A User's Guide to the Generalized Image Library

Leonard G. C. Hamey
Computer Science Department.
Carnegie-Mellon University.
Pittsburgh, PA 15213

Revised August 9, 1990

Abstract

This document describes those aspects of the generalized image library which affect the users of programs built with the library. A separate document. provides information for programmers who wish to use the library.

This research was sponsored by the Defense Advanced Research Projects Agency. DOD through ARPA Order No. 4976. and monitored by the Air Force Avionics Laboratory under contract F33615-84-K-1520. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied. of the Defense Advanced Research Projects Agency or of the U.S. Government.

Contents

1 Introduction

The generalized image library provides access to a variety of image devices and disk formats. Programs which are built with the library inherit this flexibility: the same program can access all the different image devices and disk formats depending only upon the image name which the user specifies. This document describes the image naming syntax which is supported by the generalized image library and programs built upon it.

2 What's In a Name?

When the generalized image library opens or creates an image, the name which you give serves to do more than just name a disk file in which the image will reside. At the very least, the name of an image indicates the *format* in which the image will be stored on disk: it may even indicate that the image is not to be stored on disk at all but is actually a *physical device* or a *virtual image*.

Currently, there are three image *formats* which are supported. They are: CMU format, GIF format and MIP format. Also supported are a number of frame buffer devices, xwindows and suntools displays, network access and a number of operations which can be applied to images. Figure 1 is a BNF grammar representing the set of image names which is currently supported.

Some of the above facilities can only be used when an existing image is being opened. Others are only valid when a new image is being created. The restrictions are as follows.

- Disk files cannot be created over the network. This is to prevent security problems.
- The shift keyword is not supported when a new image is being created. The constant keyword can only be used as an input image.
- The unsigned, signed and float keywords are not allowed when an existing image is being opened.

3 Formats of Images on Disk

The three image formats, CMU, GIF and MIP, have different uses and capabilities. This section describes each format briefly to assist you in choosing which to use.

3.1 CMU Format

CMU image format is an obsolete image format which was supported by the old image library. The pixel type information which was previously handled explicitly by some programs is now handled directly by the image library. This means that library programs are no longer confused about signed images and can even be expected to do reasonable things with floating-point images.

image-name:

> *(image-name)*
> -
> display
> matrox
> suntools
> xuin
> memory
> matrix
> *name*.img
> *name*.gif
> *name*.mip
> gif:*file-name*
> mip:*file-name*
> color:*image-name*
> stereo:*image-name*
> threed:*image-name*
> 3d:*image-name*
> bw:*image-name*
> digitize:*image-name*
> bands:*band-names*:*image-name*
> *printer*[:width=*width*]
> constant:*constant*[,*constant*,...]
> unsigned:*bits-per-pixel*:*image-name*
> signed:*bits-per-pixel*:*image-name*
> float:*bits-per-pixel*:*image-name*
> *machine*:*image-name*
> network:*machine*:*image-name*
> magnify:*factor*[,*factor*]:*image-name*
> quarter:*piece-number*:*image-name*
> shift:*row-start*,*column-start*:*image-name*
> multiply:*multiplier*:*image-name*
> add:*addend*:*image-name*
> ltrans:*multiplier*,*addend*:*image-name*
> crop:*row-start*,*row-end*,*column-start*,*column-end*:*image-name*
> extend:*constant*,*row-start*,*row-end*,*col-start*,*col-end*:*image-name*
> divide:*row-divisions*,*column-divisions*,*piece-number*:*image-name*
> tee:*image-name*,*image-name*

Figure 1: A BNF grammar for image names.

CMU format is useful for large images because the image data can be accessed on disk as it is needed. This is called software paging.

3.2 GIF format

GIF (generalized image format) was developed exclusively for the generalized image library. GIF format provides full support for the pixel types which tlie generalized image library implements: **unsigned. signed** and **float.** GIF format images are stored internally as matrices (see *matrix(3)*). This places some size restrictions on images which can be stored in GIF forinat as they must be entirely loaded into computer memory.

Tlie advantage of GIF format is that it provides for repetition-based packing which is especially useful for large sparse images.

3.3 MIP foriiiat

MIP format was developed on tlie Suns. A MIP image consists of 480 rows of **512** columns of 8 bit unsigned pisels. MIP format is very restrictive because there is no support for pixel types other than unsigned 8 bit pixels. and because tlie images bounds are fixed.

The only reason for using MIP format is to provide compatibility with existing software which requires MIP format.

4 Naming Images

Tlie name of an image is not simply a file name. In section **2** a BNF grammar was presented which summarizes the expressions which may be used to name images. This section descrihes each expression in detail and gives examples of its use.

4.1 Images on Disk

The name of an image on disk specifies not only its file name. but also the format in which it is stored. There are two ways in which the format is specified: *keywords* and *file types.*

File types are tlie simplest method of identifying image formats. File names which end in **.mip** are assumed to be MIP format unless a *keyword* is used. File names which end in **.gif** are assumed to be GIF format and file names which end in **.img** are assumed to be CMU format. If an image format cannot be identified by its file type. then CMU format is assumed.

If an image format is not correctly indicated by the file type. then a keyword may be used to indicate the format of the image. An image format keyword is one of the character strings **cmu.** mip or **gif** followed by a colon and preceding the iniage name. For example. if the file **sunset** is a MIP forinat image file. then the name **mip:sunset** identifies the file and specifies that it contains a MIP format image. The following command illustrates the use of the imgcp image copying prograin to convert a MIP format file into CMU format.

```
imgcp mip:sunset sunset.img
```

4.2 Piped Images

An image name which is a single hyphen (-) represents a piped image[1]. Piped images may be used for input and for output. **A** piped input image is read from the standard input and a piped output image is written to the standard output. Each program can use only one piped input image and one piped output image.

Programs cannot use piped images if they use tlie standard channels for other purposes. This means that interactive programs cannot use piped input images. It also means that programs which print messages on tlie standard output cannot use piped output images[2].

Piped images are useful for conibining simple programs in shell commands. For example. the following shell command uses the smooth program to low-pass filter an image. It then pipes the low-pass image into subiing where it is subtracted from tlie original image to produce **a** high-pass iniage.

```
smooth gauss -5 original.gif - | subimg original.gif - highpass.gif
```

4.3 Constant Images

The keyword **constant:** followed by an integer or floating-point constant may be used to open **a** constant image. Constant images are virtual images which are filled with a coiistant value. Every image fetch operation performed by a program will return the constant value.

Constant images do not have known image bounds. They are essentially unlimited in size. therefore their use requires a little care. Constant images are most useful in conjunction with programs such as **add.** Consider the following command.

```
add tree.img constant:100 bright.img
```

This command adds together tlie two images **tree. img** and **constant:100** producing the new image **bright. img.** This has the effect **of** adding tlie constant value **100** to the pisels of tree.img and storing the new values in **bright. img.**

[1]The hyphen syntax is in accordance with a Unix convention.

[2]For this reason. the generalized image library does not use the standard output. Everything printed by the library is put on the standard error channel.

It should be noted that. because a constant image contains a fixed constant. it is incorrect to use one as anything but an input image to a program. Also, because the bounds of a constant image are essentially unlimited. it is impossible to copy a constant image to a disk image without specifying the region to be copied. The following command uses **imgcp** to create a CMU format disk image with 200 columns and 100 rows containing the integer constant value **3.**

```
imgcp crop:0,99,0,199:constant:3 con3.img
```

The keyword constant: may also be used to create a multi-band constant image. Instead of a single constant value. several constants may be specified separated by commas. The number of constants must match the number of bands in the multi-band image. For example. the following command fills the matrox display with red.

```
imgcp -c con:255,0,0 matrox
```

The keyword constant: may be abbreviated to con:.

4.4 Display Device

The keyword display (which does not require a colon) indicates the display device appropriate for the machine. This keyword is commonly an alias indicating a hardware frame buffer or a display on a remote machine.

The following command copies the image tree.gif to the display device n-here it can be viewed on a monitor.

```
imgcp tree.gif display
```

The keyword display can be abbreviated to dis.

4.5 Matrox Boards

The keyword matrox (which does not require a colon) indicates the Matrox display device. This keyword can only be used on machines which have a Matrox display. The Matrox supports 8 bit unsigned integer pixels. It has 480 rows and 512 columns.

The following command copies the image **sunset.gif** to the Matrox display where it can be viewed on a monitor.

```
imgcp sunset.gif matrox
```

4.6 Androx Boards

The keyword **androx** indicates the Androx display device. This keyword can only be used on machines which have an Androx display. The Androx supports 8 bit unsigned integer pixels. It has 480 rows and 512 columns.

4.7 Selecting Pixel Characteristics

The keywords **unsigned:**, **signed:** and **float:** may be used to select tlie pixel characteristics of a new image which is being created by a program. The selected characteristics override the characteristics which tlie program would ordinarily select itself. However. the selected characteristics may he overridden by the library depending on tlie capabilities of the image file or device.

Tlie keyword **unsigned:** is followed by an integer. a colon and an image name espression. Tlie keyword selects unsigned integer pixels. An unsigned integer pixel holds only positive pixel values. For example. an unsigned 8 bit pixel holds integral pixel values in the range 0 to *23.5.* The integer. which is optional. indicates the number of bits needed to store each pixel. If it is omitted. then tlie image will be created with tlie number of bits per pisel that was selected by tlie program.

The **unsigned:** keyword niay be abbreviated to **u:**. The follow-ing example specifies that the Sobel edge detection algorithm is to allow 16 bits for the detected edge magnitudes.

```
edge Sobel house.gif -m u:16:house.mag.gif
```

The keyword **signed:** is similar to **unsigned:** but selects signed integer pixels. A signed 8 bit pixel holds integral pisel values in the range -128 to 127. The keyword **signed:** may be abbreviated to **s:**.

Tlie keyword **float:** selects floating-point pixels. As with the other pisel types. the number of bits per pixel may be omitted. In that case. it defaults to the size of a floating-point number on the machine[3]. The keyword **float:** may be abbreviated to **f:**. The following example copies an image and converts it to floating-point pixels.

```
imgcp sunset.gif f:32:sunset.float.gif
```

4.8 Network Access

Tlie keyword **netuork:** indicates that an image is to be accessed over the network. Network image access is only available when the remote machine is running the image network daemon. The **network:** keyword is followed by the name of the remote machine, a colon and the name of the

'Floating-point numbers are 32 bits on the Vax and the Sun.

image on tlie remote machine. Due to tlie difficulty of validating reniote users. it is not possible to create or modifv image files over tlie network. For example. tlie following command copies the image **ocean.img** to the display device on the **iusb** Sun (i.e. its matrox).

```
imgcp ocean.img network:iusb:display
```

Tlie keyword **network:** may be abbreviated to **net:**. Tlie keyword may also be completely oiiiitted if the name of the remote machiiie is known to the library. Thus. the following command also copies the image **ocean.img** to the display device on the **IUSB** Sun.

```
imgcp ocean.img iusb:dis
```

4.9 Shifting an Image

Tlie keyword **shift:** may he used to shift tlie co-ordinates of an image. The keyword is followed by the desired starting row and column co-ordinates. Either co-ordinate may be omitted and will default to the actual co-ordinate of the image. The image. which must already exist. will be relocated so that its rows and columns start at tlie specified co-ordinates. Tlie shifting is done entirely in software and has no effect on tlie image file itself.

The following command illustrates the use of imgcp to create a copy of **house.img** which starts at row 100 and column 200.

```
imgcp shift:100,200:house.img newhouse.img
```

4.10 Cropping an Image

Tlie keyword **crop:** can be used to select a portion of an image. The keyword is followed by tlie desired row and column start and end bounds. The specified bounds must be within the actual bounds of the image. Any of the bounds may be omitted - they will default to the actual image bounds.

Tlie following command illustrates the use of **imgcp** to create a new image **branches.img** which contains a portion of the image **tree.img**. Only the ending row bound has been specified, so the other bounds default to the actual image bounds. The selected portion of **tree.img** is thus all the rows from the top of the image to row 200.

```
imgcp crop:,200,,:tree.img branches.img
```

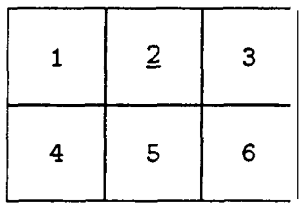

Figure 2: Result of **divide:2,3,3:display.**

The **crop:** keyword may also be used in the name of an output image which is being created by the program. In that case the generalized image library opens an existing image or a display device and allows the program to overwrite the specified portion of the image. Thus. the following command copies **small.img** to the top-left corner of the display device. An easier way to achieve the same result is presented in the next section.

```
imgcp small.img crop:0,239,0,255:display
```

4.11 Dividing an Image

It is often useful to be able to access a portion of a display device. The **divide:** keyword can be used to divide an image into pieces and provide access to a specific piece. The keyword is followed by three integers separated by commas: the number of vertical divisions. the number of horizontal divisions and the piece number. The pieces of an image are numbered from one in standard row-order sequence. For example. the following command copies a very small image to the top right-hand corner of a display which has been divided into six pieces: halves vertically and thirds horizontally.

```
imgcp tiny.img divide:2,3,3:display
```

The arrangement of the display divisions is shown in figure 2.

The **divide:** keyword is especially useful in conjunction with the automatic zoom option **-z** of **imgcp,** which magnifies the input image to fit the specified output image. This allows arbitrary images to be displayed on portions of the screen with reduced resolution. For example. the following two commands copy a color stereo image to the display. The first command copies the left portion of the image to the left half of the screen. The second command copies the right portion of the image *to* the right half of the screen.

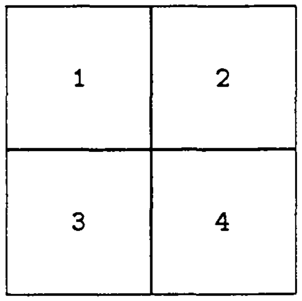

Figure 3: Quarter divisions of an image.

```
 imgcp -zc city.left.red.mip divide:1,2,1:display
imgcp -zc city.right.red.mip divide:1,2,2:display
```

The keyword **divide:** niay be abbreviated to **div:**.

4.12 Quarters of an Iiiiage

The keyword **quarter:** may be used to select a quarter of an image. It is followed by an integer which is the quarter number. then a colon and an image name expression. The quarter number ranges from 1 in the top-left corner to 4 in the bottom-right corner as shown in figure **3.**

Tlie keyword **quarter:** is **a** synonym for **divide:2,2**. It may be abbreviated to q:. For esample. the following command copies **archuay**.img to the bottom-left corner of the display.

```
 imgcp -z archway.img q:3:display
```

4.13 Extending an Iiiiage

The **extend:** keyword is the opposite of **crop:**. Instead of reducing the size of the image. **extend:** increases the image size by filling around the image with either **a** constant value or replicated pixels. The keyword is followed by an optional constant fill value or, for multi-band images, a constant fill vector enclosed in parentheses. When using parentheses on shell command lines it is important to enclose the entire image name expression in quotes.

Tlie optional fill value is followed by the row and column start and end bounds of the extended image. Any of the new bounds may be omitted and defaults to the bounds of the underlying image expression. If a fill value is specified then the image is extended with the value. If no fill value is specified then tlie image is extended by replicating the border pixels. The following esaniple extends an image by replicating the border pixels. Tlie input image has 480 rows and **512** columns and lias origin zero. The output image has 701 rows and columns with the origin at -100.

```
imgcp extend:,-100,600,-100,600:fred.mip extendfred.gif
```

As another example. the following command extends **a** color image by surrounding it with red pixels.

```
imgcp 'extend:(255,0,0),-100,600,-100,600:aus.red.mip' aus.red.img
```

4.14 Tee

The **tee:** keyword is useful for putting an output image in more than one place at once. For instance. it can be used to write tlie output of a program to an **image** file for later use aiid also display it on a display device. Tlie **tee:** keyword is followed by the names of two images. separated by a comma. If tlie first iiiiage naine contains a comma. then that name slioulcl be enclosed in parentheses.

As **an** example of the use of **tee.** tlie following comniaiid uses the *sobel(1)* program to detect edges. Tlie detected edges are stored in the image file **edge.img** aiid are also displayed for viewing while tlie program is running.

```
edge Sobel house.img -m tee:edge.img,display
```

4.15 Magnify aiid Zoom

The **magnify:** keyword may be used to magnify or reduce **an** image. Tlie keyword is followed by a rational number which is tlie magnification factor. The magnification factor may optionally be followed by a second magnification factor which is followed by a colon and an image name expression. Tlie magnification is done entirely in software and lias no permanent effect on an input image.

The magnification factor is a rational number, represented as two integers separated by a slash (/). If it is greater than one then tlie image visible to the program will be a magnified version of tlie physical image. A niagiiificatioii factor of less than one effects a reduction in which the image visible to tlie program consists of pixels selected from tlie physical image.

For esample. suppose that **small.img** is a small image and we wish to display it magnified four times. We can use tlie **magnify:** keyword and tlie **iingcp** command as follows. The magnification is performed by replicating each input pixel value to occupy 16 pixels of tlie display.

```
imgcp magnify:4:small.img display
```

As another example, suppose that `large.img` is too large to fit on tlie display device. **Imgcp** may be used to magnify it by a factor of 2/3 and display it as follows. Tlie reduction is performed by selecting four of every nine input pixels.

```
imgcp magnify:2/3:large.img display
```

When tuo magnification factors are specified. tlie first is applied to rows of the image and tlie second is applied to columns. It is thus possible to achieve an aspect ratio adjustment. For example. the Matrox has an aspect ratio of approximately four rows to every three columns. This means that iniages digitized with tlie Matrox have a greater density of pixels in the row direction. When such images are printed with a standard aspect ratio they appear vertically stretched. This effect can be corrected by magnifying tlie columns of the image by a factor of 4/3 or. equivalently. magnifying tlie rows by 3/4. The following example corrects the aspect ratio of an image and prints it using iht.

```
iht magnify:3/4,1:tree.img
```

Tlie `zoom:` keyword is similar to `magnify:` but is intended to be used with output images. especially display devices. It is important to note that both `magnify:` and `zoom:` establish a relationship between tlie external iinage which you see and the internal image which tlie program sees. The effect of `magnify:` on an output image is therefore somewhat counter-intuitive. For example. consider the following command.

```
imgcp phone.img magnify:2:display
```

At first sight. it may appear that this command displays `phone.img` enlarged by a factor of tno. Actually, the displayed image will be two times *smaller* than `phone.img`. This occurs because tlie `magnify:` keyword establishes a relationship between the external display device and the internal image in which tlie external image is half the size of the internal image. In such situations. the `zoom:` keyword should be used. `Zoom:` achieves tlie intended effect because it is designed for use with output images. For example, the following command displays `phone·img` zoomed by a factor of two.

```
imgcp phone.img zoom:2:display
```

The `zoom:` keyword is especially useful in conjunction with tlie **cursor** program. described in section 8.3.

4.16 Digitization

The generalized image library supports two methods of digitizing images. Sequences of digitized images can be processed using tlie iiiiage sequence capabilities. Individual images can be digitized when they are opened by use of the **camera:** or **digitize:** keywords.

The **camera:** keyword is followed by up to four integer parameters: tlie caniera number. the gain and offset and an interaction level. The camera number defaults to 0. The gain and offset default to reasonable values for the display device. The interaction level defaults to 1. which indicates interactive digitization. The parameters are followed by a colon and an image iianie expression.

When the generalized image library encounters tlie **camera:** keyword. it first opens the iinage name expression following the camera parameters. The library expects that iniage to be some sort of physical device which supports digitization. After opening the image it uses the specified camera number. gain and offset to activate the digitizer. If the interaction level is zero. an image is grabbed and returned to the program. If the interaction level is not zero. a siniple interactive command interpreter is entered. Tlie command interpreter recognizes simple commands which allow tlie user *to* change tlie camera nuniber. gain and offset. Figure 4 summarizes the commands which are available when digitizing an image. The user can watch the live image and press **return** when lie is ready to capture the image. Once this is done. tlie digitized image is available for tlie program to use. The following command illustrates tlie use of **iingcp** to digitize an image.

```
imgcp camera:0,255,0,1:matrox mynewimg.img
```

Color digitization is achieved by opening a color image. In the preceding example. the matrox lias been opened as a black-and-n-hite device so the digitized image will be a black-and-white image. In this case. tlie digitized image will consist of **an** average of the three color inputs: red. green aiid blue. In the case where only one input (usually tlie red input) is significant. it is necessary to either digitize a color image and discard two of the files or use tlie **bwmatrox** keyword which accesses the red board of a color matrox as though it were **a** stand-alone matros board. In both cases the live image will be displayed red but the digitized image will be true black-and-white.

Tlie **digitize:** keyword is equivalent to the **camera:** keyword with all tlie default values. It can be abbreviated to **dig:.** For example:

```
imgcp dig:bwmatrox newimg.gif
```

4.17 Files aiid Devices

The BNF grammar presented in section 2 identified certain keywords which represent physical devices. For example. tlie name **display** represents a display device. This means that an image

?	Obtain a list of commands.
yes	Digitize the image (default).
camera	Select the camera number.
setcamera	Set camera gain and offset.

Figure 4: Digitization commands.

file named 'display' cannot be specified by just giving its name. So. how can you access an image file named 'display'?

There is. of course. more than one answer. However. the most direct approach is to use one of tlie image format keywords to clearly identify that tlie name is an image file name and not a keyword. For esample. the following command copies the CMU format image file display to the physical display device.

```
imgcp cmu:display display
```

Because of tlie confusion that could result from using file names which are the same as or similar to keywords. the generalized iniage library requires iniage file names to contain a period. This restriction can be lifted by using one of the image format keywords to clearly identify the iianie as **a** file name.

5 Multi-band Images

Traditionally. an image lias been viewed as a rectangular array of pixel values. The pisel values may be intensity measurements or other scalar values obtained on an integer grid. Multi-band images estend this view by allowing each pixel position to have several different measurements. Each measurement is called a *band* of the image, and tlie multi-band image consists of **a** number of band images. For esample. a standard color image is composed of three bands: the red. green and blue bands.

When programs are operating on multi-band images. tlie generahized image library must know which bands are required by tlie program. This information is usually supplied by the program. so *in most cases you* do *not need to specify the bands of a multi-band image.* In particular. programs often open color images. In such cases it is sufficient to name **a** color device or one of the bands of a color image **on** disk. Some programs. however. require the user to indicate the bands to be used. This is true of general-purpose programs such as **imgcp** which are capable of manipulating arbitrary multi-band images. When using general-purpose programs, the **bands**: keyword may be used to specify the bands of an image.

5.1 Specifying the Bands of a Multi-band Image

The **bands:** keyword may be used to specify tlie bands of a multi-band image. The keyword is followed by a *bands specification*, then a colon and an image name espression. The **bands**: keyword is useful in two situations.

1. Some general-purpose programs such as **iingcp** allow you to specify arbitrary single-band or multi-band images. These programs rely on you to specify tlie bands of a multi-band iniage using the **bands**: keyword.

2. Many programs know in advance the type of multi-band image which they will be dealing nith. The generalized image library has limited facilities for coercing multi-band iniages from one type to another. You can therefore specify a different multi-band image and rely on the library to do the appropriate conversion. For esample. a program which espects to create a stereo image can be made to create a color image instead. The resulting color image is suitable for viewing with red/blue movie glasses.

Imgcp is a general-purpose program which is capable of copying arbitrary multi-band images. When it is opening the input image. it does not know in advance what hands the iniage sliould have'. Instead. it opens the image and then finds out what bands it happens to have. Unless you specify the image bands with a keyword. the generalized image library will assume that tlie image is an ordinary single-band image. For esample. consider the following two commands. Tlie first command copies a single-band image. Tlie image happens to be tlie red band of a color image. The second command copies **a** color image. copying all three bands at once. Notice that the **bands:** keyword is not required on tlie output image because **imgcp** assumes that the output image will have tlie same bands as the input image.

```
 imgcp sunrise.red.img display
imgcp bands:red,green,blue:sunrise.red.img display
```

The bands specification which follows the **bands:** keyword gives tlie names of all the bands contained in the image. The specification consists of one or more *band names* separated by commas. Each band name consists of one or more *attribute names* separated from each other by periods. The attribute names represent actual attributes or characteristics of the image band being referred

'**If you use the -c option to specify a color image then imgcp opens a color image instead of using the general-purpose approach. Similarly. if you use the -s option. imgcp opens stereo images.**

red	Red band of **a** color image.
green	Green band of a color image.
blue	Blue band of **a** color image.
left	Left camera position.
right	Right camera position.
int	Computed intensity feature.
hue	Computed color hue feature.
sat	Coniputed color saturation feature.

Figure 5: Some Standard Attribute Names

to. For example. the attribute name **red** indicates an image taken with a red filter. Similarly. tlie attribute name **left** indicates an image taken from tlie left camera position. Figure 5 lists some standard attribute names and their meanings.

The bands specification of **a** standard color iiiiage was used in the above **imgcp** examples. As we have already seen. such an image has three bands: one band has the **red** attribute. **a** second lias tlie **green** attribute and tlie third has tlie **blue** attribute[5]. Each band lias only a single attribute. being the color. so the band names are **red, green** and **blue** respectively. The complete bands specification is **a** combination of the three band names: **red ,green,blue. A S** another example. consider **a** stereo image. **A** stereo image lias two image bands. corresponding to the left aiid riglit camera positions. The left image has the **left** attribute and tlie right iiiiage lias the **right** attribute. so the band specification is **left ,right.**

Because of tlie multi-band conversion capabilities of tlie generalized iinage library. iiiigcp can be used *to* copy a stereo image into a color iinage. producing a color image suitable for viewing with red/blue movie glasses. The input iinage is **a** stereo iinage consisting of two ini-age files **chair.left.img** and **chair .right.img.** The output color image consists of the files **chair3d.red.img, chair3d.green.img** and **chair3d.blue. img.** In the following example tlie bands specifications have been typed in full. In practice. this is usually unnecessary because there are other keywords which represent commonly-used multi-band specifications such as color aiid stereo images.

```
imgcp bands:left,right:chair.left.img n
     bands:red,green,blue:chair3d.red.img
```

[5]The order of band names in a bands specification is significant. An image with bands **red,green,blue** is very different from an image with bands **blue,green,red.**

As a final, more complex example of a bands specification. consider a color stereo image. Such an image has six bands. Each band has two attributes being the camera position (left or right) and filter color (red. green or blue). The first band of the image the two attributes **left** and **red.** The band name is thus **left .red.** The complete bands specification for a color stereo image is

```
left.red,left.green,left.blue,right.red,right.green,right.blue
```

5.2 Multi-band Images on Disk

hlulti-band images are stored on disk with each band in a separate image file. This provides greater flexibility than if the bands were stored in a single file. However. the generalized image library must be capable of determining the naine of the file in which each band is stored. For this reason. a naming convention is employed. Multi-band images should be named according to this convention in order to facilitate their use with tlie generalized image library.

The generalized image library requires all tlie bands of a multi-band image to be stored in the same directory. When a multi-band image naine is being given. tlie user must specify tlie full name of one of the bands of the image. For example, **chair .left.img** is tlie full name of tlie left hand of a stereo image. The library uses the supplied name and the bands specification to deterniine the full names of the other bands of the image. In the example. the name of the **right** band of tlie stereo iiiiage is found by substituting **right** for **left** in the supplied name. Thus. **chair.right.img** is opened a5 the right band of the stereo image.

In order for the substitution described above to succeed. the following rules must he strictly adhered to when naming multi-band images on disk.

1. The naine of each image file must contain all the attributes which are relevant to that image. For example. the red band of tlie left image of a color stereo image must contain both the attributes **red** and **left.**

2. Attribute names are only substituted after tlie last slash **(/)**in tlie file name. All tlie files belonging to a single multi-band image must therefore reside in a single directory. However. there may be more than one multi-hand iniage in the same directory.

3. The names of corresponding image files must differ only in the attributes present in the names. Thus, **sunset.red.img** and **sunset.green.img** are part of the same color image, but **tree.red.img** and **tree.blue.gif** are not.

4. The order of corresponding attributes must be preserved. Thus. **house .left.red.img** can be part of the same color stereo image as **house .right.green.img.** However. **house.left .red.img** and **house.green.right.img** are not acceptable.

5. Attributes must be clearly delimited by lion-alphanumeric characters. For example. **house.red.img** is acceptable but **housered.img** is not. The one exception to this rule is that attributes may be preceded by numeric characters. This is to provide compatibility with the naming convention which was employed previously.

6. Attributes may not be abbreviated. with the exception of the color attributes. The color attributes may be abbreviated to single letters **r. g** and b. **If** the color attribute is abbreviated in one band of a multi-band image. then it must be abbreviated in all bands.

7. Attributes are arbitrary alpha-numeric strings. However. the standard attribute names should be used whenever possible to avoid confusion.

5.3 Color Images

The **color**: keyword may be used in place of the full bands specification **bands:red,green,blue**: when dealing with color images. The **color:** keyword may be abbreviated to **c:**. For example:

```
imgcp c:sunset.red.img display
```

5.4 Stereo Images

The **stereo:** keyword may be used instead of the full bands specification **bands:left,right:** to indicate a stereo image. The **stereo:** keyword may be abbreviated to st:. For example:

```
imgcp shift:100,200:st:house.left.img newhouse.left.img
```

5.5 Three-Dimensional Viewing with Red/Blue Movie Glasses

As explained above. the generalized image library can convert a stereo image into a color image suitable for viewing with red/blue movie glasses. This conversion takes place whenever a color image is supplied to a program which espects **a** stereo image. The keyword **threed:** can be used to explicitly force the generalized image library to create a **3D** image. The keyword **threed:** can be abbreviated to **3d:**.

```
imgcp stereo:chair.left.img 3d:chair3d.red.img
```

5.6 Black and White Images

Tlie **bw:** keyword can be used to explicitly force an image to be a black-and-white image. This is useful if a program expects a color input image but you wish to give it an ordinary single-band image instead. In such cases. the single-band image is converted to a color image in tlie obvious way; the red. green and blue intensity values at each point are exactly the intensity values in tlie black-and-white image. For example:

```
 imgcp -c bw:house.img house.red.img
```

6 Cursor Positioning

Some programs may use tlie cursor positioning facilities of tlie generalized image library. These facilities allow you to indicate pisel locations in an image by positioning a cursor. Figure 6 indicates tlie keys which may be used to position tlie cursor.

Tlie numeric cursor positioning keys are intended to be used on terminals with **a** numeric keypad. Tlie amount by which the cursor moves cannot be controlled by the shift and **control** keys as for tlie other methods of cursor positioning. Instead. three of tlie numeric keypad keys provide the ability to set tlie amount of cursor movement for the numeric cursor positioning keys. Tlie **minus (-)** key sets the cursor movement to 1 pixel. Tlie **zero** (0) key sets the cursor movement to 8 pixels. wliicli is the default. Tlie **period (.)** key sets the movement to 64 pisels.

When tlie cursor is in the desired position. press **return** and tlie cursor position will be returned to the program. At any point. you may press the **v** (value) key. The library will then report the current cursor position and tlie value(s) of tlie iiiiage at that position. The **t** key may also be used to toggle the cursor on aiid off - pressing **t** when tlie cursor is displayed causes it to disappear and pressing **t** when tlie cursor is not visible causes it to appear. The cursor is always visible after you move it.

If you wish to abort a program during cursor positioning. your usual interrupt key may be used. If you need help. the ? key results in a message which describes the keys in detail. If you press an unrecognized key, then a brief message will be printed on your terminal.

7 Environment

The generalized image library makes use of the following environment variables: **IMDEBUG. IMCURSOR. IMSYNC, IMLOAD. IMPATH** and **IMCREATE. IMDEBUG** is used for debugging and error checking control. **IMCURSOR** is used to control the method of cursor movement in programs which use the GIL cursor facilities. **IMSYNC** is used to control the sync source for display devices. **IMLOAD** is used to select tlie library version to be loaded at run time. **IMPATH** contains a directory path list which is searched

f. **k:** Cursor to the right (forward) one pixel.
F. K: Cursor to the right 8 pixels.
control-F. control-L: Cursor right 64 pixels.
b. h: Cursor to the left (backward) one pixel.
B. H: Cursor to the left 8 pixels.
control-B. control-H: Cursor left 64 pixels.
p. u: Cursor up one pixel.
P. U: Cursor **up** 8 pixels.
control-P. control-U: Cursor **up** 64 pixels.
n. j: Cursor down one pixel.
N. J: Cursor down 8 pixels.
control-N. control-J: Cursor down 64 pixels.
1: Cursor down and to the left.
2: Cursor down.
3: Cursor down and to the right.
4: Cursor left.
6: Cursor right.
7: Cursor up and to the left.
8: Cursor up.
9: Cursor up and to the right.

Figure 6: Cursor movement keys.

when opening image files. **IMCREATE** contains a single directory name which is used as a prefix when image files are being created.

The environment variable **IMDEBUG** may be used to control error checking and debugging in programs built with the generalized image library. The **IMDEBUG** environment variable contains a string of option letters which affect various portions of the library. The following options are available:

1. b: Bounds check. This option strictly checks all image accesses to ensure that they are within the allowable bounds. and reports any violations as an error. Without this option. the result of an out-of-bounds image access is not defined: it may result in a program crash or in corruption of the image.

2. c: Enable CRC check. Programs which are compiled with the option **-DDEBUG=1** have special code generated which can perform a simple cyclic redundancy check on the generalized image structure before invoking the pixel access routines. Normally. the checks are not performed because they are too time consuming. However. if the c option is enabled then the CRC checks are performed.

   Irrespective of whether the c option is present in IMDEBUG. cyclic redundancy checks are performed whenever an image is closed and at certain other strategic points in the generalized image library.

3. **d:** Dump core on error. If the library detects an error which would cause the program to abort. the d option causes a core dump to be produced. This is particularly useful when developing programs which use the library.

4. **e:** Efficiency report. The **e** option causes the generalized image library to report information which may be useful in tracking down suspected efficiency problems. This is particularly useful when doing development work on the library.

5. **i:** Identify. If this option is present. the first attempt to open or create an image will cause the generalized image library to display its version identification. This is useful if you suspect that a program may have been built with an old version of the library.

6. **l:** List active images on error abort. When the program is aborted by the generalized image library, the active generalized image structures are listed. This provides useful information for debugging.

7. n: Network debugging. This option causes the program to use a debugging version of the network server. Useful for development work on the network server.

8. **q**: Quiet mode. This option suppresses informative messages that are otherwise produced by the library. For example. tlie message associated with **IMCREATE** is suppressed if the **q** option is present in **IMDEBUG.**

9. **v**: Value check. The **v** option enforces strict pixel value checking. Every operation which fetches or stores pixels is checked to ensure that the pixel values are in tlie valid range. The first range error is reported but execution continues. All out-of-range pixels are truncated to the nearest extreme value of the range. *Unimplemented.*

10. **z**: Wizard debugging information. The **z** option is used by developers to display debugging information. This information will probably be unintelligible *to* users.

11. F: Fake forks. When debugging GIL operations which involve forking. it is sometimes useful to disable the actual fork operations. When tlie option F is set. tlie GIL does not execute tlie fork operation but continues processing as though it were the *child* process. Useful for wizards only.

12. **M**: Macro expansion debugging. The M option is used by developers to debug GIL macro translation. This information will probably be unintelligible to users.

Within the **csh** shell. the **setenv** command may be used to set the **IMDEBUG** environment variable. For example. tlie following command sequence runs tlie program **myprog** with strict bounds checking on image access and with core to be dumped if **an** error abort occurs.

```
% setenv IMDEBUC bd
% myprog in.img out.img
```

Once **IMDEBUG** has been set. the options remain in effect until it is reset. To cancel all options. **IMDEBUG** may be set to the empty string as follons.

```
% setenv IMDEBUG ''
```

The **IMCURSOR** environment variable is used to select alternate methods of moving tlie GIL cursor. By default. cursor motion is based on keyboard commands as explained below. If the environment variable **IMCURSOR** is set to **x** (or **X**) then the GIL **will** use tlie mouse under tlie X window system to move the GIL cursor.

The **IMPATH** environment variable contains a path list of directories which are searched for **an** image file that is being opened. The path list is not searched if tlie image file name is **an** *absolute* path, that is if it commences with **a** slash (/). The path list consists of any number of directory names separated from each other by colons. Normally. the first directory in the path is dot (.) which causes the library to look for the image in the current directory.

As an example. consider a user fred who has many of his images stored in the directory **/visi/fred** and some additional images in the directory **/visi/fred/extras**. The following csh shell command could be used to set the image library's path to search both of these directories after the current directory.

```
% setenv IMPATH '.:/visi/fred:/visi/fred/extras'
```

After executing the above command to establish his path. the user fred can omit the full path names of image files which reside in either of the directories **/visi/fred** or **/visi/fred/extras**. So, lie can copy the image **/visi/fred/tree.img** to the display device with the following command (assuming that there is **no** file **tree.img** in his current directory).

```
imgcp tree.img display
```

He can also copy a file called **/visi/fred/experiment/tree.img** to tlie display device with the folloning command.

```
imgcp experiment/tree.img display
```

The path searching strategy applies only when existing images are being named. When a new image is being created. the name must be specified in full unless **IMCREATE** has been set. For instance. if fred wished to copy liis image **/visi/fred/house.img** into GIF format in the same directory. he would use tlie following command.

```
imgcp house.img /visi/fred/house.gif
```

The **IMCREATE** environment variable contains a single directory name. When an image file is being created. the generalized image library prefixes the file name with tlie contents of tlie **IMCREATE** environment variable. This is not done if the file name is an absolute path. that is if it commences with a slash (/). **IMCREATE** is also not used if the image file name commences with a period (.) as it is then assumed to be explicitly named relative to tlie current directory. When **IMCREATE** is used. the library reports the full name of the image file it is creating. The message may be suppressed by the **q** option in **IMDEBUG.**

As an example. consider again the user fred who likes to store his images in **/visi/fred**. The following csh command could be used to set tlie **IMCREATE** environment variable **so** that the library would create images in **/visi/fred** by default.

```
% setenv IMCREATE '/visi/fred'
```

After executing tlie command to establish **IMCREATE.** the user **fred** can omit the full path name when creating image files in tlie directory **/visi/fred**. He can also abbreviate tlie path name of images created in directories beneath **/visi/fred**. For example. lie can copy tlie image **/visi/fred/house.img** into **GIF** format in the same directory with tlie following command.

```
imgcp house.img house.gif
```

To copy his image **/visi/fred/extras/car.img** iiito GIF format. he could use the following command.

```
imgcp car.img extras/car.gif
```

The **IMSYNC** environment variable is provided to give tlie user esplicit control ovei the sync signal used by display devices. Display devices such as the hlatros are capable of synchronizing tlieii output signals eithei to an external sync signal or to an internally generated signal. Since tlie external signal is often unstable. the library normally uses an internal signal escept wlien images are being digitized. The following command may he used to set tlie **IMSYNC** environment variable and foice the library to always use the esternal sync signal.

```
% setenv IMSYNC external
```

To return to the default mode of interiially generated sync signal. use tlie folloning command.

```
% setenv IMSYNC internal
```

Tlie **IMLOAD** environment variable is provides the user with explicit control over tlie load-at-run-time library. Normally. tlie version of tlie library which is loaded at run time is determined by the version of **libldgimage.a** with which the program was linked. However. tlie **IMLOAD** environment variable **may** be set to override this default. If the **IMLOAD** environment variable is set, then tlie coiiteiits is taken as the naine of the load-at-run-time file. For example. tlie folloning command causes the experimental library to be used.

```
% setenv IMLOAD /usr/vision/experimental/lib/libgimage.ld
% cursor
Usage:  cursor input-image(s) [-d display] [-o output]
  -c:  Use color.
  -o:  Output label points image.
For help during cursor positioning, type '?'
```

Figure 7: The Syntax Summary for Cursor.

Additional debugging information can be obtained by prefixing the file name with a hyphen. i.e.

```
% setenv IMLOAD -/usr/vision/experimental/lib/libgimage.ld
```

8 Some Useful Utilities

This section describes some useful utility programs which have been implemented using the generalized image library. These include program for copying images. printing out the header information of an image and interacting with an image via the cursor positioning facilities of the library.

If any of the programs is invoked without any arguments, it displays **a** brief description of how it is used. Once you are familiar with the program, this description will remind you of the exact syntas of the command and the switch names. For example. figure 7 shows the syntax summary for the **cursor** program.

8.1 Header

The **header** program displays the header information of an image. The command syntax is as follows.

header [-p] *image(s)*

Header can be used with any generalized image and will display the image type. bounds and pixel characteristics. Additional information describing the paging characteristics is printed for CMU format images. The command snitch **-p** causes **header to** display the property list of the image(s). The property list is used to store descriptive information.

Figure 8 shows two examples of using the **header** program. In the first example, the characteristics of a CMU format image are displayed along with its property list. In the second example, the header information for the default display device is shown.

8.2 Copying Images

One of the most useful utilities is the **imgcp** program. The examples throughout this document generally involve copying images from one format to another, so **imgcp** is used. **Imgcp** combines with the naming conventions of the generalized image library to provide a powerful tool for creating and displaying images, and converting them from one data format to another. Command line switches provide additional features. The command syntax follows.

```
imgcp [-switches] input-image output-image
```

In its simplest form, **imgcp** can be used to copy any generalized image to any other generalized image. The input and output name expressions can employ the generalized image library keywords described previously to modify to the image. If the output image refers to an existing disk file, it will be destroyed without any warning. If the output image is a display device, **imgcp** will only copy that portion of the input image which can be accommodated on the display.

The command switch `-c` indicates that the input and output images are color images. The command switch `-s` indicates that the input and output images are stereo images. Combining the two switches indicates a color stereo image.

The command switch `-r` causes pixel values to be converted from the input image pixel value range to the output image pixel value range. This is useful for displaying binary images, as in the following example.

```
imgcp -r sunthresh.img display
```

Linear transformations may also be invoked by the generalized image library naming syntax or by the `-m` and `-a` command switches. The `-m` switch is used to specify a multiplier and `-a` indicates a constant to be added after the multiplication. For example, the following command multiplies `moon.img` by three and adds 128.

```
imgcp -m3 -a128 moon.img display
```

When the pixel values of the input image are unknown, a reasonable display may be produced by using the `-n` command switch. The `-n` switch obtains a random sample of the input image and

```
 % header -p pitt.img

/usr/vision/images/pitt.img:
        Image format:  CMU
        351 rows (250:600).  381 columns (190:570)
        8 bits/pixel
        Pixel type:  'unsigned'
        Pixel range:  0:255
        132 pages:  11 down, 12 across.
            Each page holds 32 rows, 32 columns of pixels.
        Total pixel storage space = 139264 bytes

Properties       Values
----------       ------
pixel type       unsigned
pixel range      0:255
description      A scene of Pittsburgh, PA
% header display
display:
        Image format:  net
        480 rows (0:479).  512 columns (0:511)
        8 bits/pixel
        Pixel type:  'unsigned'
        Pixel range:  0:255
```

Figure 8: Examples of the Header Program

estimates the mean and standard deviation of the pisels. It then computes a linear transformation to the desired mean and standard deviation which are given as arguments to tlie **-n** switch. For example:

```
imgcp -n128,32 weird.img display
```

The -o command switch **can** be used to produce complex displays by overlaying several input images. When tlie -o option is used. **imgcp** opens tlie output image and writes the input iniage to it instead of creating the output image from scratch. It is thus possible to use the **shift:** and **crop:** keywords to position input iniages on the output.

Tlie -z command switch computes an automatic magnification and shift of the input image to fit into the output image. For esample. the following coinmand produces a reasonable display of tlie color image **myscene.red.img.** magnifying or reducing it as necessary to fit on the display.

```
imgcp -cz myscene.red.img display
```

The -S command switch is used to copy iniage sequences. When this switch is used. iingcp opens the input and output images as sequences rather than as individual images. It then proceeds to interactively copy iniages from the input sequence to the output sequence. If the input sequence is a display device capable of digitization then input images will be digitized and copied to the output iniage sequence. For esample. the following command may be used *to* digitize **a** sequence of images from live video input.

```
imgcp -S matrox newdata.seq1.img
```

8.3 Cursor

Tlie **cursor** program provides cursor-based interaction with generalized images. It allows image values to be interrogated interactively and provides the ability to record selected co-ordinates in an output image. The command syntax is as follows.

cursor *input(s)* **[-d** *display*] **[-o** *output]*

In its simplest form. the **cursor** command may be used to interrogate the pisel values of an image under tlie control of the generalized image library's cursor package. Cursor movement is described in section 6. The input image will be copied to your machine's default **display** device. If the image is larger than the screen. automatic scrolling/panning will be performed as necessary. For esample. the following command allows the image **large.img** to be viewed on tlie default display device.

```
cursor large.img
```

More than one input iinage may be specified. In that case. the cursor program will display the pixel values of **all** tlie input images whenever **return** is pressed. Only the first input image. however. will be displayed on tlie screen. For example. suppose that **house.img** is a house scene and **houseseg.img** is a segmentation labelled image. then tlie following command can be used to query the region labels at points in the image.

```
cursor house.img houseseg.img
```

The **-d** switch may be used to override the default display. This is useful not only to specify an alterative display device but also to modify the way in which the display is used. In particular. the display can be zoomed to obtain higher resolution for more accurate positioning of the cursor. The following command example uses a zoom factor of four to enable tlie cursor to be positioned to the nearest pixel accurately.

```
cursor tree.img -d zoom:4:display
```

The **cursor** program may be used to record positional data in an output image. The -o switch specifies the output **image.** The image will be created and filled with zeroes if it does not already exist. When the -o option is used. pressing **return** causes a pixel value to be stored in the output iniage. Pixel locations which have been selected will be marked with a white spot on the display. The stored value is specified interactively when **cursor** is first esecuted and may be changed at any time simply by pressing **return** twice at the same pixel location. You will then be prompted for tlie new value. To exit from the **cursor** program when using tlie -o option it is necessary to press **return** twice and select minus one[6]. Exiting the program by means of your usual interrupt character will cause the stored values to be lost. This is a bug which remains to be corrected.

For example. the following coninland could be used to locate positional features such as windows and corners in a house scene. The display is zoomed to facilitate accurate cursor positioning.

```
cursor house.img -d zoom:4:display -o housefeat.img
```

‘This **is a hack which should** be **fixed.**

9 Paying the Piper

If you think that all this flexibility makes programs big and slow, you may have a point. The speed cost of using the flexibility of the generalized image library varies, but is usually low. The generalized image library provides faster access to CMU format images than was possible with the old library. The facility for loading the library at run-time makes programs smaller but involves **a** fixed overhead of loading the entire library every time a program is started.