Mosaicking Distribution List (15 Total)

Name	M/S	Ext
Elaine Chapin	300-235	4-0497
Anhua Chu	300-243	4-0496
Eric J. Fielding	300-233	3-3614
Adam P. Freedman	300-235	4-9072
Scott Hensley	300-235	4-3322
John Holt	300-227	4-6341
Ian R. Joughin	300-235	4-1587
Yunjin Kim	300-243	4-9500
Jan M. Martin	300-235	4-0025
Francois Rogez	300-233	
Paul A. Rosen	300-235	4-0023
Eric Rignot	300-235	4-1640
Scott Shaffer	300-235	4-1217
Mark Simons	252-21 Caltech	395-6984
Paul R. Siqueira	300-227	3-9285
Jakob van Zyl	300-227	4-1365

JET PROPULSION LABORATORY TECHNICAL MEMORANDUM

334-97-004
January 30, 2000

TO	Jakob van Zyl
FROM	Elaine Chapin
SUBJECT	Preliminary Users Guide for the Suite of Mosaicking Code: Version 0.2

1 Abstract

To assist with training the Calgis representative in interferometric processing as part of the GeoSAR project, users guides have been written for the IFPROC interferometric processor, for the mosaicking software package, and for the multimosaic program. This document is the preliminary version of the Users Guide for the mosaicking package. Postscript and HTML versions of this document should be available shortly at `http://www-radar.jpl.nasa.gov/geosar/training/`

2 What is This Document?

The mosaicker written by Scott Shaffer and Scott Hensley mosaics together multiple data sets into a single image. The only thing preventing wider use of this software is that the documentation for it has not been written yet. This document is a preliminary stab at a users guide; it is an incomplete operational set on instructions and hints. This document reflects my involvement with this software as one of the "beta testers". Most of my experience has been with mosaicking TOPSAR data, but I will try to include the necessary information to assist in mosaicking other types of data. These comments apply only to running on `argus.jpl.nasa.gov`. I will try to add things as appropriate. If you find any errors or have suggestions, please tell me.

The term "mosaicker" is ambiguous since it can refer either to the `multimosaic` program or to the full suite of mosaicking software. Software in the mosaicker suite of code include:

- `ampflat`
- `ampfcheby`
- `diffcheby`
- `multimatch`
- `multicull`
- `multiaffine3d`
- `multimosaic`
- `multitps`
- `multicorner`

All executables are in `/users/topsar/Util2/`. `multimosaic` is discussed in the Users Guide for Multimosaic. Here I will try to outline the use of the other routines. The goal of the suite of code is to calculate an affine transformation for each pass that will allow the data to be mosaicked together correctly and smoothly.

3 The Big Picture: Overall Mosaicking Flow

Below is an outline of the necessary steps to make a mosaic.

Plan your Mosaic This step consists of accessing what radar and ground control data will be available to use in the mosaicking process.

Process Data Process the data and perform the usual quality assurance steps to verify the data has been processed correctly. (NB: Data output from the AIRSAR processor is usually "cranaled" using tie points as part of the quality assurance. If using data from the AIRSAR processor, request that they NOT tie point the data.)

Make Necessary Corrections to Strip Data Correct for known errors in the strip data, such as amplitude calibration problems or non-affine height errors in the data.

Make a Dead Reckoned Version of the Mosaic This step is to see where the potential problems lie, which strips might have problems, check for processing errors, etc.

Prepare Ground Truth Files Create the necessary match point files for ground control points which can not be recognized in the radar image, such as GPS ground control points.

Create Corner Reflector Tie Point Files Create the necessary match point files for points which can be recognized in the radar images, such as corner reflectors.

Do Matches between Data Runs Create tie point match files between data strips, so that the data strips match where they overlap.

Cull Matches between Data Runs Remove outliers from the tie point match files between data strips.

Begin Loop

- **Calculate Affine Transformations** Using all the tie point files created earlier, simultaneously solve for the affine transformation to be applied to each input radar data strip carefully choosing which parameters to solve for for each strip.
- **Mosaic Data Using Affine Transformations with Feathering Off**
- **Evaluate Quality of Output Mosaic** This is done using other DEMs, checking for size of the seams, calculating errors in corner reflector positions, etc.
- **If Quality of Mosaic High Enough, Exit Loop**
- **Adjust the Affine Transformation Parameters to Solve for**

End Loop

Remosaic Data into Final Output Format with Feathering On if Desired

4 Pre-Mosaicking Preparation

Before actually running any code, it is good to think through and plan out what you are going to do. The first thing to do is to make a list of all of the passes of data that you would like to mosaic together and to assign each of them a number starting with 1. For example, for the Long Valley data we had 6 passes and assigned them the following values:

Set Number	Pass Number	Orientation	Comments
1	090-4	90	Ping Pong Southernmost
2	090-1	90	Topsar South Middle
3	270-1	270	Topsar North Middle
4	270-4	270	Ping Pong Northern
5	270-5	270	Ping Pong Northernmost
6	090-5	90	Ping Pong Low Data Quality

It is also helpful to make a list of which passes overlap which other passes, since you are going to need to calculate match points for each of these overlaps. The table below lists the overlaps for Long Valley. For this example the matcher will be run 7 times.

Pass	Overlapping Pass	Pair
270-5	270-4	45
270-4	270-1	43
270-4	090-5	46
270-1	090-5	36
090-5	090-1	62
270-1	090-1	32
090-1	090-4	21

It is also useful to assess the ground control data available. Are there any corner reflectors, any kinematic gps data, any ground control points, any bodies of water of known height? Tie point files will need to be constructed between each set of ground control points and the strip data which contains the points. As an example, below is a list of Long Valley strips which contain kinematic gps data.

Pass	Pair
090-5	60
270-1	30
090-1	20
090-4	10

Below is a list of the Long Valley strips which contain gps ground control points. The Long Valley example contains no corner reflectors or large bodies of water which have a known elevation.

Pass	Pair
090-5	60
270-4	50
270-5	40
270-1	30
090-1	20
090-4	10

If you are not mosaicking TOPSAR files, you will also need to construct header files each pass of data. These headers are used by the mosaicking software and by `dgx`. A description of the format for the headers in given in the Appendix of the Users Guide for Multimosaic. After making your header file, it is recommended that you try to view the file with the header using `dgx`. As a rule of thumb, if `dgx` can correctly display the file and shows the correct latitudes and longitudes for points in the data, then the header file is correct.

5 Amplitude Adjustment

The amplitude data coming out of the IFPROC processor is not absolutely normalized. When mosaicking together multiple images, some may be much brighter or darker than others. To solve this problem, `ampflat` or `ampfcheby` needs to be run on each image. These programs renormalize the amplitudes to 1.0. Of course, if all the data that you are mosaicking is properly normalized you can skip this step.

To run `ampflat`, type the command `ampflat`. You will be prompted for the name of the rmg format input file, a name for the output file (typically named .mapf), and the image size in pixels inputting the number of range samples followed by the number of azimuth lines. Next you will be prompted for the "Start line, End line, and Skip". Usually these values would be 1, the number of azimuth lines, and the number of lines to skip between sampling the amplitude which is usually 100 or 200. Next you will be prompted for the Gradient removal (2) and the output files requested (2). In addition to the desired output file, `ampflat` also outputs two intermediate files called "gradients" and "fort.29"

To run `ampfcheby`, type `ampfcheby datafile samples ss es sl el skl deg outfile` where `datafile` is the input rmg formatted combined amplitude and height file, `samples` is the number of samples in the input file, `ss` starting sample over which to fit, `es` is the ending sample over which to fit, `sl` is the starting line over which to fit, `el` is the ending line over which to fit, `skl` is the number of lines to skip between lines used for fitting, `deg` is the degree of the Chebychev polynomial to fit for, and `outfile` is the output rmg file after the fitted Chebychev polynomial has been removed from the input data.

Running `ampfsepcheby` is similar, simply type: `ampfcheby datafile samples ss es sl el skl deg outfile [chebyfile]`. The arguments are the same as for `ampfcheby` except `datafile` and `outfile` are not rmg format but a separated real*4 amplitude file and `chebyfile` is an optional argument of a file containing the Chebychev coefficients to use instead of fitting. The `chebyfile` data is used if the file name is supplied and if `el` = `sl`. The `ampfcheby` programs output three ascii intermediate files, "fort.17" which contains the Chebychev polynomials, "fort.18" which contains the polynomial as a function of across track sample values, and "fort.19" which contains the data used for the fit.

`ampfsepcheby`, `ampfcheby`, and `ampfcheby` do similar things. The programs read in the data from lines `sl` + i `skl` where i = 1, (`el`-`sl`)/`skl`. They then fit that data with a polynomial in the case of ampflat or a Chebychev polynomial in the case of `ampfsepcheby` and `ampfcheby`. The fitted curve is then subtracted from the amplitude data over the whole input file so that the mean amplitude is 1.0 and output to the output file, which is in the same format as the input file. Before removing a 64 degree Chebychev polynomial from your amplitude strip data, you should consider what mechanism produced this artifact.

It is advisable to view the outputted flattened image when you are done. Note that `dgx` automatically scales the amplitude, so the image may not look any different.

The input to `ampflat` is an rmg formatted file. To combine separated real*4 amplitude (.mag) and height (.dte) files into one rmg file, use the command `nbymmaghgt`. Do not average down the data (acavg=1). `nbymmaghgt` output files are typically named with either the suffix map or rmg and the acavg factor (eg. .map1 or .rmg1). Similarly the command `separate` will separate an rmg formatted file back out into separated real*4 amplitude and height files.

The mosaicking software averages the amplitudes in overlapping regions. If the overlapping passes were flown at approximately parallel headings, this works well. However, if the passes have antiparallel headings, this produces a bad looking image since the bright areas of one image are the dark shadowed areas of the other. None of the programs described here will compensate for that effect.

6 Creating Match Points

The next step is to match the two data sets using the program `multimatch`. `multimatch` will either match the amplitude data for the height data. Amplitude matches work especially well in urban areas, and work poorly for perpendicular passes in areas with rough topography. Height matches work best in areas with rough topography. It is generally preferable to match the amplitude data for parallel and antiparallel flight line data. If the scenes cross at any other angle, you may have to match using the height data. It is generally best to try both and see which works better. The first step is to create a command file for `multimatch`. Below is a list of fields used in the command file.

In the command file you list two data sets to match. Order is important. The first data set that appears in the command file is the reference. The second data set is moved within the search window to match the first as well is possible. What is output is the offset needed from the dead reckoned position applied to the second data.

`rws` Reference window size in pixels, normally between 50 and 150, most commonly 64 for amplitude matches and 128 for height matches. This variable picks how big the window should be for comparing the two images. Window generally larger for height comparisons and smaller for amplitude comparisons. This field is also known as `rwsize`.

`skipxy` How many samples to skip and lines to skip between comparisons. These two values should be at least as big as the window size. Typically values are the same size as the window size across track and twice as big along track. The sample (x) and line (y) parameters can be called out separately using the `dlx` and `dly` options.

`strtxy` The across track and along track starting pixel numbers in the first image. Note this is not needed ; `multimatch` will calculate the overlap for you. The sample (x) and line (y) parameters can be called out separately using the `stx` and `sty` options.

`stopxy` The across track and along track ending pixel numbers in the first image. As with `startxy`, this is not needed. The sample (x) and line (y) parameters can be called out separately using the `spx` and `spy` options.

`srchxy` How much bigger then the window size the the pull in search window should be across track and along track. Typical values would be 3 to 10. Large search windows cause the code to run vary slowly. The x and y parameters can be called out separately using the `sdx` and `sdy` options.

`meanxy` The mean offset in pixels between the images two images being matched across track and along track. This parameter is usually 0,0 the first time multimatch is run. If a systematic shift is observed, the mean offset can be adjusted and the match rerun with a smaller search window. The x and y parameters can be called out separately using the `mdx` and `mdy` options.

`setnum` Choose to match amplitudes (1) or heights (2).

Other fields such as `typ`, `mss`, `dss`, `mbp`, `dbp`, `siz`, `off`, `spc`, `peg`, `fac`, `hdr`, `rmg`, `dte`, and `mgh` are the same as in the `multimosaic` program.

Note that I have not mentioned `mag`. Although that option exists, multimatch will produce no output file if these descriptors are used on one of the two input files. Multimatch wants to produce the full three dimensional match; if only amplitude data is available it has no way of accessing the heights. The usual work around if you have only amplitude data is to list the magnitude file as both the magnitude and height data file in multimatch. The heights are meaningless. Therefore when solving for the multiaffine transform it is important to fit only for transformations in the plane: x scale, y scale, z skew, z rotation, x translation, and y translation.

Below is a sample multimatch command file. This command file does an amplitude match of two Long Valley rmg format data files. The input files are lv270-5.mapf and lv270-4ec.mapf. The output file is mm54.out.

```
mm54.out
rws=128
skipxy=128,256
srchxy=10,10
meanxy=0,0
setnum=1
/data9/whitemtn/longvalley/lv270-5.mapf
hdr=/data9/whitemtn/longvalley/lv270-5.hdr
/data9/whitemtn/longvalley/lv270-4ec.mapf
hdr=/data9/whitemtn/longvalley/lv270-4ec.hdr
```

Having created a the input file, run `multimatch` giving the input file on the command line. `multimatch` will spawn three graphics windows: one showing the individual matches, one showing the matcher's progress through the file, and one control panel. `multimatch` will write some preliminary information to the screen before providing data about each match. Typical match information is given below.

```
257     19457    0.00   -1.00  0.672   0.042   0.011
385     19457    1.00    0.00  0.622   0.006   0.001
513     19457    1.00    0.00  0.743   0.007   0.003
1       19969    1.00    2.00  0.440   0.259   0.065
129     19969    1.00   -1.00  0.444   0.061   0.011
257     19969    1.00    0.00  0.671   0.019   0.006
385     19969    1.00    1.00  0.737   0.002   0.001
1       20481    2.00    1.00  0.283   0.097   0.020
129     20481    1.00    0.00  0.296   0.082   0.015
257     20481    1.00    0.00  0.353   0.136   0.007
385     20481    0.00    0.00  0.220   0.045   0.009
1       20993    1.00   -1.00  0.239   4.215   0.033
129     20993    2.00    2.00  0.212   0.521   0.034
257     20993    0.00    0.00  0.140   0.065   0.031
385     20993    1.00    0.00  0.413   0.002   0.002
```

The meaning of these columns is given in the table below:

Column	Description
1	Cross Track Position
2	Along Track Position
3	Cross Track Offset (Integer Pixels)
4	Along Track Offset (Integer Pixels)
5	Correlation Peak Value
6	Diagonal Term of Correlation
7	Other Diagonal Term of Correlation

```
                                        sch ; REFF Data file type
    19988           2000                    ; REFF Data file dimensions
     5.00           5.00                    ; REFF Post Spacing
```

```
  -49780.00       2935.00                ; REFF Starting corner position (s,c)
-17.8141790    145.9047043     11.1650472 ; REFF Peg position (WGS-84)


                                     sch ; SRCH Data file type
      19667          2560                ; SRCH Data file dimensions
       5.00          5.00                ; SRCH Post Spacing
  -50090.00       3275.00                ; SRCH Starting corner position (s,c)
-17.8290833    145.7144744   -168.9222099 ; SRCH Peg position (WGS-84)


584.00  328.00 760.173 18294.94 2310.41 730.993  -9.00 -4.00 1.0908 0.0119 0.0051 0.0043
584.00  456.00 720.278 18294.61 2183.41 688.238 -10.00 -4.00 1.0543 0.0160 0.0096 0.0054
584.00  584.00 734.732 18293.28 2054.41 707.578  -9.00 -3.00 1.0427 0.0094 0.0060 0.0047
584.00  712.00 712.159 18292.95 1925.41 684.236  -8.00 -3.00 1.1282 0.0052 0.0042 0.0019
584.00  840.00 730.232 18292.63 1798.41 702.564  -9.00 -3.00 1.1015 0.0038 0.0013 0.0000
584.00  968.00 685.687 18292.30 1671.41 657.721 -10.00 -3.00 1.0878 0.0094 0.0036 0.0006
584.00 1096.00 703.840 18291.97 1541.41 671.309  -8.00 -3.00 1.0346 0.0234 0.0075 0.0006
584.00 1224.00 738.741 18292.64 1414.41 706.929  -9.00 -4.00 1.0238 0.0015 0.0013 0.0004
```

Also it produces an output file, an excerpt of which is shown above. After 14 lines of header information about each of the files being matched, the file contains twelve columns of data. Columns 1-3 indicate the position in the first file at the match. Columns 4-6 indicate the position in the second file at the match. Columns 7 and 8 indicate the offset one would need to add to file 1 to make it match file 2. Columns 9-12 are diagnostics of the quality of the estimate. Those columns are described in the table below:

Column	Description
1	Along Track Position in First File (pixels)
2	Across Track Position in First File (pixels)
3	Height at Pixel in First File (m)
4	Along Track Position in Second File (pixels)
5	Across Track Position in Second File (pixels)
6	Height at Pixel in Second File (m)
7	Cross Track Offset to Add to First File to Get Second (Integer)
8	Along Track Offset to Add to First File to Get Second (Integer)
9	Signal to Noise Ratio
10	Diagonal Term of Correlation
11	Other Diagonal Term of Correlation
12	Cross Term of Covariance

The most common problem is that one data set is a significant distance from the geolocated position reported in the data set's header. If the data is too far off, the data that the matcher tries to match doesn't overlap. This causes noisy, erroneous matches. The solution to this problem is to run the matcher iteratively. Use `dgx` to help you calculate the approximate offset between the two data sets. Use that offset in the `meanxy` and run `multimatch` with a very large search window. Use the output of the first run to refine your estimate for the `meanxy` and rerun `multimatch` using a smaller search window. Repeat this process as necessary.

Another common problem is getting many "match on edge" conditions at either the beginning or the end of the data file. If the search window is too small, and at least one data set is rotated relative to its reported position, the mean offset will tend to vary systematically through the data set. Often the mean will walk out of the edge of the search window. The solution to this problem is to use a larger search window.

`Multimatch` is extremely flexible. It automatically handles matching data of different types (such at UTM and SCH), data with different post spacing (such as 5 m and 0.0008333 degrees), data in different formats (such as rmg and separated mag, dte), etc. Almost any image data that `dgx` can correctly view and geolocate, `multimatch` can match. If `multimatch` is not working the first thing to do is to display the images using `dgx` and check that the latitudes and longitudes that `dgx` reports seem reasonable.

7 Culling Match Points

Having run `multimatch`, the next task is to cull the data. First edit the `multimatch` output files eliminating all lines with fortran formatted write overflows (****).

`multicull` is run repeatedly for each `multimatch` output file. Each time that you run you cull out more points using ever more restrictive criteria. The output file of the last `multicull` pass serves as the input for the next. The command is called with two arguments: the input data file and the output data file. For example, `multicull mm54.out q` Once running you are asked to input five values. Input the MINIMUM acceptable signal to noise ratio and the MAXIMUM acceptable across track and along track covariances. I generally use a very low signal to noise threshold and try to choose the covariance thresholds just where the distribution is dropping off, to remove the tail. The final two inputs are the maximum acceptable residual values. The rule of thumb is to choose a maximum residual that is approximately twice the standard deviation. Do not choose a maximum residual of less than 1.0 for `multimatch`.

The goal is, after repeated culling, to produce an approximately flat histogram of across and down covariances and to have a residual standard deviation of less than 0.5.

8 Preparing Ground Truth

- `convert_ll_pix`
- `convert_ll_pix_old`
- `convert_datum`
- `multitps`
- `multicrtps` (NYI)

A distinction is made between a radar identifiable target, such as a corner reflector and a non-identifiable point. The data in the output file consists of six columns. The first three correspond to the "true"/surveyed position of the ground control point. The second three consist of the position of the target in the image. For corner reflectors, the later position consists the pixel location where the corner reflector was seen and the height at that pixel. For a non-radar identifiable target, `multitps` assumes that the planimetric of the strip data are perfect and outputs the pixel location where the target should have been and the height at that pixel. For TOPSAR data, which is rarely off planimetrically by more then a pixel, this works well. However, in steep terrain with a serious planimetric error, this procedure could present a problem.

`multitps` and `multicrtps` command files use the same options as multimosaic and multimatch with the following modifications.

`prj` Possible projections are `sch`, `llh`, and `eqa`. This option indicates the format for an ascii data file with of `typ=gcp`.

`aop` Indicates if the oversampled corner reflector position should be used (0) or the position of the nearest pixel (1).

`typ` In addition to the possible values for `typ` described in the Multimosaic Users Guide, the type can also be `gcp`, meaning an ASCII file consisting of three dimensional positions of ground control points.

A sample `multitps` command file is included below:

```
lvkgps6.gcp

rmg=/data9/whitemtn/mosaic/lv90-5cl.mapf

/u/argdat9/reference/ground_truth/gcps/longvalley.gcp.xyz
typ=gcp
prj=xyz
off=0,0
spc=1.,1.
```

9 Calculating Affine Transformations

9.1 What is the Multiaffine Transformation?

Multiaffine will fit for up to 12 parameters:

1. L_x, the x direction scale
2. L_y, the y direction scale
3. L_z, the z direction scale
4. K_x, the skew about the x axis
5. K_y, the skew about the y axis
6. K_z, the skew about the z axis
7. Q_x, the rotation about the x axis
8. Q_y, the rotation about the y axis
9. Q_z, the rotation about the z axis
10. T_x, the translation along the x axis
11. T_y, the translation along the y axis
12. T_z, the translation along the z axis

The x axis is the positive direction of lines in the file, the y axis is the positive direction of samples in the file, and z axis is the perpendicular to the other two. For example, for an input data file in the SCH coordinate system, the x axis is along track parallel to the peg heading, the y axis is across track perpendicular to the peg heading, and the z axis is vertical. See the Multimosaic Users Guide for more information on the order and sign for spacing, size, and offset for more details on the direction definitions. Specifically, the x axis is the U direction, and the y axis is the V direction in the nomenclature of that Section.

Each point in the data file has a three dimensional vector specifying its position. To apply the affine transformation, the input position vector, $\vec{I}$, is converted into the output positron vector, $\vec{O}$, using

$$\vec{O} = \bar{M}\vec{I} + \vec{T}, \tag{1}$$

where $\bar{M}$ is the affine transformation matrix, and $\vec{T}$ is the translation vector. These are related to the 12 fit parameters by the following equations:

$$\bar{M} = \begin{array}{l} \begin{bmatrix} L_x & 0 & 0 \\ 0 & L_y & 0 \\ 0 & 0 & L_z \end{bmatrix} + K_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + K_y \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} + K_z \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ +Q_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} + Q_y \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} + Q_z \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \tag{2}$$

and

$$\vec{T} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{3}$$

9.2 Prepare the Command File

The multiaffine command file contains two basic types of information: a list of which values to fit for and if not fit for what value to assign to that parameter and a list of all of the tie points files and information about which files each file's tie points apply to.

Multiaffine3d solves for up to 12*N parameters where N is the number of files to find affine transformations for. The parameter array is (12,N) in size. In the multiaffine command file you tell multiaffine two pieces of information about each parameter: should this parameter be fit for and what value should be assigned for this parameter. The fit indicator is either 0 (don't fit) or 1 (do fit). The fixed value can be any real number. Normally, it is set to 0.0 for the rotation, skew, and translation terms, and it is set to 1.0 for the scale terms. However, you are not limited to these values.

Although multiaffine3d can solve for up to 12*N parameters, you are very unlikely to want to solve for all of these. In particular, users should be very cautious about solving for both skew and rotation terms for the same direction, since the effects are very similar. If you have no ground truth, then one image should generally not be transformed at all. Otherwise the whole mosaic will tend to "walk away" since there will be no control points to prevent excessive image motion.

There are two ways to add comments to the multiaffine3d command file. Comments may be placed after the two real numbers on the lines where the different parameters are fit and fixed. In addition any text placed after a line with two zeros, such as "0 0 END", after the match file names will be construed as comments. Below is a sample command file. The file below was used to calculate the affine transformations for mosaicking some of the Long Valley TOPSAR data.

```
6           ; number of strips to mosaic together

0 1.   ; next 12 lines specify the characteristics
0 1.   ; of the multiaffine transformation to be
0 1.   ; applied to calculated for the first data
1 0.   ; strip.  The first column gives if the
0 0.   ; parameter should be fit for (1=fit, 0=don't fit)
0 0.   ; the second column gives the value to use for
0 0.   ; that parameter (eg. 0.0 for Z Rotation means
0 0.   ; no Z Rotation, 1.0 for Z Stretch means no stretch.)
1 0.   ; This system is very flexible, allowing the user
1 0.   ; to fix any parameter at any value the user wants
1 0.   ; or fit any parameter the user wants.
1 0.

0 1.   ; X Scale    information for file 2
0 1.   ; Y Scale
0 1.   ; Z Scale
1 0.   ; X Skew
1 0.   ; Y Skew
0 0.   ; Z Skew
0 0.   ; X Rotation
0 0.   ; Y Rotation
1 0.   ; Z Rotation
1 0.   ; X Translation
1 0.   ; Y Translation
1 0.   ; Z Translation

0 1.   ; Yes, the single blank lines between things
0 1.   ; are necessary!
0 1.
1 0.   ; strip 3 info
1 0.
0 0.
1 0.
0 0.
1 0.
1 0.
1 0.
1 0.

0 1.   ; Note the choices made for which parameters to
0 1.   ; fit for this particular data set.
0 1.   ; Strip    # of params fit   location
1 0.   ; -----    ---------------   --------
1 0.   ;   1          5             south end of mosaic
0 0.   ;   2          6             overlaps lots of kgps data
0 0.   ;   3          7             overlaps 3 other strips
0 0.   ;   4          6             second most northerly
```

```
1 0.   ;   5           6            north end of mosaic
1 0.   ;   6           5            think data is bad
1 0.   ; NB: no strip was fit for more then 7 parameters!
1 0.

0 1.   ; strip 5 info
0 1.
0 1.
1 0.
1 0.
0 0.
0 0.
0 0.
1 0.
1 0.
1 0.
1 0.

0 1.   ; strip 6 info
0 1.
0 1.   ; strip numbering is arbitrary, except
1 0.   ; "truth" is defined to be zero.
1 0.   ; in this strips 1 - 5 move from south to
0 0.   ; north.  strip 6, which has problems, was
0 0.   ; done last.
0 0.   ; You must list parameters for as many strips
1 0.   ; as you have numbered.  So if you match strip
1 0.   ; 1 to strip 6, but later decide not to use
1 0.   ; strip 6, you need to remove all strip 6 involved
1 0.   ; match point files.

2 1 ../match_data/mc21_sjs.out
3 2 ../match_data/mc32h_sjs.out
3 2 ../match_data/mc32a_sjs_sub.out
4 3 ../match_data/mc43.out
5 4 ../match_data/mc54.out
6 2 ../match_data/mc62.out
3 6 ../match_data/mc36h.out
1 0 ../gcps_data/lvkgps1.sub
2 0 ../gcps_data/lvkgps2.sub
3 0 ../gcps_data/lvkgps3.sub
6 0 ../gcps_data/lvkgps6.sub
1 0 ../gcps_data/lvgcps_lv1.dat
2 0 ../gcps_data/lvgcps_lv2.dat
3 0 ../gcps_data/lvgcps_lv3.dat
4 0 ../gcps_data/lvgcps_lv4.dat
5 0 ../gcps_data/lvgcps_lv5.dat
6 0 ../gcps_data/lvgcps_lv6.dat
6 0 ../gcps_data/wmgcps_lv6.dat
0 0 END
Comments: Long Valley data strips were matched to each other (mc*),
were matched to kinematic gps data (lvkgps*), and
```

`were matched to ground control points (lvgcps* and wmgcps*).`

9.3 Running Multiaffine3d

To calculate the affine transformations type: `multiaffine3d cmdfile outfile`
where `cmdfile` is the name of the multiaffine command file and and `outfile` is the root of the output file names. For example, if `outfile` were "affine", and you were solving for transformations for four data files, then the output files would be "affine_01", "affine_02", "affine_03", and "affine_04".

Points are weighted equally. If a given point is much more accurate than another, the only way to convey that information to multiaffine is to make duplicate copies of the point in the match files, to effectively increase the weight of that point.

In the multiaffine screen output, under the label "More Residual Analysis", the residual height errors are listed. The three components are given in the WGS-84 XYZ coordinate system in meters. The errors are NOT in the coordinate system of the input files. The mean error should always be close to zero, but the standard deviation information could be large.

10 Mosaicking the Data

This topic is covered in the Multimosaic Users Guide.

11 Limitations of the Mosaicking Software

Although the mosaicking software is very powerful, it doesn't do everything.

- Equiangular maps crossing 180° longitude, the international date line, are problematic.
- Any output map that contains the north or south poles in the mapped area and has an equiangular input file may have problems.
- The mosaicker correctly handles ground control points which are radar imageable targets. However, it does not handle ground control points which do not consist of radar imageable targets well. For more discussion of this, see Section XXX