Bridge Detection from Elevation Data Using a Classifier Cascade

Anthony Manfredi
amanfred1@swarthmore.edu

Alexandr Pshenichkin
apsheni1@cs.swarthmore.edu

Abstract

Bridges and similar non-obstructing features inhibit correct flow routing on high-resolution digital elevation models because their apparent elevation does not reflect the elevation at which water may pass underneath them. Our goal is to identify such features using the elevation data so that flow-routing algorithms may find paths under them correctly. We use an algorithm based on Viola and Jones' object-recognition system. Simple filters are applied in sequence to efficiently narrow the search space down to a final set of likely candidate features. This paper presents a successful system for identifying bridges that can be fairly easily integrated into existing GIS systems.

1 Introduction

New hi-res terrain scanning techniques such as laser altimetry (lidar) have greatly expanded the accuracy of GIS. The improved resolution has introduced many new details into digital elevation maps; many such features, however, hinder analysis of the underlying bare-earth terrain. One of the most important problem features are bridges. From the air, a bridge appears as a solid ridge, but, in reality, water can pass beneath it. While a raw data dump may contain some points that are visible underneath a bridge, current preprocessing techniques will tend to remove these, leaving a solid obstacle on the processed digital elevation model (DEM). This confuses flow-routing algorithms, which must flood terrain or search for convoluted detours to escape the local minimum created by the presence of the false ridge. Our goal is to identify bridges and similar features, such as drainage tunnels, on digital elevation models, so that water flow can be routed through them. Appropriate flow routing can be accomplished with minimal modification of existing algorithms by simply cutting through a bridge once it is marked out.

1.1 Related Work

Sithole and Vosselman (Sithole and Vosselman, 2006) describe a system for the geometric recognition of bridges as part of a general system for creating bare-earth data from raw lidar input. Their system looks for features that drop off sharply on two sides and fade smoothly into the surrounding terrain on the others. Calculating and analyzing bounding polygons for terrain features, however, is computationally intensive.

Our algorithm is inspired by computer vision research by Viola and Jones (Viola and Jones, 2002). Their system utilizes a "cascade" of simple filters, each of which is sensitive to a specific pattern. The algorithm reliably recognizes faces in real-time video. They also suggest a technique for fast computation of rectangle sums, called the integral image method. Each pixel in the integral image is the sum of the values of the pixels above and to the left of its location in the original image, which allows any rectangle sum to be computed with only four addition operations if the integral image already exists. This technique allows us to quickly calculate statistics for subsections of the map. For example, finding the average elevation in a ten-by-ten square area conventionally would require adding together one hundred values; with the integral image method, we need only access four values (the corners of the box) to get the area sum.

2 Methods

2.1 Algorithm

Our system is an implementation of the cascade concept of Viola and Jones in a novel domain. A sliding window moves over the map, examining small sections of the terrain in sequence. The window may move one or several pixels at a time: this is the step size of the window. A larger step size decreases runtime significantly but also decreases accuracy. Empirically we determined that a step size of 2 pixels did not result in a significant decrease in accuracy.

Each window is passed through a series of filters. A filter is a function that evaluates the pixels within the window statistically or geometrically and decides to accept or reject the slice. To save storage space, the algorithm applies all the filters to each window in order before moving on to the next; this way, no intermediate candidate lists (which could be quite large) are stored in memory. If any filter rejects the slice, it ceases to be relevant and the window moves to the next target. Like in the Viola-Jones algorithm, the collective action of the filters makes up for their individual inaccuracy. It is important for each individual filter to have a very low rate of false negatives, so that they do not reject good candidates prematurely.

In order to accommodate bridges of varying sizes, we make several passes over the map, changing the scale of the window each time. One can reasonably expect bridges to be at least one car lane and no more than a dozen lanes wide, and filters must take in some of the surrounding area for comparison as well. We are currently using window sizes of 100, 150, 200, 250, 300, and 400 feet in an attempt to accommodate all reasonably-sized bridges.

2.2 Filters

We have implemented several filters to detect bridge-like features. Since the overall goal is to aid hydrological modeling, we focus on discovering terrain elements that have a strong effect on existing flow-routing algorithms and trying to identify them as bridges.

1. The *high gradient* filter accepts an image if at least ten percent of the pixels in the filter window have a gradient above a certain threshold. Currently this threshold is 2.4 feet of elevation per 10 feet of translation (empirically determined), but we may adjust it in the future and analyze how it affects our results. This filter is designed to find the steep edges of bridges.

2. The *flood fill* filter accepts an image if at least thirty percent of the pixels in the filter window were flood-filled by a flow-routing algorithm. This filter is designed to capitalize on the fact that bridges in general, and particularly the bridges that we want to remove to do correct flow routing, cause flood filling along their length.

3. The *minimum fill depth* accepts an image if there is at least one pixel in the window that was flood-filled higher than 8 feet. This filter is designed to focus on areas that are significantly problematic for hydrological modeling.

4. The *low gradient* filter accepts an image if at least twenty percent of the window area is low gradient pixels, where the low gradient threshold is 0.5 feet of elevation per 10 feet of translation. This filter is designed to look for the flat area of the bridge itself.

5. The *minimum elevation difference* filter accepts an image if the difference in elevation between any two pixels in the window is above 7 feet. This filter capitalizes on the fact that bridges will be elevated above the surrounding terrain.

6. The *height bridge shape* filter accepts an image if a stripe down the middle third of the image matches a low:high:low elevation pattern when compared to the average elevation the entire window. This filter is rotated eight times at pi/8 radian intervals to catch varying bridge orientations. If any of these rotated filters match, the image is accepted. This filter is designed to find an elevation pattern that looks like a bridge: high in the middle and lower on the sides.

7. Much like the height bridge shape filter, the *gradient bridge shape* filter accepts an image if a stripe down the middle third of the image matches a high:low:high gradient pattern, using the same thresholds for low and high gradients that were used in the previous gradient filters. This filter is also rotated in the same manner as filter 6. This filter is designed to find a gradient pattern that looks like a bridge: flat in the middle and sharp on both sides.

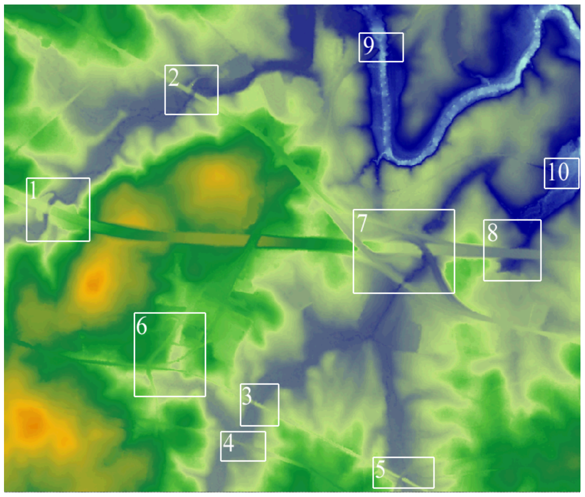

Figure 1: A DEM with hand-labeled features.

3 Results

We focused our testing on an area outside Durham, where Interstate 85 crosses highway 70. The combination of multiple roadways and a winding stream produce numerous interesting features to analyze.

Figure 1 shows a DEM of the area with hand-labeled features. Notable elements in this image are:

- Feature 1 (seen in detail in Figure 2) is a drainage pipe under a roadway. While shaped very differently from a bridge, it serves the same hydrological purpose, allowing water to pass beneath it.

- Features 2, 3, 4, and 5 are, very distinctly, bridges. Some (particularly 2 and 5) appear to have been precut. While it appears to be less pronounced than the others, Feature 4 has not been cut by the preprocessor, so it is of interest to us.

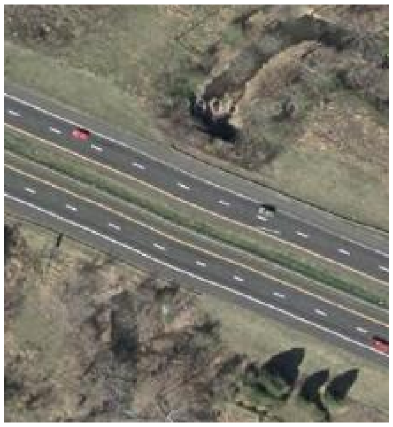

Figure 2: A drainage pipe under a roadway, corresponding to Feature 1 from Figure 1. Picture from Google Maps.

- Feature 6 is a set of small roadways, possibly with bridges.
- Feature 7 is an elevated interchange with a stream flowing underneath it. The exact pattern of flow is difficult to discern from lidar and satellite maps, but it is clear that part of this structure needs to be cut.
- Feature 8 is a roadway over an obvious depression. The sharpness of the cutoff between the road and the surrounding terrain indicates that the road is likely to be raised above the ground prominently here.
- Features 9 and 10 represent areas where a roadway seems to have been completely wiped out by the river, probably in the interpolation step. These are bridge-like features, but our algorithm should ignore them in the end.

Figure 3 shows the results of filtering our data set and grouping the selected locations using the built-in visualization tools in the GRASS software package. The algorithm clearly labels the large uncut bridge-like features in the image, such as the interchange and the drainage pipe, and avoids several pre-cut bridges. It correctly identifies the uncut bridge labeled 4, above, as a noteworthy feature, and isolates several small bridges in the tangle of elements labeled hand-labeled as feature 6. Features 9 and 10, already deeply cut, are ignored. Overall, the computer-generated map seems to capture all of the relevant features except for a few ambiguous parts of area 6, while ignoring already-cut bridges and generating fairly little noise.

Performing the feature extraction on this relatively small (609180 cells) map took 6m 45s. We expect computational complexity to be linear with respect to the

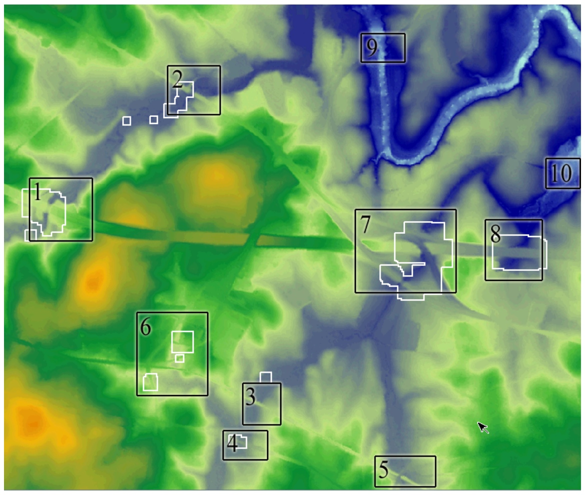

Figure 3: A comparison of hand-labeled (dark) and machine-detected (light) features.

number of cells. This bears out in practice: it takes 25 minutes to process a 21117520-cell grid with our system.

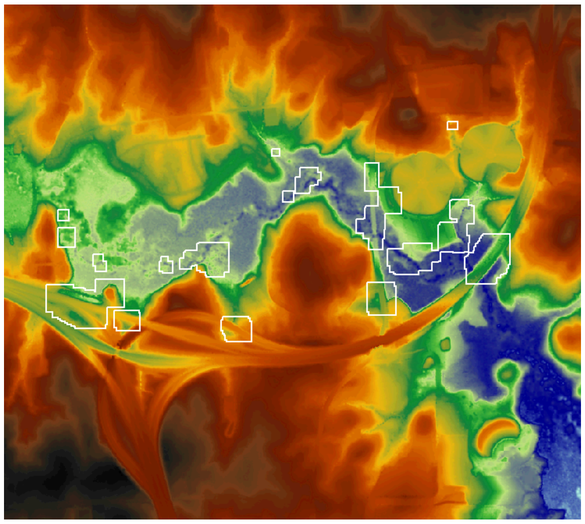

Figure 4: DEM of a road over a ravine in an urban area, demonstrating the shortcomings of the algorithm. Several bridge-like structures are correctly labeled, but these are also many false positives caused by trees in the ravine.

Figure 4 shows a less functional job. This time, the algorithm has identified a series of noisy-looking areas along what looks like a stream as being bridges, as well. Analysis of the image area reveals that the area isn't conventionally filled with water, however: the grainy lumps that have been labeled as bridge-like objects that impede the flow of water are actually trees in a ravine. While the image of all those areas being selected as good areas is rather unsightly, most represent terrain artifacts that can

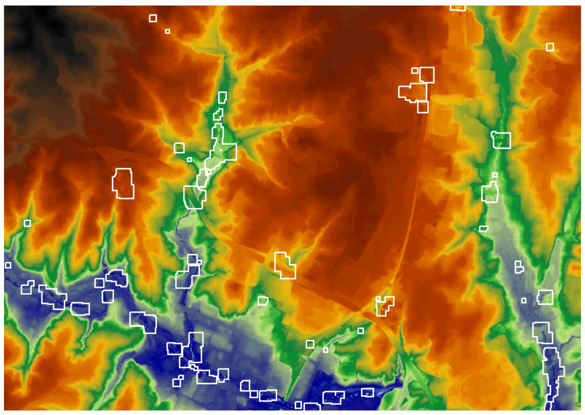

Figure 5: The bridge detection algorithm applied to a larger area. Note the tendency to over-select low areas when they are not uniform.

easily be cut; those that don't are actually major features.

Figures 5 and 6 demonstrate similar results with larger data sets. There are quite a few false positives overall, but numerous bridges are correctly detected.

4 Conclusion and Future Work

Overall, our algorithm seems quite effective in detecting bridges and similar features that impede flow routing on high-resolution DEMs.

Experimenting on other data sets, however, revealed that the algorithm is fairly sensitive to input error. While noise mostly just impairs its ability to detect useful features, errors that produce regular patterns will often lead to numerous false positives. This problem occurs because our program is searching for regular, mostly-linear features, which can be introduced into the image as artifacts during the various stages of preprocessing if the raw data is sufficiently poor. It should be noted that, while such results include a lot of false positives, most of those are clustered around image artifacts, so cutting through such areas shouldn't deform the actual map very much. Very few false positives generated by our algorithm are actually objects that would greatly affect the hydrological model if cut.

The computation time currently leaves something to be desired, however. While the integral image method speeds up the first few statistical filtering steps, we currently use naive techniques to find local extrema – these cost us a lot of time spent rescanning the same pixels as the window moves across the map. Finding the extreme value for a strip of data at a time and then simply taking the extrema of those could greatly speed up the execution of this stage. For the shape filters, a more computationally-efficient way to perform the required rotations would be ideal. Overall, the computational overhead of running the algorithm could probably be significantly reduced by running it as part of another window-sweeping algorithm and reimplementing it in C/C++.

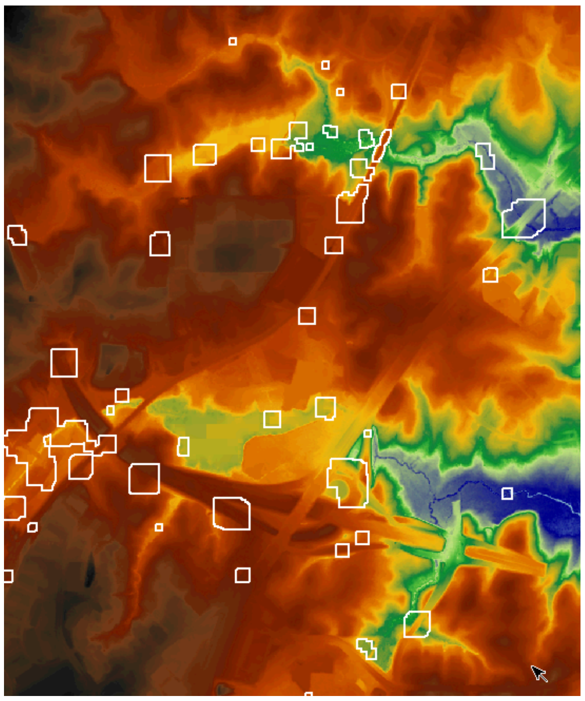

Figure 6: The bridge detection algorithm applied to a large area with a very complex road network. While the system is incapable of puzzling out the interchange, it does identify a large number of bridge-like features (and a few false positives).

It may be possible to get improvements in accuracy by running this algorithm iteratively with a bridge splicer, recalculating the flood fill depth after the removal of the current target bridge. Such a system would require a lot of repetitive computation, however.

Since most of our false positives seem to come from artifacts in the DEM, we believe that simply cutting the regions identified by our algorithm, even if the detected feature is not a bridge, will improve flow-routing.

References

G. Sithole and G. Vosselman. 2006. Bridge detection in airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 61(1):33–46, October.

Paul Viola and Michael Jones. 2002. Robust real-time object detection. *International Journal of Computer Vision - to appear*.