

# IFS Release 6.0

## Installation instructions

### Macintosh (note: this is Mac OS-X. IFS does not run under mac 9.1)

- 1) Copy the contents of the CDROM to wherever on your hard disk you want the files to reside. I recommend /Users/Shared/ifsv6.0
- 2) Under the directory ifsv6.0, you will find the following directories:
  - doc*: the documentation directory
  - Images*: directory of ifs images
  - Linux*: executables and libraries for linux.
  - MacX*: executables and libraries for Macintosh
  - Source*: include files
  - Sun*: executables and libraries for SUN Solaris
  - Win32*: executables and libraries for windows 98, 2000, and XP.

While not necessary, we recommend you delete Linux, Sun, and Win32.

### Windows

- 1) Copy the contents of the CDROM to wherever on your hard disk you want the files to reside. I recommend c:\ifsv6.0
- 2) Under the directory ifsv6.0, you will find the following directories:
  - doc*: the documentation directory
  - Images*: directory of ifs images
  - Linux*: executables and libraries for linux.
  - MacX*: executables and libraries for Macintosh
  - Source*: include files
  - Sun*: executables and libraries for SUN Solaris
  - Win32*: executables and libraries for windows 98, 2000, and XP.

While not necessary, we recommend you delete Linux, Sun, and MacX.

### Linux (note: this has been built on Redhat 7.2 )

- 1) Copy the contents of the CDROM to wherever on your hard disk you want the files to reside. We recommend YourHomeDirectory/ifsv6.0
- 2) Under the directory ifsv6.0, you will find the following directories:
  - doc*: the documentation directory
  - Images*: directory of ifs images
  - Linux*: executables and libraries for linux.

*MacX*: executables and libraries for Macintosh

*Source*: include files

*Sun*: executables and libraries for SUN Solaris

*Win32*: executables and libraries for windows 98, 2000, and XP.

3) In case you have a different version of Linux, please let me know and I can provide you with the source for IFS and wxIFSView and you should be able to rebuild it without any problems.

While not necessary, we recommend you delete MacX, Sun, and Win32.

## **SUN (note: IFS is built using Solaris 5.8)**

1) Copy the contents of the CDROM to wherever on your hard disk you want the files to reside. We recommend YourHomeDirectory/ifsv6.0

2) Under the directory ifsv6.0, you will find the following directories:

*doc*: the documentation directory

*Images*: directory of ifs images

*Linux*: executables and libraries for linux.

*MacX*: executables and libraries for Macintosh

*Source*: include files

*Sun*: executables and libraries for SUN Solaris

*Win32*: executables and libraries for windows 98, 2000, and XP.

3) In case you have a different version of Unix, please let me know and I can provide you with the source for IFS and wxIFSView and you should be able to rebuild it without any problems.

While not necessary, we recommend you delete MacX, Linux, and Win32.

## **Running IFS programs**

*UPDATES are available at <http://www.ece.ncsu.edu/imaging>*

IFS programs are designed to run from the command line. With the exception of the image viewing programs, you cannot click on an ifs application and expect it to run. Platform-specific details follow:

### **Macintosh**

Run ifs programs from a Unix terminal window. You will probably want to set your path to include the directory where you stored the ifs programs, for example you might want to set your path to

```
./Users/Shared/bin:/Users/Shared/ifsv6.0/MacX/ifsbin:/bin:/sbin:/usr/bin:/usr/sbin
```

or something similar, as long as it includes the directory where you stored the executable files.

On the Macintosh, no environment variables need to be set.

## **Windows**

Run ifs programs from a command prompt window. You will probably want to set your path to include the directory where you stored the ifs programs, for example you might want to set your path by using the command

```
set PATH=%PATH%;c:\ifsv6.0\Win32\ifsbin;
```

or something similar, as long as it includes the directory where you stored the executable files.

On the PC, no environment variables need to be set to run programs, but you will need to set some environment variable to build programs.

## **Linux**

Run ifs programs from a terminal window. You will probably want to set your path to include the directory where you stored the ifs programs, for example you might want to set your path to  
/bin:YourHomeDirectory/bin:/usr/local/bin:/usr/bin/X11:YourHomeDirectory/ifsv6.0/Linux/ifs-  
bin:

or something similar, as long as it includes the directory where you stored the executable files.

Under Linux, no environment variables need to be set.

## **SUN Solaris**

Run ifs programs from a Unix terminal window. You will probably want to set your path to include the directory where you stored the ifs programs, for example you might want to set your path to

```
./bin:/usr/local/bin:/usr/local/X11/bin:YourHomeDirectory/ifsv6.0/Sun/ifsbin
```

or something similar, as long as it includes the directory where you stored the executable files.

On the SUN, no environment variables need to be set.

# **Building IFS Applications**

## **Macintosh**

To write programs and build applications, you will need a C compiler, editor, linker, etc. If you do not already have these capabilities, you can obtain them from the Apple Developer site.

Currently, the web address for this download is <http://developer.apple.com/tools/macosx-tools.html>, however, like every other web site, this may change without notice. Your best bet is to go to the main Apple web site, go to developer, and search from there.

## Using Project Builder (Xcode)

Project builder (recently renamed to Xcode) is a convenient means for developing software projects using a relatively nice editor, the gcc compiler, and the gdb debugger. For your convenience, we have included a Project Builder file in the ifsv6.0/MacX directory. Included is the source for the ifs application ~~equipped~~. You can duplicate that and simply modify the source file to create new projects. To run Project Builder, go to the developer director which you (hopefully) downloaded or ordered from Apple. However, Project Builder is not required in order to use IFS. All you need is a C compiler, editor and linker.

The ifs libraries were built using gcc release 3.1.

## Windows

To write programs and build applications, you will need a C compiler, editor, linker, etc. If you do not already have these capabilities, you can obtain them by either ordering Microsoft Visual C++, or obtaining some other tool set such as lcc-win32 or the Win32 port of gcc.

## Using Visual Studio

Visual Studio is a convenient means for developing software projects using a relatively nice editor, compiler, and debugger. To write your own IFS-based program simply create a new project and choose a ~~to~~ console-based application ~~for~~ starters. Once you have your source code all written up, simply go to the project settings and include ifs.lib in the Object/Library Modules textbox in under the Link tab. A sample project has been included for your convenience.

The ifs libraries were built using Visual C++ 6.0; however using the same library with any other compiler should not pose any problems.

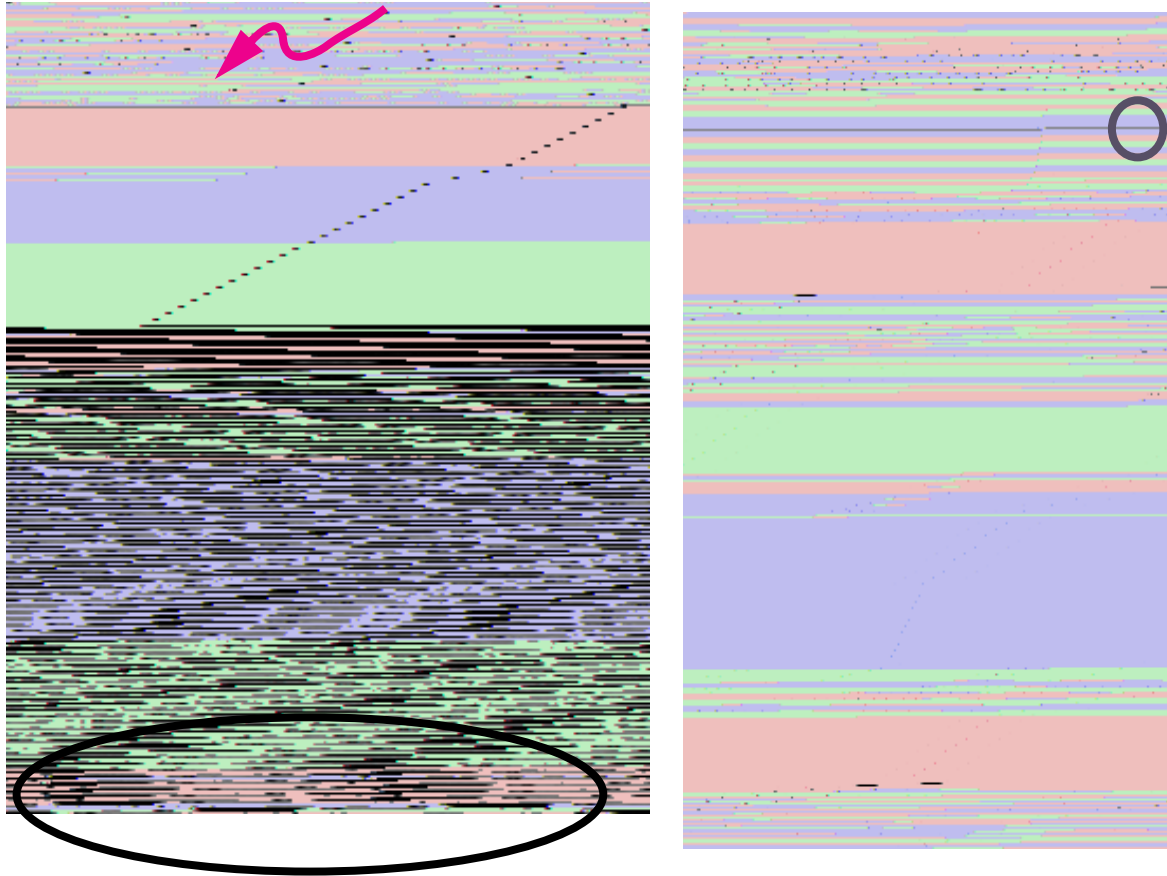
## Viewing IFS images

On all platforms, two viewers are provided, one named ~~wxIFSView~~ and one which is built on top of Java, named ~~ifs View.jar~~. ~~The~~ The Java version is older, slower, and lacks features.

wxIFSView is built using the wxWindows toolkit (release 2.4.0).

wxIFSView has a base window that is used to perform file-open routines. Each new image is opened in a separate window and has its own associated control panel to perform simple opera-

tions on the image. Below is a sample screenshot showing a sample IFS image and showing the status bar (circled in green) and the control panel for the image. The control panel is invoked by clicking on the control icon (marked with a magenta arrow).



The control panel is created dynamically based upon the content of the image. The control panel contents provide you with means to modify the colormap, the zoom factor (note that the factor displayed in blue circle is actually ten times the zoom factor), window/level settings, animation rate (if the image has multiple frames), rendering sequence of the red, green and blue channels (only if the image is multi-frame). The autoscale option needs to be used with caution - if the data has dynamic range more than that of an unsigned char, the data will be truncated (as there is overflow).

The status bar of the image provides information about the image. There are three segments to the status bar. The first segment shows the dimensions of the image along with the frame currently being displayed (if it is -1, you are seeing an RGB rendering of the image) and the zoom factor currently being used. The second and third segments are updated on a click inside the image at any location. The second segment shows the actual image value at the click location, (in the

example above, the brightness is 2) and the color it is mapped to for display purposes (in this example, 1 unit of red, 1 unit green, and 1 unit blue) and the third frame shows the location of the click.

Above the image, you will see a button with three colored squares. This is the ~~render~~ **render as color** option, which will select three frames from the (must be 3D) image and display them as the red, green, and blue components of a full color image. Otherwise, only one frame is displayed. You can select which three frames are used with the control panel.

A second button above the image shows a sequence of squares. This is the ~~animate~~ **animate** button. If you are displaying a 3D image, you can display it as a ~~movie~~ **movie** by pressing this button. The speed of playback is set using the control panel.

The File Menu for each image provides a means saving the data in any desired IFS file format or exporting the data being displayed into any of BMP, JPEG, PNG or TIFF file formats. This may be convenient for report writing purposes.

The Edit Menu provides you with the option to display a multi-frame image as an RGB rendering, to set the animation of a multiframe on or off, display the control panel for the image, use global scaling (choose min and max for display renderings from all frames) or turn on or off the autoscale routine for each frame.

The About menu provides a convenient means of determining information about the IFS file such as image size, data type, and statistics. If the annotation file for the IFS file is in the same location as the IFS image, this information is also displayed.

wxIFSView release 2.x also supports several image processing operations.

Region segmentation

You can obtain a histogram

## **Binding applications to file types**

Macintosh and Windows platforms support file-associations. Below are brief descriptions on how to configure this.

### **Macintosh**

To set a particular type of file to be opened by a particular application, click one on the file to select it. then go to the finder FILE pulldown and select GET INFO. On the info panel is another selection called ~~open with~~ **open with** which you can set to open with a particular application, such as wxIFSView.

### **Windows**

To set a particular type of file to be opened by a particular application, right click on the icon of an IFS Image and select **OPEN**. The program (wxIFSWin32) can be set as the default handler of IFS file types.

**Note:**

wxIFSView has some peculiarities: On the Macintosh, when you open it, the window size will be 1x1. You will need to grab the window and stretch it to see the image. The Win32, GTK and Motif versions do not have this problem. We expect that this problems will be cleared up by the time you read this, so check the web site <http://www.ece.ncsu.edu/imaging>, for updates.

## IFS Capabilities

External variables, Warning messages, Warnings

To set the status at which you wish IFS to return warnings, set the external variable IFSSLV to IFS\_QUIET, IFS\_WARN, IFS\_FATAL (IFS\_WARN is default)

example:

```
extern int IFSSLV;
```

```
IFSSLV = IFS_QUIET;
```

Full documentation in ifsmanual.pdf

## Subroutines

These subroutines are in libifs.a, libiptools.a, or libflip.a. The subroutines whose name starts with **ifc** are designed to run only on floating point ifs images. They are optimized for speed. We recommend that students usually convert their images to floating point (using any2any) and write programs using the flip subroutines, if speed is a consideration.

### **ifsalc**

Allocate Storage:

Full documentation in ifsmanual.pdf

### **ifscigp, ifscfgp, ifscigp3d, ifscfgp3d**

Access complex image data: ifscigp or ifscfgp, (use ifscigp3d or ifscfgp3d if image is 3D)

Full documentation in ifsmanual.pdf

### **ifscipp, ifscfpp, ifscipp3d, ifscfpp3d**

Store complex data: ifscipp or ifscfpp, (use ifscipp3d or ifscfpp3d if image is 3D)

Full documentation in ifsmanual.pdf

## **ifscreate**

Create an ifs image. Constructs a data structure, allocates memory.

Full documentation in ifsmanual.pdf

## **ifsdimen, ifssiz**

Return the dimensions of an ifs image. ifsdimen is called once for each dimension. ifssiz allocates space for storage and returns a list of sizes of all the dimensions. Caution. if you call ifssiz in a loop, you will allocate more and more memory!

Full documentation in ifsmanual.pdf

## **drand48**

Returns a random double uniformly distributed between 0.0 and 1.0

## **gaussrand**

Returns a random double which is Gaussian distributed with mean 0.0 and variance 1.0.

## **ifsexwin, ifsexwin3d**

Extract a window from an image. Use ifsexwin3d if window is 3d

Full documentation in ifsmanual.pdf

## **ifsfgp, ifsigp, ifsfgp3d, ifsigp3d**

Access ifs image. Get pixel values

Full documentation in ifsmanual.pdf

## **ifsfpp, ifsipp, ifsfpp3d, ifsipp3d**

Store data in an ifs image

Access an ifs image; converts data type of argument to data type of image

Full documentation in ifsmanual.pdf

## **ifsfree**

Free (unallocate) ifs image. NOTE. do not use the usual C free function. Use this one instead.

Full documentation in ifsmanual.pdf

## **ifsGetImg, ifspin**

Open a file and read an image. ifspin is somewhat easier to use; ifsGetImg somewhat more flexible.

Full documentation in ifsmanual.pdf

## **ifsPutImg, ifspot, ifsWrImg**

IFS Picture Output

Write an ifs image to disk. ifspot is somewhat easier to use, ifsPutImg is somewhat more flexible.

Full documentation in ifsmanual.pdf

Error codes



IFS returns a variety of error codes for conditions like bad headers, missing file, etc.  
Full documentation in ifsmanual.pdf

## Image Processing Tools Subroutines

The following subroutines operate on images in memory.

### **ifsadd, fladds, fladdv,**

Add two images.

Full documentation in ifsmanual.pdf and ifsflip.pdf

### **ifsany2any**

Convert an ifs image of any data type to an ifs image of a different data type.

Usage: int ifsany2any(IFSIMG inimage, IFSIMG outimage);

Will not work on complex data types.

Returns negative values if errors occur.

### **flcp**

copy an image. Documentation in ifsflip.pdf

### **flcp**

1. Clip (limit) image brightness

### **ifscfft2d**

Compute two dimensional fast Fourier transform

Full documentation in ifsmanual.pdf

### **ifsc2imag**

Extract imaginary part of a complex image

Full documentation in ifsmanual.pdf

### **ifsc2mag**

Extract magnitude of a complex image

Full documentation in ifsmanual.pdf

### **ifsc2phase**

Extract phase of a complex image

Full documentation in ifsmanual.pdf

### **ifsc2real**

Extract real part of a complex image

Full documentation in ifsmanual.pdf

### **fldivv, fldivs**

Divide images

Documentation in ifsflip.pdf

### **fldx, fldx\_back, fldx\_forw, fldxx, fldxy, fldxz, fldy, fldy\_back, fldy\_forw, fldyy, fldyz, fldz, fldzz,**

Differentiate images

Documentation in ifsflip.pdf

### **fexp**

Exponentiate an image

Documentation in ifsflip.pdf

### **fln**

Take logarithms of images, pixel-by-pixel

Documentation in ifsflip.pdf

### **ifsmult, flmults, flmultv,**

Multiply two ifs images

Full documentation in ifsmanual.pdf and ifsflip.pdf

### **fneg**

Negate pixel values

Documentation in ifsflip.pdf

### **flnorm**

Returns the 2-norm (square root of the sum of squared pixels)

Documentation in ifsflip.pdf

### **flone\_border, flpad, flplanar, flzero\_border**

Set border values

Documentation in ifsflip.pdf

### **ifsrecip, flrec**

Take the reciprocal of an ifs image.

Full documentation in ifsmanual.pdf and in ifsflip.pdf

### **flshx, flshy, flshxy**

Shift an image

Documentation in ifsflip.pdf

### **flsq, flsqrt**

Square and take square root of pixels  
Documentation in ifsfliip.pdf

### **ifssub, flsubs, flsubv**

Subtract two images  
Full documentation in ifsmanual.pdf and in ifsfliip.pdf

### **flthresh**

Threshold an image  
Documentation in ifsfliip.pdf

## **Programs**

### **Utility programs**

These programs are for the most part, simply ~~mainly~~ wrapped around some of the standard sub-routines documented in earlier chapters. These programs are only documented briefly here, since the operation is generally obvious.

Generally, on-line help for any program can be obtained by simply starting that program up, but providing it an incorrect number of arguments.

### **add**

add two ifs images, point by point  
usage: add infile1 infile2 outfile type  
options for data type of output file are:  
b (unsigned byte)  
s (short)  
i (int)  
f (float)  
c (complex)

### **addhdr, rmvhdr**

adds an IFS header to a raw data, producing an ifs image. ~~rmvhdr~~ is the reverse function.

### **any2any**

Converts an **ifs** image of any data type to an ifs image of any other data type. Also does some other output conversions. NOTE this program does not read IN images of other image formats like jpeg.

USAGE: any2any inputfile outputfile datatype [frame]

For output to an ifs file, any data type EXCEPT complex may be used , to convert to non-ifs data types:

datatype is jpg|jpeg|JPG|JPEG for a JPEG File

datatype is tif|tiff|TIF|TIFF for a TIFF File

datatype is bmp|BMP for a Windows Bitmap file

datatype is ppm|PPM for a Portable Pixel Map

If image is multiframe, for TIFF BMP and PPM, the user is prompted for the frame number.

If frame is specified it uses it, defaults to zero

### **atoi**

Converts an ascii input to ifs. Input is to be in the format produced by itoa using the -v switch. The -v switch on itoa adds two lines at the beginning of which specifies the size and data type. (use itoa -h for more details)

Usage: atoi inasciifile outifsfile

### **c2imag**

Take imaginary part of an ifs image, point by point

usage: c2imag infile1 outfile

### **c2mag**

Take magnitude of an ifs image, point by point

usage: c2mag infile1 outfile

### **c2phase**

Take phase of an ifs image, point by point

usage: c2phase infile1 outfile

### **c2real**

Take real part of an ifs image, point by point

usage: c2real infile1 outfile

### **compmag**

produces an ifs image (type float) equal to the log of the square of the magnitude of a complex image

### **dicom2ifs**

Reads a Dicom (medical imaging standard) image and converts it to ifs. There are many variations of the dicom standard. dicom2ifs supports most, but not them all of them..

Usage: dicom2ifs -i inputdicomimage.anyextension -o filename.ifs

### **exwin3d**

extracts a 3-D sub-image from a 3-D image

Usage: exwin3d <inputfile> <outputfile> <f0> <r0> <c0> <f1> <r1> <c1>

where f0,r0,c0 and f1,r1,c1 are the frame row and column

coordinates of the inclusive corners of the area to extract.

## **ifs2avi**

Converts a 3-D ifs image into the Microsoft avi standard. No compression is used. Note: some versions of windows media player do not support avi. We recommend realplayer on the PC, Quacktime on any platform, or xanim on the Sun.

Usage: ifs2avi -i input.ifs -o output.avi -m mapfile.map

the mapfile option allows the user to specify an interpretation of the brightness values as colors. If no colormap specified, the default gray scale is used

The colormap file contains 256 ascii lines with three (RGB) values per line.

If the colormap file name is a single digit, as defined below, then a default map will be used:

- m 0 grey scale
- m 1 inverse grey scale
- m 2 bronson
- m 3 hot metal
- m 4 thermal
- m 5 log
- m 6 random

## **info**

types information about an image, such as data type, number of dimensions, rows, cols, etc.

Usage: info file [file2 file3...]

Output is # of dimensions, data type, length of each dimension in order of ascending rank, and file name.

## **itoa**

prints an IFS 2D image in ascii format.

ifs\_to\_ascii: Prints all or part of an IFS image in ascii format

Usage: ifs\_to\_ascii [options] image-filename

If the filename is omitted, the image input is read from stdin.

there are many options. type itoa -h

## **mult**

multiply two ifs images, point by point

usage: mult infile1 infile2 outfile type

options for data type of output file are:

b (unsigned byte)

s (short)

i (int)

f (float)

c (complex)

## **profile**

Take a cross section of an IFS 2D image. Output is in standard plot format (to stdout).

## **prthdr**

Print the header structure for an IFS image (in human readable format).

Usage: prthdr file [file2 file3...]

## **recip**

take reciprocal of an ifs image, point by point

usage: recip infile1 outfile

## **rmvhdr**

Remove the header from an IFS image to yield a raw data

This program reads an IFS format file and writes the image back out,

minus its IFS header. I.E., raw data only. It will tell you the

number of dimensions, length of each dimension, and the data format

The program will prompt for the names of input & output files.

## **spin**

Construct a file to run to make a 3-d view from a range image

usage: spin [switches]

switches are

-n x     where x is the number of degrees per each step  
         default: 10

-p x     where x is the amount of pitch in degrees  
         default: 0

-y x     where x is the amount of yaw in degrees  
         default: 0

-i infilename     where infilename is an ifs range image

-o outfilename     where outfilename is an ifs range image

-r x     where x is the final offset in rows  
         default: number of input rows / 2

-c x     where x is the final offset in columns  
         default: number of input cols / 2

-C x     number of columns in output image  
         default: number of input cols\*2

-R x     number of rows in output image  
         default: number of input rows\*2

-v x     where if x is a zero, data will not be projected below the image. If x is a one, it will be  
         default: 0

## **squp**

**squareup** **image**. That is, resamples image in one or both directions. to produce cubical pixels or voxels -- generally more useful in 3D applications in which one dimension is sampled differently from the other two.

usage:squareup -i inimage -o outimg -c colspacing  
-r rowspacing -f framespacing [-z zoomamount]

## **stats**

Computes the statistics of an image: max, min, mean, variance, etc.

Usage: stats inputfile

## **subsample**

subsamples an arbitrary ifs image to be of a specified size

Usage: subsample inimage outimage dimensionsinx dimensionsiny

## **sub**

subtract two ifs images, point by point

usage: sub infile1 infile2 outfile type

options for data type of output file are:

b (unsigned byte)

s (short)

i (int)

f (float)

c (complex)

## **viewpoint**

Computes how a range image would appear if viewed from an arbitrary viewing position and orientation (consider using spin)

Usage: viewpoint [options]

Options are:

- i -- Input File name (You MUST specify an input file name)
- o -- Output File name
- r -- Display messages as each row in the image is processed
- v -- project volume beneath each pixel.
- z -- treat pixels with z=0 as background (ignore them) --  
i.e, don't project the baseplane
- m -- project volume only to nearest non-zero neighbor  
i.e, don't project volumes all the way to baseplane  
This option implies -v option.
- x -- Project pixels as boxes rather than flat plates; only  
meaningful if -v and -m NOT used, since -v/-m cause  
pixels to be projected as columns anyway.
- q -- Do not prompt the user for input. However, accept input  
as usual. This option is only useful if stdin is a file
- s -- Turn off autosize/autoshift operation. By default, the  
output image will be made large enough to include the

entire input image after rotation, and the rotated image will be shifted so that no part of it is clipped off. If -s is specified, then this won't happen and you will be asked to specify the desired size of the output image.

the easiest way to specify the transform of viewpoint is

rpyt rollindegrees pitchindegrees yawindegrees movex movey movez end. examplerpyp 10 20 30 5 4 3 end

## **vidscale**

video scale an ifs image -- produces an eight bit unsigned image with minimum of zero and maximum of 255. Useful for lots of ports to other format.

usage: vidscale input output

## **window**

This program extracts a window from an ifs image. The resultant output image is of the same data type as the input. Call:

window input output xleft ylower xright yupper

input and output are two dimensional ifs image files. output will be created by this program. xleft is the index of the left-most column of the input image which should be in the window. ylower is the index of the lowest-index row of the desired window.

xright and yupper are the other extremes. NOTE: yupper must be greater than ylower. Thus, upper and lower correspond to indices, not to a top-bottom relation on a display screen.

# **Image Synthesis programs**

## **qsyn**

Synthesize range images

using quadric surfaces

Full documentation in ifsmanual.pdf

## **3dsyn**

Synthesize volume (density) images

using quadric surfaces

This program is only available for the SUN.

Full documentation in ifsmanual.pdf

## **matte**



Synthesize luminance images

Input is a range image and a set of light source locations and brightnesses

Full documentation in ifsmmanual.pdf

### **Tomosim**

Simulate a 3d beam tomographic sensor

Can simulate either parallel beam or cone beam.

Only available for the SUN at this time

## **Optimization Programs**

### **Interopt**

Interopt is an interactive program that prompts the user for information about his/her optimization problem, writes a program to solve that problem, compiles it, and runs it. Interopt runs on Unix and Unix-like platforms (MacIntosh under Os-X, Linux, and Solaris). It is distributed as a.zip file in the ifsbin directory. Detailed documentation is available in interopt.pdf.