

IDL Lab: Interpolation and Displaying of Lidar Data

The purpose of this lab is to introduce you to some IDL functions for interpolation and visualization. Lidar is becoming an important tool for getting high quality digital terrain (or elevation) data with one of the best vertical accuracies available. However these data points (or point clouds) were unevenly distributed spatially. In order to generate a raster from them or view them as a surface image, interpolation and gridding is required.

Interpolation is widely used in many areas when there is a need to infer values at unsampled locations from neighboring sampled ones. Many spatial analyses are based on this kind of operation. Today many interpolation methods are available, such as linear, bilinear, cubic, spline, kriging and triangular irregular network (TIN).

In IDL, "Gridding, a topic closely related to interpolation, is the problem of creating uniformly-spaced planar data from irregularly-spaced data. IDL handles this type of problem by constructing a Delaunay triangulation. This method is highly accurate and has great utility since many of IDL's graphics routines require uniformly-gridded data (from IDL Online Help)."

A list of IDL routines for gridding and interpolation functions (under the Mathematics functional category) is shown below. However if you need more complicated interpolation methods not found here, you can either develop one yourself or search it online: someone might have already done the work.

Gridding and Interpolation

[BILINEAR](#) - Computes array using bilinear interpolation.

[CONGRID](#) - Shrinks or expands the size of an array by an arbitrary amount.

[GRID_INPUT](#) - Preprocesses and sorts two-dimensional scattered data points, and removes duplicate values.

[GRID_TPS](#) - Uses thin plate splines to interpolate a set of values over a regular 2D grid, from irregularly sampled data values.

[GRID3](#) - Creates a regularly-gridded 3D dataset from a set of scattered 3D nodes.

[GRIDDATA](#) - Interpolates scattered data values and locations sampled on a plane or a sphere to a regular grid.

[INTERPOL](#) - Performs linear interpolation on vectors.

[INTERPOLATE](#) - Returns an array of interpolates.

[KRIG2D](#) - Interpolates set of points using kriging.

[MIN_CURVE_SURF](#) - Interpolates points with a minimum curvature surface or a thin-plate-spline surface. Useful with CONTOUR.

[POLAR_SURFACE](#) - Interpolates a surface from polar coordinates to rectangular coordinates.

[SPH_SCAT](#) - Performs spherical gridding.

[SPL_INIT](#) - Establishes the type of interpolating spline.

[SPL_INTERP](#) - Performs cubic spline interpolation (Numerical Recipes).

[REBIN](#) - Resizes a vector or an array to a set of given dimensions.

[SPLINE](#) - Performs cubic spline interpolation.

[SPLINE_P](#) - Performs parametric cubic spline interpolation.

[TRI_SURF](#) - Interpolates gridded set of points with a smooth quintic surface.

[TRIANGULATE](#) - Constructs Delaunay triangulation of a planar set of points.

[TRIGRID](#) - Interpolates irregularly-gridded data to a regular grid from a triangulation.

[VALUE Locate](#) - Finds the intervals within a given monotonic vector that brackets a given set of one or more search values.

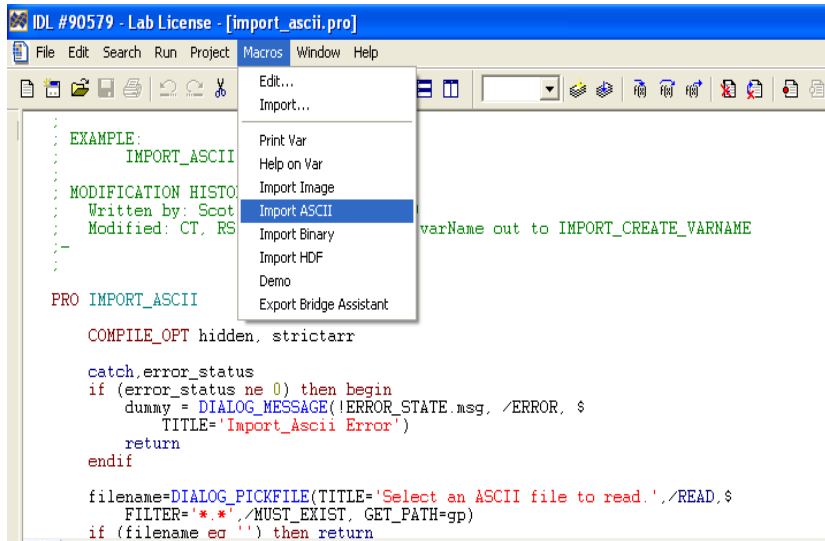
[VORONOI](#) - Computes Voronoi polygon given Delaunay triangulation.

Importing lidar data into IDL:

File "flat.csv" contains 20,000 data points, each has x, y (coordinate) and z (elevation) value. It has been filtered to represent the ground return points only. We will use this data for our experiment. This data are from Moscow Mountain.

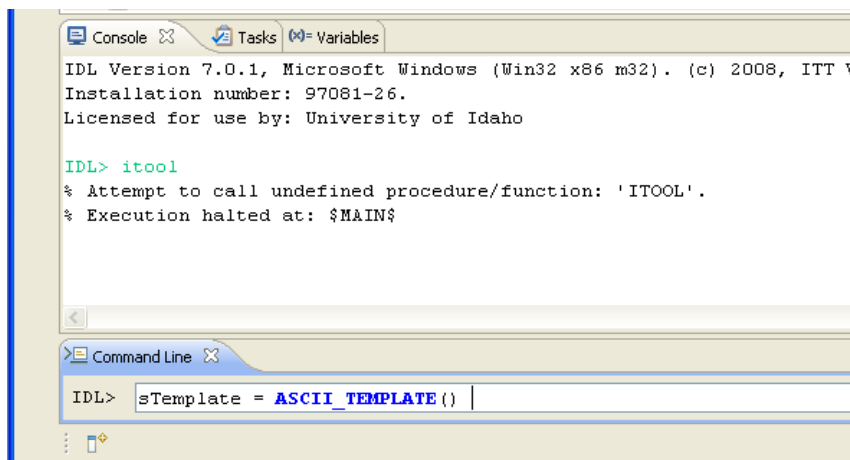
IDL provides some routines to read (import) different types of files such as ASCII or binary files. You can also write your own function to import files of various types/formats. Better yet, there are libraries that you can download to do this.

From IDL menu, Macros->Import ASCII and select the folder and file you want to open:



Or In IDL 7.0: From the IDL command line, type in

```
sTemplate = ASCII_TEMPLATE()
```



Then follow what you see on the screen to get to your data file.

Then in the ASCII Template Step 1, Data Starts at Line: change it from 1 to 2:

ASCII Template [flat.csv]

ASCII Template Step 1 of 3: Define Data Type/Range

First choose the field type which best describes your data:

☐ Fixed Width (fields are aligned in columns)

☒ Delimited (fields are separated by commas, whitespace, etc.)

Comment String to Ignore:

Data Starts at Line:

Selected Text File:

	X, Y, Z
1	
2	510816.37252200000,5353567.71601000000,664.96720000000
3	510818.45796700000,5353564.40415000000,664.88530000000
4	510817.96291700000,5353557.68865000000,664.76070000000
5	510817.79595100000,5353555.41555000000,664.78700000000
6	510812.50549000000,5353554.26313000000,664.15060000000
7	510814.25893100000,5353550.34996000000,664.59810000000
8	510818.15899200000,5353547.81917000000,664.38700000000

Click "Next" in step 2:

ASCII Template [flat.csv]

ASCII Template Step 2 of 3: Define Delimiter/Fields

Number of Fields Per Line:

Delimiter Between Data Elements:

☐ White Space ☐ Colon ☐ Tab

☒ Comma ☐ Semicolon ☐ Other:

Value to Assign to Missing Data: ☒ IEEE NaN ☐

Selected Records:

1	510816.37252200000,5353567.71601000000,664.96720000000
2	510818.45796700000,5353564.40415000000,664.88530000000
3	510817.96291700000,5353557.68865000000,664.76070000000
4	510817.79595100000,5353555.41555000000,664.78700000000
5	510812.50549000000,5353554.26313000000,664.15060000000
6	510814.25893100000,5353550.34996000000,664.59810000000
7	510818.15899200000,5353547.81917000000,664.38700000000
8	510817.12047700000,5353546.31337000000,664.04610000000

And change the names of fields1, fields2 and fields3 to x, y, z respectively; then click "Finish":

ASCII Template [flat.csv]

ASCII Template Step 3 of 3: Field Specification

	Name	Data Type
1	x	Floating
2	y	Floating
3	z	Floating

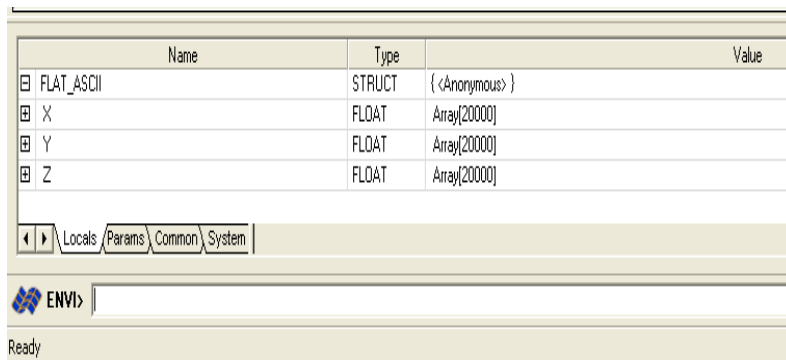
Name:

Type:

Sample Record:

	z
1	664.9672000

In the IDL Variable Watch window, you have imported x, y, z arrays into your local structure variable FLAT_ASCII.



Name	Type	Value
FLAT_ASCII	STRUCT	{ <Anonymous> }
X	FLOAT	Array[20000]
Y	FLOAT	Array[20000]
Z	FLOAT	Array[20000]

Locals Params Common System

ENVI

Ready

You can redefine x, y, z as

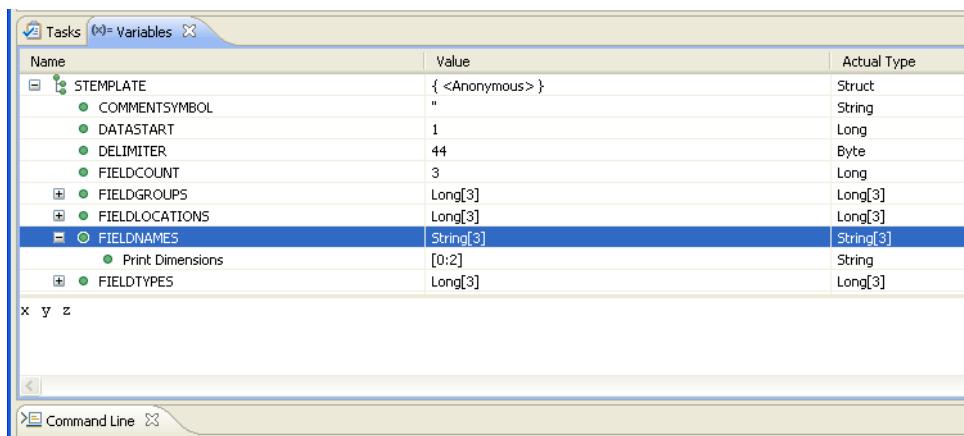
x = FLAT_ASCII.X

y = FLAT_ASCII.Y

z = FLAT_ASCII.Z

to make them more accessible.

In IDL 7.0, in the Variables window, there is a sTemplate structure created:



Name	Value	Actual Type
STEMPLATE	{ <Anonymous> }	Struct
COMMENTS	"	String
DATASTART	1	Long
DELIMITER	44	Byte
FIELD COUNT	3	Long
FIELD GROUPS	Long[3]	Long[3]
FIELD LOCATIONS	Long[3]	Long[3]
FIELD NAMES	String[3]	String[3]
Print Dimensions	[0:2]	String
FIELD TYPES	Long[3]	Long[3]

x y z

Command Line

In IDL 7.0, Use READ_ASCII to access the data, defined by the template created in the previous step:

```
data = READ_ASCII(FILEPATH('flat.csv'), $
SUBDIRECTORY=['for570', 'labn']), TEMPLATE = sTemplate)
```

The variable *data* now contains the ASCII data. Use the format *structureName.fieldName* to access individual fields of data. (Assuming your flat.csv file is located in C:\Program Files\ITT\IDL70\For570\labn. You may download the data file, uncompress it, and put it under the default IDL path. For570 and labn are two nested folders created under IDL70 folder).

Again, use the following three variables assignments to copy variables:

Name	Value	Actual Type
System		
DATA	{ <Anonymous> }	Struct
X	Float[20000]	Float[20000]
Y	Float[20000]	Float[20000]
Z	Float[20000]	Float[20000]
STEMPLATE	{ <Anonymous> }	Struct

x = data.x

y = data.y

z = data.z

Gridding and displaying lidar data

Now we will use triangulate function to display lidar data in surface and shaded surface.

TRIANGULATE, x, y, tr ;compute triangulation; tr holds the list of triangles in the Delaunay triangulation from x and y

; If you are on a TrueColor, set the DECOMPOSED keyword to the DEVICE command before running a color table

DEVICE, DECOMPOSED = 0

LOADCT, 3 ; load color table

WINDOW, XSIZE = 1024, YSIZE = 760 ;Open graphics window

!P.MULTI = [0, 2, 2] ;create 2X2 graphics panes

; Show linear surface with x axis rotate 60 degrees towards viewer

SURFACE, TRIGRID(x, y, z, tr), ax=60, skirt=600 ;default is 30 degree

; Show smooth quintic surface:

SURFACE, TRIGRID(x, y, z, tr, /QUINTIC), ax=60, skirt=600

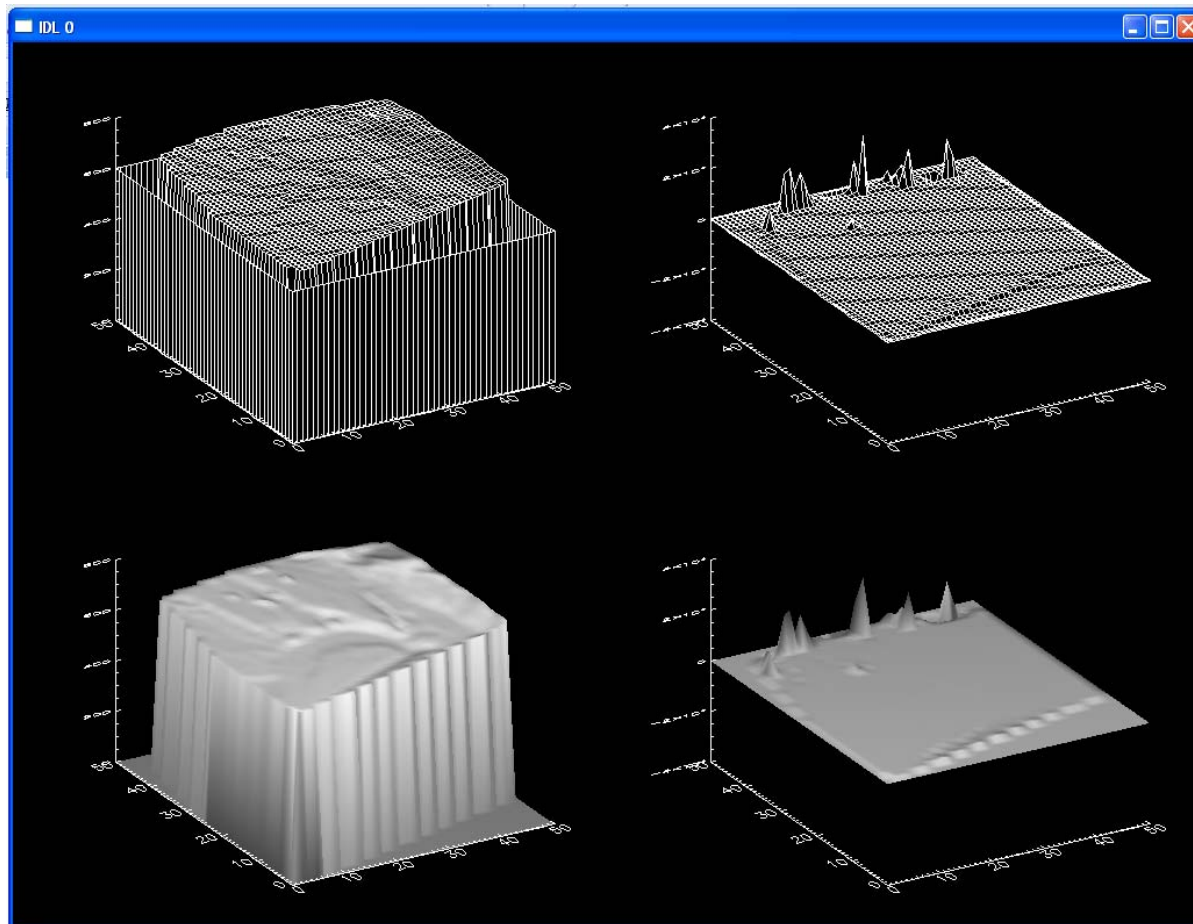
; Show linear shaded surface with x axis rotate 60 degrees towards viewer

SHADE_SURF, TRIGRID(x, y, z, tr), ax=60

; Show smooth quintic shaded surface:

SHADE_SURF, TRIGRID(x, y, z, tr, /QUINTIC), ax=60

!P.MULTI = 0 ;turn multiple panes off



Applying different interpolation methods using a fraction of data

In this exercise, I included a function called `read_csv_format_file` to read data directly from file. This function needs to be compiled first before running the following program. Then a list of random integers is generated; this integer list will be used as the indices for the data points to be involved in gridding. Although I plan to use 10% of data, due to duplicates in the data, usually a little less amount of points are used for interpolation. `GRIDDATA` function is used for inverse distance weighted (the default for this function) method; `GRID_TPS` function is called for thin plate spline method. The gridding results are shown in two separate windows.

```
;;IDL program for FOR570 ;;Harry Huang Jan 29,2009
```

```
;;read data from file; file is in the IDL path 'C:\Program Files\ITT\IDL64'; if it fails to read the file, put the whole path on:
filename = 'C:\Program Files\ITT\IDL70\for570\labn\flat.csv'
```

```
filename = 'flat.csv'
```

```
;;call the procedure to read csv file; read only one file each time
```

```
read_csv_format_file, filename, hdr, data
```

```
;;remember what we read in is string, change to float and reform
```

```
x = reform(float(data[0,*]))
```

```

y = reform(float(data[1,*]))

z = reform(float(data[2,*]))

;;change x,y values from 0 and up to make it easier to compute

x = x - min(x)

y = y - min(y)

total_pts = n_elements(x) ; total number of data points, i.e., 20000

pcnt = 0.1 ; 10% of data points will be randomly chosen for interpolation

pts_test = long(pcnt * total_pts) ; points chosen is 10000

print, 'number of interpolating points will be', pts_test

test_index = long(randomu (s, pts_test,/double)* total_pts ) ; generate random integer index between 0 to total_pts

print, 'dimension of test_index',n_elements(test_index)

test_index = test_index[sort(test_index)] ;;need to sort index before it can be used as subscript!

print,'dimension of the sorted test_index',n_elements(test_index)

intp_x = x[test_index] ;;points for interpolation

intp_y = y[test_index]

intp_z = z[test_index]

;check to cleanup points that are replicates

grid_input, intp_x, intp_y, intp_z, input_x, input_y, input_z

print, 'grid-input the # of chosen points for interpolation', n_elements(input_x)

;;use above points for interpolation and record how long it takes to interpolate

T = SYSTIME(1)

z_idw = GRIDDATA(input_x, input_y, input_z, dimension=[max(x), max(y)], start=0, delta=1) ;; inverse distance method

time= SYSTIME(1) - T

PRINT, 'IDW takes', time, 'Seconds'

T = SYSTIME(1)

z_tps = GRID_TPS(input_x, input_y, input_z, NGRID=[max(x), max(y)], START=[0,0], DELTA=[1,1]) ;thin plate spline method

time= SYSTIME(1) - T

PRINT, 'TPS takes', time, 'Seconds'

WINDOW, 0, XSIZE=800, YSIZE=600, TITLE='Interpolation using IDW'

```

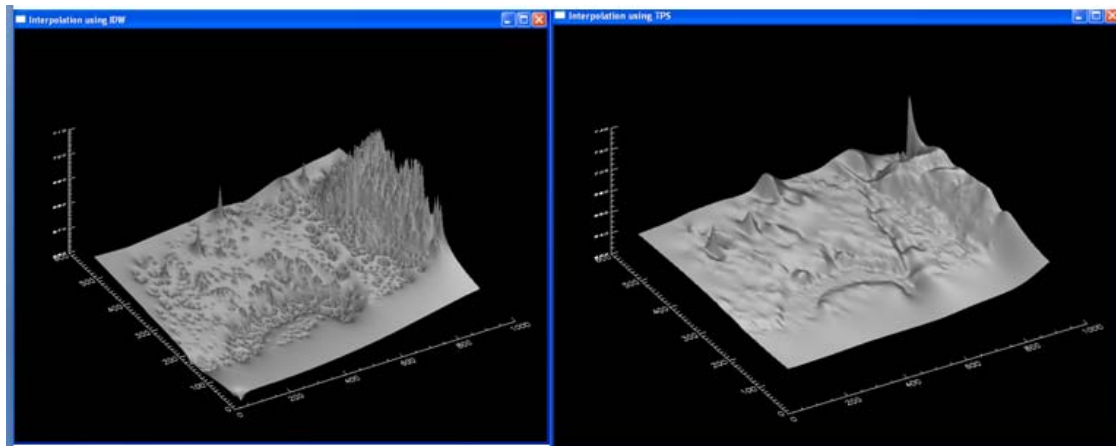
```
shade_surf, z_idw, charsize=1.5, ax=60
```

```
WINDOW, 1, XSIZE=800, YSIZE=600, TITLE='Interpolation using TPS'
```

```
shade_surf, z_tps, charsize=1.5, ax=60
```

```
end
```

While the computer is busy crunching numbers (I tested it with 50% of the data points first: in my desktop, it took more than ten minutes to finish TPS interpolation and less than two minutes by IDW, using around 8000 points; in GIS_lab, it took more than 30 minutes and 2 minutes, respectively; you might consider testing this using different percent of data points; but be warned: be patient - it might take longer than you think).



The images generated will resemble the above two diagrams.

Your task: How will you determine which interpolation method is better for your data? (hint: Evans and Hudak's 2007 IEEE paper from class reading list)

Installing LiDAR Tools for ENVI

Visit website at Idaho State University <http://geology.isu.edu/BCAL/tools/EnvTools/> for information on LiDAR Tools for ENVI.

More detailed directions on the functions and tools in this package can be found in either ISU website or ITT VIS (the company that developed the IDL) website at <http://www.ittvis.com/DownloadsHome/toolkits.asp>.

Your task: play with this tool using some of your own Lidar data or Lidar data downloaded from somewhere.