Tools, Tips, and Workflows

Breaklines, Part 3 — Z Conflation

B S A U | A W G

Lewis Graham
June 2013
Revision 1.0

A quick point before we begin - I try to write the technical article of each issue of LP360 News in such a way that it provides valuable information even if you are not a current user of LP360. With this aim in mind, much of the background material is presented in a software-independent way. Of course, we think LP360 is the best desktop LIDAR tool on the planet and will point out how our algorithms are implemented, but the overall information should prove valuable to all.

As we have discussed in the last two issues of LP360 News, breaklines are two- or three-dimensional graphic data (points, lines, polygons) that we introduce into an elevation model to alter the topology. When working with Geographical Information System (GIS) models, we nearly always model complex, irregular elevation data as a Triangulated Irregular Network (TIN). The source of the elevation points need not be from a LIDAR sensor. Surveyed mass points, scanned contour lines, correlated stereo models, and myriad other sources produce irregular elevation data that can be modeled as a TIN, and tools (such as LP360) that can manipulate random points are generally adept at processing these various data sources. Figure 1 is a point rendering of a typical airborne LIDAR scene that has been processed via automatic classification in LP360 (but not edited). Note that due to the high point density in this particular data set, features are recognizable even though the point cloud provides a fairly crude rendering.

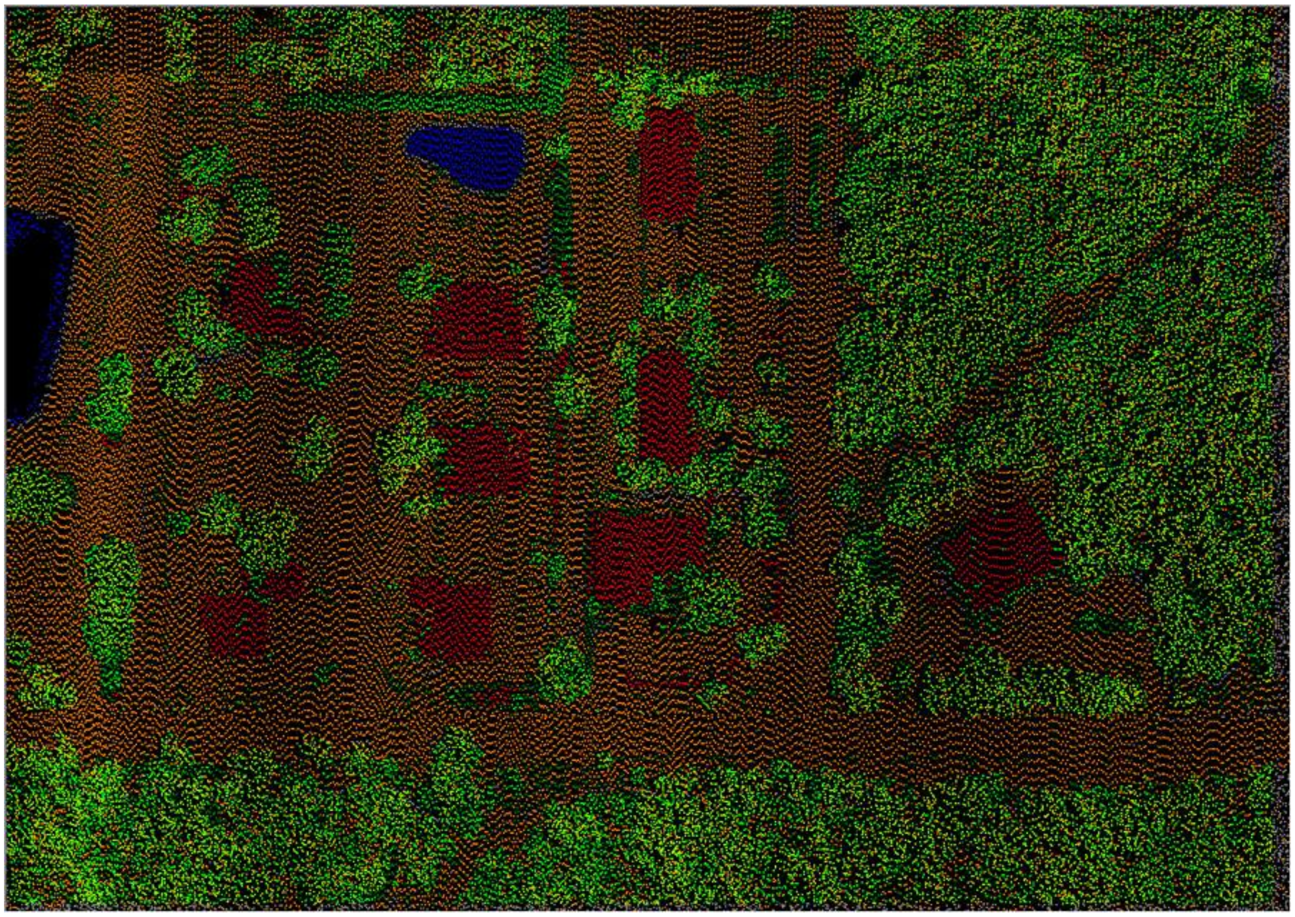

Figure 1: Points from a LIDAR Scene (data courtesy Southwest Florida Water Management District)

Recall from our previous discussions that a TIN is a convenient way to model point data because we can construct a solid surface (by filling in the triangle faces or facets) and include every point. No other geometry provides us this important capability. Note that we can even model “wrap-around” surfaces, such as the faces of tunnels with a TIN, so long as the TIN is a true 3D construct. Figure 2 is the same scene as Figure 1 but now rendered as a TIN. As you can see, the TIN generally provides a much richer viewing context.

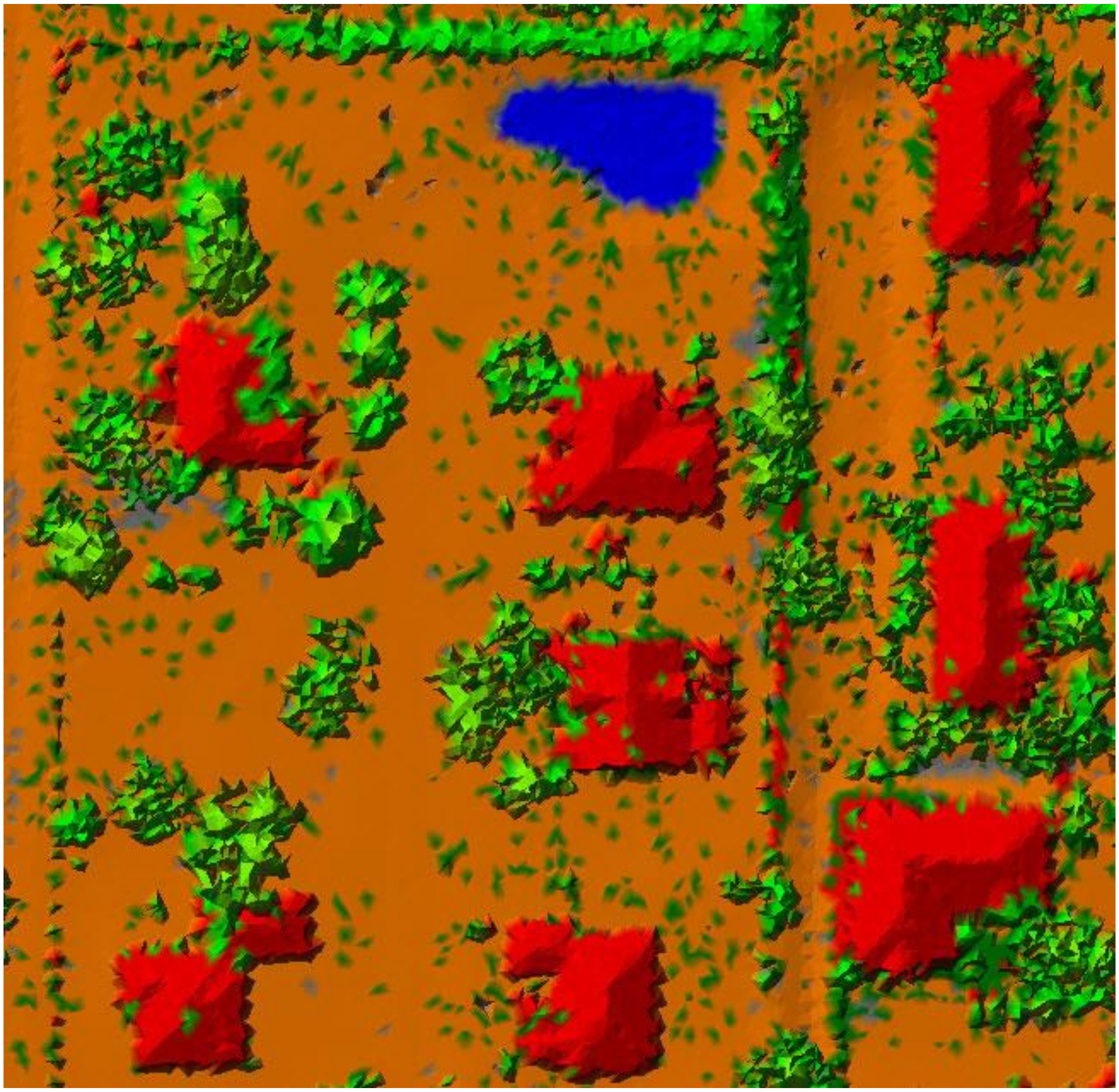

Figure 2: A TIN visualization

As previously discussed, breaklines allow us to modify the rendering of a TIN (as well as the extraction of 'derived' products from the elevation model). We discussed the idea of a 2-dimensional (2D) breakline in the first part of this series (April LP360 News). Recall that a 2D breakline, while potentially altering the node placements in a TIN, does not modify the elevation of the TIN. These 2D breaklines can be collected using a wide variety of methods from field surveys to heads-up digitizing from a digital orthophoto.

A three-dimensional (3D) breakline modifies the elevation of a TIN by introducing auxiliary "Z" points in a constrained manner. Before we dive into the details of what exactly is meant by this last sentence, let's consider how we obtain these 3D vectors. I will refer to graphical objects that are represented by nodes and lines as vectors, even though this is not technically the definition of a vector! This is a hold-over from differentiating computer displays based on raster or vector drawing techniques.

In order to enforce 3D constraints in an elevation model, we need 3D vectors. One of the huge advantages of LIDAR data is its very high vertical accuracy (15 cm or better vertical absolute accuracy is now commonplace in wide-area aerial mapping). Therefore, we can often use the LIDAR data themselves to enforce elevation models derived from the LIDAR data (this circular idea is the subject for another LP360 News session). A very common scenario is to digitize the planimetric (X, Y) portions of the vectors from a digital orthophoto and attribute the Z value from LIDAR. This process of creating a composite feature from multiple sources is termed "conflation." LP360 (both the ArcMap and Windows versions) contains a rich set of tools for conflating Z values onto 2-dimensional vectors and for densifying existing 3D geometries. For example, Figure 3 illustrates dynamic conflation using ortho photos for the planimetric capture and LIDAR for the Z component. Note that I am digitizing a lake feature in the Map View of LP360 while simultaneously viewing the vertical dimension in the LP360 Profile window. Not only does this allow me to much more accurately trace the shore line (by keeping the vertical white line in the profile view aligned with the shore) but the Z (vertical) component of my 3D shore line vector is being automatically populated from the LIDAR data via the Conflation process. This process is a very practical form of data fusion.

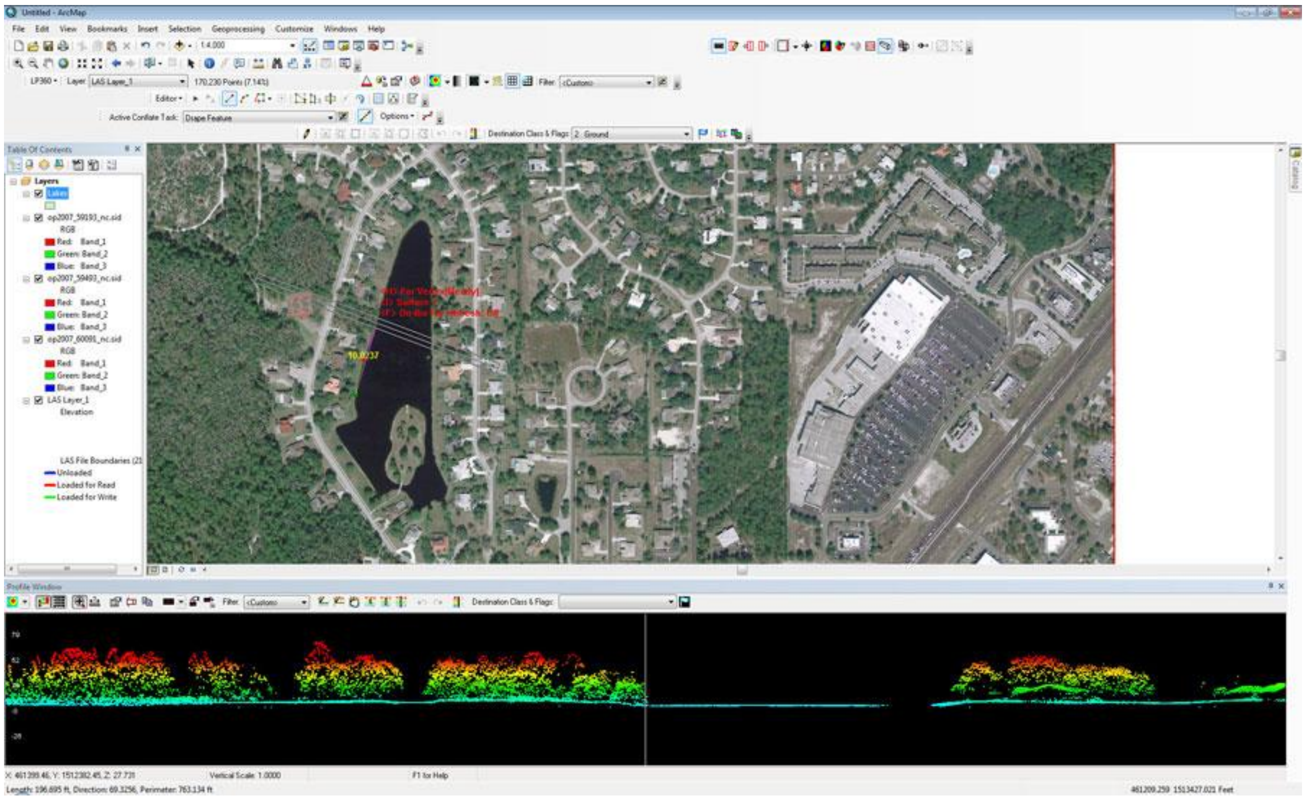

Figure 3: Dynamic Conflation using a high resolution ortho and high resolution LIDAR

By now a natural question should occur; how exactly are the Z values being computed from the LIDAR data? There are quite a few algorithms for doing this. LP360 supports a fairly robust set for dealing with common situations. As time goes forward, we will be adding more to the library.

Pure Drape – This is a common technique that is used when you want a 3D line to exactly follow the surface of a TIN. The line work is "draped" over the TIN and vertices are directly computed from the line, TIN intersection. LP360 supports a variety of population schemes - including keeping the original vector vertex spacing, inserting new vertices at a user-defined interval, or inserting vertices where the vector crosses TIN edges.

Downstream Constraint – This is a drape method that ensures that the vector monotonically decreases (or increases, depending on digitizing direction) from vertex to vertex. Thus, it is enforcing a "correct" water flow. Obviously this requires excursions above and below the TIN surface. LP360 allows you to control the maximum excursions so as not to exceed a surface error tolerance. Figure 4 provides an example of a downstream constraint. The ground points are depicted in yellow. The constrained vector created by LP360 is blue. Notice in the area circled in red that the constrained vector had to be moved below the LIDAR surface to enforce the monotonically decreasing rule. LP360 has triggers that will alert you if the deviation from the LIDAR surface exceeds a specified error tolerance.

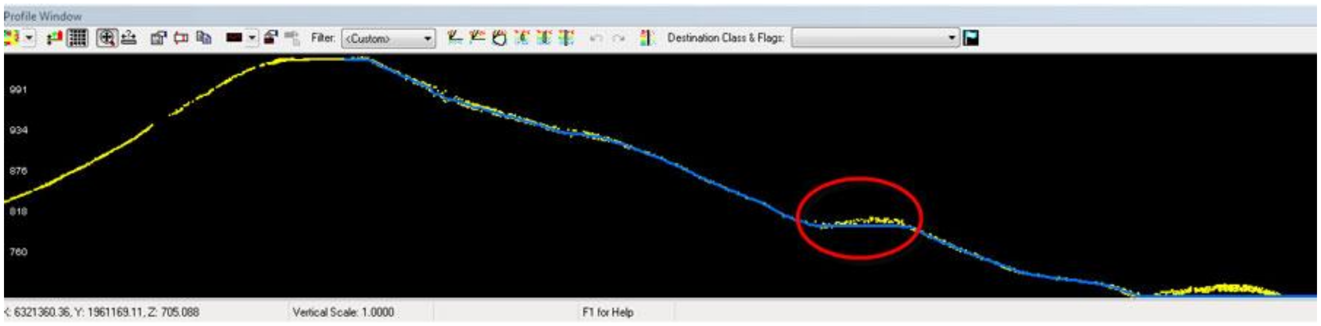

Figure 4: Downstream Constraint (ground is yellow, constrained vector is blue)

Constant Z – This mode computes a value for elevation and then applies this value to all of the vertices of the vector. Since a constant is being used, this value can also be stored in the attribute table associated with the vector. A constant Z algorithm is typically used for water body "flattening."

During the computation process, a number of different methods of computing the Z value for each vertex are supported. The method selected depends on the type of breakline you are creating (we will address this in a later edition) as well as the quality of the LIDAR data being used. We support a wide variety of options that comprise various combinations of:

Highest, Lowest Z – This method looks in a radial direction a user-specified distance from the vertex and locates the highest or lowest elevation. An example might be digitizing a ridge line in an area of grass. Since many LIDAR pulses will reflect from grass, a "lowest" Z algorithm might be appropriate.

Average Z – This method is similar to the previous method, except that all of the points within the user-specified radius are averaged and this average Z is used. This can be useful for LIDAR over hard surfaces where your desire is to either average out noise or average out local slope.

Closest Z – This selects the point closest to the vertex.

Surface Z – This method simply interpolates the Z from the TIN facet.

In addition to the above methods of computing Z, we also support a retaining wall method. This special method constructs, from a single input line, a set of two parallel lines a user-specified distance apart. These lines then have Z attributed based on both a high and low Z algorithm. The result is a line that represents the base of the wall and a second line that represents the top of the wall.

As you can imagine, there are all sorts of options with each of the above methods that allow you to tailor to many different situations.

Some of the more common breakline collection scenarios for which LP360 is used include:

- Water Body Flattening
- Hydro-correct streams (down-stream constraints)
- Double line drain (River flattening – a scheme whereby the river flows downstream but opposite banks are at the same elevation)

In this article I have focused on how a breakline is computed and also discussed some of the methods incorporated in LP360 for conflating elevation values. In the next edition, I will explore how these methods are applied in actual 3D breakline enforcement scenarios.