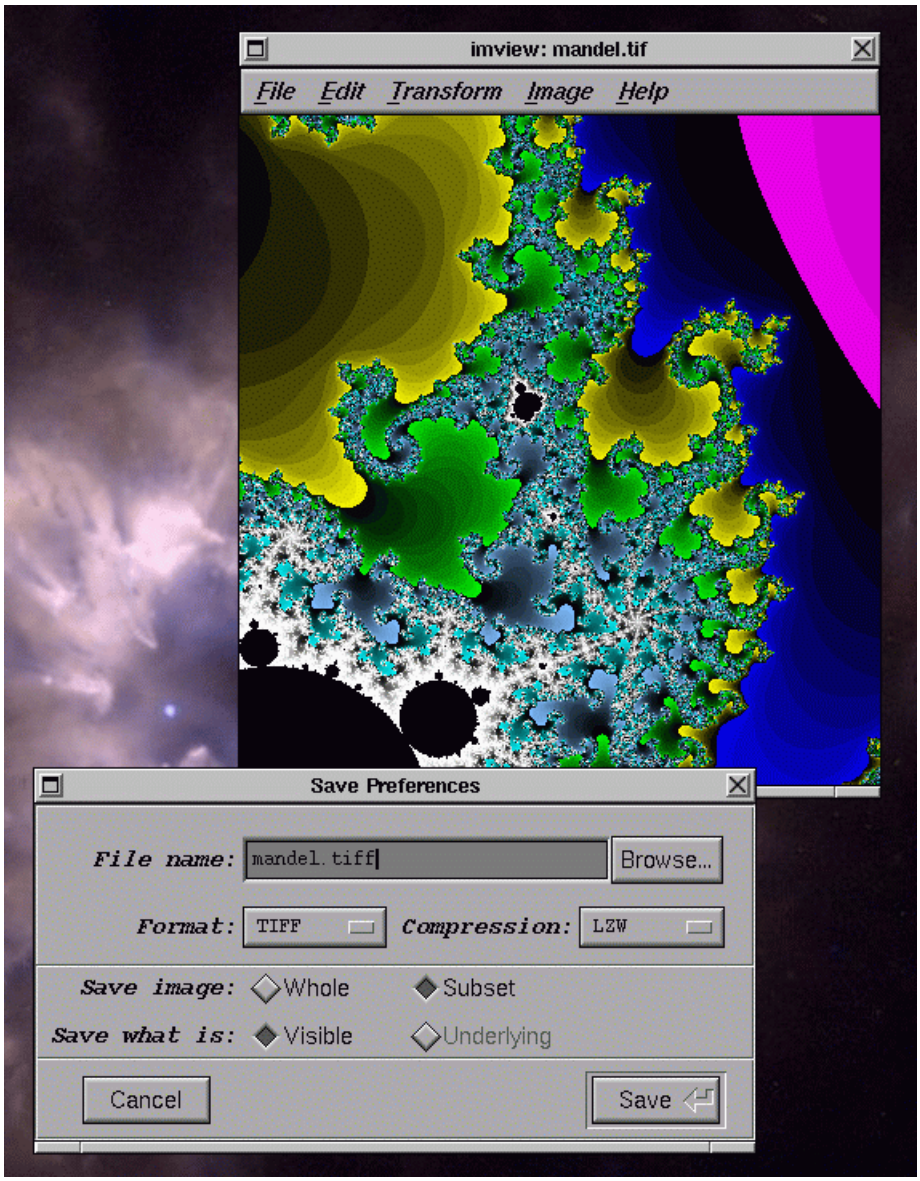
ImView: a portable image display application

Hugues Talbot June 19, 2003– version 1.0.1

This is the documentation for `imview`, a portable image viewing application.



Contents

Forewords

When I started `imview`, I had no idea it would take me so long to get where I wanted, that it would be so big, or that I would write a 100+ pages document about it.

So you're staring at a monster. I've intended this documentation primarily for my colleagues and as a reference document, not as bedtime reading.

Normally, `imview` should be somewhat intuitive, but has a few quirks. Hopefully I've documented all of them. When you encounter something strange, look in this document and you should see a few words about it. If not contact me.

Thansk for your interest in `imview`.

Chapter 1

What is new?

This chapter allows bleeding-edge users to get the last word on what is happening. Assuming anyone is interested, of course.

See appendix D for a detailed history of changes and more recently to the `ChangeLog` file in the source distribution.

- This documentation now refers to version 1.0.1 of `imview` and anticipate in places features present in 1.1.x (these are clearly marked). This version is pretty much 0.9.x with a number of bugs removed. In addition there is a new colour brightness-contrast-gamma panel to supplement the grey-level one. `Imview` 1.0.x is based on FLTK-1.0.x, future releases will be based on FLTK-1.1.x.

- version 0.9.x introduced image rotations (2D and 3D) and mirroring, improved pointfile handling (much faster), image overlays including transparency, text annotation (coupled with points), a primitive toolbar, the possibility to hide the main menu, saving of the raw image data, a 'stretch-to-fit' mode where resizing the main window also resizes the image within it, improved dialog resizing, POSIX IPC (on Solaris and Compaq Tru64 only), a TCP client so that several imviews can be linked now, and lots of bugs fixes and minor enhancements.

- Since the last beta (0.2.x) a lot has changed. `Imview` can now behave as an 'image server'. This has yet to be fully documented, but is already used heavily in Voir, our group's new image analysis environment. Have a look at chapter 5 for the current state of the documentation on this issue.

- Imview now links with ImageMagick, which gives it access to a lot of image format (read and write). It also comes with a suite of new hassles.

- a lot of Windows-specific quirks have been eliminated. It even works on multi-cpu machines, with the server stuff and all. Quite an achivement I thought...

- This document now has a windows-specific section in Appendix A.
- `Imview` now has a interactive bivariate histogram window. Useful for colour or multi-spectral image segmentation.
- `Imview` can now rotate (in 90^{o}increments) and flip images.
- `Imview` supports the `-gamma <value>` command line argument to setup a default gamma for all images.
- `Imview` can print a list of images on one page, just like the `laser` function of Z-IMAGE. There is a preview option on the print panel (Ghostview or some postscript previewer must be installed however).
- `Imview` now works fine with SZ.
- `Imview` now interfaces with Ghostscript to display PS files (any file, not just images). Rendering options are user-configurable (resolution, etc).
- `Imview` now saves and reads user-configurable options.
- `Imview` now runs on Windows NT and Windows'95! Tested on NT3.51, NT4.0 and at home on '95. It seems to work best in NT3.51 for some reason.
- `Imview` can show 1-D arbitrary profiles of 2-D images. These profiles can be saved and printed.
- Zoom regions of interest (ROI) can now be defined with arbitrary aspect ratios.
- `Imview` has some hyperspectral images support: spectra can be displayed interactively, and can be saved or printed.
- `Imview` now has contrast, brightness and gamma control. It is now a complete replacement for `x11()`. You can use these controls for interactive thresholding!
- `Imview` can now save images to a file (in TIFF or Z-IMAGE formats only at present, Postscript is also supported under a different mechanism (printing)).
- `Imview` can now print or save images in Postscript format.
- `Imview` now supports multi-frame (or multi-page) documents, such as TIFF stacks, animated GIF, and non-multispectral, multi-component Z-IMAGE files.
- `Imview` now has a `help` menu. It points both to a quick online help and this present document in HTML format (it automatically launches netscape to read it).
- `Imview` now supports pointfiles in a complete way. 'breaks' (or point grouping) support is now working and is much more useable than under `x11()`: for a start the groupings are now visible (points in the same group are linked by line overlays).

- `Imview` now reads the ICS format for grey-level images (char, short and floats).
- `Imview` redraws images and overlays (such as information box and zoom boxes) much faster than before.
- `Imview` now supports 3-D images in the same way that `x11()` and scilimage do: users can flip to and fro through 2-D sections of the Z plane.
- Updated feature list, TODO list and bug list.
- Possibly interesting CPU and memory benchmark section (see section B).

Chapter 2

A (long) introduction to imview

People not interested in self-justificating rambling and other dubious philosophical questions on application development, but want to know how to use the damn software can jump straight to chapter 4.

On the other hand, those who want to know a bit more about `imview`, where it comes from and where it is headed can read on at their own risk.

Be aware that some comments may be rather cryptic if you are not one of my immediate colleague, in particular all the business about `x11()`, which is basically the image viewer that my research group was using before `imview`.

2.1 What is imview?

`imview` is a Unix and MS-Windows application originally designed both to replace the `x11()` function of Z-IMAGE, that is now quite dated, and also to provide a general image viewing tool.

For users who are not directly my colleagues or who have never worked with Z-IMAGE, `x11()` is a basic image display program for Unix/X11 that rely on pseudocolour terminal functionalities, written using the X toolkit (`Xt`) and the Athena widget set.

`Imview` is more modular in design than `x11()` is, and provides vastly more functionality than `x11()`. It is being built in C++ with a GUI-builder library called `fltk`, available at

`http://www.fltk.org`

It is based on a prototype which was developed between 1991 and 1994 in Objective C on NeXTStep by the same author. It is concurrently being developed on Linux, DEC Unix, Solaris and Windows NT. SGI IRIX will probably enter the development circle one day.

2.2 Why did I start on `imview`?

In 1994 when I joined the Image Analyis project at CSIRO, the Commonwealth Scientific and Industrial Research Organization, Sydney, Australia, The de-facto standard image analysis package for this group was a home-grown solution called Z-IMAGE. It was a Unix-only system build around a non-traditional shell which somewhat resembled the statistical `S` package from Bell Labs. It has nice image arithmetics, advanced functions were run as separate processes communicating with the shell using files.

At that time 24-bit, full-colour displays were still an expensive rarity. We still had binary X-terminals (who could only display pure white and pure black. One had to use dithering to get a vague impression of grey-levels), and most terminals or workstation only used 8-bit colour. The image display program everybody used was called `x11()`, which had been developed for Z-IMAGE and *only* supported 8-bit colour.

I had been spoiled during my previous PhD years with the use of a NeXT cube with NeXT-dimension full colour graphic card, which had a separate i960 processor and 8MB of memory just for itself (the cube had 32MB, a lot in those days). After a few hard months I finally got a lowly PC (a Pentium 75) at home, with a run-of-the-mill S3 video card which could do 1024x768 in 16-bit colour (in truth a bit more), running Linux 1.2.x (Slackware distribution). At work I also eventually got an Alpha with a very nice 24-bit display. No one was talking of accelerated 3D except the SGI guys in those days.

Well, the Z-IMAGE `x11()` function never worked well for me, especially as I work on a TrueColor display both at work and at home. `x11()` simply didn't work at all on my 16-bit display at home, and worked very poorly on my 24-bit screen at work (colours were unpredictable, brightness and contrast manipulation didn't work, colours flashed all the time).

Before I started on `imview`, on X I was always using `xv`, which is still today a very good program even though the last revision was sometime in 1993, but was noticeably slower than `x11()`, is no longer maintained anymore, and has a few quirks that I find annoying. Most importantly it doesn't have colourmap manipulation of the kind that we used in Z-IMAGE, it doesn't understand anything other than CHAR images (grey-level or RGB), it doesn't use dithering with or without error diffusion except for 8-bit colour, so grey-level images looked actually quite poor on my screen at home, `xv` was never a real Z-IMAGE function and so has poor command line interface with Z-IMAGE (impossible to display an image with a given colourmap for example) and finally it didn't work with pointfiles. The upshot is that I was constantly switching between those programs (except at home where I couldn't anyway because `x11()` didn't work unless I downgraded my screen quality to 8-bit...).

`imview` started as a project provide the facilities I most want from both programs, and a lot more if at all possible.

2.3 Why not simply improve x11()?

I tried to do that, but although x11() is well-written and efficient, it is designed to work with X11 pseudo colour displays, i.e: 8-bit displays with colourmap. Making x11() work on truecolor displays would have been very onerous and equivalent to starting all over again.

This is what I did with the help of a reasonable GUI-buiding library. Finding a reasonable library that could work well with images was difficult by the way. I was looking for something free (GNU licence if possible), with good documentation, source availability, if possible support, consistent, fast and in C++ or Objective-C (because object-oriented approaches work really well with GUI development). Fltk is the only one that fitted this bill at the time of writing. Possibly today I would start with Qt, but I'm not even sure. The FLTK is really well done, concise and efficient. There are a number of GUI building libraries in the public domain, but they either don't work well with images (V), require a non-free library (wxWindows used to require MOTIF. It now works with the GTK, however), are not very well documented (countless others including the GTK), don't work with C++ or Objective-C (gtk, although this is possibly improving), don't come with the source (Xforms), or are not really available yet or too incomplete (GNUstep). I found fltk by chance after I had started with Qt, and couldn't believe my eyes. It even looked good! (as good as anything can be on X11) and uses a subset of C++ that I could perfectly live with (no templates, exceptions, friend classes, operator overload, etc) yet provides an object-oriented framework. Fltk is exceptionally stable, and its author and contributors provides excellent support via a mailing list and a web site. On top of it, it works with Windows!

It is also amazingly simple to program with. I recommend it to anybody willing to start on GUI programming. Quite an eye-opener compared to the excesses of more advanced toolkits.

2.4 imview's aims

The ultimate end of imview is to be able to

- understand a large number of image formats,
- display 2D and 3D images in a reasonable way,
- display multicomponent and multispectral images,
- display INT and DOUBLE (and more pixel types) images transparently,
- display time series of images (as a movie or frame by frame),
- display labelled objects (and their fitted ellipse, or maximum enscribing rectangle, etc),
- apply simple transforms to images (such as apply colourmaps, rotate images, etc),

- provide information on 1-D profiles anywhere in the image.
- manipulate image histograms (apply gamma correction, manipulate brightness and contrast, provide interactive thresholding, etc),
- work with pointfiles,
- get pixel-wise information (grey-level or multispectral information under the cursor),
- work on most (if not all) X11 screen depths (8-bit, 16-bit, 24-bit, 32-bit),
- provide a reasonable command-line interface at start-up in addition to the GUI. `Imview` will perform as well under Z-IMAGE as under the normal Unix prompt.
- and more.

Obviously all of this did not happen in one day. I started in 1997, and in 2003 I was amazed to see that "display labelled objects" is still not implemented.

In general, `imview` does provide access to image data and some image manipulation facilities (like zoom and rotations for example) to display image data in a more useful manner than is currently possible with other available tools. However, the principal aim of `imview` is not to provide image manipulation tools for its own sake, as with Photoshop or the GIMP or even `xv`. In some sense `imview` is an *image analysis* tool.

2.5 What `imview` can do at present

As of this writing, `imview`'s version is 1.0.1 which means its first stable release. It is still incomplete, however, it has survived three Beta programs (including external paying clients!) so I'm becoming more and more confident about this software now, even though it still has obvious flaws. The current `imview` release has the following features:

- Displays Z-IMAGE, TIFF, GIF, ICS, PNM and JPEG images. Can display INT and DOUBLE Z-IMAGE images with the wrong endianness transparently (ie: a Z-IMAGE file created on a Sun can be displayed on a DEC, and vice-versa). Images can be read in from the standard input. Imview interfaces with Ghostview to display postscript files.
- Version compiled with ImageMagick can read about 60 other common image formats, among them PNG, Sun Raster, etc.
- Displays 2D images with a very good zoom and pan feature.
- Displays 3D images reasonably. The same zoom and pan feature is available but only one image slice can be viewed at any time (no rendering).

- Displays Grey-level as well as RGB data. Other multispectral images are read in but only one spectrum can be displayed at a given time. The user can flip back and forth through the components, of course.
- Displays INT and DOUBLE images transparently (and other pixel types such as SHORT and FLOAT also available in TIFF and ICS images).
- Has useful features for hyperspectral images, such as a real time display of the spectrum at each point.
- Arbitrary 1-D profile of 2-D images (or 2-D slices of 3-D images) can be displayed.
- Works with multi-page documents (TIFF stacks, animated GIFs and heterogeneous multi-component Z-IMAGE files).
- Works with pointfiles (in a rather complete and fancy way...). Points are clearly visible and points belonging to a group (separated by 'breaks') are linked on-screen by overlaid lines.
- Images can be zoomed, panned and colourmaps can be applied on CHAR, SHORT and INT data.
- Has pixel-wise information available (displays FLOAT, DOUBLE, SHORT and INT information in their native format).
- Works on most screen depths (8-bit to 32-bit). It doesn't work on monochrome (1-bit) displays and probably never will, and it is designed to work best on truecolor displays (15-bit and above).
- Images or parts of images can be printed to a device or to a file.
- Images or parts of images can be saved in TIFF or Z-IMAGE format.
- `Imview` has brightness/contrast/gamma control. These can be used for interactive thresholding.
- Many options are already available on the command line. `Imview` does work both as a Z-IMAGE function and as a Unix command line function.
- `Imview` is portable: it works the same under Windows (win32).
- `Imview` usually runs as fast or nearly as fast as `x11()`, much faster than `xv`, and uses significantly less memory than either. AFAIK nobody cares about `x11()` anymore, even my most conservative co-workers. Other viewers have sprouted thanks to the wonderful KDE and Gnome efforts, but still none really compares to the old `xv` in my opinion. For what I do `imview` is better than `xv` but not for everything. (for example `xv` can grab windows, put images in the root window, and does offer some basic filtering).

Most of the work so far has been spent in providing the code infrastructure to make this possible. Most of the work is actually invisible (no visible user controls correspond to them). Just putting an image on the screen is not so easy under X11, even with the help of fltk.

2.6 What `imview` cannot do at present

Of course, many features are missing from an ideal point of view, but my primary goal was to write a complete replacement for `x11()` that would also work on non-8-bit, true-colour displays. I feel that I've achieved this goal since about version 0.9.0, and that `imview` is indeed a complete replacement for `x11()`, except possibly on pure 8-bit screens. As far as I know no one uses those anymore for anything serious.

There is a wish-list of the features I personally want put into `imview` in section 6, I'm welcoming feedback on this issue and the rest as well, of course.

Chapter 3

Tutorial

This chapter is a quick tutorial to `imview`.

3.1 Starting up

Currently, the images used in this tutorial are located in

`/home/talbot/projects/GUI`

Change directory to there or copy the images somewhere else and follow the directions.

3.2 Basic GUI

3.2.1 Menu basics

Type

```
% imview
```

This is the splash screen. Try selecting the `File` menu with the mouse. Then click anywhere on the main window to make the submenu go away. Try selecting an item with the mouse, for example `Open` in the `File` submenu (we will note this item `File->Open` from now on). Click on `Cancel` to dismiss the dialog.

Now try holding down the `Alt` key while pressing the `f` key. In the future we we note this `alt+f` and call this a *shortcut*. Try navigating the menu with the arrow keys. Notice as the selection skips over non-available items. Notice also that most items have a shortcut attached to them (`alt`+something most of the time). The main submenu have a letter underlined (for

example `File` has an underlined `F`). This indicates that this letter is a shortcut available with the `alt` key. Indeed `alt+f` is a shortcut to access the `File` submenu.

Now select one of the available items by typing return. For example let us try `alt+h`, then arrow key down, then return. You should see the `About Imview` dialog. Type Esc once to quit the dialog.

Now try `alt+f`, then arrow key down, then return. This is the `Open Image` dialog. Type Esc to close the dialog. Now try `alt+o`. This does the same thing: This is a menu item shortcut. It allows the user to select the item in a minimal number of key strokes.

3.2.2 Loading an image

Type `alt+o`, or open the `Open Image` dialog any way you like. It this dialog, the button on the left hand side navigate the directory hierarchy. Try going up and down directories. The underlined letters in the left hand side buttons also indicate shortcuts, just like in the main menu. Try them.

To go up a directory, press the `Up one directory` button. To go down one directory, select any directory available from the right hand side list. **Do not double-click on them**. Directories end with a `/`. Normal files do not. To go back to the directory where you started `imview` in, press the `Current dir` button.

To view only the directories, press the `*/Directories` button. Click on `*` to see the normal files again. To see all the files, including the normally hidden one (under Unix, those starting with `.` and a few others), press the `* All files` button.

To go to your home directory, click on `/Home`. To go to the Root of the current disk, click on `/ Root`.

Try selecting a normal file (click on an item on the right-hand side list that do not end in `/`). This file name appears on the input field at the bottom of the dialog. Try navigating the list of files and directories with the up and down arrow. See what happens when you select a directory this way and press the `Enter` key.

Now click in the input area at the bottom of the dialog. Try typing a full path to a known image, for example `/home/talbot/projects/GUI/ibr001_z`. What happens when you type after each keypress? After having typed `/home/tal`, try pressing the `Tab` key. Notice that the directory name is completed for you. This allows the user to type in full paths quickly.

If you know Emacs, try typing the line editing command keys, such as `Control+a`, `Control+e`, `Control+f`, `Control+b`, etc. Try selecting a section of the path with the mouse, and then erase it with `Backspace`.

Now try selecting an image, for example `ibr_001.tiff`. Press the `Enter` key, or click on the `OK` button, or double-click on the file name. An image should appear.

3.2.3 A look at the menus again

Notice that now the menus are a bit different. Have a look. For example, in the `File` submenu, the `Print` and `Close` items are selectable. Try closing the image with the `Close` item.

Then try loading the image again. Notice that the `Open Image` dialog is in the same state as just before you loaded the image into `imview`. Load the same image again,

Then try typing `alt+i`. This gives you some information on the current image.

Try loading different kind of images. `Imview` understands Z-IMAGE, TIFF, GIF, PNM and ICS images.

3.2.4 A systematic try of all the menu items

This is not the matter of this short tutorial. Look up the user manual to see what each menu item does.

3.3 The mouse on the image

3.3.1 Pixel information

With an image loaded into `imview`, place the mouse anywhere on the image, and then press and hold down the left button mouse. Then drag the mouse around. Notice that information on the pixel just underneath the mouse pointer on the screen is made available to you.

Notice that when you first click the left mouse button, the information window appears at the top or at the bottom of the image depending which way is furthest from the position of the mouse.

3.3.2 Distance and angles

With an image loaded into `imview`, place the mouse anywhere on the image, hold down the `Control` key, click and hold down the left mouse button, then drag the mouse around. Notice that a line is being drawn on the image, and that corresponding distance and angular measurements are made available.

3.3.3 Zoom with the mouse

With an image loaded into `imview`, place the mouse anywhere on the image, and then press the middle button, then drag the mouse down and to the right. This defines at *Region of Interest* (ROI). Then release the middle button.

Notice that the ROI has the same aspect ratio as the main window. This is called a *constrained zoom*. Notice that the main window does not need to be resized when doing this sort of zoom, and that therefore the zoom is fast.

Now on the zoomed image, click with the right button mouse. This is the *Unzoom* mouse button.

Now hold the `Shift` key down, press the middle mouse button and then drag the mouse down and to the right. Notice that this time the ROI is not constrained to the aspect ratio of the main window. Release the middle button. This is called an *unconstrained zoom*. Notice that this time the main window needs to be redrawn and is a bit slower.

Click with the right button mouse to unzoom.

3.3.4 Profiles

Press `Alt+1` or select the `Image->1-D profile` menu item. The `Display Profile` dialog appears. Now try to obtain some distance and angular measurement on the image just like in section 3.3.2. That is, hold down the `Control` key, press and hold the left mouse button and drag the mouse around.

The grey-level profile under the line defined in this way is shown

3.3.5 Points

To define "geom" points, hold down the `Shift` key and click on the image with the left mouse button. Geom points can be deleted via the menus or with `alt+d`.

To define a "break", hold down the `Shift` key and click on the image with the right mouse button. The breaks can also be deleted.

Point can be saved in files (pointfiles). Pointfiles can be read in, etc.

3.3.6 Colourmaps

3.3.7 Contrast/Gamma

3.3.8 Histogram

3.3.9 Printing

3.3.10 Saving

3.4 The command line

Type `imview -h` for help on the Unix command line.

3.5 Interaction with Z-IMAGE

Attach `imview` like any normal Z-IMAGE function. Type `imview()` for help on the available command line arguments.

3.5.1 Useful tricks

3.5.1.1 Many images on the command line

Try `imview(image1,image2,...)`

Switch between them with `<spacebar>` and `Shift+<spacebar>`. Try this while zoomed in.

3.5.1.2 Parameter search

Press `<spacebar>` to reload a modified image.

Chapter 4

User manual

This chapter describes the user interface to `imview`.

4.1 Conventions

Documenting software usage can be a bit difficult. In the following sections, some conventions on typography and names are used.

4.1.1 Typed text

In order to help the reader a little bit with visual cues, the following textual conventions are used:

Text that can be typed or viewed at the console appears in fixed-width font `like so`.

Long stretches of such text, such as example output produced by the program appear in framed centered boxes.

Menu shortcuts appear in fixed-width font with `a box` around them.

4.1.2 Menus and menu items

Figures of menus and menu items will be shown whenever possible. Menus and menu items will always be shown in fixed-font when described in the text, and will sometimes be referred to with a text construction like this: `Transform->Pointfiles->New pointfile`. This construction refers to the `New pointfile` menu item under the `Pointfile` submenu of the `Transform` submenu of the main menu, as shown in Fig 4.1:

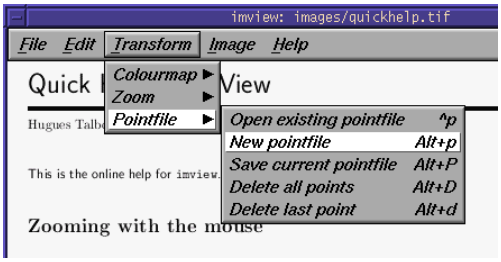


Figure 4.1: This is the `Transform->Pointfiles->New Pointfile` menu item.

4.1.3 Mini-glossary

The following naming conventions are used:

4.1.3.1 Menu

Main menu	The main menu of the application
Menu	Any of the top-level items, itself a menu, of the main menu: `File`, etc.
Submenu	Any of the items, itself a menu, of a menu or a submenu: `Transform->Zoom`, etc
Item	Any of the items, *NOT* a menu, of a menu or a submenu: `File->Open`, etc

4.1.3.2 Mouse

Click	Pressing and releasing a mouse button quickly.
Press	Pressing one of the mouse buttons and keep it pressed.
Drag	Pressing a mouse button and moving the mouse about with the button still pressed.
Release	Releasing a mouse button.

4.1.3.3 Windows

Main window	The window with the main menu on top, where images are displayed.
Dialog	A secondary window that is usually not present but pops up as the result of some action.
Modal dialog	A dialog that interrupts the flow of the application and that must be attended to for the normal flow of the program to resume.

4.1.3.4 Subdividing the image

`Imview` supports a great variety of images. It is not limited to 2-D or grey-scale images or even RGB images. The price of this absence of limitation is a relatively high degree of complexity. One problem is to name unambiguously all the subsets of image data. Unfortunately there isn't a single non-ambiguous nomenclature concerning this topic. We will follow the following conventions:

- The size of an image along the x dimension is sometimes called the number of **columns** of the image.
- The size of an image along the y dimension is sometimes called the number of **rows** of the image.
- The size of an image along the z dimension is sometimes called the number of **planes** of the image.
- Images come in image files. A single file can contain more than one individual image. Such files are called **multicomponent images**, for want of a better expression. Components in a multicomponent image need not have any relation to each other, although normally they do.
- A multicomponent image made of images that are all of the same pixel type and of the same dimensions, and such that each pixel location in each of the components correspond to the same *physical* location, is called a **multispectral image**. For example an RGB image is a multispectral image, and so is a Landsat or a SPOT image. Multispectral images with more than a handfull of components will be called **hyperspectral images**, such as for example the ARIES data.
- Each individual component in a multispectral image is called an **image sample**, for example the G component of an RGB image.
- The succession of values along all samples in a multispectral image at a given location on the image is called a **spectrum**.

In summary:

Number of columns	size of the image in x
Number of rows	size of the image in y
Number of planes	size of the image in z
Image subset	any part of an image. A single line or even a pixel in any image is an image subset
Image component	an element of an arbitrary multicomponent image that is still an individual image, i.e: the G component of the lowest resolution sub-image of a multiresolution Kodak photodisk image.
Image sample	an individual component of a multispectral image, i.e: the R component of an RGB image.
Image spectrum	the succession of values along all samples in a multispectral image at a given location.
Image frame	a single multispectral image in a multicomponent image i.e: a single RGB image in a movie is an image frame.

It follows from this that an image composed of a single RGB component is a single-frame image according to this definition. However, it is a multi-sample, or multispectral image, as well as a multi-component image, of course.

In contrast, an image composed of a series of RGB images, for example the scanning of the pages of a colour book, is a multi-frame image. Frames in a multi-frame image need not have the same type or the same dimensions.

4.2 Starting up

Starting up `imview` yields different results depending on whether it is started from the Unix prompt or from the Z-IMAGE prompt.

4.2.1 The Unix prompt

At the Unix prompt, if the following text is typed:

```
% imview
```

then a "splash screen" appears which at the moment is a bit gaudy (Fig. 4.2):

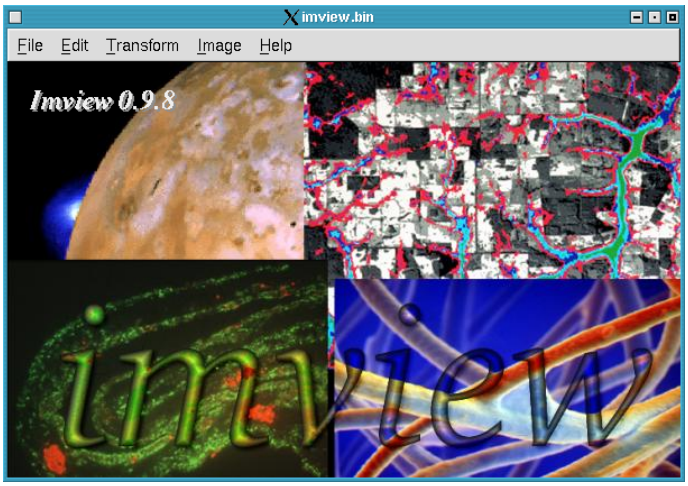


Figure 4.2: Newfangled garish splash screen.

If the following text is typed:

```
% imview <animage> <anotherimage> ...
```

Then the first of these images in the argument list will show on screen (if it is in one of the supported formats).

4.2.2 The Z-IMAGE prompt

At the Z-IMAGE prompt, if `imview` is typed, the `usage` is printed, as is the standard under Z-IMAGE.

```
Zimage-5.0.14 > imview
Purpose:  displays images under X11
Usage :  imview(<image1>,[cmap:"std_grey"],<image2>,[cmap:"std_grey"],...
                ,[CMAP:"std_grey"],[debug:"T/Y"])
...  etc ...
```

To actually display an image under Z-IMAGE, something along the line of:

```
Zimage-5.0.14 > files
myimage
Zimage-5.0.14 > imview(myimage)
```

must be typed. We can now proceed to the description of the menus and other features.

4.3 The main menu

The `imview` main menu is a standard horizontal menu as found on most GUI applications (see Fig. 4.3).

Figure 4.3: Standard application menu.

Menu items can be pointed at with the mouse as usual. They can also be called with "shortcuts", which means holding the `alt` key and simultaneously pressing a letter. All the underlined letters in the main menu indicate a shortcut for a submenu. For example `alt+f` (holding the `alt` key and pressing the `f` key) invokes the `File` submenu.

After invoking a submenu, the whole menu can be navigated with the arrow keys. Typing `return` on a menu item selects that item. Some items can be selected with shortcuts without needing to go to a submenu and navigating them. A complete list of the shortcuts is given in section 4.13, and shortcuts are also indicated in the following explanatory sections.

4.3.1 The `File` submenu

This submenu can be invoked with the `alt+f` shortcut and deals with the reading and writing of images.

Figure 4.4: The `file` submenu.

The `file` submenu performs the following actions:

4.3.1.1 The `Open` item

Can be invoked with `alt+o`.

Selecting this item causes the `Open image` dialog to be called up (see Fig 4.5).

Most of the controls on this dialog are relatively obvious. Use the left hand side buttons to change the current directory. Clicking only once on a directory in the file list opens it in the dialog. Double-clicking on a file opens it in `imview`, and so does single-clicking on it and then clicking on the OK button. The Cancel button dismisses the dialog.

In addition, a path to an image can be typed by hand in the input section at the bottom of the dialog. The input supports file name completion (with the `tab` key); the content of the

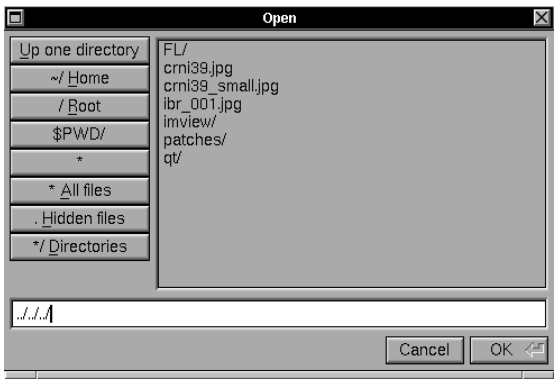


Figure 4.5: The `Open Image` dialog.

directory display will change after each keystroke when the path is typed, and a subset of the Emacs control keys are supported:

- `control+a` to go back to the beginning of the line,
- `control+e` to go to the end of the line,
- `control+f, or right arrow key` to advance one character to the right,
- `control+g, or left arrow key` to go back one character to the left,
- `control+k` to kill the end of the line,
- `control+w` to kill the selection,
- `control+_` one level undo,
- `Tab` to perform name completion,
- RETURN accept the seletion (same as clicking on the OK button),
- Esc dismiss the dialog (same as clicking on the Cancel button).

4.3.1.2 The `Save` item

Can be invoked with `alt+s`

This item is available only if an image is currently being displayed. If selected, the `Save Preferences` modal dialog pops up (see Fig 4.6).

In this dialog, the diamond-shaped buttons are called 'check buttons'. A check button is selected if it appears pushed in and dark. It is deselected if it doesn't appeared pushed and is of a lighter colour. In Fig 4.6, on the '*Save image*' line, 'Whole' is selected while 'Subset' is not.

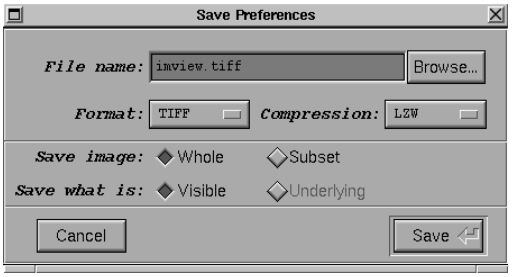


Figure 4.6: The `Save Preferences` dialog.

The check buttons function in groups: only one of these buttons can be selected in a given line at a given time. For example, on the '*Save image*' line, if 'Whole' is pressed, 'Subset' gets deselected if it had been selected before.Check buttons are used to express a choice between limited alternatives.

The *File name* input field: The name of the output file can be typed in this field. Pressing the 'Browse' button pops up a dialog nealy identical to the `Open Image` dialog for selecting the output file interactively (see section 4.3.1.1).

The *Format* choice menu: Pressing this button-like menu pops up a short list of available output formats. At present, `imview` can only save images in TIFF and Z-IMAGE formats.

The *Compression* choice menu: If this menu is greyed out, this means that the selected output image format does not support compression. When available, this menu shows a list of available compresssions for the selected format and the currently displayed image.

The Z-IMAGE format does not support compression.

The TIFF format supports the following compressions:

Compression name	Meaning	Availability
None	No compression	For all images
LZW	Lempel-Zif and Welch	For all images
Packbits	Packbits	For all images
Group 3	CCITT Group 3 (fax)	Binary images only
Group 4	CCITT Group 4 (fax)	Binary images only
RLE	Run-length encoding	Binary images only
RLEW	Wide Run-length encoding	Binary images only
Thunder	Thunderscan	Binary images only

The *Save image* check button group: This group allows the user to elect to save the entire image being currently displayed, irrespective of zoom and pan (`Whole` button), or only

the portion of the image currently being shown (`Subset` button).

The *Save what is* check button group: This group allows the user to elect to save the image as it is interpreted on-screen (that is: as a binary or char 2-D image, with colourmap applied to it, etc), by selecting the `Visible` button; or to save the underlying data by selecting the `Underlying` button (that is: save the original CHAR, INT or DOUBLE image, without colourmap present, possibly 3D, etc). Obviously for some images this doesn't make any difference.

The *Cancel* button: This button makes the dialog go away, no action are taken.

The *Save* button: This button causes the saving of the current image to be undertaken with the preferences expressed by the user. If the saving is successful, the dialog is closed and the user is returned to the main application. If an error occurs, an error dialog pops up and the `Save Preferences` dialog stays up in order to let the user take corrective action.

4.3.1.3 The `Close` item

Can be invoked with `alt+c`

This item is available only if an image is currently being displayed. It will close the image presently being displayed, and will delete it from the image list (see section 4.3.4). The next image on the list will be displayed instead. If no images are left to be displayed, the splash screen (see Fig 4.2) is shown and this menu is disabled.

4.3.1.4 The `Print` item

Can be invoked with `control+P`

This item is available only if an image is currently being displayed. If selected, the `Print Preferences` modal dialog pops up (see Fig 4.7).

See section 4.3.1.2 for a quick reminder of what 'check buttons' are.

The *Print to* check button group: One can either print to an actual printer or to a file. If 'Printer' is selected, then the *Print command* input field is available. If 'File' is selected, then the *File name* input field becomes available.

The *Print command* input field: Type in this field the command that is necessary to print on the current system. `Imview` makes a reasonable guess based on the user's environment. The result of the print process will be piped to this command.

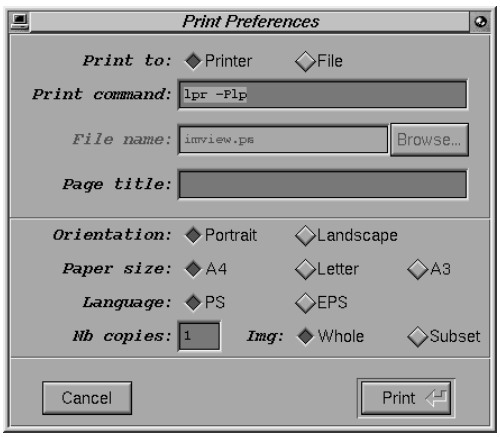


Figure 4.7: The `Print Preferences` dialog.

This input field is available only if the 'Printer' check button is selected. Otherwise this input field appears in a lighter shade of grey and cannot be edited.

The *File name* input field: Type the output file name in this field. The 'Browse' button permits the user to specify an output file via a dialog nearly identical to the `Open Image` dialog (see section 4.3.1.1).

This input field is available only if the 'File' check button is selected. Otherwise this input field appears in a lighter shade of grey and cannot be edited.

The *Page title* input field: A title for the printed page can be entered in this field. It will be printed at the bottom of the page underneath the image. Only one line can be entered. This input field will be unavailable if the 'EPS' check button is selected, because a title cannot be attached to such an output format.

The *Orientation* check button group: One can print either in portrait (upright page) or landscape (page on the side) mode. The two check buttons correspond to these alternatives.

The *Paper size* check button group: Three paper sizes are supported: a4 (European letter size), letter (U.S. letter size) and a3 (large European format).

The *Language* check button group: One can print in normal Postscript (PS) or in Encapsulated Postscript (EPS) formats. The PS format is to make a simple, standalone document. A title can be added to the bottom of the page (see above, the *Page tile* input field). The EPS format is to create a document better suited to be included in other documents (such as this documentation for example). No title can be attached to an EPS file, and they

are not meant to be sent directly to a printer, although `imview` will not prevent the user from doing so.

The *Nb copies* input field: This field allows the user to specify the number of copies to be sent to the printer. The content of this field is ignored for EPS files.

The *Img:* check button group: This group allows the user to decide whether to print the whole image currently being displayed (irrespective of zoom, pan, etc), or to only print the subset of the image that is currently shown on screen.

The `Cancel` button: Pressing this button will make the dialog go away without any action taken. If the user selects the `Print` menu item again, the button selection, file name, title, etc, will be remembered.

The `Print` button: Pressing this button will cause the user-chosen action to be undertaken. If an error occurs, an error dialog will show up, and the `Print preferences` dialog will stay around in order to let the user make corrective action. If no error occurs, the `Print preferences` dialog will simply go away.

4.3.1.5 The `Quit` item

Can be invoked with `alt+q`

Selecting this item will quit the application. There is no warning or dialog asking for confirmation!

4.3.2 The `Edit` submenu

This submenu can be invoked with `alt+e`.

This menu is mostly disabled except for the preference submenu (see Fig 4.8).

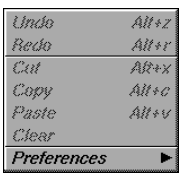


Figure 4.8: The `Edit` menu.

4.3.2.1 The `Preferences` submenu

Can be invoked with `alt+p`

A number of user preferences can be set with the help of the `Preferences` menu. At present, there is no way to save the user preferences from session to session.

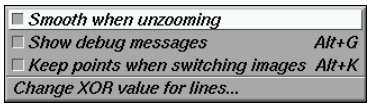


Figure 4.9: The `Preferences` submenu.

Figure 4.9 shows the appearance of the `Preferences` submenu.

Smooth when zooming out (no shortcut)

By default, images displayed after zooming in and out are not smoothed for speed reasons. Turning this option on by selecting the corresponding item causes zoomed out images to be smoothed (using a bilinear interpolation). The result is much more pleasing to the eye but takes about 3 times longer to display.

Show debug messages This item can be invoked with `alt+G`

Turning this option on will result in a large number of debugging messages to be printed. Only for developers.

Keep points when switching images This item can be invoked with `alt+K`

Normally, points and lines (see section 4.10) are deleted and saved to file when switching from one image to the next in the image list (see section 4.3.4). Turning this option on will result in the points and lines defined in an image to be kept around when switching images. This allows the user to compare points in two or more similar or related images.

Change XOR value for lines... For the record, this experimental feature allows the user to change the appearance of overlay lines. One can safely ignore this.

4.3.3 The `Transform` submenu

This submenu can be invoked with `alt+t`

This submenu is itself made of three submenus (see Fig. 4.10).

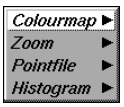


Figure 4.10: The transform submenu.

4.3.3.1 The `Colourmap` submenu

There is no shortcut for this submenu.

Fig 4.11 gives an example of a shortish colourmap submenu. This particular submenu is dynamic in the sense that it will record the colourmaps associated with new images read via the `Open` submenu (see section 4.3.1).

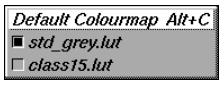


Figure 4.11: The colourmap submenu.

The colourmaps on this menu are standard Z-IMAGE colourmaps and are searched for in the present working directory, and in the directories indicated by the environment variables `$ZHOME/etc` and `$IMVIEWHOME`. The `$ZHOME` environment variable is usually defined by Z-IMAGE, so many colourmaps will appear automatically under this menu if `imview` is called from this environment. `$IMVIEWHOME` can point to a variety of colon-separated locations, as in:

```
IMVIEWHOME="/home/joe/colourmaps:/usr/local/Z-IMAGE/etc:$HOME/myluts"
```

Under Windows, `IMVIEWHOME` can also point to several location, but the separator is a comma, not a colon (because of the drive name problem, as in `C:\blabla`.

All these locations are searched in order of appearance. Duplicate colourmaps will appear duplicated in the list.

4.3.3.2 The `Zoom` submenu

There is no shortcut for this submenu.

The zoom submenu (see Fig 4.12) allows the user to zoom in or zoom out an image depending on the item selected. There are a number of shortcuts corresponding to the functions available via this submenu:

- `alt+>` zooms the image in by a factor of 100%,
- `alt+.` zooms the image in by a factor of 10%,

- `alt+<` zooms the image out by a factor of 100%,
- `alt+,` zooms the image out by a factor of 10%
- `alt+'` sets the current zoom factor as the default,
- `alt+z` calls up a modal dialog to set the current zoom factor,
- `alt+Z` calls up a modal dialog to set the default zoom factor,
- `alt+n` returns the current zoom factor to the default.

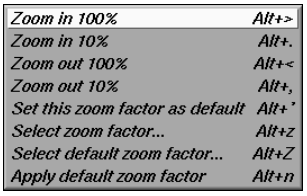


Figure 4.12: The zoom submenu.

Current and default zoom factors The *current* zoom factor is the one used to display the image currently on screen. In most situation this is 1.0, unless the image is too small to accomodate the main menu, or the user has specified a different factor for example on the command line or with the currently described submenu.

The *default* zoom factor is the one the application reverts to by default: when switching images, when clicking the right button of the mouse on the image or when typing the shortcut `alt+n`.

Both these values can be set to any strictly positive real number, however images must be larger than some minimal dimension, discussed below.

Zoom submenu items The zoom submenu allows the user to do all sort of things with the current and default zoom factors.

The first four items allow to zoom the image in and out by small and large increment. This only affects the current zoom factor. The default zoom factor remains unchanged by these operations.

In contrast, selecting the menu item `Set this zoom factor as default` does precisely this: the current zoom factor becomes the new default.

The menu item `Select zoom factor...` calls up a modal dialog box where the current zoom factor can be entered by hand, as shown in Fig. 4.13. As described above, any strictly positive real value is acceptable, but the real zoom factor will depend on the minimum image dimensions, as described below.

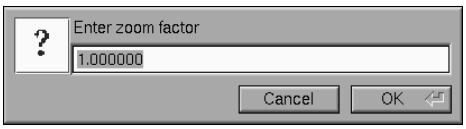


Figure 4.13: The `select zoom factor` modal dialog.

The menu item `Select default zoom factor..` calls up a nearly identical modal dialog (see Fig. 4.14), but this time the default zoom factor can be entered by hand.

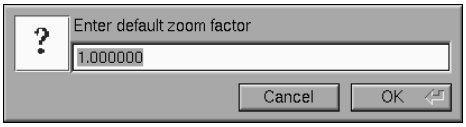


Figure 4.14: The `select default zoom factor` modal dialog.

Selecting the menu item `Apply default zoom factor` applies the default zoom factor to the current image.

Zoom factor and minimum image dimensions: Displayed images must have a minimum width of 240 pixels to accomodate most of the main menu, and a minimum arbitrary height of 10 pixels. The minimum zoom factor is computed for each image in order to achieve these minimum dimensions. Users are not allowed to zoom out an image beyond this factor. If an image is small enough in either dimensions, it might be displayed for the first time using its minimum zoom factor which in this case is necessarily greater than 1.0. In this case, the default zoom factor is the minimum zoom factor, and the user will only be able to zoom the image in.

Zooming in and out can also be achieved efficiently with the mouse. See section 4.5 for details.

4.3.3.3 The `Pointfile` submenu

Point files (also called "geom" files) are simple text files where statistics from manually selected points in the image are recorded. These statistics can then be used in turn by other external programs, for example to obtain pixelwise, spatial or angular measurements, extract subimages, etc. `imview` allows the creation and management of point files. Operations on points and pointfiles are described in detail in section 4.10.

Figure 4.15 shows the `Pointfile` submenu.

The items on this submenu cause the following actions:

- `Add point mode` (shortcut `control+g`) If this item is activated (the square dot gets filled with black on the left of the item), the mouse controls change to a different mode:

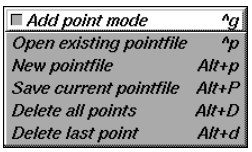


Figure 4.15: The `Pointfile` submenu.

the add point mode. In this mode, a left-click of the mouse on the image adds a point to the pointfile, and a right-click of the mouse creates a 'break' (grouping of points). The middle button can still be used for zooming, but in order to unzoom, the right button cannot be used in this mode: the shortcut `alt+n` must be used instead.

When this mode gets deactivated (select the item again, and the square dot should become empty), the mouse controls return to normal operations.

The quick, and often best way to activate the `Add point mode` is to hold down the shift key. As long as the shift key is held down, mouse clicks on the image will behave the same as in the `Add point mode`. I.e: shift-left click will add a point to the pointfile and shift-right click will add a break.

Releasing the shift key deactivates the `Add point mode`.

- `Open existing pointfile` (shortcut `control+p`) calls an open dialog (same as in section 4.3.1.1), which allows the user to select an already existing pointfile. The points defined in this files are read in, and displayed on the image. New points can be added to it.

- `New pointfile` (shortcut `alt+p`) calls an open dialog which allows the user to select a new point file name. All new points will be saved to this file.

- `Save current pointfile` (shortcut `alt+P`) allows the user to save all the currently defined points to the point file.

- `Delete all points` (shortcut `alt+D`) will delete all the points currently defined on the image.

- `Delete last point` (shortcut `alt+d`) will delete the last defined point.

4.3.3.4 The `Histogram` submenu

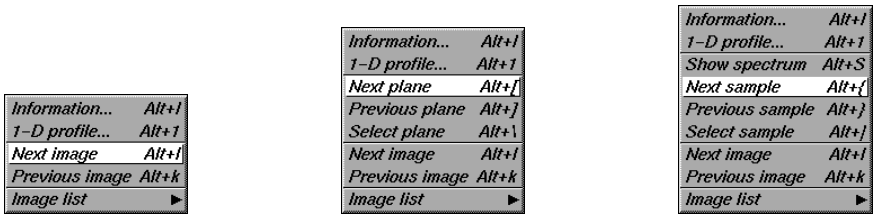
There are two planned items in this submenu, but only one is functional now: the `Contrast-brightness` item (see Fig. 4.16.

Selecting this item brings up the `Edit transfer function` panel, see section 4.11

Figure 4.16: The `Histogram` submenu.

4.3.4 The `Image` submenu

This submenu can be invoked with `alt+i`. This submenu is dynamic in the sense that its content varies depending on the characteristics of the image being displayed, as shown on Fig 4.17.



(a) (b) (c)

Figure 4.17: The image submenu for a grey-level image (a), a 3-D image (b) and a multispectral image (c).

4.3.4.1 Image information

The first item in this menu is the `Information` item. This item is always present in the menu. Selecting this item brings up an information panel (Fig. 4.18:

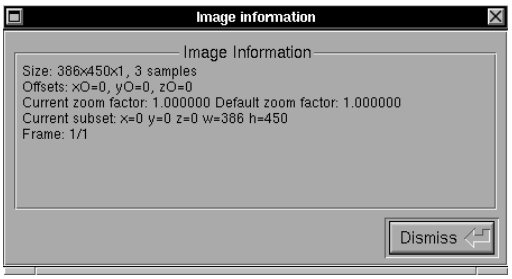


Figure 4.18: The `Image Information` panel.

This panel allows the user to obtain some useful information on the image currently being displayed.

4.3.4.2 1-D profiles

The second item, also always present in the menu, is the `1-D profile` item When selected, this item brings up the `Display Profile` panel (see Fig 4.19).

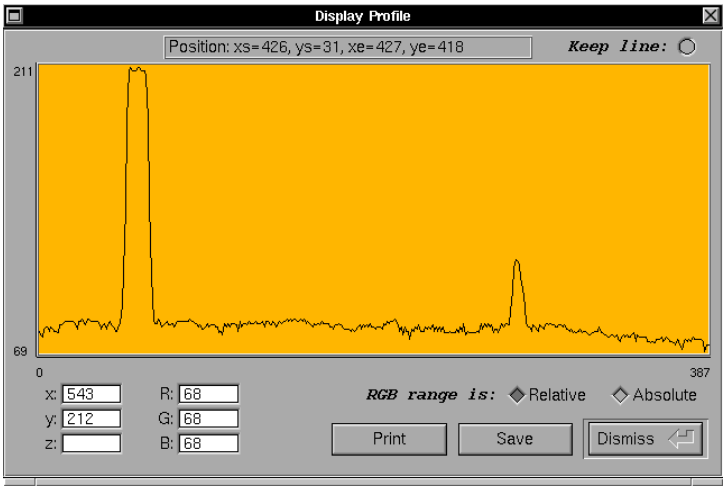


Figure 4.19: The `Display Profile` panel for a grey-level image.

For anything interesting to be displayed on this panel, a *profile* must have been defined. This can be done by holding down the `Control` button while dragging the mouse on the image. The last profile defined in this way remains visible on the `Display Profile` panel.

See section 4.8 for more informations on profiles.

4.3.4.3 Navigating the images

This submenu (see Fig 4.20(a)) allows the user to view all the images present in the image list. Each time an image is opened, either via the command line at startup or via the file menu (see section 4.3.1), it is recorded in the image list under the present submenu (see Fig 4.20(b)).

(a) (b)

Figure 4.20: The image submenu (a) and an example image list (b).

Some portions of this menu may or may not be available depending on the properties of the image currently being displayed. The multispectral operations will not be available on a single sample image, for example.

The items in this menu perform the following actions:

- Operations available for multispectral images: Single sample and 3-sample multispectral

images are rendered respectively as grey-level images and RGB images. Multi-sample images with 2 or more than 3 samples are rendered as a stack of 2-D grey-level images. Each sample can be viewed separately. The following menu items allow to navigate in the sample stack.

- `Next sample` (shortcut `alt+{`, or `Home`) This item displays the next sample.
- `Previous sample` (shortcut `alt+}, or End`) This item displays the previous sample.
- `Select sample` (shortcut `alt+|`) This item calls up a modal dialog allowing to select a sample number for display, as shown on Fig. 4.21. With the help of this

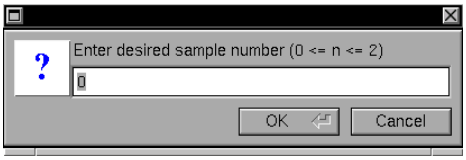


Figure 4.21: The `Select sample` modal dialog.

modal dialog, the user can enter a sample number to be displayed. The range of valid sample numbers is indicated in the message (in the example, between 0 and 3).

The window title shows, in between parentheses, the sample number currently being displayed together with the total number of samples in the multispectral image, itself in angle brackets (see Fig. 4.22).

Figure 4.22: Window title for a multispectral image. the sample being displayed is indicated next to the image name (sample number 32, image has 56 samples in total).

- Operations available for 3-D images: 3-D images are currently displayed as a stack of 2-D images. There is no volume rendering facility in `imview` for the time being, and there is no facility to flip the axes either: the z direction is always up and down the stack.
  - `Next plane` (shortcut `alt+[`, or `Insert`) This item displays the next image plane.
  - `Previous plane` (shortcut `alt+]` or `Delete`) This item displays the previous image plane.
  - `Select plane` (shortcut `alt+\`) This item calls up a modal dialog allowing to select an image plane for display, as shown on Fig 4.23.

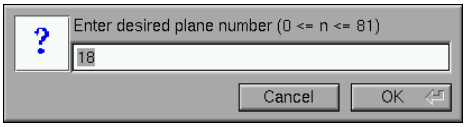


Figure 4.23: The `Select plane` modal dialog.

The window title will show, in between square brackets, the number of the plane currently being displayed together with the total number of planes that the image contains, itself in angle brackets (see Fig. 4.24).

Figure 4.24: Window title for a 3d image. The current plane number (18 in this example) and the total number of planes in the image (82 in this example) are displayed next to the image name.

- Operations available for multi-frame images: `Imview` has some limited support for multi-frame images. Examples of those include animated GIF images, and multi-page TIFF documents, such as faxes for example. Support for selecting a frame and going up and down the frame is available under the same menu. The frame number appears in the title between curly brackets, together with the total number of frames that the image contains, itself in angle brackets, as shown on Fig. 4.25.

  A shortcut for going up and down the frame stack are the `Page up` and `Page down` keyboard keys.

Figure 4.25: Window title for a multi-frame image. The current frame number (4 in this example) and the total number of frames in the image (36 in this example) are displayed next to the image name.

- Operations available for all images: All images specified on the command line or manually opened via menus appear in the image list. The following items allow to navigate in the image list:

  - `Next image` (shortcut `alt+l`) Displays the next image in the list.
  - `Previous image` (shortcut `alt+k` Displays the previous image in the list.
  - Moreover, selecting any image in the image list displays it. Closing an image makes it disappear from the image list. The next image in the list is displayed in its place.

4.3.4.4 The `Help` submenu

The `Help` submenu is there to give help! Figure 4.26 shows the appearance of this submenu.

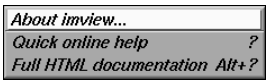


Figure 4.26: The `Help` menu.

The `Help` menu items have the following actions:

- `About imview...` (no shortcut). This item brings up a modal dialog where some information about `imview` is displayed, as shown on Fig. 4.27 The information on this

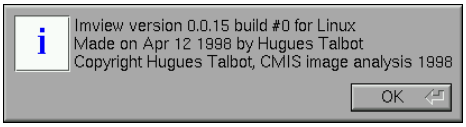


Figure 4.27: The `About imview...` modal dialog.

  dialog is similar to what is available through the `-v` Unix command line switch (see section 4.12.1).

- `Quick online help` (shortcut `?`). This item causes a short help document to be loaded. This document is actually only an image that shows up in the image list, and can be manipulated like any other image. Closing this image (see section 4.3.1.3) makes the help document go away.

- `Full HTML documentation` (shortcut `alt+?`). This item launches Netscape and points it to a Web page where an HTML version of this present document can be found. If Netscape cannot be launched, a modal dialog pops up with the address of the web page. This address can be used in any Web browser.

4.4 Obtaining pixelwise information with the mouse

To obtain concise pixelwise information at the cursor location, just press the left mouse button anywhere on the image. A status line appears at the top or at the bottom of the image, whichever is the furthest from the position of the mouse when its left button is first pressed. Figure 4.28 shows the appearance of the status line on a typical image.

Figure 4.28: A typical status line at the top of the image.

The status line goes away when the left mouse button is released.

4.4.1 Content of the status line

The information displayed in the status line depends on the characteristics of the image.

The general form of the status line is as follows

x=xx,y=yy[,z=zz],{r=rr,g=gg,b=bb|gl=gl} [(X=XX,Y=YY[,Z=ZZ]] [< TYP=t_1,t_2,t_3,... >]

Possibly more clearly, this means:

- `x` and `y` indicate the position of the cursor in the window. This information is always present.
- The triplet `r,g,b` for RGB images, or the single value `gl` for grey-level images, indicates the value of the pixel at the indicated location.
- If the image is 3-D, `z` indicates the current slice. This information is also available in the title of the main window (see Fig. 4.24).
- If the image origin is not (0,0,0), then the triplet (X=xx,Y=yy,Z=zz) shows the current pixel coordinate taking the real origin of the image into account. If the image is not 3-D, only (X=xx, Y=yy) is displayed.
- if the image is not CHAR, the type of the pixel and its real value is shown between angle brackets. The possible types are:
  - BIN: binary
  - CHR: char
  - SHT: short
  - INT: int

  - FLT: float
  - DBL: double

- If a colourmap has been applied to the image:
  - If the image is CHAR, the `r, g, b` values displayed are those of the colourmap. The index in the colourmap (or underlying grey-level) is displayed in between angled brackets following the word LUT, like this: `<LUT 125>`
  - If the image is INT, the same is true, but since INT value can exceed 255, both the colour map index and the real underlying INT value are displayed between the angled brackeds after the word (INT). The real INT value is the one displayed in between parentheses, like this: `<LUT 74 (2378)>`.

- if the image is multispectral, at most 3 sample values are displayed in order, in between angled bracket, after the type of the pixel, like so: `<FLT 3.2, 45.3, 33.2>`. If more than 3 samples are present,
  - if the current sample is one of the *first* 2, then all first three samples are shown, followed by 3 dots, as in: `<SHT 120,1024,325,...>`,
  - if the current sample is one of the *last* 2, then all last three samples are shown, preceded by 3 dots, as in: `<CHR ...,12,10,220>`
  - if the current sample is after the first 2 but before the last 2, three samples are shown, with the current sample in the middle, and 3 dots are shown before and after, as in: `<DBL ...,1.24,10.2,0.334,...>`.

  in this manner, the value of the pixel displayed under the mouse indicator is always shown.

4.4.2 Status line in `add point mode`

In the `add point mode` (see section 4.10), the status line is still active. Points are only added to the pointfile when the left mouse button is *released*. As long as this mouse button is held down, the status line is visible, in reverse video (i.e: white characters on a black background, as in Fig 4.29)

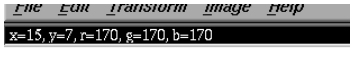


Figure 4.29: The status line in `add point mode`.

4.5 Zoom and pan with the mouse

Zoom and pan with the mouse is a central feature of `imview`. The implementation of this feature in `imview` is fast and memory-efficient.

To use the zoom feature, a 3-button mouse is needed. The middle button is used to define the region of interest that will be zoomed in, and the right button returns the zoom factor to the default value.

4.5.1 Zooming in a region of interest

To define a rectangular region of interest (ROI) within an image, click and hold the middle button where the top-left corner of the ROI is wanted. Then drag the mouse, with the middle button still pressed, to extend the selection rectangle. The outline of a rectangular area is drawn on the image. The top-left corner of this rectangular outline is fixed where the middle button was first clicked on the image. The bottom-right corner of this outline should closely follow the mouse pointer.

When the ROI so defined is what is wanted, release the middle button of the mouse. The area just defined will be zoomed in. Fig 4.30 illustrates this process.

(a) (b)

Figure 4.30: Pressing the middle button of the mouse and dragging the mouse defines a ROI. The outline of the ROI so defined is visible in (a). Releasing the middle button zooms the ROI as seen in (b).

4.5.1.1 Why the ROI does not follow the mouse exactly:

By default, the ROI is constrained to have the same aspect ratio as the main window. The bottom-left corner of the ROI will not exactly follow the mouse, but will define a ROI with the same aspect ratio as the main window that closely matches the position of the mouse.

This constraint is useful for several reasons:

- Knowing that the zoomed images will fit exactly in the existing window makes the zoom function precise and efficient: the area in the ROI will be exactly what will be displayed after the middle mouse button is released.
- Conversely, knowing that the zoomed image will most likely *not* fit in the existing window make the process slower, less efficient and ambiguous: should the window be enlarged or shrunk? should the program alter the aspect ratio? Since most of the time the user just wants to inspect part of an image from closer up, a constrained zoom window is not a problem.

4.5.1.2 Unconstraining the zoom ROI

Bearing in mind the above paragraph, it is possible to unconstrain the zoom ROI. To do so, hold the `shift` key while dragging the middle mouse button. This time the ROI follows the mouse exactly. Most likely the window will be resized after this. The rules are as follows:

- `Imview` will try to enlarge the window as much as possible, but the end result will be made to fit on screen. This means that the window can actually be shrunk in one dimension while being elarged in another, depending on the initial window geometry and the shape of the unconstrained ROI.
- The x and y dimensions of the window after zoom will not be smaller than the set minimum dimensions (see section 4.3.3.2). Combined with the above constraint, this means that if the ROI is either too tall and narrow or too short and wide, then the final window may not have the desired aspect ratio after zoom.

4.5.1.3 Information about the Zoom ROI

The zoom ROI can be used to interactively subset, or crop, the image. For precision, the information panel (see section 4.3.4.1) displays the dimensions and offset in the image of the ROI instead of the dimension of the image, while a ROI is being defined.

4.5.2 Zooming out

Clicking the right button of mouse on the image reestablishes the default zoom factor.

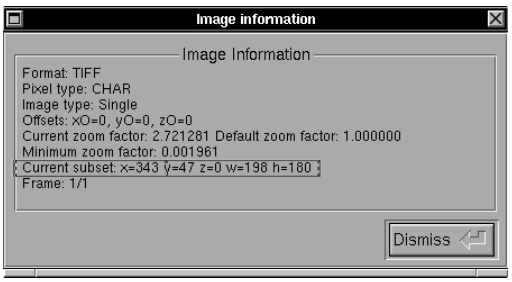


Figure 4.31: The `current subset` line in the image info panel shows the offset and size of the zoom ROI while it is being defined.

4.5.3 Panning with the mouse

An image that is too large to display entirely in the on-screen window can be panned using the scrollbars present on the left of and below the image.

There is nothing mysterious about their usage...

4.5.4 Resizing the window

A window containing an image can be resized using the familiar window manager controls.

There are constraints on such windows, however: the minimum width of an image window is currently 200 pixels (to accomodate the width of the application menu), and its minimum height is 10 pixels below the menu. Image window absolute maximum width and height depend on the dimension of the screen it is being displayed on. The true window maximum width and height depend on the content of the window. Images smaller than the screen dimensions will constrain the window dimensions to their own.

4.6 Tool bar

A minimalistic tool bar has been added to `imview`. Type `Shift+t` to show it or use the menu `Transform->Toolbar`, see Fig. 4.32.

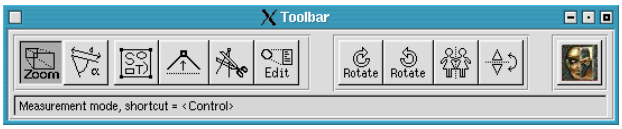
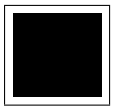


Figure 4.32: The fairly minimal toolbar.

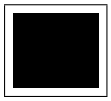
Because of the limitations of the FLTK GUI toolkit, this toolbar cannot be docked, and the status bar replaces tooltips (a.k.a. balloon help). Here is the meaning of all the buttons:

4.6.1 Point manipulation modes

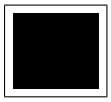
Pressing any of these button on the toolbar will set `imview` into a specific mode which changes the mouse buttons behaviour.



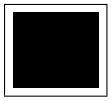
Default mode (zoom). In this mode the left mouse buttons makes pixelwise measurements, the middle button selects a region of interest for zooming and the right button unzooms.



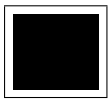
Measure mode. In this mode the left button measures angles and distances. The middle and right button function as in the default mode. Keeping the `Control` key pressed is a shortcut for this mode.



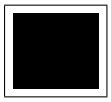
Select point mode. The left mouse button draws a region of interest around points, that become selected (they change colour). They can then be mass-deleted for example. There is currently no shortcut for this mode.



Add point mode. The left mouse button adds points. You must still use shift-right click to add a break (to group points). Keeping the `Shift` key pressed is a shortcut for this mode

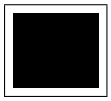


Remove point mode. The left mouse button removes existing points. When the mouse hovers close to a point the cursor changes to a delete item cursor. Keeping the `Shift+Alt` keys pressed is a shortcut for this mode.

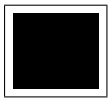


Edit point mode. When the mouse comes close to a point, it changes shape to the text edit caret. Clicking on the point brings up the annotate panel (see section. 4.6.3). There is currently no shortcut for this mode.

4.6.2 Transform modes



Rotate the image 90°counterclockwise.



Rotate the image 90°clockwise.

4.6.3 Point annotation

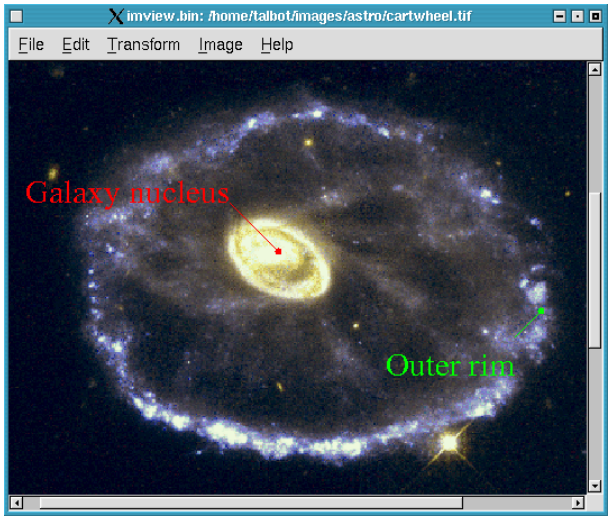


Figure 4.33: Example of annotation (in red and green).

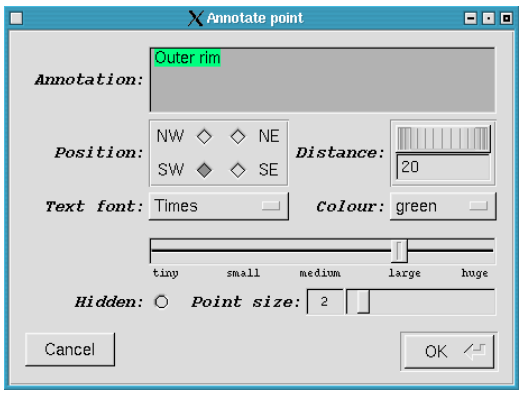


Figure 4.34: Point annotate panel.

4.7 Colour maps

Currently, colour maps can only be applied to single-sample CHAR images. Use the `Transform->colourmap` submenu to apply a colourmap to an image being currently displayed interactively (see section 4.3.3 and Fig 4.11).

Once a colourmap has been assigned to an image, it will stay assigned to it and will be automatically reapplied to the image for the duration of the session with `imview` each time it is redisplayed. To assign the default colourmap to the image being displayed[1], select the "Default colourmap" item in the Colourmap submenu. A shortcut for this is `alt+C`.

The list of colourmaps present on the menu depend on the content of the `$IMVIEWHOME` and `$ZHOME` environment variable as explained in section 4.3.3. The `$ZHOME` environment variable should not really be used, as it is related to Z-IMAGE and not so much to `imview`.

Z-IMAGE format images can also have their own colourmap, which is just a file with the same base name as the image, but with the `.lut` extension instead of `_z`, present in the same directory as the image. `Imview` handles this case in the following way: the default colourmap of these images is their own. Different colourmaps than their own can be assigned to such images in the usual way. For these images, selecting the "Default Colourmap" item will make `imview` revert to the image's own colourmap (if a standard grey colourmap is desired for such an image, it can generally be found in the colourmap list).

[1] which usually means "no colourmap", i.e: the normal greyscale colourmap, unless a different default colourmap has been specified on the command line, or in the case of images with their own colourmap

4.8 Profiles, distance and angular measurements

Arbitrary 1-D profiles of 2-D images can be viewed and obtained with `imview`. The same process allows users to make simple distance and angular measurements on 2-D images with the mouse.

To create a profile or start a measurement, hold down the `Control` key while clicking the mouse anywhere on the image. Then drag the mouse with or without the `Control` key down.

To see the content of a profile, the `Display profile` panel must be visible. To do so, select the `Image->1D Profile` menu item. Fig 4.35 shows the result of distance and angle measurements, and the corresponding profile.

(a) (b)

Figure 4.35: Measuring distance and angles (a). Corresponding profile (b).

Figure 4.36 shows an example of profile of a colour image, grey-level images profiles are shown in black only, as in Fig. 4.19.

On this panel,

- The plot title displays the start and end point coordinates of the profile in the image,
- The `Keep line` check button next to the plot title governs the behaviour of the profile line displayed on the image. If the button is checked (red), then the profile line does not disappear when the mouse button is released, but instead remains visible on the image (and can be zoomed in, etc). If the button is unchecked (grey), then the profile line does not remain visible when the mouse button is released. Any previous visible profile line is erased. By default the check button is unchecked.
- The plotting area itself shows the profile. If the image being displayed is grey-level, the profile is shown in black. If the image is RGB, 3 plots are shown at once, which

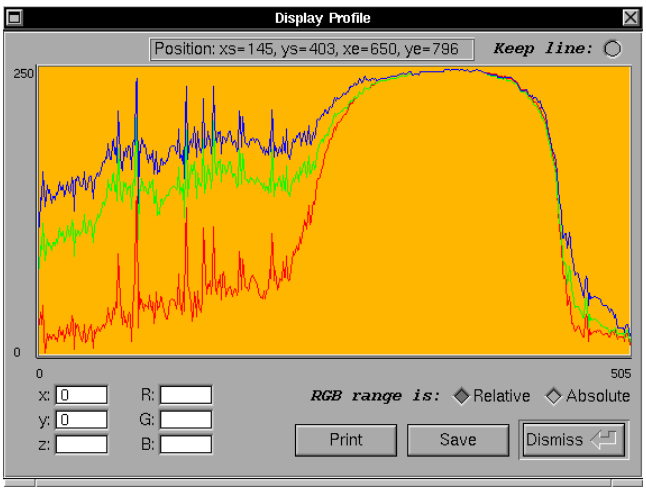


Figure 4.36: The `Display Profile` panel for a colour image.

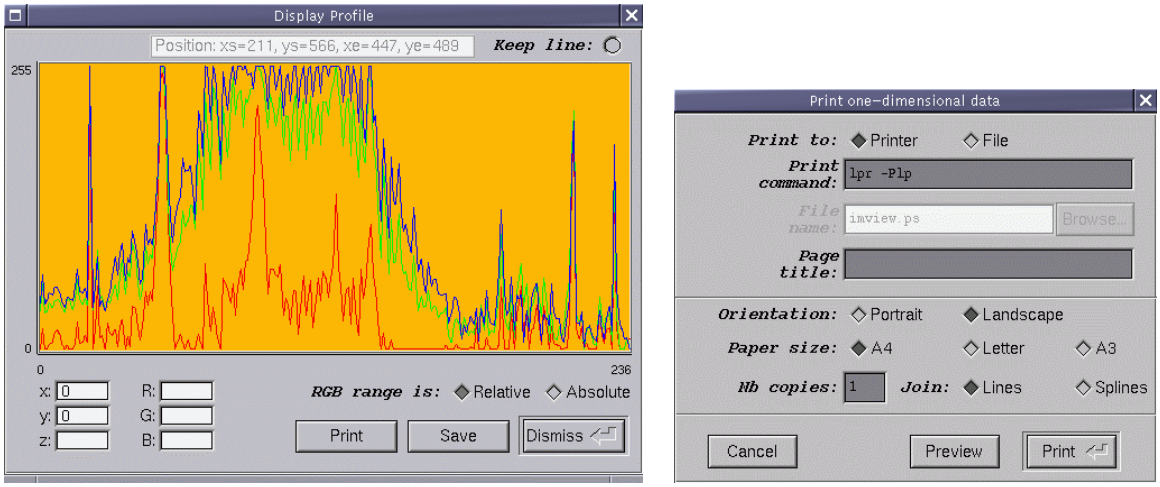
correspond to the 3 red, green and blue samples. The colour allocations are obvious (Red plot to the red sample, etc).

- The `x`, `y`, `z` output boxes display the coordinates in the original image of a point along the profile line. Click and drag the left button mouse anywhere on the plotting area for anything to be displayed in these boxes.
- The `R`, `G`, `B` output boxes display the corresponding RGB values in the original image of any point along the profile line. Click and drag the left button mouse anywhere on the plotting area for anything to be displayed in these boxes. If the image is RGB then all three boxes can be expected to display different values. If the image is grey-level, then all three boxes will show the same value.
- The `RGB range is` check button group can be set to either `Relative` or `Absolute`. If it is set to `Relative`, then the profile is stretched to fit in the vertical space. Numbers on the left of the graph indicate the range. If it set to `Absolute`, the range is set from 0 to 255, and the profile is plotted within that range. The default is `Relative`.
- The Print button calls up a dialog that allows to print the profile.
- The Save button calls up a dialog that allows to save the profile.

As with all similar `imview` dialogs, profiles can be printed or saved.

4.8.1 Printing a profile

To print an existing profile, press the `Print` button on the profile dialog. Figure 4.37(a) shows a sample profile dialog. Figure 4.37(b) shows the corresponding `Print profile` dialog. This is exactly the same as in section 4.9.



(a) (b)

Figure 4.37: A sample profile and the print profile dialog.

In the `Print profile` dialog (the title of the window is actually `Print one-dimensional data` as it is shared between spectra and profiles), the input fields and choice buttons are very similar to the `Print preference` dialog of section 4.3.1.4. Refer to this section for details. The only notable difference is the `Join` choice. It can be set to either `Lines` or `Splines`. In spectra or profiles with little data, choosing `Splines` causes a small degree of smoothing. When a lot of data points are present, this choice has little effect.

Previewing a profile If `Preview` is pressed instead of `Print`, then a preview panel is shown instead, as in Fig. 4.38.

Press the `Preview` button again in the `Print profile` panel to close the preview window. While the `Preview` panel is showing, the `Preview` button stays depressed. If the `Preview` panel is closed using the windows manager-supplied buttons, the `Preview` button still needs to be pressed a second time before a new preview can occur.

This behaviour is necessary due to the way GNUPlot functions.

NOTE: The software GNUPlot `must` be installed on the system for the print or preview features to work.

4.8.2 Saving a profile

To save a profile, press the `Save` button on the profile dialog. Figure 4.39 shows a sample `save profile` dialog. The dialog window's title is in fact `Save one-dimensional data` as it is shared between profiles and spectra.

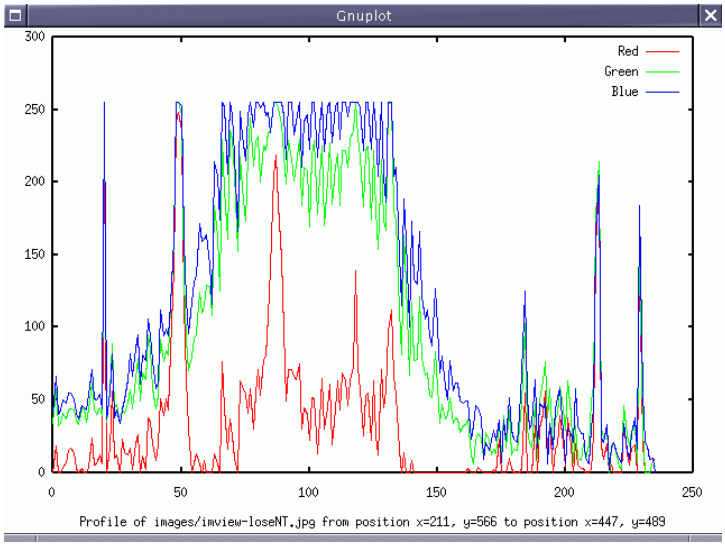


Figure 4.38: Profile preview before printing.

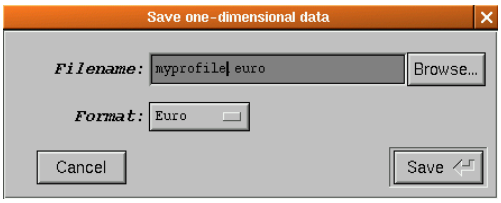


Figure 4.39: Saving a profile.

Profiles can be saved in TIFF, Z-IMAGE or Euro format. TIFF and Z-IMAGE are binary image data (basically the profile is saved as a one-dimensional image with a single row). The Euro format is basically text and is very simple, here's an example:

```
238 3
        39         9        66        77        36
        59        57        32        20        31
        15        28        16        23        32
        27        20        31        63        37
        26        61        23        41        51
        22        27        13        34        38
        41        63        43        47        50
        41        40        65        41        37
...
```

In this example, the first line indicates that we have a series of 238×3 values, The red components values are printed first, then the green values, then the blue values. There are 5 values per line.

4.9 Working with multi-spectral images

`Imview` now has some basic, but very useful facilities for working with multi– and hyperspectral images. Users can now display the spectrum at each location and interactively segment colour and multispectral images based on bivariate histograms.

4.9.1 Displaying the spectrum at each point

If an hyperspectral image is being displayed (meaning an image with at least 3 samples per pixels), a new panel becomes available under `Image->Show Spectrum`, see Fig. 4.40

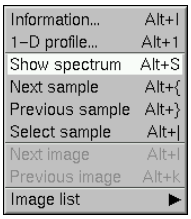


Figure 4.40: The menu item for the `Display spectra` panel.

If this item is selected, the `Display spectra` panel is displayed, as in Fig. 4.41.

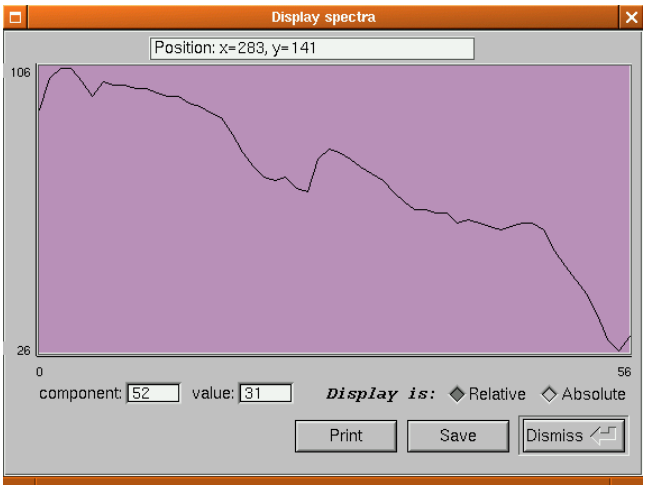


Figure 4.41: The `Display spectra` panel itself.

This panel has the following characteristics:

- In this example, the corresponding image has 57 samples per pixel (that makes this image hyperspectral alright...), The spectrum at location $(283, 141)$ is displayed.
- The numbers on the ordinate axis (y axis) show the dynamic range (from 26 to 106 in this case). Presently the range is represented as a CHAR, and so the maximum range

is between 0 and 255, even if the input image is not CHAR (a DOUBLE image for example).

- The number on the absissa axis (x axis) represent the spectrum index, i.e: 0 is the first spectrum and 56 is the last one.
- The two output fields show the values along the spectrum when the user clicks and drag the mouse on the spectrum view.
- The `Display is Relative/Absolute` choice button change the way the spectrum is displayed. If `Relative` is selected, the range of the spectrum is fitted to the height of the spectrum view (to compare shapes between spectra); otherwise (when `Absolute` is selected), the spectrum is displayed in a vertical range from 0 to 255, for absolute value comparison.
- An individual spectrum can be printed, previewed and saved. The `Print` and `Save` panels are exactly the same as in section 4.8.1 and section 4.8.2. Please refer to these for details.

At the moment only one spectrum can be displayed at a given time.

4.9.2 Multispectral interactive segmentation

It is often useful to represent the values of each pixel in a multispectral image by plotting one spectrum against another. The result is called a bivariate histogram. As with a single-variable histogram, it is useful to perform interactive thresholding on such representations.

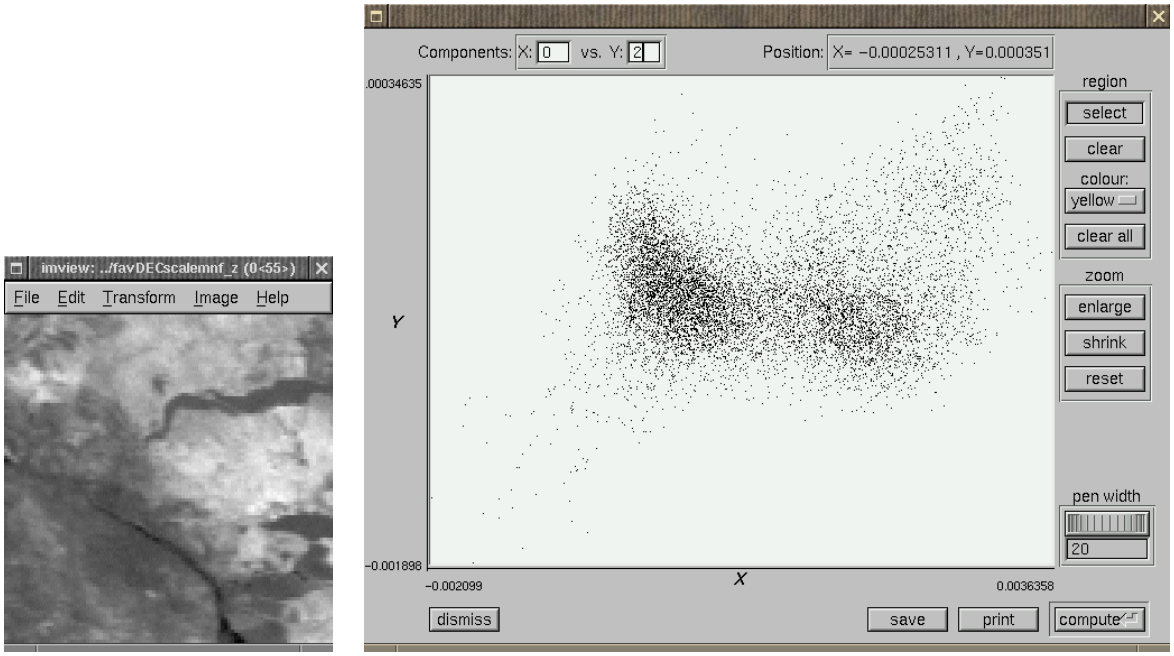
In the case of the bivariate histogram, boundaries (or thresholds) must be provided in 2D. There is no reason why boundaries should be restricted to vertical or horizontal lines. In fact arbitrary closed regions can be defined as "thresholds". This process is called "segmentation". A special panel has been designed to perform this action.

Figure 4.42: The `Bivariate histogram` menu item.

If the image being displayed has more than one spectrum, the menu item `Transform->Histogram->Bivariate Histogram` becomes available (see Fig. 4.42). If the user selects it, the `Bivariate histogram` panel becomes visible, as shown on Fig. 4.43.

The characteristics of this panel are as follows:

- The X and Y components (spectra) are entered in the top left input fields. Spectra start at 0.



(a) (b)

Figure 4.43: A sample multispectral image with a bivariate histogram. The first spectrum is plotted against the third (0 vs. 2).

- The `Position` indicates where the mouse is when the user moves the mouse over the bivariate histogram view (no need to click). The format reflects the input data. In the example the pixel type is DOUBLE.
- The white background area is the bivariate histogram view. It is initially empty. To compute the histogram, press the `Compute` button at the bottom right of the panel. The content of the view reflects what is *displayed* in the main `imview` window. If for example only a subset of the image is shown in the main window, the bivariate histogram is computed on this subset *only*.
- The X and Y axes show the range of the data. This is currently not updated when zooming on the data.
- The Region group of button works in the following way:
  - If the `Select` button is depressed, clicking and dragging the mouse in the bivariate histogram view turns histogram and corresponding image points in the selected colour (or "paints" them in this colour). This is how interactive segmentation is achieved. The size of the "brush" is given by the `Pen width` input field.
  - If the `Clear` button is depressed, the colour is cleared. One and only one of `Select` or `Clear` can be depressed at any one time

  - The colour can be chosen from a small list.
  - the `Clear all` button does nothing at present. To clear an image and a bivariate histogram from all colours, reload the image and recompute the bivariate histogram.

- The `zoom` group of button works relatively intuitively. If `Enlarge` is pressed, the view area remains the same in size but the data is spread across an area twice as big in both directions. Scrollbars are then provided to move around the data. `Shrink` does the opposite: it shrinks the data by 50% in both directions. If the data fits in the window, scrollbars disappear. `Reset` fits the data spread to the view. This is useful to press after a resize of the main window for example.

- The `Pen width` combination input field/rotating knob allows the user to change the size of the "brush" when painting points of the bivariate histogram. By default it is 1, and can vary up to 40. To change the value, click and drag the mouse on the knob or on the input field. Drag to the right to increase and to the left to decrease.

  You may notice that with a large value, an ellipse is painted rather than a disk of expected diameter. This reflects the difference in range between the X and the Y axis. In the data space, a disk is painted with 1:1 ratio.

- The `Save` and `Print` buttons don't work yet.

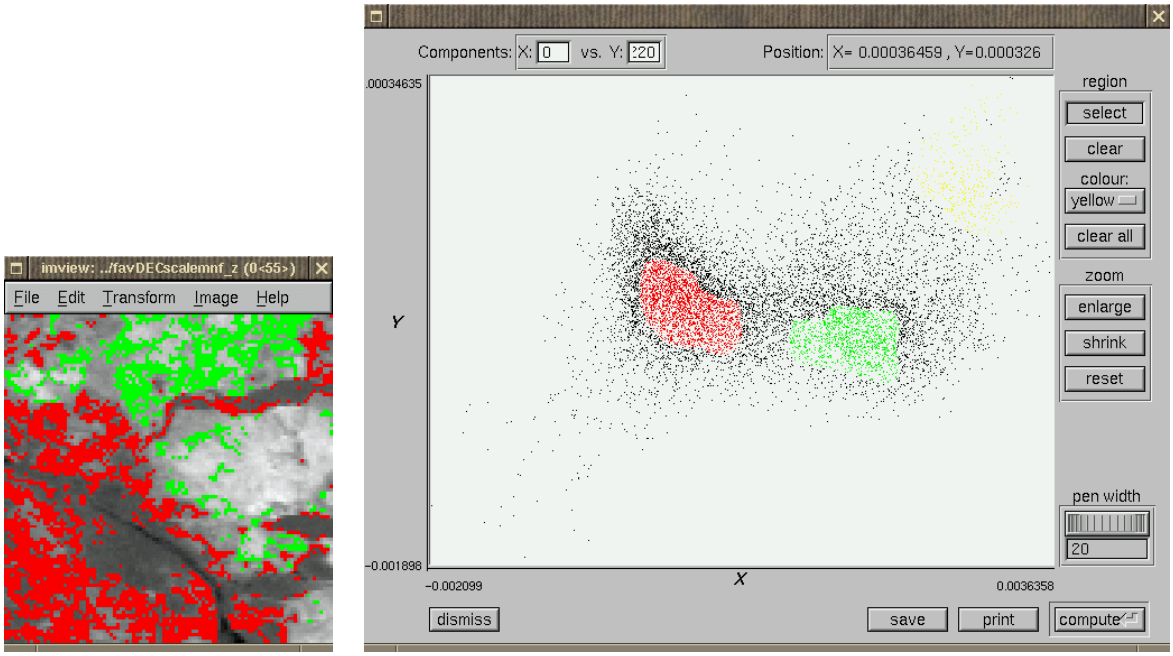
- The `Compute` button allows to re-compute the bivariate histogram at any time. At present, the only way to clear a bivariate histogram is to reload the image in the main window (thereby discarding all the overlayed colours), and to press `Compute`.

The effect of "painting" a portion of the bivariate histogram is immediately reflected on the image displayed in the main window of `imview`. The corresponding pixels are also painted with the same colour, as illustrated in Fig 4.44.

NOTE: It is important to note that the image used to build the bivariate histogram (the one that is displayed when pressing the `Compute` button) need not be the same as the one on which the colours are overlayed. They must both have the same X and Y dimensions, but this is the only restriction.

This way, one can compute the principal component analysis (PCA), say, of a given image, load both the original and the PCA images (they must both have the same dimensions) into `imview`, compute the bivariate histogram on the PCA image, display the original image in the main window, and conduct the histogram segmentation *then*.

This way the segmentation will be overlayed on the original image, not the PCA one.



(a) (b)

Figure 4.44: Segmented bivariate histogram and corresponding segmentation on the image.

4.10 Pointfiles

`imview` allows the creation of *pointfiles*[2], which are simple text files where pixelwise information is recorded. These files can then be used by other programs or applications.

4.10.1 Opening a point file

There are several ways by which a point file might be opened.

4.10.1.1 Default point file

By default, if nothing else is done, the point file is called `pointfile` and is located in the current directory.

4.10.1.2 Command-line option

At the Unix prompt, a different file name for the pointfile can be specified with the `-p` option, as in:

[2]a.k.a. *geom* files after the `x11()` menu they are created from

```
% imview -p myptfile myimage.tiff
```

At the Z-IMAGE prompt, the point file name can be specified with the `file:` option, as in:

```
Zimage-5.0.16> imview(myimage, file:"myptfile")
```

4.10.1.3 Menu

A point file name can be specified using the menus under `Transforms->Pointfiles`. Selecting either of the `Open existing pointfile` or `New pointfile` menu items will cause an open file dialog to appear, where a file name can be specified. Section 4.10.5 explains the difference between the two actions (briefly: the former will display the existing points in an existing point file, allowing to append more points to this file, while the latter will start from scratch on a new point file).

4.10.1.4 Note

Note that the point file is never opened, created or written to by default. The user has to perform actions for any of this to occur. This has the advantage that using `imview` doesn't require write permission in the current directory (an infamous problem with `x11()`) and that point files are never overwritten by default either. On the other hand, users have to perform the action of saving the pointfile when they want to use it.

4.10.2 Adding points to pointfiles

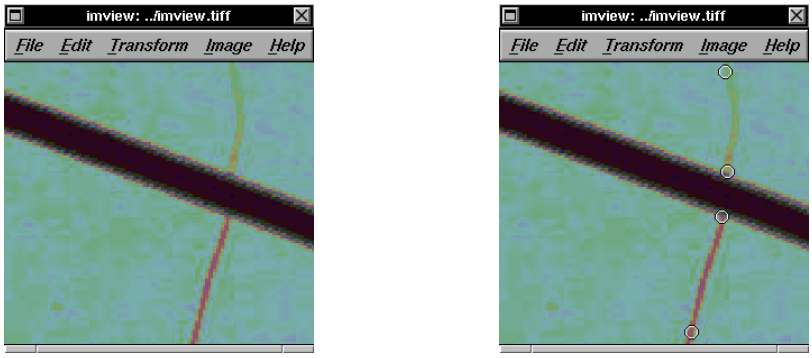
To add points to a pointfile, the `add point mode` must be enabled. There are two ways to do this:

- The first way is to activate the menu item `Transform->Pointfile->Add point mode` by selecting it.
- The second way is simply to hold down the `shift` key.

While in the `add point mode`, clicking anywhere on an image with the left button of the mouse adds a point to the current pointfile. The pixelwise information under the mouse (i.e: the position of the point in the 2D or 3D image, its RGB or Grey-level or multispectral information, etc) is recorded *in memory* when the mouse button is *released*.

While the left mouse button is held down, an information box similar to the one presented in section 4.4, but this with inverted video contrast, is put up on the image (see Fig. 4.29. In this box it is possible to find out the coordinates and pixel information at the location of the mouse. The mouse cursor is changed to a cross.

When the mouse button is released, a circle appears on the image around the location of the recorded point, as illustrated on Fig 4.45



(a) (b)

Figure 4.45: Adding points to an image: (a) original image, (b) four points added.

If the `add point file mode` was activated by holding down the `Shift` key, releasing this key *before* releasing the mouse button causes no point to be recorded.

Leaving the `add point mode` can be achieved by simply releasing the `shift` key, or by selecting the `Transform->Pointfile->Add point mode` menu item again.

4.10.2.1 Grouping points together

Points can be logically grouped together by inserting 'breaks'[3].

Inserting a 'break' can only be done in `add point mode`, and is achieved by clicking the right mouse button in this mode (i.e: do a shift-right click, or select the `Transform->Pointfile->Add point mode` menu item and then do a right click). The points defined since the last break (or since the beginning of the file if no breaks has been defined so far) will be logically grouped together. This grouping is illustrated on screen by a line which joins the corresponding points, as shown in Fig. 4.46

One can define as many point groups as needed in a given image. If the `add point file mode` was activated by holding down the `Shift` key, releasing this key *before* releasing the mouse button causes no grouping to be recorded.

The line that joins a group of points is drawn from the earliest defined point to the latest. No line is drawn if a group consists of less than two points.

[3]This is the old `x11()` terminology, and comes from the fact that the text 'Break' appears in the saved pointfile to indicate grouping.

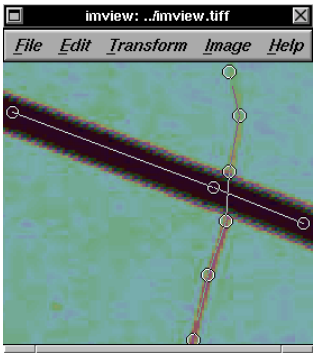


Figure 4.46: Grouping points together. On this image, two separated groups of points were defined.

4.10.3 Deleting points or groups from the pointfile

If a mistake has been made, the last point or the last break that was entered can be deleted by either selecting the menu item `Transforms->Pointfiles->delete last point`, or by using the shortcut `alt+d`. Breaks or points are deleted in the reverse order in which they were entered.

All the entered points and groups can be deleted at once by selecting `Transforms->Pointfiles->Delete all points` or by using the shortcut `alt+D`.

Finally any single point among all those present on the image can be deleted by activating the `add point group` mode (with the menu item or by holding down the `Shift` key), holding down the `Alt` keys, and clicking on or near the point with the left button mouse. The point will be deleted only when the left button mouse is *released*. If both the `Shift` and `Alt` keys are released before releasing the mouse button, no point gets deleted. If the mouse button is released far enough from any point (a few pixels is enough), no point gets deleted either. It is not possible to delete groupings (breaks) with this method.

If a point is deleted in the middle of a group (this is only possible with the last method), the line that illustrates the grouping is redrawn to reflect that fact.

4.10.4 Saving the pointfile

All the actions described in the previous paragraphs only work in memory. No file gets written to until a saving action takes place.

To save the pointfile under the current point file filename (as defined in section 4.10.1), invoke the menu item `Transforms->Pointfile->Save current Pointfile`, or use the shortcut `alt+P`.

If no point is defined, no file gets written (so existing point files don't get overwritten).

The point file gets automatically saved when quitting the application or switching images (to a new image in the image list), including when a new image is opened.

The point file is *not* saved if the application is interrupted (by Control-C or Control-Z or if it is killed). This is possibly a feature.

4.10.5 Appending to an existing pointfile

It is possible to read in the content of an existing point file and have the points defined therein displayed on the screen on top of the image. More points can then be added to the collection already present.

To do this, several options are possible:

4.10.5.1 From the command line

Under Unix, the `-a` option will cause the current pointfile to be read (either the default one or the one specified with the `-p` option). The points and groups defined in it that can fit in the image are then displayed.

Under Z-IMAGE, the `append:"T"` option will do the same thing with either the default point file or the one defined with the `file:` option.

4.10.5.2 Using the menu

Selecting `Transforms->Pointfile->Open existing pointfile` (or using the `Control-p` shortcut) will cause an Open file dialog to appear. If a valid point file is then selected with that dialog, the points contained therein will be displayed on the screen.

4.10.6 Point file format

There are two point file formats, unfortunately.

4.10.6.1 The old point file format

This point file format is identical to the one produced by the `x11()` application, it is extremely simple:

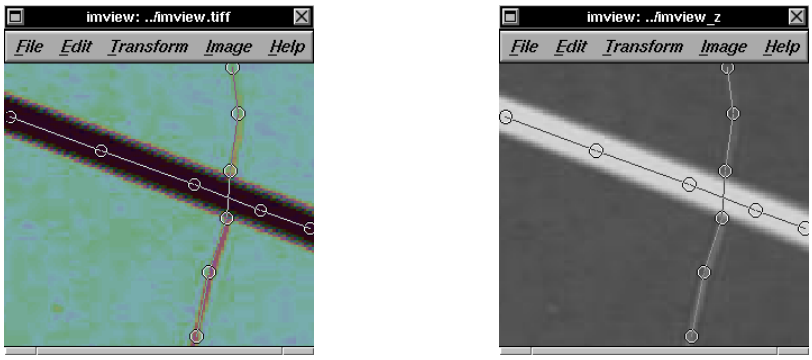
```
123 342 25
231 433 34
break
121 23 128
```

Each line corresponds to a point. The first number is the `y` coordinate of the point (or the row), the second number is the `x` coordinate (the column) and the third number is the grey-level value of the corresponding pixel.

The `x` and `y` coordinates take the image offset into account (i.e: if the origin of the image is not (0,0)). The grey-level value is just that: a value between 0 and 255. The 'break' on a line by itself indicates that the points above it are grouped together.

4.10.6.2 Sticky points

Normally points are saved and deleted when switching from one image to the next in the image list (see section 4.3.4). However, by selecting the menu item `Edit->Preferences->Keep points when switching images`, or `alt+K`, points stay around, allowing to compare point locations precisely when switching images, as illustrated in Fig. 4.47



(a) (b)

Figure 4.47: Keeping points when switching images: (a) two points groups on first image, (b) same points and groups on second image.

The values attached to the points are those corresponding to the image the points were defined in.

4.10.6.3 The new point file format

Obviously, the old point file format does not support `INT` or `DOUBLE` values, RGB data, multispectral data, and 3-D images. The new image format is as follows:

```
# pointfile for ../crni39.jpg
#   Y   X    R    Z   G   B     x    y    z   n LBL  ...
  196 313  213    0 255 255   196  313    0   3 CHR  213  255  255
  205 358  228    0 255 255   205  358    0   3 CHR  228  255  255
  239 361  228    0 240 255   239  361    0   3 CHR  228  240  255
break
  263 347  228    0 252 255   263  347    0   3 CHR  228  252  255
  294 449  234    0 255 255   294  449    0   3 CHR  234  255  255
break
```

The first two lines are comment lines. The first line indicates which image this pointfile corresponds to, the second line is a key to the point values.

Each line starting from the third corresponds to a point just as in the old format. The first three values (Y, X, and R) are equivalent to the fields of the old format, namely:

- `Y` is the `y` coordinate taking the image offset into account,
- `X` is the `x` coordinate taking the image offset into account,
- `R` is the value of the red channel at this point. If the image is grey-level this corresponds to the luminance.

The remaining values are the following:

- `Z` is the `z` coordinate taking the image offset into account,
- `G` is the value of the green channel at this point,
- `B` is the value of the blue channed at this point,
- `x`, `y`, and `z` are the coordinates of the point, *not* taking into account the image offset (as if the image origin was (0,0,0)).
- `n` indicates the number of multispectral samples to follow. This number can be different from 1 and 3, and corresponds to the actual number of samples at the present pixel position in the image.
- `LBL` is a 3 letter label. It can be:
  - `BIN` indicating a BINARY image,
  - `CHR` indicating a CHAR image,
  - `SHT` indicating a SHORT image,
  - `INT` indicating an INT image,
  - `FLT` indicating a FLOAT image,

  - `DBL` indicating a DOUBLE image,
  - `LUT` indicating that the pixel saved in the point file comes from an image with a colour map. The number to follow is the look-up table entry corresponding to the RGB colour of the point.

- There will be as many remaining values as indicated by the `n` entry. Each will have the type indicated by the `LBL` entry and correspond to the real multispectral values in the original image at the indicated coordinates.

Point grouping is indicated by the lines with a 'break' by itself, just as in the old format.

4.10.7 Point files and special conditions

4.10.7.1 3-D images

Points defined on 3-D images work as expected, i.e: points defined in a given `z` plane do not appear on any other.

4.10.7.2 Zooming in and out on points

Visible circles around points have a minimum size of 5 pixels in diameter, and a maximum size smaller than the smaller dimension of the display window, thus points defined in the area of the image being viewed, even at extreme magnifications, should always be visible.

It is of course possible to add and delete points at any zoom factor. A high magnification is recommended for precision.

4.10.8 Point file annotation

Point files can also be annotated. For this

4.11 Contrast, brightness and gamma control

`Imview` allow to edit the *transfer function*, which is a fancy way of saying that it is possible to change the contrast, brightness and gamma of an image interactively.

4.11.1 Grey-level transfer panel

The simplest panel works on all displayed channels at once. You can invoke this panel via the `Transform->Histogram->Contrast-Brightness` menu item. Selecting this item brings up the `Edit transfer function` panel (see Fig 4.48.

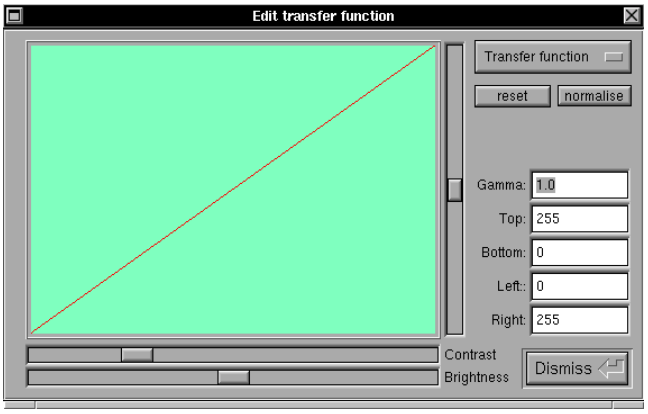


Figure 4.48: The transfer panel.

In this panel, at the left is the transfer function. The sliders at the bottom control the brightness and contrast, which are reflected in the offset and slope of the transfer function. The upright slider controls the gamma between the extremities of the tranfer function. The gamma value can also be entered in the corresponding input at the right of the panel.

The Top, Bottom, Left and Right values are the coordinates of the points where the transfer function intersects the box.

The reset button resets the transfer function to the default transfer function, which normally is the identity, but can be different if a `-gamma` has been specified on the command line. The normalize button does the equivalent of a 'fitchar' on the image, that is: it sets the transfer function so that the darkest pixel gets the value 0 while the brightest pixel gets the value 255.

One can display the histogram by selecting this in the choice menu at the top right of the panel. The appearance of the panel changes to that of Fig 4.49.

4.11.2 Notes

While interacting with the sliders in the transfer panel, the effect of the change in contrast, brightness and gamma is immediately reflected in the transfer function, but only gets reflected

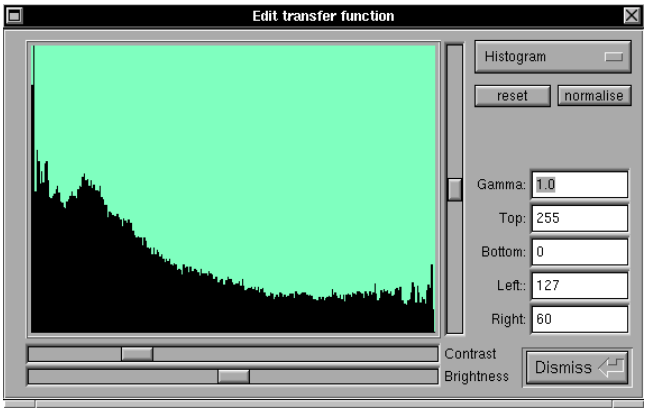


Figure 4.49: The histogram panel.

on the image when the user *releases* the slider button.

It is possible to perform interactive thresholding by pushing the contrast button to either extremity and using the brightness slider to select the threshold, which appears in both the top and bottom output fields.

This panel's functionality has been superseded by the RGB transfer panel. In future releases this panel might be disabled.

4.11.3 The RGB transfer panel

As of release 0.9.8 a new RGB panel has become available. It allows to modify brightness, contrast and gamma on all channels at once or separately. The appearance of the new panel is very similar to the previous one, as shown in Fig. 4.50

This panel has basically the same functionality as the panel of section 4.11.1, except it works on all 3 colour channel separately.

Colour channels can be linked together by clicking on the "link" radio button right of the contrast/brightness sliders. Linked channels will function as one. If the user links all three channels, the functionality of this panel is basically the same as for the grey-level transfer panel.

For some applications it might be useful to link only two of the channels. For example in some situation two grey-level images are being matched. It can be useful to place the first image in the red channel and the second in the combined green+blue channel. Obviously linking only the blue and green channels together makes sense in this situation.

At the moment the row of three "Affected" toggle button has no effect, this is for future extension.

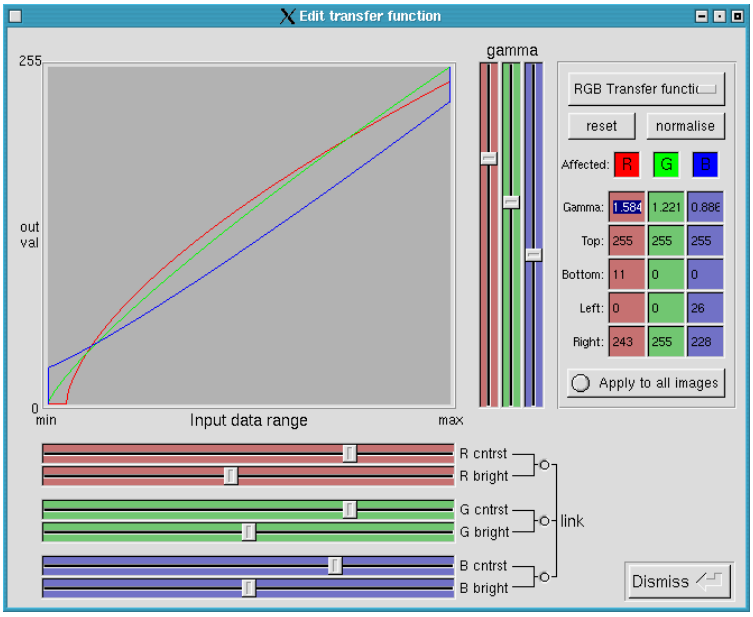


Figure 4.50: The RGB transfer panel.

4.12 Command-line interface reference

As outlined in section 4.2, `imview` can both be used at the Unix command line or with the Z-IMAGE interface.

4.12.1 Unix command line interface

The usage for Unix is as follows:

```
 Purpose   :  this application displays images
 Usage     :  imview [-h] [-v] [-debug] [-stopdebug] [-h] [-v] [-C defaultLUT]
           [-p pointfile] [-a] [-] [-mag defaultMagnification]
           [-wt <sometitle>] [-delete]
           [image1 [-c clut1]] [image2 [-c clut2]] ...
Imview-specific options:
------------------------
              -v:  prints version number
              -h:  this help
          -debug:  prints gobs of debugging messages
      -stopdebug:  not only that, but stops at each message
  -gamma <value>:  specifies <value> as the default gamma
  -p <pointfile>:  specifies a point file name
              -a:  reads and appends to an existing pointfile
               -:  reads from the standard input
    -mag <value>:  changes the default magnification to <value>
 -C <defaultlut>:  changes the default lut for all images
     -wt <title>:  sets the window title to <title>
         -delete:  deletes images after use
           image:  TIFF, GIF, JPEG or Z-IMAGE image
    -c <lutname>:  Z-IMAGE type look-up table attached to preceding image
Other standard X11 options:
---------------------------
-display host:n.n
-geometry WxH+X+Y
-title windowtitle
-name classname
-iconic
-fg color
-bg color
-bg2 color
```

4.12.1.1 Main usage

Images `image1`, `image2`, etc (all these being in a supported image file format) can be displayed by simply typing `imview image1 image2 ...`. Normal Unix special characters are of course accepted (`*`, `?`, etc), as in `imview *.jpg *_z mypics[0-9].tiff`.

4.12.1.2 The `-v` and the `-h` options

The `-v` option shows the version number (and some extra information such as the build number, build date and build author).

The `-h` option returns the usage shown above.

4.12.1.3 The `-debug` and `-stopdebug` options

You don't want to use this one unless you are developping for `imview`. `-debug` prints debugging messages on the console, and makes imview event handling synchronous with the X server. `-stopdebug` does all that and stops for user input after each debugging messages for even finer control.

4.12.1.4 The `-gamma` option

This options allows the user to specify a default gamma setting to all images. The default gamma is 1.0. Any strictly positive value is acceptable, but the useful range is between 0.3 and 3.0 (approximately). This value is the one that is reverted to when the `reset` button is pressed in the histogram control panel (see section 4.11).

4.12.1.5 Pointfile control with the `-p` and `-a` options

These options allows the user to specify a filename where to save point statistics (position and multispectral values under the cursor).

The `-a` option tells the application to read in a poinfile, display the existing points defined in this file, and to append all new points to this pointfile. The pointfile by default is called 'pointfile', and can be changed with the `-p` option, as in:

```
% imview mona.tiff -p mona.pt -a
```

If the `-a` option is not used, the values in the poinfile will not be read in, and will be overwritten when new points are defined.

4.12.1.6 Reading files via the standard input: the `-` option

it is possible to specify an image to be read in via the standard input using the `-` option. This is most useful when using imview after a filter. For example, the following command line reads a Sun raster image file, even though `imview` does not understand this format natively[4]:

```
% rasttopnm earth.ras | imview -
```

Note: The `-` option works by creating a temporary file that is deleted after use. If your temporary storage (`/tmp` by default) is not very big, you may experience problems when loading images in this fashion. You can alleviate this by setting the standard `$TMPDIR` environment variable, which should point to a place with sufficient free space. For example:

[4]If ImageMagick has been compiled in, `imview` does in fact supports the Sun Raster format.

```
% setenv TMPDIR /usr/tmp # for csh or tcsh
% TMPDIR=/usr/tmp; export TMPDIR # for sh, ksh, bash
```

4.12.1.7 Deleting images after use with `-delete`

When using the `-delete` option, any image loaded in `imview` will be deleted when `imview` terminates. Be careful when using this option.

It can be useful in shell scripts or embedded applications that create temporary files as the output. In the case of multiple files, the `-` is not sufficient because only one file can be sent to `imview` via the standard input.

4.12.1.8 Changing the windows running title with `-wt`

The title of the application can be changed from `imview` to anything using the `-wt` option, followed by the desired name.

With this option, the information that is normally displayed alongside the application name, such as the file name of the image currently on screen is still displayed.

This can be useful in shell scripts or embedded applications for example.

Note: The `-wt` option is not the same as the standard X11 `-title` option. The latter changes the *whole* title; this title doesn't get remembered if a new image is loaded in, for example. The former remplaces `imview` in the title with the given one; loading a new image will not overwrite this.

4.12.1.9 Colourmap control with `-c` and `-C`

Basically, the `-c` option followed by a colourmap name (with or without the `.lut` extension) allows the user to attribute a colourmap to the preceding image on the command line, as in

```
% imview mona.tiff -c heat
```

The `-C` option followed by a colourmap name (with or without the `.lut` extension) changes the default colourmap for all the images on the list to this colourmap, except for those for which a `-c` option has been specified, as in

```
% imview mona.tiff lisa.jpg -C heat lena_z -c blue
```

With this command line, "mona" and "lisa" will use the "heat" colourmap, while "lena" will use the "blue" colourmap.

Colourmaps are searched in the directories specified with the `$ZHOME` and `$IMVIEWHOME` environment variables.

4.12.1.10 Standard X11 options

Nothing unusual here, except `-bg1`, `-bg2` and `fg`, which allow the user to change the colour of the background, highlight colour and font colour respectively. (grey, white and black are the respective defaults).

4.12.2 Z-IMAGE command line interface

The usage for Z-IMAGE is as follows:

```
Purpose   :  displays images under X11
Usage     :  imview( <image1>,[cmap:"std_grey"],<image2>,[cmap:"std_grey"],...
          ,[CMAP:"std_grey"],[file:"pointfile"],[append:"F/T"]
          ,[mag:1.0], [xmax:1.0], [ymag:1.0], [gamma:1.0],[debug:"T/Y"])
<image1>, <image2>, ...  is a list of arbitrary images
cmap:"some_LUT"          applies a colour LUT only to preceding image
                         on command line
CMAP:"some_LUT"          applies a colour LUT to all images on the command
                         line, except for those with a cmap:  argument
file:"pointfile"         names the default file where points
                         location and values can be saved
append:"F/T"             appends to an existing pointfile
mag/xmag/ymag:<xx.xx>    changes the default magnification to <xx.xx>
gamma:<x.xx>             changes the default gamma value
debug:"T/Y"              turns on some debugging messages
```

Basically, nothing special here: the syntax is slightly different, but the functionality is exactly the same as in the Unix command line case.

This syntax follows the `x11()` syntax, except that with `x11()` one could only specify one image on the command line for display.

4.13 Menu item shortcuts reference

Main shortcut	Alternative	Meaning
	R	Resets the display (default zoom, maximises window)
	m	Maximizes the window
	r	Redraws the image (at the current zoom and pan)
`alt+c`	c	Closes the current image
`alt+shift+c`		Apply the default colourmap
`alt+d`		deletes the last entered group or point
`alt+shift+d`		delete all the points and groups
`alt+e`	e	Selects the `Edit` submenu
`alt+f`	f	Selects the `File` submenu
`control+g`		Toggle the `add point mode`
`alt+shift+g`		Turns the debugging messages on
`alt+h`	h	Selects the `Help` submenu
`alt+i`	i	Selects the `Image` submenu
`alt+k`	k or <spacebar>	Displays the next image in the list
`alt+shift+k`		Keeps the points when switching images
`alt+l`	i or Shift-<spacebar>	Displays the previous image in the list
`alt+n`	n	Returns to the default zoom factor
`alt+o`	o	Opens a new image
`alt+p`		Opens a new point file
`alt+shift+p`	p	saves the point file
`control+p`		append to a pointfile
`control+shift+p`		Prints the image
`alt+q`	q, Esc	Quits the application
`alt+z`		To enter the current zoom factor by hand
`alt+shift+z`		To enter the default zoom factor by hand
`alt+.`	.	Zooms in by 10%
`alt+,`	,	Zooms out by 10%
`alt+'`		Declare the present zoom factor as the default
`alt+>`	>	Zooms in by 100%
`alt+<`	<	Zooms out by 100%
`alt+[`	Insert, [	Displays the next image plane (3D)
`alt+]`	Delete,]	Displays the previous image plane (3D)
`alt+|`		Allows the user to select an image plane manually
`alt+{`	Home	Go to the next image sample
`alt+}`	End	Go to the previous image sample

`alt+\`		Selects the sample by hand
	PageUp	Displays the next frame
	PageDown	Displays the previous frame
`?`		Displays the quick online help
`alt+?`		Invoke netscape on the HTML version of this document
...	...	to be completed ...

Chapter 5

The `imview` server

5.1 Introduction

`Imview` can now function as an image server. The concept is a little bit like an FTP server where one can upload images. Uploaded images are then displayed automatically. The `imview` server also accepts commands that, for example, allow users to impose a look-up-table on an image, move the main window or any of the dialogs, obtain point data, etc. The rationale behind this, the way it works and the implementation are discussed in this chapter.

5.2 Caveat – security

Please note that the `imview` server is **not** secure, that it may create a **gaping security hole** in your system, should you choose to use it. If you indeed choose to run the `imview` server, you do so at your own risk. `Imview` is provided with **NO** warranty, expressed or implied. Do not blame anybody but yourself if you experience loss of data, disclosure of precious corporate secretes, etc.

If you still want to investigate the `imview` server and even use it, be aware that unless your machine is firewalled from the Internet somehow, anybody on the web who knows your username can kill your `imview` server, upload nasty images to them, and possibly obtain a shell somehow by exploiting a bug in the code none of the Imview developers know about. For all I know the latter is in fact quite likely, but I'm not a cracker[1].

Should `imview` ever become popular, this very important issue will be addressed in a future release. Indeed, if you are interested you can contribute. In fact I would really welcome help on this issue!

Note that the `imview` server is not started by default. The non-server side of `imview` should

[1] Or even a proper Hacker.

be as secure as any other X or Windows application. Again, no warranty of any kind is provided.

5.3 Rationale and benefits

So if you are still with me, why did I start this crazy idea? There is more than one reason:

5.3.1 Embedding `imview`

`Imview` has not been designed at all to be embedded. It does not consist of well designed, separate widgets that would be easy to integrate in other applications. Instead `imview` uses a coupled class system that would probably be a prime example of how not to design . This would be nice but I didn't know enough OOP when I started. `Imview` is a rather monolithic application, at its core. If I started all over again making `imview` more modular would be a goal of mine, but I'm afraid this will not happen anytime soon.

So how can other applications use `imview`? Typically an image analysis application, under the Unix paradigm, would be a separated executable, probably build around an interpreter (examples I know of use LISP, S/Splus, R, Perl, Python or specialized interpreters such as Z-IMAGE or the Micromorph language), that uses anonymous pipes or files to transfer an image to one or more viewers. `Imview` fits this paradigm very well. You can tell imview to load a file by a simple `system()` call or pipe an image in whatever format into it. This works beautifully if the image analysis environment is file-based, such as Khoros, i.e: all the images are on disk files. Any image analysis 'function' (really a separate executable) reads images from files (in TIFF format, say), operates on them and write one or more files as output. In this context `imview` is just another image analysis function that understands those files.

However, many image analysis interpreters are not file-based. They operate on objects in main memory (RAM), therefore displaying an image involves writing an object from main memory into a file and then telling `imview` somehow to read that file to display the image. This is at best inefficient and time-consuming. If the objective is to study image motion, stereo imaging or hyperspectral data, one might be in for a bit of waiting around...

The solution to this might be to use a form of IPC (inter-process communication) to exchange data from an image analysis system to `imview`. The challenge is then to choose which form of IPC is suitable, portable, efficient, etc, and of course to implement it.

5.3.2 Remote-controlling `imview`

An alternative solution to embedding `imview` might then be to provide a server onto which applications can connect, and that allows them to do anything they want with a running

instance of `imview`. For example uploading an image, telling it to load a LUT, etc. To do this, each instance of `imview` must provide:

1. its own server,
2. a command interpreter,
3. a command protocol, and
4. an image upload protocol.

At this stage, this is all. Image download (from a running `imview`) might be a possibility in the future, as well as other, as yet unspecified features. To provide these services from a GUI application, they must run in a separate thread.

5.3.3 Sockets

Arguably, TCP/IP sockets offer the best server solution. One could also use named pipes, unix-domain sockets or various IPC messaging methods, but the TCP/IP sockets are portable, cross-platform, well understood, generally well optimized and powerful. The only drawbacks are security (opening up one more port) and relative scarcity of port number. There are only 32k or so ports available for use and each instance of an `imview` server running on a machine consumes two (one for commands and one for data). Well-known and not-so-well-known services usually run on these machines and must not be interfered with. At the moment `imview` uses the range 7600-7699, which means only 50 imviews can be opened at once on a given machine. This is only adequate if there are few simultaneous users.

Otherwise sockets are certainly fast enough for the command protocol, and if one wants to upload images to a remote instance of `imview`, on a windows machine for example (where X is not generally available), there is no faster alternative.

Image upload using sockets on the local machine certainly works, but is no faster than using files (in general). For image communication between an application and one or more instances of `imview` on the same machine, an alternative is preferred, such as shared memory.

5.3.4 Shared memory

Shared memory is the fastest form of IPC [1]. one can in theory transfer data from an application to another for the cost of a `memcpy`. In practice some handshake mechanism (using semaphores, say) is usually necessary. `Imview` uses the Unix System V shared memory IPC and associated semaphores. Using this mechanism, data transfers at the full main memory bus bandwidth is possible and has been observed in practice.

The Unix SYSV IPC is nice but has some drawbacks, in particular it is pretty hard sometimes to clean up after oneself, and the command-line level tools are a bit unusual (`ipcs`, `ipcrm`, etc), yet must often be used to do the cleanup by hand that the applications hasn't done (because of crashes, interruptions, errors, all pretty common in a development environment).

A better API for IPC is the POSIX one, but it is not implemented everywhere. At present it is implemented on Compaq Tru64 and Solaris, but only works on the former. This form of IPC is expected to work for Linux 2.6 and above.

5.3.5 Benefits of this effort

With all of these facilities implemented, it is now relatively easy to do the following:

- Communicate easily between an image application and one or more instances of `imview`.
- Instead of having a new instance of `imview` pop up each time one wants to view an image, they can all appear in one, or a small number of `imview` windows. This is great for monitoring progress on a long processing run, for doing demos, for debugging an image analysis procedure, etc.
- When working at home or remotely from a long distance over the internet (say), it is possible to operate the image analysis environment (if text-based) remotely and setup the display on the local machine using `imview`. The image data is transfered only once when a display command is issued. Once displayed the image can be manipulated locally without further network transfer. Try doing this through a remote X11 connection!. It can also happen if the local and remote platforms are different. Typically the image analysis application will run on some large Unix server and the display might occur at home on a PC running windows. This works fine.
- Have a regression test of imview. Commands can be stored in a file and sent to a running `imview`. Great for testing (otherwise, how can you test a GUI?).
- Have several linked instances of `imview`. Zoom and pan happen in the same way on all instances: great for before/after filtering comparisons. Points defined in an instance can show up in another, etc.
- With shared memory, full motion video transfer rates are possible between the client application and an instance of `imview`.

For me this was just too great to pass up. Anyway, enough rambling, back to hard and cold data.

Figure 5.1: The main window title contains the image server port number information in square brackets (here 7600).

5.4 Imview in server mode

This section assumes a basic knowledge of TCP/IP.

5.4.1 Starting up

Currently the imview server can only be started as a command-line option when first running `imview`. It cannot be started from the GUI after `imview` itself has started. The server-related command-line options are the following:

`-fork`	Sends imview to background with foreground task synchronization.
`-portfile <file>`	Writes the port number to given file.
`-server`	Starts the server.

The necessity for the `-fork` and `-portfile` options stem from the fact that the port number that `imview` uses is not fixed (a user is likely to use more than one `imview` at a time).

Typically a proper way to start the server from within another application would use all three options. If `-fork` is not used then the port number cannot reliably be known by the time control is returned to the command line. Note that this option also works under Windows (where the default command line does not support a simple way to send a task to the background). If `-portfile` is not used then the port number is echoed to the command line.

However starting the server from the command line or a file browser interactively does not require either `-fork` or `-portfile`, as the port number will show up in the main window title in square brackets, as show in Fig. 5.1.

5.4.2 Port numbers

The port number of the `imview` server needs to be known so that clients can connect to it. As mentionned previously, several `imview` servers can be running at once on the same machine, and so must provide different port numbers. This situation is unusual – normally TCP/IP server will be advertised at a single, well-known port number, e.g. 80 for a web server, 21 for telnet, etc – but by no means unsupported by TCP/IP itself.

The compiled-in range for `imview` servers is 7600-7699 inclusive. At the time of writing that range did not seem taken by any registered application, but that may have changed. Each

`imview` instance will consume two port numbers, the first channel is for commands and the second for data, as with FTP, so up to 50 imviews can run on the same machine under ideal conditions. At the moment there is no easy way to specify a different port range but by re-compiling.

The server will start to secure the lower port numbers first. If a port number is already occupied it will move up until it either finds a free one or gives up if the upper port number limit is reached. Note that under most operating systems, there is a timeout after a server closes down. Its port number is not recycled for other servers to use until after a few minutes. This means that if somehow a single imview server is started and stopped in rapid succession it can still run out of port numbers quickly.

5.4.3 Telnet to `imview`

Once an `imview` server is up and running a simple and easy way to test it is to `telnet` to it, e.g:

```
% telnet <host> 76xx
```

Where `<host>` is the machine where `imview` is running (often `localhost`) and `76xx` is the appropriate port number. If `telnet` does not fail immediately, one is then connected to the `imview` server. Login is very "informal" at present:

```
% telnet localhost 7602
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
user talbot
Welcome, talbot /tmp/fileHsYkHD 000 OK
```

where the bits in bold face (`user talbot`) is what the user typed. Of course one needs to replace `talbot` by one's login name (or username). The username must correspond to that of the user who started the `imview` process.

The `/tmp/fileHsYkHD` return argument points to a temporary file which is used for IPC synchronization.

The reply `000 OK` means the command completed successfully.

NOTE Again: at present there is *NO* security. No password, no encryption, nothing! To log onto any `imview` server, a hacker only needs the username of whoever started that `imview`. Almost certainly there is an as-yet unknown buffer overflow attack or something similar from this point on, and the hacker can probably get a shell. At the very least DoS attacks are trivial.

5.5 Available commands

Once logged onto `imview` in this fashion, the command set can be explored. The documented subset of the available commands can be listed by typing `help`. That list is fairly long so we don't reproduce it here. In the following we document these commands.

5.5.1 Help

`help/?`	prints the list of commands and arguments

5.5.2 Connection management

`user <name>`	Log in command. `<Name>` can have whitespaces in it.
`quit/bye/end/exit`	Any of these four commands logs the user off.
`kill`	Terminates the `imview` server.
`link <addr> <port>`	Links to another `imview`. At present zooms and drags will be linked.
`unlk <addr> <port>`	Unlinks from another imview.

For

5.5.3 Image uploading

`put <many args>`	Upload an image via TCP sockets.
`putm <many args>`	Upload an image via SysV shared memory.
`putp <many args>`	Upload an image via POSIX shared memory.

the "many args" are, in order:

- `"image name"`, a string,
- number of components, integer,
- for each component:
  - X dimension, integer,
  - Y dimension, integer,
  - Z dimension, integer,
  - X origin, integer,
  - Y origin, integer,
  - Z origin, integer,
  - Image type, integer,

- Pixel type, integer,
- Samples per pixel, integer.

At present only spectrum-type images (where all the dimensions, origin, image and pixel types are the same) are supported. This means the number of components must be 1. There is no limit on the number of samples per pixel.

The `putm` and `putp` don't return a response immediately. Instead shared memory and semaphores are set up for communications. A success or error message is returned at the end of the data exchange.

The `put` command returns a short and an unsigned int in text form. The first value is the binary port for data transmission. The second is a magic number that must be put out in the transmission as the first 4 bytes. This magic number is needed to match the data transmission to the data header (the arguments to the `put` command). The endianness of the data transmission does not matter, but must match that of the magic number.

For more details, a sample client implementation is provided ; see section 5.8.

5.5.4 Pointfiles

`pf`	List content of pointfile. Retuns `Empty` if so.
`stpf <name>`	Sets the point file name.
`ldpf <name>`	Read a point file in.
`svpf <name>`	Saves the points to given file.
`clpf`	Clears (empties) the point file.

See section 4.10 on pointfiles for more explanations.

5.5.5 Image management

`load <file>`	load the image from `<file>`
`cmap [name]`	apply colour map to current image, default if no name
`zoom box <x> <y> <w> <h>`	zoom to specified box
`zoom factor <zf>`	specify zoom factor (real number)
`zoom default <zf>`	specify default zoom factor
`pan <x> <y>`	pan to specified location, keeping zoom intact
`reset`	reset the image
`close [name]`	close an image (name optional)
`close <overlay>`	close the overlay plane

The `load` command loads an image from an existing file.

The name of the colour look-up table must be the complete name (no path needed).

The difference between `zoom factor` and `zoom default` is that both change the current zoom factor to the given value, but only `zoom default` makes that value the new default (unzooming with the right mouse button reverts to the default zoom).

The `<x>` and `<y>` coordinates for the `pan` command are the coordinate of the top-left corner of the main window from the top-left corner of the image. This only makes sense in a zoomed in image.

The command `close <overlay>` is a bit unfortunate. Whereas `<x>` means any value suitable for x, in this case `<overlay>` must be send verbatim.

5.5.6 Window management

`move <x> <x>`	move main window to new position
`size <W> <H>`	change the window dimensions
`raise`	raise/deiconize the main window to the top
`icon/hide`	iconize the application
`show|hide`	show or hide the main window
`... print`	show or hide the print image dialog
`... save`	show or hide the save image dialog
`... info`	show or hide the info image dialog
`... transfer`	show or hide the edit transfer dialog
`... spectra`	show or hide the spectra display dialog
`... profile`	show or hide the profile display dialog
`... user`	show or hide the user preference dialog
`... bivhist`	show or hide the bivariate histogram dialog
`... progress`	show or hide the image download progress dialog
`... help`	show or hide the online help dialog
`... toolbar`	show or hide the toolbar

These commands move, resize, icon and de-icon the main window, or show/hide any of the application's dialogs.

5.5.7 Variables

`set <var> <value>`	set some variable
`get <variable>`	obtain a variable value
`list`	lists all the valid variables

Currently all variables are defined at compile time. The only available variables are the following:

connectionList	list of active connections (string)
lutWraparound	toggle whether LUT wrap around or not (bool)
stretchtofit	toggle whether displayed images fit the available window or not (bool)
syncDisplay	toggle display synchronization (bool)
useOffscreenBuff	toggle the use of offscreen buffers for quicker redraws (bool)

The variables are case-insensitive as is the rest of the language.

5.6 The command protocol

The command interpreter embedded in `imview` is extremely basic. It only accepts pre-defined commands followed by an arbitrary number of arguments, and does not constitute a proper language[2].

5.6.1 Command syntax

The syntax is pretty basic. Each command must be send as a single line. Separators are arbitrary whitespaces. The first word is the command. The remaining words are the argument. An argument with whitespaces in it can be enclosed in double quotes (`"`). Commands and arguments are case-insensitive.

Each command must be followed by a TCP/IP line terminator (Carriage return followed by Linefeed, i.e: "`\r\n`"). The available commands were detailed in the previous section. Commands are executed immediately.

In case of success, each command returns `000 OK`, or a 3 digit error code that starts with 5, followed by a short messages, e.g: `500 Command failed`.

5.6.2 Error codes

The available error codes are the following:

```
500 Command failed
501 Unknown command
502 Unchanged
503 Not moved
504 Not sized
505 Load failed
506 Arghh!
```

[2]Certainly not Turing-complete, however the parser only takes 30 lines of C++

```
507 Not logged in
508 No data
509 Incorrect image type
510 Incorrect pixel type
511 Incomplete argument list
512 No label
513 Invalid command (
514 No image present
515 Not implemented yet
516 Unknown dialog box
517 Connection failed
518 Socket exception
519 Non-existent variable
520 Pointfile failure
000 OK
```

These are detailed below:

`500 Command failed:` This error is returned after almost every error, i.e. most errors will return *two* error messages, the first proper error message which explained what happened, and the second which confirms that the command failed. This simplifies the design of a client to some extent, but that behaviour will probably change in a future release.

`501 Unknown command:` This error is returned if the first word of a command is not recognized as a proper command. This error is however `NOT` followed by `500 Command failed`. This is the one exception at present.

`502 Unchanged:` Response to the `cmap` command: Colourmap was unchanged.

5.7 The data transfer protocol

5.7.1 Using sockets

5.7.2 Using shared memory

5.8 A simple client program: imclient

5.9 A more complex sample client application: Voir

5.10 Contributing

5.10.1 Adding a new command

5.10.2 Ideas for enhancement

- A compressed data transfer protocol for slow connections (modems).
- More security:
  - restricting to the local host,
  - password,
  - encryption,
  - tcp-wrapper-like security (connection list, etc).
- integrating some client code in `imview`: towards the linked windows idea.

Bibliography

[1] W. R. Stevens. *Unix Network Programming*, volume 2, Inter Process Communications. Prentice Hall, 2nd edition, 1998.

Chapter 6

Wish list - TODO list

Here's a vaguely prioritized list of things to do:

6.1 Easy stuff (so I think...)

- Enhance the overlay function (loading/saving an overlay would be really useful!) *update: done for the loading.*
- Enhance the spectrum display function (it only display the CHARed value in y, and a conversion vector in X would be great).
- Read ErMapper images. *Update:* This is turning to be more work than expected, because I got into other things, and now I want an arbitrary reader for uncompressed images!
- Read colour ICS images (grey-level OK right now).
- Allows to change the image aspect ratio (with mouse, keyboard entry or command line option).

6.2 Ambitious projects

- arbitrary 2D rotates.
- Bivariate histograms. *Update:* Has progressed a lot!
- Work in a reasonable way with 3D images (rendering, arbitrary 3D rotations, not just slices...). *New plan: just do arbitrary 90 degree rotations.*
- Colourmaps for multispectral images (should be simple...). There is the problem of the 3-component images, which are always interpreted as RGB... *update: done.*

- Improve interactive histogram display and manipulation for grey-level images
- Show 2D images as 3D surface (communicate with openGL).
- Same as above for multispectral images (tough).
- Handle time series of images better, a bit like xanim. *Update:* lots of work on this with shared memory.
- What can be done with stereo? see Changming.
- Periodic images ?.
- Run-time interpreter (to load, save and manipulate images via command line or scripts). *Update: mostly done.*
- Communication to-from `imview` by pipe or socket (make it an image display server). *Update: DONE! major feature now.*
- Have more functions available on the command line. *in progress.*
- Improve shortcuts and general user interface *Some progress there.*
- Comprehensive on-line help *in progress here too. Infrastructure is in place.*

6.3 Code issues

Any volunteers?

- Convert the lot to templates! (I mean, at least the different types stuff). *in progress, won't do until most compilers are ISO C++ compliant.*
- The command line argument parsing stuff is horrible! Use Stewart's library?
- Devise some testing routines, now that we have an interpreter (beef that up,too).

Chapter 7

Known bugs – strange features

- The fit-to-char semantics is inconsistent.
- The following is puzzling: load a small image, close the image, load a larger one. Watch what happens!
- When using sticky points on more than one image, the final content of the pointfile will strongly depend on the content of the image each point was defined in. The end result can be quite weird, but I think this is a feature. Only the final list of component values (for the new format) will solely depend on the last image. (30 Apr 1998: definitely not a feature, but I won't fix it right now).
- redrawing on top of points (when deleting points from pointfiles) at high zoom factors is not quite correct (image flickers a bit and pixels are not quite aligned). Hard to fix. Addendum (7 May 1998): sometimes the redraw is completely out of place, but I haven't been able to reproduce this consistently.
- fast flicker (using alt+k and alt+l) in the image list between images of varying dimensions can result in incorrect boxing of the image in the window. When one waits until an image is fully displayed before switching to the next, there is no problem. This might actually be a serious problem if the image is big and the display slow (such as over the network).
- Sometimes points are not redrawn when the image does not have the focus. This may be a feature of fltk, but I doubt it, as the image does get redrawn. This seems to be happening more often when the image is zoomed out for some reason. (18/03/98: still not fixed, but 'r' keyboard shortcut allows to redraw the content of the image).
- The 'break' lines are often redrawn incorrectly, if the user masks/unmasks part of the image, or pulls down menus, etc. The semantics of the XOR lines are hard to master. *Fixed by not using XOR lines.*

- When rotating an image, the gamma is reapplied!. If different from 0.0, this means the image gets washed out black or white very rapidly... (23 Oct 1999: this is now fixed, but gamma is incompatible with LUTs now, more exactly, applying a gamma on an image with a LUT will disable the LUT. It can be reapplied on the image after). *Fixed.*

- Rotation and flips are not remembered. I must come up with a grammar of those rotations that allows them to be remembered. If you think about it it's not so easy. *Update:* Grammar done, partially implemented (promptly forgot about all that...).

Appendix A

Windows-specific details

A.1 Introduction

`Imview` works under windows. There are not many toolkits that allows this sort of portability while retaining speed and ease of use. Figure A.1 is a screenshot of `imview` under Windows NT[1].

`Imview` is being developed under several brands of Unix, and is only *ported* to windows. I'm not a windows expert, I can use it but I don't know much about it. I'm using strictly Unix tools to produce the Windows version of `imview`. In fact, the latest version was produced using a cross compiler in a Linux environment... You can't get further away from Microsoft development tools than that. I've also been evaluating the Cygwin tools, and I like the Linux environment better, except for debugging, which can only occur on the Windows side of things...

Of course at some point I may "graduate" to Visual C++, but I'm not in a hurry. For the foreseable future, the Windows version of `imview` is likely to lag a bit behind the Unix ones.

As a result, `imview` has a distinctive non-windows flavor. It has it's own look and feel, somewhat different from the rest of the standard windows applications. It doesn't use standard windows dialogs for file inputs, printing, etc. Rather, it looks *exactly* like its Unix counterpart. This is a feature of FLTK, the toolkit used to develop it.

Depending on your expectations, you may find this irritating (you know only windows and appreciate its standard look across the board), or refreshing (you are used to Unix and don't mind applications looking different all the time), or possibly you are used to the Unix version of `imview`, or something else entirely. If you don't like it, tough. There are other image viewing programs under windows, though none with the particular feature set of `imview`, of course.

[1]The astute reader will notice that NT is in fact running under vmware, itself under Linux. This is still Windows nonetheless.

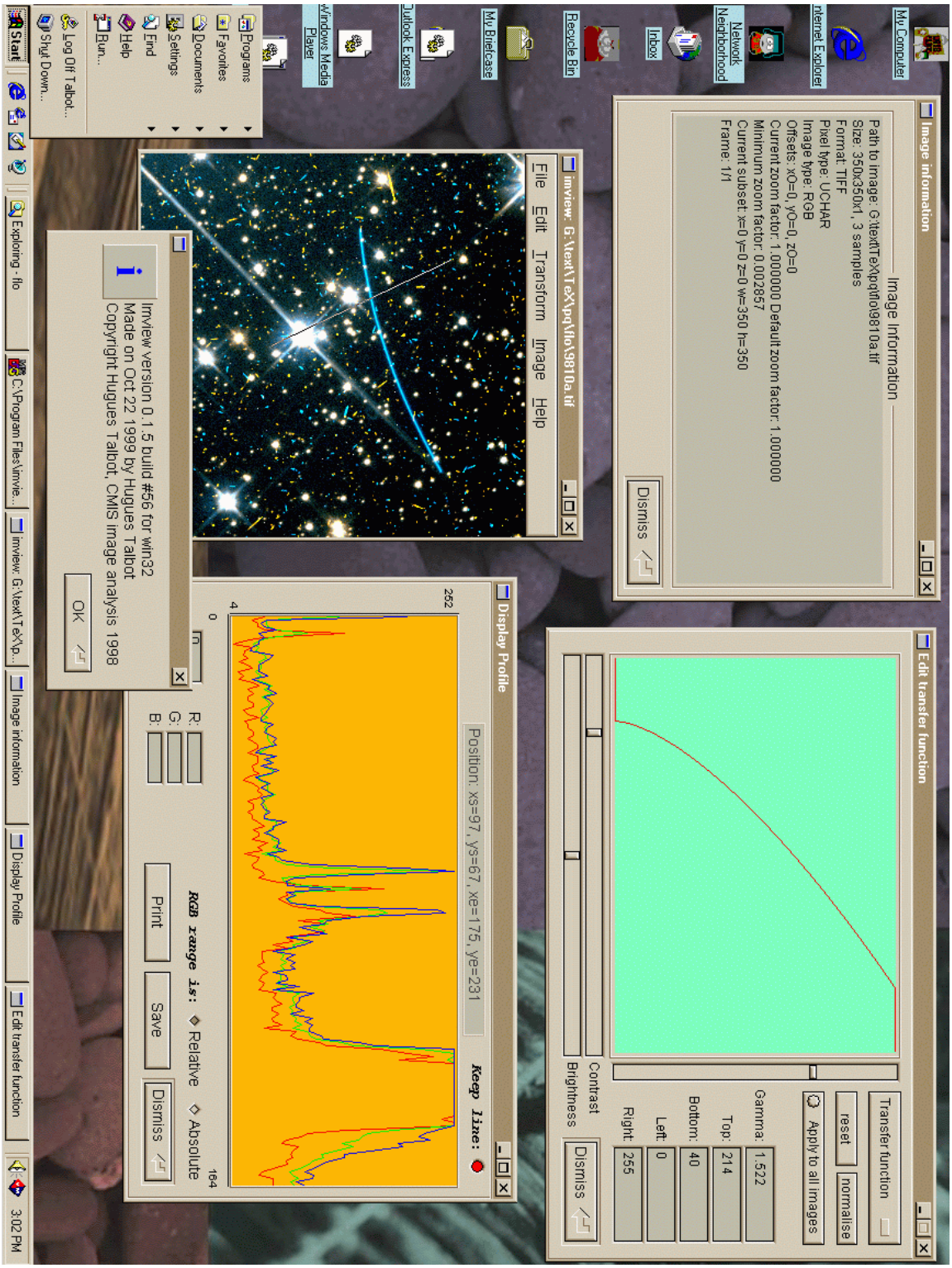


Figure A.1: `Imview` running under Windows.

A.2 Using imview under MS-Windows (95,98,NT)

If you are still with me, in this section we'll explore the few Windows-specific issues regarding imview.

A.2.1 Installing imview

Normally all applications under windows require an 'installer': one usually double-clicks on a setup.exe and something happens.

No such thing with imview. I don't know what installers do. Probably rearrange your DLLs to suit the particular application being installed, and other wicked things, pretending to remember the changes so that you can undo them. Of course if you don't uninstall applications in the precise order you installed them problems may occur, and sometimes do.

A.2.1.1 Nitty-gritty details

In your case, the imview distribution comes as a single .zip file, or maybe as a self-extracting .exe archive.

Main points Create a directory somewhere, for example C:\Program Files\imview, probably where you would like imview to reside, and unzip the distribution there. You should see a imview.exe (the GUI-only executable), an INSTRUCTIONS file, which may contain information more up-to-date than this one, a bunch of .lut files (the Look-Up-Table files) and a .dll file and a doc subdirectory that contains a few .html files.

If you want you can make a link from imview.exe to the desktop so that you can start imview easily. For this just drag and drop imview.exe onto the desktop.

Finally go to the control panels, start System, click on the Environment tab, add a IMVIEWHOME environment variable and set it to the path where you installed imview. This is essential for imview to find all the look-up table files (.lut files). See Fig. A.2 for an illustration.

Now you should be able to double-click on the imview link you made on the desktop. Imview should start, you should be able to open an image, and if it is single-spectrum, apply a look-up-table to it.

Making imview the default for image files If you want imview to start up for you when you double-click on a TIFF, GIF or JPEG file in the explorer, you'll have to set this up manually. For this, start up the Windows Explorer, select View->Folder Options, select the File types tab, select GIF image (for example), select the boldface **open**, press the Edit

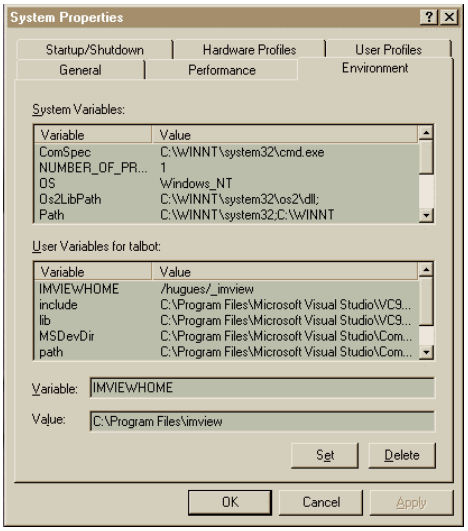


Figure A.2: Editing the system properties to set up `IMVIEWHOME`.

button, unselect the `use DDE` checkmark, and replace whatever is in the `Application used to perform action` input field by the path to `imview.exe`, enclosed in double quotes, as in Fig. A.3. The process needs to be repeated for all the file types that `imview` supports and that you want `imview` to basically "own".

There might be an easier way to do this but I don't know it. Probably some horrible registry hack. I'll probably have to learn about windows installers at some point. Windows users please not that the situation on this point is no better on the Unix front. With both the Gnome or the KDE version of the Explorer (or file manager), a similar effort must be performed.

Start-up directory To specify `imview`'s start-up directory, and if you've made a link to it (for example to start it from the desktop or from the `Start` menu, right-click on the link, and select `Properties`.

Then select the `Shortcut` tab, and enter the desired start-up directory in the `Start in` input field, as in Fig. A.4

A.2.2 Troubleshooting

If you can't make the look-up table work (i.e: the `Transform->Colourmap` submenu only contains `Default Colourmap`), start `imview` from a DOS console with the `-debug` switch, and try to figure out what the problem is. Usually spelling the directory name wrong...

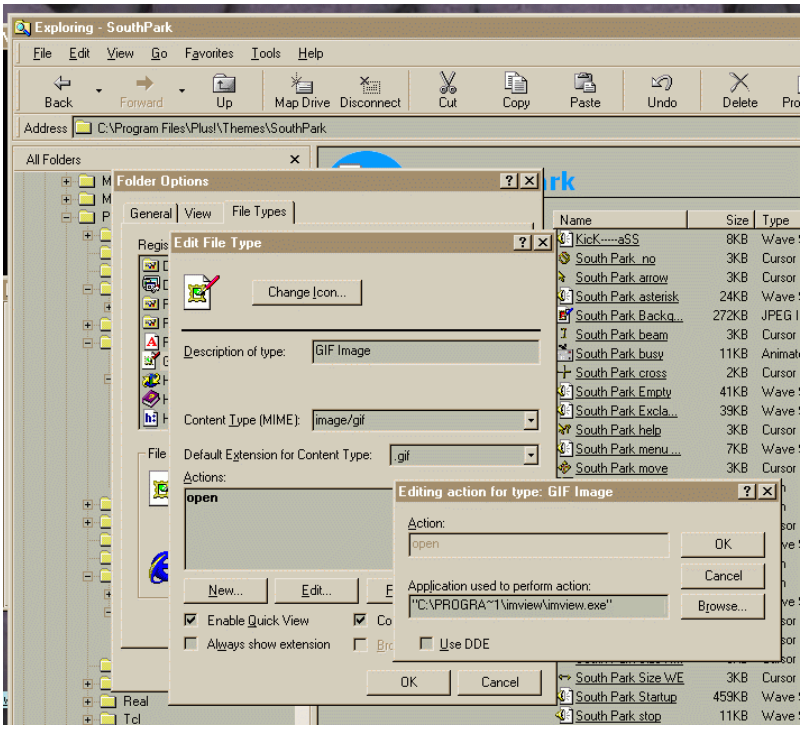


Figure A.3: Editing file type so that `imview` starts automatically when double-clicking on a GIF image.

A.2.3 Un-installing `imview`

To uninstall `imview`, simply delete all the files in the installation directory, this is all. `Imview` doesn't mess with DLLs, registry or anything like this, so you can do this safely at any time.

You can also delete all the links to `imview.exe` and the `IMVIEWHOME` environment variable safely.

A.2.4 Mis-features specific to Windows

Because `imview` is developed under Unix, a few things don't work straight in `imview` under Windows yet. Here's a probably incomplete list.

- Straight printing doesn't work. You can save an image to Postscript and print that somehow (with ghostscript possibly), but that's all for now. I haven't figured out how printing works under Windows. Help would be appreciated.

- Previews usually don't work either (as in `File->Print` and then press the `Preview` button). This is likely to cause a crash actually.

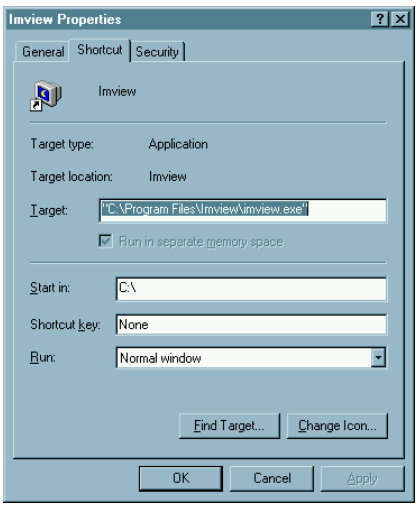


Figure A.4: Editing a link property so that `imview` starts up in a given directory.

- Reading a postscript file probably doesn't work. It should with ghostscript installed but I haven't tested that.
- Paths are automatically translated to their POSIX counterpart. `C:\temp\img.tif` becomes `c:/temp/img.tif`. You can type either of `/` or \ in the `Open file` dialog for example, but `/` will be printed. Backslashes (\) are acceptable in the `IMVIEWHOME` environment variable. Path completion works as under Unix.
- The separator in `IMVIEWHOME` is '`,`', not '`:`'.
- `Imview` doesn't care about MS SDI/MDI conventions. It uses an SDI window with lots of dialogs, in the Unix style. With the usual 'click and raise to get focus' feature of the default Windows graphical shell. This doesn't necessarily work brilliantly. Microsoft actually released a small utility called 'tweak UI' that acts as a control panel. This small application allows users to change a few user interface parameters, in particular to let the keyboard focus follow the mouse without having to click in the window (this is called the 'X-mouse' protocol by Microsoft). If you plan to work a lot with `imview`, get it, it will make your life a bit easier. It can be found on the Microsoft web site, for example at the time of writing the URL was:

  `http://www.microsoft.com/NTWorkstation/downloads/PowerToys/Networking/NTTweakUI.asp`

  The URL implies it's for NT only but it works for Windows 95 also (not 98). Also look for 'power toys' if you can't find it. It may also be on the Windows 98 CD.
- `Imview` has no icon and I don't know how to add one to it.
- 3 button mouse! `imview` works much better with a 3-button mouse. Intellimice are OK. I'm looking forward to the Mac version...

After this frank exposé, I hope you're not being put off... Do try `imview` for yourself and let the author know what you think.

Appendix B

Performances

The data on this chapter is not *seriously* outdated, but I've kept it as a sort of historical perspective.

Inefficient software is at best a proof of concept, at worst a pile of junk. I could have written `imview` in Tk/Tcl or java, but apart from the novelty value, no one would have been interested in actually using the resulting application.

This section shows the results of a short series of benchmarks. The task was simply to open an image and display it on screen. This probably reflects at least 50% of what people do with image display software. Rarely are any of the fancy features used to gain extra insight on the image being displayed. Moreover people get really annoyed if they ask for some image to be displayed and it takes forever, so this task is really the most important among those that an image viewing software must do.

B.1 Setup

These benchmarks were realized on a PC/Pentium 100MHz running Linux 2.0.33/Elf and X11R6.3 (XFree86 3.3) on an S3/864 run-of-the-mill video card with 2MB of video memory on board. Benchmarks compare `x11()`, `xv` and `imview` in the simple task described above, on a series of representative images. Timings were obtained with the `time` TC-shell built-in command. User time and System time are both reported. The user's task was to type the command needed to display one image with the application being tested, wait until the image was fully displayed, and quit the application by the simplest method. Experiments showed that users' influence is insignificant (in particular, the length of "dead" time when the application does nothing waiting for the user to quit is of no influence on these results). Timings were measured three to five times and the median value was retained.

`x11()`, `xv` and `imview` were compared on 8-bit pseudo-colour display and Z-IMAGE format images, because this is the only common ground between the 3 applications (see table B.1).

Table B.1: Timing comparision between three image display applications. Task was to bring up an image on pseudo-colour 8-bit screen. Timing in seconds: first timing is the user time (u), second timing is the system time (s). The sum of both timings is most significant. Best timing sum for each image is higlighted.

Image name	type	format	`x11()`	`imview`	`xv`	image size	dimensions
ibr001_z	CHAR	Z	**0.18u + 0.09s**	0.25u + 0.10s	0.31u + 0.15s	300kB	560 × 530
lw1arb_z	BINA	Z	**0.26u + 0.23s**	0.58u + 0.11s	8.39u + 0.38s	936kB	882 × 1062
mars_z	CHAR	Z	0.53u + 0.60s	**0.57u + 0.29s**	6.82u + 0.28s	2MB	1024 × 1920

`xv` and `imview` were further compared on 16-bit true-colour display using the TIFF and JPEG image formats (see table B.3).

B.2 3-way comparision between `x11()`, `xv` and `imview`

B.2.1 Comments on table B.1:

All three applications perform well enough on small images such as `ibr001_z`. However, on the whole, `x11()` is nearly twice as fast as `xv`, and 1.3 times faster than `imview` on these images. This edge is small in actual fact and is usually lost in the time needed for the user to move the image around before it is shown on screen. It is a good measure on how snappy the application will feel when doing redraws, etc.

On a large binary image (`lw1arb`), `xv` performs appalingly: it is more than 20 times slower than `x11()`, which is again the fastest of the three on this image. `Imview` performs honorably: about 1.4 times slower than `x11()`, but it is still quite responsive.

On a large greyscale image, `imview` becomes the fastest application by a small margin (it is 1.2 times faster than `x11()`). `xv` performs appalingly again: nearly 10 times slower than `imview`.

Although `x11()` is the fastest application on the whole, it pays to remember that this application can cut a few corners because it only understands the Z-IMAGE format, supports only 8-bit pseudo-colour images in addition to greyscale and binary, and only works on pseudo-colour X11 displays.

B.2.2 Comments on table B.2:

On all of those three test images, `imview` is less memory-intensive by a noticeable margin than both `xv` and `x11()`.

Table B.2: Memory comparision between three image display applications. Task was to bring up an image on pseudo-colour 8-bit screen. Size is the total size in megabytes (MB) of the application as reported by the `top` utility after the image was loaded and displayed. Best score for each image is hilighted.

Image name	type	format	`x11()`	`imview`	`xv`	image size	dimensions
ibr001_z	CHAR	Z	2.1 MB	**1.5 MB**	2.0 MB	300kB	560 × 530
lw1arb_z	BINA	Z	4.3 MB	**3.6 MB**	5.7 MB	936kB	882 × 1062
mars_z	CHAR	Z	7.2 MB	**4.0 MB**	7.0 MB	2MB	1024 × 1920

Table B.3: Comparision between three image display applications. Task was to bring up an image on true-colours 16-bit screen. Timing in seconds: first timing is the user time (u), second timing is the system time (s). The sum of both timings is most significant. Best timing sum for each image is higlighted.

Image name	type	format	`imview`	`xv`	image size	dimensions
crni39.jpg	RGB-CHAR	JPEG	**2.15u + 0.18s**	6.16u + 0.19s	115kB	773 × 900
crni39_s.jpg	RGB-CHAR	JPEG	**0.52u + 0.04s**	0.57u + 0.10s	26kB	328 × 385
ibr0001.jpg	CHAR	JPEG	**0.53u + 0.06s**	0.62u + 0.10s	14kB	560 × 510
ibr0001.tif	CHAR	TIFF	**0.52u + 0.07s**	0.59u + 0.09s	100kB	560 × 510
ibr0001_z	CHAR	Z	**0.34u + 0.07s**	0.42u + 0.14s	300kB	560 × 510
lw1arb.tif	BINA	TIFF	**0.98u + 0.16s**	8.44u + 0.31s	2k	882 × 1062
mars.jpg	CHAR	JPEG	**4.22u + 0.25s**	10.52u + 0.33s	1.5MB	1024 × 1920
mars_z	CHAR	Z	**0.85u + 0.22s**	7.05u + 0.20s	2MB	1024 × 1920

B.3 2-way comparision between xv and imview

B.3.1 Comments on table B.3:

In this table, `imview` is always the faster application. The margin is usually slim, except on large images when `xv` starts performing poorly. On `mars_z`, `xv` is nearly 10 times slower! It is interesting to note that compressed images are always displayed more slowly. This was on a machine were the file was stored locally. The outcome would certainly have been different if the files had been stored on a networked file system. In the case of `mars.jpg`, the image file is only 0.7 times smaller, but the display time is up to 5 times slower. In contrast, in the case of `lw1arb.tif`, the display time is about twice slower (compare with `lw1arb_z` in table B.1), but the file is 500 times smaller (2k vs 1MB)! These things need to be considered carefully sometimes.

Appendix C

Code issues

If you're still with me, you may have started to use `imview` or even you've had a peek at the code.

If so, don't be *too* frightened. `Imview` is not small and not particularly well written. This is basically my first real-world C++ project (I did toy things before) and I come from a strong C background. I tend to design huge classes with big interfaces not at all elegant, but that do tend to work. Someday I'll have a picture here showing that I did have an object-oriented design for the whole thing, and that I went through an early design trauma to get something halfway decent.

Also C++ changed quite a bit while I was writing it. I started with gcc 2.7.2 which didn't have an STL implementation to speak of, and the fltk style encourages somewhat "bad" OO approaches (like overuse of globals). In the middle of developing `imview` I finally graduated to reading the Stroustrup book (3rd edition) instead of not-so-good C++ introductory books, egcs became available, the elegance of the STL dawned on me and C++ became an ANSI standard. If I were to restart today I would certainly write `imview` very differently.

As it is I don't care too much. The benefits of redisigning the whole works are not clear to me now. It works for me. I've experimented with various C++ features with it. I've basically had fun doing this and I'm not a style zealot. I'm now reading more advanced C++ books so maybe my next project will take advantage of the new knowledge, or maybe not. Or maybe I'll do it in Ada, as I'm beginning to think that C++ is quite a bit ugly.

Anyway, be critical, please. I welcome constructive comments. The following sections are a roadmap to the `imview` code.

C.1 Malloc vs. New

The code uses both `malloc` and `new`. Obviously `new` is better in general, but when I started there was no standard C++, and no compiler that would implement it, so I did rely on the *perfectly standard* C library, a style that `fltk` encouraged. Also I used a number of free libraries, such as the wonderful `TIFF` library and the `JPEG` library, which allocate their buffer through `malloc`, so I had to keep track of them and `free` them, not `delete` them.

This is not so much of a problem because most buffers are short-lived, of course except the main image buffer (`currBuffp` in the code). The `currBuffp`s are `malloc`ed, not `new`ed. Also some buffers are `strdup`ed, such as `clutpath`. You need to `free` those.

Finally, a pretty important buffer: `imdata`, is always `new`ed, *not* `malloc`ed. Be careful not to confuse `currBuffp` and `imdata`.

C.2 Directory structure

To be written.

C.3 Core code

To be written.

C.4 Extras

Here is an explanation on some of the non-critical aspects of `imview`.

C.4.1 Zooming on the bivariate histogram

I was interested to see if the zoom feature I've implemented for images could have been made simpler using a `Scroll` widget. The answer is yes but the memory cost would have been prohibitive. I'm happy I've done it the way I did. For the bivariate histogram, all we have is a box and a bunch of points. The size of the box is irrelevant to `imview` (although it must cost something to the X or gdi server, I'm not sure), and the memory size of the points is fixed regardless. So we can enlarge or shrink the box to our heart's content with little cost. In this case a `Scroll` and a `Box` implementation was the best idea.

Anyway, the formula for zooming on a bivariate histogram is a bit surprising, but here's the explanation:

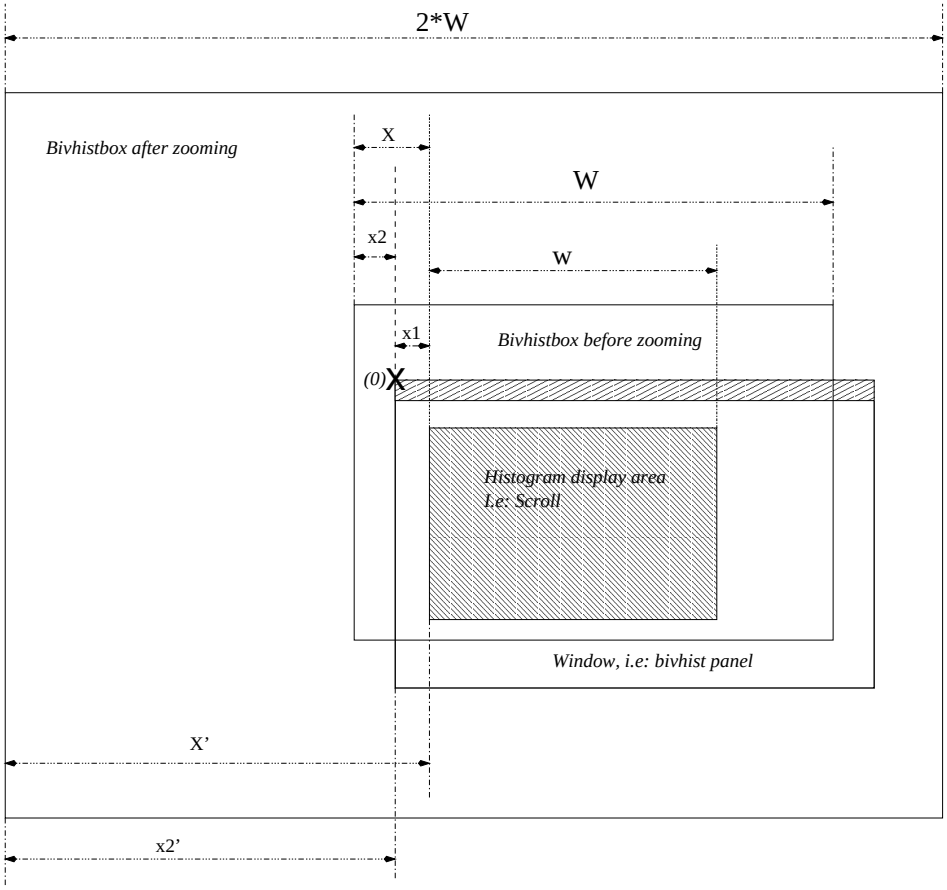


Figure C.1: Derivation of the bivariate zoom formula for the X axis.

We only do the X axis. The Y axis discussion is identical. Let x_1 be the position of the `Scroll`, x_2 the position of the `BivHistBox` with respect to the bivariate histogram panel (window). We assume a zoom factor of 2, but it could be anything.

Then $X = x_1 - x_2$ is the position of the pre-zoom box with respect to the display area (patterned area in the window). X' is the position of the post-zoom box. We have

$$X' = 2.X + \frac{w}{2}$$

where w is the width of the display area. What we want is x_2', the coordinate of the post-zoom box with respect to the panel (window). This is simply given by:

$$x_2' = x_1 - X'$$

Obviously, if W is the width of the pre-zoom box, then $W' = 2.W$ is the new, post-zoom width. All of this is trivially demonstrated by Fig. C.1. You have to assume signed values, again with respect to the window.

To unzoom, invert the formulae. I did that without even thinking and it worked, of course (well, you have to allow for the display area not being a strict subset of the box (i.e: showing

things that are non-box), but the `Scroll` allows that and the scrollbars in particular manage this gracefully).

C.5 Fluid stuff

I should have written a 'panel' class from which all the actual panel derive.

Appendix D

History

Most-recent-first history of `imview`. Not necessarily complete. People contributing to the project should fill this in.

10 Jun 2003 Documentation of version 1.0.1. That version of `imview` was deemed stable. A version is in CVS, which works with FLTK-1.1.x. This history list will be mostly for documentation changes from now on. Software changes are put in `Changelog`.

4 May 2001 Works continue unabated, about to release 0.9.7, tested imview with newer 2.95.3 and 2.96-RH compilers, minor issues with the macro ## hack. Fluid bug in FLTK has been acknowledged and fixed in the CVS repository, but version 1.0.11 is not out yet. When it is, I'll be ready to announce Imview for good.

17 Feb 2001 The technical history has been moved mostly to the ChangeLog file (standard GNU practice). This is good as it lists the changes from versions to version. I have now a set of clients in the shape of PSL's Ed and Ronald. They have a long list of features they want me to implement. I've started on them. I'm now to version 0.9.5 and things are moving rapidly. I still haven't announced `imview` globally. There is an issue with the latest version of FLTK (1.0.10) which makes things difficult, as `imview` does not compile with it due to a bug in FLUID.

02 Jan 2001 A new millennium. Now builds to RPM, relocatable. version 0.9.1

29 Dec 2000 Released 0.9.0 in source form last week. Since then no news because I didn't announce it anywhere. Now trying to make up an 'install' target to the makefile. It's the hardest thing...

There is now a valid "make install" target. Seems to work well, with one level of undo!

20 Dec 2000 Zoé was born early (12/09/2000) so I didn't release `imview` in time... Now that she sleeps a bit better so do we and so I have a bit of time to work on `imview` again. I've fixed a few long-standing bugs, such as the infuriating one with the transfer function panel. Specifying numbers by hand other than the gamma value were just ignored! Also `imview` supports a few more pixel types now. More and more users needed SHORT and Unsigned SHORT support (as many camera acquire 12 to 14 bits these days). Finally the PDF conversion of this present documentation has become a whole lot easier thanks to pdfLaTeX.

28 Aug 2000 An unbelievable amount of work lately. In short I fixed a very serious bug with multithreading on multi-cpu machines under NT. I just had to recompile with Visual C++, a real shame, and something that required, by itself, a lot of effort (renaming all the files, wading through tons of incompatibilities, etc). I've finally done the autoconf business, it works well and is quite nice, now I can have several version of `imview` in several subdirectories for different platforms or options, and maintain them all in parallel. I'm doing that now with about a dozen versions at once. This is really cool. I'm also finally documenting the server stuff and thinking of finishing the arbitrary read business. I just need an ErMapper header parser and I'll be done. I actually have a finite list of things to do **and** a deadline: I want this out before the birth of my first child, due 21/09/2000. Very very soon! I've also fixed countless small bugs, all due to the port to MSVC++ (the fixes, not the bugs!). This illustrate the importance of trying a given piece of code on as many machines as possible: each new one unearthes some long-standing bug!

3 Aug 2000 A lot of work lately. Lots of bug fixes, a new port to windows, up-to-date versions on all platforms, better integration with Voir (and any socket client), and integration with ImageMagick (at the C api level, compiled in!). This allows `imview` to read and write just about any image format known to Man. This comes at a price though: now a statically linked version is quite bulky and memory hungry. 33kLoC.

08 Jun 2000 Several items:

- Authorities have given me the OK for releasing `imview`. That was a long battle!
- I've discussed the license with Ed Breen: it's going to be GPL
- I've got too many things going at once; the socket/shm stuff isn't finished, I'm working on this 'arbitrary read' function (I'm finally using my own templates, I should have done that earlier, very cool), and

users want more features left, right and center. Plus I'm really busy and Annick belly looooooms. Impending Baby means no more work on `imview` unless my activitites at work change dramatically.

- `imview` is no longer portable until I make the socket stuff work under Windows. Should be easy, except there is the multithreading business.

SO, I need to make this thing compile easily. The things I want to do before releasing are:

1. Finish the 'arbitrary read'. This should make it easy for people to add new format (as long as compression is not involved).
2. Make the stuff compile under windows.
3. Adapt to autoconfig **VERY IMPORTANT**.
4. Document socket and shm. Also important, since I will forget all about it if I don't do it, and this is are tremendous feature for extending `imview` easily (and to interface it to arbitrary envs).

The rest can wait. BTW, version 0.1.8 is looming too, and this will be over 30kLoC.

10 Apr 2000 I've been working like a log on the server issue. Now the imview server works. It uses TCP/IP for command and either the same thing again for data transfer (networked hosts) or System-V style shared memory if both the client and servers are on the same host. There is still an elusive (real-time related, I think) bug which is not obvious. But it is usable at present. Many other little things, but Annick found a really good bug with TIFF reading today: I was allocating N times too much memory, where N is the byte size of the pixel (i.e: 8 times too much for a DOUBLE image!!!!).

15 Mar 2000 I've done a LOT of work on `imview` lately, I'm trying to transform it into a client/server application. This has started to work, and this is version 0.1.7. Imview is 28kLoC now. As of writing this work is not documented yet.

30 Oct 1999 Fixed small bug in rotation: had forgotten to rotate the overlay when present as well!

27 Oct 1999 A lot of documentation effort. Ironing out a few long-standing problems with unrecognized formats (double error message), and the problem with ':' as a separator under Windows. This is version 0.1.6

23 Oct 1999 I've corrected the bug explained in the next item, but I've got a new one now. I've been working a bit on the Windows version of `imview` (mostly

the documentation, badly needed). Work on `imview` hasn't progressed a lot due to lack of time, but the beta testing program has born fruits. A lot of feedback (not all positive, which is good), and lots and lots of things to do. Exciting!

28 Jun 1999 (actually 27/06 but it's really late...) I've been working for ages on this one little bug: when an image is rotated the gamma is applied repeatedly. I could have fixed it with a hack (probably), but now I'm extending gamma to work with different transfer functions per channel, and to work with no loss of precision on INT and DOUBLE images (ie: the gamma is no longer applied to the 8 bit image but to the underlying data. The underlying data remains unchanged, only its display varies). This is not easy to do and the size of the code has grown a little: 25kLoC now. I hope this is good. We have temporarily lost a lot of gamma functionality (apply gamma to all, remember gamma, etc), but they will return shortly!

I've been contacted by Suzanne Furby to add reading ErMapper data. This will complicate matters as they will also want to display as RGB 3 arbitrary channels in a multispectral image. Adding this feature is feasible as I'd left some hooks in the code from the very beginning but won't be as easy as a lot of features have been piled on top of things.

24 May 1999 There's been a lot of work on `imview` but I've not been filling up this portion of the documentation at all. Here's the skinny on the changes:

- Added the interactive bivariate histogram. Extremely useful feature.
- Debugged with `insure`. Found *very* few bugs. This is pretty amazing, meaning constant use insures almost a much as good debuggers do.
- Added the overlay feature. This was necessary for the bivariate histogram, and is used only with this at the moment, but potentially can be used for anything (drawing on the image, superimposing text, etc). Somewhere down the track this will be done. At the moment this is not properly implemented either (as it's proper class) but this will be done for 0.2.0.

The current version is 0.1.5.

16 Apr 1999 In the last few days I've added the `-gamma` command line argument (not so easy), and I've tested `imview` with the ElectricFence library. I've found and corrected a number of long-standing bugs this way. Because of fudges with version numbers and build numbers, I'll call this version 0.1.4.

17 Mar 1999 Bug correction day.

24 Feb 1999 Not much work on `imview` over the last few weeks. I've fixed a few bugs, can't remember which. The most important problem was that the default visual was not always the best (32 bit) one, especially on Solaris. This has been fixed but requires a recent version of fltk.

One can now print a list of images in one go on a single page (in effect, this make `imview` a GUI front end to laser). I'll be churning up the minor version more often, each time I do a recompile for all platforms. This is version 0.1.2.

8 Jan 1999 Official 0.1.0 release. Fixed bug with the activation of the scrollbar when trying to define a point just next to the right or lower border when scrollbars were present. Coordinates are now constrained to the visible part of the image. Compiled on all Unix platforms OK. I'm waiting for fltk-1.0 to do further work on NT.

31 Dec 1998 Today I've done the hack that the hyperspectral community wanted me to do: 3D images can be considered hyperspectral-like as well. This makes the wishlist empty for version 0.1.0. So this is it: only 29 builds for 0.0.21, 21.8kLoC.

10 Dec 1998 I haven't worked on `imview` for a while, hoping to find more bugs and fix them. Today I fixed a bug with the redraw of small images. The white background wouldn't redraw if the image itself wasn't damaged. This was because of a change in numbering in the `FL_DAMAGE` enum (fltk change, not mine).

13 Nov 1998 I've "ported" `imview` to DEC Unix + DEC cxx in order to take advantage of the DEC debugging environment. It's not overwhelming, but it's better than nothing, given that Purify does not support gcc 2.8.x or egcs, and neither does sentinel (what a nightmare that is). Found a few minor bugs this way, a phantom leak, but the mere act of compiling on a different platform altogether did show up a few bugs in both the toolkit and `imview`. Finally got to implement the map idea for Postscript files. Now postscript files are no different to the user than any other image file format, except there are a few rendering options. Neat. Finally the reading and writing of the user preferences is working well. We can probably call this version 0.0.21. Version 0.0.20 went up to build #242 and 21.6kLoC.

9 Nov 1998 Small amounts of work on the project. I've moved to the dated '1998...' versions of fltk in an effort to improve bug fixing in the toolkit. This seems to work fine, I've had a few annoying bugs fixed that way. Hoping version 1.0 is just around the corner.

8 Nov 1998 Work on warnprintf, on a small message browsing widget, and on improving the Postscript support.

22 Oct 1998 Finally gave the tutorial. Did I impress anyone? As a result there is a growing tutorial section in this document. Slides are on a separate LaTeXfile.

15 Oct 1998 Giving tutorial on `imview` today. Postponed till next week for lack of attendance (people sick or on holiday).

14 Oct 1998 Worked on imview to make it compile and run on Windows NT (win32). Managed that feat by using a Linux-¿NT cross compiler. The new version runs just fine on these platforms. Also ported it to the latest beta release of fltk (19981006). Seems to be OK. Will allow the general style of `imview` to be changed easily (fonts, sizes, etc).

OK, so this is version 0.0.20. Version 0.0.19 went up to build #206 and 20.1kLoC.

8 Oct 1998 A lot of work on the documentation, following Leanne's comments and the new features of 0.0.18 and 0.0.19. Preparing for the tutorial next week.

2 Oct 1998 A lot of work on the 1-D profiles. It works fine now: One can view them and save them. They display in RT on the profile panel. When I can print them, this will be version alpha 20. fixed bug with DOUBLE Z image for which the endianness was never recognized properly.

30 Sep 1998 I consider that the following has been fixed:

> "At high magnifications, even though the zoom box tries to fall on pixel boundaries, the subset of the image that is finally zoomed in does not necessarily correspond exactly to what has been outlined (might be +/- one pixel off). This is no big deal but what is the point of being precise in the outline if this precision is not used? (30 Apr 1998: I know why this is now, but this is a bit hard to fix and not a priority at this point...)."

29 Sep 1998 Improving selection of same and non-same aspect ratio subwindows using mouse. Corrected small bugs with window dimensions reporting.

18 Sep 1998 Small work these days trying to allow a non-same aspect ratio zoom, and to improve the accuracy of the estimated zoomed window. I got the position more or less right, but the width and height are still off by a little in same-aspect ratio, and by a lot in not-same.

2 Sep 1998 There has been little work on `imview` in the last month, however I've managed to put together the spectrum viewer for Mark. It works really well now, and from tonight can print and save spectra. 18.6kLoC.

30 Jul 1998 Added a few informative bits in the image information panel. Releasing 0.0.19. Previous release went up to #105, 16.3 kLoC.

29 Jul 1998 Very productive day: `imview` can now read all PNM formats, and can read from the standard input. That makes reading in silly image formats much easier (`xxxtopnm | imview -`).

28 Jul 1998 Fixed problem with white background that was redrawn too often when viewer 3-D or multispectral images.

22 Jul 1998 Worked on ironing out some more bugs with switching images. Now the transfer function (gamma, etc) can be kept when going from one image to the next.

17 Jul 1998 Working hard on fixing the 'display of small images' problem, and the long-standing problem of the redisplay when switching from one image to the next.

9 Jul 1998 Fixed the following item:

> "When defining a zoombox with scrollbar present, dragging the mouse over the scrollbars causes the zoombox to disappear, the image to zoom and pan to correspond to the mouse position. This is not the intended effect!"

I'll call this version 0.0.18, and we'll move on. Announcement for the first Beta will be delayed a bit until this version is better tested. I don't trust fltk-0.99 too much at the moment. 0.0.17 went up to #235.

8 Jul 1998 Back from Europe, I've been working on porting `imview` to fltk-0.99. This proved a bit difficult since a lot of things have changed. I think things are OK now. I've improved things a bit also by removing an unecessary redisplay of the whole image when changing the main window size. This essentially happened when switching to another display. I've also made the image list a bit more reasonable by listing the images in alphabetical order, not the other way around.

7 May 1998 After a few long evenings, `imview` now has the long-awaited contrast,brightness and gamma controls! This is really good, and make the software that much more useful. I'll be able to call this the first Beta when I get the chance to clean some of the cruft and finish the documentation, among others. Version is up to 0.0.17#117, 14.4kLoC. Possibly the last thing that I'll add will be the spectrum viewer for Mark. We'll see if I get the time...

4 May 1998 Nearly finished with contrast/etc. I'm building a version for Solaris, and testing the software under Purify.

2 May 1998 Started working on contrast/brightness control. This is a tiny bit tough.

29 Apr 1998 `Imview` can now save images as TIFF or Z-IMAGE files. The whole image or just parts of it can now be saved easily. This is now version 0.0.17, project went up to 0.0.16#40, 13.5kLoC.

26 Apr 1998 `Imview` can now print correctly, either the whole image or parts of it. This is great! Extensive testing with gs. Releasing 0.0.16. Project went up to 0.0.15#122, 12kLoC.

22 Apr 1998 Little work on `imview` these past days. However yesterday and today I worked on the print preference panel using fluid. This is a test bed for the really complex panel for the histogram control. This is great fun still, however.

17 Apr 1998 Added support for LUT on INT images. I decided to display grey-level or RGB INT images 'fitchared', i.e: the whole INT dynamic reduced to 0-255 linearly, except if the INT value all fit between 0 and 255, in which case they are displayed as they are. However, if a LUT is applied, the INT values are taken as is and the LUT is applied on the low 8 bits of the INT value ('chared').

14 Apr 1998 Added minimal support for multi-page documents. This is fun though. One can now view animated GIFs in `imview`, and use `imview` as a LaTeXpreviewer...

13 Apr 1998 Serious work on the documentation. Split the manual into two: the "rambling" part with the rationale, the wish list, the TODO list, the benchmarks and the history on the one hand (not terribly interesting to the average user), and the user manual on the other. They still make up a single document, but the user manual can be distributed separately very easily. I tried to cover all aspects of version 0.0.15.

12 Apr 1998 releasing 0.0.15. Added and tested line overlay support. This is used in visualizing the 'breaks' in pointfiles. Project went up to 0.0.14#151, 9.7kLoC.

8 Apr 1998 Added one-page on-line help in the form of an image (strange idea, that!), corrected byte swap problem with Solaris (I still don't understand what the problem was, but anyway), added link to full documentation (starts netscape if present). corrected small bug found out by my one Beta Tester: Ronald: when clicking to get pixelwise information, you had to wiggle the mouse about a little to get the status line to appear. Project is at 0.0.14#108, 9.4kLoC.

7 Apr 1998 I'm so busy! very little progress on imview. Nevertheless, over the last few days: added support for 'breaks' in point file (both in read and write form). Corrected a small bug (uninitialized pointer) that only showed up with Carolyn!. Simplified the zoom selection box code by using `fl_overlay_rect()`. Had I known it existed, it would probably have saved me several hours of coding... Also corrected bug in pointfiles: format was actually row, column, value, not x,y, value. Bloody statisticians.

31 Mar 1998 Corrected small bug in startup code. Made the change of cursor work for DEC Unix 4.0b.

27 Mar 1998 Implemented default zoom factor. Now any zoom factor can become the default, not just 1.0 anymore. Implemented several ways to set the zoom default (menus, command line,etc). Improved command line handling of colour maps (they don't have to immediately follow the image name anymore). Implemented sticky points (they stay around when flipping images, which can be handy) as a preference. Corrected a nasty bug in pointfile (case of a single component double image causing a segfault). Updated bits of the doc.Project is at 0.0.14#76, 9.1 kLoC.

26 Mar 1998 In the last few days (not just today): added support for 'short' (16 bits) images. `Imview` now reads the SCILimage format (ics). Some confusing error messages fixed (and subsequent bugs fixed as well...). Added a few menu items to make setting the zoom, z slice and component number easier.

19 Mar 1998 Work on the documentation. Trying to catch up with all the features... Also added a simple dialog box for displaying error messages.

18 Mar 1998 Releasing 0.0.14. Fixed up some bugs, added support for reading/parsing poinfiles (so that pointfile can be exported across images), cooked up a new format for pointfiles, added support for extra keyboard shortcut (I like the toggle idea), added support for pointfile control both from the GUI and the startup options. 0.0.13 went up to build #268, and 7.2 kLoC, the most difficult release so far. I deem the "travelling window bug" fixed.

16 Mar 1998 Some work on `imview` during last week at idle times (few and far between, as this is the week of the melanoma report). Added support for off-screen buffer at zoom 1.0 when the image is small enough to fit entirely on the screen (no buffer otherwise). This speeds up redraws tremendously for most images. Added support for deleting random points (in pointfiles) via alt-shift click. Added sensing of actual screen dimensions to specify maximum sizes. Done some timing on "Fire". On this machine, `imview` is always faster than `x11()`!

10 Mar 1998 Major progress last night. Got the zoom box redraw working properly at all zoom factors. Point drawing works well. Saving points in files works well also. Switching image saves and discards all the points. There are still a few problems (see bug list) with information box redraws and point redraw but when that is fixed we can call this release alpha 14 and update this document more fully. project is at 0.0.13#184 (!), 6.6 kLoC.

9 Mar 1998 Problem with zoombox not being redrawn properly has been fixed (no need for a queue of event, I just protected the access to the drawing method with a trivial semaphore).

9 Mar 1998 Lots and lots of work on pointfiles. Accelerated the redraws of most things (most noticeable on zoom boxes) to make drawing of cute little round indicators for points in pointfiles workable. The redraw of boxes with zoom present is still far from perfect, though.

1 Mar 1998 Started work on pointfiles. Worked out how to detect the Shift key when the mouse is dragged (nothing like reading the documentation!). I want to avoid the pitfalls of `x11()` on this matter but it's not as easy as it seems. Looks like shift+mouse operation will control point operations. Now debugging messages can be turned on and off interactively (via preferences). Project is at 5.5 kLoC.

24 Feb 1998 Released 0.0.13. Still more improvements in the code. `imview` now supports 3-D images. Did benchmarks for comparision between `xv`, `imview` and `x11()`. Reports will follow shortly but upshot is `xv` is a pig for big images (why?), `x11()` is very fast on all but the bigger images, and `imview` is OK. It is still slower than `x11()` except when the images become very large, but it is always faster than `xv`, and **much** faster on large images.

21 Feb 1998 Released 0.0.12. Lots of improvements in the code, now much more object-oriented. Got rid of many global functions, eliminated dead code. The end result seems to be working OK. Still gobs to do but we are getting closer to a better model for image display. The question is: what should the interface remember when going from an image to the next in the image list, and how to implement the memory?

17 Feb 1998 Still more work on the documentation. No time for development, unfortunately.

11 Feb 1998 More work on the documentation. Fixed bug with splash screen (when no image was given on the command line, program would crash instead of showing splash screen).

9 Feb 1998 Worked on version 0.0.12 at home over the week-end. I'm trying to reorganize the I/O mechanism with a more central class to handle all inputs from the GUI. Unfortunately my screen blew up over the WE and will hamper development a bit, I think. We'll see.

6 Feb 1998 Started working on this documentation. Current release is `imview` version 0.0.11 built #15. This document placed under CVS as well.

5 Feb 1998 Automatic increment of the build number.

1 Feb 1998 Bug fixing mode: mouse events, CLUTs, resize, etc.

28 Jan 1998 Change to fltk-0.98.

22 Jan 1998 Working on the Colourmap issue. Common interface for GIF and JPEG. Trying to deal with thin and wide images, or narrow and tall ones.

21 Jan 1998 `Imview` can now read the Z-IMAGE format. `Imview` can work as a Z-IMAGE function transparently.

12 Jan 1998 Implemented the TIFF reader. Previous to this, `imview` could only read JPEG images...

7 Jan 1998 Implementing the dynamic image list. Ran through Purify.

16 Dec 1998 On the road to add command line argument checking.

4 Dec 1998 version 0.0.10. Most feature are working OK: menu, image loading (Z and Tiff), etc, but the overall design of the application is not good. It can't grow easily.

16 Nov 1998 Started implementing dynamic menus.

15 Nov 1998 version 0.0.4. Working menu. Working zoom. Patchlevel file created.

11 Nov 1998 Changeover to FL-0.96

8 Nov 1998 version 0.0.3. Embryonic menu. Buggy zoom.

31 Oct 1997 Started working on ImView. CVS repository created. Version 0.0.1 compiled: displays a JPEG image with no control whatsoever, using FL-0.95.

Appendix E

GNU Free Documentation License

Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

E.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text

near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

E.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

E.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

E.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

E.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

E.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

E.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts

may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

E.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

E.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

E.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.