Voronoi Natural Neighbors Interpolation

Chris Harman
charman1@cs.swarthmore.edu

Mike Johns
mjohns2@cs.swarthmore.edu

Abstract

Implementing point cloud to grid conversion for digital elevation maps (DEM) presents one with many options for interpolation and we intend to explore algorithms for interpolation during the conversion process with a specific focus on Voronoi natural neighbor interpolation. By partitioning the environment into Voronoi cells and using the information from neighboring cells and their respective generating point's elevation, we can achieve aesthetically pleasing interpolation results with runtimes competitive with lower-order interpolation algorithms. We compare our natural neighbors interpolation method with a linear interpolation method and a regularized spline interpolation method quantitatively using cross-validation and qualitatively by rendering the interpolted meshes using GRASS. [1]

1 Introduction

When dealing with discrete sets of elevation data such as elevation information collected using methods such as laser range finding or LIDAR, it is often desirable to be able to estimate or predict the values of unsampled locations using the available information. One application where interpolation can be useful is visualization of elevation data. Point elevation data obtained through LIDAR or other remote sensing methods is not always uniformly sampled. One way to visualize the data set would be to simply triangulate the point set into a triangular mesh and render that mesh, but the resulting image would not model the underlying data well. By using different interpolation methods on the original elevation data, more accurate rendering can be performed and depending on the method of interpolation, the resulting visualization can change dramatically.

There are many interpolation methods available to estimate these unsampled values, and the different methods usually offer tradeoffs between the accuracy of the prediction and the efficiency of the computation. One of the simplest interpolation methods would be to assign each unsampled location a value based upon which location within the set of known values it lies closest to. Although this simple method is computationally efficient, the resulting function is not continuous everywhere, specifically along the edges where locations are equidistant from the samples. In most cases these discretized estimations will not be as accurate as the results from other more computationally expensive interpolations. The discontinuities in the resulting function also do not model the sampled data in an aesthetically pleasing manner. Methods such as interpolation by regularized spline with tension result in functions which have regular derivatives of all orders everywhere allowing for analysis of surface geometry as well as improved accuracy in estimation (**?**).

Natural neighbors interpolation provides a good tradeoff between computational efficiency and accuracy. In this method the value of an unsampled point is determined through a weighted average of the values of the interpolation points neighbors within the sample set. These natural neighbors are determined by finding which Voronoi regions from the original point set would intersect the Voronoi region of the interpolation point, if it were to be inserted. The resulting function is continuous everywhere within the convex hull of the sample set and mimics a taut rub-

[1] http://grass.itc.it/

ber sheet being stretched over the data. Our hope is that Natural Neighbors Interpolation will provide a computationally efficient method with which to accurately visualize elevation data.

2 Related Work

There are an overwhelming number of options when choosing which method to use for interpolation when converting from a point cloud to a grid for a DEM. (**?**) details many low-level routines for interpolation including: the level plane; linear plane; double linear; bilinear; biquadratic; Jancaitis biquadratic polynomial; piecewise cubic; bicubic; and biquintic interpolation method. (**?**) draws the conclusion that higher- order interpolation methods will always outperform those with linear complexity in terms of modeling the terrain accurately. The problem with choosing higher-order interpolation routines is that the computational backlash is not necessarily proportional to the gain in accuracy; this begs the question: How do we balance modeling accuracy and compuational efficiency?

Other interpolation methods such as the regularized spline with tension described in (**?**) attempt to balance computational efficiency and modeling accuracy. The regularized spline with tension deliberately attempts to smooth the surface being iterpolated and tweaks the mesh appropriately. Though a little less straightforward than other lower-level methods of interpolation, this method is a good benchmark for performance and modeling accuracy; as such, we will use it as a comparison for our Voronoi natual neighbors method in this paper. We expect Voronoi natural neighbors interpolation to represent a desirable balance between computational complexity and aesthetics.

3 Methods

3.1 Computing the Delaunay triangulation

The Delaunay triangulation for a set of points P is a triangulation DT(P) such that no point in P is within the circumcircle of any triangle within the triangulation. This is also the triangulation which maximizes the minimum angle within the triangulation. DT(P) is computed using a randomized incremental approach as outlined in (**?**). Each point in P is inserted incrementally and the Delaunay triangulation for that set of points is computed. Initially the triangulation consists of a single triangle which is defined to contain all of the points within P. As each point is inserted, the appropriate edges are added to the current Delaunay triangulation so that it remains a triangulation although it is not necessarily still a Delaunay triangulation. In order to ensure that no point within the current triangulation lies within the circumcircle of any triangle, illegal edges must be flipped. An illegal edge is one where the circumcircle defined by the three vertices of one adjacent triangle contains the outlying point from the other adjacent triangle. Special care must be taken when legalizing edges where one or more of the vertices involved belong to the initial bounding triangle. Only edges whose adjacent triangles have been changed due to the insertion of the new point can be illegal since the triangulation was legal beforehand. As a result when an edge is illegal and must be flipped, the other edges incident to the involved triangles must also be legalized. Since the angle measure of the triangulation increases with each edge flip and there is a maximum angle measure for the given set of points, the edge legalization is guaranteed to terminate. Once all of the points have been inserted into the triangulation, the vertices of the initial bounding triangle and all of the edges incident to these three vertices must be removed from the triangulation.

3.2 Transforming to the Voronoi Diagram

The Voronoi Diagram of a set of points P Vor(P) can be computed easily given its dual DT(P). Each triangle within DT(P) corresponds to a vertex in Vor(P). Each edge in DT(P) corresponds to an edge in Vor(P). For a triangle in DT(P), the location of the vertex in Vor(P) can be calculated by determining the center of the circle circumscribed by the three vertices of the triangle. This point can be found by determining the intersection of the perpindicular bisectors of each edge in the triangle. Each edge in DT(P) is adjacent to two triangles and corresponds to an edge in Vor(P) connecting the two vertices which are the duals of these adjacent triangles. Using this dual transformation it is a simple procedure to compute Vor(P) given DT(P).

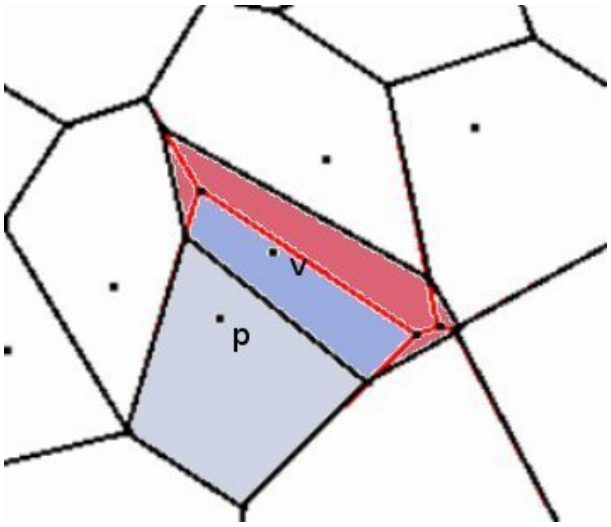

Figure 1: This figureashows the Voronoi cell for the inserted point v overlayed on the Voronoi diagram for the original point set. One of v's natural neighbors, p, is shown with its Voronoi cell shaded. The area stolen by the insertion of v can be seen in the overlap between the two Voronoi cells.

3.3 Computing the Natural Neighbors Interpolant

When computing the natural neighbors interpolant, it is important to intuit the relative 'neighborliness,' as (**?**) calls it, of adjacent Voronoi cells. These neighbors are the points within the original point set whose Voronoi cells intersect the Voronoi cell of the interpolated point, if it were to be added to the point set. The interpolated value is computed as a weighted average of the area stolen from each neighbor by the insertion of this point.

To compute this weighted average, we first compute the Delaunay triangulation of the original set of points P. Given DT(P) we need to calculate the Voronoi cell of the point v, whose value we want to estimate, if it were to be added to P. Since the insertion of v will only alter DT(P) and consequently Vor(P) locally, we do not need to recompute the entire Delaunay triangulation. We only need to compute a Delaunay triangulation for the points which would be neighbors of v. In order to determine this local point set we can use the existing triangulation DT(P)and corresponding Voronoi diagram. The area stolen from each neighbor of v can be computed by finding the difference between the area of Voronoi cell in Vor(P) and the area within the local Voronoi diagram. The interpolated value for v is then calculated as: $\sum_{\text{neighbors of} v} \frac{\text{areastolen}}{\text{area of the Voronoi cell of} v} \alpha$ where alpha is the value, possibly an elevation value, for the specific neighbor.

To compute DT(P) and transform to Vor(P) as described earlier takes O(n logn) time where n is the size of P. Vor(P) must be calculated once for the original point set. The local Voronoi diagram must be calculated for each interpolated point. Since the size of this local subset of P does not depend on n but rather the distribution of points in P and the number of neighbors that the inserted point would have, we can assume that for large point sets, this local set of points will be much smaller than n and will not depend on n. This means calculating the local Voronoi diagram will take a relatively constant amount of time independent of the size of P. In order to determine which points belong in this local point set we determine which triangle in DT(P) the point lies within. We can follow the outgoing edges from the vertices of this triangle to determine the appropriate neighbors to include. This process again depends on the the size of the local point set which does not depend on n. To calculate the weighted average we must locate Voronoi cells a constant number of times. We can use the DAG search structure for DT(P) to locate points and follow the dual pointers into Vor(P). Locating a Voronoi cell will take O(logn) time. For each point we need to interpolate we need to perform a constant number of logn searches plus a relatively constant amount of work to compute the local Voronoi diagram, so to interpolate k points for a point set of size n would take O((n+k)logn) time.

4 Results

A set of 50 points within a 400 by 400 region were chosen. Elevation data for these points was then obtained by sampling an elevation image which can be seen in Figure 2 (far left). Three different interpolation methods were performed on a small subregion of the image using the 50 sampled points. The first method used was our natural neighbor interpolation algorithm. The results of our interpolation were then compared to two of GRASS' built in interpolation routines, regularized spline with tension and inverse distance weighting. The differences between the resulting interpolated images and the original image

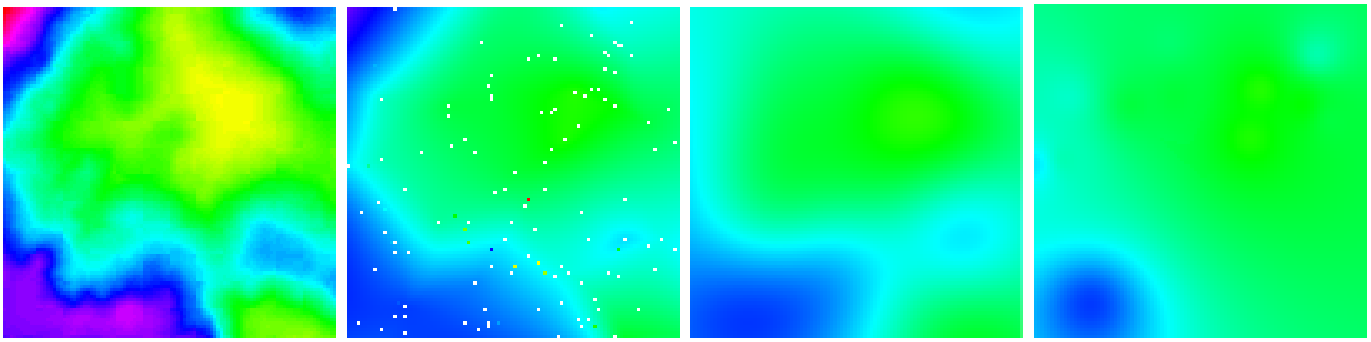

Figure 2: This figure shows the original image, the image generated using our natural neighbors interpolation routine, GRASS' regularized spline with tension and GRASS' inverse distance weighting from left to right, respectively.

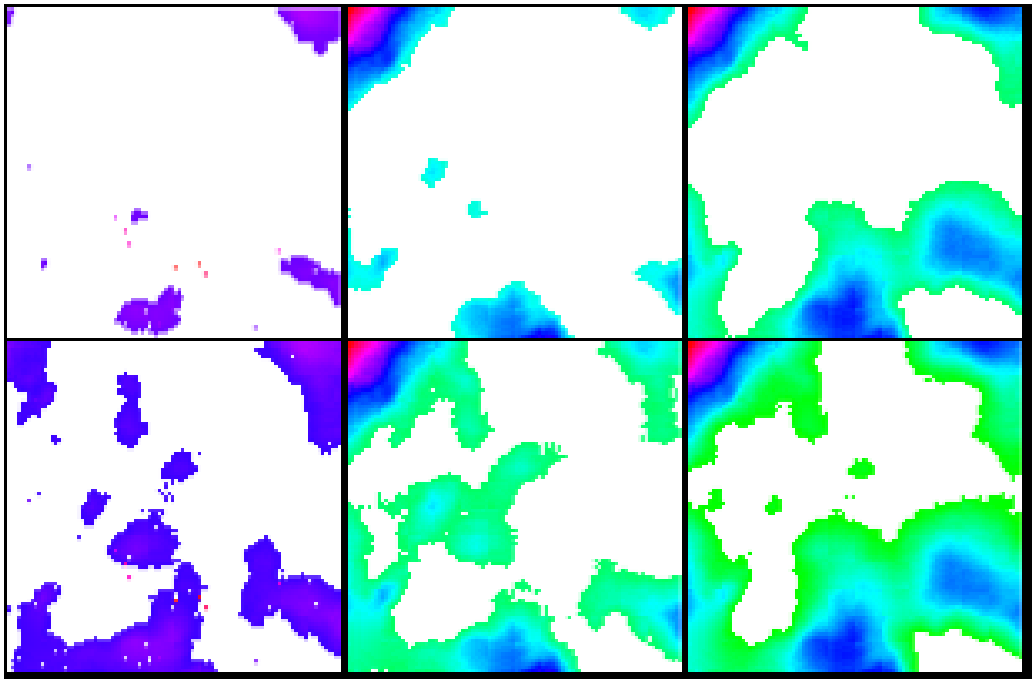

Figure 3: This figure shows the difference images generated by subtracting the interpolated images from the original image. The difference images were then thresholded to only display values above a certain value. Their arrangement is identical to the table below.

Threshold	NNI	RST	IDW
10	5.1%	13.5%	38.9%
2	28.4%	44.4%	57.1%

Table 1: This table displays the percent of the interpolated image that differs from the original image by a defined threshold.

were calculated within the appropriate region. The results were thresholded at varying levels and percentages for estimated values lying outside of the threshold were calculated in table 1.

For the given data set and sampled points, our natural neighbor interpolation routine outperformed the built in regularized spline with tension and inverse distance weighting interpolation routines. At a threshold of 10 units, only 5.1% of the pixels were incorrectly estimated by our interpolation routine compared with 13.5% and 38.9% incorrect respectively for the regularized spline with tension and inverse distance weighting routines. At a lower threshold of 2 units, our routine still dramatically outperformed the other two methods.

5 Discussion

All three methods of interpolation seemed to have trouble estimating values in relatively similiar areas. This could result from the sample points being more sparse in these areas or the frequency of the data in the original image being higher. Looking at the difference images generated for each of the interpolation routines reveals that these trouble zones for interpolation are common across each method.

Due to a lack of robustness in our interpolation routine, values at certain locations could not be estimated which can be seen with the speckling in the difference image. This leads to a greater percent incorrect which leads us to believe that our method would perform even better given a more robust implementation. The relative aesthetic advantage of natural neighbors interpolation over the regularized spline with tension method and the inverse distance weighting method is readily apparent. It seems that the natural neighbors method is much more capable of dealing with a sparse set of points to interpolate. This advantage could be because the tesselation underlying the interpolation routine extends beyond the boundary of the eventual image, allowing for better interpolation around the boundary of the image relative to the other methods.

We chose not to include runtime comparisons because our method took far longer to finish compared to the others. Each time we ran the natural neighbors routine, it required approximately an entire night to complete. When GRASS interpolated using its built-in routines, they both finished in about one second - negligible compared to our runtime. The extreme expedience of GRASS' routine is likely due to the sparse number of points that were interpolated over.

6 Future Work

There are several improvments to our interpolation method that would greatly enhance the usefullness of our implementation. The first necessary improvement centers around the degeneracies hinder the proper triangulation of the environment. Having a more robust Delaunay triangulation would remove the uninterpolated holes in the final image. This could have also been worked around easily by averaging over the gaps in interpolation once the Delaunay triangulation failed. Improving our point insertion method during the interpolation step would help cut down on some of the computational costs as well.

Creating a more extensive set of comparisons by varying the number of points interpolated over, varying the image frequency and complexity and including other interpolation methods would provide a more accurate gauge of our interpolation method's relative performance. A more comprehensive runtime comparison between methods would hopefully help determine the practical usage of our interpolation method too.