# 1 Program to3d (batch mode)

## 1.1 Purpose

Program to3d converts 2D image files into 3D datasets for use with AFNI.

## 1.2 Usage

The command line format for program to3d is as follows:

**to3d** [**options**] **image_files ...**

## 1.3 Options

The available options are

**-help** show this message

**-type** declare images to contain data of a given type, where 'type' is chosen from the following options:

| ANATOMICAL TYPES | | FUNCTIONAL TYPES | |
|---|---|---|---|
| spgr | Spoiled GRASS | fim | Intensity |
| fse | Fast Spin Echo | fith | Inten+Thr |
| epan | Echo Planar | fico | Inten+Cor |
| anat | MRI Anatomy | fitt | Inten+Ttest |
| ct | CT Scan | fift | Inten+Ftest |
| spct | SPECT Anatomy | fizt | Inten+Ztest |
| pet | PET Anatomy | fict | Inten+ChiSq |
| mra | MR Angiography | fibt | Inten+Beta |
| bmap | B-field Map | fibn | Inten+Binom |
| diff | Diffusion Map | figt | Inten+Gamma |
| omri | Other MRI | fipt | Inten+Poisson |
| abuc | Anat Bucket | fbuc | Func Bucket |

Note: for paired (+) types above, images are fim first, then followed by the threshold (etc.) image files.

**-statpar value value ... value** This option is used to supply the auxiliary statistical parameters needed for certain dataset types (e.g., 'fico' and 'fitt'). For example, a correlation coefficient computed using program fim2 from 64 images, with 1 ideal, and with 2 orts could be specified with

```
-statpar 64 1 2
```

**-prefix name** will write 3D dataset using prefix 'name'

**-session name** will write 3D dataset into session directory 'name'

**-geomparent fname** will read geometry data from dataset file 'fname'. Note that geometry data does NOT include time-dependence

**-anatparent fname** will take anatomy parent from dataset file 'fname'

**-nosave** will suppress autosave of 3D dataset, which normally occurs when the command line options supply all needed data correctly

**-view type** Will set the dataset's viewing coordinates to 'type', which must be one of these strings: orig acpc tlrc

## 1.4 Time Dependent Datasets

**-time:zt nz nt TR tpattern OR -time:tz nt nz TR tpattern**
These options are used to specify a time dependent dataset.

**-time:zt** is used when the slices are input in the order z-axis first, then t-axis.

**-time:tz** is used when the slices are input in the order t-axis first, then z-axis.

**nz** = number of points in the z-direction

**nt** = number of points in the t-direction (thus exactly nt * nz slices must be read in)

**TR** = repetition interval between acquisitions of the same slice, in milliseconds (or other units, as given below)

**tpattern** = Code word that identifies how the slices (z-direction) were gathered in time. The values that can be used:

| | | | | |
|---|---|---|---|---|
| alt+z | = | altplus | = | alternating in the plus direction |
| alt-z | = | altminus | = | alternating in the minus direction |
| seq+z | = | seqplus | = | sequential in the plus direction |
| seq-z | = | seqminus | = | sequential in the minus direction |
| zero | = | simult | = | simultaneous acquisition |
| | | @filename | = | read temporal offsets from 'filename' |

For example, if nz = 5 and TR = 1000, then the inter-slice time is taken to be dt = TR/nz = 200. In this case, the slices are offset in time by the following amounts:

| | SLICE NUMBER | | | | | |
|---|---|---|---|---|---|---|
| tpattern | 0 | 1 | 2 | 3 | 4 | Comment |
| altplus | 0 | 600 | 200 | 800 | 400 | Alternating in the +z direction |
| altminus | 400 | 800 | 200 | 600 | 0 | Alternating in the -z direction |
| seqplus | 0 | 200 | 400 | 600 | 800 | Sequential in the +z direction |
| seqminus | 800 | 600 | 400 | 200 | 0 | Sequential in the -z direction |
| simult | 0 | 0 | 0 | 0 | 0 | All slices acquired at once |

If @filename is used for tpattern, then nz ASCII-formatted numbers are read from the file. These are used to indicate the time offsets (in ms) for each slice. For example, if 'filename' contains

```
0 600 200 800 400
```

then this is equivalent to 'altplus' in the above example.

## 1.5 Notes

- Time-dependent functional datasets are not yet supported by `to3d` or any other MCW *AFNI* package software. For many users, the proper dataset type for these datasets is '-epan'.

- Time-dependent datasets with more than one value per time point (e.g., 'fith', 'fico', 'fitt', 'fift') are also not allowed by `to3d`.

- If you use program `abut` to fill in gaps in the data and/or to subdivide the data slices, you will have to use the @filename form for tpattern, unless 'simult' or 'zero' is acceptable.

- At this time, the value of 'tpattern' is not actually used in any MCW *AFNI* program. The values are stored in the dataset .HEAD files, and will be used in the future.

- The values set on the command line can't be altered interactively.

- The units of TR can be specified by the command line options below:

  | | | |
  |---|---|---|
  | -t=ms or -t=msec | –> | milliseconds (the default) |
  | -t=s or -t=sec | –> | seconds |
  | -t=Hz or -t=Hertz | –> | Hertz (for chemical shift images?) |

  Alternatively, the units symbol ('ms', 'msec', 's', 'sec', 'Hz', or 'Hertz') may be attached to TR in the '-time:' option, as in '-time:zt 16 64 4.0sec alt+z'

## 1.6 Command Line Geometry Specification

**-xFOV <dimen1><direc1>-<dimen2><direc2>**
or

**-xSLAB <dimen1><direc1>-<direc2>**
(Similar -yFOV, -ySLAB, -zFOV and -zSLAB option are also present.)

These options specify the size and orientation of the x-axis extent of the dataset. <dimen#> means a dimension (in mm); <direc> is an anatomical direction code, chosen from

| | | |
|---|---|---|
| A (Anterior) | P (Posterior) | L (Left) |
| I (Inferior) | S (Superior) | R (Right) |

Thus, 20A-30P means that the x-axis of the input images runs from 20 mm Anterior to 30 mm Posterior. For convenience, 20A-20P can be abbreviated as 20A-P.

**-xFOV** is used to mean that the distances are from edge-to-edge of the outermost voxels in the x-direction.

**-xSLAB** is used to mean that the distances are from center-to-center of the outermost voxels in the x-direction.

Under most circumstance, -xFOV , -yFOV , and -zSLAB would be the correct combination of geometry specifiers to use. For example, a common type of run at MCW would be entered as

```
-xFOV 120I-S -yFOV 120A-P -zSLAB 60S-50I
```

## 1.7 Input Image Formats

Image files may be single images of unsigned bytes or signed shorts (64x64, 128x128, 256x256, 512x512, or 1024x1024) or may be grouped images (that is, 3- or 4-dimensional blocks of data). In the grouped case, the string for the command line file spec is like

| | |
|---|---|
| 3D:hglobal:himage:nx:ny:nz:fname | 16 bit input |
| 3Ds:hglobal:himage:nx:ny:nz:fname | 16 bit input, swapped bytes |
| 3Db:hglobal:himage:nx:ny:nz:fname | 8 bit input |
| 3Di:hglobal:himage:nx:ny:nz:fname | 32 bit input |
| 3Df:hglobal:himage:nx:ny:nz:fname | floating point input |
| 3Dc:hglobal:himage:nx:ny:nz:fname | complex input |

where

| | |
|---|---|
| '3D:' or '3Ds:' | signals this is a 3D input file of signed shorts |
| '3Db:' | signals this is a 3D input file of unsigned bytes |
| '3Di:' | signals this is a 3D input file of signed ints |
| '3Df:' | signals this is a 3D input file of floats |
| '3Dc:' | signals this is a 3D input file of complex numbers<br>(real and imaginary pairs of floats) |

| | |
|---|---|
| hglobal | = number of bytes to skip at start of whole file |
| himage | = number of bytes to skip at start of each image |
| nx | = x dimension of each image |
| ny | = y dimension of each image |
| nz | = number of images |
| fname | = actual filename on disk to read |

- The ':' separators are required. The k-th image starts at BYTE offset

  hglobal + (k+1)*himage + vs*k*nx*ny

  in file 'fname' for k=0,1,...,nz-1.

- Here, vs = voxel length = 1 for bytes, 2 for shorts, 4 for ints and floats, and 8 for complex numbers.

- As a special case, hglobal = -1 means read data starting at offset

  len - nz*(vs*nx*ny+himage),

  where len = file size in bytes. (That is, to read the needed data from the END of the file.)

- Note that there is no provision for skips between data rows inside a 2D slice, only for skips between 2D slice images.

- The int, float, and complex formats presume that the data in the image file are in the 'native' format for this CPU; that is, there is no provision for data conversion (unlike the 3Ds: format).

- Whether the 2D image data is interpreted as a 3D block or a 3D+time block depends on the rest of the command line parameters. The various 3D: input formats are just ways of inputting multiple 3D slices from a single file.

- The 'raw pgm' image format is also supported; it reads data into 'byte' images.

## 1.8 Notes (continued)

- Not all MCW AFNI programs support all datum types. Shorts and floats are safest. (See the '-datum' option below.)

- If '-datum short' is used or implied, then int, float, and complex data will be scaled to fit into a 16 bit integer. If the '-gsfac' option below is NOT used, then each slice will be SEPARATELY scaled according to the following choice:

  (a) If the slice values all fall in the range -32767 .. 32767, then no scaling is performed.

  (b) Otherwise, the image values are scaled to lie in the range 0 .. 10000 (original slice min -> 0, original max -> 10000.

  This latter option is almost surely not what you want! Therefore, if you use the 3Di:, 3Df:, or 3Dc: input methods and store the data as shorts, I suggest you supply a global scaling factor. Similar remarks apply to '-datum byte' scaling, with even more force.

- to3d now incoporates POSIX filename 'globbing', which means that you can input filenames using 'escaped wildcards', and then to3d will internally do the expansion to the list of files. This is only desirable because some systems limit the number of command-line arguments to a program. It is possible that you would wish to input more slice files than your computer supports. For example,

  ```
  to3d exp.?.*
  ```

  might overflow the system command line limitations. The way to do this using internal globbing would be

  ```
  to3d exp.\?.\*
  ```

  where the \ characters indicate to pass the wildcards ? and * through to the program, rather than expand them in the shell.

  (a) Note that if you choose to use this feature, ALL wildcards in a filename must be escaped with \ or NONE must be escaped.

  (b) Using the C shell, it is possible to turn off shell globbing by using the command 'set noglob' – if you do this, then you do not need to use the \ character to escape the wildcards.

  (c) Internal globbing of 3D: file specifiers is supported in to3d. For example,

  ```
  '3D:0:0:64:64:100:sl.\*'
  ```

  could be used to input a series of 64x64x100 files with names 'sl.01', 'sl.02' .... This type of expansion is specific to to3d; the shell will not properly expand such 3D: file specifications.

(d) In the C shell (csh or tcsh), you can use forward single 'quotes' to prevent shell expansion of the wildcards, as in the command

```
to3d '3D:0:0:64:64:100:sl.*'
```

The globbing code is adapted from software developed by the University of California, Berkeley, and is copyrighted by the Regents of the University of California (see file mcw_glob.c).

## 1.9 More Options

**-2swap** This option will force all input 2 byte images to be byte-swapped after they are read in.

**-4swap** This option will force all input 4 byte images to be byte-swapped after they are read in.

**-gsfac value** will scale each input slice by 'value'. For example, '-gsfac 0.31830989' will scale by 1/Pi (approximately). This option only has meaning if one of '-datum short' or '-datum byte' is used or implied. Otherwise, it is ignored.

**-datum type** will set the voxel data to be stored as 'type', which is currently allowed to be short, float, byte, or complex. If -datum is not used, then the datum type of the first input image will determine what is used. In that case, the first input image will determine the type as follows:

| | |
|---|---|
| byte | –> byte |
| short | –> short |
| int, float | –> float |
| complex | –> complex |

If -datum IS specified, then all input images will be converted to the desired type. Note that the list of allowed types may grow in the future, so you should not rely on the automatic conversion scheme. Also note that floating point datasets may not be portable between CPU architectures.

**-in:1** Input of huge 3D: files (with all the data from a 3D+time run, say) can cause to3d to fail from lack of memory. The reason is that the images from a file are all read into RAM at once, and then are scaled, converted, etc., as needed, then put into the final dataset brick. This switch will cause the images from a 3D: file to be read and processed one slice at a time, which will lower the amount of memory needed. The penalty is somewhat more I/O overhead.

**-orient code** Tells the orientation of the 3D volumes. The code must be 3 letters, one each from the pairs (R,L) (A,P) (I,S). The first letter gives the orientation of the x-axis, the second the orientation of the y-axis, the third the z-axis:

| | | | | | |
|---|---|---|---|---|---|
| R | = | right-to-left | L | = | left-to-right |
| A | = | anterior-to-posterior | P | = | posterior-to-anterior |
| I | = | inferior-to-superior | S | = | superior-to-inferior |

Note that the -xFOV, -zSLAB constructions can convey this information.

## 1.10 Options that Affect the X11 Image Display

These options only affect to3d if it is run in "interactive" mode (see the next section).

**-gamma gg** The gamma correction factor for the monitor is 'gg' (default gg is 1.0; greater than 1.0 makes the image contrast larger – this may also be adjusted interactively).

**-ncolors nn** Use 'nn' gray levels for the image displays (default is 80).

**-xtwarns** Turn on display of Xt warning messages.

## 1.11 Notes (continued)

- Different computers use different formats for storage of two byte (short) integers. Some computers store the most significant (high) byte first, followed by the least significant (low) byte. Other computers store the low byte first, followed by the high byte. It is *very important* that the image data files be converted to the correct format *before* running program to3d, or else the -2swap option should be used *inside* program to3d. For example, suppose that the 2D image files have been transferred from a RISC workstation to an Intel CPU computer. Since these computers use different formats for storage of short integers, it is necessary to swap bytes in the image files prior to using program to3d (by using program 2swap), or the -2swap option should be used when running to3d.

## 1.12 Examples

**Example 1.** Suppose that the user has a collection of 2D image files, consisting of 16 z-slices for 68 time steps, 5 seconds apart. The files fred01.0001, fred01.0002, ... , fred16.0068 contain the 2D image data, where the file format is: fred*ij*.*klmn*, where $ij$ = number of z-slice, and $klmn$ = number of time slice. The command line to create an *AFNI* 3D+time dataset is:

```
to3d  -epan -time:tz 68 16 5000 alt+z \
      -xFOV 120S-I -yFOV 120A-P -zSLAB 52L-53R \
      -prefix fred -session ../ \
      fred*.*
```

The command line options indicate that this is an Echo Planar (epan) dataset; this is a time dependent data set, with files ordered by time first, then by z-slice; there are 68 time slices and 16 z-slices, with 5 seconds between time slices. The order of the z-slices is alternating in the plus z-direction. The output, an *AFNI* 3D+time dataset, is to be stored in file fred+orig.BRIK (and .HEAD) in the next higher directory (../). The wild card characters '*.*' indicate that the data files all begin with 'fred'.