Vis5D Documentation

Vis5D Documentation

Table of Contents

Chapter 1. Overview of Vis5D

Vis5D is a software system that can be used to visualize both gridded data and irregularly located data. Sources for this data can come from numerical weather models, surface observations and other similar sources. Vis5D can work on data in the form of a five-dimensional rectangle. That is, the data are real numbers at each point of a "grid" which spans three space dimensions, one time dimension and a dimension for enumerating multiple physical variables. Of course, Vis5D works perfectly well on data sets with only one variable, one time step (i.e. no time dynamics) or one vertical level. However, your data grids should have at least two rows and columns. Vis5D can also work with irregularly spaced data which are stored as "records". Each record contains a geographic location, a time, and a set of variables which can contain either character or numerical data.

A major feature of Vis5D is support for comparing multiple data sets. This extra data can be incorporated at run-time as a list of `*.v5d` files or imported at anytime after Vis5D is running. Data can be overlaid in the same 3-D display and/or viewed side-by-side spread sheet style. Data sets that are overlaid are aligned in space and time. In the spread sheet style, multiple displays can be linked. Once linked, the time steps from all data sets are merged and the controls of the linked displays are synchronized.

The Vis5D system includes the **vis5d** visualization program, several programs for managing and analyzing five-dimensional data grids, and instructions and sample source code for converting your data into its file format. We have included the Vis5D source code so you can modify it or write new programs. You can download the sample data sets from the LAMPS model and from Bob Schlesinger's thunderstorm model, so you can work through our examples. (We have also included a small dataset, `hole.v5d`, so you can verify that Vis5D works and try a couple of simple plots.)

Vis5D version 1.0 was written by Bill Hibbard and Dave Santek of the University of Wisconsin Space Science and Engineering Center, supported by the NASA Marshall Space Flight Center, and by Marie-Francoise Voidrot-Martinez of the French Meteorology Office. Later version enhancements were written by Bill Hibbard, Brian Paul, Johan Kellum, and Andre Battaiola. Dave Kamins and Jeff Vroom of Stellar Computer, Inc. provided substantial help and advice in using the Stellar software libraries. Simon Baas and Hans de Jong of the Netherlands ported Vis5D to HP workstations. Pratish Shah of Kubota Inc. ported Vis5D to the Kubota Alpha/Denali workstation. Mike Stroyan of Hewlett Packard added PEX support. Steven G. Johnson of MIT added autoconf/automake support.

Vis5D is offered under the terms of the GNU General Public License, which you can find in the file COPYING, with the copyright statement in the file COPYRIGHT. As the notice states, there is no warranty for the Vis5D system, but we would be interested in hearing about your questions and problems. You can join the Vis5D mailing list by sending email to <`majordomo@ssec.wisc.edu`> with:

subscribe vis5d-list

as the first line of the message body (not the subject line). You can also send questions to Bill Hibbard <`whibbard@macc.wisc.edu`> and Johan Kellum <`johan@ssec.wisc.edu`>.

Our postal address is: This document is the complete guide for using Vis5D.

Vis5d+

Over time, various enhanced versions of Vis5d have accumulated, which for one reason or another weren't folded into the main Vis5d tree. These forks unfortunately did not remain in sync with the main tree, nor were they coordinated with one another. Vis5d+ is a project that, with the original Vis5d authors' blessing, intends to be a central repository for third-party Vis5d enhancements in the future. We continue to communicate closely with the Vis5d authors, track changes in the main tree, and share bugfixes where possible. This project benefits from the collaboration and hosting services provided by SourceForge[1].

The initial version of Vis5d+ was based on a replacement of the original Vis5d's build process, which relied on specific `Makefile` rules for each supported system, with a more automated and portable system based on the GNU autoconf[2] and automake[3] tools. This led the way to a number of other improvements. Although this work was originally intended for use in the main Vis5d tree, the Vis5d authors' conservative approach towards major changes necessitate that, for now, we maintain it separately.

Vis5D Documentation on the Web

The Vis5D Home Page [4] and Vis5d+ Home Page[5] are available on the World Wide Web. You can also find this manual[6] online.

The Vis5d page links to another Web document [7] that describes how to use Vis5D files as a World Wide Web medium for exchanging model output. A second document describes the Vis5D API [8] (Application Programmer Interface), intended to help system developers use Vis5D as a visualization subsystem of other systems. Finally, there is a document describing the Vis5D Tcl scripting interface.[9]

Notes

1. http://www.sourceforge.net/
2. http://sources.redhat.com/autoconf/
3. http://sources.redhat.com/automake/
4. http://www.ssec.wisc.edu/~billh/vis5d.html
5. http://vis5d.sourceforge.net/
6. http://vis5d.sourceforge.net/doc/
7. http://www.ssec.wisc.edu/~billh/view5d.html
8. http://www.ssec.wisc.edu/~billh/api52.html
9. http://www.ssec.wisc.edu/~billh/script52.html

Chapter 2. System Requirements and Installation

In the following sections we describe the hardware and software required to run Vis5D and detail how to install Vis5D on your system.

System Requirements

Vis5d+ should, in principle, run on any system with X windows, OpenGL libraries, and at least an 8-bit color display. 32MB of RAM is recommended, and an ANSI C compiler is required to build the program. It is known for certain to run on the following systems:

1. *Silicon Graphics:* IRIX version 4.0.1 or higher; Multiple processors are used when present; IrisGL is not supported
2. *IBM RS/6000:* Model 320H or higher; AIX version 3 or later; 3-D graphics hardware is supported through OpenGL
3. *HP series 7000 or 9000 workstations:* HP-UX A.09.01 or later; PEX optional
4. *Sun SPARC:* SunOS 5.x or later

   IMPORTANT NOTE FOR SunOS 5 USERS: The X shared memory extension may not work correctly. If Vis5D prints an error message to the effect of "Shared memory error" then you'll have to append the following three lines to the end of your `/etc/system` file then reboot:

   ```
   set shmsys:shminfo_shmmax = 0x2000000
   set shmsys:shminfo_shmmni = 0x1000
   set shmsys:shminfo_shmseg = 0x100
   ```

5. *Compaq Alpha:* OSF/1 V1.3 or later; Kubota Denali Graphics hardware supported with KWS V1.3.3 or later and NPGL Run-time license

   IMPORTANT NOTE FOR OSF USERS: you need to run 'limit stacksize 32m' before you run Vis5D

6. *Linux:* x86, PowerPC, and Alpha Linux systems have been tested. A 75MHz Pentium CPU or faster is recommended. gcc 2.95.2 on PowerPC has a bug that causes problems; be sure to get a version with the latest developer patches.

Note: Note that on systems which don't have 3-D graphics hardware or OpenGL, all 3-D rendering is done in software using Mesa (an OpenGL work-alike). 3-D graphics hardware is recommended for high-end use. If you do not have Mesa, you must download and install it as described below.

If you would like to port Vis5D to a new graphics system or workstation read the PORTING file which gives more information. If you succeed, please inform us so that we may add your work to the distribution. With the new autoconf-based installation procedure, porting to new machines should be considerably easier than before (and may often work with no modifications at all).

Installing Vis5D

Before installing Vis5D, you may want to download the following:

Mesa

Mesa is a free OpenGL replacement, and is *required* if your system does not already have OpenGL, IrisGL, or PEX (an old 3d library for X). If you don't know whether you have Mesa or one of these other libraries, you can skip this step for now; when you try to configure Vis5D (in section 2.2.3) , the configure script will stop and complain if it can't find one of the libraries. Then you can go back to this step.

To install Mesa on your machine, go to the Mesa home page[1] and download it.

You'll get a file something like `MesaLib-3.1.tar.gz` Unpack it:

```
% gunzip MesaLib-3.1.tar.gz
% tar -xf MesaLib-3.1.tar
```

Then, cd to the MesaLib-3.1 directory and type:

```
% ./configure
% make
```

This should configure and compile Mesa. Finally, switch to root (superuser) and type:

```
% make install
```

If you don't have access to the root account on your machine, and your system administrator is unwilling to install Mesa for you (unlikely, since Mesa is very standard and widely-used), you can instead install it to a location in your home directory, e.g. `~/Mesa`. To do this, you would use the following steps instead of the above:

```
%./configure --prefix=$HOME/Mesa
% make
% make install
```

The last step will copy the Mesa libraries and headers to `~/Mesa/lib`, `~/Mesa/include`, etcetera.

When you run 'make install', it may print out some important directions regarding shared libraries. If you install in a non-standard location, it will typically tell you add the `prefix/lib` directory to some environment variable. On Linux systems, you may need to run **/sbin/ldconfig**, etcetera.

NetCDF (optional)

NetCDF is a free library for reading scientific data that can optionally be used by Vis5D. If you do not have NetCDF, then certain features of Vis5D will be disabled, notably irregular dataset support.

To get NetCDF, go to the NetCDF home page[2] and download the NetCDF package. As of this writing, the current version is 3.4.[3]

Once you have downloaded it, unpack and compile the package with:

```
% uncompress netcdf-3.4.tar.Z
% tar xf netcdf-3.4.tar
% cd netcdf-3.4/src
% ./configure
% make
```

On Linux systems, you may instead have to use:

```
% make CC="gcc -Df2cFortran"
```

because NetCDF doesn't know how to deal with GNU g77.

Vis5D only needs the `libnetcdf.a` file, located in `libsrc/libnetcdf.a`. If you have downloaded and unpacked Vis5D already, you can copy this file to the Vis5D directory now; otherwise, just leave it there and remember where it is.

Vis5d+

Since you are reading this file, you have probably already downloaded and unpacked Vis5d+. Otherwise, go to the Vis5d+ home page[4] and download the latest version. It will be something like `vis5d+-1.0.1.tar.gz`. Unpack it with:

```
% gunzip vis5d+-1.0.1.tar.gz
% tar xf vis5d+-1.0.1.tar
```

Now, cd to the Vis5D directory (cd vis5d+-1.0.1 if you were following the steps above). At this point, you will run a "configure" script to automatically configure Vis5D for your machine. The most basic way to run **configure** is by:

```
% ./configure
```

but there are several optional arguments that you may want to pass it:

Optional Arguments

`--prefix=dir`

By default, Vis5D will be set up to install (when you run 'make install' below) under `/usr/local` on your system. That is, the programs will be installed in `/usr/local/bin`, etcetera. This argument allows you to install to a different place. For example, if you don't have root access you may want to install to your home directory by using:

`--prefix=$HOME/Vis5D`

`--with-netcdf=lib`

This allows you to specify the location (`lib`) of the `libnetcdf.a` file. You don't need to do this if you copied `libnetcdf.a` to the Vis5D directory or if you installed it to a standard system location. Contrariwise, `--without-netcdf` forces Vis5D to *not* use NetCDF even if it could find the library.

`--enable-threads`

Turn on multi-threading mode in Vis5d (if a threads library is found); should enhance responsiveness, even on single-CPU systems.

`--with-memory=MB`

Set the maximum amount of memory to use `MB` megabytes (defaults to 32). Vis5d normally uses a bounded amount of memory to avoid swapping. When the limit is reached, the least-recently viewed graphics will be deallocated. If `MB` is set to 0, there will be no memory limit and Vis5d will never deallocate graphics.

`--with-mesa`

Force the use of Mesa even if a "real" OpenGL library is available. This just means it looks for `libMesaGL` libraries, etcetera, before `libGL` instead of afterwards. This option isn't useful with recent versions of Mesa, which install as `libGL`, etcetera.

`--with-mcidas=lib`

Specify the location of a McIDAS library file, an optional (proprietary) library that Vis5D can use. (Vis5D comes with an older McIDAS library binary for IRIX, which will be used if it works with your system.)

`--disable-fortran`

Don't compile any of the Fortran utility programs, etcetera, even if there is a Fortran compiler on your system (it will normally be used automatically). You can use this if your Fortran compiler gives you troubles when compiling Vis5D.

`--disable-shared`

`--disable-static`

Disable creation of the shared (`.so`) and static (`.a`) libaries, respectively. Don't disable both, as the Vis5D program requires a library to link to!

You can also set one or more environment variables before running **configure**, to help it find libraries and otherwise control its behavior. (You can go back and do this if configure has trouble.) You set an environment variable by the syntax `setenv variable value` (or `export variable=value` if you are using **ksh**, **bash**, or the like). Most of the time, you won't need to do this, but if you do, some possible environment variables are:

- CC: the name of the C compiler
- CPPFLAGS: `-I dir` flags to tell the C compiler where to look for header files

- CFLAGS: C compiler flags
- F77: the name of the Fortran compiler
- FFLAGS: Fortran compiler flags
- LDFLAGS: linker flags (e.g. `-L dir` to look for libraries in `dir`)

The most common use for these variables is when you have libraries installed in nonstandard locations, in which case you will use the CPPFLAGS and LDFLAGS environment variables. For example, consider the case above where you installed the Mesa library in `~/Mesa`. You would then tell **configure** where to find the Mesa libraries and header files with:

```
% setenv CPPFLAGS "-I$HOME/Mesa/include"
% setenv LDFLAGS "-L$HOME/Mesa/lib"
```

Be sure to scan the **configure** output for important warning messages, which will be surrounded by asterisks (`*******`). Once you have run **configure** successfully, you can compile Vis5D by typing:

```
% make
```

If you want, you can run the program right away by typing:

```
% src/vis5d hole.v5d -path src
```

`hole.v5d` is a sample data file included with Vis5D. The `-path src` argument is necessary to tell it where the map data files are located; it will no longer be need after you do a 'make install' (see below). Click and drag in the display window to rotate it, and click on buttons in the lower portion of the control window to plot the data.

If this completes succesfully, you will then want to install Vis5D:

```
% make install
```

(You may need to switch to root first, or do `su -c "make install"`, if you are installing in system locations.)

Note that the `make install` output may contain important instructions regarding shared libraries (if you want to link to the Vis5D libraries in your programs). e.g. on Linux you'll need to run **/etc/ldconfig** and possibly edit `/etc/ld.so.conf`.

Once this is done, the **vis5d** program, utilities, data, and libraries are installed under `/usr/local`, or wherever you specified with the `--prefix` argument to **configure** (as described above). You should now be able to run it by simply typing **vis5d** (if you installed in a system directory) or by typing `prefix/bin/vis5d` (otherwise, where `prefix` was the argument to `--prefix`), in either case followed by the name of a data file.

One sample data file, `hole.v5d`, is included with Vis5D, and more may be downloaded from the Vis5D ftp site[5].

If you want to uninstall Vis5d at any time, you can do:

```
% make uninstall
```

which removes the files installed by `make install`.

Manifest

The files installed by Vis5D include:

Table 2-1. Programs (in `prefix/bin`):

vis5d	this is the vis5d visualization program
v5dappend	utility to join v5d files together
v5dinfo	utility to see summary of a v5d file
v5dstats	utility to see statistics of a v5d file
v5dedit	utility to edit the header of a v5d file
v5dimport	utility to convert, resample, and reduce v5d files
comp_to_v5d	utility to convert (a) comp5d file(s) to v5d format
listfonts	utility to list fonts (only on SGI systems for IRIS GL)
topoinfo	utility to see info about a .topo file
fromxwd	utility to convert .xwd files to SGI .rgb format
gr3d_to_v5d	utility to convert a McIDAS GR3D file to v5d format
gg3d	McIDAS grid utility
igg3d	McIDAS grid utility
igu3d	McIDAS grid utility

Table 2-2. Libraries (in `prefix/lib`):

`libv5d`	.v5d file I/O API
`libvis5d`	the rest of the Vis5D API

(Programs using the full Vis5D API should link with *`-lvis5d -lv5d`*, while programs just doing file I/O can link with just *`-lv5d`*.)

Table 2-3. Headers (in `prefix/include/vis5d`):

`v5d.h,binio.h`	the file I/O API definitions
`v5df.h`	file I/O API for Fortran
`api.h`	rest of the Vis5D API

Table 2-4. Data (in `prefix/share/vis5d+`):

`OUTLSUPW`	World continental map lines file
`OUTLUSAM`	Medium resolution map of US with state boundaries
`EARTH.TOPO`	Earth topography file

Table 2-5. Uninstalled files and subdirectories:

`README`	short description of the program
`COPYING`	the GNU general public license
`COPYRIGHT`	copyright notice
`PORTING`	an ASCII document with notes on porting Vis5D
`hole.v5d`	sample dataset (dielectric function and z-comp. of a magnetic field guided in a dielectric slab with holes)
doc/	this manual, in PDF and HTML formats
scripts/	example scripts
src/	source code for **vis5d** and **v5dimport**
util/	source code for the Vis5D utilities
lui5/	source code for LUI user interface library
userfuncs/	directory of user-written analysis functions
contrib/	software contributed by Vis5D users
convert/	source code for sample data conversion programs

You can also download from the Vis5D web site [6] sample code [7] for user supplied data formats, including:

`LAMPS.v5d`: Sample LAMPS data set

`SCHL.v5d`: Sample data set; Bob Schlesinger's thunderstorm model

Customizing

After installation and testing you may want to customize the vis5d program by editing the `src/vis5d.h` file:

1. The visualization program vis5d assumes your system has 32 megabytes of memory. Although you can override this when you invoke vis5d, it may be convenient to change the default if your system has more than 32MB. This value is now controlled by the `--with-memory=MB` option to the **configure** script (see above).
2. There are two ways to specify a different topography and/or map file. One way is to edit `src/vis5d.h` and change the values for TOPOFILE and/or MAPFILE. For example, if you move the map and topography files to /usr/local/data, you would specify `/usr/local/data/EARTH.TOPO` and `/usr/local/data/OUTLUSAM` respectively. The other way is press the 'DISPLAY' button on the main contral panel, then press the 'Options' button found above each display menu. The 'toponame' and/or 'mapname' fields can be changed accordingly.

   By default, these files are looked for in `/usr/local/share/vis5d+`, or whereever Vis5D was installed. You can change this by either the `-path` argument to Vis5D or the VIS5D_PATH environment variable.

When you are finished changing the `src/vis5d.h` file, you must recompile. the programs.

Notes

1. http://www.mesa3d.org/
2. http://www.unidata.ucar.edu/packages/netcdf/
3. ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-3.4.tar.Z
4. http://vis5d.sourceforge.net/
5. ftp://www.ssec.wisc.edu/pub/
6. http://www.ssec.wisc.edu/~billh/vis5d.html
7. ftp://www.ssec.wisc.edu/pub/vis5d-5.2/vis5d-userdata.tar.Z

Chapter 3. Putting Your Data Into Vis5D

Vis5D can work with data organized as a 5-D rectangle. The first 3 dimensions are spatial: rows, columns, and levels (or latitutude, longitude, and height). The 4th dimension is time. The 5th dimension is the enumeration of multiple physical variables such as temperature, pressure, water content, etc.

In addition to the data itself, there are a number of parameters needed to describe a Vis5D dataset: the sizes of the five dimensions (number of rows, columns, levels, timesteps, and variables), geographic position and orientation of the data (map projection), the names of the variables, the actual times and dates associated with each timestep, etc.

The Vis5d visualization program accepts two standard file formats: v5d files and comp5d files. Both store 3-D data in a compressed format which vis5d can use quickly and efficiently. Comp5d files are those which were produced by the comp5d program in previous versions of Vis5D. The v5d file format is the new, and prefered, file format used in version 4.0 and later of Vis5D. It is intended to be a replacement for the comp5d format because it more flexible and may be extended in the future.

Vis5d can also accept a user defined file format. A user can create a new file format and the functions to read this format. See section 3.4 below.

To view your data with vis5d you will typically write a conversion program to convert your data files to v5d format. To help you do this we've included four sample conversion programs to guide you. Basically, you just add the instructions to read your file format, we provide the instructions to write the v5d file. See section 3.1 below. You then link your program with `-lv5d` (the installed `libv5d` library).

If you have used Vis5D in the past, you may continue to convert your data to McIDAS format and use comp5d to make a compressed file. However, to take full advantage of the new map projections and vertical coordinate system in version 4.0 and higher, you should write a new conversion program to make v5d files.

Another option for getting your data into Vis5D is the **v5dimport** utility. **v5dimport** is a program for file conversion, combining, and resampling. It reads a number of different file formats and can be extended to read new formats. See chapter 7 for more details.

If your data is in HDF5[1] format, you may find the **h5tov5d** conversion program useful; it is available as part of the free h5utils[2] package.

Vis5D can also work with non-gridded data. This type of data is stored as records. Each record has a geographic location and a set of variables containing character or numerical data. Vis5D does not have it's own file format for this irregular data. The two methods for getting irregular data into Vis5D are using the current irregular import program to read certain NetCDF files, or creating a new import process using calls from the `irregular_v5d` library. See section 3.5 for more details.

Converting Your Data to v5d Format

Files in the v5d format are created with functions from the v5d library. We've included four sample conversion programs which outline how to make a v5d file. They are located in the convert/ subdirectory. You can choose which one to use as a template for your data converter:

`foo_to_v5d.f`	A Fortran program which assumes a rectangular lat/lon map projection and equally spaced linear vertical coordinate system.
`foo2_to_v5d.f`	A Fortran program which allows any map projection and vertical coordinate system as well as a different number of vertical levels for each variable.
`foo_to_v5d.c`	A C program which assumes a rectangular lat/lon map projection and equally spaced linear vertical coordinate system.
`foo2_to_v5d.c`	A C program which allows any map projection and vertical coordinate system as well as a different number of vertical levels for each variable.

Note: The files in the `convert/` subdirectory link directly with the object and header files in the `src/` directory. In your own program, you'll want to instead link with `-`*`lv5d`* (the installed `libv5d` library) and include the installed headers via:

```
#include <vis5d/v5d.h>
```

In any case, each conversion program uses three functions to write the v5d file: `v5dCreate` (or `v5dCreateSimple`), `v5dWrite`, and `v5dClose`. `v5dCreateSimple` is used to create v5d files that only specify the most basic parameters. `v5dCreate` allows more complicated parameters. There are versions of these functions for C and Fortran programs.

Here are the descriptions of the `v5dCreate` and `v5dCreateSimple` functions in a format similar to man page documentation. C programmers should note that in the argument descriptions we describe arrays by Fortran convention, i.e. `A(1)` is the first element of `A`, whereas in C this would be `A[0]`.

Fortran-callable functions:

```
integer function v5dcreatesimple( name, numtimes, numvars,
nr, nc, nl, varname, timestamp, datestamp, northlat,
latinc, westlon, loninc, bottomhgt, hgtinc )

character* (*) name
integer numtimes
integer numvars
integer nr
integer nc
integer nl
```

```
character*10 varname(MAXVARS)
integer timestamp(*)
integer datestamp(*)
real northlat
real latinc
real westlon
real loninc
real bottomhgt
real hgtinc

integer function v5dcreate( name, numtimes, numvars, nr, nc,
nl, varname, timestamp, datestamp, compress, projection,
proj_args, vertical, vert_args )

character* (*) name
integer numtimes, numvars
integer nr
integer nc
integer nl(*)
character*10 varname(MAXVARS)
integer timestamp(*)
integer datestamp(*)
integer compress
integer projection
real    proj_args(*)
integer vertical
real    vert_args(*)
```

C-callable functions:

```
int v5dCreateSimple( name, numtimes, numvars, nr, nc,
nl, varname, timestamp, datestamp, northlat, latinc,
westlon, loninc, bottomhgt, hgtinc )

char  *name;
int   numtimes;
int   numvars;
int   nr, nc, nl;
char  varname[MAXVARS][10];
int   timestamp[], datestamp[];
float northlat, latinc;
float westlon, loninc; float bottomhgt, hgtinc;

int v5dCreate( name, numtimes, numvars, nr, nc, nl,
varname, timestamp, datestamp, compress, projection,
proj_args, vertical, vert_args )

char  *name;
int   numtimes, numvars;
int   nr, nc, nl[];
char  varname[MAXVARS][10];
int   timestamp[], datestamp[];
int   compress;
int   projection;
float proj_args[];
int   vertical;
```

```
   float vert_args[];
```

Arguments used by v5dCreate and v5dCreateSimple:

`name`

The name of the v5d file to create

`numtimes`

Number of timesteps (at least 1)

`numvars`

Number of variables (at least 1)

`nr`

Number of rows in all 3-D grids (at least 2)

`nc`

Number of columns in all 3-D grids (at least 2)

`varname`

Array of variable names:
varname(1) = name of first variable
varname(2) = name of second variable
...
varname(numvars) = name of last variable

`timestamp`

Array of time labels for the timesteps in HHMMSS format:
timestamp(1) = time of first timestep
timestamp(2) = time of second timestep
...
timestamp(numtimes) = time of last timestep

`datestamp`

Array of date labels for the timesteps in YYDDD format
datestamp(1) = date of first timestep
datestamp(2) = date of second timestep
...
datestamp(numtimes) = date of last timestep

Arguments used only by v5dCreateSimple:

`nl`

Number of levels in all 3-D grids (at least 1)

`northlat`

Latitude of northern edge of box in degrees

`latinc`

increment between rows in degrees (positive)

`westlon`

Longitude of western edge of box in degrees (positive Westlongitude)

`loninc`

Increment between columns in degrees (positive)

`bottomhgt`

Bottom boundary of box in km

`hgtinc`

Increment between levels in km (positive)

Arguments used only by v5dCreate:

`nl`

Number of levels in the 3-D grids per variable:
nl(1) = number of levels for first variable
nl(2) = number of levels for second variable
...
nl(numvars) = number of levels for last variable

`compress`

Compression mode (1, 2 or 4 bytes per grid point)

`projection`

Indicates type of map projection:
0 = linear, rectangular, generic units
1 = linear, rectangular, cylindrical-equidistant
2 = Lambert Conformal
3 = Stereographic 4 = Rotated

`proj_args`

Projection arguments:

```
if projection=0 then

    proj_args(1) = North boundary of 3-D box
    proj_args(2) = West boundary of 3-D box
    proj_args(3) = Increment between rows
    proj_args(4) = Increment between columns

else if projection=1 then

    proj_args(1) = North Latitude bound of 3-D box
    proj_args(2) = West Longitude bound of 3-D box
    proj_args(3) = Increment between rows in degrees
    proj_args(4) = Increment between cols in degrees

else if projection=2 then

    proj_args(1) = Standard Latitude 1
    proj_args(2) = Standard Latitude 2
    proj_args(3) = Row of North/South pole
    proj_args(4) = Column of North/South pole
    proj_args(5) = Longitude parallel to columns
    proj_args(6) = Increment between columns in km

else if projection=3 then

    proj_args(1) = Latitude of center (degrees)
    proj_args(2) = Longitude of center (degrees)
    proj_args(3) = Row of center of projection
    proj_args(4) = Column of center of projection
    proj_args(5) = Spacing between columns at center

else if projection=4 then

    proj_args(1) = North boundary on rotated sphere
    proj_args(2) = West boundary on rotated sphere
    proj_args(3) = Increment between rows
    proj_args(4) = Increment between columns
    proj_args(5) = Earth Latitude corresponding to (0,0)
    proj_args(6) = Earth Longitude corresponding to (0,0)
    proj_args(7) = Rotation angle

endif
```

`vertical`

Indicates type of vertical coordinate system:
0 = equally spaced levels in generic units
1 = equally spaced levels in km
2 = unequally spaced levels in km
3 = unequally spaced levels in mb

`vert_args`

Vertical coordinate system arguments:

```
if vertical=0 then

    vert_args(1) = height of bottom level
    vert_args(2) = spacing between levels

else if vertical=1 then

    vert_args(1) = height of bottom level in km
    vert_args(2) = spacing between levels in km

else if vertical=2 then

    vert_args(1) = height (km) of grid level 1 (bottom)
    vert_args(2) = height (km) of grid level 2
    ...
    vert_args(N) = height (km) of grid level N (top)
    where N is the maximum value in the nl array.

else if vertical=3 then

    vert_args(1) = pressure (mb) of grid level 1 (bottom)
    vert_args(2) = pressure (mb) of grid level 2
    ...
    vert_args(N) = pressure (mb) of grid level N (top)
    where N is the maximum value in the nl array.

endif
```

The v5dWrite function is used to write a single 3-D grid of data to a v5d file. The grid is identified by a timestep and physical variable number. Here is the synopsis of v5dWrite:

Fortran-callable function:

```
 integer function v5dwrite( time, var, data )
   integer time
   integer var real data(*)
```

C-callable function:

```
   int v5dWrite( time, var, data )
   int time;
   int var;
   float data[];
```

Table 3-1. Arguments descriptions:

time	A timestep number in the range [1..numtimes]
var	A variable number in the range [1..numvars]
data	3-D array of grid values; number of values = nr*nc*nl(var) ordered as data[row+nr*(col+nc*lev)] where row increases from North to South, col increases from West to East, and lev increases from bottom to top

The v5dClose function closes the v5d file after the last grid has been written. No arguments are needed. Here is the synpsis of v5dClose:

Fortran-callable function:

```
 integer function v5dclose
```

C-callable function:

```
 int v5dClose()
```

Each of the create functions returns 1 when successful and 0 when an error occurs.

Looking at any of the example data conversion programs, you'll see that there are variables which directly correspond to the arguments to v5dCreate/v5dCreateSimple. It is up to you to initialize these variables. For example, you'll have to assign to numtimes the number of timesteps in your dataset, assign to numvars the number of variables in your dataset, etc. After you've initialized all these variables, the v5dCreate (or v5dCreateSimple) call will create the v5d file. If you've failed to initialize any of the variables you will see an appropriate error message.

Next, the conversion program will enter a nested loop inside of which you must insert the code to read your data for the appropriate time step and physical variable number. Read your data into the array specified. The v5dWrite call will then compress and write the data to the v5d file. Finally, the v5dClose function will be called after all the data has been written.

After you've written and compiled your file converter, you should test it with one of your data files then check that it worked by running the v5dinfo and v5dstats utility programs on the v5d file. If everything looks OK, try running vis5d.

Here is an example of typical values that might be assigned to each variable if one were using the `foo_to_v5d.f` program:

Assignment	**Comments**
numtimes = 5	5 time steps
numvars = 4	4 physical variables
nr = 30	30 rows in each 3-D grid
nc = 40	40 columns in each 3-D grid
nl = 20	20 levels in each 3-D grid
varname(1) = "U"	U (east/west) wind component
varname(2) = "V"	V (north/south) wind component
varname(3) = "T"	Temperature
varname(4) = "P"	Pressure
timestamp(1) = 140000	2:00:00 pm
timestamp(2) = 141500	2:15:00 pm
timestamp(3) = 143000	2:30:00 pm
timestamp(4) = 144500	2:45:00 pm
timestamp(5) = 150000	3:00:00 pm
datestamp(1) = 94036	36th day of 1994 (February 5)
datestamp(2) = 94036	"
datestamp(3) = 94036	"
datestamp(4) = 94036	"
datestamp(5) = 94036	"
northlat = 60.0	Northern boundary of box is at 30 degrees latitude
latinc = 1.0	There is 1 degree of latitude between each of the 30 rows
westlon = 100.0	Western boundary of 3-D box is at 100 degrees longitude
loninc = 0.5	0.5 degree of longitude between each of the 40 columns

Assignment	Comments
bottomhgt = 0.0	Bottom of box is at 0km (sea level)
hgtinc = 1.0	1 km between each of the 20 grid levels (top at 19.0km)

The product of the number of rows, columns, levels, timesteps, and variables is the total number of data points. In this example: 30*40*20*5*4 = 480,000. A real dataset may be 100 rows by 100 columns by 20 levels, have 50 timesteps, and 10 variables for a total of 100,000,000 data points.

The difference between the foo_to_v5d program (which uses v5dCreateSimple), and the foo2_to_v5d program (which uses v5dCreate), is the later allows you to specify any map projection, vertical coordinate system, a different number of grid levels for each physical variable, and to control data compression. To specify a map projection, you must set the value of projection to 0,1,2 or 3 to indicate which projection, then specify the projection-dependent parameters in the proj_args arrray. Specifying the vertical coordinate system is done similarly.

It is sometimes useful to specify a different number of grid levels for each variable. For example, suppose most of your variables have 30 grid levels but a some variables have fewer grid levels, perhaps only one. Prior to version 4.0 of Vis5D, you would have had to fill in the extra levels with redundant, missing or dummy data values. With the v5dCreate function you can specify how many grid levels are present for each individual physical variable with the nl array parameter. Be aware that the amount of data passed to the v5dWrite call will depend on which variable you're writing. For example, if your grid has C columns and R rows then the number of values in the data array passed to v5dWrite for variable V must equal C*R*nl(V).

By default, the bottom-most grid level of each variable is displayed at the bottom of the 3-D box; each grid extends upward for how ever many levels are present. Sometimes, however, the bottom-most grid level of a particular variable should be positioned higher up. An example of this is a combined ocean/atmosphere dataset. There may be a total of 18 grid levels: the bottom 8 grid levels being ocean data and the top 10 grid levels being atmospheric data. In this case, the bottom of the atmospheric data should be offset or shifted upward by 8 grid levels.

Elaborating on the ocean/atmosphere example, suppose we have 2 ocean variables named S (salinity) and T (temperature) and 2 atmosphere variables named P (pressure) and T1 (temperature). There are 8 layers of ocean data and 10 layers of atmospheric data. Here is a summary showing how the lowlev array is the solution to this situation:

varnum	varname(varnum)	nl(varnum)	lowlev(varnum)
1	S	8	0
2	T	8	0
3	P	10	8
4	T1	10	8

The lowlev array is not specified in the v5dCreate function because it was developed after the v5dCreate function was well established. Instead, the new v5dSetLowLev function is called with the lowlev array. This separate function was added to extend the functionality of v5dCreate without changing its calling sequence. Here is the synopsis of v5dSetLowLev:

Fortran-callable function:

```
integer function v5dsetlowlev( lowlev )
integer lowlev(*)
```

C-callable function:

```
int v5dSetLowLev( lowlev )
int lowlev[];
```

Argument description:

lowlev: Specifies the vertical offset, in grid levels, for each variable.

lowlev(1) = offset for first variable

lowlev(2) = offset for second variable ... = ... lowlev(numvars) = offset for last variable

v5dSetLowLev may be called at any point between v5dCreate and v5dClose.

The v5dCreate and v5dcreate functions allow you to control how the grid data are compressed. The default is for grid values to be linearly scaled to one byte integers. This works very well for most data sets, since the scaling factors are chosen independently for each combination of time step, variable and vertical level. Furthermore, the compression to one byte per grid point enables Vis5D's high degree of interactivity, since compression allows entire data sets to be resident in memory. However, the compress argument of the v5dCreate and v5dcreate functions lets you pick whether grid point values are scaled to 1-byte integers, scaled to 2-byte integers, or left as 4-byte floating point values (no compression).

Tip: We recommend that you try compression to 1-byte integers first, and only use 2 or 4 bytes if you have precision problems at 1-byte.

Vis5D version 4.2 and later allow you to specify the physical units for each variable in your dataset. The v5dSetUnits() function takes two arguments: a variable number and a units character string. If the first variable in your file is P and the units are millibars then you can specify that with:

C: v5dSetUnits(1, "millibars")

Fortran: call v5dsetunits(1, "millibars")

The units will be displayed by the v5dinfo program and in Vis5D when using the probe.

To compile your program which uses v5dCreate, v5dWrite, and v5dClose you must link with the `src/v5d.o` and `src/binio.o` files. Alternatively, link with `-lv5d` to link with the installed `libv5d` library. See the makefiles in the convert/ directory for examples.

Finally, if your data is generated by an atmospheric or oceanic model, you may want to consider modifying your model to generate v5d files directly using the v5dCreate, v5dWrite, and v5dClose functions. Look at the sample data conversion programs for ideas.

Map Projections and Vertical Coordinate Systems

Version 4.0 of Vis5D added support for new map projections and vertical coordinate systems. When we use the term map projection, we're referring to the relationship between the rows and columns of data in the 3-D grid to the latitude/longitude of the earth. The term vertical coordinate system refers to the relationship between the vertical levels of data in the 3-D grid to altitude in the atmosphere (or depth in the ocean).

Map projections

Vis5D 5.2 supports the following map projections:

Generic rectilinear

This is a linear, regularly-spaced coordinate system with no implied units. This system is useful when your data is not related to earth science (computational fluid dynamics for example.) North/south coordinates increase upward and east/west coordinates increase to the left. The projection is defined by four parameters:

NorthBound:	Northern boundary of 3-D box
WestBound:	Western boundary of 3-D box
RowInc:	Increment (spacing) between grid columns
ColInc:	Increment (spacing) between grid rows

Example: Suppose your 3-D grid has 80 rows and 60 columns and NorthBound = 100.0 meters, WestBound = 50.0 meters, RowInc = 0.5 meters, and ColInc = 0.5 meters, then:

the south boundary will be at 60.5 meters, i.e.

southbound = NorthBound - (RowInc * (rows-1))

and the east boundary will be at 20.5 meters, i.e.

eastbound = WestBound - (ColInc * (columns-1))

Rectilinear lat/lon (cylindrical equidistant)

This is the rectangular latitude/longitude coordinate system used in previous versions of Vis5D. Latitude increases to the North (upward in the graphical display) and longitude increases to the West (leftward in the graphical display; positive west latitude). The projection is defined by four parameters:

NorthBound:	Northern boundary of 3-D box in degrees of latitude in therange [-90S,90N].
WestBound:	Western boundary of 3-D box in degrees of longitude in therange [-180E,180W].
RowInc:	Increment (spacing) between grid rows in degrees of latitude greater than zero.
ColInc:	Increment (spacing) between grid columns in degrees of longitude greater than zero.

Example: If your 3-D grid has 30 rows and 60 columns and if NorthBound = 70.0, WestBound = 140.0, RowInc = 1.0, and ColInc = 0.5, then:

the south boundary will be at 41 degrees latitude. i.e.

(NorthBound - RowInc * (rows-1))

and the east boundary will be at 110.5 degrees longitude. i.e.

(WestBound - ColInc * (columns-1))

Lambert conformal

This is a conic projection defined by the following six parameters.

Lat1, Lat2:	First and second standard latitudes in the range [-90S,90N]. Lat1 and Lat2 define where the imaginary cone intersects the sphere of the Earth. Lat1 and Lat2 must have the same sign, that is, they must both be positive or both negative. Also, Lat1 must be greater than or equal to Lat2.

PoleRow, PoleCol:	These parameters indicate the position of the north or south pole with respect to the 3-D grid coordinate system. These values may be outside the 3-D grid. If Lat1 and Lat2 are positive, the north pole is assumed, else, the south pole is assumed.
CentLon:	Central longitude: this parameter indicates which Earth longitude is to be parallel to the 3-D grid columns.
ColInc:	Increment (spacing) between grid columns at the central longitude and standard latitudes, in km. This parameter controls the scale of the projection.

Example 1: Suppose your 3-D grid has 35 rows and 40 columns and you want a Lambert conformal projection of the United States centered over Wisconsin:

Lat1 = 70.0 Lat2 = 20.0 PoleRow = -35.0 PoleCol = 20.0 Central Longitude = 90.0 ColInc = 100.0

Example 2: Suppose your 3-D grid has 35 rows and 40 columns and you want a Lambert conformal projection over Australia:

Lat1 = -20.0 Lat2 = -70.0 PoleRow = 60.0 PoleCol = 20.0 Central Longitude = -130.0 ColInc = 200.0

Note: Beware that when the pole is visible in a Lambert conformal projection, there is usually a wedge-shaped region (with its apex at the pole) which is undefined (i.e. Longitude is >180 AND <-180). In this region, there will be no map lines and the topography will be incorrect.

Azimuthal Stereographic

An aximuthal stereographic projection defined by five parameters:

CentLat, CentLon:	Latitude and longitude of the center of projection. The apex of the imaginary cone will be over this coordinate.

CentRow, CentCol:	Row and column of the center of projection. The grid row and column indicated will be at the center of the projection. These values may be outside the 3-D box.
ColInc:	Increment (spacing) between grid columns in km at the center of the projection. This parameter controls the scale of the projection.

Example: Suppose your 3-D grid has 40 rows and 40 columns and want an azimuthal stereographic projection centered over of the north pole:

CentLat = 90.0 CentLon = 0.0 CentRow = 20.0 CentCol = 20.0 ColInc = 200.0

Rotated rectilinear lat/lon

This is the rectangular latitude/longitude coordinate system on a sphere rotated with respect to the Earth's natural latitude/longitude. North/south coordinates increase upward on the rotated sphere and east/west coordinates increase leftward on the rotated sphere. The projection is defined by seven parameters:

NorthBound:	Northern boundary of 3-D box in degrees of latitude in the range [-90S,90N].
WestBound:	Western boundary of 3-D box in degrees of longitude in the range [-180E,180W].
RowInc:	Increment (spacing) between grid rows in degrees of latitude greater than zero.
ColInc:	Increment (spacing) between grid columns in degrees of longitude greater than zero.
CentLat, CentLon:	Latitude and longitude on Earth corresponding to Latitude/Longitude = (0,0) on the rotated sphere.
Rotation:	Clockwise angle of rotation of rotated sphere about its (0,0) point.

Example: Over small regions the Earth is nearly flat and we can exploit this to create nearly square grids for small scale models. We can generate a nearly square grid of 41 rows by 41 columns over a small region over Wisconsin with:

NorthBound = 2.0

WestBound = 2.0

RowInc = 0.1

ColInc = 0.1

CentLat = 43.0

CentLon = 90.0

Rotation = 0.0

Vertical coordinate systems

Vis5D 5.2 supports the following vertical coordinate systems:

Equally spaced, generic units:

This is a linear vertical coordinate system in which levels in the 3-D grid are equally spaced. No specific units are implied. The coordinate system is defined by two parameters:

BottomBound:	Bottom boundary boundary of 3-D box.
LevInc:	Increment(spacing) between grid levels.

Example: Suppose your 3-D grid has 20 levels and you want the bottom boundary to be 0.0 meters and you want .1 meters between levels. Then:

BottomBound = 0.0

LevInc = 0.1

Equally spaced, kilometers

This is a linear vertical coordinate system used in previous versions of Vis5D. Grid levels are equally spaced. The coordinate system is defined by two parameters:

BottomBound:	Bottom boundary of 3-D box in km.

LevInc:	Increment (spacing) between grid levels in km greater than zero.

Example: Suppose your 3-D grid has 20 levels and you want .5 kilometers between grid levels. Then:

BottomBound = 0.0

LevInc = 0.5

Unequally spaced, kilometers

This is a linear vertical coordinate system in which grid levels can be unequally spaced. The coordinate system is defined by an array of N height parameters where N is the number of levels in the 3-D grids. If the number of grid levels is different for each variable, N is the maximum number of grid levels.

Height(1):	Height of first (bottom) grid level in km
Height(2):	Height of second grid level in km
...	...
Height(N):	Height of Nth (top) grid level in km

Note: Note that the Height values must increase with N.

Example: Suppose your 3-D grids have 10 levels and you want the grid levels to be more closely spaced near the bottom than near the top. Then:

Height(1) = 0.0

Height(2) = 0.1

Height(3) = 0.2

Height(4) = 0.3

Height(5) = 0.4

Height(6) = 0.6

Height(7) = 0.8

Height(8) = 1.0

Height(9) = 1.3

Height(10) = 1.6

It is also possible to display the vertical axis on a logarithmic scale. This is done with the -log command line option when you start vis5d. In this case, the vertical axis is logarithmic with respect to height but linear with respect to pressure. The relationship between height (H) and pressure (P) is:

P = 1012.5 * exp[H / -7.2]

H = -7.2 * Ln[P / 1012.5]

The constants 1012.5 and -7.2 are just defaults which can be overriden when you specify the `-log` option. See section 6.1 for details.

Unequally spaced, millibars

This is a linear vertical coordinate system in which grid levels can be unequally spaced. The coordinate system is defined by an array of N pressure parameters where N is the number of levels in the 3-D grids. If the number of grid levels is different for each variable, N is the maximum number of grid levels.

Pressure(1):	Pressure of first (bottom) grid level in km
Pressure(2):	Pressure of second grid level in km
...	...
Pressure(N):	Pressure of Nth (top) grid level in km

Note: Note that the Pressure values must decrease with N.

For the purposes of calculating wind trajectories, Vis5D assumes the relationship between height (H) and pressure (P) is:

P = 1012.5 * exp[H / -7.2]

H = -7.2 * Ln[P / 1012.5]

Only the v5d file format is capable of storing the new map projection and vertical coordinate system information. When a v5d file is read into vis5d this information is used to setup the topography, map lines, and compute wind trajectories.

The vis5d program also supports two other display projections: spherical and cylindrical. Instead of drawing a rectangular 3-D box, these projections will actually warp the 3-D box into a spherical or cylincrical shape. These projections are used by specifying the `-projection` option with the value spherical or cylindrical; they are not specified in the v5d file. The spherical option can be used to display your data on a

3-D globe. The cylindrical option can be used to display your data on a flat, round topography. It's probably best to just experiment with these options using the LAMPS dataset for example. See the section on Vis5d's command line options for more information.

Special Variables and Data Values

Analysis and visualization of wind information is an important part of Vis5D. Specifically, the vis5d program looks to see if your data set contains variables named U, V and W. If present, they are assumed to be the three components of wind vectors and are used to display trajectory tracings and wind slices.

The U wind component is parallel to rows, with positive U values pointing toward increasing column numbers (i.e., positive eastward in a cylindrical equidistant map projection). The V wind component is parallel to columns, with positive V values pointing toward decreasing row numbers (i.e., positive northwardward in a cylindrical equidistant map projection). Positive W values are upward, negative W is downward. The units for U, V and W are assumed to be meters per second except when a generic map projection or vertical coordinate system is used. In that case, the units are in X per second where X is the units used to specify the northbound, westbound, rowinc, and colinc parameters.

If you do not like to use U, V, and W for wind vector components you can either specify other wind variable names on the vis5d command line or enter them while running vis5d.

Strictly speaking, U, V and W do not have to represent wind motion. They can be used to represent any flow field such as ocean currents. However, you may want to scale U, V, and W by some constant for visualization purposes.

Vis5D allows any grid data value to be undefined or 'missing'. For example, datasets based on observations are often incomplete or contain erroneous values. In your data conversion program you can indicate a grid value is missing by assigning it a value greater than 1.0e30. Missing data in vis5d will show up as holes in isosurfaces and contour slices and as black regions in colored slices. The data probe will report missing values as 'Missing'.

User Defined File Formats

This section is intended for experienced programmers. In order to allow the reading of a different file format the functions in `/vis5d-5.2/src/user_data.c` must be changed. There are two functions in `user_data.c` which need to be changed, user_data_get_header and user_data_get_grid. There is currently sample code in both of these functions to use as a template to create a new data file format. The sample code reads the sample user data in `/vis5d-5.2/user_data`.

A user can also define new topography and map file formats. In order to read a new topography file format the function user_data_get_topo must be changed, and for a new map file format the function user_data_get_map must be changed. Both of these functions are also found in `/vis5d-5.2/src/user_data.c` and they both have sample code.

In order to invoke the reading of new file format for data, topography or map files the command line option *`-userdata`* followed by a 'G' (for grid data), 'M' (for map data) and/or 'T' (for topography data). For example to use the sample code in `user_data.c` to read grid data and topography data, the following line would be used:

```
vis5d user_data/ETA.dat -user_data GT -topo user_data/ETA_TOPO.dat
```

Getting Irregular Data into Vis5D

One method for getting irregular data into vis5d is to use the irregular importer. The 'IRG IMPORT' button can be found on the Vis5D control panel which will invoke the irregular importer. The control panel will be brought up when loading a regular v5d file or it can be brought up by typing

```
vis5d -nofile
```

In the importer there is a "File List." Pressing the "Load File..." button will bring up a file browser where one can select the NetCDF files to load. Any number of NetCDF files may be loaded at one time. Once these files are loaded a list of unique times and variables will be created and displayed. The user can highlight/unhighlight desired times and variables. Frequently there may be a large number of times which do not contain much data. The "FILTER" button can be used to quickly select certain times.

Once the desired files have been selected the user will need to assign a "Dataset Name" and select the memory pool size (default 32M). The "Load" button will then load the selected files and data into vis5d.

The irregular importer can currently read three types of NetCDF files. The first type of file is created by translating the Aviation Routine Weather Reports (METARs) and the Aviation Selected Special Reports (SPECIs) to NetCDF format, using the Unidata decoders[3]. More information about these decoders and METAR data can be found at the Unidata[4] web site. The other two types of data that can be read are METARs converted to the FSL (Forecast Systems Laboratory)[5] NetCDF format and profiles which are also converted to the FSL NetCDF format.

It is possible to read other NetCDF formats with the irregular importer but it requires the addition of source code to the importer. If one is familiar with C programming and the NetCDF [6] file structure it would not be a hard task. The two files: `file.c` and `iapi.c` would require the additions and they are found in the source code.

Sample METAR data [7] in both the Unidata and FSL formats can be downloaded. This tar file also contains a gridded v5d file of an NGM model run. The time frame of this v5d file coincides with the Unidata METAR files. One can overlay both a regular v5d data set and an irregular data set in the same display.

The second method for incorporating irregular data into vis5d is the creation of a new import program or converter. This program would read the user's file format and insert the data into vis5d via the irregular_v5d library calls. The functions that would need to be called and information about them are found in the src file `/vis5d-`

`5.2/src/irregular_v5d.c`. This route for getting irregular data into vis5d is suggested for experienced programmers only.

Notes

1. http://hdf.ncsa.uiuc.edu/HDF5/
2. http://ab-initio.mit.edu/h5utils/
3. ftp://ftp.unidata.ucar.edu/pub/decoders/decoders-2.3.5.tar.Z
4. http://www.unidata.ucar.edu
5. http://www.fsl.noaa.gov/
6. http://www.unidata.ucar.edu/packages/netcdf/index.html
7. ftp://www.ssec.wisc.edu/pub/vis5d-5.2/vis5d-irregular-data.tar.Z

Chapter 4. McIDAS 3D Grid Data Files

In previous versions of Vis5D, it was standard practice to put one's data into a McIDAS GR3D file, then compress it with comp5d prior to using vis5d.

While directly converting to the v5d format is prefered, we still include this information on the McIDAS format. If you don't want to put your data into McIDAS files, you may skip to chapter 5 now.

Warning

We recommend against this way of getting your data into Vis5d. Instead, use the techniques described in chapter 3 of this manual.

A McIDAS GR3D file contains a sequence of 3-D grids of data. The three- dimensional grids are organized into short sequences to enumerate the values of multiple physical variables at a single time. The short sequences of physical variables are repeated into a longer sequence which steps through many time steps. These files have a names of the form GR3Dnnnn where nnnn is a 4-digit number between 0001 and 9999. The McIDAS utility programs then refer to files only by a number (1 through 9999).

A 3D grid file contains a directory entry for each 3D grid, which describes the size and geographic location of the grid, and the date, time and name of physical variable of the data in the grid array. A five-dimensional data set consists of a sequence of 3D grids in a 3D grid file, all with the same size and geographic locations. The grid sequence repeats the same short sequence of physical variables stepping forward through time. For example, the grid sequence from a weather model could be:

GRID NUMBER	DATE	TIME	PHYSICAL VARIABLE NAME
1	88035	000000	U
2	88035	000000	V
3	88035	000000	W
4	88035	000000	T
5	88035	000000	P
6	88035	010000	U
7	88035	010000	V
8	88035	010000	W
9	88035	010000	T
10	88035	010000	P
11	88035	020000	U
12	88035	020000	V
13	88035	020000	W
14	88035	020000	T

GRID NUMBER	DATE	TIME	PHYSICAL VARIABLE NAME
15	88035	020000	P

This data set consists of 3 time steps of 5 physical variables. The physical variables are the U, V and W components of the wind vector, the temperature T and the pressure P. The date is February 4, 1988 and the time steps are midnight, 1 AM and 2 AM. Dates are in YYDDD format and times are in HHMMSS format as described earlier.

Putting Your Data into a McIDAS 3D Grid File

The following sample program creates a 3D grid file and fills its 3D grids with data for a five-dimensional data set. This program can be found in the file `sample.F`, it's makefile is `sample.m`. The easiest way to read your data into a 3D grid file is to alter the `sample.F` program. The subroutines it calls are all in the libmain.a library, and their source is in the src subdirectory. Here is a listing of `sample.F`:

```
C THE MAIN PROGRAM OF YOUR CONVERSION PROGRAM MUST
C BE NAMED SUBROUTINE MAIN0
C
      SUBROUTINE MAIN0
C
C THE NEXT TWO COMMENTS ARE PRINTED BY THE ’help sample’ COMMAND
C ? SAMPLE program to convert data to 3D grid files
C ? sample gridf#
C
C DIMENSIONS OF 3D GRID
C NOTE NLATS AND NLONS MUST BOTH BE LESS THAN OR EQUAL TO 150
C NLATS, NLONS AND NHGTS MUST ALL BE AT LEAST 2
      PARAMETER (NLATS=31,NLONS=51,NHGTS=16)
C
C NUMBER OF PHYSICAL VARIABLES AND NUMBER OF TIME STEPS
C NOTE EITHER OR BOTH MAY BE EQUAL TO 1.  THAT IS, Vis5D DOES
C NOT FORCE YOU TO HAVE MULTIPLE VARIABLES OR TIME DYNAMICS.
      PARAMETER (NVARS=5,NTIMES=100)
C
C ARRAY FOR 3D GRID DATA
      REAL*4 G(NLATS, NLONS, NHGTS)
C ARRAYS FOR GRID FILE ID AND GRID DIRECTORY
      INTEGER ID(8), IDIR(64)
C ARRAY FOR VARIABLE NAMES
      CHARACTER*4 CNAME(5)
C
C LATITUDE, LONGITUDE AND HEIGHT BOUNDS FOR SPATIAL GRID
      DATA XLATS/20.0/,XLATN/50.0/
      DATA XLONE/70.0/,XLONW/120.0/
      DATA XHGTB/0.0/,XHGTT/15.0/
C
C STARTING DATE IN YYDDD AND TIME IN HHMMSS
      DATA JDAY/88035/,JTIME/020000/
C TIME STEP IN HHMMSS
      DATA JSTEP/000100/
C
```

```
C NAMES OF THE FIVE PHYSICAL VARIABLES
      DATA CNAME/'U   ', 'V   ', 'W   ', 'T   ', 'P   '/
C INITIALIZE GRID DIRECTORY TO ZEROS
      DATA IDIR/64*0/
C
C READ GRID FILE NUMBER FROM COMMAND LINE.  IPP WILL
C CONVERT THE PARAMETER # 1 TO AN INTEGER, WITH A DEFAULT
C VALUE OF 0.
      IGRIDF=IPP(1,0)
C IF ILLEGAL GRID FILE NUMBER, PRINT ERROR MESSAGE AND RETURN
      IF(IGRIDF .LT. 1 .OR. IGRIDF .GT. 9999) THEN
        CALL EDEST('BAD GRID FILE NUMBER ',IGRIDF)
        CALL EDEST('MUST BE BETWEEN 1 AND 9999 ',0)
        RETURN
      ENDIF
C
C CALCULATE GRID INTERVALS
      XLATIN=(XLATN-XLATS)/(NLATS-1)
      XLONIN=(XLONW-XLONE)/(NLONS-1)
      XHGTIN=(XHGTT-XHGTB)/(NHGTS-1)
C
C DATE AND TIME FOR FIRST TIME STEP
C IDAYS CONVERTS YYDDD FORMAT TO DAYS SINCE JAN. 1, 1900
      IDAY=IDAYS(JDAY)
C ISECS CONVERTS HHMMSS FORMAT TO SECONDS SINCE MIDNIGHT
      ISEC=ISECS(JTIME)
C
C INITIALIZE GRID IDENTIFIER TEXT TO BLANKS
C NOTE LIT CONVERTS A CHARACTER*4 TO AN INTEGER*4
      DO 10 I=1,8
10    ID(I)=LIT('    ')
C
C SET UP DIRECTORY ENTRY
C
C DIMENSIONS OF GRID
      IDIR(1)=NLATS*NLONS*NHGTS
      IDIR(2)=NLATS
      IDIR(3)=NLONS
      IDIR(4)=NHGTS
C
C LATITUDES AND LONGITUDES IN DEGREES * 10000
      IDIR(22)=4
      IDIR(23)=NINT(XLATN*10000.)
      IDIR(24)=NINT(XLONW*10000.)
      IDIR(25)=NINT(XLATIN*10000.0)
      IDIR(26)=NINT(XLONIN*10000.0)
C
C HEIGHTS IN METERS
      IDIR(31)=1
      IDIR(32)=NINT(XHGTT*1000.)
      IDIR(33)=NINT(XHGTIN*1000.)
C
C CREATE THE GRID FILE
      CALL IGMK3D(IGRIDF, ID, NLATS*NLONS*NHGTS)
C
C LOOP FOR TIME STEPS
      DO 200 IT=1,NTIMES
C
C SET DATE AND TIME IN DIRECTORY ENTRY
```

```
 96 C IYYDDD CONVERTS DAYS SINCE JAN. 1, 1900 TO OUR YYDDD FORMAT
 97       IDIR(6)=IYYDDD(IDAY)
 98 C IHMS CONVERTS SECONDS SINCE MIDNIGHT TO OUR HHMMSS FORMAT
 99       IDIR(7)=IHMS(ISEC)
100 C
101 C LOOP FOR PHYSICAL VARIABLES
102       DO 190 IV=1,NVARS
103 C
104 C SET VARIABLE NAME IN DIRECTORY ENTRY
105       IDIR(9)=LIT(CNAME(IV))
106 C
107 C ************************************************************
108 C READ YOUR DATA FOR TIME STEP NUMBER IT AND VARIABLE NUMBER IV
109 C INTO THE ARRAY G HERE.
110 C NOTE THAT G(1,1,1) IS THE NORTH WEST BOTTOM CORNER AND
111 C G(NLATS,NLONS,NHGTS) IS THE SOUTH EAST TOP CORNER.
112 C MARK A GRID POINT AS 'MISSING DATA' BY SETTING IT = 1.0E35
113 C ************************************************************
114 C
115 C CALCULATE 3D GRID NUMBER
116       IGRID=IV+NVARS*(IT-1)
117 C WRITE DATA IN G AND DIRECTORY IN IDIR TO 3D GRID
118 C NOTE WE PASS THE NEGATIVE OF THE GRID NUMBER (I.E. -IGRID)
119       CALL IGPT3D(IGRIDF,-IGRID,G,NLATS,NLONS,NHGTS,IDIR,IGNO)
120 C
121 C END OF PHYSICAL VARIABLE LOOP
122 190   CONTINUE
123 C
124 C INCREMENT DATE AND TIME, CONVERT JSTEP FROM HHMMSS TO SECONDS
125       ISEC=ISEC+ISECS(JSTEP)
126 C IF SECONDS CARRY PAST ONE DAY, ADJUST SECONDS AND DAYS
127       IDAY=IDAY+ISEC/(24*3600)
128       ISEC=MOD(ISEC,24*3600)
129 C
130 C END OF TIME STEP LOOP
131 200   CONTINUE
132 C
133       RETURN
134       END
```

The routines IGMK3D and IGPT3D are the interface to the 3D grid structures. The call to IGMK3D at line 90 creates a 3D grid file. Its parameters are:

1	INTEGER*4 - number of 3D grid file to create
2	array of 8 INTEGER*4 - a 32 byte text ID for the file
3	INTEGER*4 - maximum number of grid points in any 3D grid.

After the 3D grid file is created, IGPT3D is called in line 119 once for each combination of time step and physical variable to put 3D grids into the file. Its parameters are:

1	INTEGER*4 - number of 3D grid file to write to
2	INTEGER*4 - minus the number of the 3D grid to write. This is 0 or positive to indicate write to next empty grid.
3	array of REAL*4 - array of grid points to write
4	INTEGER*4 - first dimension of grid array, # of latitudes
5	INTEGER*4 - second dimension of grid array, # of longitudes
6	INTEGER*4 - third dimension of grid array, # of heights
7	array of 64 INTEGER*4 - directory for 3D grid
8	INTEGER*4 - number of 3D grid actually written, returned by IGPT3D.

Vis5D allows data sets which span more than one 3D grid file. In this case the grid sequence of repeating variables and repeating time steps continues across grid file boundaries. A single 3D grid file is limited to 100,000,000 grid points (400 megabytes). If your data set contains more than this number of grid points, then you should alter `sample.F` to create a new 3D grid file (by incrementing IGRIDF and calling IGMK3D) on every Nth time step, where N time steps will fit in one 3D grid file. Note that the comp5d command described in chapter 5 references data sets as sequences of 3D grid files.

The Vis5D system processes the gridded data based on the information in the grid directories, which is contained in the IDIR array in the sample.F program. It is a good idea to initialize IDIR to all zeros, as in line 40. The size of the 3D grid is set in entries 1 to 4 of IDIR (lines 72 to 75). Note the restrictions on data set size described in section 4.2 of this document.

The date and time of the 3D grid are set in entries 6 and 7 of IDIR, as in lines 97 and 99. Note that they are represented in our YYDDD and HHMMSS formats described above. Four functions are available in `libmain.a` for converting between these formats and a format which makes date and time calculations easy. The IDAYS function converts YYDDD format to days since January 1, 1900, as in line 60. The ISECS function converts HHMMSS format to seconds since midnight, as in lines 62 and 125. This makes it easy to do calculations with dates and times, as in lines 125, 127 and 128. Then the IYYDDD function converts days back to YYDDD and the IHMS function converts back to HHMMSS, as in lines 97 amd 99.

The physical variable name is 4 ASCII characters packed into entry 9 of IDIR, as in line 105. The LIT function in `libmain.a` converts a CHARACTER*4 to an INTEGER*4.

The spatial location of the grid is described in terms of latitude and longitude in ten-thousandths of a degree, and in terms of height (altitude) in meters. The grid element G(1,1,1) is in the north west bottom corner of the grid, and the grid element G(NLATS,NLONS,NHGTS) is in the south east top corner. The grid latitude and longitude are described in entries 21 to 25 of IDIR, as in lines 78 to 82. The grid heights are described in entries 31 to 33, as in lines 85 to 87. The NINT function is a FORTRAN intrinsic for converting a REAL to the nearest INTEGER. The latitude, longitude and height spacings are simply the distances between between successive grid points. Latitudes are positive in the northern hemisphere, longitudes are positive in the western hemispere, and of course heights are positive above sea level.

The real work in modifying the `sample.F` program is writing code for getting your data into the G array, in lines 107 to 113. For some data you may want to fake the latitude, longitude and height coordinates. However, if your data is geographical and large scale, then you may want to describe its location accurately, and it may be necessary to resample your data to a regularly spaced grid in latitude, longitude and height from some other map projection. It may also be necessary to transpose your data array to get the index order to be LAT, LON and HGT, and to invert your data array in some index to make sure G(1,1,1) is the north west bottom corner. Even in faked coordinates, you may need to transpose or invert your data array to get the right 'handedness' in the display. The Vis5D system allows grid points marked as missing, indicated by array values greater than 1.0E30. If you do fake the latitude, longitude and height coordinates, then the topography and map display of the vis5d program will be meaningless. If you calculate trajectories for your data set, either use accurate coordinates, or take great care to get relative time, distance and velocity scales consistent in the faked coordinates. Otherwaise trajectory paths will not be realistic.

The IPP function in `libmain.a` returns the value of a command parameter as INTEGER*4, as in line 45. There are similar functions CPP and DPP in libmain.a which return CHARACTER*12 (converted to upper case) and REAL*8 values for command parameters. They get command parameters based on their sequential position in the command line. They all have similar function parameters:

INTEGER*4 - sequence number of command parameter

(IPP) INTEGER*4 - default value of command parameter

or

(CPP) CHARACTER*12 - default value of command parameter

or

(DPP) REAL*8 - default value of command parameter.

There is also a mechanism for picking up command parameters based on keywords. This is done with the functions IKWP, CKWP and DKWP in libmain.a. They get command parameters based on position after a keyword of the form `-keyword`. IKWP returns an INTEGER*4, CKWP returns a CHARACTER*12 (converted to upper case) and DKWP returns a REAL*8. They all have similar function parameters:

CHARACTER*12 - keyword string in command line

INTEGER*4 - sequence number of command parameter after keyword

(IKWP) INTEGER*4 - default value of command parameter

or

(CKWP) CHARACTER*12 - default value of command parameter

or

(DKWP) REAL*8 - default value of command parameter.

The NKWP function in `libmain.a` returns the number of sequential parameters after a keyword. Its function parameter is:

CHARACTER*12 - keyword string in command line.

On the most machines the REAL*4 format is not a subset of the REAL*8 format, so make sure to declare DPP and DKWP as REAL*8, as well as their third function parameters (for default values of command parameters).

If you would rather write your grid conversion program in C instead of FORTRAN, look at the file `sample.c`. It contains examples of how to easily read and write grid files using C structures and routines in stdio.

Using the McIDAS Utilities

Once your data set is in a 3D grid file, you can list directory information about the grids using the command:

```
igg3d list I J -gr3df N
```

where N is the 3D grid file number, and I and J give the range of grid numbers to list. You can get a quick idea of the data values using the command:

```
igg3d info I J -gr3df N
```

which will list the minimum and maximum values, the mean, the standard deviation and the number of grid points marked for missing data, for grid numbers I to J in 3D grid file number N.

There are restrictions on the dimensions of data sets which can be visualized using the vis5d program. Currently, you are limited to a maximum of 30 physical variables and 400 times steps. The vis5d program will also fail if there is a trivial spatial dimension:

NLATS < 2

NLONS < 2

NHGTS < 2

The vis5d program will perform badly, possibly making errors, if the total 5-D size:

NLATS * NLONS * NHGTS * NTIMES * NVARS

is too large. The limit depends on the amount of memory in your system. For a 64MB system, the limit is around 25,000,000, with performance degrading as the data set size exceedes the limit.

Vis5D provides the gg3d and igg3d programs which can be used to reduce the resolution and scale of a data set to meet these limits. The gg3d program resamples a 3D grid to new array dimensions and new extents in latitude, longitude and height, using the command:

```
gg3d samp N I M J
```

```
gg3d ave  N I M J
```

where N and I are the numbers of the source 3D grid file and grid, and M and J are the numbers of the destination 3D grid file and grid. The 'samp' version calculates destination grid point values by linearly interpolating between source grid point values, and is appropriate for increasing resolution. The 'ave' version calculates destination grid points by averaging multiple source grid point values, and is appropriate for decreasing resolution. Without any keywords gg3d will do a straight copy operation. Invoke the gg3d command with the keyword:

`-size NLATS NLONS NHGTS`

to set the grid dimensions for the destination grid as different from the dimensions for the source grid. Invoke gg3d with the keywords:

`-lat XLATS XLATN`

`-lon XLONE XLONW`

`-hgt XHGTB XHGTT`

to set extents (range bounds) for the latitude, longitude and height for the destination grid as different from the extents for the source grid. The *`-lat`*,*`-lon`* and *`-hgt`* keywords take real arguments.

The igg3d program provides options for copying and deleting 3D grids and for interpolating between 3D grids in time. Sequences of 3D grids are copied using the command:

```
     igg3d get N I J M K
```

where N is the source 3D grid file number, I and J are the range of source grid numbers, M is the destination grid file number, and K is the starting destination grid number. A single grid may be copied within a 3D grid file using the command:

```
     igg3d copy I J -gr3df N
```

where N is the 3D grid file number, I is the number of the source grid and J is the number of the destination grid. A range of grids may be deleted with the command:

```
 igg3d del I J -gr3df N
```

where N is the 3D grid file number and grid numbers between I and J are to be deleted.

The igg3d command provides two different options for time interpolation. The first is:

```
 igg3d ave K I J D T -gr3df N
```

where grid number K is produced by interpolating between grid numbers I and J, all in 3D grid file number N. Grid number K will be assigned day D (in YYDDD format) and time T (in HHMMSS format). The relative weighting of grids I and J is calculated from this date and time, assuming linear time interpolation. If grid K is not between grids I and J in date and time, igg3d prints an error message. The igg3d command also provides a more complex time interpolation option:

```
igg3d int I D T -setdel S M -lag U V -gr3df N
```

This will put a grid in the next empty slot of 3D grid file number N, assigned to day D (in YYDDD format) and time T (in HHMMSS format). This grid will be interpolated from a sequence of grids, all in file number N, at grid numbers I, I+S, I+2S, ... , I+(M-1)S. This sequence of grids should be ascending in date and time. igg3d will search the sequence and linearly interpolate between the two consecutive grids from the sequence which bracket day D and time T. Furthermore, the interpolation will be done in a coordinate system moving at constant velocity (U, V), where U and V are in meters per second, with V positive for motion from south to north and U positive for motion from west to east. The two bracketing grids from the sequence will be shifted in latitude and longitude to their positions at day D and time T, and the result interpolated between these two spatially shifted grids. Furthermore, if the grids in the sequence are identified in their directory entries with variable name 'U ' or 'V ', then the corresponding component of the velocity (U, V) will be subtracted from the grid values.

The 'int' option of igg3d may seem complex, but it is just what you need if you want to write a script to re-interpolate a five-dimensional data set to a new sequence of time steps. It is particularly useful if the source sequence does not have uniform time steps, or if the physics are moving through the spatial grid and you want to avoid blurring in the time re-interpolation. You would set M equal to the number of time steps and S equal to the number of physical variables in the source five-dimensional data set. The I parameter would be set equal to the grid number in the first time step of the variable being interpolated. Note that this igg3d option will put the new grid at the end of the grid file containing the source data set, but you can use 'igg3d get' to move it to another grid.

You can use the command:

```
igu3d make N M
```

to create 3D grid file number N, which allows 3D grids of up to M points each. The names of 3D grid files have the form: `GR3Dnnnn`, where `nnnn` is the four digit decimal grid file number, padded with leading zeros if needed to make four digits.

Chapter 5. Vis5D Utilities

Vis5D includes a number of utility programs. This section describes each one. The v5dimport program is described separately in chapter 7.

- **v5dinfo** Usage: v5dinfo file

  Description: v5dinfo prints information about the given v5d file such as the size of the 3-D grid, the number of time steps, the names of the variables, etc.

  This program will also work on comp5d files. Therefore, the old compinfo program has been removed.

- **v5dstats** Usage: v5dstats file

  Description: v5dstats prints simple statistical information about the grid data in the named v5d file. Again, comp5d files are also accepted.

- **v5dedit** Usage: v5dedit file.v5d

  Description: v5dedit allows you to change header information such as the map projection, vertical coordinate system and variables names in the named file. It is an interacive, menu-driven program and is intended to be self explanatory. This program does NOT work with comp5d files.

- **v5dappend** Usage: v5dappend [-var] [...] file.v5d [...] target.v5d

  Description: v5dappend allows you to append a number of v5d files together to make one larger file. This might be useful if your weather model generates a separate `.v5d` file for each timestep because you'll want to join those files together to view the data in vis5d.

  The arguments are, in order:

  1. An optional list of variables to omit from the output file. For example, if you want to omit the variables U and THETA you would use the arguments *`-U`* and *`-THETA`*.
  2. The list of v5d files to append onto the target file.
  3. The name of the target v5d file to create (if it doesn't exit) or append onto (if the target file already exists).

  Note that the dimensions of the grids (rows, columns and levels) must be the same in each file to append them together. The map projection and vertical coordinate system information will be taken from the first input file and ignored the the remaining files.

- **gr3d_to_v5d** Usage: gr3d_to_v5d N M file.v5d C

  Description: gr3d_to_v5d converts (a) McIDAS GR3D file(s) to a v5d file. N is a number which indicates the name of the first grid file, M is the number of grid files to convert, `file.v5d` is the name of the file to produce, and C is 1, 2 or 4 to indicate how many bytes per grid point to use for compression (the default is 1).

Example: if N=20 and M=4 then the files GR3D0020, GR3D0021, GR3D0022, and GR3D0023 will be read an converted to the named `file.v5d`.

- **igg3d** Usage: igg3d ...

  Description: igg3d is used to perform a variety of manipulations on McIDAS GR3D files. See section 4.2 for more details.

- **igu3d** Usage: igu3d ...

  Description: igu3d is a utility to perform a variety of manipulations on McIDAS GR3D files. See section 4.2 for more details.

- **gg3d** Usage: gg3d ...

  Description: gg3d is a utility for resampling McIDAS GR3D files. See section 4.2 for more details.

- **listfonts** Usage: listfonts

  Description: listfonts, used on SGI systems only, lists the IRIS GL fonts available for use in vis5d's 3-D window. After listing the fonts you may use one in vis5d by specifying it with the *`-font`* option. For non-SGI systems or systems using OpenGL, use the xlsfonts or xfontsel program to select a font.

- **comp5d** Usage: comp5d N M filename

  Description: comp5d converts one or more McIDAS GR3D files to the comp5d format used in previous (and the current) versions of vis5d.

  N is the first 3D grid file number and M is the number of grid files in the data set. The M parameter allows data sets which span multiple grid files and should not be confused with the total number of 3D grids in the data set.

  filename is the name of the compressed grid file. You can choose whatever name you want, but note that comp5d will convert the name to all upper case characters.

  If your data set contains wind vector components you can use the *`-wind`* keyword to select a subset of wind components or calculate horizontal wind speed, named 'SPD ', for the compressed file. The longitude, latitude, and vertical components of the wind vector must be named 'U ', 'V ' and 'W ' respectively. If you use the *`-wind`* keyword, then only those wind-relevant variables (i.e. U, V, W & SPD) whose names are listed after *`-wind`* will be included in the compressed file. For example, to include SPD and W in the compressed file, from a 3D grid file containing U, V and W components, use the command:

  comp5d N M F -wind SPD W

- **help** Usage: help utilityname

  Description: The help command will list a quick reference to the parameter formats for the named utility such as igg3d, igu3d, gg3d, and comp5d utilities.

  Example: help igg3d

- `maketopo.c`

  This program, found in the util directory, is a template program for generating your own new topography (*.TOPO) files. Read the information at the top of the file for instructions. To compile maketopo see the makefile named `maketopo.m`.

- `makemap.c`

  This program, found in the util directory, is a template program for generating your own new McIDAS map outline (OUTL*) files. Read the information at the top of the file for instructions. To compile makemap see the makefile named `makemap.m`. If you create a map with lots too many line segments, it will be displayed with some line segments missing and some extra crazy line segments. You can fix this by increasing MAXMAPVERT and MAXMAPSEG in `src/globals.h`, then re-making vis5d.

- `newmap.c`

  This program and `mapfunc.f`, found in the util directory, is used to transform the vertices of an existing map outline file to make a new map outline file. This might be useful if you need to transform a map to a new coordinate system. Read the `newmap.c` and `newmap.m` files for more information.

Chapter 6. Using Vis5D to Visualize Your Data

This section describes how to use the Vis5D visualization program, vis5d. It is almost completely controlled using the mouse with a graphical user interface. The best way to learn to use it is to experiment. There is no way to harm your data from within the program.

Starting vis5d

Important note for OSF users: you need to run `limit stacksize 32m` before you run vis5d.

Important note for SunOS 5 users: The X shared memory extension may not work correctly. If Vis5D prints an error message to the effect of "Shared memory error" then you'll have to append the following three lines to the end of your `/etc/system` file then reboot:

```
set shmsys:shminfo_shmmax = 0x2000000
set shmsys:shminfo_shmmni = 0x1000
set shmsys:shminfo_shmseg = 0x100
```

After you have made a v5d file, you can interactively visualize one or more file with the command:

```
vis5d file1.v5d [options] file2.v5d [options]...
```

[options] may be any combination of the following (though none are usually needed):

`-alpha`

Use alpha blending instead of "screen door" transparency.

`-area N`

[SGI only] Specifies the first of a sequence of McIDAS area files to read and then display inside the 3-D box. See section 6.15 for more information.

`-box x y z`

This lets you specify the aspect ratio or proportions of the 3-D box. Default values are 2 2 1.

`-circle`

Display a circular clock

`-cpgeom WIDTHxHEIGHT+X+Y (or WIDTHxHEIGHT or +X+Y)`

Specify the geometry (position only) of the control panel. Since the control panel must be a fixed width, any WIDTH specification will be ignored.

`-barbs`

Use wind barbs in place of wind vectors.

`-date`

Use 'dd month yy' in place of 'yyddd' on the clock.

`-dwell`

Set the animation dwelling time when stepping from time=NumTimes to time=0

`-font xfontname`

`-font PEXfontname height`

Set the X or PEX font and its height used in the 3D window. Since 3D fonts have explicit heights, size controls the line spacing. A size of 0 will restore the line spacing to the font's height. PEX fonts (those having "PEX" in their names) are scalable. To restore the default PEX font, use " " for the name. A PEX font's default size may be resotred by specifying a size of 0.

Example:
```
vis5d LAMPS.v5d -font helvb24 20
```

`-full`

Open the 3-D window as a borderless, full-screen size window.

`-funcpath pathname`

Specify the directory to search for user Fortran functions.

Example:
```
vis5d LAMPS.v5d -funcpath /usr/local/vis5d/userfuncs
```

`-geometry WxH+X+Y (or WxH or +X+Y)`

Specify the geometry of the 3-D window.

Example:
```
vis5d LAMPS.v5d -geometry 640x480-10+10
```

`-hirestopo`

Display a high-resolution topography. This is only recommended on systems with fast graphics hardware.

`-legend position size x y`

Set color legend position and size. Position values are 1 (bottom, the default), 2 (top), 3 (left) and 4 (right). Size is the height of the legend bar and is between 10 and 1000 (default=128). Position is controlled by adding an offset value in the X and Y direction from the default position.

`-log [a] [b]`

Display height on a logarithmic axis instead of linear. This is discussed in section 3.2. The optional arguments a and b are the scale and exponent factors in the height/pressure equation. The defaults are 1012.5 and -7.2, respectively.

`-map file`

Use a map file other than the default of `OUTLSUPW`. See section 2.2.5 to setup a different default.

Example:

```
vis5d LAMPS.v5d -map OUTLUSAL
```

`-top_margin size`

Specify the size in points for the top margin of the 3d windows

`-bottom_margin size`

Specify the size in points for the bottom margin of the 3d windows

`-left_margin size`

specify the size in points for the left margin of the 3d windows

`-right_margin size`

Specify the size in points for the right margin of the 3d windows

`-mbs n`

Override the assumed system memory size of 32 megabytes. See section 2.2.5 to setup a different default value.

`-nofile`

Run vis5d with out loading a vis5d data set.

`-offscreen`

Do off screen rendering. This is used in conjunction with the `-script` option.

`-path pathname`

Use a different path for map and topo files instead of the current.

Example:
```
vis5d LAMPS.v5d -path /usr3/data
```

`-projection p`

Set the display map projection, default is to display data in its natural projection (obtained from the data file). p may be one of:

cylindrical - display data on a cylindrical Earth
spherical - display data on a spherical Earth

Only the first 3 characters are significant/needed. You will be promted for additional parameters.

Example:
```
vis5d LAMPS.v5d -projection spherical
```

`-quickstart`

Don't load any grids when starting vis5d, even if the whole file will fit into memory. The grids will be read as needed. This option is useful when reading a file via NFS.

`-rate ms`

Change the default animation rate. ms is the minimum delay in milliseconds between frames. Default is 100 ms.

`-samescale`

Set the scale for the vertical plot variables to be the same for all three variables

`-script script.tcl`

Specifies a Vis5D/Tcl script to execute automatically.

`-sequence filename`

[not available on all systems] Specifies a file containing a sequence of images to texture map over the topography. See section 6.15 for more information.

`-soundfont fontname`

Set the X font used in the sounding window. Use xlsfonts to list your system's fonts

`-title x y font "string"`

This is used in conjunction with the `-_margin` option. This will print the string at location (x,y) in the BigWindow. Any number of titles can be created. The strings will not show up if they are not in a margin

`-texture rgbfile`

[not available on all systems] Specify an SGI .rgb file to texture map over the topography. See section 6.15 for more information.

`-topo file`

Use a topography file other than the default of `EARTH.TOPO`. See section 2.2.5 to setup a different default.

`-topobase [lev_value]`

Display a base below the topography. lev_value is the vertical grid level value to which the base will extend. Typically, this is a negative value so that the base is below the lowest level (0.0). If lev_value is omitted, 0.0 will be used

Example:
```
vis5d LAMPS.v5d -topobase -3.0
```

`-trajvars uvar vvar [wvar]`

Specify which variables are to be used for trajectory tracing. Defaults are U, V, and W.

Example:
```
vis5d LAMPS.v5d -trajvars U2 V2 W2
```

`-userdata flags`

Use user-provided functions to read data, maps, or topography . flags is a string that may contain any combination of:

g or G - use the function for reading grid data
m or M - use the function for reading map data
t or T - use the function for reading topography data

Example:
```
vis5d dataset -userdata g
```

`-vertical v`

Set the vertical coordinate system, default is obtained from datafile. v may be one of:

generic - linear, equally spaced levels in generic units
equal - linear, equally spaced levels in km
nonequal - linear, unequally spaced levels in km

Only the first 3 characters of v are significant/needed. You will be prompted for additional parameters.

Example:

```
vis5d LAMPS.v5d -vertical nonequal
```

`-wdpy xdisplay`

Put the widgets on a different X display. Useful in combination with `-full` for making slides and videos.

Example:
```
vis5d LAMPS.v5d -full -wdpy pluto:0
```

`-wide w`

Set width of line segments in pixels (default is 1.0). Again, useful for making videos.

Example:
```
vis5d LAMPS.v5d -wide 3.0
```

`-wind2 uvar vvar [wvar]`

Specify the names of a secondary set of U, V, and (optionally) W wind component variables to use when drawing the Hwind2, Vwind2 and Strm2 vector slices. Useful when you have two sets of wind vector components that you want to visualize simultaneously.

Example:
```
vis5d MYDATA -wind2 U2 V2 W2
```

Most of the above arguments can also be changed by entering the 'Options' menu after pressing the 'DISPLAY' button found on the main control panel.

If you start vis5d without arguments you will get a list of all the command line options and keyboard functions. Otherwise, vis5d will begin by reading the data file.

Previous versions of vis5d required that the entire file be read into main memory; if you didn't have sufficient memory you couldn't visualize the file. In version 4.0 and higher, this restriction is lifted; you may visualize files which are larger than main memory. This is implemented with a grid cache: vis5d reads data only when needed and discards it on a least-recently-used basis. Small files will be read in their entirety as in previous versions.

For the user, this means vis5d will allow you to visualize large files even with only 32MB of main memory. However, performance will degrade as the ratio of file size to main memory size increases. If you observe sluggish performance and a lot of disk activity while running vis5d you should get more memory.

The Control Panel

After vis5d has opened/read your file/s, two windows will appear: a large window on the right containing one or more 3-D display windows and the current control panel on the left of the screen. The 3-D window/s are used to view and interact with the data. When there are multiple 3-D display windows a control panel is assigned to each. The current control panel will appear when ever a mouse event occurs in the associated 3-D display. In the upper-left corner of the 3-D display is a combination analog/digital clock which indicates the current time step. The control panel contains several groups of buttons.

Starting at the top, the first button group contains the following buttons:

[ANIMATE]	[STEP]	NEW VAR	EXIT
[TEXTURE]	TOP	SOUTH	WEST
[TOPO]	[MAP]	BOX	CLOCK
SAVE	RESTORE	GRID #'s	CONT #'s
[ANIM-REC]	REVERSE	[SAVE PIC]	[PERSPEC]
SCRIPT	INTERP	UVW VARS	LEGENDS
IMPORT	DISPLAY		

These buttons are used to control the primary functions of vis5d. Some of the above buttons are enclosed in brackets [] to indicates that they may be blank upon starting vis5d. This will happen when the button does not apply to the current data set, because the button would conflict with a command line option, or because the feature is not available on your hardware.

The next group of radio buttons control the viewing mode which determines how the mouse is used in the 3-D window:

Normal	Normal mouse mode is used to rotate, zoom, and pan the graphics in the 3-D window. See section 6.4 .

Trajectory	This mode is used for creating and displaying wind trajectories. See section 6.8 .
Slice	This mode is used to reposition horizontal and vertical slices. See section 6.6 .
Label	This mode is used to create and edit text labels in the 3-D window. See section 6.10 .
Probe	This mode is used to inspect individual grid values by moving a 3-D cursor through the 3-D grid. Using the right mouse button when clicking on this button will cause the probe to attach to the head of a trajectory. See section 6.11 .
Sounding	This mode is used to display a vertical sounding and SkewT at the location of a moveable vertical cursor. See section 6.12 .
Clipping	This mode is used to reposition the six clipping planes. See section 6.18 .

These modes are mutually exclusive; only one may be selected at a time. To the immediate right of these buttons is the mouse button legend. It is there to remind you of the use of each mouse button in the 3-D window for the currently selected mode.

Next are buttons labeled:

Hwind1 Vwind1 HStrm Hwind2 Vwind2 VStrm

A wind vector slice (Hwind or Vwind) depicts wind values by drawing small arrows which point in the direction of the wind. The length of each line segment indicates its magnitude. The tails of the line segments are all anchored within a horizontal or vertical plane through the 3-D box. The horizontal wind streamline slice (HStrm) depicts wind values by drawing streamlines on a horizontal plane. The vertical wind streamline slice (VStrm) depicts wind values by drawing streamlines on a vertical plane. The location of slice planes can be changed with the mouse while in "Slice" mode. See section 6.5 for more details.

The bottom part of the control panel window may contain a 2-D matrix of buttons when a regular v5d file is loaded. Each row corresponds to a physical variable in your dataset. Each column corresponds to one type of graphical representation. By selecting the correct row and column you can view any variable as a 3-D isosurface, horizontal contour slice, vertical contour slice, horizontal colored slice, vertical colored slice, or volume rendering. This matrix of button is scrollable if there are more rows of buttons than will fit in the window. You can use the mouse to drag the scrollbar or press the up/down arrow keys on your keyboard to scroll the button matrix. If an irregular data set is loaded, another button matrix will be available. The buttons will contain the name of the irregular v5d data set. Only the Text Plot option is currently available. See section 6.22 on viewing text plots.

When multiple data sets are viewed in the same display a period and an index number will be appended to the end of each variable name. This will be a common notation seen in several display widgets when ever multiple data sets are loaded.

The display of any graphic is controlled by clicking on its widget button with the left mouse button. Each type of graphic also has a small pop-up control window which appears when turned on. The control windows are different for each type of graphic and are explained below. To bring up a graphic's control window without toggling its display, use the middle mouse button. When the graphic is displayed it will be the same color as the widget button, making it easy to distinguish and identify different variables in the display. To change the color of the graphic, click on its widget button with the right mouse button and a small window with four slider widgets will appear. By changing the levels of red, green, and blue you can make any color.

If the control panel window becomes obscured by other windows, you can bring it to the top by pressing the **F1** key while the mouse pointer is in the 3- D window. This is especially useful when using the `-full` option.

When mutliple data sets are displayed in different displays there is the option of grouping these them together. This synchronizes the control of one display with the others. See section 6.19 for more information.

Controlling vis5d

The topmost group of buttons in the control panel operate the main functions of vis5d. Some will be discussed in more detail later.

ANIMATE	This toggle button turns animation on or off. Use the left or middle mouse buttons for forward animation and the right mouse button for reverse animation. Does not appear when viewing data sets with one time step. To make the animation slower or faster, hit the S and F key on the keyboard while the mouse cursor is inside the 3-D viewing window.
STEP	This button has three possible uses depending on which mouse button is pressed: Left Button - Step ahead one time step Middle Button - Go to first time step. Right Button - Backward one time step. This button does not appear when viewing data sets with one time step.
NEW VAR	Used to duplicate physical variables or invoke external analysis functions. This is explained further in section 6.13.
EXIT	Exit the program. A window will appear to ask you to verify your decision.

TEXTURE	Toggles display of texture maps on/off if they are loaded. See section 6.15 for more information.
TOP	Depending on which mouse button is pressed: Left or Middle: Reset the 3-D window to the default top-view. Right: Set the 3-D window to a bottom-view.
SOUTH	Depending on which mouse button is pressed: Left or Middle: Set 3-D window to a south-view. Right: Set 3-D window to a north-view.
WEST	Depending on which mouse button is pressed: Left or Middle: Set 3-D window to a west-view. Right: Set 3-D window to an east-view.
TOPO	Toggle the display of topography. This button will not appear if the topography file was not found. Click on TOPO with the right mouse button to edit the topography color.
MAP	Toggle the display of map lines. This button will not appear if the map file was not found. Click on MAP with the right mouse button to edit the color of the map lines.
BOX	Toggle the display of the 3-D box.
CLOCK	Toggle the display of the clock.
SAVE	Save current graphics and colors. After you've setup a variety of isosurfaces, slice, wind trajectories and colors it is useful to be able to save them and restore them the next time the data set is visualized. You'll be prompted for a filename. The file format, as of Vis5D 4.2 is a Tcl script. See section 6.16 for more information. Using the right mouse button when clicking on this button will allow the current vis5d data set to be saved to a file in the v5d format. See section 6.21 .
RESTORE	Restore the information save with the SAVE button. See section 6.16 for more information.
GRID #s	Normally the bounds of the data set in latitude, longitude and kilometers are displayed along the edges of the box. Use this button to display the numbers in grid coordinates instead.

CONT #s	The numbers which are drawn on contour line slices can be toggled on or off with this button.
[ANIM-REC]	This button works just like ANIMATE but allows fast animations on system with slow 3-D rendering. After each time step is rendered the image is saved in memory. When the animation loop repeats the images are quickly copied from memory to the 3-D window resulting in a faster animation.
REVERSE	Normally, the 3-D box and clock are drawn in white on a black background. This option reverses that and draws a black box and clock on a white background. This is useful for making paper print outs.
SAVE PIC	Used to save the image in the 3-D or sounding window to a file. When the 'Sounding' mode is chosen from the radio widgets the sounding window will be saved. If any other mode such as 'Normal' or 'Slice' is chosen the 3-D window (including all displays if multiple displays exist) will be saved. Depending on what system you're using a number of different picture file formats are supported. On SGI systems be sure you have the 'tops', 'frombin', and 'togif' program installed from your IRIX CD-ROM. When using OpenGL on SGIs the 'fromxwd' program is also needed. Unfortunately there is a bug in this program which often causes it to fail. Included with vis5d however is a patched version of fromxwd. An alternate way of insuring more save formats is to download ImageMagick's 'convert' program. See section 6.14 .
PERSPEC	Toggle between perspective and orthogonal viewing projections.
SCRIPT	Used to run Vis5D Tcl scripts. When you click on this button a file request will appear in which you can select the Tcl script to run. For more information see section 6.16 .

INTERP	Starts the Vis5D interactive interpreter. In your shell window you may then enter Tcl commands. Vis5D will be suspended while the interpreter is active. Type 'exit' to exit the interpreter. For more information see section 6.16 .
UVW VARS	Opens a window in which you can specify the names of the variables to use for computing trajectories and wind slices.
LEGENDS	Toggles the display of colorbar legends in the 3-D window.
IMPORT	Invokes the import utility program for bringing new data sets into a Vis5D session. For more information see section 7.1 and section 7.2 .
DISPLAY	Opens a window which allows you manipulate the assignment of data sets to display windows and change various values associated with display windows. For more information see section 6.20 .

Viewing Modes

In 'Normal' mouse mode the mouse is used to view the data in the 3-D window. By pressing the left mouse button and moving the mouse while the cursor is in the 3-D window, the 3-D image can be rotated. At any instant you can only control two of the three degrees of freedom of box rotations. However, by releasing and re-pressing the left mouse button you can change your "grip" on the box. With practice you will learn to control the box through a series of mouse moves, releasing and re-pressing the left button between moves.

The center button controls two very different things depending on how the mouse is moved. Holding the center button down and sliding the mouse away from yourself zooms in, making the box get bigger. Sliding the mouse towards yourself zooms out and makes the box get smaller. Holding the center button down and sliding the mouse right moves a plane of invisibility (i.e. a clipping plane) into the box, creating a cut away view of the box contents. Sliding the mouse left brings the clipping plane toward yourself, eventually out of the box altogether.

The right mouse button is pressed to translate the box in the window. This is useful if you want to zoom in to something that is not in the center of the box. Note that the center of rotation for box rotations stays at the center of the screen rather than in the center of the box.

The other five viewing modes will be discussed in detail in following sections.

Isosurfaces

An isosurface (3-D contour surface) shows the 3-D volume bounded by a particular isovalue. The isosurface has the specified iso-level, the volume inside contains values greater (or less) than the isovalue. The volume outside contains values less (or greater) than the isovalue.

The first column of buttons in the control panel's button matrix controls isosurfaces. Clicking on one of these buttons with the left mouse button causes a pop-up window with a slider and OK button to appear below. Select an isovalue on the slider and click on the OK button to generate an isosurface for all time steps.

Toggling ANIMATE on will let you watch the time dynamics of the iso-level contour surfaces. Note that the surfaces are generated asynchronously with the animation, so you may not see the surfaces for all the time steps as the clock hand makes it revolution. The new surfaces will appear on successive clock revolutions.

Clicking on an isosurface button with the middle mouse button will summon the pop-up window without toggling the surface on or off.

Isosurface Color

An isosurface may either be drawn entirely in one color or colored according to the values of another physical variable.

To change the color of an isosurface, click on the appropriate isosurface button with the right mouse button. A window will appear with a column of variable names (first button labeled "monocolor") and four sliders labeled red, green, blue, and transparency.

By default, monocoloring is used. To change the isosurfaces's color just move the red, green, and blue sliders.

If you click on a button other than "monocolor" you will tell vis5d to draw the isosurface according to another physical variable. The red,green,blue sliders will be replaced with a color table editor. You can change the color table (which maps data values to colors) by drawing new curves with the mouse or by pressing the up, down, left, and right cursor keys on your keyboard.

As an example, suppose you're viewing the LAMPS.v5d data set. Make an isosurface of wind speed at 40 m/s. The isosurface should be blue. Click on the SPD isosurface button with your right mouse button. The color window appears. Click on the T button in that window and the isosurface will now be colored according to temperature. You can modify the mapping from temperature values to colors by "drawing" the red, green, and blue curves in the color table window with the mouse buttons or by pressing the cursor keys. Changing the color table is explained more below in the section about colored slices.

Slices

Slices allow you to look at planar cross sections of data in the 3-D box. These slices can be oriented either horizontally or vertically and may depict either contour lines, colored slices, wind vectors, or wind stream lines.

As described in section 6.2, the last group of buttons on the control panel is a matrix of buttons, the second through fifth columns of which control slices. There is a column of buttons for horizontal contour slices, vertical contour slices, horizontal colored slices and vertical colored slices, respectively. If your data set contains U, V, and W variables, there will also be a row of wind vector slice buttons as described in 6.2. There are two buttons for horizontal wind slices and two buttons for vertical wind slices.

To activate/turn on a slice, click on the appropriate widget button with the left mouse button. The initial position for slices is the middle of the box. The exact slice location in terms of latitude, longitude or elevation is given by a small numeric labels near the one corner of each slice. To print the numbers as grid coordinates instead of geographic coordinates, toggle the "GRID #s" widget button on the control panel.

The position of slices can be changed interactively using the mouse. To do so you must first be in SLICE mode by selecting the SLICE radio button. To move any slice, simply point at the slice's corner with the mouse, press the right mouse button and drag it to a new position. Vertical slices may also be moved in a perpendicular motion by "grabbing" the middle of the top or bottom edge and dragging it. A slice may be moved while in animation mode, however, some jumpiness may occur because new slices are computed asynchronously.

Slices may also be linked in a chain. When slices are linked, the state and movement of all the linked slices are synchronized. Horizontal and vertical slices can not be linked. Slices can be linked by using the API function call 'vis5d_link_slices' and 'vis5d_unlink_slice' from a script or via the 'INTERP' button in the control panel.

Contour Line Slices

When viewing a horizontal or vertical contour line slice (button columns two and three) a small control window will appear as well. In this pop-up window you can enter the interval to use between contour lines. Just type in a new number to change the interval. Decreasing the interval will cause denser contour lines to be generated, increasing the interval will result in sparser lines.

If you enter a negative interval then all contour lines with a negative value will be drawn with dashed lines while positive values will be drawn with solid lines.

Optionally, after the interval value you may specify a range of values (a,b) which will cause only contour values between a and b to be drawn. For example, suppose you enter

-10 (-30,20)

This will result in contour lines for values between -30 and 20 at intervals of 10 with negative lines drawn as dashed lines.

The "CONT #s" button on the control panel toggles the display of the contour numbers within the slice.

The SFC button will cause the horizontal slice data to be sampled from the topography surface then displayed on top of the topograhpy.

Colored Slices

When a viewing a horizontal or vertical colored slice (button columns four and five) a color table window will appear. In this pop-up window you can change the mapping from data values to colors. If the LEGENDS control panel button is selected the color table will also be displayed in the 3-D viewing window.

The window shows graphs of red, green, and blue over the range of data values. To change the red, green, or blue function press the left, middle, or right mouse button, respectively, and drag the mouse to draw a new function. By default, low data values are mapped to blue and high data values are mapped to red.

Instead of using the mouse you can use the keyboard cursor (arrow) keys to modify the shape and position of the default function curves. Press the left/right keys to move the curves left or right. Press the up/down keys to change the shape of the curves.

You may also change the transparency of the slice as a function of the data values. Press and hold the SHIFT key while using the mouse or up/down keys to change the transparency.

There are a number of other keyboard controls for the color table window:

r	reset red, green and blue values
R	reset transparency values
c	copy color to an off-screen clipboard
p	paste colors from the off-screen clipboard
s	save color values to a file, enter filename in your shell window
l	load color values from a file, enter filename in your shell window

Wind Vector Slices

Wind vector slices are displayed with the buttons near the center of the control panel labeled HWIND-1, VWIND-1, HWIND-2 and VWIND-2. The pop-up window for these graphics contains two type-in fields to control the density and scaling of the wind vectors. The scale parameter is used to multiply the length of vectors drawn. If you want to double the length of all vectors, enter 2.0. If you want to halve the lengths, enter 0.5. The density parameter controls how many wind vectors are displayed. This value can only be between zero and one. To make one-half the number of vectors, enter 0.5, for one-fourth enter 0.25, etc. The default values for both parameters is 1.0.

The SFC button for horizontal wind slices will cause the slice data to be sampled from the topography surface then displayed on top of the topograhpy.

Wind Stream Slices

Wind stream slices show the path of wind as connected line segments. The pop-up control window contains a type-in widget to control the density of streamlines (note that the scale parameter is not used). The density parameter controls how many streamlines are displayed. This value can only be between 0.5 and 2.0. To make one-half the number of streamlines, enter 0.5, to make twice the number of streamlines, enter 2.0, etc. The default density is 1.0.

The SFC button will cause the horizontal slice data to be sampled from the topography surface then displayed on top of the topograhpy.

Slice colors

The color of a slice's control button matches that of the slice itself (except for colored slices for which the slice's tick mark matches the slice's button.) To change the color of a slice click on the slice's button with the right mouse button. A window with red, green, and blue sliders will appear. Move the sliders to change the color.

Volume Rendering

Volume rendering is a technique for displaying a 3-dimensional field as a semi-transparent colored fog. Though volume renderings of some physical variables don't look, others can be displayed very effectively with the right color mapping.

The volume rendering feature is available in Vis5D on almost all systems. Be warned that systems without 3-D graphics hardware (i.e. those using Mesa) will render volumes very slowly.

The sixth column of buttons on the control panel are the volume buttons. Only one may be displayed at a time. When a volume rendering is activated a pop- up window with a color table appears. This color table is used in exactly the same way as described for colored slices above. That is, using the mouse or keyboard you can change the function which maps data values to color and transparency. Again, the transparency can be changed while holding down the SHIFT key and drawing a curve with the mouse or pressing the up/down keys.

For those who are curious about the implementation of this feature, the volume rendering is made as follows:

1. Examine the current viewing transformation to determine which axis of the 3-D box is most nearly parallel to the view direction.
2. Create a number of colored slices perpindicular to that axis which map data values to colors and opacity.
3. Render the colored slices in back to front order. The alpha values at vertices are interpolated and blended to make smooth transitions between and within slices.

Despite the simplicity of the algorithm, most fields are rendered acceptably. Those that aren't can be improved by adjusting the color and opacity mappings. While more attractive volume rendering techniques are known, this technique can be implemented quickly on many systems.

Wind Trajectories

Wind trajectories trace the motion of air through the 3-D volume much line smoke trails in a wind tunnel. To enter trajectory mode select the TRAJECTORY radio button on the control panel. A pop-up window will appear near the bottom of the screen and a 3-D cursor will appear inside the 3-D view box. This 3-D cursor is used to specify where a new wind trajectory should be made. The STEP button on the main control panel is also important because it is used to select the time step at which to create the trajectory.

Wind trajectories are dealt with in sets. Currently, eight sets are available. Each set is represented in the trajectory window with a button labeled Set1, Set2, ..., Set8. Each set can be individually displayed, colored, or deleted. As you create new trajectories you may want to group them in sets corresponding to location, time, etc.

The first step in creating a trajectory is to select a position with the 3- D cursor. Use the right mouse button to drag the 3-D cursor around inside the 3- D box. The 3-D cursor will move in 2-D in a plane parallel to the plane of projection. That is, the cursor will stay at a constant distance of depth. By alternately rotating the view box with the left mouse button and placing the cursor with the right mouse button, the 3-D cursor can be placed anywhere inside the view box. The TOP, SOUTH, and WEST buttons as explained in section 6.2 can also be useful when making trajectories.

Second you should select a time step with the STEP button on the control panel. When the trajectory is made, it will be traced forward from the current time step to the last time step and will be traced backward through time to the first time step.

Finally, to make a trajectory at the current cursor location and current time step, press the middle mouse button when pointing inside the 3-D window. The trajectory will appear as a line segment. By turning on the ANIMATE button, you can observe how the trajectory travels through time and space. Typically, you will repeat the process of positioning the 3-D cursor and clicking the middle mouse button to create a set of trajectories.

Interesting results can be seen by making a trajectory when the ANIMATE button is turned on: a trajectory will be created for every time step instead of just one. This will show you the path of every air parcel which passes through a single point in space.

Here is a summary of the various trajectory functions:

1. To position the 3-D cursor, use a combination of rotating the view box with the left mouse button and dragging the 3-D cursor with the right mouse button.
2. Use the STEP button or ANIMATE option to select a time step.
3. Press the middle mouse button to create a trajectory at the current cursor location and time step.
4. To toggle the display of a trajectory set on or off, click on the set button with the left mouse button.
5. Select the current trajectory set by clicking on the set button with the middle mouse button.

6. A trajectory set may be deleted with the 'Delete Set' button in the trajectories window. You will asked to verify your decision.
7. You can delete the last trajectory made by clicking on the 'Delete Last' button in the trajectories window.

Wind trajectories can be depicted in two ways: as line segments or as ribbons. You can select ribbons by clicking on the RIBBON button in the trajectory window. Toggling the RIBBON button will not effect trajectories you have already made; it only controls how new trajectories will be displayed.

The trajectory window also contains two type-in widgets labeled STEP and LENGTH. The STEP value is used to control the step size used in the trajectory tracing algorithm. The LENGTH value is used to control the length of trajectories. 1.0 is the default value for each. Each acts as a multiplier. If you want the trajectory tracer to integrate in steps 1/2 the default size, enter a step value of 0.5. If you want trajectories to be twice as long as the default length, enter a length value of 2.0

The color of trajectories is controlled in the same way as for isosurfaces. That is, a trajectory set may either be mono-colored or colored according to another physical variable. Click on the trajectory set button with the right mouse button to bring up its color window. See section 6.5.1 for details on using the color window.

When viewing color-mapped trajectores be aware that the color of a trajectory is time dependent. Only the head of the trajectory is colored according to the value of another variable for the current time step. The tail of the trajectory is colored according to the color of the other variable when the head was at that location.

Wind Variables

By default, wind trajectories and the first set of wind slices are computed from the variables named U, V, and W while the second set of wind slices are computed from the variables named U2, V2, and W2. Other variables can be specified through the "UVW VARS" button on the control panel. When you click on this button a pop-up window appears in which you can specify the names of the variables to use for computing trajectories, the first set of wind slices, and the second set of wind slices. Just type in the new variables names. Be aware that uppercase and lowercase are significant. Be sure you enter valid names otherwise vis5d may not compute the graphic you select.

When multiple data sets are being viewed in one display a period and data context index number must be appended to the end of each variable to specify a data set index number. For example the first set of UVW's might be " U.0 V.0 W.0" while the second set might be "U.2 V.2 W.2". The wind variables belonging to a set MUST be from the same data set other wise they will be considered invalid.

After you've entered new wind component variable names click on APPLY to use the new values but keep the window visible. Click on OK to use the new values and close the window. Click on CANCEL to discard your changes and close the window.

You can also specify the wind component variables on the command line when you start vis5d. See section 6.1 .

Text Labels

Text labels are used to annotate the image in the 3-D viewing window. Typically this is used for making presentation graphics. You could add a title, your name, the date, highlight a particular feature of the data, or document the meaning of the data seen in the window.

To enter text labeling mode select the LABEL radio button on the control panel.

To create a text label position the mouse pointer somewhere in the 3-D window and press the left mouse button. A vertical bar cursor will appear at that location and you can now type in the text. The **Backspace** key can be used to correct errors. When you are finished, press **Return**.

To move a text label to a new position, point at it with the mouse, hold down the middle mouse button and drag the mouse. As you move the mouse an outline of the text will be dragged with the pointer until you release the mouse button.

To delete a text label, pointing at it with the mouse and pressing the right mouse button. Be careful, you will not be asked for verification before deleting a label. Once it's deleted you can only restore it by retyping it.

The SAVE button on the control panel will save any text labels you have made.

Use the `-font` command option to select a different font or change the 'fontname' field in the 'Options' menu.

Data Probe

Sometimes it's useful to be able to inspect individual data values at various locations in the 3-D volume. You can do this with the data probe. Click on the PROBE radio button on the control panel. A 3-D cursor appears in the 3-D box which you can move around using the right mouse button. For each physical variable the value for the current time step is printed along the left edge of the 3-D window. If physical units are specified for the variable they will be printed next to the value. Units can be assigned with the v5dSetUnits() function in your data conversion program as described in section 3.1.

If you turn on the GRID #'s button, the probe will be constrained to integral grid coordinates. That is, the cursor will 'snap' to the nearest discrete grid coordinate.

When the PROBE radio button is clicked on with the right mouse button is will be attached to the head of a trajectory. The probe will search for the first set of trajectories being displayed, then attach to the head of the first trajectory made in that set. The probe will also follow the movement of the trajectory during animation. This feature is useful for identifying the values at the location of a trajectory as is moves along its path.

One may also select which variables to display while in the PROBE mode. The API function 'vis5d_set_probe_vars' can be used in a script or via the 'INTERP' button.

Vertical Sounding and SkewT

When you select Sounding mode, Vis5D displays a vertical line cursor that runs from the bottom to the top of the 3-D box, and displays a vertical sounding window. The sounding window includes a SkewT diagram generated for grid data at the location of the vertical cursor (unless your data set uses generic vertical coordinates), and vertical plots of up to three variables. The user can interactively select and change which variables are used for temperature, dew point and wind fields in the SkewT diagram. Variables named T, TD, U and V are used as defaults if they exist. The user can also interactively select and change which variables are used for the vertical plots. Users can interactively control the background of the SkewT diagram and vertical plots, including dry adiabats, moist adiabats, constant mixing ratio lines, temperature lines and height tick marks. They can also interactively resize and reposition the sounding window.

When multiple data sets are being viewed in one display a period and data context index number must be appended to the end of each vertical plot variable (e.g., "T.1" or "TD.2").

The scale for the vertical plot variables can be forced to be the same for all three variables. This can be achieved by using the command line option `-samescale` or by entering the display widget (press ’DISPLAY’) then pressing the ’OPTIONS’ button and selecting ’samescale’.

Making New Variables

The NEW VAR button on the control panel is used to add new physical variables to the button matrix. There are three kinds of new variables you can add:

1. Cloned variables: these are copies of existing variables. You can use a cloned variable to make two different isosurfaces of the same variable simultaneously, for example.
2. External function variables: you can invoke an external function (which you write) to compute a new variable as a function of existing variables.
3. Computed variables: you can compute a new variable by typing in a formula involving values of existing variables.

When you click on the NEW VAR button a window appears which lists the variables that you can clone, lists the external functions that you can invoke, and lets you type in a formula for computing a new variable. After a new variable has been created a new row of buttons will be added to the control panel for the new variable. You can then make isosurfaces, contour slices, etc. of the that variable like any other.

Cloned Variables

Suppose you want to clone the U wind component variable so that you can make both +20 and -20 isosurfaces of it. First, click on NEW VAR and then select U from the pop-up window. The cloned variable will be named U’. You can then treat U’ as any other variable and make an isosurface of it.

Type-in Formulas

Type-in formulas let you type in mathematical expressions to compute new variables as a function of existing variables. For example, to compute wind speed from U, V, and W you would enter the formula:

```
SPD3D = SQRT( U*U + V*V + W*W )
```

To compute the ration of the dew point (TD) to the temperature you would enter the formula:

```
RATIO = TD / T
```

Formulas may use the names of existing variables, numbers, the arithmetic operations +, -, *, / and ** (exponentiation), and the functions SQRT, EXP, LOG, SIN, COS, TAN, ATAN (arc tangent), ABS (absolute value), MIN and MAX. MIN and MAX take two arguments, while the other functions all take one argument.

There is a special operator ’time’. This will allow one to choose the grid values for a certain time step relative to the current time step. The format is ’time(var, time_step_change). If the var is subsituted with the word ’time’ then the grid will be a constant value, the number of second since the first time step. For example in order to create a time derivative of U of the type:

U(t+1) - U(t-1)

Time(t+1) - Time(t-1)

you would enter the formula:

```
    dUdt = ( time(U,1) - time(U,-1) ) / ( time(time,1) - time(time,-
1) )
```

Click on the OK button to compute the new variable or CANCEL to discard the formula. You can edit the formula later by selecting it again from the NEW VAR pop-up window.

When multiple data sets are being viewed in one display a period and data context index number must be appended to the end of each variable in the formula. For example the above formula might instead look like:

```
SPD3D.2 = SQRT(U.0*U.0 + V.2*V.2 + W.2*W.2)
```

To compute the difference of two seperate variables you would enter the formula:

```
DIFFT.0 = T.0 - T.1
```

If there are multiple data sets but no index number appended then it will be assumed that the variable belongs to the first data set (i.e., index 0) deposited into the display. In some cases the time steps for two different variables will not coincide. When this happens the variable is computed according to the time steps of the data set which the new variable belongs to. Time steps will be matched as close as possible and time steps out of range from one other will be void of data. For example, in the formula "DIFFT.0 = T.0 - T.1", if the time steps were such:

Table 6-1. Data Context 0

	Date(YYDDD)	Time(HH:MM:SS)
Time 0	82091	0:00:00
Time 1	82091	2:00:00
Time 2	82091	4:00:00
Time 3	82091	6:00:00
Time 4	82091	8:00:00
Time 5	82091	10:00:00

Table 6-2. Data Context 1

	Date(YYDDD)	Time(HH:MM:SS)
Time 0	82091	2:30:00
Time 1	82091	4:30:00
Time 2	82091	6:30:00
Time 3	82091	8:30:00

The resulting variable "DIFFT.0" would contain the data:

Time 0	No Data
Time 1	No Data
Time 2	T.0(time=2) - T.1(time=1)
Time 3	T.0(time=3) - T.1(time=2)
Time 4	T.0(time=4) - T.1(time=3)
Time 5	No Data

External Analysis Functions

External analysis functions are an advanced feature, so new Vis5D users may want to skip this section for now.

An external analysis function is a function written by you in FORTRAN which is called by Vis5D to produce a new variable as a function of the existing variables. As an example, there is included a function SPD3D which computes wind velocity as: SPD3D = SQRT(U*U + V*V + W*W). Be aware that the external function feature is intended for experienced Vis5D users who are also proficient FORTRAN programmers.

All external functions must be placed in a directory named "userfuncs" (this may be changed in the `vis5d.h` file). This is relative to the current directory when you run vis5d. For example, suppose you always run vis5d while in "/usr/jones/data", then your analysis functions must be in "/usr/jones/data/userfuncs". Also, this directory contains a script "externf" which is used to compile your function.

To write an external function it's best to copy one of the supplied examples and then modify it. The included "userfuncs/example.f" is fully commented for this purpose. Later, when you call your function from within vis5d, the function will be invoked once for each time step. The arguments passed to the function include (the data set is the first data set in the display associated with the current control panel):

1. the number of physical variables in the data set
2. the name of each variable
3. the size of the 3-D grid
4. the date and time of the time step
5. map projection and vertical coordinate system information
6. the actual 3-D grids of data for each physical variable

Your function will have to scan the list of variable names to find the ones it needs for the computation. Then it must do the actual computation, generating a new grid of data to return to vis5d. The examples we've included demonstrate how to do this. Specifically, you should look at example.f which has detailed documentation of the function arguments. The map projection and vertical coordinate system arguments work in exactly the same way as the v5dCreate library call discussed in section 3.1 .

Suppose you want your function to be named "delta". Then the name of the FORTRAN program must be "delta.f". You would compile the function by typing "externf delta". If there are no errors, an executable file "delta" will be written. Then in vis5d when you select NEW VAR, "delta" should appear in the list of functions in the pop-up window.

There are two places for vis5d to get the grid data which it passes to your external function: from the original, uncompressed McIDAS file or the compressed v5d/comp5d file. The uncompressed McIDAS data is better because it has more precision. If the McIDAS file can't be found, then the compressed data which vis5d has in memory will be passed to your external function. Note that this has no bearing whatsoever on the construction of your external function.

You can retrieve the position and values of the data probe from within your function. To get the position of the probe use:

```
CALL PROBEPOS( ROW, COL, LEV, LAT, LON, HGT )
```

The position in grid coordinates will be returned in ROW, COLumn, and LEVel. The position in geographic coordinates will be returned in LATitude, LONgitude, and HeiGhT.

To get the value of any physical variable at the current probe position and current time step use:

```
VALUE = PROBEVAL( VAR )
```

where VAR specifies which physical variable you want.

Saving Image Files and Printing

The SAVE PIC button on the control panel can be used to save the image in the 3-D or sounding window to a file. If you are in 'Sounding' mode the sounding window will be saved, any other mode will save the 3-D window (including all displays if multiple displays exist). When you click on SAVE PIC a pop-up window appears in which you can select the file format and filename. The choices of file formats depends on the computer you're using or whether or not you have the 'convert' program installed. The formats supported by Vis5D are:

XWD	X Window Dump, displayable with xwud or xv.
RGB	SGI image file format, displayable with ipaste or xv.
GIF	Standard GIF format, displayable with xv and many other programs.
PostScript	may be printed or viewed on-screen with a program like ghostview.
Color PostScript	may be printed or viewed with a ghostview-like program.
PPM	standard unix file format.
TGA	only available with the 'convert' program

It is recomended that the 'convert' program be downloaded. This ensures more save file formats and in some cases produces higher quality save pictures. ImageMagick's 'convert' program can be downloaded at: ftp://ftp.wizards.dupont.com/pub/Imagemagick and the www page is http://www.wizards.dupont.com/cristy/ImageMagick.html Once the 'convert' binary has been downloaded or compiled it has to be placed in a directory ’util’ where vis5d is being run. For example if you are running vis5d from ’/usr/home/jane’ then the convert program must be placed in ’/usr/home/jane/util’.

Configurations of Vis5D using OpenGL directly write XWD files. The following methods are used if no ’convert’ program is present. To make an RGB file the fromxwd program is used. Unfortunately, the fromxwd program shipped by SGI has a bug which causes it to fail. Since source code for fromxwd is shipped with IRIX we include a patched version which works correctly. To make a gif file requires both fromxwd and togif (only available on SGI systems). To make a grayscale PostScript file requires the xpr utility (standard with X11). To make a color PostScript file the tops program is needed (only available on SGI systems).

If you don’t have any of the utilities mentioned above you should try using xv [1] to convert your image files.

To print a Vis5D image, position the mouse pointer over the 3-D window and press the **P** key. If you are in ’Sounding’ mode the sounding window will be printed, any other mode and the 3-D window will be printed. You’ll be asked to verify your action. Vis5D uses lpr to send a PostScript image file to the default printer or the printer specified by the PRINTER environment variable. To generate the PostScript file Vis5D uses the utilities described above. If you have problems printing you should try to first save your image as a PostScript file then try to print it manually using lpr or lp. Another option is to save your image as an XWD file then use xpr (a standard X11 utility) to convert it to PostScript and print it.

To learn more about the xwud, xpr, fromxwd, tops and togif program read the man pages. Many of these programs have options which you may find useful.

Texture mapping

Texture mapping is a term from computer graphics which means to display a 2- D image over a surface in 3-D. In Vis5D you can display images over the topography (or bottom of the 3-D box when topography is turned off) such as satellite or map images. Texture mapping is only available on SGI systems and those using the Mesa library. Hardware support for texture mapping is highly recommended.

There are three types of texture/image mapping in Vis5D which can specified on the command line or in the ’Options’ menu:

`-area N`

N is the number of the first of a sequence of McIDAS area files. The number of files read equals the number of timesteps in your datafile. Images should all be of the same size. You must use McIDAS to do remapping if necessary.

Example: Suppose your datafile has 4 time steps and you specify `-area 100`, then AREA0100, AREA0101, AREA0102 and AREA0103 will be loaded and displayed.

This option needs the McIDAS library which is only available on SGI systems.

`-sequence file`

This works like the -area option, except that the data come from a very simple file format rather than from McIDAS area files. The file starts with 3 int's that contain the number of images in the sequence, the number of lines per image, and the number of pixels per line. The rest of the file contains the images, one byte per pixel. The function read_texture_sequence in the `image.c` file of the src directory reads this file and serves as a file format reference for those wishing to create such image sequence files.

`-texture file`

This options specifies a single image to display over the topography for all time steps. The file format is the SGI RGB format. The free XV program [2] can be used to convert your image to RGB format.

When a texture map is available the TEXTURE button on the control panel is used to toggle the display of the imagery on or off.

Note these command line options can be used with each data set named on the command line, and can be entered under Options for each display in the display widget window described in Section 6.20 .

Tcl scripting

Vis5D 5.2 features a scripting facility. That is, you can control Vis5D with a text file of commands using the Tcl language. Scripting is an advanced subject and is documented separately in the Vis5D scripting document[3].

Note that the SAVE and RESTORE buttons on the control panel write and read Tcl files. You may want to use bits of these files as a basis a new Tcl scripts.

Keyboard Functions

The following keyboard functions can be invoked while the mouse pointer is inside the 3-D viewing window:

Key	Function
F1	Raise or lower the control panel window. This is useful with the *`-full`* option.
F2	Toggle display of system information including memory used and number of graphics to be computed.
P	Print the current window image. A PostScript printer must be available. Set the PRINTER environment variable from your shell to specify which printer to use.

Key	Function
S	Slower animation - increases the minimum time between frames by 10 msec.
F	Faster animation - decreases the minimum time between frames by 10 msec.
R	Reset the clipping planes to there default positions
B	Turn of the window borders.(This is useful when using Sun workstations, an unsolved bug causes the borders to flash in a very annoying manner!)
l	Reduce the size fo the Vis5D logo
L	Increase the size of the Vis5D logo

If you want to program your own keyboard functions look the in the file src/gui.c for the func1(), func2(), func3(), etc functions. They are called when the corresponding function key is pressed.

The Clipping Planes

Six new clipping planes has been added in Vis5D version 5.2. This allows you to manipulate the viewing volume in a more precise manner. Press the 'Clipping' radio button in order to enter the clipping mode.

There are two horizontal clipping planes the top and bottom. There are also four vertical clipping planes, the north, south, east and west. Only one clipping plane may be moved at a time. The middle mouse button, when clicked in the 3-D display window, will cycle through and highlight the six planes. The clipping planes are then moved in the same manner as the vertical and horizontal slices. Simply click on a corner or midpoint of the clipping slice with the right mouse button and drag it. The clipping planes may set back to there default positions by pressing the **R** key.

Grouping

When there are multiple displays you can group them together in up to 9 groups. To group a display press the desired group number 1..9 on the keypad. This puts you in group toggling mode. No other mouse or GUI event will be accepted while in this mode. Then click with the left mouse button in a 3-D display window to add it to or delete it from the group. If the display selected is not already in the group it will be added, and 'Group #' will show up in the upper right hand corner of the display window. If the display selected is in a different group it will be switched to the current group. If the selected display is already in the group you are toggling for then it will be ungrouped and the 'Group #' message will dissapear. Any 3-D display will be toggled in this manner until you exit group toggling mode by simply pressing a number 1..9 on the keypad.

If two or more displays are grouped together than the following will happen:

- If one display is rotated, zoomed or translated, all displays will change in the same manner.
- If a graphic for a certain variable is toggled in one display, the same variable name is searched for in the other displays. If the variable exists, the same graphic will be toggled.
- Isosurfaces are set to the same value.
- Slices will move in sync. They will move to the same geographical position.
- Many buttons on the control panel will be in sync.
- The time steps for all grouped displays will be merged for animation.
- The probe and sounding cursor will move in the same geographic position.
- All of the radio buttons will be in sync. If sounding mode is chosen, the sounding window for all displays will be mapped.
- Slices will be linked from all displays, sometimes creating a large chain of linked slices

When a display is grouped or ungrouped the buttons on its control panel are set to their default settings. The graphics for every variable are also turned off. If more than one data set is attached to a grouped display and they contain similar named variables, only the first data set can display graphics for that variable.

The Display Widget Window

The display widget window is the control center when multiple data sets are loaded and is invoked by pressing the 'DISPLAY' button on the main control panel. A data set is always assigned to exactly one display, but a display may contain zero, one or more data sets. A display includes a 3-D graphics window and defines a virtual grid space where all the graphics from its data sets are mapped. If the data grid is different from the virtual grid the appropriate resampling is applied. In some cases this resampling will cause graphics generation to be slow.

The display widget window allows you to: change the number of displays and how they are arranged, change the assignment of data sets to displays, change the parameters of a display which define the virtual grid, and change most of the command line options.

Each display's user interface includes a set of buttons and type-in fields and a data set browser. At the top of the user interface is the 'Display #' which gives the index of the display. Below that is a browser showing a list of data sets attached to that display. Each entry in the browser contains the context index of the data set and the context name. There is a scroll bar to the right of the browser which can scroll though the list if it gets larger than the browser window. Above the browser is the 'Options' button which is described in section 6.20.4 . Below the browser is the 'SET DISPLAY PARAMETERS' which is described in section 6.20.3 . Currently there can be a total of 16 displays. Watch out though, opening many displays can eat up a lot of memory.

Changing the Number of Displays

In the upper left hand corner is the 'Display Matrix' with 9 buttons, '1x1', '1x2', '1x3', '2x2', '3x2', '2x3', '3x3', '3x4', '4x3', and '4x4'. These buttons control the number and layout of displays. Underneath this is the 'Return to Vis5D' button. This will exit from the display widget window and map the 3-D graphics window of each display according to the chosen button matrix. If a display is created but has no data set attached to it then the 3-D window is left blank.

Changing the Display Assignment of a Data Set

In order to change the aassignment a data set to a different display you have to highlight the desired data set. This is done by simply clicking on the appropriate entry in the display browser. You then click inside the data set browser of another display and the data set transfered. If a data set is transfered to an empty display it may take several seconds while the display is initialized.

Changing the Parameters of a Display

The parameters of a display define the virtual grid. These parameters include: the number of rows, columns and levels of the grid, the map projection, and the vertical coordinate system. There are three ways of changing these parameters. When an index number is typed into the 'FROM Context #' field, the parameters are copied from the specified data context over to the current display. When an index number is typed into the 'FROM Display #' field, the parameters are copied from the specified display over to the current display. The third option is to manually edit one or all of the parameters by pressing the 'User Input' button. A set of fields associated with each parameter will then appear on the left hand side. When you are finished changing the parameters press the 'Done' button found at the bottom.

Changing the 'Options' for a Display

In Vis5D version 5.2 the command line options (refer to section 6.1) can now be changed at any time instead of just once while first running vis5d. When the 'Options' button is pressed at the top of the display browser a set of fields will appear on the left hand side. These fields correlate to the command line options.

Saving the current v5d file

When the user clicks on the SAVE button with the right mouse button a window will show up prompting for a file name. The current vis5d data set in the display will then be saved to a file in v5d format. This is only useful when new variables have been created via the NEW VAR button or thru the use of external functions.

Viewing Text Plots

When a button is clicked under the heading 'Text Plot', the text plot widget window will appear. This will allow you to select which variable you want to plot. Only one variable per data set can be selected.

The density of the text plots can be controlled by changing the number in the 'Spacing' field. The higher this number, the more distance there is between the plots. To view all the data points, which may look jumbled in some cases, set the 'Spacing' field to zero.

The font size and spacing between characters can be controlled by entering a number into the font fields, or by clicking on the + or -.

By clicking on the selected variable with the right mouse button, a color widget appears, allowing the user to alter the color of the text. If the chosen variable contains numerical data, the option for 'Multicolor' apperas. When this is clicked on the text characers will be mapped to their color value.

Final Notes

If `--enable-threads` was used when configuring Vis5d, then the program uses multiple threads and CPUs (if available) to compute graphics in the background; thereby increasing vis5d's speed. Otherwise, vis5d tries to interleave the computation of graphics with user interaction. This results in the user interface being a bit sluggish until all pending graphics computations are completed.

The vis5d user interface may be complex to describe in words, but we have tried hard to make it simple in reality. After a little practice using the sample data sets we hope it feels natural.

Since version 3.2 of Vis5D there is a user-contributed software directory: contrib/. See the README file in that directory for a description of current contributions.

Notes

1. http://www.trilon.com/xv/
2. http://www.trilon.com/xv/
3. http://www.ssec.wisc.edu/~billh/script52.html

Chapter 7. The v5dimport Utility

The **v5dimport** utility is a program for converting grid files to v5d format, combining multiple source files, resampling to new coordinate systems and culling variables and timesteps. It has both a graphical and command line user interface.

For example, you may use **v5dimport** to read 2 McIDAS GR3D files and a 2-D McIDAS GRID file, resample all the data to a Lambert Conformal projection, omit the CWAT and VORT variables and then write the data to a Vis5D file called `lambert1.v5d`.

The basic order of events when using **v5dimport** is:

1. Read the input file(s).
2. Select grids for output according to timestep, physical variable, map projection or vertical coordinate system.
3. Setup a map projection and vertical coordinate system for the output file.
4. Write the output file. Resampling is done at this time.
5. Optionally, start Vis5D on the output file.

Currently, **v5dimport** can read the following file formats:

McIDAS GR3D and GRID files

Vis5D v5d and comp5d files

GRADS files

"UW vis" files (used at the University of Wisconsin)

EPA MM4 and RADM files (on Crays only)

Using v5dimport's graphical interface

Start **v5dimport** from your shell with

```
v5dimport [-path pathname] [files]
```

where [files] is an optional list of input files and [*`-path pathname`*] specifies that the directory named "pathname" is to be used as the default, in place of the current directory, for the input file browser and for making output files.

When **v5dimport** has started you'll see its main window appear. It consists of:

1. a scrollable list of all grids scanned from the input files
2. buttons used for selecting/culling grids according to variable name, timestep, projection or vertical coordinate system

3. buttons and type-in fields for describing and creating the output file.

Reading input grids

You may read additional grid files into **v5dimport** at any time by clicking on the "Read file..." button. Use the file selector to locate your file and click on OK or CANCEL. It's best to read all input files right at the beginning because whenever a new file is read all grids are selected for output, overriding any selections you may have previously made.

The button labeled "Discard all grids" does exactly what it says. It's equivalent to exiting **v5dimport** and restarting it.

After reading each input file, the list of grids shown in the top half of the window, will be resorted by time then variable name.

The columns in this list are:

Grid	grid number (no significant meaning)
YYDDD	the year and date of the grid
HHMMSS	the time of the grid in hours, minutes, and seconds
Variable	the variable name
Nr	number of grid rows
Nc	number of grid columns
Nl	number of grid levels
Proj#	the projection number (see "Select by projection..." window)
VCS#	the vertical coordinate system number (see "Select by VCS...")
Filename	name of file the grid was found in

Selecting grids for output

It's often the case that one wants to discard some physical variables or timesteps from the input file so they aren't written to the output file. By default, all variables are selected for output.

To select/cull variables, click on the "Select by variable..." button. A pop-up window will appear which lists all the variables. The ones that are highlighted are selected for ouput. Click on variables names to select or deselect them.

Similarly, you can select timesteps via the "Select by time..." button. A pop-up window listing all time steps will appear. Use the mouse to select the time steps you want and unselect the timesteps you wish to omit. Note that you can select/deselect a number of timsteps by just dragging the mouse while holding down the button.

Finally, grids may be selected or discarded according to their map projection or vertical coordinate system (VCS) via the "Select by projectiion..." and "Select by VCS... buttons.

Note that as you select/deselect timesteps, variables, projections, or VCSs the effected grids will be high-lighted/unhigh-lighted in the main grid list.

The "Select All" and "Select None" buttons do just what they imply.

Defining the output file

The default parameters for the output file (grid size, projection, etc) are taken from the first file read in. You should always review these parameters before making your output file. It will often be necessary to change these values.

The number of rows, columns, and levels for the output file is specified by the type-in fields on the main window labeld "Rows", "Columns" and "Max Levels". Type in new values if the defaults are incorrect.

The map projection for the output file can be viewed and changed by clicking on the "Map projection..." button. In this pop-up window you'll be able to choose a map projection type then enter the specific projection parameters. There is also a "Guess" button which will attempt to find a reasonable output projection given the currently selected grid list. It's often helpful to have the "Select by Projection" pop-up window on-screen to compare the output projection to the input projections.

The vertical coordinate system for the output file can be viewed and changed by clicking on the "Vertical Coord System..." button. In this pop-up window you'll be able to choose a vertical coordinate system type and enter the specific parameters. This window also has a "Guess" button to try to find a reasonable default. Similarly, it's often helpful to have the "Select by VCS" pop-up window on-screen to compare the output VCS to the input VCSs.

Making the output file

Enter a filename for the output file in the type-in field at the bottom of the main window then click on "Make". Messages will be printed as the file conversion takes place. If there are any errors the process will halt. Note that generating the output file can be time-consuming if data must be resampled from the input grid's coordinate system to a new coordinate system for the output file.

If you click on "Visualize" this will make the file and then automatically start up Vis5D on that file (i.e., you don't need to click on "Make" first). If you type a filename in the type-in field, it wil use that name. Otherwise, it will use your login name followed by ".v5d". If you want command line options on the Vis5D command, put them in a file named `vis5d_options`. For example, *`-mbs 64`*.

Miscellaneous

An options window is available by clicking on the "Options..." button.

The first item controls the "combining of co-located data". It may be the case that several 3-D grids, selected for output, are co-located in space and time. When computing the value to put in the output file you can either choose the data value from the higher resolution grid at that location, or take the average of all grid values at that grid location.

The second item controls how grid data is compressed in the output file. By default, grid values are scaled down to 1-byte integers. Alternately, you can scale down to 2-byte integers for better resolution, or perform no compression/scaling by selecting 4-byte floating point values. This option respresents a tradeoff in file size and precision.

The v5dimport User Interface Embedded in Vis5D

As of Vis5d version 5.2, **v5dimport** has been incorporated into the main control panel and is invoked by pressing the "IMPORT" button. This version of **v5dimport** is the same as the one mentioned above except that you can load the newly created v5d file without exiting from the program. When the "Make" button is pressed it will create the file and name it using the field "File name:". If there is a name in the field "Context name:" the file will be loaded into Vis5d and attached to the first display. A number must also be entered into the "MBS" field to specify the amout of memory to allocate for the data set. A filename must always be entered to use for disk backup for caching grids.

Using v5dimport's text interface

The text/type-in interface to **v5dimport** is useful when X is not availableor when you want to run **v5dimport** with a script. To start **v5dimport** in text mode enter:

```
v5dimport -t [-path pathname] [files]
```

where [`files`] is an optional list of input files and [`-path pathname`] specifies that the directory named "pathname" is to be used as the default, in place of the current directory, for the input file browser and for making output files. Through the text interface it's possible to run **v5dimport** with a script by using your shell's import redirection feature:

```
v5dimport -t < script
```

After you've invoked **v5dimport** with the -t option you'll see a >> prompt at which you can issue any of these commands:

exit	exit v5dimport
help	online help
list	show lists of grids, timesteps, variables, map projections, or vertical coordinate systems.
read	read an input file

keep/omit	used to select which grids, according to timestep, variable, map projection or vcs, are to be included in or omitted from the output file.
info	display parameters of output file
rows	specify number of grid rows for output file
columns	specify number of grid columns for output file
levels	specify max number of grid levels for output file
projection	specify the output file's map projection
vertical	specify the output file's vertical coordinate system
make	make the output file
visualize	make the output file and start Vis5D

Using the text interface to **v5dimport** is similar in strategy to the graphical interface:

1. Read input files
2. Select grids by timestep, variable, projection, and/or VCS. This is typically done by a series of list, omit, and keep commands.
3. Set/adjust output file parameters. Typically a series of info, rows, columns, levels, projection, and vertical commands.
4. Make the output file, or make the output file and start Vis5D.

Use the **help** command to learn the exact syntax for each command.

A **v5dimport** script is simply an ASCII file of **v5dimport** commands and their arguments. In the simplest case it may contain only a few commands such as:

```
# read my file, omit two vars, write v5d file
read mydata.dat
omit var CW
omit var RW
make outdata.v5d
exit
```

As **v5dimport** executes a script it prints each command and its result. Lines that start with a "#" are considered comments and ignored.

Adding support for new file format

v5dimport was written so that adding code to read new file formats should be easy. The source code for **v5dimport** is in the `src/` subdirectory (see especially `v5dimport.c` and `file_i.c`). Look for the comment:

```
/*** ADD NEW FORMATS HERE ***/
```

to see where code has to be added to support a new file format.

Basically, you need to write two new functions. One which scans your file format to build a list of grid_info structs. The other reads the actual grid data from your file given a grid_info struct. These functions should be put in a new file named `read_foo.c` where foo is the name of your file format. Then, update the `file.c` file to use your functions. Use the existing `read_*.c` files as a guide.

Notes on specific file formats

The symbol EPA is defined on the cc command line with *`-DEPA`* only on systems which can read EPA files. Currently, only Cray systems can read EPA files because the EPA-provided file reading functions only work on Cray computers.

The symbol MCIDAS is defined on the cc command line with *`-DMCIDAS`* only on systems which can use the `libmcidas.a` file. Only SGI's in 32-bit mode are supported now.

Chapter 8. Sample Data Sets

To demonstrate or experiment with Vis5D we provide two sample datasets.

Bob Schlesinger's thunderstorm simulation

To visualize the Schlensinger thunderstorm file enter the command

```
vis5d SCHL.v5d
```

To view an isosurface of QL (moisture content):

1. Click on the QL button in the left column of the button matrix.
2. On the slider, select a value near 1.0, then click on the OK button.
3. Turn on animation with the ANIMATE button.

To view a vertical contour line slice of QL:

1. Turn off animation by clicking on ANIMATE again.
2. Click on the QL button in the third column.
3. Move the slice by first selecting the SLICE radio button. Then use the right mouse button to drag any corner of the slice along the edges of the 3-D box.

LAMPS model

To visualize a LAMPS (Limited Area Meso-Scale Prediction System) model simulation of an extratropical cyclone, enter the command:

```
vis5d LAMPS.v5d
```

To view an isosurface of wind speed over a topography with map lines:

1. Click on the TOPO and MAP buttons.
2. Click on the SPD button in the first column. Then select a value near 45.0 on the slider and click on OK.
3. Turn on ANIMATE and you will see an animation of the 45 m/s wind isosurface.

To make some interactive wind trajectories:

1. Turn off the wind speed isosurface by clicking on the SPD button again
2. Select the TRAJECTORY button.
3. Move the mouse pointer into the 3-D window and press the middle mouse button. You will get a series of white wind trajectory lines passing through the 3-D cursor location.
4. Move the 3-D cursor by dragging it with the right mouse button then click the middle button to make more trajectories.
5. Select RIBBON and then the SET 2 button and try making some yellow ribbon trajectories.

Example McIDAS files and utilities

The Schlesinger and LAMPS data sets are also available as the 3D McIDAS grid files named `GR3D0001` and `GR3D0002`. They are available on the Vis5D ftp site[1]. See chapter 2 for more information.

To list the grids in `GR3D0001` and to see statistics about them, enter the commands:

```
igg3d list 1 190 -gr3df 1
igg3d info 1 190 -gr3df 1
```

The `SCHL.v5d` file was made from the `GR3D0001` file with the command:

```
gr3d_to_v5d 1 1 SCHL.v5d
```

To list the grids in `GR3D0002` and to see statistics about them, enter the commands:

```
igg3d list 1 189 -gr3df 2
igg3d info 1 189 -gr3df 2
```

The `LAMPS.v5d` file was made from the `GR3D0002` file with the command:

```
gr3d_to_v5d 2 1 LAMPS.v5d
```

A variety of other sample datasets are available on the ftp site[2] or upon request.

Notes

1. ftp://www.ssec.wisc.edu/pub/vis5d-5.2/
2. ftp://www.ssec.wisc.edu/pub/vis5d-5.2/

Chapter 9. Version History

Descriptions of the latest revisions to Vis5d+ can be found in the online release notes[1] or in the included `NEWS` file. Below, we describe the revision history of the original Vis5d.

This is a summary of the versions of Vis5D

1.0 (December 1988)

This was the original version of Vis5D for the Stellar GS-1000. It was used to give demonstrations at the ECMWF in December 1988 and at the AMS conference in Anahiem in January 1989. It had the following features:

- Depict time series of multivariate 3-D grids by animated isosurfaces and horizontal contour line slices.
- World topography map with map boundaries.
- Wind trajectory tracing with the traj5d program.

2.0 (Fall 1991)

This version was only available for the Stellar GS-1000/2000 and introduced the following features:

- Faster isosurface generation.
- Horizontal and vertical slices moved interactively with the mouse.
- Colored slices.
- Interactive wind trajectory creation.
- Ribbon trajectories.
- Label / text annotations.
- "Pretty" rendering option.

The format of the compressed grid file was changed slightly with version 2.0. Specifically, the trajectory files of version 1.0 were eliminated, trajectories are now stored in the compressed grid file itself. Also, the internal storage representation for surfaces and slices has been changed.

2.1 (February 1992)

This is the first version of Vis5D available for the SGI and IBM workstations. It was also modified to use less memory during isosurface generation.

2.2 (April 1992)

This version of Vis5D runs on the base SGI Indigo with 8-bit color though some features not available. It also has the following improvements:

- The *`-box`* option for changing the proportions of the 3-D box (SGI and Stardent only).

- User topography files. Vis5D now uses the `EARTH.TOPO` file instead of `TOPOHRES` to make the map.
- The `maketopo.c` program shows how to make new .TOPO files. (SGI and Stardent only)

3.0 (August 1992)

This version features the following improvements:

- Horizontal and vertical wind vector slices.
- Improved SAVE and RESTORE functionality.
- New trajectory widget options.
- Separate map and topography controls.
- CLONE option added.
- Simultaneous colored and contour line slices.
- Improved transparency, *`PRETTY`* option on SGI.
- Same source code for SGI, Stardent, and IBM.
- Improved portability and porting guide added.
- New video and hardcopy convenience features.

3.1 (July 1993)

New features:

- User-written analysis functions.
- SAVE PIC button to save window image to a file.
- Perspective viewing mode.
- New contour line options to draw dashed negative lines and restrict contouring to a specific range of values.
- Data Probe mode.
- Topography color editing.
- Grid compression done layer-by-layer.

3.2 (August 1993)

New features and changes:

- Volumetric rendering on SGI systems with VGX, VGXT, VTX, RE, or RE2 graphics hardware.

- User-contributed software directory.
- 2-D contour function rewritten in C.

3.3 (January 1994)

Vis5D ported to HP, DEC, Sun, and Kubota (DEC Alpha) workstations. The most important part of this work was the enhancement and integration of the VOGL library. This work was done by Simon Baas and Hans de Jong for the Dutch Meteorological Institute, KNMI. Porting to the Kubota Denali graphics system was done by Pratish Shah of Kubota Inc. Thanks guys!

New features:

- `wdpy` option now creates a window on the widget display which can be used to move and interact with the 3-D view using the widget display's mouse.
- SAVEPIC button let's you save the window image in PostScript or color PostScript formats (SGI only).
- `wind2` option added to specify a second set of U,V,W variables for the second set of wind vector slices.
- `texture` option added for a texture mapping an image onto the topography (SGI only).
- user functions are computed faster on SGI multi-processor systems by computing time steps in parallel.

4.0 (December 1994)

New features:

- Map projections and new vertical coordinate systems.
- Type-in formulas for computing new variables.
- Time sequences of satellite images can be texture mapped onto the topography for visual comparison with model data.
- Data may be displayed over a spherical Earth.
- File caching: compressed grid files which are too large to read into memory in their entirety are read in piece-by-piece as needed, a least-recently-used replacement policy is used to purge data when memory is full.
- New compressed grid format. New format allows new header information to be added in the future, currently stores additional projection information. Also allows control of data compression.
- New command line options: `-geometry`, `-trajvars`, `-projection`, `-vertical`, `-area`, `-sequence`
- External functions can query the probe position and values with PROBEPOS and PROBEVAL functions.

- Interactive control over animation rate (using **F** and **S** keys)
- When the "GRID #'s" button is turned on, the probe/trajectory cursor snaps to discrete grid points.
- New utilities for .v5d files: v5dinfo, v5dstats, v5dedit, comp_to_v5d, and gr3d_to_v5d.

4.1 (May 1995)

New features:

- Rotated map projection.
- Improved widgets.
- Stored-frame animation.
- Better 3-D rendering in software using Mesa instead of VOGL.
- Vis5D files defined as a World Wide Web medium for exchanging model output.

4.2 (April 1996)

New features:

- Wind streamlines.
- Colored isosurfaces and trajectories.
- Scripting with Tcl.
- UVW variable widget.
- Pressure vertical coordinate system.
- programmer's API between Vis5D and its user interface.
- v5dSetLowLev function allows fields to occupy any sub-interval of vertical levels.
- physical units can be specified for each variable in a v5d file.
- v5dimport program.

4.3 (April 1997)

New features:

- Vertical sounding window.

5.0 (July 1998)

New features:

- Display multiple v5d datasets in one window.
- Display multiple v5d datasets side-by-side in multiple windows.

- Link user interface controls in 'groups' of windows.
- Type-in formulas can combine fields from multiple data sets.
- Open or create new datasets during a Vis5D session (v5dimport embedded in Vis5D).

5.1 (March 1999)

New Features:

- Slice Linking - slices can be linked together so that the status and movement of all the slices in the link are synchronized. Horizontal slices can not be linked with vertical slices.
- Vertical Slice Flipping - the two corners of a vertical slice are flipped as needed so that the contour numbers will never be draw backwards.
- Circular Clock - using the command line option `-circle` will draw a circle around the analog clock.
- Probe Var Selection - the number and type of variables can be controlled when using the probe. Usefule for data sets with many variables. Transparency in Legends - the legends now reflect the transparency of the color map.
- User Data - user specified formats can be used in loading grid data, topography data, and map data.
- Surface Graphics - horiz. contour slices, horiz. Wind slices, horiz. Stream lines slices, and the map can now be resampled to the topography and displayed on top of the topography.
- Horz. Slice Sliders - GUI sliders have been added to control the level at which slices are displayed.
- Changing Map Height - a user can select the level at which the map is displayed at. A GUI slider has been added to control this.
- Volume Rendering for PEX
- New Sounding Features - the look of the sounding has changed. Temperature lines can be displayed. The sounding data is also clipped at the level of the topograpy.
- TopoBase - a solid base can extend below the topography.
- Animation Dwell - a pause can be inserted between timestep=Numtimes and timestep= 0.
- Time Derivatives - the 'time' command can be used in the formula for creating new variables.
- More Formats in Saving Pictures - when ImageMagicks's convert program is installed more formats are accepted, and better save quality.
- Probe and Trajectory Can Be Coupled
- Offscreen Rendering

5.2 (December 1999)

New Features:

- Irregular Data and Text Plots

Notes

1. http://vis5d.sourceforge.net/NEWS.html

Chapter 10. Acknowledgments

The Vis5D project has benefited over the years from the contributions of countless organizations and individuals. We would like to express our sincere gratitude to all of them, but we'll have to make do with mentioning those few whose help has been recorded in the sparse logs.

First on any list, of course, must be the authors of the original Vis5D: Bill Hibbard, Johan Kellum, and Brian Paul. They and their generous sponsor, the University of Wisconsin-Madison's Space Science and Engineering Center[1] (SSEC), made this considerable project a reality long before "open source" was in vogue.

Other contributors to Vis5D include Andre Battaiola of CPTEC (Sao Paolo, Brazil), Dave Santek of SSEC, Marie-Francoise Voidrot-Martinez of the French Meteorology Office, Dave Kamins and Jeff Vroom of Stellar Computer Inc., Simon Baas and Hans de Jong of the Netherlands for the HP/VOGL port, Pratish Shah of Kubota Computer for the Kubota port, and Mike Stroyan of HP for the PEX support.

The Vis5d+ project has been aided by the efforts of Axel Reimann of Humboldt Univ. (who converted the manual to DocBook), Jim Edwards (many API cleanups, etc.), and Conrad Poelman of Stellar Science Ltd. Co. (improved v5dimport, etc.). Jim Edwards would like to thank Instituto Nacional de Meteorologia Brasil[2] (INMET) for its support of his work on Vis5d+.

Steven G. Johnson of MIT[3] would also like to extend a special thanks to his physics advisor, Prof. John Joannopoulos, for unstinting encouragement despite his student's dubious pastimes. Steven's work was supported in part by the Materials Research Science and Engineering Center program of the National Science Foundation under Grant No. DMR-9400334.

Finally, this project, like so many others, is deeply indebted to the free software community and especially the GNU project[4] for many invaluable tools and libraries, a free operating system to run them on, and inspiration not least of all. And VA Linux[5] deserves the last mention for graciously providing us with hosting on SourceForge[6], a service that we highly recommend.

Notes

1. http://www.ssec.wisc.edu/
2. http://www.inmet.gov.br/
3. http://web.mit.edu/
4. http://www.gnu.org/
5. http://www.valinux.com/
6. http://www.sourceforge.net/

Chapter 11. License and Copyright

Notes

1. http://www.gnu.org/copyleft/gpl.html
2. http://www.gnu.org/