# Sparse Interpolation Using Delaunay Triangulation

Connor McCullough, University of Miami

***Abstract*—Delaunay Triangulation can serve as an ideal means of interpolating an image with only sparse and random data available. The process includes thresholding the greyscale image to find those pixels with available data, splitting the image into RGB or HSV channels to process separately, forming the Delaunay Triangulation based on available data, determining corresponding triangle for each missing pixel, and interpolating data for these mixing pixels based on proximity to the vertices of the corresponding Delaunay Triangle. Results were best for uniform loss of data, with some visual artifacts for images with random loss. Splitting the image into RGB proved more robust than HSV due to issues with interpolating hue channel.**

***Keywords—Delaunay, interpolation, triangulation, RGB, HSV***

## I. Introduction

Interpolation can be used as an image processing technique for restoring previously lost information in an image. There are various techniques to interpolate this lost data, but one method which is optimal for a sparse, random set of data points is using a Delaunay triangulation for interpolation. The triangulation subdivides an area into triangles based on the criteria that the circumcircle of any triangle (the circle which goes through all three points on the triangle), does not contain any other points of the triaangulation. In order to use the Delaunay triangulation to interpolate sparse data, the points with data must be found and the triangulation must be created based on these points. Next, the pixels without data in the image must be determined, and these are the points where interpolation will be done for. In order to interpolate data for these points, the Delaunay triangle that the point is within must be found. Lastly, the data is interpolated for the point by summing different ratios of the three points in the image based on the proximity of the point to each of the vertices.

## II. Thresholding to Identify usable Data

In order to determine which points are being used to create the triangulation, and which points do not have data and must be interpolated, a threshold must be set based on overall color intensity. Depending on the file type, pixels without data may be stored with a 0 value intensity, or a nonzero value. Compression artifacts associated with .jpg compression can lead blank pixels to have nonzero pixel values, and therefore a nonzero threshold must be set. The processing program therefore must set different thresholds depending on the input data type to obtain the optimal results. One problem associated with using pixels with low-level intensity as the criteria for determining missing data is that areas in the image with actual low levels of intensity may be mistaken as pixels without any data and be interpolated, replacing the data with the data from surrounding pixels. While this only applies to low light levels for intensity data, if thresholding is performed in the RGB or HSV domains, pixels without a specific color, lack of saturation, or hue value close to zero may also trigger a pixel to be set as a blank pixel. Therefore, thresholding is best done on the greyscale image, as this will trigger the least amount of false positives. This should have little impact on the overall image as dark areas of the image will typically be surrounded by other dark pixels.

## III. Delaunay Triangulation

Delaunay Triangulation is an ideal method for interpolation of sparse data where random pixels of data will be missing, because it can be applied to a nonuniform pattern of available and missing data. The triangulation subdivides a grid into a series of triangles, using the criteria that the circumcircle, the circle which travels through each vertex, of each triangle must not contain any other points that are not part of the triangle. This prevents the Delaunay triangulation from creating excessively long, irregularly shaped triangles. There are a variety of algorithms for finding the Delaunay Triangulation. One method is using evaluating the determinate of the three vertex points, to determine if another point is within the circumcircle. Other methods include flipping the edges of a triangle to determine if it is non-Delaunay, and to add one vertex at a time and re-triangulate the effected portions of the graph. For this application, each point of a Delaunay Triangle represents a point with available image data, and the points in between must be interpolated.

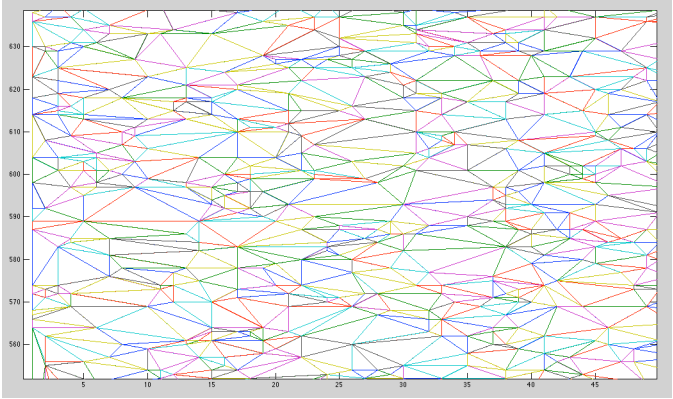



Fig. 1. Delaunay Triangulation for randomly available data.

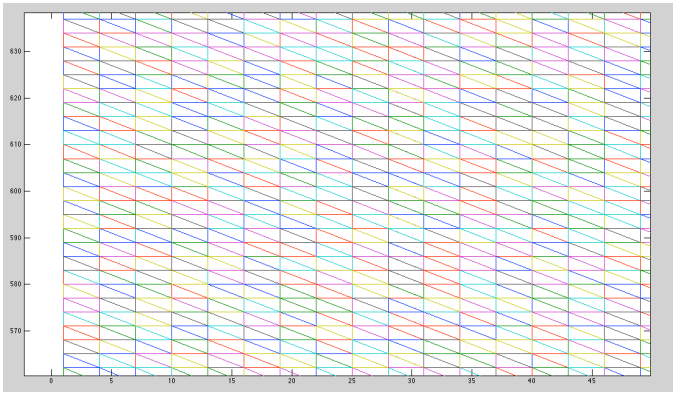


Fig. 2. Set of data for uniformly available data.

## IV. Determing Delaunay Triangle For Each BLank Pixel

While determining whether a point is inside a triangle is easy by visual inspection, creating a robust and efficient algorithm for determining the containing Delaunay Triangle for each blank pixel proves much more difficult. Checking every single triangle for every single point is not possible, due to the slow processing speed of FOR loops in MATLAB. Therefore, an approach that vectorizes these FOR loops must be used for optimal processing speed. First, the maximum and minimum X and Y points for each Delaunay triangle are found. For each blank pixel, a small number of possible Delaunay triangles are found, whose maximum and mimimum vertices enclose the blank pixel. This reduces the size of the second FOR loop significantly to the point where MATLAB can complete the operation in a reasonable amount of time. If a point is within a triangle, the angles between the lines attaching the point and the vertices of the triangle will add up to 180 degrees. This is checked on the remaining possible triangles to determine which triangle contains the point.

## V. Interpolation

Once the correct Delaunay Triangle has been determined for the given point, the interpolation equation must be applied. The following equations are used to find the value at the blank pixel based on the values at the other three pixels of the triangle.

$$f_q = c_1 f_1 + c_2 f_2 + c_3 f_3 \qquad (1)$$

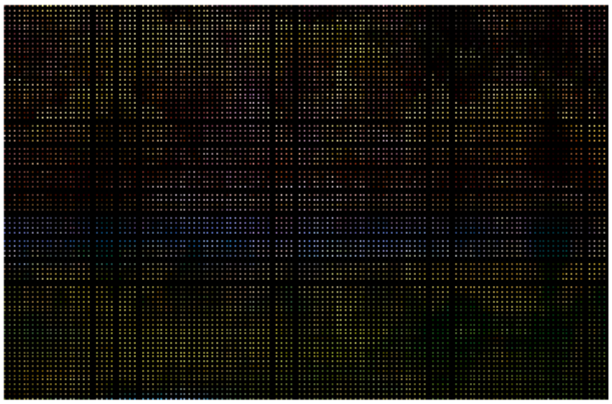
$$c_1 = \frac{x_3 y_2 - x_2 y_3 + x_q(y_3 - y_2) - y_q(x_3 - x_2)}{(x_2 - x_3)y_1 + (x_3 - x_1)y2 + (x_1 - x_2)y_3} \qquad (2)$$

$$c_2 = \frac{x_1 y_3 - x_3 y_1 + x_q(y_1 - y_3) - y_q(x_1 - x_3)}{(x_2 - x_3)y_1 + (x_3 - x_1)y2 + (x_1 - x_2)y_3} \qquad (3)$$

$$c_3 = \frac{x_2 y_1 - x_1 y_2 + x_q(y_2 - y_1) - y_q(x_2 - x_1)}{(x_2 - x_3)y_1 + (x_3 - x_1)y2 + (x_1 - x_2)y_3} \qquad (4)$$

A ratio c of the distance from each vertex, normalized by zero is multiplied by the value at each of the three points on the triangle. These three values of c will add up to 1. This entire equation is vectorized so that every variable in (1-4) is a vector and each multiplication and division is a dot operation instead of a matrix operation. Vectorizing the entire equation instead of running it point by point as a FOR loop greatly increases the processing speed in MATLAB and reduces wait times to a manageable time.

## VI. Results



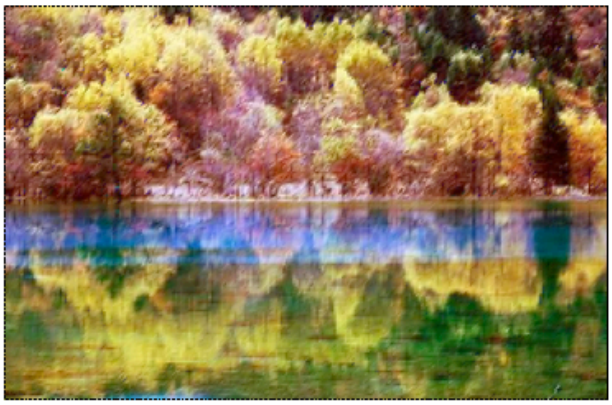
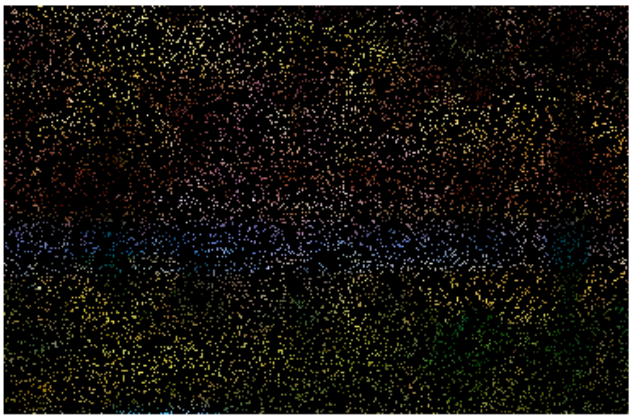
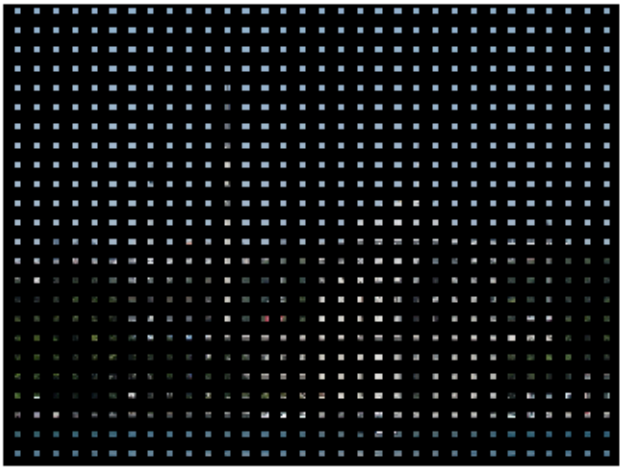


Fig. 3. Test Image 1 with every 3rd pixel present.





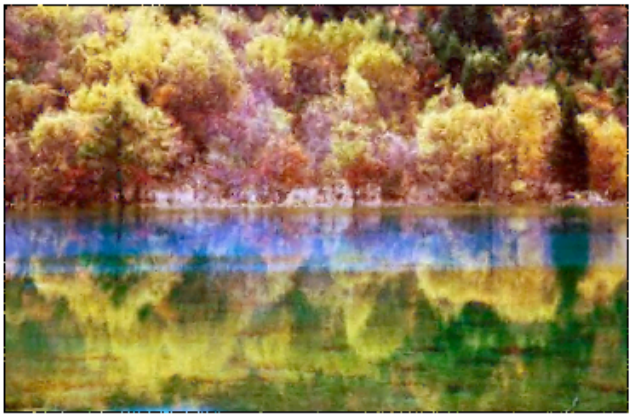


Fig. 4. Test Image 1 with randomly missing pixel data.

Fig. 5. Test Image 2 with every other pixel present.

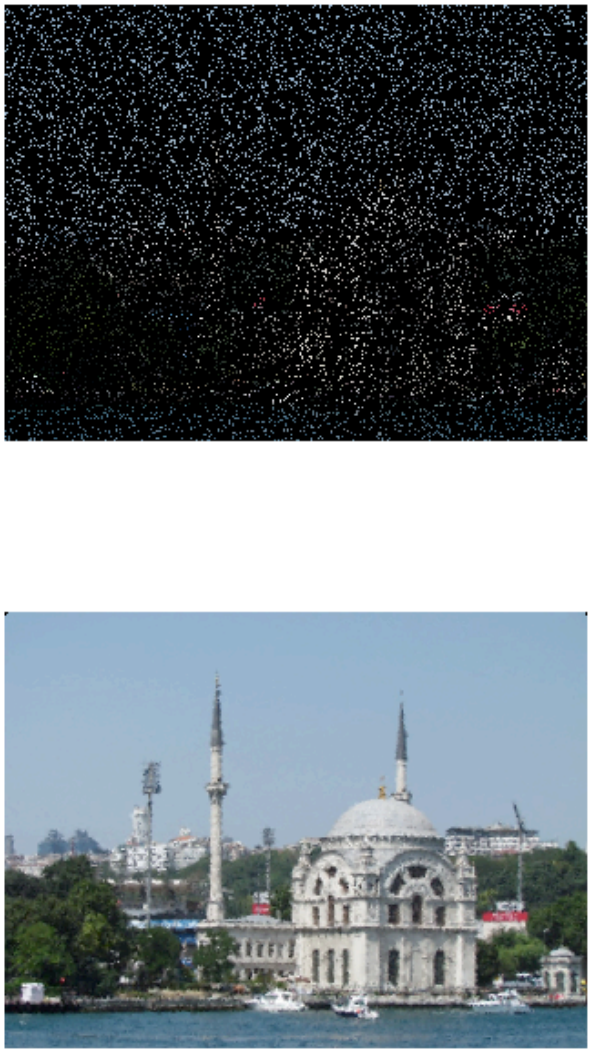


Fig. 6. Test Image 2 with randomly missing data.

The general trend for each of the images was that those images with a uniform loss of data had less distortion than those with random blank pixels. There was little visual difference in the images which had data for every other pixel and every third pixel. The result was what appeared to be a slightly blurry, low resolution image without any patches with exceptional distortion or loss of information. When data loss was random however, the interpolation was much more patchy, and appeared like a water colored painting or a mosaic. The Delaunay Interpolation is a form of linear interpolation, which works fine when there is data uniformly surrounding each blank pixel. However, when forced to interpolate larger, irregularly shaped patches of blank pixels, this interpolation falls short, because linearity is not present in nature and so the interpolated image does not look as natural in these sections. For images which are known to contain random loss of data, a parametric model such as the Facet model may be more ideal to accurately and naturally restore data.

## Comparison of RGB and HSV Processing

For Part V, each of the resulting images was a result of thresholding the image in greyscale, interpolating each of the three R,G,B channels independently of each other, and concatenating the results at the end. An alternate method would be to convert the image from RGB to HSV (Hue, Saturation, Intensity) and processing each of these channels. The result was that in almost every circumstance, the RGB and HSV processed images were indistinguishable when placed next to each other in nearly every circumstance. The one exception was with the first test image, which consists of very rapid shifts in hue. In each of the three images, there are lines of blue when the hue is transitioning from red to magenta, as seen in Fig. 7. This makes sense because while these colors may be similar in R,G, and B values, the hue values are on opposite sides of the spectrum. The interpolated value of these two colors with very high and very low hue values is a value right in the middle which is completely unrelated to either of the colors. This shows a shortcoming in HSV processing. While the hue dimension is actually an angle, it must be represented in MATLAB as linear. Therefore, the data at very high and very low values is identical and can lead to undesirable results if hue is processed like any other channel. This could be avoided by customizing the processing for the hue channel, but this would lead to extra code and processing.

One other minor difference can be seen in Fig. 8, where the hue of the sky is slightly different for the RGB and HSV channels. This can be attributed to the difference in hue processing that takes place from having color levels on three separate channels, and having the hue channel completely isolated.

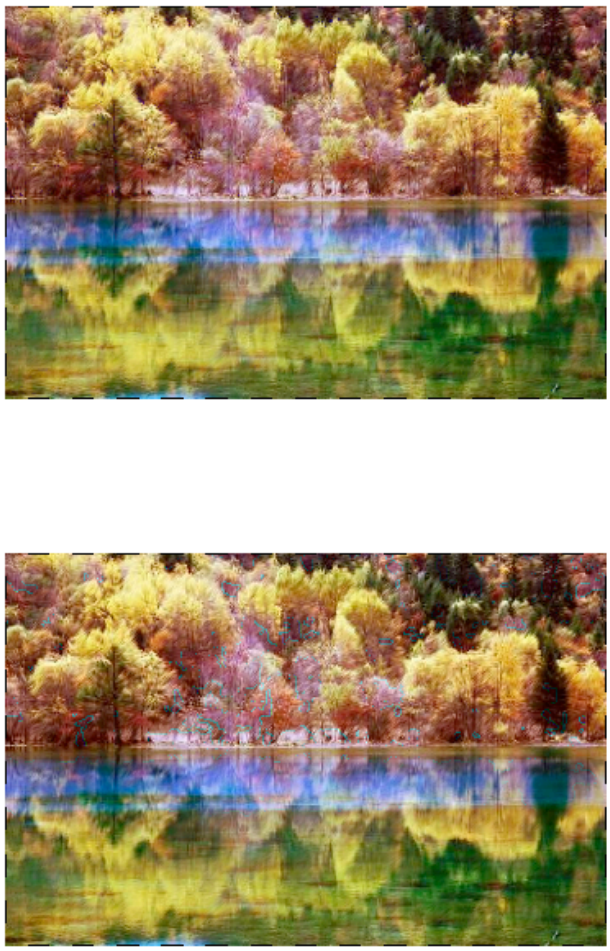


Fig. 7. Comparison of RGB (top) and HSV (bottom) processing for test image 1.

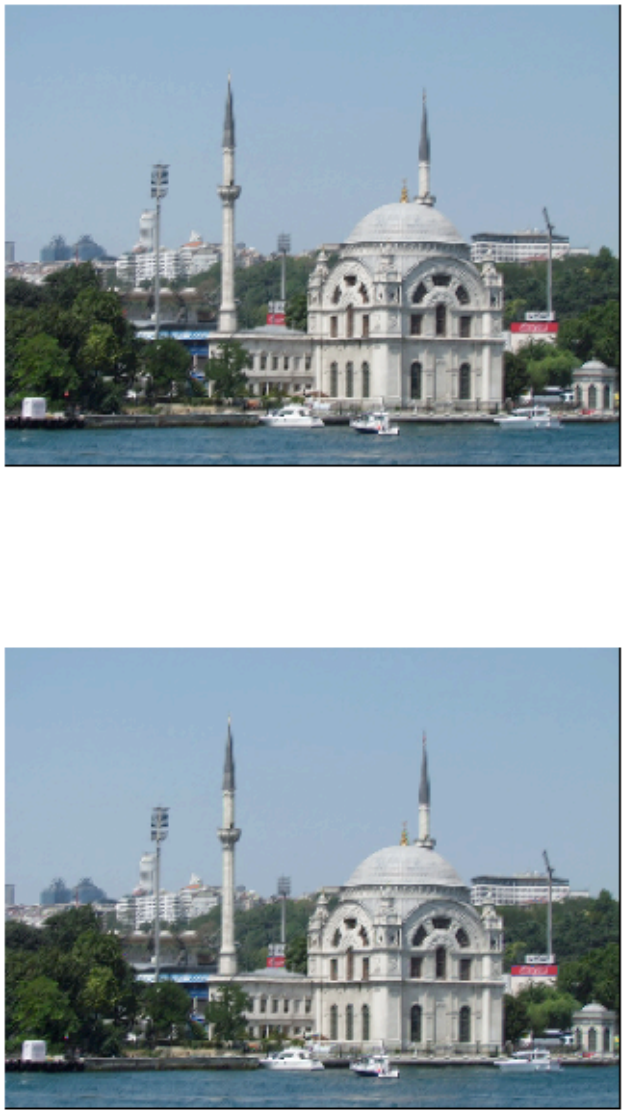




Fig. 8 Comparison of RGB (top) and HSV (bottom) processing for test image 2.

## CONCLUSION

The use of Delaunay Triangulation for interpolating sparse data proved to be sucessful in interpreting data without significant levels of distortion or loss of clarity. While images with a uniform pattern of pixels with and without data were slightly clearer, even images with random loss of data were still usable. There were some differences between splitting channels into RGB and HSV, mostly in hue. With rapid changes in hue that have similar hues but contrasting hue values, interpolated values will be innacurate and lines of innacurate color will appear in the result image. There are also minor hue differences in areas without color change due to the differences in processing.