-size *<width>x<height>{+offset}{!}{%}*

Lets you specify the width and height of a raw image whose dimensions are unknown, such as GRAY, RGB, or CMYK.

In addition to *width* and *height*, use `-size` to skip any header information in the image or tell the number of colors in a MAP image file, for example,

```
-size 640x512+256
```

-solarize *factor*

Lets you negate all pixels above a threshold level. Specify *factor* as a percentage of the intensity threshold from 0 - 99.9%.

Note: This option produces a solarization effect seen when exposing a photographic film to light during the development process.

-spread *amount*

Lets you displace image pixels by a random amount.

Amount defines the size of the neighborhood around each pixel from which to choose a candidate pixel to swap.

-swirl *degrees*

Lets you swirl image pixels about the center of an image.

Degrees defines the tightness of the swirl.

-transparency *color*

Lets you make a specified color in an image transparent.

-texture *filename*

Lets you specify a file, which contains a texture, to tile onto an image's background.

-threshold *value*

Threshold lets you create a bi-level image such that any pixel intensity that is equal to or exceeds the threshold *value* you specify is reassigned the maximum intensity. Otherwise, it's reassigned the the minimum intensity.

-treedepth *value*

Lets you choose an optimal tree depth for the color reduction algorithm. Normally, *value* is 0 or 1.

An optimal depth generally provides the best representation of the source image with the fastest computational speed and the least amount of memory. However, the default depth is inappropriate for some images. To assure the best representation try values between 2 and 8. See Appendix D, Quantize for details.

Note: The `-colors` or `-monochrome` option is required for treedepth to take effect.

-undercolor *<undercolor factor>x<black-generation factor>*

Lets you control undercolor removal and black generation on CMYK images (i.e., images to be printed on a four-color printing system).

You can control the amount of cyan, magenta, and yellow to remove from your image and the amount of black to add to it. The standard undercolor removal is 1.0x1.0. You'll frequently get better results though if the percentage of black you add to your image is slightly higher than the percentage of C, M, and Y you remove from it. For example, you might try 0.5x0.7.

-verbose

Lets you print the following detailed information about an image:

- image name
- image size
- image depth
- image format
- image comment
- image scene number
- image class (DirectClass or PseudoClass)
- total unique colors
- number of seconds to read and transform the image
- whether a matte is associated with the image

- the number of runlength packets

-view *string*

-wave *<amplitude>x<wavelength>*

Lets you alter an image along a sine wave.

Specify *amplitude* and *wavelength* to affect the characteristics of the wave.

Segmenting Images

Use `-segment` to segment an image by analyzing the histograms of the color components and identifying units that are homogeneous with the fuzzy c-means technique. The scale-space filter analyzes the histograms of the three color components of the image and identifies a set of classes. The extents of each class are used to coarsely segment the image with thresholding. The color associated with each class is determined by the mean color of all pixels within the extents of a particular class. Finally, any unclassified pixels are assigned to the closest class with the fuzzy c-means technique.

The fuzzy c-Means algorithm can be summarized as follows:

- Build a histogram, one for each color component of the image.

- For each histogram, successively apply the scale-space filter and build an interval tree of 0 crossings in the second derivative at each scale. Analyze this scale-space “fingerprint’’ to determine which peaks or valleys in the histogram are most predominant.

- The fingerprint defines intervals on the axis of the histogram. Each interval contains either a minima or a maxima in the original signal. If each color component lies within the maxima interval, that pixel is considered ‘‘classified’’ and is assigned an unique class number.

- Any pixel that fails to be classified in the above thresholding pass is classified using the fuzzy c-Means technique. It is assigned to one of the classes discovered in the histogram analysis phase.

The fuzzy c-Means technique attempts to cluster a pixel by finding the local minima of the generalized within group sum of squared error objective function. A pixel is assigned to the closest class of which the fuzzy membership has a maximum value.

For additional information see Young Won Lim, Sang Uk Lee. “On the Color Image Segmentation Algorithm Based on the Thresholding and the Fuzzy c-Means Techniques,” *Pattern Recognition, Volume 23, Number 9,* pages 935–952, 1990.

Chapter 9

Mogrify

Overview

Mogrify transforms an image or a sequence of images. These transformations include image scaling, image rotation, color reduction, and others. The transmogrified image overwrites the original image.

Syntax

```
mogrify [ options ...] file [ [ options ...] file ...]
```

Examples

- To convert all the TIFF files in a particular directory to JPEG, use

  ```
  mogrify -format jpeg *.tiff
  ```

- To scale an image of a cockatoo to exactly 640 pixels in width and 480 pixels in height, use

  ```
  mogrify -geometry 640x480! cockatoo.miff
  ```

Mogrify Options

-align *type*

Lets you specify how to align text.

- Left (default)
- Center
- Right

See -draw for details.

-blur *factor*

Lets you blur an image. Specify *factor* as a percentage of enhancement from 0.0–99.9%.

-border *<width>x<height>*

Lets you surround an image with a colored border.

The color of the border is obtained from the X server and is defined as *borderColor* (class *BorderColor*). See the X Windows system manual at http://www.x.org for details about the specification.

-box *color*

-draw

-cache_threshold *value*

number of megabytes available to the pixel cache.

Image pixels are stored in memory until 80 megabytes of memory have been consumed. Subsequent pixel operations are cached on disk. Operations to memory are significantly faster but if your computer does not have a sufficient amount of free memory you may want to adjust this threshold value.

-charcoal *factor*

Lets you simulate a charcoal drawing. See the X Windows system manual at http://www.x.org for details about the specification.

-colorize *value*

value

`0/0/50`

-colors *value*

Lets you specify the preferred number of colors in an image.

The actual number of colors in the image may be fewer than you specify, but will never be more.

Note: This is a color reduction option. Duplicate and unused colors will be removed if an image has fewer unique colors than you specify. See Appendix D, Quantize for more details. The options `-dither`, `-colorspace`, and `-treedepth` affect the color reduction algorithm.

-colorspace *value*

Lets you specify the type of colorspace.

- GRAY
- OHTA
- RGB
- Transparent
- XYZ
- YCbCr
- YIQ

- YPbPr
- YUV
- CMYK

Color reduction by default, takes place in the RGB color space. Empirical evidence suggests that distances in color spaces such as YUV or YIQ correspond to perceptual color differences more closely than distances in RGB space. These color spaces may give better results when color reducing an image. See Appendix D, Quantize for details.

Note: The transparent colorspace is unique. It preserves the matte channel of the image if it exists.

Tip! The `-colors` or `-monochrome` option is required for the transparent option to take effect.

-comment *string*

Lets you annotate an image with a comment.

By default, each image is commented with its file name. Use this option to assign a specific comment to the image.

Optionally you can include the image filename, type, width, height, or scene number in the label by embedding special format characters. The following table shows these characters and their values.

Special Format Characters

Special Character	Value
%b	file size
%d	directory
%e	filename extention
%f	filename
%h	height
%i	input filename
%l	label
%m	magick
%n	number of scenes
%o	output filename
%p	page number
%q	quantum depth

Special Format Characters

Special Character (Cont.)	Value
%s	scene number
%t	top of filename
%u	unique temporary filename
%w	width
%x	x resolution
%y	y resolution
\n	newline
\r	carriage return

For example,

```
-comment “%m:%f %wx%h”
```

produces for an image—titled bird.miff whose width is 512 and height is 480—the comment

```
MIFF:bird.miff 512x480
```

Note: If the first character of *string* is `@`, the image comment is read from a file titled by the remaining characters in the string.

-compress *type*

Lets you specify one of the following types of image compression:

- None
- Bip
- Fax
- Group 4
- JPEG
- LZW
- RunlengthEncoded
- Zip

Specify

```
+compress
```

to store the binary image in an uncompressed format. The default is the compression type of the specified image file.

-contrast

Lets you enhance or reduce the intensity differences between the lighter and darker elements of an image.

Use

```
-contrast
```

to enhance the image or

```
+contrast
```

to reduce the image contrast.

-crop *<width>x<height>{+-}<x offset>{+-}<y offset>{%}*

Lets you specify the size and location of a cropped image. See the X Windows system manual at http://www.x.org for details about the geometry specification.

To specify the width or height as a percentage, append %. For example to crop an image by 10% on all sides, use

```
-crop 10%
```

Use cropping to apply image processing options to, or display, a particular area of an image. Omit the *x offset* and *y offset* to generate one or more subimages of a uniform size.

Use cropping to crop an area of an image. Use

```
-crop 0x0
```

to trim edges that are the background color. Add an *x offset* and *y offset* to leave a portion of the trimmed edges with the image. The equivalent X resource for this option is *cropGeometry* (class *CropGeometry*). See Appendix B, X Resources for details.

-cycle *amount*

Amount

-delay *<1/100ths of a second>x<seconds>*

Displays the next image after pausing.

This option is useful for regulating the display of the sequence of GIF images in Netscape. 1/100ths of a second must pass before the image sequence can be displayed again.

The default is no delay between each showing of the image sequence. The maximum delay is 65535.

The *seconds* value is optional. It lets you specify the number of seconds to pause before repeating the animation sequence.

-density *<width>x<height>*

Lets you specify in pixels the vertical and horizontal resolution of an image.

This option lets you specify an image density when decoding a PostScript or Portable Document page. The default is 72 pixels per inch in the horizontal and vertical direction.

-despeckle

-display *host:display[.screen]*

Specifies the X server to contact. See the X Windows system manual at http://www.x.org for details about the specification.

-dispose

Lets you specify one of the following GIF disposal methods:

GIF Disposal Methods

This method…	Specifies…
0	no disposal specified

GIF Disposal Methods

This method…	Specifies…
1	do not dispose between frames
2	overwrite frame with background color from header
3	overwrite with previous frame

-dither

Lets you apply Floyd/Steinberg error diffusion to an image.

Dithering trades intensity resolution for spatial resolution by averaging the intensities of several neighboring pixels. You can use this option to improve images that suffer from severe contouring when reducing colors.

Note: The `-colors` or `-monochrome` option is required for dithering to take effect.

Tip! Use `+dither` to render PostScript without text or graphic aliasing.

-draw *string*

Lets you annotate an image with one or more of the followinggraphic primitives:

Graphic Primitives

This…	Requires…
point	a single coordinate
line	a single coordinate, start and end coordinates,
rectangle	*upper-left and lower-right coordinates*
fillRectangle	*upper-left and lower-right coordinates*
circle	center and an outer edge coordinates
fillCircle	center and an outer edge coordinates
polygon	three or more coordinates to define its boundaries
fillPolygon	three or more coordinates to define its boundaries
color	a single coordinate
matte	a single coordinate
text	a single coordinate
image	a single coordinate

Coordinates are integers separated by an optional comma. For example, to define a circle centered at 100,100 that extends to 150,150 use

```
-draw 'circle 100,100 150,150'
```

Consider the target pixel as that specified by your coordinate. Use *color* to change the color of a pixel. Follow the pixel coordinate with one of the following methods:

- *point* recolors the target pixel
- *replace* recolors any pixel that matches the color of the target pixel
- *floodfill* recolors any pixel that matches the color of the target pixel and its neighbor pixel
- *reset* recolors all pixels

Use *matte* to the change the pixel matte value to transparent. Follow the pixel coordinate with one of the following methods:

- *point* changes the matte value of the target pixel
- *replace* changes the matte value of any pixel that matches the color of the target pixel
- *floodfill* changes the matte value of any pixel that matches the color of the target pixel and its neighbor.
- *reset* changes the matte value of all pixels

Use *text* to annotate an image with text. Follow the text coordinates with a string.

Tip! If the string has embedded spaces, enclose it in double quotes.

Optionally you can include the image filename, type, width, height, or scene number in the label by embedding special format characters. The following table shows these characters and their values.

Special Format Characters

Special Character	Value
%b	file size
%d	directory
%e	filename extention
%f	filename
%h	height
%i	input filename
%l	label
%m	magick
%n	number of scenes
%o	output filename
%p	page number

Special Format Characters

Special Character (Cont.)	Value
%q	quantum depth
%s	scene number
%t	top of filename
%u	unique temporary filename
%w	width
%x	x resolution
%y	y resolution
\n	newline
\r	carriage return

For example,

```
-draw 'text 100,100 "%m:%f %wx%h"'
```

annotates an image—titled bird.miff whose width is 512 and height is 480—with

```
MIFF:bird.miff 512x480
```

To generate a Unicode character (TrueType fonts only), embed the code as an escaped hex string, for example,

```
\\0x30a3
```

Use `-image` to composite an image with another image. Follow the image coordinates with the filename of an image. If the first character of the string is `@`, the text is read from a file titled by the remaining characters in the string.

You can set the primitive color, font color, and font bounding box color with `-pen`, `-font`, and `-box`, respectively. Options are processed in command-line order so be sure to use `-pen` *before* the `-draw` option.

-edge *factor*

Lets you detect edges within an image. Specify *factor* as a percentage of the enhancement from 0.0–99.9%.

-emboss

-enhance

Lets you apply a digital filter to enhance a noisy image.

-equalize

Lets you perform histogram equalization on an image.

-filter *type*

Lets you specify one of the following filters to use when you resize an image:

- Point
- Box
- Triangle
- Hermite
- Hanning
- Hamming
- Blackman
- Gaussian
- Quadratic
- Cubic
- Catrom
- Mitchell (default)

- Lanczos
- Bessel
- Sinc

See -geometry.

-flip

Lets you create a mirror image by reflecting the scanlines in the vertical direction.

-flop

Lets you create a mirror image by reflecting the image scanlines in the horizontal direction.

-format *type*

type

`image.gif`

```
image.tiff
```

Appendix A, Supported Image Formats

-font *name*

Font lets you specify the font to use when annotating an image with text.

If the font is a fully-qualified X server font name, the font is obtained from an X server, for example,

```
-*-helvetica-medium-r-*-*-12-*-*-*-*-*-iso8859-*
```

To use a TrueType font, precede the TrueType filename with @, for example,

```
@times.ttf
```

Otherwise, specify a PostScript font, for example,

```
helvetica
```

-frame *<width>x<height>+<outer bevel width>+<inner bevel width>*

Lets you surround an image with an ornamental border. See the X Windows system manual at http://www.x.org for details about the specification.

Note: The color of the border is specified with the `-mattecolor` command line option.

-gamma *value*

Lets you specify the level of gamma correction for an image.

The same color image displayed on different workstations may look different because of differences in the display monitor. Use gamma correction to adjust for this color difference. Reasonable values range from 0.8–2.3.

You can apply separate gamma values to the red, green, and blue channels of an image with a gamma value list delineated with slashes, for example,

```
1.7/2.3/1.2
```

Use `+gamma` to set the image gamma level without actually adjusting the image pixels. This option is useful if the imagehas a known gamma that isn’t set as an image attribute, such as PNG images.

-geometry *<width>x<height>{!}{<}{>}{%}*

Lets you specify the size and location of an image window. See the X Windows system manual at http://www.x.org for details about the geometry specification. By default, the window size is the image size. You specify its location when you map it.

The width and height, by default, are maximum values. That is, the image is expanded or contracted to fit the width and height value while maintaining the aspect ratio of the image.

Append an exclamation mark to the geometry to force the image size to exactly the size you specify. For example,

```
640x480!
```

sets the image width to 640 pixels and height to 480. If you specify one factor only, both the width and height assume that value.

To specify a percentage width or height instead, append %. The image size is multiplied by the width and height percentages to obtain the final image dimensions. To increase the size of an image, use a value greater than 100 (e.g., 125%). To decrease an image's size, use a percentage less than 100.

Use > to change the dimensions of the image only if its size exceeds the geometry specification. If the image dimension is smaller than the geometry you specify, < resizes the image. For example, if you specify

```
640x480>
```

and the image size is 512x512, the image size does not change. However, if the image is 1024x1024, it's resized to 640x480.

Tip! There are 72 pixels per inch in PostScript coordinates.

-implode *amount*

amount

-interlace *type*

Lets you specify one of the following interlacing schemes:

- none (default)

- line
- plane
- partition

Interlace also lets you specify the *type* of interlacing scheme for raw image formats such as RGB or YUV.

Interlace Types

Scheme	Description
none	does not interlace (e.g., RGBRGBRGBRGBRGBRGB...)
line	uses scanline interlacing (e.g., RRR...GGG...BBB...RRR...GGG...BBB...)
plane	uses plane interlacing (e.g., RRRRRR...GGGGGG...BBBBBB...)
partition	similar to plane except that different planes are saved to individual files (e.g., image.R, image.G, and image.B)

Tip! Use `line`, or `plane` to create an interlaced GIF or progressive JPEG image.

-label *name*

Lets you assign a label to an image.

-layer *type*

- red
- green
- blue
- matte

-linewidth *value*

Lets you set the width of a line. See -draw for details.

-loop *iterations*

iterations

-map *type*

Lets you display an image using one of the following standard colormap types:

- best
- default
- gray
- red
- green
- blue

The X server must support the colormap you choose, otherwise an error occurs. For *type* specify `list` and display searches the list of colormap types in top-to-bottom order until one is located. For one way of creating standard colormaps see *xstdcmap*, an X11 client program that's available with an X11 distribution.

-matte

Lets you store the matte channel (i.e., the transparent channel) if an image has one.

-median *radius*

radius .

-modulate *value*

```
-modulate 20/-10
```

-monochrome

Lets you transform an image to black and white.

-negate

Lets you apply color inversion to an image.

The red, green, and blue intensities of an image are negated. Use `+negate` to negate only the grayscale pixels of the image.

-noise

Lets you add noise to or reduce noise in an image.

The principal function of the noise peak elimination filter is to smooth the objects within an image without losing edge information and without creating undesired structures.

The algorithm replaces a pixel with its next neighbor in value within a 3 x 3 window, if this pixel is noise. A pixel is defined as noise if and only if the pixel is a maximum or minimum within the 3 x 3 window.

Use `+noise` followed by a noise type to add noise to an image. Choose from the following noise types:

- Uniform
- Gaussian
- Multiplicative
- Impulse
- Laplacian
- Poisson

-normalize

Lets you transform an image to span the full range of color values using this contrast enhancement technique.

-opaque *color*

-pen

-page *<width>x<height>{+-}<x offset>{+-}<y offset>{!}{<}{>}{%}*

Lets you set the size and location of an image canvas. Use this option to specify the dimensions of a

- PostScript page in dots per inch (dpi) or a
- TEXT page in pixels

This option is used in concert with `-density`.

The choices for a PostScript page are

Postscript Page Sizes

Media	Size (pixel width by pixel height)
11x17	792 1224
Ledger	1224 792
Legal	612 1008
Letter	612 792
LetterSmall	612 792
ArchE	2592 3456
ArchD	1728 2592

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
ArchC	1296 1728
ArchB	864 1296
ArchA	648 864
A0	2380 3368
A1	1684 2380
A2	1190 1684
A3	842 1190
A4	595 842
A4Small	595 842
A5	421 595
A6	297 421
A7	210 297
A8	148 210
A9	105 148

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
A10	74 105
B0	2836 4008
B1	2004 2836
B2	1418 2004
B3	1002 1418
B4	709 1002
B5	501 709
C0	2600 3677
C1	1837 2600
C2	1298 1837
C3	918 1298
C4	649 918
C5	459 649
C6	323 459

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
Flsa	612 936
Flse	612 936
HalfLetter	396 612

You can specify the page size by media (e.g. , A4, Ledger, etc.). Otherwise, `-page` behaves much like `-geometry` (e.g., `-page letter+43+43>`).

- To position a GIF image, use

  ```
  -page {+-}<x offset>{+-}<y offset>
  ```

  for example,

  ```
  -page +100+200
  ```

For a PostScript page, the image is sized as in `-geometry` and positioned relative to the lower-left hand corner of the page by *{+-}<x offset>{+-}<y offset>*. The default page dimension for a TEXT image is 612x792.

- To position a TEXT page, use

  ```
  -page 612x792>
  ```

  to center the image within the page.

Tip! If the image size exceeds the PostScript page, it's reduced to fit the page.

-paint *radius*

radius .

-pen *color*

Lets you set the color of the font or opaque color. See -draw for details. See the X Windows system manual at http://www.x.org for details about the *color* specification.

-pointsize *value*

Lets you specify the point size of a PostScript font.

-quality *value*

Lets you specify one of the following compression levels:

- JPEG with a *value* from 0–100 (i.e., worst to best); the default is 75
- MIFF with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)

- PNG with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)

The following are valid filter types:

- 0 for none; used for all scanlines
- 1 for sub; used for all scanlines
- 2 for up; used for all scanlines
- 3 for average; used for all scanlines
- 4 for Paeth; used for all scanlines
- 5 for adaptive filter; used when quality is greater than 50 and the image doesn't have a colormap; otherwise no filtering is used
- 6 or higher for adaptive filtering; used with minimum-sum-of-absolute-values

Note: The default is quality is 75—nearly the best compression with adaptive filtering.

For more information, see the PNG specification (RFC 2083) at http://www.w3.org/pub/WWW/TR.

-region *<width>x<height>{+-}<x offset>{+-}<y offset>*

`-region`

-rotate *degrees{<}{>}*

Applies Paeth image rotation to the image.

Use **>** to rotate the image only if its width exceeds the height. If the image width is less than its height, **<** rotates the image.

For example, if you have an image size of 480x640 and you specify

```
-90>
```

the image is not rotated by the specified angle. However, if the image is 640x480, it's rotated by -90 degrees.

Note: Empty triangles left over from rotating the image are filled with the color defined as bordercolor (class BorderColor). See the X Windows system manual at http://www.x.org for details.

-roll *{+-}<x offset>{+-}<y offset>*

Lets you roll an image vertically or horizontally. See the X Windows system manual at http://www.x.org for details about the geometry specification.

A negative x offset rolls the image left to right. A negative y offset rolls the image top to bottom.

-sample *geometry*

-geometry

-scene *value*

Lets you specify the image scene number.

-seed *value*

-segment *value*

Lets you eliminate insignificant clusters.

The number of pixels in each cluster must exceed the cluster threshold to be considered valid.

-shade *<azimuth>x<elevation>*

azimuth *elevation* `+shade`

-sharpen *factor*

Lets you sharpen an image. Specify *factor* as a percentage of enhancement from 0.0–99.9%.

-shear *<x degrees>x<y degrees>*

Lets you create a parallelogram by shearing (i.e., sliding) an image along its x or y axis by a positive or negative shear angle.

An x-direction shear slides an edge along the x axis, while a y-direction shear slides an edge along the yaxis. The amount of the shear is controlled by the shear angle. For x-direction shears, *x degrees* is measured relative to the yaxis. For y-direction shears, *y degrees* is measured relative to the x axis.

Empty triangles left over from shearing the image are filled with the color defined as bordercolor (class BorderColor). See the X Windows system manual at http://www.x.org for details.

-size *<width>x<height>{+offset}{!}{%}*

Lets you specify the width and height of a raw image whose dimensions are unknown, such as GRAY, RGB, or CMYK.

In addition to *width* and *height*, use `-size` to skip any header information in the image or tell the number of colors in a MAP image file, for example,

```
-size 640x512+256
```

-solarize *factor*

Lets you negate all pixels above a threshold level. Specify *factor* as a percentage of the intensity threshold from 0 - 99.9%.

Note: This option produces a solarization effect seen when exposing a photographic film to light during the development process.

-spread *amount*

Lets you displace image pixels by a random amount.

Amount defines the size of the neighborhood around each pixel from which to choose a candidate pixel to swap.

-swirl *degrees*

Lets you swirl image pixels about the center of an image.

Degrees defines the tightness of the swirl.

-transparency *color*

Lets you make a specified color in an image transparent.

-texture *filename*

Lets you specify a file, which contains a texture, to tile onto an image's background.

-threshold *value*

Lets you create a bi-level image such that any pixel whose intensity is equal to or greater than the threshold *value* you specify is reassigned the maximum intensity. Otherwise, it's reassigned the the minimum intensity.

-treedepth *value*

Lets you choose an optimal tree depth for the color reduction algorithm. Normally, *value* is 0 or 1.

An optimal depth generally provides the best representation of the source image with the fastest computational speed and the least amount of memory. However, the default depth is inappropriate for some images. To assure the best representation try values between 2 and 8. See Appendix D, Quantize for details.

Note: The `-colors` or `-monochrome` option is required for treedepth to take effect.

-undercolor *<undercolor factor>x<black-generation factor>*

Lets you control undercolor removal and black generation on CMYK images (i.e., images to be printed on a four-color printing system).

You can control the amount of cyan, magenta, and yellow to remove from your image and the amount of black to add to it. The standard undercolor removal is 1.0x1.0. You'll frequently get better results though if the percentage of black you add to your image is slightly higher than the percentage of C, M, and Y you remove from it. For example, you might try 0.5x0.7.

-verbose

Lets you print the following detailed information about an image:

- image name
- image size
- image depth
- image format
- image comment
- image scene number
- image class (DirectClass or PseudoClass)
- total unique colors
- number of seconds to read and transform the image
- whether a matte is associated with the image

- the number of runlength packets

-view *string*

-wave *<amplitude>x<wavelength>*

Lets you alter an image along a sine wave.

Specify *amplitude* and *wavelength* to affect the characteristics of the wave.

Segmenting Images

```
-segment
```

- Build a histogram, one for each color component of the image.
- For each histogram, successively apply the scale- space filter and build an interval tree of zero crossings in the second derivative at each scale. Analyze this scale-space ``fingerprint'' to determine which peaks or valleys in the histogram are most predominant.

- The fingerprint defines intervals on the axis of the histogram. Each interval contains either a minima or a maxima in the original signal. If each color component lies within the maxima interval, that pixel is considered ``classified'' and is assigned an unique class number.

- Any pixel that fails to be classified in the above thresholding pass is classified using the fuzzy c-Means technique. It is assigned to one of the classes discovered in the histogram analysis phase. The fuzzy c-Means technique attempts to cluster a pixel by finding the local minima of the generalized within group sum of squared error objective function. A pixel is assigned to the closest class of which the fuzzy membership has a maximum value.

Pattern Recognition

Chapter 10

Identify

Overview

Identify describes the format and characteristics of one or more image files. It will also report whether an image is incomplete or corrupt. The information displayed for an image includes the following:

- the scene number
- file name
- image width and height
- whether the image is colormapped
- the number of colors in the image
- the number of bytes in the image
- the image format (JPEG, PNM, etc.)
- the number of seconds it took to read and process the image

```
images/aquarium.miff 640x480 PseudoClass 256c 308135b MIFF 1s
```

```
-verbose
```

```
Image: images/aquarium.miff
    class: PseudoClass
    colors: 256
    signature: eb5dca81dd93ae7e6ffae99a5275a53e
    matte: False
    geometry: 640x480
    depth: 8
    bytes: 308135
    format: MIFF
    comments:
Imported from MTV raster image:  aquarium.mtv
```

Syntax

```
identify file [ file ... ]
```

Identify Options

-cache_threshold *value*

number of megabytes available to the pixel cache.

Image pixels are stored in memory until 80 megabytes of memory have been consumed. Subsequent pixel operations are cached on disk. Operations to memory are significantly faster but if your computer does not have a sufficient amount of free memory you may want to adjust this threshold value.

-ping

Lets you determine image characteristics efficiently.

This is a less memory-intensive way to query whether an image exists and what its size is.

Note: Only the size of the first image in a multiframe image file is returned.

-size *<width>x<height>{+offset}{!}{%}*

Lets you specify the width and height of a raw image whose dimensions are unknown, such as GRAY, RGB, or CMYK.

In addition to *width* and *height*, use `-size` to skip any header information in the image or tell the number of colors in a MAP image file, for example,

```
-size 640x512+256
```

-verbose

Lets you print the following detailed information about an image:

- image name
- image size
- image depth

- image format
- image comment
- image scene number
- image class (DirectClass or PseudoClass)
- total unique colors
- number of seconds to read and transform the image
- whether a matte is associated with the image
- the number of runlength packets

Chapter 11

Combine

Overview

Combine lets you combine two or more images into a new image.

Syntax

```
combine [ options… ] image composite [ mask ] combined
```

Examples

- To combine a image of a cockatoo with a perch, use

  ```
  combine cockatoo.miff perch.ras composite.miff
  ```

- To compute the difference between images in a series, use

  ```
  combine -compose difference series.1 series.2
  åÂdifference.miff
  ```

- To combine a image of a cockatoo with a perch starting at location (100,150), use

```
combine -geometry +100+150 cockatoo.miff perch.ras
åÂcomposite.miff
```

- To tile a logo across your image of a cockatoo, use

```
convert +shade 30x60 cockatoo.miff mask.miff
combine -compose bumpmap -tile logo.gif cockatoo.miff mask.miff composite.miff
```

- To combine a red, green, and blue color plane into a single composite image, try

```
combine -compose ReplaceGreen red.png green.png red-green.png
combine -compose ReplaceBlue red-green.png blue.png cmposite.png
```

Combine Options

-blend *value*

Blend lets you blend two images a given percentage.

-cache_threshold *value*

number of megabytes available to the pixel cache.

Image pixels are stored in memory until 80 megabytes of memory have been consumed. Subsequent pixel operations are cached on disk. Operations to memory are significantly faster but if your computer does not have a sufficient amount of free memory you may want to adjust this threshold value.

-colors *value*

Lets you specify the preferred number of colors in an image.

The actual number of colors in the image may be fewer than you specify, but will never be more.

Note: This is a color reduction option. Duplicate and unused colors will be removed if an image has fewer unique colors than you specify. See Appendix D, Quantize for more details. The options `-dither`, `-colorspace`, and `-treedepth` affect the color reduction algorithm.

-colorspace *value*

Lets you specify the type of colorspace.

- GRAY
- OHTA
- RGB
- Transparent
- XYZ

- YCbCr
- YIQ
- YPbPr
- YUV
- CMYK

Color reduction by default, takes place in the RGB color space. Empirical evidence suggests that distances in color spaces such as YUV or YIQ correspond to perceptual color differences more closely than distances in RGB space. These color spaces may give better results when color reducing an image. See Appendix D, Quantize for details.

Note: The transparent colorspace is unique. It preserves the matte channel of the image if it exists.

Tip! The `-colors` or `-monochrome` option is required for the transparent option to take effect.

-comment *string*

Lets you annotate an image with a comment.

By default, each image is commented with its file name. Use this option to assign a specific comment to the image.

Optionally you can include the image filename, type, width, height, or scene number in the label by embedding special format characters. The following table shows these characters and their values.

Special Format Characters

Special Character	Value
%b	file size
%d	directory
%e	filename extention
%f	filename
%h	height
%i	input filename
%l	label
%m	magick
%n	number of scenes
%o	output filename
%p	page number
%q	quantum depth

Special Format Characters

Special Character (Cont.)	Value
%s	scene number
%t	top of filename
%u	unique temporary filename
%w	width
%x	x resolution
%y	y resolution
\n	newline
\r	carriage return

For example,

```
-comment "%m:%f %wx%h"
```

produces for an image—titled bird.miff whose width is 512 and height is 480—the comment

```
MIFF:bird.miff 512x480
```

Note: If the first character of *string* is `@`, the image comment is read from a file titled by the remaining characters in the string.

-compose *operator*

Lets you specify the type of image composition.

By default, each of the composite image pixels are replaced by the corresponding image tile pixel. You can choose an alternate composite operation. Each operator's behavior is described below.

Composition Operators

This operator…	Results in…
over	the union of the two image shapes, with the *composite image* obscuring the *image* in the region of overlap
in	*composite image* cut by the shape of the *image*; none of the image data of image will be in the result
out	*composite image* with the shape of the *image* cut out
atop	the same shape as image *image*, with *composite image* obscuring *image* where the image shapes overlap; (**Note:** This differs from *over* because the portion of *composite image* outside *image's* shape does not appear in the result.)

Composition Operators

This operator… (Cont.)	Results in…
xor	the image data from both *composite image* and *image* that is outside the overlap region; the overlap region will be blank
plus	just the sum of the image data; output values are cropped to 255 (no overflow); this operation is independent of the matte channels
minus	*composite image* minus *image*, with underflow cropped to 0; the matte channel is ignored (set to 255, full coverage)
add	*composite image* plus *image*, with overflow wrapping around (mod 256)
subtract	*composite image* minus *image*, with underflow wrapping around (mod 256); the add and subtract operators can be used to perform reversible transformations
difference	The result of abs (*composite image* minus *image*); this is useful for comparing two very similar images
bumpmap	*image* shaded by *composite image*
replace	*image* replaced with *composite image*; here the matte information is ignored

The image compositor requires a matte or alpha channel in the image for some operations. This extra channel usually defines a mask that represents a sort of a cookie-cutter for the image.

This is the case when matte is 255 (full coverage) for pixels inside the shape, 0 outside, and between 0 and 255 on the boundary. For certain operations, if *image* does not have a matte channel, it's initialized with 0 for any pixel matching in color to pixel location (0,0). Otherwise it's 255.

Note: To work properly, *borderwidth* must be 0.

-compress *type*

Lets you specify one of the following types of image compression:

- None
- Bip
- Fax
- Group 4
- JPEG
- LZW
- RunlengthEncoded
- Zip

Specify

```
+compress
```

to store the binary image in an uncompressed format. The default is the compression type of the specified image file.

-density *<width>x<height>*

Lets you specify in pixels the vertical and horizontal resolution of an image.

This option lets you specify an image density when decoding a PostScript or Portable Document page. The default is 72 pixels per inch in the horizontal and vertical direction.

-display *host:display[.screen]*

Specifies the X server to contact. See the X Windows system manual at http://www.x.org for details about the specification.

-displace *<horizontal scale>x<vertical scale>*

Lets you shift image pixels as defined by a displacement map. With this option, a composite image is used as a displacement map.

In the displacement map

- black is a maximum positive displacement
- white is a maximum negative displacement

- middle gray is neutral

The displacement is scaled to determine the pixel shift. By default, the displacement applies to both the horizontal and vertical directions. However, if you specify `mask`, the composite image is the horizontal X displacement and mask is the vertical Y displacement.

-display *host:display[.screen]*

Specifies the X server to contact. See the X Windows system manual at http://www.x.org for details about the specification.

-dispose

Lets you specify one of the following GIF disposal methods:

GIF Disposal Methods

This method…	Specifies…
0	no disposal specified
1	do not dispose between frames
2	overwrite frame with background color from header
3	overwrite with previous frame

-dither

Lets you apply Floyd/Steinberg error diffusion to an image.

Dithering trades intensity resolution for spatial resolution by averaging the intensities of several neighboring pixels. You can use this option to improve images that suffer from severe contouring when reducing colors.

Note: The `-colors` or `-monochrome` option is required for dithering to take effect.

Tip! Use `+dither` to render PostScript without text or graphic aliasing.

-font *name*

Font lets you specify the font to use when annotating an image with text.

If the font is a fully-qualified X server font name, the font is obtained from an X server, for example,

```
-*-helvetica-medium-r-*-*-12-*-*-*-*-*-iso8859-*
```

To use a TrueType font, precede the TrueType filename with @, for example,

```
@times.ttf
```

Otherwise, specify a PostScript font, for example,

```
helvetica
```

-geometry *<width>x<height>{!}{<}{>}{%}*

Lets you specify the size and location of an image window. See the X Windows system manual at http://www.x.org for details about the geometry specification. By default, the window size is the image size. You specify its location when you map it.

The width and height, by default, are maximum values. That is, the image is expanded or contracted to fit the width and height value while maintaining the aspect ratio of the image.

Append an exclamation mark to the geometry to force the image size to exactly the size you specify. For example,

```
640x480!
```

sets the image width to 640 pixels and height to 480. If you specify one factor only, both the width and height assume that value.

To specify a percentage width or height instead, append **%**. The image size is multiplied by the width and height percentages to obtain the final image dimensions. To increase the size of an image, use a value greater than 100 (e.g., 125%). To decrease an image's size, use a percentage less than 100.

Use **>** to change the dimensions of the image only if its size exceeds the geometry specification. If the image dimension is smaller than the geometry you specify, **<** resizes the image. For example, if you specify

```
640x480>
```

and the image size is 512x512, the image size does not change. However, if the image is 1024x1024, it's resized to 640x480.

Tip! There are 72 pixels per inch in PostScript coordinates.

-gravity *direction*

Lets you specify the direction an image gravitates within a tile. See the X Windows system manual at http://www.x.org for details about the gravity specification.

-geometry

direction

-interlace *type*

Lets you specify one of the following interlacing schemes:

- none (default)
- line
- plane
- partition

Interlace also lets you specify the *type* of interlacing scheme for raw image formats such as RGB or YUV.

Interlace Types

Scheme	Description
none	does not interlace (e.g., RGBRGBRGBRGBRGBRGB...)
line	uses scanline interlacing (e.g., RRR...GGG...BBB...RRR...GGG...BBB...)
plane	uses plane interlacing (e.g., RRRRRR...GGGGGG...BBBBBB...)
partition	similar to plane except that different planes are saved to individual files (e.g., image.R, image.G, and image.B)

Tip! Use `line`, or `plane` to create an interlaced GIF or progressive JPEG image.

-matte

Lets you store the matte channel (i.e., the transparent channel) if an image has one.

-monochrome

Lets you transform an image to black and white.

-negate

Lets you apply color inversion to an image.

The red, green, and blue intensities of an image are negated. Use `+negate` to negate only the grayscale pixels of the image.

-page *<width>x<height>{+-}<x offset>{+-}<y offset>{!}{<}{>}{%}*

Lets you set the size and location of an image canvas. Use this option to specify the dimensions of a

- PostScript page in dots per inch (dpi) or a
- TEXT page in pixels

This option is used in concert with `-density`.

The choices for a PostScript page are

Postscript Page Sizes

Media	Size (pixel width by pixel height)
11x17	792 1224
Ledger	1224 792

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
Legal	612 1008
Letter	612 792
LetterSmall	612 792
ArchE	2592 3456
ArchD	1728 2592
ArchC	1296 1728
ArchB	864 1296
ArchA	648 864
A0	2380 3368
A1	1684 2380
A2	1190 1684
A3	842 1190
A4	595 842
A4Small	595 842

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
A5	421 595
A6	297 421
A7	210 297
A8	148 210
A9	105 148
A10	74 105
B0	2836 4008
B1	2004 2836
B2	1418 2004
B3	1002 1418
B4	709 1002
B5	501 709
C0	2600 3677
C1	1837 2600

Postscript Page Sizes

Media (Cont.)	Size (pixel width by pixel height)
C2	1298 1837
C3	918 1298
C4	649 918
C5	459 649
C6	323 459
Flsa	612 936
Flse	612 936
HalfLetter	396 612

You can specify the page size by media (e.g. , A4, Ledger, etc.). Otherwise, `-page` behaves much like `-geometry` (e.g., `-page letter+43+43>`).

- To position a GIF image, use

  ```
  -page {+-}<x offset>{+-}<y offset>
  ```

  for example,

  ```
  -page +100+200
  ```

For a PostScript page, the image is sized as in `-geometry` and positioned relative to the lower-left hand corner of the page by *{+-}<x offset>{+-}<y offset>*. The default page dimension for a TEXT image is 612x792.

- To position a TEXT page, use

  ```
  -page 612x792>
  ```

  to center the image within the page.

Tip! If the image size exceeds the PostScript page, it's reduced to fit the page.

-quality *value*

Lets you specify one of the following compression levels:

- JPEG with a *value* from 0–100 (i.e., worst to best); the default is 75
- MIFF with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)
- PNG with a *value* from 0–100 (i.e., worst to best); sets the amount of image compression (quality/10) and filter-type (quality % 10)

The following are valid filter types:

- 0 for none; used for all scanlines
- 1 for sub; used for all scanlines

- 2 for up; used for all scanlines
- 3 for average; used for all scanlines
- 4 for Paeth; used for all scanlines
- 5 for adaptive filter; used when quality is greater than 50 and the image doesn't have a colormap; otherwise no filtering is used
- 6 or higher for adaptive filtering; used with minimum-sum-of-absolute-values

Note: The default is quality is 75—nearly the best compression with adaptive filtering.

For more information, see the PNG specification (RFC 2083) at http://www.w3.org/pub/WWW/TR.

-scene *value*

Lets you specify the image scene number.

-size *<width>x<height>{+offset}{!}{%}*

Lets you specify the width and height of a raw image whose dimensions are unknown, such as GRAY, RGB, or CMYK.

In addition to *width* and *height*, use `-size` to skip any header information in the image or tell the number of colors in a MAP image file, for example,

```
-size 640x512+256
```

-stereo

Lets you combine two images to create a stereo anaglyph.

The left side of the stereo pair is saved as the red channel of the output image. The right side is saved as the green channel.

Note: You need red-blue stereo glasses to properly view the stereo image.

-tile *<width>x<height>*

Lets you specify the number of tiles to appear in each row and column of a composite image.

Specify the numbr of tiles per row with *width* and the number of tiles per column with *height*. For example, if you want one tile in each row and up to 10 tiles in the composite image, use

```
-tile 1x10
```

The default is five tiles in each row and four tiles in each column of the composite.

-treedepth *value*

Lets you choose an optimal tree depth for the color reduction algorithm. Normally, *value* is 0 or 1.

An optimal depth generally provides the best representation of the source image with the fastest computational speed and the least amount of memory. However, the default depth is inappropriate for some images. To assure the best representation try values between 2 and 8. See Appendix D, Quantize for details.

Note: The `-colors` or `-monochrome` option is required for treedepth to take effect.

Using Mask

If combined already exists, you will be prompted to overwrite it.

Chapter 12

PerlMagick

Overview

PerlMagick is an objected-oriented Perl interface to ImageMagick. You can use it to read, manipulate, or write an image or image sequence from within a Perl script. This makes it very suitable for web CGI scripts.

For either Perl script or CGI scripts to work, you must have the following installed on your system:

- ImageMagick 5.1.0 or later
- Perl 5.002 or later

Note: Perl version 5.005_02 or later is required for PerlMagick to work on an NT system.

There are a number of useful scripts available to show you the value of PerlMagick. You can do web-based image manipulation and conversion with MogrifyMagick, or use L-systems to create images of plants using mathematical constructs. Finally , you can navigate through collections of thumbnail images and select an image to view with the WebMagick Image Navigator.

An object-oriented Python interface to ImageMagick is also available, see PythonMagick at http://starship.skyport.net/crew/zack/pymagick/.

Installing PerlMagick

Instructions for installing PerlMagick are organized by platform in the following sections.

Installing for Unix

ImageMagick must already be installed on your system.

Note: For Unix, you typically need to be root to install the software. There are ways around this. Consult the Perl manual pages for more information.

1 Download the PerlMagick distribution from ftp://ftp.wizards.dupont.com/pub/ImageMagick/perl.

2 Unpack the distribution by typing the following at the system prompt:

```
gunzip -c PerlMagick-5.10.tar.gz | tar -xvf - cd PerlMagick
```

3 Edit Makefile.PL and change LIBS and INC to include the appropriate path information to the required libMagick library.

Note: You will also need paths to the JPEG, PNG, TIFF, etc. delegates if they were included with your installed version of ImageMagick.

4 Type the following to build and install PerlMagick:

```
perl Makefile.PL make make install
```

Installing for Windows NT/95/98

ImageMagick must already be installed on your system. The ImageMagick source distribution for Windows NT is also required and you must have the `nmake` from the Visual C++ or J++ development environment.

1 Copy \bin\IMagick.dll and \bin\X11.dll to a directory in your dynamic load path, such as c:\perl\site\5.00502.

2 Type

```
cd PerlMagick
copy Makefile.nt Makefile.PL
perl Makefile.PL
nmake
nmake install
```

Running the Regression Tests

1 To verify a correct installation, type

```
make test
```

Use `nmake test` under Windows. A few demonstration scripts are available to exercise many of the functions PerlMagick can perform.

2 Type

```
cd demo
make
```

You are now ready to use the PerlMagick methods from within your Perl scripts.

Using PerlMagick within PerlScripts

Any script that uses PerlMagick methods must first define the methods within its namespace and instantiate an image object. Do this with:

```
use Image::Magick;
$image=Image::Magick->new;
```

The new method takes the same parameters as *SetAttribute*. For example,

```
$image=Image::Magick->new(size=>'384x256');
```

Next you'll want to

- read an image or image sequence,
- manipulate it, then
- display or write it.

The remainder of this chapter is divided into the following sections:

- Reading and Writing an Image defines the input and output methods for PerlMagick.

- Setting an Image Attribute identifies methods that affect the way an image is read or written.
- Manipulating an Image provides a list of methods you can use to transform an image.
- Getting an Image Attribute describes how to retrieve an attribute for an image.
- Creating an Image Montage provides details about tiling your images as thumbnails on a background.
- Miscellaneous Methods describes methods that don't neatly fit into any of the above categories.

Destroying PerlMagick Objects

Once you're finished with a PerlMagick object you should consider destroying it. Each image in an image sequence is stored in virtual memory. This can potentially add up to mega-bytes of memory. After you destroy a PerlMagick object, memory is returned for use by other Perl methods. The recommended way to destroy an object is with `undef`.

```
undef $image
```

To delete all the images but retain the `Image::Magick` object use

```
undef @$image
```

To delete a single image from a multi-image sequence, use

```
undef $image->[x];
```

The next section illustrates how to use various PerlMagick methods to manipulate an image sequence.

Some of the PerlMagick methods require external programs such as Ghostscript. This may require an explicit path in your PATH environment variable to work properly. For example,

```
$ENV{PATH}='/bin:/usr/bin:/usr/local/bin';
```

Examples

The following are an examples of scripts to get you started.

- The following script reads three images, crops them, and writes a single image as a GIF animation sequence.

```
#!/usr/local/bin/perl
use Image::Magick;

my($image, $x);

$image = Image::Magick->new;
$x = $image->Read('girl.gif', 'logo.gif', 'rose.gif'); warn "$x" if "$x";

$x = $image->Crop(geometry=>'100x100+100+100');
warn "$x" if "$x";

$x = $image->Write('x.gif');
warn "$x" if "$x";
```

- In many cases you may want to access individual images of a sequence. The next example illustrates how this is done:

```
#!/usr/local/bin/perl
use Image::Magick;
```

```
my($image, $p, $q);

$image = new Image::Magick;
$image->Read('x1.gif');
$image->Read('j*.jpg');
$image->Read('k.miff[1, 5, 3]');
$image->Contrast;
for ($x = 0; $image->[x]; $x++)
{
 $image->[x]->Frame('100x200') if $image->[x]->Get('magick') eq 'GIF';
 undef $image->[x] if $image->[x]->Get('columns') < 100;
}
$p = $image->[1];
$p->Draw(pen=>'red', primitive=>'rectangle', points=>20, 20 100, 100');
$q = $p->Montage();
undef $image;
$q->Write('x.miff');
```

- Suppose you want to start out with a 100 x100 pixel black canvas with a red pixel in the center. Try

```
$image = Image::Magick->new;
$image->Set(size=>'100x100');
$image->ReadImage('xc:white');
$image->Set('pixel[49, 49]'=>'red');
```

- Perhaps you want to convert your color image to grayscale. Try

```
$image->Quantize(colorspace=>'gray');
```

- Other clever things you can do with PerlMagick objects include

```
$i = $#$p+1;# return the number of images associated with object p
```

```
push(@$q, @$p);# push the images from object p onto object q
undef @$p;# delete the images but not the object p
```

Reading and Writing an Image

Use the methods listed below to read, write, or display an image or image sequence.

Read, write, and display methods

Read/Write Methods/Description	Parameters	Return Value
Read reads an image or image sequence.	one or more filenames	the number of images read
Write writes an image or image sequence.	filename	the number of images written
Display displays an image or image sequence to an X server.	server name	the number of images displayed
Animate animates an image sequence to an X server.	server name	the number of images animated

For convenience, the *Write*, *Display*, and *Animate* methods can take any parameter *SetAttribute* recognizes. For example,

```
$image->Write(filename=>'image.png', compress=>'None');
```

Use `-` as the filename to method *Read* to read from standard in or to method *Write* to write to standard out, for example,

```
binmode STDOUT; $image->Write('gif:-');
```

Examples

- To read an image in the GIF format from a PERL filehandle, use

```
$image = Image::Magick->new(magick=>'GIF');
open(DATA, 'image.gif');
$image->Read(file=>DATA);
close(DATA);
```

- To write an image in the PNG format to a PERL filehandle, use

```
$filename = "image.png";
open(DATA, ">$filename");
$image->Write(file=>DATA, filename=>$filename); c
lose(DATA);
```

- You can optionally add `Image` to any method name. For example, `ReadImage` is an alias for method `Read`.

Manipulating an Image

Once you create an image with method `ReadImage`, for example, you may want to operate on it. The following is an example of a call to an image manipulation method:

```
$image->Crop(geometry=>'100x100+10+20');
```

```
$image->[x]->Frame("100x200");
```

The following table shows additional image manipulation methods you can call.

Image Manipulation Methods

Image Manipulation Method/Description	Parameters
AddNoise adds noise to an image.	noise=>{Uniform, Gaussian, Multiplicative, Impulse, Laplacian, Poisson}
Annotate annotates an image with text.	text=>*string*, font=>*string*, pointsize=>*integer*, density=>*geometry*, box=>*colorname*, pen=>*colorname*, geometry=>*geometry*, server=>{*string*, *@filename*}, gravity=>{*NorthWest*, *North*, *NorthEast*, *West*, *Center*, *East*, *SouthWest*, *South*, *SouthEast*}, x=>*integer*, y=>*integer*,degrees=>*double*
Blur blurs an image.	factor=>*percentage*
Border surrounds an image with a colored border.	geometry=>*geometry*, width=>*integer*, height=>*integer*, x=>*integer*, y=>*integer*
Charcoal simulates a charcoal drawing.	factor=>*percentage*
Chop chops an image.	geometry=>*geometry*, width=>*integer*, height=>*integer*, x=>*integer*, y=>*integer*
Clone makes a copy of an image.	n/a

Image Manipulation Methods

Image Manipulation Method/Description (Cont.)	Parameters
Coalesce merges a sequence of images.	n/a
ColorFloodfill changes the color value of any neighboring pixel that matches the color of the target pixel. If you specify a border color, the color value is changed for any neighboring pixel that isn't that color.	geometry=>*geometry*, x=>*integer*, y=>*integer*, *pen*=>*colorname*, bordercolor=>*colorname*
Colorize colorizes an image with the pen's color.	color=>*colorname*, pen=>*colorname*
Comment adds a comment to an image.	string
Composite composites one image onto another.	compose=>{Over, In, Out, Atop, Xor, Plus, Minus, Add, Subtract, Difference, Bumpmap, Replace, ReplaceRed, ReplaceGreen, ReplaceBlue, ReplaceMatte, Blend, Displace}, image=>*image-handle*, geometry=>*geometry*, x=>*integer*, y=>*integer*, gravity=>{*NorthWest*, *North*, *NorthEast*, *West*, *Center*, *East*, *SouthWest*, *South*, *SouthEast*}

Image Manipulation Methods

Image Manipulation Method/Description (Cont.)	Parameters
Condense compresses an image to take up the least amount of memory.	n/a
Contrast enhances or reduces the image contrast.	sharpen=>{True, False}
Crop crops an image.	geometry=>*geometry*, width=>*integer*, height=>*integer*, x=>*integer*, y=>*integer*
Deconstruct break down an image sequence into constituent parts.	n/a
Despeckle displaces the image colormap by an amount.	amount=>*integer*
Draw annotates an image with one or more graphic primitives.	primitive=>{point, Line, Rectangle, FillRectangle, Circle, FillCircle, Ellipse, FillEllipse, Polygon, FillPolygon, Color, Matte, Text, Image, *@filename*}, points=>*string*, method=>{*Point*, *Replace*, *Floodfill*, *FillToBorder*, *Reset*}, pen=>*colorname*, bordercolor=>*colorname*, linewidth=>*integer*, server=>*string*
Edge detects edges in an image.	factor=>*percentage*
Emboss embosses an image.	n/a

Image Manipulation Methods

Image Manipulation Method/Description (Cont.)	Parameters
Enhance applies a digital filter to enhance a noisy image.	n/a
Equalize performs a histogram equalization to an image.	n/a
Flip creates a mirror image by reflecting the image scanlines vertically.	n/a
Flop creates a mirror image by reflecting the image scanlines horizontally.	n/a
Frame surrounds an image with an ornamental border.	geometry=>*geometry*, width=>*integer*, height=>*integer*, inner=>*integer*, outer=>*integer*, color=>*colorname*
Gamma gamma corrects an image.	gamma=>*double*, red=>*double*, green=>*double*, bue=>*double*
Implode implodes image pixels about the image center.	factor=>*percentage*
Label assigns a label to an image.	string

Image Manipulation Methods

Image Manipulation Method/Description (Cont.)	**Parameters**
Layer extracts a layer from an image.	layer={Red, Green, Blue, Matte}
Magnify doubles the size of an image.	n/a
Map chooses a particular set of colors from an image.	image=>*image-handle*, dither={True, False}
MatteFloodfill change the matte value of any pixel that matches the color of the target pixel and is a neighbor. If you specify a border color, the matte value is changed for any neighbor pixel that's not that color.	geometry=>*geometry*, width=>*integer*, height=>*integer*, matte=>*integer*, border=>*colorname*
MedianFilter replace each pixel with the median color in the neighborhood.	radius=>*integer*
Minify reduces the size of an image by half.	n/a
Modulate varies the brightness, saturation, and hue of an image.	brightnes=>*double*, saturation=>*double*, hue=>*double*
Negate applies color inversion to an image.	gray=>{True, False}

Image Manipulation Methods

Image Manipulation Method/Description (Cont.)	**Parameters**
Normalize transforms an image to span the full range of color values.	n/a
OilPaint simulates an oil painting.	color=>*colorname*, pen=>*colorname*
Opaque changes the color to the pen color in the image.	color=>*colorname*, pen=>*colorname*
Quantize is the preferred number of colors in an image.	colors=>*integer*, colorspace=>{RGB, Gray, Transparent, OHTA, XYZ, YCbCr, YIQ, YPbPr, YUV, CMYK}, treedepth=>*integer*, dither=>{True, False}, measure_error=>{True, False}, global_colormap=>{True, False}
Raise lightens or darkens image edges to create a 3D effect.	geometry=>*geometry*, width=>*integer*, height=>*integer*, x=>*integer*, y=>*integer*, raise=>{True, False}
ReduceNoise adds or reduces the noise in an image.	n/a
Roll rolls an image vertically or horizontally.	geometry=>*geometry*, x=>*integer*, y=>*integer*
Rotate rolls an imag vertically or horizontally.	degrees=>*double*, crop=>{True, False}, sharpen=>{True, False}

Image Manipulation Methods

Image Manipulation Method/Description (Cont.)	**Parameters**
Sample scales an image with pixel sampling.	geometry=>*geometry*, width=>*integer*, height=>*integer*
Scale scales an image to a specified size.	geometry=>*geometry*, width=>*integer*, height=>*integer*
Segment segments an image by analyzing the histograms of color components adn identifying units that are homogeneous.	colors=>*integer*, colorspace=>{RGB, Gray, Transparent, OHTA, XYZ, YCbCr, YIQ, YPbPr, YUV, CMYK}, verbose=>{True, False}, cluster=>*double*, smooth=>*double*
Shade shades an image using a distant light source.	geometry=>*geometry*, azimuth=>*double*, elevation=>*double*, color=>{True, False}
Sharpen sharpens an image.	factor=>*percentage*
Shear shears an image along the X or Y axis by a positive or negative shear angle.	geometry=*geometry*, x=>*double*, y=>*double*, crop=>{True, False}
Signature generates an MD5 signature for an image.	n/a
Solarize negates all pixels above a threshold level.	factor=>*percentage*

Image Manipulation Methods

Image Manipulation Method/Description (Cont.)	Parameters
Spread displaces image pixels by a random amount.	amount=>*integer*
Stereo combines two images and produces a simgle image that's the composite of a left and right image of a stereo pair.	image=>*image-handle*
Stegano hides a digital watermark in an image.	image=>*image-handle*, offset=>*integer*
Swirl swirls image pixels about the center.	degrees=>*double*
Texture specifies name of a texture to tile onto an image background.	filename=>*string*
Threshold thresholds an image.	threshold=>*integer*
Transform crops or resizes an image with a fully-qualified geometry specification.	crop=>*geometry*, geometry=>*geometry*, filter->{Point, Box, Triangle, Hermite, Hanning, Hamming, Blackman, Gaussian, Quadratic, Cubic, Catrom, Mitchell, Lanczos, Bessel, Sinc}

Image Manipulation Methods

Image Manipulation Method/Description (Cont.)	Parameters
Transparent makes the specified color transparent in an image.	color=>*colorname*
Trim removes from an image edges that are the background color.	n/a
Wave alters an image along a sine wave.	geometry=>*geometry*., amplitude=>*double*, wavelength=>*double*
Zoom scales an image to a specified size. Use blur > 1 for blurry or < 1 for sharp.	geometry=>*geometry*, width=>*integer*, height=>*integer*, filter=>{Point, Box, Triangle, Hermite, Hanning, Hamming, Blackman, Gaussian, Quadratic, Cubic, Catrom, Mitchell, Lanczos, Bessel, Sinc}, blur=>*double*

Note: A `geometry` parameter is a short cut for the width and height parameters, for example,

```
geometry=>'106x80'
```

is equivalent to width=>106, height=>80).

You can specify `@filename` in both `Annotate` and `Draw`. This reads the text or graphic primitive instructions from a file on disk. For example,

```
$image->Draw(pen=>'red', primitive=>'rectangle', points=>'20, 20 100, 100 40, 40 200,
Â200 60, 60 300, 300');
```

is eqivalent to

```
$image->Draw(pen=>'red', primitive=>'@draw.txt');
```

where `draw.txt` is a file on disk that contains

```
rectangle 20, 20 100, 100
rectangle 40, 40 200, 200
rectangle 60, 60 300, 300
```

The text parameter for methods *Annotate, Comment, Draw,*and *Label*can include the image filename, type, width, height, or other image attribute by embedding the following special format characters:

Special Format Characters

Special Character	Value
%b	file size
%d	directory
%e	filename extention
%f	filename
%h	height

Special Format Characters

Special Character (Cont.)	Value
%i	input filename
%l	label
%m	magick
%n	number of scenes
%o	output filename
%p	page number
%q	quantum depth
%s	scene number
%t	top of filename
%u	unique temporary filename
%w	width
%x	x resolution
%y	y resolution
\n	newline

Special Format Characters

Special Character (Cont.)	Value
\r	carriage return

Optionally you can add `Image` to any method name. For example, *TrimImage* is an alias for method *Trim*.

Most of the attributes listed above have an analog in `convert`. See Chapter 8, Convert for a detailed description of these attributes.

Setting an Image Attribute

Use method *Set* to set an image attribute. For example,

```
$image->Set(dither=>'True');
$image->[$x]->Set(delay=>3);
```

The following are image attributes you can set.

Read/Write Image Attributes

Attribute/Description	Values
`adjoin` joins images into a single mult-image file	True, False
`antialias` removes pixel aliasing	True, False
`background` is the image's background color	*string*
`blue_primary` is the chromaticity of the blue primary point (e.g., 0.15, 0.06)	*x-value, y-value*
`bordercolor` sets the images border color	*string*
`cache_threshold` is the amount of memory that must be consumed by image pixels before they are cached to disk. The default is 80 megabytes. If your computer has limited memory resources, consider lowering this value.	*integer*
`colormap[i]` is the color name (e.g., red) or hex value (e.g., #ccc) at position *i*	*string*
`colorspace` is the type of colorspace	RGB, CMYK
`compress` is the type of image compression	none, BZip, Fax, JPEG, LZW, Runlength, Zip

Read/Write Image Attributes

Attribute/Description (Cont.)	Values
`delay` is the number of 1/100ths of a second that must expire before displaying thenext image in a sequence	*integer*
`density` is te vertical and horizontal resolution of an image in pixels	*geometry*
`depth` is the image depth	*integer*
`dispose` is the GIF disposal method	*1, 2, 3, 4*
`dither` applies the Floyd/Steinberg error diffusion to an image	True, False
`display` specifies an X server to contact	*string*
`file` sets the image filehandle	*filehandle*
`filename` sets the image file name	*string*
`font` is used when annotating an image with text	*string*
`fuzz` specifies the distance within which colors are considered equal	*integer*
`green_primary` is the chromaticity of the green primary point (e.g., 0.3, 0.6)	*x-value, y-value*

Read/Write Image Attributes

Attribute/Description (Cont.)	Values
`interlace` is the type of interlacing scheme	None, Line, Plan, Partition
`iterations` adds a Netscape loop to a GIF animation	*integer*
`loop` adds a Netscape loop to a GIF animation	*integer*
`magick` sets theimage format	*string*
`mattecolor` sets the image matte color	*string*
`monochrome` transforms an image to black and white	*string*
`page` is the preferred size and location of an image canvas	Letter, Tabloid, Ledger, Legal, Statement, Executrive, A32, A4, A5, B4, B5, Folio, Quarto, 10x14, or geometry
`pen` is the color name (e.g., red) or hex value (e.g., #ccc) for annotating or changing an opaque color	*color*
`pixel[x,y]` is the color name (e.g., red) or hex value (e.g., #ccc) at poisition (*x*,*y*)	*string*
`pointsize` is te size of the PostScript of TrueType font	*integer*

Read/Write Image Attributes

Attribute/Description (Cont.)	Values
`preview` is the type of preview for the Preview image format	Rotate, Shear, Roll, Hue, Saturation, Brightness, Gamma, Spiff, Dull, Grayscale, Quantize, Despeckle, ReduceNoise, AddNoise, Sharpen, Blue, Threshold, EdgeDetect, Spread, Solarize, Shade, Raise, Segment, Swirl, Implode, Wave, OilPaint, CharcoalDrawing, JPEG
`quality` is the JPEG/MIFF/PNG compression level	*integer*
`red_primary` is the chromaticity of the red primary point (e.g., 0.64, 0.33)	*x-value, y-value*
`rendering_intent` is the type ofrendering intent	Undefined, Saturation, Percetual, Absolute, Relative
`scene` is the image scene number	*integer*
`subimage` is part of an image sequence	*integer*
`subrange` is the number of images relative to the base image	*integer*

Read/Write Image Attributes

Attribute/Description (Cont.)	Values
`server` specifies an X server to contact	*string*
`size` is the width and height of a raw image	*string*
`tile` is the tile name of an image	*string*
`texture` is the name of the texture totile ontoan image background	*string*
`verbose` prints detailed information about an image	True, False
`white_primary` is the chromaticity of the white primary point (e.g., 0.3127, 0.329)	*x-value, y-value*

Note: The geometry parameter is a short cut for the width and height parameters, for example,

```
geometry=>'106x80'
```

is equivalent to

```
width=>106, height=>80).
```

SetAttribute is an alias for method *Set*.

Most of the attributes listed in the table above have an analog in `convert`. See Chapter 8, Convert for a detailed description of these attributes.

Getting an Image Attribute

Use method *Get* to get an image attribute. For example,

```
($a, $b, $c) = $image->Get('colorspace', 'magick', 'adjoin');
$width = $image->[3]->Get('columns');
```

In addition to all the attributes listed in Setting an Image Attribute, you can get these additional attributes:

Read-Only Image Attributes

Attribute/Description	Values
`base_columns` is the base image width (before transformations)	integer
`base_filename` is the base image file name (before transformations)	string
`base_rows` is the base image height (before transformations)	integer
`class` is the image class	Direct, Pseudo
`colors` is the number of unique colors in an image	integer
`comment` is the image comment	string
`columns` is the image width	integer

Read-Only Image Attributes

Attribute/Description (Cont.)	Values
`directory` is the tile names from within an image montage	string
`filesize` is the number of bytes of an image on disk	integer
`format` gets the descriptive image format	string
`gamma` is the gamma level of an image	double
`geometry` is the image geometry	string
`height` is the number of row or heith of an image	integer
`label` is the image label	string
`matte` is the image transparency (true means an image has tranparency)	True, False
`mean` is the mean error per pixel computed whn animage is color reduced	double
`montage` is the tile size and offset within an image montagw	geometry
`normalized_max` is the nomralized max error per pixel computed when an image is color reduced	double
`normalized_mean` is the normalized mean error per pixel coputed when an image is color reduced	double

Read-Only Image Attributes

Attribute/Description (Cont.)	Values
`pakcketsize` is the numbe rof byptes in each pixel packet	integer
`packets` is the number of runlength-encoded packets in an image	integer
`rows` is the number of rows or height of an image	integer
`signature` is the MD5 signature associated with an image	string
`text` is any text associated with an image	string
`type` is the image type	bilevel, greyscale, palette, true color, true color with transparency, color separation
`units` is the units of resolution	string
`view` is the FlashPix viewing parameters	string
`width` is the number of columns or width of an image	integer
`x-resolution` is the x resolution of an image	integer
`y-resolution` is they resolution of an image	integer

GetAttribute is an alias for method *Get*.

Most of the attributes listed above have an analog in convert. See Chapter 8, Convert for a detailed description of these attributes.

Creating an Image Montage

Use method *Montage* to create a composite image by combining several separate images. The images are tiled on the composite image with the name of the image optionally appearing just below the individual tile. For example,

```
$image->Montage(geometry=>'160x160', tile=>'2x2', texture=>'granite:');
```

Montage parameters you can set are:

Montage Options

Parameter/Description	Values
`background` is the X11 color name	color
`borderwidth` is the image border width	integer
`compose` is the composite operator	Over, In, Out, Atop, Xor, Plus, Minus, Add, Subtract, Difference, Bumpmap, Replace, MatteReplace, Mask, Blend, Displace
`filename` is the name of a montage image	string
`font` is the X11 font name	string
`frame` surrounds an image with an ornamental border	geometry
`geometry` is the preferred tile and border size of each tile of a composit image	geometry

Montage Options

Parameter/Description (Cont.)	Values
`gravity` is the direction an image gravitates within a tile	NorthWest, North, NorthEast, West, Center, East, SouthWest, South, SouthEast
`label` assigns a label to an image	string
`mode` specifies thumbnail framing options	Frame, Unframe, Concatenate
`pen` is the color for annotation text	string
`pointsize` is the size of a PostScript or TrueType font	integer
`shadow` adds a shadow beneath a tile to simulate depth	True, False
`texture` is the name of a texture to tile onto an image background	string
`tile` is the number of tiles per row and column	geometry
`title` assigns a title to an image montage	string

Montage Options

Parameter/Description (Cont.)	Values
`transparent` specifies the color to make transparent within an image	string

Note: The geometry parameter is a short cut for the width and height parameters, for example,

```
geometry=>'106x80'
```

is equivalent to

```
width=>106, height=>80)
```

MontageImage is an alias for method *Montage*.

Most of the attributes listed in the table above have an analog in montage. See Chapter 7, Montage for a detailed description of these attributes.

Miscellaneous Methods

Append

The *Append* method appends a set of images. For example,

```
$x = $image->Append(stack=>{true,false});
```

appends all the images associated with object `$image`. All the specified images must have the same width or height. Same-width images are stacked top to bottom. Same-height images are stacked left to right. Rectangular images are stacked left to right when the stack parameter is `False`. When the parameter is True, rectangular images are stacked top to bottom.

Average

The *Average* method averages a set of images. For example,

```
$x = $image->Average();
```

averages all the images associated with object `$image`.

Morph

The *Morph* method morphs a set of images. Both the image pixels and size are linearly interpolated to give the appearance of a metamorphosis from one image to the next, for example,

```
$x = $image->Morph(frames=>integer);
```

where `frames` is the number ofintermediate images to generate. The default is `1`.

Mogrify

The *Mogrify* method is a single entry point for the image manipulation methods (see Manipulating an Image). The parameters are the name of a method followed by any parameters the method may require. For example, these calls are equivalent:

```
$image->Crop('340x256+0+0');
$image->Mogrify('crop', '340x256+0+0');
```

MogrifyRegion

The *MogrifyRegion* method applies a transformation to a region of an image. It's similiar to *Mogrify* but it begins with a region's geometry. For example, suppose you want to brighten a 100x100 region of an image at location (40, 50):

```
$image->MogrifyRegion('100x100+40+50', 'modulate', brightness=>50);
```

Clone

The *Clone* method copies a set of images. For example,

```
$p = $image->Clone();
```

copies all the images from object `$q` to `$p`.

Use this method for multi-image sequences. PerlMagick transparently creates a linked list from an image array. If two locations in the array point to the same object, the linked list goes into an infinite loop and your script will run continuously until it's interrupted. Instead of

```
push(@$images, $image);
push(@$images, $image); # warning duplicate object
```

use cloning to prevent an infinite loop, such as,

```
push(@$images, $image);
$clone=$image->Clone();
push(@$images, $clone); # same image but different object
```

Ping

Ping accepts one or more image file names and returns their respective width, height, size in bytes, and format (e.g. GIF, JPEG, etc.). For example,

```
($width, $height, $size, $format) = split(‘,’, $image->Ping(‘logo.gif’));
```

This is a more efficient and less memory-intensive way to query whether an image exists and what its characteristics are.

Note: Information about the first image only in a multi-frame image file is returned.

You can optionally add *Image* to any method name above. For example, *PingImage* is an alias for method *Ping*.

RemoteCommand

Use *RemoteCommand* to send a command to an already running *Display* or *Animate* application. The only parameter required is the name of the image file you want to display or animate.

QueryColor

The QueryColor method accepts one or more color names or hex values and returns their respective red, green, and blue color values:

```
($red, $green, $blue) = split(‘, ‘, $image->QueryColor(‘cyan’));
($red, $green, $blue) = split(‘, ‘, $image->QueryColor(‘#716bae’));
```

Troubleshooting

All successful PerlMagick methods return an undefined string context. If a problem occurs, an error is returned as a string with an embedded numeric status code.

- A status code of less than 400 is a warning. This means that the operation did not complete but was recoverable to some degree.
- A numeric code equal to or greater than 400 is an error and indicates the operation failed completely.

Errors are returned for the different methods as follows:

- Methods that return a number (e.g., *Read*, *Write*)

```
$x = $image->Read(...);
    warn “$x” if “$x”; # print the error message
    $x =~ /(\d+)/;
    print $1; # print the error number
    print 0+$x; # print the number of images read
```

- Methods that operate on an image (e.g., *Zoom*, *Crop*)

```
$x = $image->Crop(...);
    warn “$x” if “$x”; # print the error message
    $x =~ /(\d+)/;
    print $1; # print the error number
```

- Methods that return images (e.g., *Average*, *Montage*, *Clone*) should be checked for errors this way:

```
$x = $image->Montage(...);
    warn “$x” if !ref($x); # print the error message
    $x =~ /(\d+)/;
    print $1; # print the error number
```

Error messages look similar to

```
Error 400: Memory allocation failed
```

The following is a table of of errors and warning codes:

Errors And Warning Codes

Code	Mnemonic	Description
0	Success	method completed without error or warning
300	ResourceLimitWarning	a program resource is exhaused (e.g., not enough memory)
305	XSwerverWarning	an X resource is unavailable
310	OptionWarning	a command-line option was malformed
315	DelegateWarning	an ImageMagick *delegate* returned a warning
320	MissingDelegateWarning	the image type can't be read or written because the appropriate *delegate* is missing
325	CorruptImageWarning	the image file nay be corrupt
330	FileOpenWarning	the image file could not be opened
335	BlobWarning	a binary large object could not be allocated
340	CacheWarning	pixels could not be saved to the pixel cache
400	ResourceLimitError	a program resource is exhaused (e.g., not enough memory)

Errors And Warning Codes

Code	Mnemonic	Description
405	XServerError	an X resource is unavailable
410	OptionError	a command-line option was malformed
415	DelegateError	an ImageMagick *delegate* returned an error
420	MissingDelegateError	the image type can’t be read or written because the appropriate *delegate* is missing
425	CorruptImageError	the image file may be corrupt
430	FileOpenError	the image file could not be opened
435	BlobError	a binary large object could not be allocated
440	CacheError	pixels could not be saved to the pixel cache

You can use a numeric status code as follows:

```
$x = $image->Read(‘rose.gif’);
$x =~ /(\d+)/;
die “unable to continue” if ($1 == ResourceLimitError);
```

Chapter 13

Magick++

Overview

Magick++ provides a simple C++ API to the ImageMagick image processing librarywhich supports reading and writing a huge number of image formats as well as supporting a broad spectrum of traditional image processing operations. The ImageMagick C API is complex and the data structures are currently not documented. Magick++ provides access to most of the features available from the C API but in a simple object-oriented and well-documented framework.

Magick++ is intended to support commercial-grade application development. In order to avoid possible conflicts with the user's application, all symbols contained in Magick++ (included by the header <Magick++.h>) are scoped to the namespace Magick. Symbols from the ImageMagick C library are imported under the MagickLib namespace to avoid possible conflicts and ImageMagick macros are only included within the Magick++ implementation so they won't impact the user's application.

The core class in Magick++ is the Image class. The Image class provides methods to manipulate a single image frame (e.g. a JPEG image). Standard Template Library (STL) compatable algorithms and function objects are provided in order to manipulate multiple image frames or to read and write file formats which support multiple image frames (e.g. GIF animations, MPEG animations, and Postscript files).

The Image class supports reference-counted memory management which supports the semantics of an intrinsic variable type (e.g. 'int') with an extremely efficient operator = and copy constructor (only a pointer is assigned) while ensuring that the image data is replicated as required so that it the image may be modified without impacting earlier generations. Since the Image class manages heap memory internally, images are best allocated via C++ automatic (stack-based) memory allocation. This support allows most programs using Magick++ to be written without using any pointers, simplifying the implementation and avoiding the risks of using pointers.

The image class uses a number of supportive classes in order to specify arguments. Colors are specified via the Color class. Colors specified in X11-style string form are implicitly converted to the Color class. Geometry arguments (those specifying width, height, and/or x and y offset) are specified via the Geometry class. Similar to the Color class, geometries specified as an X11-style string are implicitly converted to the Geometry class. Two dimensional drawable objects are specified via the Drawable class. Drawable objects may be provided as a single object or as a list of objects to be rendered using the current image options. Montage options (a montage is a rendered grid of thumbnails in one image) are specified via the Montage class.

Errors are reported using C++ exceptions derived from the Exception class, which is itself derived from the standard C++ exception class. Exceptions are reported synchronous with the operation and are caught by the first matching try block as the stack is unraveled. This allows a clean coding style in which multiple related Magick++ commands may be executed with errors handled as a unit rather than line-by-line. Since the Image object provides reference-counted memory management, unreferenced images on the stack are automatically cleaned up, avoiding the potential for memory leaks.

Enumerations

Magick++ uses enumerations to specify method options or to return image format information. The available enumerations are shown in the following tables:

ClassType

ClassType specifies the image storage class.

ClassType

Enumeration	Description
UndefinedClass	Unset value.
DirectClass	Image is composed of pixels which represent literal color values.
PseudoClass	Image is composed of pixels which specify an index in a color palette.

ColorspaceType

The *ColorspaceType* enumeration is used to specify the colorspace that quantization (color reduction and mapping) is done under or to specify the colorspace when encoding an output image. Colorspaces are ways of describing colors to fit the requirements of a particular application (e.g. Television, offset printing, color monitors). Color reduction, by default, takes place in the *RGBColorspace*. Empirical evidence suggests that distances in color spaces such as *YUVColorspace* or *YIQColorspace* correspond to perceptual color differences more closely han do distances in RGB space. These color spaces may give better results when color reducing an image. Refer to *quantize* for more details.

When encoding an output image, the colorspaces *RGBColorspace*, *CMYKColorspace*, and *GRAYColorspace* may be specified. The *CMYKColorspace* option is only applicable when writing TIFF, JPEG, and Adobe Photoshop bitmap (PSD) files.

ColorspaceType

Enumeration	Description
UndefinedColorspace	Unset value.
RGBColorspace	Red-Green-Blue colorspace.
GRAYColorspace	
TransparentColorspace	The Transparent color space behaves uniquely in that it preserves the matte channel of the image if it exists.
OHTAColorspace	
XYZColorspace	
YCbCrColorspace	
YCCColorspace	
YIQColorspace	

ColorspaceType

Enumeration	Description
YPbPrColorspace	
YUVColorspace	Y-signal, U-signal, and V-signal colorspace. YUV is most widely used to encode color for use in television transmission.
CMYKColorspace	Cyan-Magenta-Yellow-Black colorspace. CYMK is a subtractive color system used by printers and photographers for the rendering of colors with ink or emulsion, normally on a white surface.
sRGBColorspace	

CompositeOperator

CompositeOperator is used to select the image composition algorithm used to compose a composite image with an image. By default, each of the composite image pixels are replaced by the corresponding image tile pixel. Specify *CompositeOperator* to select a different algorithm.

CompositeOperator

Enumeration	Description
UndefinedCompositeOp	Unset value.

CompositeOperator

Enumeration	Description
OverCompositeOp	The result is the union of the the two image shapes with the composite image obscuring image in the region of overlap.
InCompositeOp	The result is a simply composite image cut by the shape of image. None of the image data of image is included in the result.
OutCompositeOp	The resulting image is composite image with the shape of image cut out.
AtopCompositeOp	The result is the same shape as image image, with composite image obscuring image there the image shapes overlap. Note that this differs from OverCompositeOp because the portion of composite image outside of image's shape does not appear in the result.
XorCompositeOp	The result is the image data from both composite image and image that is outside the overlap region. The overlap region will be blank.
PlusCompositeOp	The result is just the sum of the image data. Output values are cropped to 255 (no overflow). This operation is independent of the matte channels.
MinusCompositeOp	The result of composite image - image, with overflow cropped to zero. The matte chanel is ignored (set to 255, full coverage).
AddCompositeOp	The result of composite image + image, with overflow wrapping around (mod 256).
SubtractCompositeOp	The result of composite image - image, with underflow wrapping around (mod 256). The add and subtract operators can be used to perform reverible transformations.

CompositeOperator

Enumeration	Description
DifferenceCompositeOp	The result of abs(composite image - image). This is useful for comparing two very similar images.
BumpmapCompositeOp	The result image shaded by composite image.
ReplaceCompositeOp	The resulting image is image replaced with composite image. Here the matte information is ignored.
ReplaceRedCompositeOp	The resulting image is the red layer in image replaced with the red layer in composite image. The other layers are copied untouched.
ReplaceGreenCompositeOp	The resulting image is the green layer in image replaced with the green layer in composite image. The other layers are copied untouched.
ReplaceBlueCompositeOp	The resulting image is the blue layer in image replaced with the blue layer in composite image. The other layers are copied untouched.

CompositeOperator

Enumeration	Description
ReplaceMatteCompositeOp	The resulting image is the matte layer in image replaced with the matte layer in composite image. The other layers are copied untouched. The image compositor requires a matte, or alpha channel in the image for some operations. This extra channel usually defines a mask which represents a sort of a cookie-cutter for the image. This is the case when matte is 255 (full coverage) for pixels inside the shape, zero outside, and between zero and 255 on the boundary. For certain operations, if *image* does not have a matte channel, it is initialized with 0 for any pixel matching in color to pixel location (0,0), otherwise 255 (to work properly *borderWidth* must be 0).

CompressionType

CompressionType is used to express the desired compression type when encoding an image. Be aware that most image types only support a sub-set of the available compression types. If the compression type specified is incompatable with the image, ImageMagick selects a compression type compatable with the image type.

CompressionType

Enumeration	Description
UndefinedCompression	Unset value.

CompressionType

Enumeration	Description
NoCompression	No compression
BZipCompression	BZip (Burrows-Wheeler block-sorting text compression algorithm and Huffman coding) as used by bzip2 utilities
FaxCompression	CCITT Group 3 FAX compression
Group4Compression	CCITT Group 4 FAX compression (used only for TIFF)
JPEGCompression	JPEG compression
LZWCompression	Lempel-Ziv-Welch (LZW) compression (caution, patented by Unisys)
RunlengthEncodedCompression	Run-Length encoded (RLE) compression
ZipCompression	Lempel-Ziv compression (LZ77) as used in PKZIP and GNU gzip.

FilterType

FilterType is used to adjust the filter algorithm used when resizing images. Different filters experience varying degrees of success with various images and can take sigificantly different amounts of processing time. ImageMagick uses the *LanczosFilter* by default since this filter has been shown to provide the best results for most images in a reasonable amount of time. Other filter types (e.g. *TriangleFilter*) may execute much faster but may show artifacts when the image is re-sized or around diagonal lines. The only way to be sure is to test the filter with sample images.

FilterType

Enumeration	Enumeration
UndefinedFilter	Unset value.
PointFilter	Point Filter
BoxFilter	Box Filter
TriangleFilter	Triangle Filter
HermiteFilter	Hermite Filter
HanningFilter	Hanning Filter
HammingFilter	Hamming Filter
BlackmanFilter	Blackman Filter
GaussianFilter	Gaussian Filter

FilterType

Enumeration	Enumeration
QuadraticFilter	Quadratic Filter
CubicFilter	Cubic Filter
CatromFilter	Catrom Filter
MitchellFilter	Mitchell Filter
LanczosFilter	Lanczos Filter
BesselFilter	Bessel Filter
SincFilter	Sinc Filter

GravityType

GravityType specifies positioning of an object (e.g. text or image) within a bounding region (e.g. an image). Gravity provides a convenient way to locate objects irrespective of the size of the bounding region, in other words, you don't need to provide absolute coordinates in order to position an object. A common default for gravity is *NorthWestGravity* (top -left corner of region).

GravityType

Enumeration	Description
ForgetGravity	Don't use gravity.
NorthWestGravity	Position object at top-left of region.
NorthGravity	Postiion object at top-center of region
NorthEastGravity	Position object at top-right of region
WestGravity	Position object at left-center of region
CenterGravity	Position object at center of region
EastGravity	Position object at right-center of region
SouthWestGravity	Position object at left-bottom of region
SouthGravity	Position object at bottom-center of region
SouthEastGravity	Position object at bottom-right of region

ImageType

The *ImageType* enumeration indicates the type classification of the image.

ImageType

Enumeration	Description
UndefinedType	Unset value.
BilevelType	Monochrome image
GrayscaleType	Grayscale image
PaletteType	Indexed color (palette) image
TrueColorType	Truecolor image
MatteType	Truecolor with opacity image
ColorSeparationType	Cyan/Yellow/Magenta/Black (CYMK) image

InterlaceType

InterlaceType specifies the ordering of the red, green, and blue pixel information in the image. Interlacing is usually used to make image information available to the user faster by taking advantage of the space vs time tradeoff. For example, interlacing allows images on the Web to be recognizable sooner and satellite images to render with image resolution increasing over time.

Use *LineInterlace* or *PlaneInterlace* to create an interlaced GIF or progressive JPEG image.

InterlaceType

Enumeration	Description
UndefinedInterlace	Unset value.
NoInterlace	Don't interlace image (RGBRGBRGBRGBRGBRGB...)
LineInterlace	Use scanline interlacing (RRR...GGG...BBB...RRR...GGG...BBB...)
PlaneInterlace	Use plane interlacing (RRRRRR...GGGGGG...BBBBBB...)
PartitionInterlace	Similar to plane interlaing except that the different planes are saved to individual files (e.g. image.R, image.G, and image.B)

LayerType

LayerType is used as an argument when doing color separations. Use *LayerType* when extracting a layer from an image. *MatteLayer* is useful for extracting the opacity values from an image.

LayerType

Enumeration	Description
UndefinedLayer	Unset value.

LayerType

Enumeration	Description
RedLayer	Select red layer
GreenLayer	Select green layer
BlueLayer	Select blue layer
MatteLayer	Select matte (opacity values) layer

NoiseType

NoiseType is used as an argument to select the type of noise to be added to the image.

NoiseType

Enumeration	Description
UniformNoise	Uniform noise
GaussianNoise	Gaussian noise
MultiplicativeGaussianNoise	Multiplicative Gaussian noise
ImpulseNoise	Impulse noise
LaplacianNoise	Laplacian noise
PoissonNoise	Poisson noise

PaintMethod

PaintMethod specifies how pixel colors are to be replaced in the image. It is used to select the pixel-filling algorithm employed.

PaintMethod

Enumeration	Description
PointMethod	Replace pixel color at point.
ReplaceMethod	Replace color for all image pixels matching color at point.
FloodfillMethod	Replace color for pixels surrounding point until encountering pixel that fails to match color at point.
FillToBorderMethod	Replace color for pixels surrounding point until encountering pixels matching border color.
ResetMethod	Replace colors for **all** pixels in image with pen color.

RenderingIntent

Rendering intent is a concept defined by ICC Spec ICC.1:1998-09, "File Format for Color Profiles". ImageMagick uses *RenderingIntent* in order to support ICC Color Profiles.

From the specification: "Rendering intent specifies the style of reproduction to be used during the evaluation of this profile in a sequence of profiles. It applies specifically to that profile in the sequence and not to the entire sequence. Typically, the user or application will set the rendering intent dynamically at runtime or embedding time."

RenderingIntent

Enumeration	Description
UndefinedIntent	Unset value.
SaturationIntent	A rendering intent that specifies the saturation of the pixels in the image is preserved perhaps at the expense of accuracy in hue and lightness.
PerceptualIntent	A rendering intent that specifies the full gamut of the image is compressed or expanded to fill the gamut of the destination device. Gray balance is preserved but colorimetric accuracy might not be preserved.
AbsoluteIntent	Absolute colorimetric
RelativeIntent	Relative colorimetric

ResolutionType

By default, ImageMagick defines resolutions in pixels per inch. *ResolutionType* provides a means to adjust this.

ResolutionType

Enumeration	Description
UndefinedResolution	Unset value.
PixelsPerInchResolution	Density specifications are specified in units of pixels per inch (english units).
PixelsPerCentimeterResolution	Density specifications are specified in units of pixels per centimeter (metric units).

Exception

Exception represents the base class of objects thrown when ImageMagick reports an error. Magick++ throws C++ exceptions synchronous with the operation when an error is detected. This allows errors to be trapped within the enclosing code (perhaps the code to process a single image) while allowing the code to be written simply.

A try/catch block should be placed around any sequence of operations which can be considered a unit of work. For example, if your program processes lists of images and some of these images may be defective, by placing the try/catch block around the entire sequence of code that processes one image (including instantiating the image object), you can minimize the overhead of error checking while ensuring that all objects created to deal with that object are safely destroyed (C++ exceptions unroll the stack until the enclosing try block, destroying any created objects).

The pseudocode for the main loop of your program may look like:

```
for each image in list
  try {
    create image object
    read image
    process image
    save result
  }
  catch( ErrorFileOpen error )
  {
    process Magick++ file open error
  }
  catch( Exception error )
  {
    process any Magick++ error
}
```

```
catch( exception error )
{
  process any other exceptions derived from standard C++ exception
}
catch( ... )
{
  process *any* exception (last-ditch effort)
}
```

This catches errors opening a file first, followed by any Magick++ exception if the exception was not caught previously.

The *Exception* class is derived from the C++ standard *exception* class. This means that it contains a C++ string containing additional information about the error (e.g to display to the user). Obtain access to this string via the what() method. For example:

```
catch( Exception error_ )
    {
      cout << "Caught exception: " << error_.what() << endl;
    }
```

The classes *Warning* and *Error* derive from the *Exception* class. Exceptions derived from *Warning* are thrown to represent non-fatal errors which may effect the completeness or quality of the result (e.g. one image provided as an argument to montage is defective). In most cases, a *Warning* exception may be ignored by catching it immediately, processing it (e.g. printing a diagnostic) and continuing on. Exceptions derived from *Error* are thrown to represent fatal errors that can not produce a valid result (e.g. attempting to read a file which does not exist).

The specific derived exception classes are shown in the following tables:

Warning Exception Classes

Warning	Warning Description
WarningUndefined	Unspecified warning type.
WarningResourceLimit	A program resource is exhausted (e.g. not enough memory)
WarningXServer	An X resource is unavailable
WarningOption	An option was malformed or out of range
WarningDelegate	An ImageMagick delegate returned an error
WarningMissingDelegate	The image type can not be read or written because the appropriate Delegate is missing
WarningCorruptImage	The image file is corrupt (or otherwise can't be read)
WarningFileOpen	The image file could not be opened (permission problem, wrong file type, or does not exist).

Error Exception Classes

Error	Error Description
ErrorUndefined	Unspecified error type.

Error Exception Classes

Error	Error Description
ErrorResourceLimit	A program resource is exhausted (e.g. not enough memory)
ErrorXServer	An X resource is unavailable
ErrorOption	An option was malformed or out of range
ErrorDelegate	An ImageMagick delegate returned an error
ErrorMissingDelegate	The image type can not be read or written because the appropriate Delegate is missing
ErrorCorruptImage	The image file is corrupt (or otherwise can't be read)
ErrorFileOpen	The image file could not be opened (permission problem, wrong file type, or does not exist).

Color

Color is the base color class in Magick++. It is a simple container class for the raw red, green, blue, and alpha values scaled appropriately. Normally users will instantiate a class derived from Magick::Color which supports the color model that fits the needs of the application. The Magick::Color class may be constructed directly from an X11-style color string.

Available derived color specification classes are shown in the following table:

Derived Color Classes

Class	Representation
ColorRGB	Representation of RGB color with red, green, and blue specified as ratios (0 to 1)
ColorGray	Representation of grayscale RGB color (equal parts red, green, and blue) specified as a ratio (0 to 1)
ColorMono	Representation of grayscale RGB color (equal parts red, green, and blue) specified as a ratio (0 to 1)
ColorYUV	Representation of a color in the YUV colorspace

Color Class

The Color base class is not intended to be used directly. Normally a user will construct a derived class or inherit from this class. Color arguments must be scaled to the *Quantum* size (8 or16 bits depending on how ImageMagick was configured). The *ScaleDoubleToQuantum* and *ScaleQuantumToDouble* macros can aid with this task.

An alternate way to contruct the class is via an X11-compatable color specification string.

```
class Color
{
  friend class Image;
public:
  Color ( Quantum red_, Quantum green_, Quantum blue_ );
  Color ( const std::string x11color_ );
  Color ( const char * x11color_ );
  Color ( void );
  virtual ~Color ( void );

  // Does object contain valid color?
  void  isValid ( bool valid_ );
  bool  isValid ( void ) const;

  // Set color via X11 color specification string
  const Color&operator = ( std::string x11color_ );
  const Color&operator = ( const char * x11color_ );

  // Return X11 color specification string
  /* virtual */operator std::string() const;

protected:
```

```
  void  redQuantum ( Quantum red_ );
  Quantum redQuantum ( void ) const;

  void  greenQuantum ( Quantum green_ );
  Quantum greenQuantum ( void ) const;

  void  blueQuantum ( Quantum blue_ );
  Quantum blueQuantum ( void ) const;

};
```

ColorRGB

Representation of an RGB color. All color arguments have a valid range of 0.0 - 1.0.

```
class ColorRGB : public Color
{
public:
  ColorRGB ( double red_, double green_, double blue_ );
  ColorRGB ( const string x11color_ );
  ColorRGB ( void );
  /* virtual */ ~ColorRGB ( void );

  void  red ( double red_ );
  double red ( void ) const;

  void  green ( double green_ );
  double green ( void ) const;

  void  blue ( double blue_ );
  double blue ( void ) const;
};
```

ColorGray

Representation of a grayscale color (in RGB colorspace). Grayscale is simply RGB with equal parts of red, green, and blue. All double arguments have a valid range of 0.0 - 1.0.

```
class ColorGray : public Color
{
public:
  ColorGray ( double shade_ );
  ColorGray ( void );
  /* virtual */ ~ColorGray ();

  void           shade ( double shade_ );
  double         shade ( void ) const;
};
```

ColorMono

Representation of a black/white pixel (in RGB colorspace). Color arguments are constrained to 'false' (black pixel) and 'true' (white pixel).

```
class ColorMono : public Color
{
public:
  ColorMono ( bool mono_ );
  ColorMono ( void );
  /* virtual */ ~ColorMono ();

  void           mono ( bool mono_ );
  bool           mono ( void ) const;
```

```
};
```

ColorHSL

Representation of a color in Hue/Saturation/Luminosity (HSL) colorspace.

```
class ColorHSL : public Color
{
public:
  ColorHSL ( double hue_, double saturation_, double luminosity_ );
  ColorHSL ( );
  /* virtual */  ~ColorHSL ( );

  void           hue ( double hue_ );
  double         hue ( void ) const;

  void           saturation ( double saturation_ );
  double         saturation ( void ) const;

  void           luminosity ( double luminosity_ );
  double         luminosity ( void ) const;
};
```

ColorYUV

Representation of a color in YUV colorspace (commonly used to encode color for television transmission).

Argument ranges:

Y: 0.0 through 1.0

U: -0.5 through 0.5

V: -0.5 through 0.5

```
class ColorYUV : public Color
{
public:
  ColorYUV ( double y_, double u_, double v_ );
  ColorYUV ( void );
  /* virtual */ ~ColorYUV ( void );

  void          u ( double u_ );
  double        u ( void ) const;

  void          v ( double v_ );
  double        v ( void ) const;

  void          y ( double y_ );
  double        y ( void ) const;
};
```

Geometry

Geometry provides a convenient means to specify a geometry argument. The object may be initialized from a C string or C++ string containing a geometry specification. It may also be initialized by more efficient parameterized constructors.

X11 Geometry Specifications

X11 geometry specifications are in the form "<width>x<height>{+-}<xoffset>{+-}<yoffset>" (where *width*, *height*, *xoffset*, and *yoffset* are numbers) for specifying the size and placement location for an object.

The *width* and *height* parts of the geometry specification are measured in pixels. The *xoffset* and *yoffset* parts are also measured in pixels and are used to specify the distance of the placement coordinate from the left or right and top and bottom edges of the image, respectively. Both types of offsets are measured from the indicated edge of the object to the corresponding edge of the image. The X offset may be specified in the following ways:

X Offset

Xoffset	Placement
+xoffset	The left edge of the object is to be placed xoffset pixels in from the **left edge** of the image.
-xoffset	The right edge of the window is to be placed xoffset pixels in from the **right edge** of the image.

The Y offset has similar meanings:

Y Offset

Yoffset	Placement
+yoffset	The top edge of the object is to be yoffset pixels **below** the **top edge** of the image.
-yoffset	The bottom edge of the object is to be yoffset pixels **above** the **bottom edge** of the image.

Offsets must be given as pairs; in other words, in order to specify either *xoffset* or *yoffset* both must be present. Objects can be placed in the four corners of the image using the following specifications:

Offset Pairs

Offset	Placement
+0+0	upper left hand corner.
-0+0	upper right hand corner.
-0-0	lower right hand corner.
+0-0	lower left hand corner.

ImageMagick Geometry Extensions

ImageMagick has added a number of qualifiers to the standard geometry string for use when resizing images. The form of an extended geometry string is "<width>x<height>{+-}<xoffset>{+-}<yoffset>{%}{!}{<}{>}". Extended geometry strings should **only** be used **when resizing an image**. Using an extended geometry string for other applications may cause the API call to fail. The available qualifiers are shown in the following table:

Geometry Exensions

Qualifier	Description
%	Interpret width and height as a **percentage** of the current size.
!	Resize to width and height **exactly**, loosing original aspect ratio.
<	Resize only if the image is **smaller** than the geometry specification.
>	Resize only if the image is **greater** than the geometry specification.

Postscript Page Size Geometry Extension

Any geometry string specification supplied to the Geometry contructor is considered to be a Postscript page size nickname if the first character is not numeric. The Geometry constructor converts these page size specifications into the equivalent numeric geometry string specification (preserving any offset component) prior to conversion to the internal object format. Postscript page size specifications are short-hand for the pixel geometry required to fill a page of that size. Since the 11x17 inch page size used in the US starts with a digit, it is not supported as a Postscript page size nickname. Instead, substitute the geometry specification "792x1224>" when 11x17 output is desired.

An example of a Postscript page size specification is "letter+43+43>"

The following table shows the available postscript page size nicknames and their equivalents..

Page Size Specifications

Nickname	Equivalent Geometry Specification
Ledger	1224x792>
Legal	612x1008>
Letter	612x792>
LetterSmall	612x792>
ArchE	2592x3456>
ArchD	1728x2592>
ArchC	1296x1728>
ArchB	864x1296>
ArchA	648x864>

Page Size Specifications

Nickname	Equivalent Geometry Specification
A0	2380x3368>
A1	1684x2380>
A2	1190x1684>
A3	842x1190>
A4	595x842>
A4Small	595x842>
A5	421x595>
A6	297x421>
A7	210x297>
A8	148x210>
A9	105x148>
A10	74x105>
B0	2836x4008>
B1	2004x2836>

Page Size Specifications

Nickname	Equivalent Geometry Specification
B2	1418x2004>
B3	1002x1418>
B4	709x1002>
B5	501x709>
C0	2600x3677>
C1	1837x2600>
C2	1298x1837>
C3	918x1298>
C4	649x918>
C5	459x649>
C6	323x459>
Flsa	612x936>
Flse	612x936>
HalfLetter	396x612>

Geometry provides methods to initialize its value from strings, from a set of parameters, or via attributes. The methods available for use in Geometry are shown in the following table:

Geometry Methods

Method	Return Type	Signature(s)	Description
Geometry		unsigned int width_, unsigned int height_, unsigned int xOff_ = 0, unsigned int yOff_ = 0, bool xNegative_ = false, bool yNegative_ = false	Construct X11 geometry via explicit parameters.
		const string geometry_	Construct geometry from C++ string
		const char * geometry_	Construct geometry from C string
width	void	unsigned int width_	Width
	unsigned int	void	
height	void	unsigned int height_	Height
	unsigned int	void	
xOff	void	unsigned int xOff_	X offset from origin
	unsigned int	void	

Geometry Methods

Method	Return Type	Signature(s)	Description
yOff	void	unsigned int yOff_	Y offset from origin
	unsigned int	void	
xNegative	void	bool xNegative_	Sign of X offset negative? (X origin at right)
	bool	void	
yNegative	void	bool yNegative_	Sign of Y offset negative? (Y origin at bottom)
	bool	void	
percent	void	bool percent_	Width and height are expressed as percentages (%)
	bool	void	
aspect	void	bool aspect_	Resize without preserving aspect ratio (!)
	bool	bool	
greater	void	bool greater_	Resize if image is greater than size (>)
	bool	void	
less	void	bool	Resize if image is less than size (<)
	bool	bool	

Geometry Methods

Method	Return Type	Signature(s)	Description
isValid	void	bool isValid_	Object contains valid geometry.
	bool	void	
operator =	const Geometry&	const std::string geometry_	Set geometry via C++ string
		const char * geometry_	Set geometry via C string
operator string	std::string	Geometry&	Obtain C++ string representation of geometry
operator<<	std::ostream&	ostream& stream_, const Geometry& geometry_	Stream onto std::ostream

Drawable

Drawable provides a convenient interface for preparing vector, image, or text arguments for the Image::draw() method. Each instance of Drawable represents a single drawable object.

The following is an example of how Drawable might be used:

```
#include <Magick++.h>

using namespace std;
using namespace Magick;

int main(int argc,char **argv)
{
    // Create base image (white image of 600 by 400 pixels)
    Image image( "600x400", "xc:white" )

    // Set draw options
    image.penColor("red");
    image.lineWidth(5) ;

    // Draw a circle
    Drawable drawable;
    drawable.circle( 100,100, 150,150 );
    image.draw( drawable );

    // Draw a rectangle (re-use drawable object)
    drawable.rectangle( 200,200 300,300 );
```

```
        image.draw( drawable );

        // Display the result
        image.display( );
    }
```

Since Drawable is an object it may be saved in an array or a list for later (perhaps repeated) use. Drawable depends on the simple Coordinate class which represents a pair of x,y coodinates. The methods provided by the Coordinateclass are shown in the following table:

Coordinate Class Methods

Method	Signature	Description
Coordinate	void	Default Constructor
	double x_, double y_	Constructor, setting x & y
x	double x_	Set x coordinate
	void	Get x coordinate
y	double y_	Set y coordinate
	void	Get y coordinate

The methods available in the Drawable class are shown in the following table:

Drawable Class Methods

Method	Signature	Description
point	double x_, double y_	Draw a point using current pen color and thickness at coordinate
	Coordinate coordinate	
line	double startX_, double startY_, double endX_, double endY_	Draw a line using current pen color and thickness using starting and ending coordinates
	Coordinate startCoordinate_, Coordinate endCoordinate_	
rectangle	double upperLeftX_, double upperLeftY_, double lowerRightX_, double lowerRightY	Draw a rectangle using current pen color and thickness from upper-left coordinates to lower-right coordinates
	Coordinate upperLeftCoordinate_, Coordinate lowerRightCoordinate_	

Drawable Class Methods

Method	Signature	Description
fillRectangle	double upperLeftX_, double upperLeftY_, double lowerRightX_, double lowerRightY	Draw a filled rectangle using current pen color from upper-left coordinates to lower-right coordinates
	Coordinate upperLeftCoordinate_, Coordinate lowerRightCoordinate_	
circle	double originX_, double originY_, double perimX_, double perimY_	Draw a circle using current pen color and thicknews using specified origin and perimeter coordinates
	Coordinate originCoordinate_, Coordinate perimCoordinate_	

Drawable Class Methods

Method	Signature	Description
fillCircle	double originX_, double originY_, double perimX_, double perimY_	Draw a filled circle using current pen color, origin and perimeter coordinates
	Coordinate originCoordinate_, Coordinate perimCoordinate_	
ellipse	double originX_, double originY_, double width_, double height_, double arcStart_, double arcEnd_	Draw an ellipse using current pen color, pen thickness, specified origin, width & height, as well as specified start and end of arc in degrees.
	Coordinate originCoordinate_, double width_, double height_, double arcStart_, double arcEnd_	

Drawable Class Methods

Method	Signature	Description
fillEllipse	double originX_, double originY_, double width_, double height_, double arcStart_, double arcEnd_	Draw a filled ellipse using current pen color, specified origin, width & height, as well as specified start and end of arc in degrees.
	Coordinate originCoordinate_, double width_, double height_, double arcStart_, double arcEnd_	
polygon	const std::list<Coordinate> &coordinates_	Draw an arbitrary polygon using current pen color and pen thickness consisting of three or more coordinates contained in an STL list
fillPolygon	const std::list<Coordinate> &coordinates_	Draw an arbitrary filled polygon using current pen color and pen thickness consisting of three or more coordinates contained in an STL list
color	double x_, double y_, PaintMethod paintMethod_	Color image according to *paintMethod*. The *point* method recolors the target pixel. The *replace* method recolors any pixel that matches the color of the target pixel. *Floodfill* recolors any pixel that matches the color of the target pixel and is a neighbor, whereas *filltoborder* recolors any neighbor pixel that is not the border color. Finally, *reset* recolors all pixels.
	Coordinate coordinate_, PaintMethod paintMethod_	

Drawable Class Methods

Method	Signature	Description
matte	double x_, double y_, PaintMethod paintMethod_	Change the pixel matte value to transparent. The *point* method changes the matte value of the target pixel. The *replace* method changes the matte value of any pixel that matches the color of the target pixel. *Floodfill* changes the matte value of any pixel that matches the color of the target pixel and is a neighbor, whereas *filltoborder* changes the matte value of any neighbor pixel that is not the border color, Finally *reset* changes the matte value of all pixels.
	Coordinate coordinate_, PaintMethod paintMethod_	
text	double x_, double y_, std::string text_	Annotate image with text using current pen color, font, font pointsize, and box color (text background color), at specified coordinates. If text contains special format characters the image filename, type, width, height, or other image attributes may be incorporated in the text (see label()).
	Coordinate coordinate_, std::string text_	
image	double x_, double y_, const string &image_	Composite image (file) with image file at specified coordinates.
	Coordinate coordinate_, const std::string &image_	

Special Format Characters

The Magick::Image methods *annotate*, *draw*, *label*, and the template function *montageImages* support special format characters contained in the argument text. These format characters work similar to C's *printf*. Whenever a format character appears in the text, it is replaced with the equivalent attribute text. The available format characters are shown in the following table:

Special Format Characters

Format Character	Description
%b	file size
%d	directory
%e	filename extension
%f	filename
%h	height
%m	magick (e.g GIF)
%p	page number
%s	scene number
%t	top of filename

Special Format Characters

Format Character	Description
%w	width
%x	x resolution
%y	y resolution
\n	newline
\r	carriage return

Montage

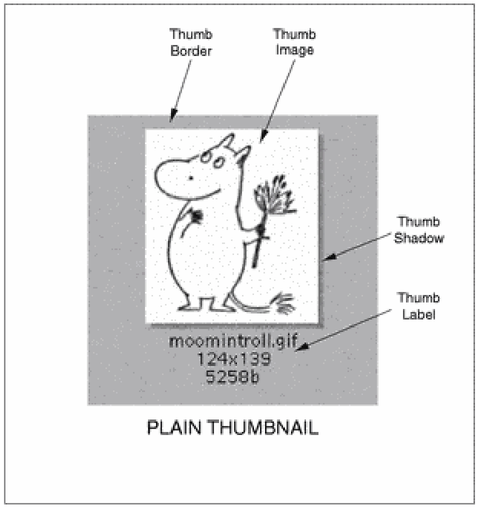
A montage is a single image which is composed of thumbnail images composed in a uniform grid. The size of the montage image is determined by the size of the individual thumbnails and the number of rows and columns in the grid.

The illustration shows a montage consisting of three columns and two rows of thumbnails rendered on a gray background.

Montages may be either "plain" (undecorated thumbnails) or "framed" (decorated thumbnails). In order to more easily understand the options supplied to *MontageImages()*, montage options are supplied by two different classes: *Montage* and *MontageFramed*.

Plain Montages

Montage is the base class to provide montage options and provides methods to set all options required to render simple (un-framed) montages. See *MontageFramed* if you would like to create a framed montage.



Un-framed (plain) thumbnails consist of four components: the thumbnail image, the thumbnail border, an optional thumbnail shadow, and an optional thumbnail label area as shown in the illustration.

Montage Methods

Method	Return Type	Signature(s)	Description
Montage		void	Default constructor
backgroundColor	void	const Color &backgroundColor_	Specifies the background color that thumbnails are imaged upon.
	Color	void	
compose	void	CompositeOperator compose_	Specifies the image composition algorithm for thumbnails. This controls the algorithm by which the thumbnail image is placed on the background. Use of OverCompositeOp is recommended for use with images that have transparency. This option may have negative side-effects for images without transparency.
	CompositeOperator compose_	void	
fileName	void	std::string fileName_	Specifies the image filename to be used for the generated montage images. To handle the case were multiple montage images are generated, a printf-style format may be embedded within the filename. For example, a filename specification of image%02d.miff names the montage images as image00.miff, image01.miff, etc.
	std::string	void	

Montage Methods

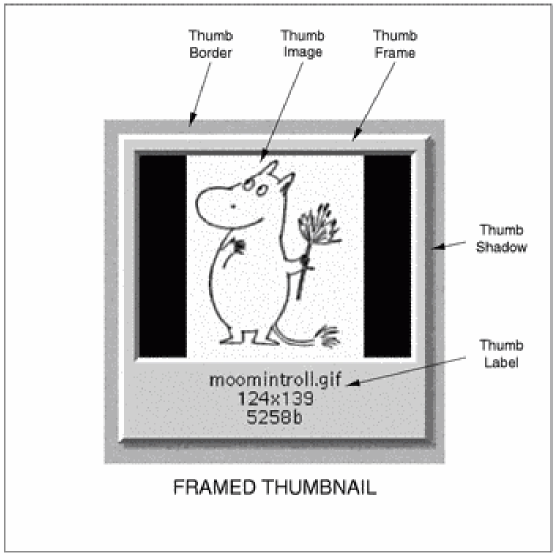
Method	Return Type	Signature(s)	Description
font	void	std::string font_	Specifies the thumbnail label font.
	std::string	void	
geometry	void	const Geometry &geometry_	Specifies the size of the generated thumbnail.
	Geometry	void	
gravity	void	GravityType gravity_	Specifies the thumbnail positioning within the specified geometry area. If the thumbnail is smaller in any dimension than the geometry, then it is placed according to this specification.
	GravityType	void	
label	void	std::string label_	Specifies the format used for the image label. Special format characters may be embedded in the format string to include information about the image.
	std::string	void	
penColor	void	const Color &pen_	Specifies the pen color to use for the label text.
	Color	void	
pointSize	void	unsigned int pointSize_	Specifies the thumbnail label font size.
	unsigned int	void	

Montage Methods

Method	Return Type	Signature(s)	Description
shadow	void	bool shadow_	Enable/disable drop-shadow on thumbnails.
	bool	void	
texture	void	std::string texture_	Specifies a texture image to use as montage background. The built-in textures "granite:" and "plasma:" are available. A texture is the same as a background image.
	std::string	void	
tile	void	const Geometry &tile_	Specifies the maximum number of montage columns and rows in the montage. The montage is built by filling out all cells in a row before advancing to the next row. Once the montage has reached the maximum number of columns and rows, a new montage image is started.
	Geometry	void	
transparentColor	void	const Color &transparentColor_	Specifies a montage color to set transparent. This option can be set the same as the background color in order for the thumbnails to appear without a background when rendered on an HTML page. For best effect, ensure that the transparent color selected does not occur in the rendered thumbnail colors.
	Color	void	

Framed Montages

MontageFramed provides the means to specify montage options when it is desired to have decorative frames around the image thumbnails. *MontageFramed* inherits from Montage and therefore provides all the methods of *Montage* as well as those shown in the table "MontageFramed Methods".



Framed thumbnails consist of four components: the thumbnail image, the thumbnail frame, the thumbnail border, an optional thumbnail shadow, and an optional thumbnail label area as shown in the illustration.

MontageFramed Methods

Method	Return Type	Signature(s)	Description
MontageFramed		void	Default constructor (enable frame via frameGeometry).
borderColor	void	const Color &borderColor_	Specifies the background color within the thumbnail frame.
	Color	void	
borderWidth	void	unsigned int borderWidth_	Specifies the border (in pixels) to place between a thumbnail and its surrounding frame. This option only takes effect if thumbnail frames are enabled (via *frameGeometry*) and the thumbnail geometry specification doesn't also specify the thumbnail border width.
	unsigned int	void	
frameGeometry	void	const Geometry &frame_	Specifies the geometry specification for frame to place around thumbnail. If this parameter is not specified, then the montage is un-framed.
	Geometry	void	
matteColor	void	const Color &matteColor_	Specifies the thumbnail frame color.
	Color	void	

Image

Image is the primary object in Magick++ and represents a single image frame (see design). The *STL interface* must be used to operate on image sequences or image formats which are comprized of multiple image frames. Various image manipulation operations may be applied to the image. Attributes may be set on the image to influence the operation of the manipulation operations. As a convenience, including `<Magick++.h>` is sufficient in order to use the complete Magick++ API. The Magick++ API is enclosed within the Magick namespace so you must either add the prefix "`Magick::`" to each class/enumeration name or add the statement "`using namespace Magick;`" after including the `Magick++.h` header.

Image is very easy to use. For example, here is a the source to a program which reads an image, crops it, and writes it to a new file (the exception handling is optional):

```
#include <Magick++.h>
#include <iostream>
using namespace std;
using namespace Magick;
int main(int argc,char **argv)
{
  try {
    // Create an image object and read an image
    Image image( "girl.gif" );

    // Crop the image to specified size
    image.crop("100x100+100+100" ); // Geometry implicitly initialized by char *

    // Write the image to a file
    image.write( "x.gif" );
  }
```

```
    catch( Exception error_ )
      {
        cout << "Caught exception: " << error_.what() << endl;
        return 1;
      }
    return 0;
  }
```

The following is the source to a program which illustrates the use of Magick++'s efficient reference-counted assignment and copy-constructor operation which minimizes use of memory and eliminates unncessary copy operations. The program accomplishes the following:

1 Read master image.

2 Assign master image to second image.

3 Zoom second image to the size 640x480.

4 Assign master image to a third image.

5 Zoom third image to the size 800x600.

6 Write the second image to a file.

7 Write the third image to a file.

```
#include <Magick++.h>
#include <iostream>
using namespace std;
using namespace Magick;
```

```
int main(int argc,char **argv)
{
    Magick::Image master("horse.jpg");
    Magick::Image second = master;
    second.zoom("640x480");
    Magick::Image third = master;
    third.zoom("800x600");
    second.write("horse640x480.jpg");
    third.write("horse800x600.jpg");
    return 0;
}
```

During the entire operation, a maximum of three images exists in memory and the image data is never copied.

The following is the source for another simple program which creates a 100 by 100 pixel white image with a red pixel in the center and writes it to a file:

```
#include <Magick++.h>
using namespace std;
using namespace Magick;
int main(int argc,char **argv)
{
    Image image( "100x100", "xc:white" );
    image.pixelColor( 49, 49, "red" );
    image.write( "red_pixel.png" );
    return 0;
}
```

If you wanted to change the color image to grayscale, you could simply add the lines:

```
image.quantizeColorSpace( GRAYColorspace );
image.quantize( options );
```

prior to writing the image.

Image Manipulation Methods

Image supports access to all the single-image (versus image-list) manipulation operations provided by the ImageMagick library. These operations are shown in the following table:

Image Manipulation Methods

Method	Signature(s)	Description
addNoise	NoiseType noiseType_	Add noise to image with specified noise type.
annotate	const std::string &text_, const Geometry &location_	Annotate image (render text on image) at specified location.
	const std:string text_, const Geometry &location_, GravityType gravity_	Annotate image (render text on image) at specified location and influenced by gravity.
	const std::string &text_, const Geometry&location_, GravityType gravity_, double degrees_	Annotate image (render text on image) starting at specified location, influenced by gravity, and rendered at specified rotation angle.
	const std::string &text_, GravityType gravity_ = NorthWestGravity	Annotate image (render text on image) at location implied by gravity.
blur	double factor_	Blur image with specified blur factor

Image Manipulation Methods

Method	Signature(s)	Description
border	const Geometry &geometry_ = "6x6+0+0"	Border image (add border to image). The color of the border is specified by the *borderColor* attribute.
charcoal	double factor_ = 50	Charcoal effect image (looks like charcoal sketch)
chop	const Geometry &geometry_	Chop image (remove vertical or horizontal subregion of image)
colorize	const Color &opaqueColor_, const Color &penColor_	Colorize opaque color in image using pen color
composite	const Image &compositeImage_, int xOffset_, int yOffset_, CompositeOperator compose_ = InCompositeOp	Compose an image onto another at specified offset and using specified algorithm
	const Image &compositeImage_, const Geometry &offset_, CompositeOperator compose_ = InCompositeOp	

Image Manipulation Methods

Method	Signature(s)	Description
condense	void	Condense image (Re-run-length encode image in memory).
contrast	unsigned int sharpen_	Contrast image (enhance intensity differences in image)
crop	const Geometry &geometry_	Crop image (subregion of original image)
cycleColormap	int amount_	Cycle image colormap
despeckle	void	Despeckle image (reduce speckle noise)
display	void	Display image on screen. Caution: if an image format is is not compatable with the display visual (e.g. JPEG on a colormapped display) then the original image will be altered. Use a copy of the original if this is a problem.
draw	const Drawable &drawable_	Draw shape or text on image.
	const std::list< Drawable> &drawable_	Draw shapes or text on image using a set of Drawable objects contained in an STL list. Use of this method improves drawing performance and allows batching draw objects together in a list for repeated use.
edge	double factor_	Edge image (hilight edges in image)
emboss	void	Emboss image (hilight edges with 3D effect)

Image Manipulation Methods

Method	Signature(s)	Description
enhance	void	Enhance image (minimize noise)
equalize	void	Equalize image (histogram equalization)
flip	void	Flip image (reflect each scanline in the vertical direction)
floodFillColor	int x_, int y_, const Color &fillColor_	Flood-fill color across pixels that match the color of the target pixel and are neighbors of the target pixel. Uses current fuzz setting when determining color match.
	const Geometry &point_, const Color &fillColor_	
	int x_, int y_, const Color &fillColor_, const Color &borderColor_	Flood-fill color across pixels starting at target-pixel and stopping at pixels matching specified border color. Uses current fuzz setting when determining color match.
	const Geometry &point_, const Color &fillColor_, const Color &borderColor_	

Image Manipulation Methods

Method	Signature(s)	Description
floodFill-Texture	int x_, int y_, const Image &texture_	Flood-fill texture across pixels that match the color of the target pixel and are neighbors of the target pixel. Uses current fuzz setting when determining color match.
	const Geometry &point_, const Image &texture_	
	int x_, int y_, const Image &texture_, const Color &borderColor_	Flood-fill texture across pixels starting at target-pixel and stopping at pixels matching specified border color. Uses current fuzz setting when determining color match.
	const Geometry &point_, const Image &texture_, const Color &borderColor_	
flop	void	Flop image (reflect each scanline in the horizontal direction)
frame	const Geometry &geometry_ = "25x25+6+6"	Add decorative frame around image
	unsigned int width_, unsigned int height_, int x_, int y_, int innerBevel_ = 0, int outerBevel_ = 0	

Image Manipulation Methods

Method	Signature(s)	Description
gamma	double gamma_	Gamma correct image (uniform red, green, and blue correction).
	double gammaRed_, double gammaGreen_, double gammaBlue_	Gamma correct red, green, and blue channels of image.
implode	double factor_	Implode image (special effect)
layer	LayerType layer_	Extract layer from image. Use this option to extract a particular layer from the image. *MatteLayer*, for example, is useful for extracting the opacity values from an image.
magnify	void	Magnify image by integral size
map	const Image &mapImage_ , bool dither_ = false	Remap image colors with closest color from reference image. Set *dither_* to true in to apply Floyd/Steinberg error diffusion to the image. By default, color reduction chooses an optimal set of colors that best represent the original image. Alternatively, you can choose a particular set of colors from an image file with this option.
matteFloodfill	const Color &target_, unsigned int matte_, int x_, int y_, PaintMethod method_	Floodfill designated area with a matte value
minify	void	Reduce image by integral size

Image Manipulation Methods

Method	Signature(s)	Description
modulate	double brightness_, double saturation_, double hue_	Modulate percent hue, saturation, and brightness of an image
negate	bool grayscale_ = false	Negate colors in image. Replace every pixel with its complementary color (white becomes black, yellow becomes blue, etc.). Set *grayscale* to only negate grayscale values in image.
normalize	void	Normalize image (increase contrast by normalizing the pixel values to span the full range of color values).
oilPaint	unsigned int radius_ = 3	Oilpaint image (image looks like oil painting)
opaque	const Color &opaqueColor_, const Color &penColor_	Change color of pixels matching *opaqueColor_* to specified *penColor_*.
ping	const std::string &imageSpec_	Ping is similar to read except only enough of the image is read to determine the image columns, rows, and filesize. The columns, rows, and fileSize attributes are valid after invoking ping. The image data is not valid after calling ping.
quantize	bool measureError_ = false	Quantize image (reduce number of colors). Set *measureError_* to true in order to calculate error attributes.

Image Manipulation Methods

Method	Signature(s)	Description
raise	const Geometry &geometry_ = "6x6+0+0", bool raisedFlag_ = false	Raise image (lighten or darken the edges of an image to give a 3-D raised or lowered effect)
read	const std::string &imageSpec_	Read image into current object
	const Geometry &size_, const std::string &imageSpec_	Read image of specified size into current object. This form is useful for images that do not specifiy their size or to specify a size hint for decoding an image. For example, when reading a Photo CD, JBIG, or JPEG image, a size request causes the library to return an image which is the next resolution greater or equal to the specified size. This may result in memory and time savings.
reduceNoise	void	Reduce noise in image using a noise peak elimination filter.
roll	int columns_, int rows_	Roll image (rolls image vertically and horizontally) by specified number of columnms and rows)
rotate	double degrees_, bool crop_ = false, unsigned int sharpen_ = false	Rotate image counter-clockwise by specified number of degrees. Optionally crop image to original size and sharpen image.
sample	const Geometry &geometry_	Resize image by using pixel sampling algorithm

Image Manipulation Methods

Method	Signature(s)	Description
scale	const Geometry &geometry_	Resize image by using simple ratio algorithm
segment	double clusterThreshold_ = 1.0, double smoothingThreshold_ = 1.5	Segment (coalesce similar image components) by analyzing the histograms of the color components and identifying units that are homogeneous with the fuzzy c-means technique. Also uses *quantizeColorSpace* and *verbose* image attributes. Specify *clusterThreshold_*, as the number of pixels each cluster must exceed the the cluster threshold to be considered valid. *SmoothingThreshold_* eliminates noise in the second derivative of the histogram. As the value is increased, you can expect a smoother second derivative. The default is 1.5.
shade	double azimuth_ = 30, double elevation_ = 30, bool colorShading_ = false	Shade image using distant light source. Specify azimuth_ and elevation_ as the position of the light source. By default, the shading results as a grayscale image.. Set*colorShading_* to true to shade the red, green, and blue components of the image.
sharpen	double factor_	Sharpen pixels in image. Specify factor as the percent enhancement (0.0 - 99.9%).

Image Manipulation Methods

Method	Signature(s)	Description
shear	double xShearAngle_, double yShearAngle_, bool crop_ = false	Shear image (create parallelogram by sliding image by X or Y axis). Shearing slides one edge of an image along the X or Y axis, creating a parallelogram. An X direction shear slides an edge along the X axis, while a Y direction shear slides an edge along the Y axis. The amount of the shear is controlled by a shear angle. For X direction shears, x degrees is measured relative to the Y axis, and similarly, for Y direction shears y degrees is measured relative to the X axis. Empty triangles left over from shearing the image are filled with the color defined as borderColor. Specify *crop_* as *true* to crop the sheared image to the original size.
solarize	double factor_ = 50.0	Solarize image (similar to effect seen when exposing a photographic film to light during the development process)
spread	unsigned int amount_ = 3	Spread pixels randomly within image by specified amount
stegano	const Image &watermark_	Add a digital watermark to the image (based on second image)
stereo	const Image &rightImage_	Create an image which appears in stereo when viewed with red-blue glasses (Red image on left, blue on right)
swirl	double degrees_	Swirl image (image pixels are rotated by degrees)
texture	const Image &texture_	Layer a texture on image background
threshold	double threshold_	Threshold image

Image Manipulation Methods

Method	Signature(s)	Description
transform	const Geometry &imageGeometry_	Transform image based on image and crop geometries. Crop geometry is optional.
	const Geometry &imageGeometry_, const Geometry &cropGeometry_	
transform-ColorSpace	ColorspaceType colorSpace_	Transform the image representation to a different colorspace.
transparent	const Color &color_	Add matte image to image, setting pixels matching color to transparent.
trim	void	Trim edges that are the background color from the image.
wave	double amplitude_ = 25.0, double wavelength_ = 150.0	Alter an image along a sine wave.
write	const std::string &imageSpec_	Write image to a file using filename imageSpec_. Caution: if an image format is selected which is capable of supporting fewer colors than the original image or quantization has been requested, the original image will be quantized to fewer colors. Use a copy of the original if this is a problem.
zoom	const Geometry &geometry_	Zoom image to specified size.

Image Attributes

Image attributes are set and obtained via methods in Image. Except for methods which accept pointer arguments (e.g. chromaBluePrimary) all methods return attributes by value. Within the image object, attributes may be properties of the image, the user-options, or both. In the case where the attribute is a property of both the image and the user-options, the attribute associated with the image is returned if operations on the image can usefully update it, or the user-options if not. In all cases, the value set is equivalent to the next returned value. It is an error (an exception will be thrown) to attempt to set an attribute which is only a property of the image if no image is contained within the object. In the case of setting an attribute which is both a property of the image and the user-options and no image is present, the user-options are set and no error is reported.

The supported image attributes and the method arguments required to obtain them are shown in the following table:

Image Attributes

Method	Return Type	Signature(s)	Description
adjoin	bool	void	Join images into a single multi-image file.
	void	bool flag_	
antiAlias	bool	void	Control antialiasing of rendered Postscript and Postscript or TrueType fonts. Enabled by default.
	void	bool flag_	

Image Attributes

Method	Return Type	Signature(s)	Description
animationDelay	unsigned int	void	Time in 1/100ths of a second (0 to 65535) which must expire before displaying the next image in an animated sequence. This option is useful for regulating the animation of a sequence of GIF images within Netscape.
	void	unsigned int delay_	
animation-Iterations	unsigned int	void	Number of iterations to loop an animation (e.g. Netscape loop extension) for.
	void	unsigned int iterations_	
background-Color	Color	void	Image background color
	void	const Color &color_	
background-Texture	std::string	void	Image to use as background texture.
	void	const string &texture_	
baseColumns	unsigned int	void	Base image width (before transformations)
baseFilename	std::string	void	Base image filename (before transformations)
baseRows	unsigned int	void	Base image height (before transformations)
borderColor	Color	void	Image border color
	void	const Color &color_	

Image Attributes

Method	Return Type	Signature(s)	Description
boxColor	Color	void	Base color that annotation text is rendered on.
	void	const Color &boxColor_	
chroma-BluePrimary	void	float *x_, float *y_	Get chromaticity blue primary point
	void	float x_, float y_	Set chromaticity blue primary point (e.g. x=0.15, y=0.06)
chroma-GreenPrimary	void	float *x_, float *y_	Get chromaticity green primary point
	void	float x_, float y_	Set chromaticity green primary point (e.g. x=0.3, y=0.6)
chroma-RedPrimary	void	float *x_, float *y_	Get chromaticity red primary point
	void	float x_, float y_	Set chromaticity red primary point (e.g. x=0.64, y=0.33)
chroma-WhitePoint	void	float *x_, float *y_	Get chromaticity white point
	void	float x_, float y_	Set chromaticity white point (e.g. x=0.3127, y=0.329)
classType	ClassType	void	Image class

Image Attributes

Method	Return Type	Signature(s)	Description
colorFuzz	unsigned int	void	Colors within this distance are considered equal. A number of algorithms search for a target color. By default the color must be exact. Use this option to match colors that are close to the target color in RGB space.
	void	unsigned int fuzz_	
colorMap	Color	unsigned int index_	Color at color-palette index.
	void	unsigned int index_, const Color &color_	
columns	unsigned int	void	Image width
comment	std::string	void	Comment image (add comment string to image). By default, each image is commented with its file name. Use this method to assign a specific comment to the image. Optionally you can include the image filename, type, width, height, or other image attributes by embedding special format characters.
	void	const std::string &comment_	
compressType	CompressionType	void	Image compresion type. The default is the compression type of the specified image file.
	void	CompressionType compressType_	

Image Attributes

Method	Return Type	Signature(s)	Description
density	Geometry	void	Vertical and horizontal resolution in pixels of the image (default 72x72). This option specifies an image density when decoding a Postscript or Portable Document page. Often used with *psPageSize*.
	void	const Geometry &density_	
depth	unsigned int	void	Image depth (8 or 16). Used to specify the bit depth when reading or writing raw images. Defaults to the quantum depth that ImageMagick is built with.
	void	unsigned int depth_	
directory	std::string	void	Tile names from within an image montage
fileName	std::string	void	Image file name.
	void	const string &fileName_	
fileSize	unsigned int	void	Number of bytes of the image on disk
filterType	FilterType	void	Filter to use when resizing image. The reduction filter employed has a sigificant effect on the time required to resize an image and the resulting quality. The default filter is *Lanczos* which has been shown to produce high quality results when reducing most images.
	void	FilterType filterType_	

Image Attributes

Method	Return Type	Signature(s)	Description
font	std::string	void	Text rendering font. If the font is a fully qualified X server font name, the font is obtained from an X server. To use a TrueType font, precede the TrueType filename with an @. Otherwise, specify a Postscript font name (e.g. "helvetica").
	void	const string &font_	
fontPointsize	unsigned int	void	Text rendering font point size
	void	unsigned int pointSize_	
format	std::string	void	Long form image format description.
gamma	double	void	Gamma level of the image. The same color image displayed on two different workstations may look different due to differences in the display monitor. Use gamma correction to adjust for this color difference.
geometry	Geometry	void	Preferred size of the image when encoding.
	void	Geometry	

Image Attributes

Method	Return Type	Signature(s)	Description
gifDispose-Method	unsigned int	void	GIF disposal method. This option is used to control how successive frames are rendered (how the preceding frame is disposed of) when creating a GIF animation. { 0 = Disposal not specified, 1 = Do not dispose of graphic, 3 = Overwrite graphic with background color, 4 = Overwrite graphic with previous graphic. }
	void	unsigned int *disposeMethod_*	
iccColorProfile	Blob	void	ICC color profile. Supplied via a Blob since Magick++/ and ImageMagick do not currently support formating this data structure directly. Specifications are available from the International Color Consortium for the format of ICC color profiles.
	void	const Blob &colorProfile_	

Image Attributes

Method	Return Type	Signature(s)	Description
interlaceType	InterlaceType	void	The type of interlacing scheme (default *NoInterlace*). This option is used to specify the type of interlacing scheme for raw image formats such as RGB or YUV. *NoInterlace* means do not interlace, *LineInterlace* uses scanline interlacing, and *PlaneInterlace* uses plane interlacing. *PartitionInterlace* is like *PlaneInterlace* except the different planes are saved to individual files (e.g. image.R, image.G, and image.B). Use *LineInterlace* or *PlaneInterlace* to create an interlaced GIF or progressive JPEG image.
	void	InterlaceType interlace_	
iptcProfile	Blob	void	IPTC profile. Supplied via a Blob since Magick++ and ImageMagick do not currently support formating this data structure directly. Specifications are available from the International Press Telecommunications Council for IPTC profiles.
	void	const Blob& iptcProfile_	

Image Attributes

Method	Return Type	Signature(s)	Description
label	std::string	void	Assign a label to an image. Use this option to assign a specific label to the image. Optionally you can include the image filename, type, width, height, or scene number in the label by embedding special format characters. If the first character of string is @, the image label is read from a file titled by the remaining characters in the string. When converting to Postscript, use this option to specify a header string to print above the image.
	void	const std::string &label_	
lineWidth	unsigned int	void	Line width for drawing lines, circles, ellipses, etc. See Drawable.
	void	unsigned int lineWidth_	
magick	std::string	void	Get image format (e.g. "GIF")
	void	const std::string &magick_	
matte	bool	void	True if the image has transparency. If set True, store matte channel if the image has one otherwise create an opaque one.
	void	bool matteFlag_	

Image Attributes

Method	Return Type	Signature(s)	Description
matteColor	Color	void	Image matte (transparent) color
	void	const Color &matteColor_	
meanError-PerPixel	double	void	The mean error per pixel computed when an image is color reduced. This parameter is only valid if verbose is set to true and the image has just been quantized.
monochrome	bool	void	Transform the image to black and white
	void	bool flag_	
montage-Geometry	Geometry	void	Tile size and offset within an image montage. Only valid for montage images.
normalized-MaxError	double	void	The normalized max error per pixel computed when an image is color reduced. This parameter is only valid if *verbose* is set to true and the image has just been quantized.
normalized-MeanError	double	void	The normalized mean error per pixel computed when an image is color reduced. This parameter is only valid if *verbose* is set to true and the image has just been quantized.

Image Attributes

Method	Return Type	Signature(s)	Description
packets	unsigned int	void	The number of runlength-encoded packets in the image
packetSize	unsigned int	void	The number of bytes in each pixel packet
penColor	Color	void	Pen color to use when annotating on or drawing on image.
	void	const Color &penColor_	
penTexture	Image	void	Texture image to paint with (similar to penColor).
	void	const Image & penTexture_	
pixelColor	Color	unsigned int x_, unsigned int y_	Get/set pixel color at location x & y.
	void	unsigned int x_, unsigned int y_, const Color &color_	
psPageSize	Geometry	void	Postscript page size. Use this option to specify the dimensions of the Postscript page in dots per inch or a TEXT page in pixels. This option is typically used in concert with density.
	void	const Geometry &pageSize_	

Image Attributes

Method	Return Type	Signature(s)	Description
quality	unsigned int	void	JPEG/MIFF/PNG compression level (Range 0 to 100 with default of 75).
	void	unsigned int quality_	
quantizeColors	unsigned int	void	Preferred number of colors in the image. The actual number of colors in the image may be less than your request, but never more. Images with less unique colors than specified with this option will have any duplicate or unused colors removed.
	void	unsigned int colors_	
quantize-ColorSpace	ColorspaceType	void	Colorspace to quantize colors in (default RGB). Empirical evidence suggests that distances in color spaces such as YUV or YIQ correspond to perceptual color differences more closely than do distances in RGB space. These color spaces may give better results when color reducing an image.
	void	ColorspaceType colorSpace_	
quantizeDither	bool	void	Apply Floyd/Steinberg error diffusion to the image. The basic strategy of dithering is to trade intensity resolution for spatial resolution by averaging the intensities of several neighboring pixels. Images which suffer from severe contouring when reducing colors can be improved with this option. The *quantizeColors* or *monochrome* option must be set for this option to take effect.
	void	bool flag_	

Image Attributes

Method	Return Type	Signature(s)	Description
quantizeError	unsigned int	void	Quantization error. Only valid if *verbose* is set to true prior to executing quantize and the value is read back immediately.
quantize-TreeDepth	unsigned int	void	Depth of the quantization color classification tree. Values of 0 or 1 allow selection of the optimal tree depth for the color reduction algorithm. Values between 2 and 8 may be used to manually adjust the tree depth.
	void	unsigned int treeDepth_	
renderingIntent	RenderingIntent	void	The type of rendering intent
	void	RenderingIntent render_	
resolutionUnits	ResolutionType	void	Units of image resolution
	void	ResolutionType units_	
rows	unsigned int	void	The number of pixel rows in the image
scene	unsigned int	void	Image scene number
	void	unsigned int scene_	
signature	std::string	bool force_ = false	Image MD5 signature. Set *force_* to *true* to force re-computation of signature.

Image Attributes

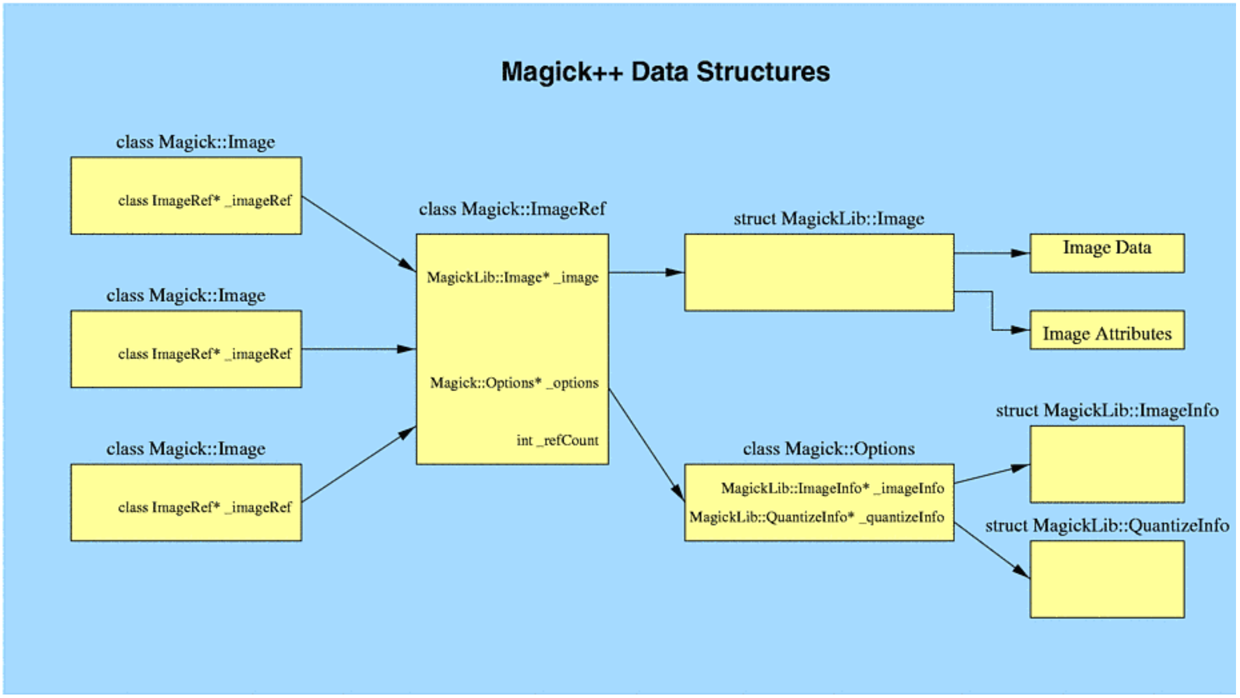
Method	Return Type	Signature(s)	Description
size	Geometry	void	Width and height of a raw image (an image which does not support width and height information). Size may also be used to affect the image size read from a multi-resolution format (e.g. Photo CD, JBIG, or JPEG.
	void	const Geometry &geometry_	
subImage	unsigned int	void	Subimage of an image sequence
	void	unsigned int subImage_	
subRange	unsigned int	void	Number of images relative to the base image
	void	unsigned int subRange_	
text	std::string	void	Any text associated with the image
tileName	std::string	void	Tile name
	void	const std::string &tileName_	
totalColors	unsigned long	void	Number of colors in the image
type	ImageType	void	Image type

Image Attributes

Method	Return Type	Signature(s)	Description
verbose	bool	void	Print detailed information about the image
	void	bool verboseFlag_	
view	std::string	void	FlashPix viewing parameters.
	void	const string &view_	
x11Display	std::string	void	X11 display to display to, obtain fonts from, or to capture image from (e.g. "hostname:0.0")
	void	const string &display_	
xResolution	double	void	x resolution of the image
yResolution	double	void	y resolution of the image

Image Data Structures

The class Magick::Image is a simple handle which points to a reference-counted image representation. This allows multiple Magick::Image instances to share the same image and attributes. At the point in time that the image data, or image attributes are modified and the current reference count is greater than one, the image data and attributes are copied to create a new image with a reference count of one and the reference count on the old image is decremented. If the reference count on the old image becomes zero, then the associated reference and data are deleted. This strategy represents a simple (but effective) form of garbage collection.


Magick++ Data Structures
class Magick::Image
class ImageRef* _imageRef
class Magick::Image
class ImageRef* _imageRef
class Magick::Image
class ImageRef* _imageRef
class Magick::ImageRef
MagickLib::Image* _image
Magick::Options* _options
int _refCount
struct MagickLib::Image
Image Data
Image Attributes
struct MagickLib::ImageInfo
class Magick::Options
MagickLib::ImageInfo* _imageInfo
MagickLib::QuantizeInfo* _quantizeInfo
struct MagickLib::QuantizeInfo

STL Support

Magick++ provides a set of STL algorithms for operating across ranges of image frames in a container. It also provides a set of STL unary function objects to apply an operation on image frames in a container via an algorithm which uses unary function objects. A good example of a standard algorithm which is useful for processing containers of image frames is the STL for_each algorithm which invokes a unary function object on a range of container elements.

Magick++ uses a limited set of template argument types. The current template argument types are:

Container

A container having the properties of a Back Insertion Sequence. Sequences support forward iterators and Back Insertion Sequences support the additional abilty to append an element via *push_back()*. Common compatable container types are the STL *<vector>* and *<list>* template containers. This template argument is usually used to represent an output container in which one or more image frames may be appended. Containers like STL *<vector>* which have a given default capacity may need to have their capacity adjusted via *reserve()* to a larger capacity in order to support the expected final size . Since Magick++ images are very small, it is likely that the default capacity of STL *<vector>* is sufficient for most situations.

InputIterator

An input iterator used to express a position in a container. These template arguments are typically used to represent a range of elements with *first_* representing the first element to be processed and *last_* representing the element to stop at. When processing the entire contents of a container, it is handy to know that STL containers usually provide the *begin()* and *end()* methods to return input interators which correspond with the first and last elements, respectively.

The following is an example of how frames from a GIF animation "test_image_anim.gif" may be appended horizontally with the resulting image written to the file "appended_image.miff":

```
#include <list>
#include <Magick++.h>
using namespace std;
using namespace Magick;

int main(int /*argc*/,char **/*argv*/)
{
   list<Image> imageList;
   readImages( &imageList, "test_image_anim.gif" );

   Image appended;
   appendImages( &appended, imageList.begin(), imageList.end() );
   appended.write( "appended_image.miff" );
   return 0;
}
```

The available Magick++ specific STL algorithms for operating on sequences of image frames are shown in the following table

STL Algorithms

Algorithm	Signature	Description
animateImages	inputIterator first_, InputIterator last_	Animate a sequence of image frames. Image frames are displayed in succession, creating an animated effect. The animation options are taken from the first image frame. This feature is only supported under X11 at the moment.
appendImages	Image *appendedImage_, InputIterator first_, InputIterator last_, bool stack_= false	Append a sequence of image frames, writing the result to *appendedImage_*. All the input image frames must have the same width or height. Image frames of the same width are stacked top-to-bottom. Image frames of the same height are stacked left-to-right. If the *stack_* parameter is false, rectangular image frames are stacked left-to-right otherwise top-to-bottom.
averageImages	Image *averagedImage_, InputIterator first_, InputIterator last_	Average a sequence of image frames, writing the result to *averagedImage_*. All the input image frames must be the same size in pixels.
coalesceImages	InputIterator first_, InputIterator last_	Merge a sequence of images. This is useful for GIF animation sequences that have page offsets and disposal methods. The input images are modified in-place.

Algorithm	Signature	Description
displayImages	inputIterator first_, InputIterator last_	Display a sequence of image frames. Through use of a pop-up menu, image frames may be selected in succession. This feature is fully supported under X11 but may have only limited support in other environments. Caution: if an image format is is not compatable with the display visual (e.g. JPEG on a colormapped display) then the original image will be altered. Use a copy of the original if this is a problem.
mapImages	InputIterator first_, InputIterator last_, const Image& mapImage_, bool dither_, bool measureError_ = false	Replace the colors of a sequence of images with the closest color from a reference image. Set *dither_* to *true* to enable dithering. Set *measureError_* to *true* in order to evaluate quantization error.
montageImages	Container *montageImages_, InputIterator first_, InputIterator last_, const Montage &montageOpts_	Create a composite image by combining several separate image frames. Multiple frames may be generated in the output container *montageImages_* depending on the tile setting and the number of image frames montaged. Montage options are provided via the parameter *montageOpts_*. Options set in the first image frame (backgroundColor,borderColor, matteColor, penColor,font, and fontPointsize) are also used as options by *montageImages()*.

STL Algorithms

Algorithm	Signature	Description
morphImages	Container *morphedImages_, InputIterator first_, InputIterator last_, unsigned int frames_	Morph a seqence of image frames. This algorithm expands the number of image frames (output to the container *morphedImages_*) by adding the number of intervening frames specified by *frames_* such that the original frames morph (blend) into each other when played as an animation.
readImages	Container *sequence_, const std::string &imageSpec_	Read a sequence of image frames into existing container (appending to container *sequence_*) with image names specified in the string *imageSpec_*.
writeImages	InputIterator first_, InputIterator last_, const std::string &imageSpec_, bool adjoin_ = true	Write images in container to file specified by string *imageSpec_*. Set *adjoin_* to false to write a set of image frames via a wildcard *imageSpec_* (e.g. image%02d.miff). Caution: if an image format is selected which is capable of supporting fewer colors than the original image or quantization has been requested, the original image will be quantized to fewer colors. Use a copy of the original if this is a problem.
quantizeImages	InputIterator first_, InputIterator last_, bool measureError_ = false	Quantize colors in images using current quantization settings. Set *measureError_* to true in order to measure quantization error.

Magick++ Unary Function Objects

Magick++ unary function objects inherit from the STL unary_function template class . The STL unary_function template class is of the form

```
unary_function<Arg, Result>
```

and expects that derived classes implement a method of the form:

```
Result operator()( Arg argument_ );
```

which is invoked by algorithms using the function object. In the case of unary function objects defined by Magick++, the invoked function looks like:

```
void operator()( Image &image_);
```

with a typical implementation looking similar to:

```
void operator()( Image &image_ )
      {
        image_.contrast( _sharpen );
      }
```

where *contrast* is an Image method and *_sharpen* is an argument stored within the function object by its contructor. Since constructors may be polymorphic, a given function object may have several constructors and selects the appropriate Image method based on the arguments supplied.

In essence, unary function objects (as provided by Magick++) simply provide the means to construct an object which caches arguments for later use by an algorithm designed for use with unary function objects. There is a unary function object corresponding each algorithm provided by the Image class and there is a contructor available compatable with each synonymous method in the Image class. The class name is the same as the Image class method name with the string “Image” appended. For example, *read* becomes *readImage*.

Function objects are available to set attributes on image frames which are equivalent to methods in the Image object. These function objects allow setting an option across a range of image frames using for_each.

The following code is an example of how the color 'red' may be set to transparent in a GIF animation:

```
list<image> images;
readImages( &images, "animation.gif" );
for_each ( images.begin(), images.end(), transparentImage( "red" )  );
writeImages( images.begin(), images.end(), "animation.gif" );
```

Installing Magick++

General

In order to compile Magic++ you must have access to a standard C++ implementation and have *ImageMagick* installed (*ftp://ftp.wizards.dupont.com/pub/ImageMagick/*). Magick++ is co-packaged as a subdirectory of ImageMagick as of ImageMagick version 4.2.2 and later. The author uses the *egcs 1.1.2 version of GNU C++* which is available under UNIX and under the *Cygwin UNIX-emulation environment* for Windows. Standards compliant commercial C++ compilers should also work fine. Most modern C++ compilers for PCs should also work (project files are provided for Microsoft Visual C++ 6.0).

The compiler must support the following recent C++ standard features:

- bool type
- string class (<string>)
- exceptions (<exception>)
- namespaces
- C++ versions of standard C headers (e.g. <cstring>)
- Standard Template Library (STL) (e.g. <list>, <vector>)

I have personally verified that Magick++ compiles and runs using the following compiler/platform combinations:

Tested Configurations

Operating System	Architecture	Compiler
Solaris 2.6	SPARC	egcs 1.1.1
Solaris 2.6	SPARC	egcs 1.1.2
FreeBSD 2.2.7	Intel Pentium II	egcs 1.1.2
Windows NT 4.0 SP3	Intel Pentium II	Visual C++ Standard Edition

Please let me know if you have successfully built and executed Magick++ using a different configuration so that I can add to the table of verified configurations.

UNIX

To install the package under Unix, installation should be similar to

```
./configure [--prefix=/prefix]
make
make install
```

The configure script uses the compiler/linker flags it obtains from the installed 'Magick-config' script when performing the build. The library is currently named similar to 'libMagick++.a' and is installed under prefix/lib while the headers are installed under prefix/include.

To influence the options the configure script chooses, you may specify environment variables when running the script. For example, the command

```
CXX=CC CXXFLAGS=-O2 LIBS=-lposix./configure
```

specifies additional options to the configure script. The following table shows the available options:

Configuration Environment Variables

Environment Variable	Description
CXX	Name of C++ compiler (e.g. 'CC -Xa') to use compiler 'CC -Xa'
CXXFLAGS	Compiler flags (e.g. '-g -O2') to compile with
CPPFLAGS	Include paths (-I/somedir) to look for header files
LDFLAGS	Library paths (-L/somedir) to look for libraries. Systems that support the notion of a library run-path may additionally require -R/somedir or '-rpath /somedir' in order to find shared libraries at run time.
LIBS	Extra libraries (-lsomelib) required to link

Windows ‘9X and Windows NT

Visual C++

To build using Visual C++, extract the contents of Magick++-version.zip (preserving sub-directories) in the ImageMagick distribution directory. This will create the directory Magick++-version containing the sub-directories 'demo', 'doc', 'lib', and 'tests'. Open the workspace file Magick++.dsw and build the project Magick++ in order to build the library. The library is output to the same directory as the ImageMagick libraries.

The available projects are:

Visual C++ Projects

Project	Description
Magick++	the Magick++ library
attributes	test setting image attributes
manipulate	test manipulating images
button	program to create a simple rectangular button with an annotation
flip	program to invert and morph images in an existing GIF animation
demo	program to demonstrate the image manipulation primitives
shapes	program to demonstrate use of the drawing primitives

Test and demonstration programs are built in the directory which contains their sources. The Magick++ library is placed in the ImageMagick/lib directory alongside the ImageMagick library.

Cygwin & EGCS

It is possible to build both ImageMagick and Magick++ under the Cygwin Unix-emulation environment for Windows NT. Obtain and install Cygwin from http://sourceware.cygnus.com/cygwin/ and update to the latest EGCS compiler from http://www.xraylith.wisc.edu/~khan/software/gnu-win32/egcs.html. X11R6.4 libraries are available from http://dao.gsfc.nasa.gov/software/grads/win32/X11R6.4/. To build using Cygwin and EGCS, follow the instructions for building under Unix. ImageMagick and Magick++ do not yet include support for building Windows DLLs under Cygwin so do not enable dynamic libraries when building ImageMagick.

Appendix A

Overview

ImageMagick™ supports over fifty image formats. Some of the image formats require additional programs or libraries. See the ImageMagick ReadMe file for information about where to find the related materials.

Image Formats

Format	Description	Notes
AVS	AVS X image file	
BMP	Microsoft Windows bitmap image file	
BMP24	Microsoft Windows 24-bit bitmap image file	
CGM	Computer graphics metafile	requires ralcgm; read only
CMYK	raw cyan, magenta, yellow, and black bytes	user -size command line option to specify width and height

Image Formats

Format (Cont.)	Description	Notes
DCM	Digital Imaging and Communications in Medicine image format	read only
DCX	ZSoft IBM PC multipage Paintbrush file	
DIB	Microsoft Windows bitmap image file	
EPDF	Encapsulated Portable Document Format file	
EPS	Adobe Encapsulated PostScript file	requires Ghostscript
EPS2	Adobe Level II Encapsulated PostScript file	requires Ghostscript
EPSF	Adobe Encapsulated PostScript Interchange format	requires Ghostscript
EPSI	Adobe Encapsulated PostScript Interchange format	requires Ghostscript
FAX	Group 3	
FIG	TransFig image format	requires TransFig
FITS	Flexible Image Transport System	

Image Formats

Format (Cont.)	Description	Notes
FPX	FlashPix format	use -DHasFPX to compile; requires FlashPIX SDK
GIF	CompuServer graphics interchange format	8-bit color
GIF87	CompuServer graphics interchagne format	8-bit color (version 87a)
GRADATION	gradual passing from one shade to another	specify the desired shading as the filename (e.g., gradation: red-blue)
GRANITE	granite texture	
GRAY	raw gray bytes	use -size command line option to specify width and height
HDF	Hierarchical Data Format	use -DHasHDF to compile
HISTOGRAM	histogram of an image	

Image Formats

Format (Cont.)	Description	Notes
HTML	Hypertext Markup Language with a client-side image map	requires HTML2PS to read this format
JBIG	Joint Bi-level Image Experts Group file interchange format	use -DHasJBIG to compile
JPEG	Joint Photographic Experts Group JFIF format	use -DHasJPEG to compile
ICO	Microsoft icon	read only
LABEL	text image format	specify label text as the filename (e.g., label:This is a label)
MAP	colormap intensities and indices	
MIFF	Magick Image File Format	
MNG	Multiple Image Network Graphics	
MPEG	Motion Picture Experts Group file interchange format	use -DHasMPEG to compile
MTV	MTV Raytracing image format	
NETSCAPE	Netscape 216 color cube	

Image Formats

Format (Cont.)	Description	Notes
NULL	null image	useful for creating blank tiles with montage
PBM	portable bitmap format (black and white)	
PCD	Photo CD	maximum resolution written is 512 x 768 pixels
PCDS	Photo CD	decode with the sRGB color tables
PCL	Page Control Language	write only
PCX	ZSoft IBM PC Paintbrush file	
PDF	Portable Document Format	requires Ghostscript
PGM	portable graymap format (grayscale)	
PICT	Apple Macintosh QuickDraw/PICT file	
PIX	Alias/Wavefront RLE image format	read only

Image Formats

Format (Cont.)	Description	Notes
PLASMA	plasma fractal image	specify the base color as the filename (e.g., plasma:blue-yellow); use fractal to initialize randome value (e.g., plasma:fractal)
PNG	Portable Network Graphics	
PNM	portable anymap	use +compress to produce ASCII renditions
PPM	portable pixmap format (color)	
PWP	Seattle Film Works	read only
P7	Xv's visual schnauzer format	
PS	Adobe PostScript file	requires Ghostscript
PS2	Adobe Level II PostScript file	requires Ghostscript
PSD	Adobe Photoshop bitmap file	
RAD	Radiance image file	

Image Formats

Format (Cont.)	Description	Notes
RGB	raw red, green, and blue bytes	use -size command line option to specify width and height
RGBA	raw red, green, blue, and matte bytes	use -size command line option to specify width and height
RLA	Alias/Wavefront image file	read only
RLE	Utah run length encoded image file	read only
SCAN	Import image from a scanner device	requires SANE; specify device name and path as the filename (e.g., scan:mustek:/dev/scanner)
SFW	Seattle Film Works	read only
SGI	Irix RGB image file	
SHTML	Hypertext Markup Language with a client-side image map	write only

Image Formats

Format (Cont.)	Description	Notes
SUN	SUN rasterfile	
TEXT	raw text file	read only
TGA	Truevision Targa image file	
TIFF	Tagged Image File Format	use -DHasTIFF to compile
TIFF24	24-bit Tagged Image File Format	use -DHasTIFF to compile
TILE	tile image with a texture	read only
TIM	PSX TIM file	read only
TTF	TrueType font file	read only
UIL	X-Motif UIL table	
UYVY	16-bit/pixel interleaved YUV	use -size command line option to specify width and height
VICAR		read only
VID	Visual Image Directory	

Image Formats

Format (Cont.)	Description	Notes
VIFF	Khoros Visualization Image File Format	
WIN	select image from or display image to your computer screen	
WMF	Windows Meta Format	read only
X	select image from or display image to your X server screen	
XC	constant image of X server color	use -size command line option to specify width and height
XBM	X Windows system bitmap (black and white only)	
XPM	X Windows system pixmap file (color)	
XWD	X Windows system window dump file (color)	
YUV	CCIR 601 4:1:1 file	use -size command option to specify width and height

On some platforms, ImageMagick processes the following extensions automatically:

- .gz for Zip compression
- .Z for Unix compression
- .bz2 for block compression
- .pgp for PGP encryption

For example, a PNM image called image.pnm.gz is decompressed and read with the gzip program automatically.

Appendix B

X Resources

Overview

Several of the ImageMagick features use X resources.

These resources are identified in the table in alphabetical order.

X Resources

X Resource	Function
background (class **Background**) *Used by animate, display, montage*	Specifies the preferred color to use for the Image window background. The default is **#ccc**.
borderColor (class **BorderColor**) *Used by animate, display, montage*	Specifies the preferred color to use for the Image window border. The default is **#ccc**.
borderWidth (class **BorderWidth**) *Used by animate, display, montage*	Specifies the width in pixels of the Image window border. The default is **2**.
browseCommand (class **browseCommand** *Used by display*	Specifies the name of the preferred browser when displaying ImageMagick documentation. The default is **netscape %s**.

X Resources

X Resource (Cont.)	Function
confirmExit (class **ConfirmExit**) *Used by display*	Prompts the user to confirm exiting the program when exiting ImageMagick. Set this resource to `False` to exit without a confirmation.
displayGamma (class **DisplayGamma**) *Used by display*	Specifies the gamma of your X server. You can apply separate gamma values to the red, green, and blue channels of an image with a gamma value list delineated with slashes—1.7/2.3/1.2.
displayWarnings (class **DisplayWarnings**) *Used by display*	Displays a warning message when appropriate. Set this resource to `False` to ignore warning messages.
editorCommand (class **editorCommand**) *Used by display*	Specifies the name of the preferred editor when editing image comments. The default is **xterm -title "Edit Image Comment" -e vi %s**.
font (class **Font** or **FontList**) *Used by animate, display, montage*	Specifies the name of the preferred font to use in normal formatted text. The default is **14 point Helvetica**.

X Resources

X Resource (Cont.)	Function
font[1–9] (class **Font[1–9]**) *Used by display*	Specifies the name of the preferred font to use when annotating an image window with text. The default fonts are **fixed**, **variable**, **5x8**, **6x10**, **7x13bold**, **8x13bold**, **9x15bold**, **10x20**, and **12x24**. See Image Annotation for details.
foreground (class **Foreground**) *Used by animate, display, montage*	Specifies the preferred color to use for text within the Image window. The default is **black**.
gammaCorrect (class **gammaCorrect**) *Used by display*	This resource, if true, will lighten or darken an image of known gamma to match the gamma of the display. See the resource *displayGamma*. The default is **True**.
geometry (class **geometry**) *Used by animate, display*	Specifies the preferred size and position of the image window. It is not necessarily obeyed by all window managers.
iconGeometry (class **IconGeometry**) *Used by animate, display, montage*	Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

X Resources

X Resource (Cont.)	Function
iconic (class **Iconic**) *Used by animate, display, montage*	Specifies you would prefer an application's windows not be visible initially, as if the windows had been immediately iconified by you. Window managers may choose not to honor the application's request.
magnify (class **Magnify**) *Used by display*	Specifies an integral factor by which an image should be enlarged. The default is **3**.
matteColor (class **MatteColor**) *Used by animate, display, montage*	The color of windows. It's used for the backgrounds of windows, menus, and notices. A 3D effect is achieved by using highlight and shadow colors derived from this color. The default is **#ddd**.
name (class **Name**) *Used by animate, display, montage*	The name under which resources for the application should be found. This resource is useful in shell aliases to distinguish between invocations of an application without resorting to creating links to alter the executable file name. The default is the application name.

X Resources

X Resource (Cont.)	Function
pen[1–9] (class **Pen[1–9]**) *Used by display*	Specifies the color of the preferred font to use when annotating an image window with text. The default colors are **black**, **blue**, **green**, **cyan**, **gray**, **red**, **magenta**, **yellow**, and **white**. See Image Annotation for details.
printCommand (class **PrintCommand**) *Used by display*	This command is executed when ever Print is issued. See Buttons. In general, it's the command to print PostScript to your printer. The default value is **lpr -r %s.**
sharedMemory (class **SharedMemory**) *Used by animate, display, montage*	Whether animate should attempt to use shared memory for pixmaps. ImageMagick must be compiled with shared memory support, and the display must support the MIT-SHM extension. Otherwise, this resource is ignored. The default is **True**.
textfont (class **textFont**) *Used by animate, display, montage*	The name of the preferred font to use in fixed (typewriter style) formatted text. The default is **14 point Courier**.

X Resources

X Resource (Cont.)	Function
title (class **Title**) *Used by animate, display, montage*	The title to use for the Image window. This information is sometimes used by a window manager to provide some sort of header to identify the window. The default is the image file name.
undoCache (class **UndoCache**) *Used by display*	Specifies, in megabytes (Mb), the amount of memory in the undo edit cache. Each time you modify the image, it's saved in the undo edit cache as long as memory is available. You can subsequently undo one or more of these transformations. The default is **16Mb.**

X Resources

X Resource (Cont.)	Function
usePixmap (class **UsePixmap**) *Used by display*	Images are maintained as an ximage by default. Set this resource to `True` to use a server pixmap instead. This is useful if your image exceeds the dimensions of your server screen and you intend to pan the image. Panning is much faster with pixmaps than with ximages. Pixmaps are considered a precious resource; use them with discretion. To set the geometry of the *Magnify* or *Pan* window, use the geometry resource. For example, to set the pan window geometry to 256x256, use `display.pan.geometry: 256x256`.

Appendix C

MIFF

Overview

Magick Image File Format (MIFF) is a platform-independent format for storing bitmap images. MIFF is a part of the ImageMagick toolkit of image manipulation utilities for the X Window System. ImageMagick is capable of converting many different image file formats to and from MIFF (e.g., JPEG, XPM, TIFF, etc.).

A MIFF image file consist of two sections.

- a header composed of keywords describing the image in text form
- the binary image data

The header is separated from the image data by a colon (:) character immediately followed by a ctrl-Z (^Z).

The MIFF header is composed entirely of LATIN-1 characters. The fields in the header are a keyword and value combination in the *keyword=value* format. Each keyword and value is separated by an equal sign (=). Each keyword=value combination is delimited by at least one control or whitespace character.

Comments may appear in the header section and are always delimited by braces. The MIFF header always ends with a colon (:) character, followed by a ctrl-Z character (^Z). It's also common for a formfeed and a newline character to appear before the colon. You can then list the image keywords with the Unix *more program*, *wi*thout printing the binary image that follows the colon separator. The ctrl-Z character has the same effect with *type* from the Win32 command line.

The following is a list of keyword=value combinations that may be found in a MIFF file:

Keyword/Value Combinations

Keyword=value	**Definition**
background-color=x,y border-color=x,y matte-color=x,y	These optional keywords reflect the image background, border, and matte colors, respectively.
class=DirectClass , class=PseudoClass	The type of binary image data stored in the MIFF file. If this keyword is not present, DirectClass image data is assumed.
colors=value	The number of colors in a DirectClass image. For a PseudoClass image, this keyword specifies the size of the colormap. If this keyword is not specified in the header, and the image is PseudoClass, a linear 256 color grayscale colormap is used with the image data.
colorspace=RGB, colorspace=CMYK	The colorspace of the pixel data. The default is RGB.
columns=value	The width of the image in pixels. This is a required keyword and has no default.
color-profile=value	The number of bytes in the International Color Consortium color profile. The profile is defined by the ICC profile specification.
compression=RunlengthEncoded, compression=Zip, compression=BZip	The type of algorithm used to compress the image data. If this keyword is not present, the image data is assumed to be uncompressed.

Keyword/Value Combinations

Keyword=value (Cont.)	Definition
delay <1/100ths of a second>	The interframe delay in an image sequence. The maximum delay is 65535.
depth=8, depth=16	The depth of a single color value representing values from 0 to 255 (depth 8) or 65535 (depth 16). If this keyword is absent, a depth of 8 is assumed.
dispose=value	GIF disposal method. The valid methods are: 0, No disposal specified; 1, Do not dispose; 2, Restore to background color; 3, Restore to previous.
gamma=value	Gamma of the image. If it is not specified, a gamma of 1.0 (linear brightness response) is assumed,
id=ImageMagick	Identifies the file as a MIFF-format image file. This keyword is required and has no default. Although this keyword can appear anywhere in the header, it should start as the first keyword of the header in column 1. This will allow programs like file(1) to easily identify the file as MIFF.
iterations=value	The number of times an image sequence loops before stopping.
label=”value”	This optional keyword defines a short title or caption for the image. If any whitespace appears in the label, it must be enclosed within double quotes.
matte=True, matte=False	Specifies whether a DirectClass image has matte data. Matte data is generally useful for image compositing. This keyword has no meaning for pseudocolor images.

Keyword/Value Combinations

Keyword=value (Cont.)	Definition
montage=<width>x <height>{+-}<x offset> {+-}<y offset>	Size and location of the individual tiles of a composite image. See X(1) for details about the geometry specification. Use this keyword when the image is a composite of a number of different tiles. A tile consists of an image and optionally a border and a label. <width> is the size in pixels of each individual tile in the horizontal direction and <height> is the size in the vertical direction. Each tile must have an equal number of pixels in width and equal in height. However, the width can differ from the height. <xoffset> is the offset in number of pixels from the vertical edge of the composite image where the first tile of a row begins and <y offset> is the offset from the horizontal edge where the first tile of a column begins. If this keyword is specified, a directory of tile names must follow the image header. The format of the directory is explained below.
packets=value	The number of compressed color packets in the image data section. This keyword is optional for RunlengthEncoded images, mandatory for Zip or BZip compressed images, and not used for uncompressed image.
page=value	Preferred size and location of an image canvas.
red-primary=x,y, green-primary=x,,y blue-primary=x,,y white-point=x,y	This optional keyword reflects the chromaticity primaries and white point.

Keyword/Value Combinations

Keyword=value (Cont.)	Definition
rendering-intent= saturation, rendering-intent=perceptual, rendering-intent=absolute, rendering-intent= relative	Rendering intent is the CSS-1 property that has been defined by the International Color Consortium.
resolution= <x-resolution>x <y-resolution>	Vertical and horizontal resolution of the image. See units for the specific resolution units (e.g., pixels per inch).
rows=value	The height of the image in pixels. This is a required keyword and has no default.
scene=value	The sequence number for this MIFF image file. This optional keyword is used when a MIFF image file is one in a sequence of files used in an animation.
signature=value	This optional keyword contains a string that uniquely identifies the image pixel contents. RSA's Data Security MD5 Digest Algorithm is recommended.
units=pixels-per-inch, units=pixels-per-centimeter	Image resolution units.

The following is a sample MIFF header. In this example, <FF> is a formfeed character:

```
id=ImageMagick class=PseudoClass colors=256
compression=RunlengthEncoded
```

```
packets=27601 columns=1280 rows=1024
scene=1
signature=d79e1c308aa5bbcdeea8ed63df412da9
{
Rendered via Dore by Sandi Tennyson.
}
<FF>
:
```

Note that *keyword=value* combinations may be separated by newlines or spaces and may occur in any order within the header. Comments (within braces) may appear anywhere before the colon.

If you specify the montage keyword in the header, follow the header with a directory of image tiles. This directory consists of a name for each tile of the composite image separated by a newline character. The list is terminated with a NULL character.

If you specify the color-profile keyword in the header, follow the header (or montage directory if the montage keyword is in the header) with the binary color profile.

Next comes the binary image data itself. How the image data is formatted depends upon the class of the image as specified (or not specified) by the value of the class keyword in the header.

DirectClass images (class=DirectClass) are continuous-tone, RGB images stored as intensity values in red-green-blue order. Each color value is one byte in size for an image depth of 8 and there are three bytes per pixel (four with an optional matte value). If the depth is 16, each color value is two bytes with the most significant byte being first. The total number of pixels in a DirectClass image is calculates by multiplying the rows value by the column value in the header.

PseudoClass images (class=PseudoClass) are colormapped RGB images. The colormap is stored as a series of red-green-blue pixel values, each value being a byte in size. If the image depth is 16, each colormap entry is two bytes with the most significant byte being first. The number of colormap entries is indicated by the colors keyword in the header, with a maximum of 65,535 total entries allowed. The colormap data occurs immediately following the header (or image directory if the montage keyword is in the header).

PseudoClass image data is an array of index values into the color map. If these are 256 or fewer colors in the image, each byte of image data contains an index value. If the image contains more than 256 colors or the depth is 16, the index value is stored as two contiguous bytes with the most significant byte being first. The total number of pixels in a PseudoClass image is calculated by multiplying the rows value by the columns value in the header.

The image data in a MIFF file may be uncompressed or may be compressed using one of two algorithms. The compression keyword in the header indicates how the image data is compressed. The run-length encoding (RLE) algorithm may be used to encode image data into packets of compressed data. For DirectClass images, runs of identical pixels values (not BYTE values) are encoded into a series of four-byte packets (five bytes if a matte value is included). The first three bytes of the packet contain the red, green, and blue values of the pixel in the run. The fourth byte contains the number of pixels in the run. This value is in the range of 0 to 255 and is one less than the actual number of pixels in the run. For example, a value of 127 indicates that there are 128 pixels in the run.

For *PseudoClass* images, the same RLE algorithm is used. Runs of identical index values are encoded into packets. Each packet contains the colormap index value followed by the number of index values in the run. The number of bytes n a PseudoClass RLE packet will be either two or three, depending upon the size of the index values. The number of RLE packets stored in the file is specified by the packets keyword in the header, but is not required.

Use Zip or BZip compression to achieve a greater compression ratio than run-length encoding. The number of compressed packets stored in the file is specified by the packets keyword in the header.

MIFF files may contain more than one image. Simply concatenate each individual image (composed of a header and image data) into one file.

Appendix D

Quantize

Overview

This document describes how ImageMagick performs color reduction on an image. To fully understand this chapter, you should have a knowledge of basic imaging techniques and the tree data structure and terminology.

For purposes of color allocation, an *image* is a set of n pixels, where each pixel is a point in *RGB space*. RGB space is a 3-dimensional vector space, and each pixel, p_i, is defined by an ordered triple of red, green, and blue coordinates, (r_i,g_i,b_i).

Each primary color component (red, green, or blue) represents an intensity that varies linearly from 0 to a maximum value, Cmax, which corresponds to full saturation of that color. Color allocation is defined over a domain consisting of the cube in RGB space with opposite vertices at (0,0,0) and $(C_{max},C_{max},C_{max})$. ImageMagick requires $C_{max}= 255$.

The algorithm maps this domain onto a tree in which each node represents a cube within that domain. In the following discussion, these cubes are defined by the coordinate of two opposite vertices—the vertex nearest the origin in RGB space and the vertex farthest from the origin.

The tree's root node represents the the entire domain, (0,0,0) through $(C_{max},C_{max},C_{max})$. Each lower level in the tree is generated by subdividing one node's cube into eight smaller cubes of equal size. This corresponds to bisecting the parent cube with planes passing through the midpoints of each edge.

The basic algorithm operates in three phases:

- Classification, which builds a color description tree for the image
- Reduction, which collapses the tree until the number it represents, at most, is the number of colors desired in the output image
- Assignment, which defines the output image's color map and sets each pixel's color by reclassification in the reduced tree

Our goal is to minimize the numerical discrepancies between the original colors and quantized colors. To learn more about quantization error, see Measuring Color Reduction Error.

Classification

Classification begins by initializing a color description tree of sufficient depth to represent each possible input color in a leaf. However, it's impractical to generate a fully-formed color description tree in the classification phase for realistic values of C_{max}. If color components in the input image are quantized to k-bit precision, so that $C_{max} = 2^k-1$, the tree would need k levels below the root node to allow representing each possible input color in a leaf. This becomes prohibitive because the tree's total number of nodes = $1+\mathrm{Sum}(8^i)$, i=1,k

For k=8, Number of nodes= $1 + (8^1+8^2+....+8^8)$ 8^8 - 1 = 1 + 8.----------- 8 - 1 = 19,173,961

Therefore, to avoid building a fully populated tree, ImageMagick does the following:

- Initializes data structures for nodes only as they are needed
- Chooses a maximum depth for the tree as a function of the desired number of colors in the output image (currently based-two logarithm of C_{max}).

```
For Cmax=255,
Maximum tree depth = log (255) 2= log (255) / log (2) e e=7.99 ~= 8
```

A tree of this depth generally allows the best representation of the source image with the fastest computational speed and the least amount of memory. However, the default depth is inappropriate for some images. Therefore, the caller can request a specific tree depth.

For each pixel in the input image, classification scans downward from the root of the color description tree. At each level of the tree, it identifies the single node which represents a cube in RGB space containing the pixel's color. It updates the following data for each such node:

Node Data

Node	Data
n_1	Number of pixels whose color is contained in the RGB cube which this node represents
n_2	Number of pixels whose color is not represented in a node at lower depth in the tree; initially, n_2=0 for all nodes except leaves of the tree.
S_r,S_g,S_b	Sums of the red, green, and blue component values for all pixels not classified at a lower depth. The combination of these sums and n_2 will ultimately characterize the mean color of a set of pixels represented by this node.

Node Data

Node	Data
E	The distance squared in RGB space between each pixel contained within a node and the nodes' center. This represents the quantization error for a node.

Reduction

Reduction repeatedly prunes the tree until the number of nodes with $n_2 > 0$ is less than or equal to the maximum number of colors allowed in the output image. On any given iteration over the tree, it selects those nodes whose E value is minimal for pruning and merges their color statistics upward. It uses a pruning threshold, E_p, to govern node selection as follows:

```
Ep = 0
while number of nodes with (n2 > 0) > required maximum number of colors
prune all nodes such that E <= Ep
Set Ep to minimum E in remaining nodes
```

This has the effect of minimizing any quantization error when merging two nodes together.

When a node to be pruned has offspring, the pruning procedure invokes itself recursively in order to prune the tree from the leaves upward. The values of n_2,S_r, S_g, and S_b in a node being pruned are always added to the corresponding data in that node's parent. This retains the pruned node's color characteristics for later averaging.

For each node, n_2 pixels exist for which that node represents the smallest volume in RGB space containing those pixel's colors. When $n_2 > 0$ the node will uniquely define a color in the output image. At the beginning of reduction, $n_2 = 0$ for all nodes except the leaves of the tree which represent colors present in the input image.

The other pixel count, n_1, indicates the total number of colors within the cubic volume which the node represents. This includes $n_1 - n_2$ pixels whose colors should be defined by nodes at a lower level in the tree.

Assignment

Assignment generates the output image from the pruned tree. The output image consists of two parts.

- A color map, which is an array of color descriptions (RGB triples) for each color present in the output image.
- A pixel array, which represents each pixel as an index into the color map array.

First, the assignment phase makes one pass over the pruned color description tree to establish the image's color map. For each node with $n_2 > 0$, it divides S_r, S_g, and S_b by n_2. This produces the mean color of all pixels that classify no lower than this node. Each of these colors becomes an entry in the color map.

Finally, the assignment phase reclassifies each pixel in the pruned tree to identify the deepest node containing the pixel's color. The pixel's value in the pixel array becomes the index of this node's mean color in the color map.

Empirical evidence suggests that the distances in color spaces such as YUV, or YIQ correspond to perceptual color differences more closely than do distances in RGB space. These color spaces may give better results when color reducing an image. Here the algorithm is as described except each pixel is a point in the alternate color space. For convenience, the color components are normalized to the range 0 to a maximum value, Cmax. The color reduction can then proceed as described.

Measuring Color Reduction Error

Depending on the image, the color reduction error may be obvious or invisible. Images with high spatial frequencies (such as hair or grass) will show error much less than pictures with large smoothly shaded areas (such as faces). This is because the high-frequency contour edges introduced by the color reduction process are masked by the high frequencies in the image.

To measure the difference between the original and color reduced images (the total color reduction error), ImageMagick sums over all pixels in an image the distance squared in RGB space between each original pixel value and its color reduced value. ImageMagick prints several error measurements including the mean error per pixel, the normalized mean error, and the normalized maximum error.

The normalized error measurement can be used to compare images. In general, the closer the mean error is to zero the more the quantized image resembles the source image. Ideally, the error should be perceptually-based, since the human eye is the final judge of quantization quality.

These errors are measured and printed when -verbose and -colors are specified on the command line:

- *mean error per pixel* is the mean error for any single pixel in the image

- *normalized mean square error* is the normalized mean square quantization error for any single pixel in the image

  This distance measure is normalized to a range between 0 and 1. It's independent of the range of red, green, and blue values in the image.

- *normalized maximum square error* is the largest normalized square quantization error for any single pixel in the image.

  This distance measure is normalized to a range between and blue values in the image.

Appendix E
XTP

Overview

XTP is a utility for retrieving, listing, or printing files from a remote network site, or sending files to a remote network site. XTP performs most of the same functions as the FTP program, but it doesn't require any interactive commands. You simply specify the file transfer task on the command line and XTP performs the task automatically.

Syntax

```
xtp [ -options ... ] <uniform resource locator>
```

Examples

- To retrieve the file bird.jpg in directory images from host wizard.mystic.es.dupont.com, use

  ```
  xtp ftp://wizard.mystic.es.dupont.com/images/bird.jpg
  ```

- To retrieve all the files from directory images from host wizard.mystic.es.dupont.com, use

  ```
  xtp -retrieve ftp://wizard.mystic.es.dupont.com/images/
  ```

You will be prompted for a password.

- To retrieve all the files from directory images as user cristy and password magick from host wizard.mystic.es.dupont.com, use

```
xtp -retrieve ftp://cristy:magick@wizard.mystic.es.dupont.com/images/
```

XTP Options

-account *password*

-binary

`+binary`

-directory

-exclude *expression*

`-directory -print -retrieve`

-file name

-get -put

-get

ftp get
ftp mget -file

-ident *password*

-port *number*

-proxy *hostname*

variable xtp_proxy Environment
+proxy

-print

-prune

Note: This option does not recursively search for files.

-put

`ftp put`

`ftp mput`

`-file`

-retrieve

-timeout *seconds*

-type *name*

-verbose

Using XTP Options

-directory

-print -put -retrieve

-get

This option has the format

```
protocol://host/[directory/[filename]]
```

protocol *host*

User *password* *directory/[filename]*

`ftp://host//tmp/anyfile`

`-get` `-put`

`-get` `-put`
*ls-lls-l([Rt])+([Rt])**

`-prune`

Regular Expressions

regular expression `|` *branch*

piece `*` `+` `?` `*`
`+` `?`

Files

~/.netrc

Environment

xtp_proxy

-proxy hostname

Appendix F

Acknowledgments

Author

John Cristy, *magick@wizards.dupont.com*, E.I. du Pont de Nemours and Company Incorporated.

Contributors

Rod Bogart and **John W. Peterson**, University of Utah. Image compositing is loosely based on rlecomp of the Utah Raster Toolkit.

Bob Friesenhahn contributed and maintains the Configure scripts. In addition, Bob wrote a PERL script to format the ImageMagick C API documentation, wrote the PerlMagick regression tests, proposed the Delegate subsystem, and wrote Magick++, an ImageMagick C++ API wrapper.

Michael Halle, Spatial Imaging Group at MIT, contributed the initial implementation of Alan Paeth's image rotation algorithm.

Peder Langlo, Hewlett Packard, Norway, submitted hundreds of suggestions and bug reports. Without Peder, ImageMagick would not be nearly as useful as it is today.

Rick Mabry added tiled drawing pens to ImageMagick, as well as anti-aliased drawing primitives.

The **MIT X Consortium** made network transparent graphics a reality.

David Pensak, E. I. du Pont de Nemours and Company, provided a computing environment that made this program possible.

Bill Radcliffe, contributed the FlashPix and IPTC support.

Paul Raveling, USC Information Sciences Institute. The spacial subdivision color reduction algorithm is based on his Img software.

Steve Singles, University of Delaware, contributed the initial implementation of xtp.

Henry Spencer, University of Toronto, contributed the implementation of the xtp regular expression interpreter and the text in Regular Expressions on page 465.

Many thanks to the hundreds of people who have submitted email with bug reports and suggestions for improving ImageMagick.

Manual Design and Compilation

Rebecca Richardson, technical writer, gathered the web resources, edited them, and formatted them into this guide. Rebecca can be contacted at *recbecca1@earthlink.net*

Index

Numerics

A

B

C

D

J

K

L

M

O

P

Q

R

S

Z