

Fetch Geometry Calculator Version 1.0 – User Guide

Scitech Environmental Consulting

Revised: November 12, 2014

Summary

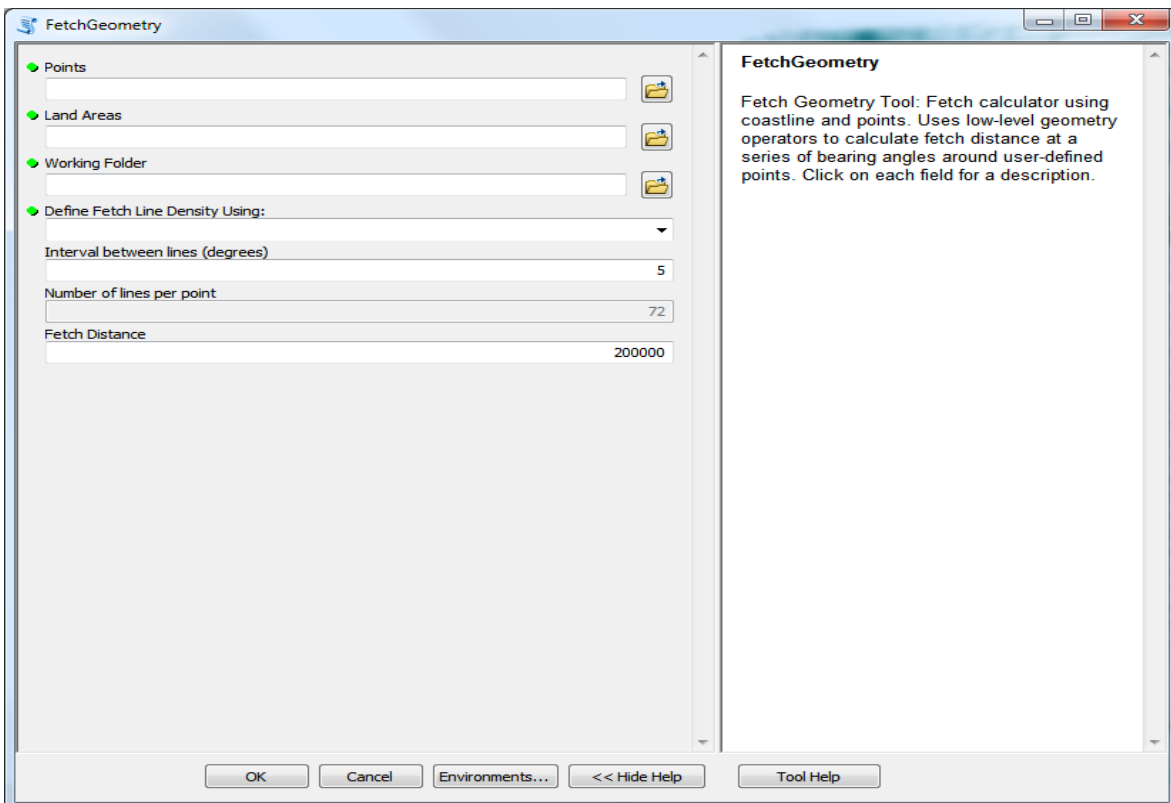
The Fetch Geometry Calculator is an ArcGIS geoprocessing toolbox with a Python script that calculates directional fetch for a specified set of point features, based on a specific set of land barriers. A typical application is predicting potential wind wave exposure at points along a coastline.

Installation

This tool is designed for ArcGIS 10.2 with Python 2.7. It may work with earlier versions, but performance is not guaranteed. To install, open ArcMap and the ArcToolbox window. Right-click on ArcToolbox at the top of the pane and choose Add Toolbox, then navigate to the toolbox file FetchCalculator.tbx. Ensure the associated Python script (FetchGeom.py) is in the same folder.

Operation

To run the tool, open the FetchCalculator toolbox (by clicking the plus sign next it) and double-click on the FetchGeometry script. This will open the input below, prompting for six inputs from the user:



The Fetch Geometry input window in ArcToolbox

Points: a feature class containing points for which fetch is to be calculated.

Land Areas: a feature class containing polygons that will limit the fetch distance.

Working folder: a folder where the tool can write temporary and output files. Do not specify a geodatabase because the script uses temporary csv files. The output will be placed in a geodatabase (Fetch.gdb), created in this folder.

Define fetch line density using: This dropdown list allows the choice between specifying the fetch line density using a bearing interval, or a number of bearing lines. One of the two options must be chosen.

Bearing interval or Bearing Lines: The interval in degrees (integer) between each bearing line, or the number of fetch lines desired for each point.

Fetch distance: the maximum distance over which fetch will be calculated, measured in projected map units.

Data preparation and performance optimization

Careful preparation of the input data will help to ensure the smooth operation of this tool. Please review the following cautions and tips on optimizing performance:

- The point and land area feature classes must have the same projection. If they do not, the script will stop with an error message.
- Input points and land areas can be either shapefiles or geodatabases, or some mixture of both. All output feature classes are written to a new file geodatabase.
- The script calculates fetch using the distance units in the projected map space. It will not prevent the use of lat-long (i.e., unprojected) files or other non-equidistant coordinate systems as inputs, but the results will be difficult to interpret.
- Any input points that are on land (i.e., within a land polygon) will have a fetch distance of zero for all bearings.
- We recommend some caution in defining the input points. While it is possible to use a set of points generated along a land barrier (e.g., a coastline), this is not ideal because limitations in precision may result in some points being "eaten" by the adjacent land polygon during processing, giving erroneous fetch distances. For this reason, we strongly recommend generating fetch points on a *small buffer* (e.g., one metre) away from the land areas.
- Internally, the script uses the low-level geometry operator "difference" to clip the fetch lines to the nearest land polygon. The performance of this operation is strongly influenced by both the number of and complexity of the land polygons. Thus, simplifying the land area as much as possible, and limiting the spatial extents to areas within the maximum fetch distance (e.g., given a fetch distance of 100 km, buffering the input points file to 100,000 m, and using that buffer to clip the land areas) will maximize performance.

- Land area polygons should be dissolved to a feature class that contains only one multi-part feature (using the Geoprocessing Dissolve tool with the create multi-part option enabled).
- Stress testing indicates that very occasionally, the "difference" operator will fail. We believe this is due to errors in the land polygon. We therefore strongly recommend running the "Repair Geometry" tool on the final land areas feature class prior to running the script. We also recommend removing polygon ZM geometries as these dimensions are irrelevant for this analysis and their removal will decrease processing time (simply use the Copy tool, with the Z and M geometries disabled). If any errors occur, these points will be written to a feature class called "GeometryErrors" in the output file geodatabase.

Processing Steps

The script begins by loading the input feature classes and ensuring 1) that points and land have the same projection; and 2) the land polygons are dissolved into one multi-part feature. It then creates data structures to hold temporary results, including "in_memory" feature classes to save time writing to disk. The script then processes the point file, one point at a time.

For each point, the script creates a series of bearing lines around the point using the bearing interval and fetch distance specified by the user. These fetch lines are stored in a low-level geometry object. The script then differences (erases) the fetch lines with the geometry of the land areas. This step can take some time for complex land areas with millions of vertices. After erasing, the script selects the fetch line segments that touch the original input point and discards the rest. The bearing and distance of each retained fetch line is recorded in a Python dictionary and stored as a csv file on disk. Fetch distances are rounded to the nearest integer in projected map units, and are saved to this file after processing each point to ensure the results are retained in case the process is interrupted.

The script writes a line in the ArcMap Results window for each completed point, noting the Point ID and the processing time, in fetch lines processed per second. (For example, if the bearing interval is 5 degrees, there are $360/5 = 72$ fetch lines for each point. A result such as "72 lines/sec" indicates that about one point is being processed every second.) Although it is difficult to anticipate how long the difference operation will take, it is closely related to the extent and complexity of the land polygon. Nevertheless, the processing time can be considered Order (N), where N is the number of input points. In other words, an input file with 1000 points will take about 10 times longer to process than one with 100 input points. The performance in lines/sec can be increased somewhat by specifying a smaller number of fetch lines per point, but this will not appreciably reduce the time required to erase the fetch lines.

Outputs

The script writes the resulting fetch points and fetch lines to the Fetch.gdb geodatabase, created in the user-specified working folder. Both feature classes are named with a date and time stamp of when the script started, in the format MMDD_HHmm.

The fetch point file is created upon completion of the script by joining the input point file to the fetch distances csv file and copying the result to the feature class FetchPoints_MMDD_HHmm. This is the file containing the final calculated fetch distances at each bearing for each point.

The fetch lines are retained for reference. Fetch lines are written to the FetchLines_MMDD_HHmm feature class every 10 points to ensure intermediate results are preserved. This feature class has an attribute *Point_ID* which can be linked to the *ObjectID* of the Fetch Points feature class (note that the fetch lines will not reliably link to the user-specified original input point file).

The tool uses an intermediate point file (InputPoints_MMDD_HHmm) to avoid corrupting the user's input data.

Calculating fetch statistics

The calculation of fetch statistics is straightforward, but can be rather tedious within ArcGIS given the large number of fetch lines. We therefore suggest doing such calculations in MS Excel, and using ArcGIS to join the results back to the Fetch Points. The steps include:

- 1) Export the attribute table of the completed FetchPoints feature class to a DBF, and open it in Excel.
- 2) Copy the ObjectID (or other key field) to a new spreadsheet.
- 3) In the new spreadsheet, create columns for the desired statistics.
- 4) Use Excel functions to summarize the desired fields from the fetch point DBF, ensuring the formula is correctly copied down all the rows.
- 5) Save this worksheet and close Excel.
- 6) In ArcGIS, load the new Excel worksheet and JOIN the data to the original point feature class using the ObjectID or other key field.

This approach avoids any potential problems arising from ArcGIS being unhappy with external table modifications, while overcoming the loss of DBF save functionality in Excel (apparently discontinued in 2007 and later versions of MS Office).

Running from the command line

The Fetch Script can be run from the command line, bypassing the need to open ArcGIS. If the ArcGIS python library is accessible, then the following command will work:

```
python FetchGeom.py "D:/Projects/inpoints.shp"  
"D:/Projects/land_area.shp" "D:/Projects/fetchWorkingDir" "Fetch Line  
Interval (degrees)" 5 72 200000
```

If Python is not accessible on the system path, then providing the full python path (e.g., below) should also work:

```
c:\python27\ArcGIS10.2\python FetchGeom.py "D:/Projects/inpoints.shp"  
"D:/Projects/land_area.shp" "D:/Projects/fetchWorkingDir" "Number of  
Fetch Lines per Point" 5 72 200000
```

Parameters template:

```
<input point file> <input land file> <working folder> <fetch interval  
definition> <bearing interval (degrees)> <number of fetch lines> <fetch  
length>
```