Advanced Computer Graphics (2020)
Faculty of Computer and Information science, University of Ljubljana

Seminar work report

Reconstruction of bridges in aerial LiDAR point cloud data

Nik Zupančič[1]

[1]Faculty of Computer and Information science, University of Ljubljana

Abstract

The development of technology and computational capabilities in recent years allows us to collect, manipulate and process large amounts of various data. ***LIDAR*** *is a technology that creates high resolution elevation models. Because the data is captured using the equipment attached to an airplane, certain parts of terrain, buildings or other structures can be represented inaccurately in the resulting point cloud.*

1. Introduction

The focus of this seminar is on the bridges in Lidar data. Because of the nature of collecting the data, a bridge in the point cloud is only seen as a single plane and when compared to a road we can't see much difference. Our goal is to obtain Lidar data and information about possible bridges within that area and then try and reconstruct the bridge to a more realistic shape by adding new points to existing point cloud.

2. Input data

We will use Slovenian LIDAR data provided by Slovenian Environment Agency ARSO which is available for download to public [evo]. The files used in the development are in GKOT LAZ format and D96TM projection. A single file contains a point cloud describing an area of 1 km^2.

Next we would like to identify location of the bridges within the point cloud. For that purpose a shapefile describing all roads in Slovenia is used. It is available for download at e-prostor [EPR], however free registration is required. Shapefile is a file for storing geometric location and attribute information of geographic features. We first read our input point cloud and then based on the maximal and minimal coordinates find information about bridges in the area inside the provided shapefile.

3. Bridge representation

Each bridge contains a bounding box which is given as a rectangle but due to its' inaccuracy we decided to instead use the point representation of the bridge. If the bridge is straight then only 2 points are given which are the start and end points of a vector which runs along the middle of the bridge. If bridge is not perfectly straight then it contains multiple segments which means 3 or more points are given. We will try and model all segments and not just a bridge in a straight line. One of the attributes that is also provided for each bridge is the width of a bridge. Width combined with the points of the bridge allows us to build our own rectangular representation. For simplicity we assume that each segment of the bridge is rectangular. While this means that it is not a perfect recreation especially in the case of curved bridges this will simplify our further work. The process to generating the rectangular bridge shape is trivial and can be seen in figure 1. The vector between 2 points is rotated by -90 and +90 degrees and scaled to obtain all 4 points of rectangle.

4. Reading point cloud

During the initialization phase we calculate 3 bounding points of a rectangle for each bridge. Because the size of point clouds can get rather large and searching for suitable points through all the points is inefficient, the whole cloud is only read once at the beginning of execution and only those points that are in the vicinity of the bridge are saved. This means that during reconstruction phase of each bridge only the points within the extended area of that bridge will be searched and used. Relevant points for a single bridge can be seen as red colored points in figure 2. A point is relevant to the bridge if it lies within the extended area of the bridge rectangle shape.
The test whether a point T lies inside a rectangle with point P_1, P_2, P_3 is done by calculating a scalar product of vec-

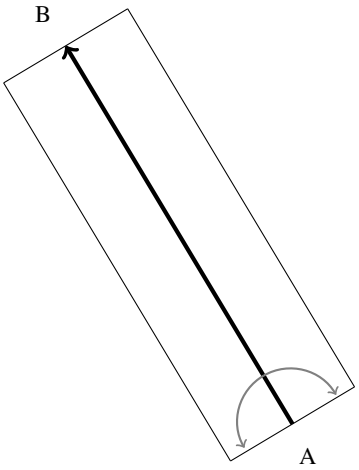

Figure 1: *Generating rectangular shape from vector 2 points and width information*

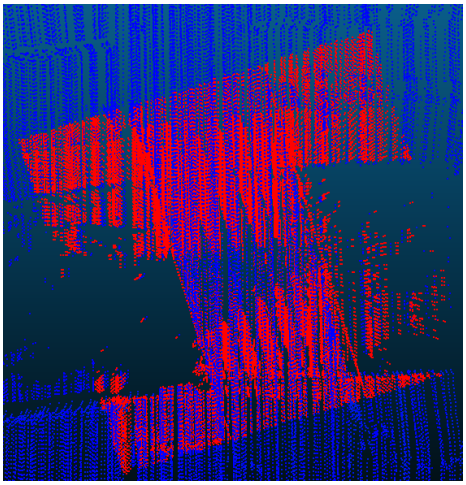

Figure 2: *The points used within reconstruction of a single bridge*

tors. If T is inside a rectangle then following must hold: $0 < \overrightarrow{P_1T} \cdot \overrightarrow{P_1P_2} < \overrightarrow{P_1P_2} \cdot \overrightarrow{P_1P_2}$ and $0 < \overrightarrow{P_1T} \cdot \overrightarrow{P_1P_3} < \overrightarrow{P_1P_3} \cdot \overrightarrow{P_1P_3}$

5. Reconstructing shoreline

First step of reconstruction process takes place below the bridge. The points of shoreline and river below the bridge do not exist in the input point cloud so we would also like to reconstruct the shoreline below the bridge. For this purpose we first extract points that are located left and right of the bridge but are still close enough to it, while their z-component is below the bridge surface. To do this we take a vector that runs along both sides of the bridge and calculate distance between the point and its' orthogonal projection on

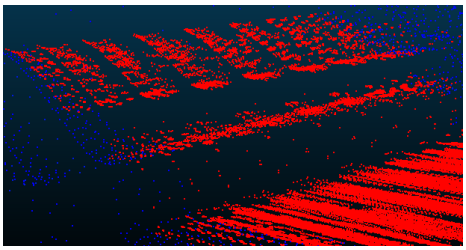

Figure 3: *Reconstructed shoreline with z restriction at 10 cm.*

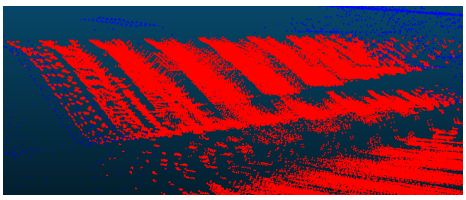

Figure 4: *Reconstructed shoreline with relaxed z restriction (100 cm).*

the vector. If the distance is within 2.5 meters the point is saved.
We have 4 bands of points that belong to the shoreline next to the bridge, each running along one side (left or right) on both ends of the bridge. Next the bands are connected using interpolation between its' points. However the 2 points are only interpolated if the distance between them is within a certain range of bridge width and the difference between their z-components is small enough. Initially the difference between z values of both points was required to be as small as 10 cm but because in some areas the points within certain heights are sparse this restriction was relaxed to 100 cm to ensure higher density of points in interpolated shoreline. The difference between results for more and less strict interpolation are demonstrated in figures 3 and 4.

To create a new path between 2 points (one in the left and one in the right band) we use a cosine interpolation [Bou99] or Hermite interpolation [Sal05]. Cosine interpolation requires only starting and ending point of the segment and is simpler than Hermite. When using Hermite interpolation we need to select 4 points for each segment, where 2 points will represent start and end of the segment and the other 2 points will influence the shape of the curve. During Hermite interpolation we use starting and ending point and their nearest neighbors in each step however this means that because searching for nearest neighbors has to be done in each step the interpolation process will also take longer. Figure 5 shows a reconstruction using only cosine interpolation and 6 a reconstruction using Hermite interpolation. For this reason cosine interpolation is used by default however it is possible to specify the use of Hermite interpolation with command

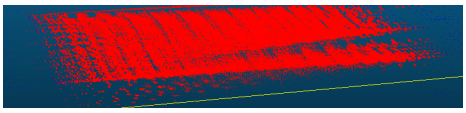

Figure 5: *Shoreline reconstructed with cosine interpolation.*

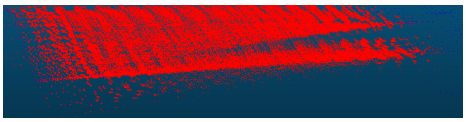

Figure 6: *Shoreline reconstructed with Hermite interpolation.*

line arguments when starting the script. (*see Github repository and readme*).

5.1. Removing outliers

During development we noticed that some noise appears in the bands along the bridge. These points clearly do not belong to the shoreline or terrain and influence the interpolation of shoreline points. Before starting the interpolation we take each band and remove the outliers. Because the bands only include few hundred points, we used a simplified method for removal. For each point in the band we compute average distance to its' 10 nearest neighbors and in the end remove all points whose average distance to nearest 10 neighbors is larger than $1.7 \cdot global\_average$ (note: 1.7 factor was obtained empirically).

5.2. Adding new band points

While outlier removal did remove some of the points that caused issues during interpolation, it also removed some of the points that could otherwise be useful in further process of reconstruction. Because of that and also because some parts of the shoreline are not as densely populated as the other, we additionally generate new points to reduce the influence of less populated areas on the shape and look of the shoreline. We follow these steps: first find the lowest and the highest point (based on value of z coordinate) in the existing band and form 2 new points by moving the lowest one to the water surface and the highest one just below the bridge's bottom face (their x and y position stays the same only their height changes). Finally we iterate through our list of points, connecting 2 consecutive points using Hermite interpolation. Number of new points on each segment is decided by density of existing points within the current band of points and distance between start and end of each segment so in the end all bands have approximately the same amount of points.

6. Reconstructing the bridge

After shoreline next step is the bridge reconstruction. As the shape of the bridge is only defined with previously mentioned vector(s) and width, the thickness of reconstructed bridge is proportional to its' length, meaning the longer the bridge the thicker it is. It is calculated as $thickness = length/25\ [cm]$ but bounded between 1 and 2.5 meters. Top face of the bridge is already present in the point cloud and the same points can be reused again to reconstruct the bottom face. As in initialization we again look for points within the rectangle shape of a bridge, only difference being the rectangle is not extended this time. These points are then added inside the point cloud with a modified height parameter, to place it on the bottom face of the bridge.
Because of the way a bridge is defined inside a shapefile that was used, it is possible that a part of the bridge will reach inside of the terrain and in that case we would like to limit lower face only to the area 'outside of terrain'. Since we already reconstructed the shoreline we create a vector that connects a point on left and a point on the right shoreline of the bridge. We do this for both sides of the bridge. Because we are only interested in 2D vector, the height is not important so we pick the points on the shoreline with the highest z component. We now have a vector along the shoreline on each side of the bridge and we can easily compute the center point of the bridge. We use that as a reference point and check whether this point lies to the left or to the right of the shoreline vector. If a new point will lie on the same side as center point of bridge for both vectors then this will be a valid point and we can add it to our point cloud. In case it lies on opposite side of either of the vectors then this means it would be placed inside the terrain and we will not be adding it to the point cloud. The process is visualized in figure 7, where the white arrow represents a vector and there are 2 points marked on the bridge. In the case of first one (point 1) this will be a valid point because it lies on the same side of vector as center of the bridge, while point 2 will not be used in reconstruction of lower face of the bridge because it lies on the other side of the vector and points inside that area would be created 'inside' terrain.

The height of the bridge is also interpolated by calculating the distance between starting point of the bridge and current position. Since we know the heights of starting and ending point we can compute the change of height δz in each step we make towards the end of the bridge. Step size is fixed at 40 cm. Depending on number of steps needed to the current position we then compute the height: $p_z = \#steps \cdot \delta z$. This is especially noticeable in ascending or descending bridges as seen in figure 8.

Sides of the bridge can not be reused but have to be newly generated. To do this we move along both sides of the bridge and at each step add a straight line of points at a fixed distance of 15 cm, connecting top and bottom face of a bridge. To prevent the side of the bridge reaching inside the terrain

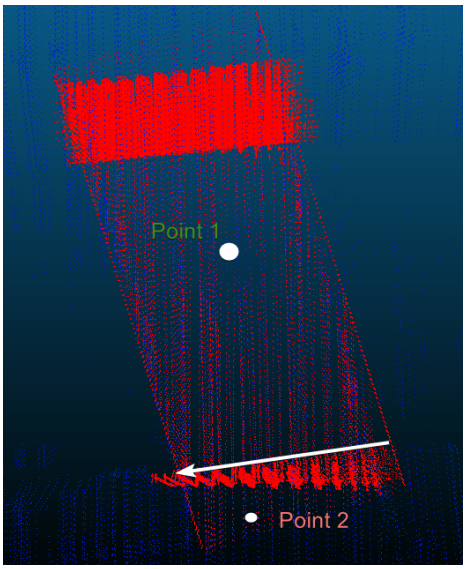

Figure 7: *Detecting whether a point lies to the left or right of the vector.*

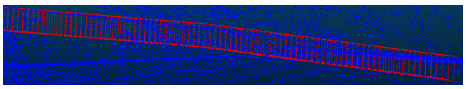

Figure 8: *Side view of a bridge with noticeable change in its' height.*

identical process as described in figure 7 by using center point of the bridge segment as a reference.

7. Results

The final point cloud with a reconstructed bridge contains added points for both sides of the bridge, bottom face and shoreline points as seen in red color in figure 9. While the bridge in resulting point cloud is closer to a real bridge than our input the result is still not ideal. The density of the points on the right shoreline of the bridge in figure 9 was much more sparse than the left side but by generating several new points this was equalized and closer to the more equal representation of points on both sides.

However despite the observed reconstruction there are some weakness to the proposed process. The first issue is the interpolation and generation of new points on the shoreline. If we look at the reconstructed shoreline from the front especially in the cases where the amount of points in original point cloud was small we can see that the generated points don't look very natural compared to the rest of point cloud 10. This could be solved using a better method for generating new points on both sides of shoreline so the points are distributed more evenly throughout the band instead of only following the path between existing points.

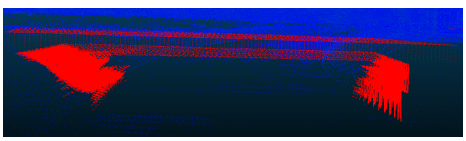

Figure 9: *Reconstructed bridge*

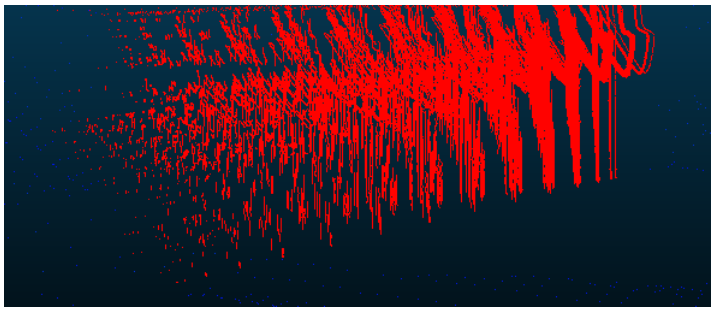

Figure 10: *Reconstructed shoreline with small amount of original points.*

Another issue was the currency of data. The shapefile data that was used was more than 10 years old while Lidar data was only a few years old. Due to this newer bridges could not be reconstructed. Another problem we encountered was incorrect classification of some roads as a bridge which caused bridge reconstruction in an area without a bridge. We were not able to identify whether there used to be a bridge in the past or it is just a classification error for that segment of the road.

8. Conclusion

In this seminar we developed a simple algorithm that takes Lidar point cloud and shapefile containing information about bridge as an input and using this data tries to reconstruct all the bridges along with the shoreline below each bridge and outputs a new point cloud with added points. Because the data is not always current there could be some incorrect or missing bridge reconstructions in the result. We focused on creating bottom face of the bridge based on some thickness together with the side surfaces and additionally fill the empty spaces in terrain below the bridge. We did not focus on reconstructing the water surface as such algorithms already exists and it was not part of our focus.

References

[Bou99] BOURKE P.: Interpolation methods. *Miscellaneous: projection, modelling, rendering 1* (1999). 2

[EPR] E-prostor. http://gis.arso.gov.si/evode/

profile.aspx?id=atlas_voda_Lidar@Arso&culture=en-US. Accessed: 8.5.2020. 1

[evo] Slovenian lidar data. http://gis.arso.gov.si/evode/profile.aspx?id=atlas_voda_Lidar@Arso&culture=en-US. Accessed: 8.5.2020. 1

[Sal05] SALOMON D.: *Curves and Surfaces for Computer Graphics*. Springer-Verlag, Berlin, Heidelberg, 2005. 2