Fall 2017

Voronoi Stippling Report

Stephen Arnold - G00864316 *CS633 - Computational Geometry*

1 Summary of the Two Methods

In this introductory section, the methods and algorithms employed by the two stippling techniques shall be reviewed.

1.1 Voronoi Method

The provided implementation of Voronoi (VR) stippling makes use of Fortune's Algorithm for the calculation of the Voronoi Cells. Initially, the input parameters are read in and stored. Next, an initial random distribution of points is created within the boundaries of the original image. These data points are the initial configuration stored in the stippler data structure.

Using an iterative approach, a Voronoi Diagram is created from the set of inital points using Fortune's Algorithm. Next, the applied stipples are redistributed and the displacement from the new locations to the previous centroids is calculated. This process is repeated until the average change in displacements across all cell meets a minimum threshold. Once the centroids have settled, the image is stippled by placing circles sized in proportion to the number of iterations required to refine the Voronoi Cell centroid.

1.2 Hedcuter Method

Using an iterative approach, the hedcuter (HC) stippler uses simple rejection sampling to generate an initial distribution of points, or centroids, within the confines of a given image. The algorithm compares the grayscale value of the randomly selected location within the image and applies a gaussian scaling factor. The point is either accepted or rejected based on satisfying a predetermined threshold - namely the grayscale value of the pixel. Once the inital points are determined, a set of right cones are generated with apexes at the cell generator. Initially the cones have equal height. Each cone is also assigned a unique "color" to identify it later. The intersection of the cones in the z-direction will determine to which cone the pixels of the original image belong. In this way, the pixels are assigned a unique value which determines to which Voronoi Cell the pixels belong.

The weighted Centroidal Voronoi Tessellation (CVT) used by the hedcuter algorithm makes use of a density function to distribute, and realign, the centroids of the Voronoi Diagram. Darker areas (lines, edges, etc.) are assigned a higher density than lighter areas. As the algorithm iterates through the image, centroids tend to migrate towards higher density regions.

The stippling method invoked by the hedcuter places a variably sized disk, centered at the centroid of the cell. Two options exist within the code: uniformly sized disks and variably-sized disks. Various levels of gray may be obtained by uniform-sized disks at varoius spacing densities or by

varying the sizes of the placed disks. Colored disks may also be used in conjunction to create various tones. [1]

2 Comparison of the two methods

After compilation of both stippling methods, various test cases were run to compare each methods effectiveness as a stippling method. Those results shall be discussed here.

A single image was used to allow for simple comparrison of the significant variables of interest. The image, seen in Figure 1, is a grayscale PNG-format image of Abraham Lincoln, 16th president of the United States, from the year 1865. The image size is 1332px x 1644px, and is composed of 256 levels of grayscale. Unless otherwise noted, this image was used for all variations.

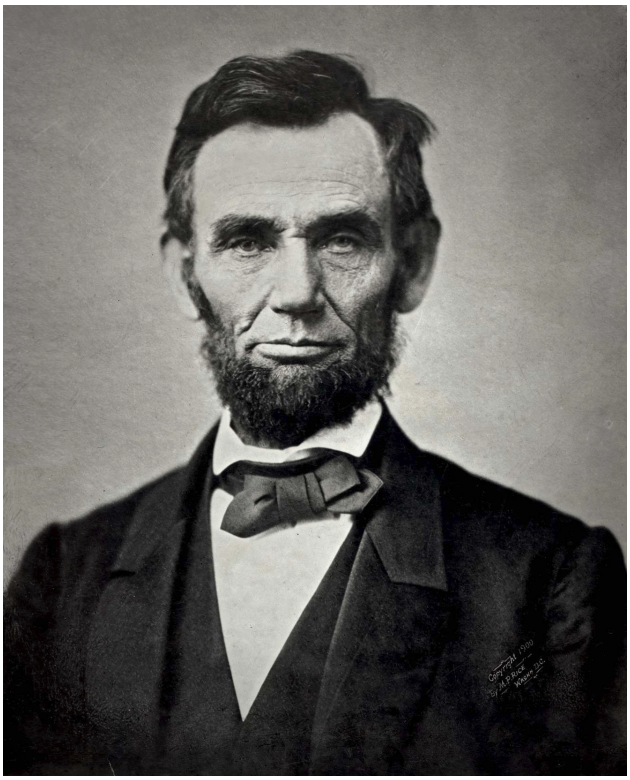


Figure 1: Abraham Lincoln, 1865.

2.1 Multipass Results

To determine if there are any statistically significant variations in running the same program on the same image multiple times, a script was written to test this condition. For VM, the results of each run using 4000 points, subpixel density 5, non-overlapping, and variable radius were extremely consistent. Minor variations in the duration of the runs were seen. Over 5 runs, times ranged from 56.78 to 57.28 seconds. Of note, when visually comparing the VM processed images, no descernable differences in final pixel placement are found. This is due to the approached used to

shift the centroid of the Voronoi Cell - initially determined from random selection of points within the image. The shifting is based on actual color values of nearby pixels. Eventually the centroids will converge to similar points.

For a similar run using HC, using 4000 points, max disk size of 5, and non-uniform disk sizing, program run times ranged from 263.41 to 453.37 seconds. This variation is times is most likely due to the tight constraints of the Point select/reject function.Also, using HC results in differing placement of disks in the final image. The initial selection of points uses a probabilistic approach in determining the voronoi cell centroids. These centroids then produce selection cones in a psudo z-axis to properly assign pixels to a particular voronoi cell. The selection cones, having their apex at randomly selected locations can produce differing end result. These pixel differences can be seen in Figure 2. These minor variations would become less pronounced at higher pixel densities.

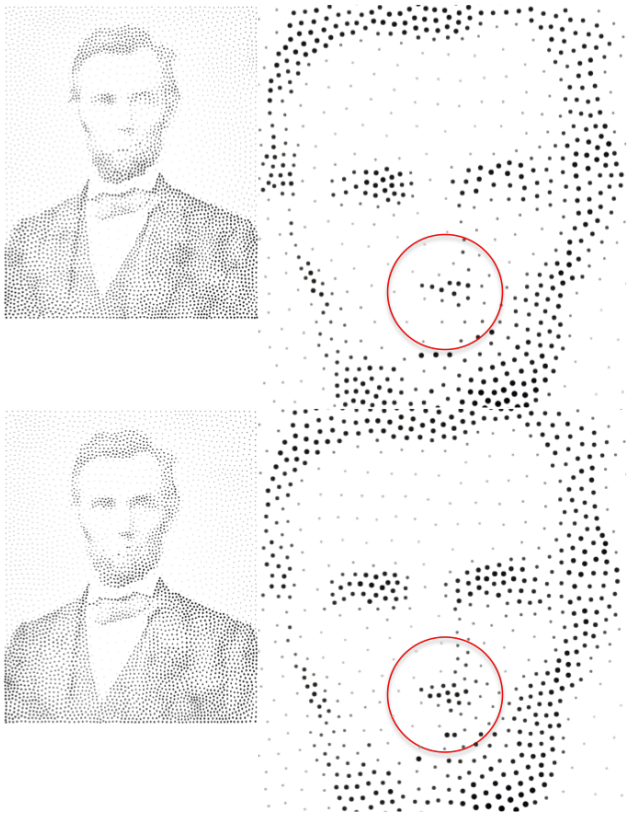
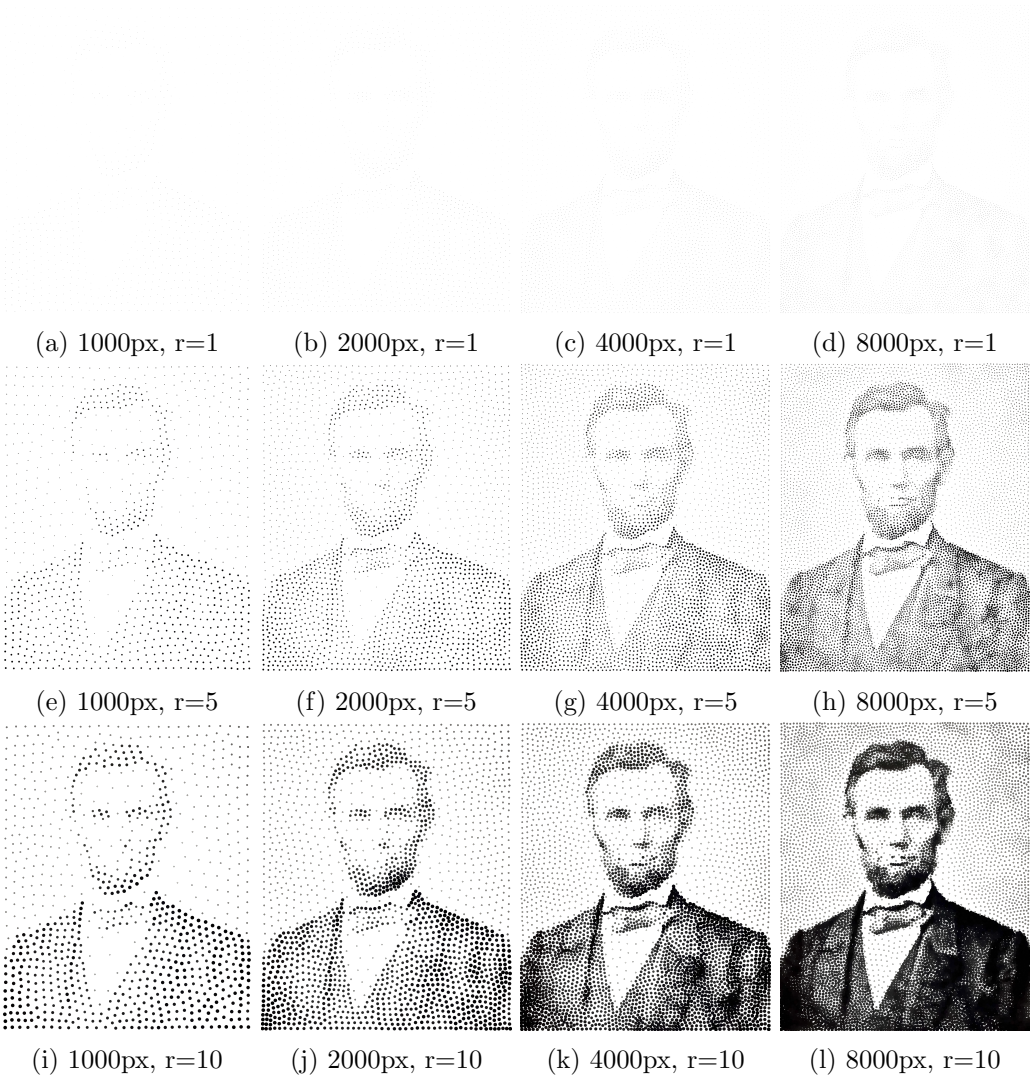


Figure 2: Pixel placement differences between hedcuter runs.

2.2 Variations on Number of Disks

Varying the number of disks has some effect on the final output images produced by the two algorithms. For voronoi, the original disk is present in its former location, but is a reduced size and has additional (also smaller) neighbors to account for the greater quantity. Also, during the initial step of the hedcuter algorithm, randomly sampled points are either selected or rejected based on a Gaussian Probability distribution against the "darkness" of a given pixel. Since both the selected point and whether or not the point will be accepted/rejected can lead to different distributions ... and can consequently explain the differences in completion times of the hedcutter algorithm.



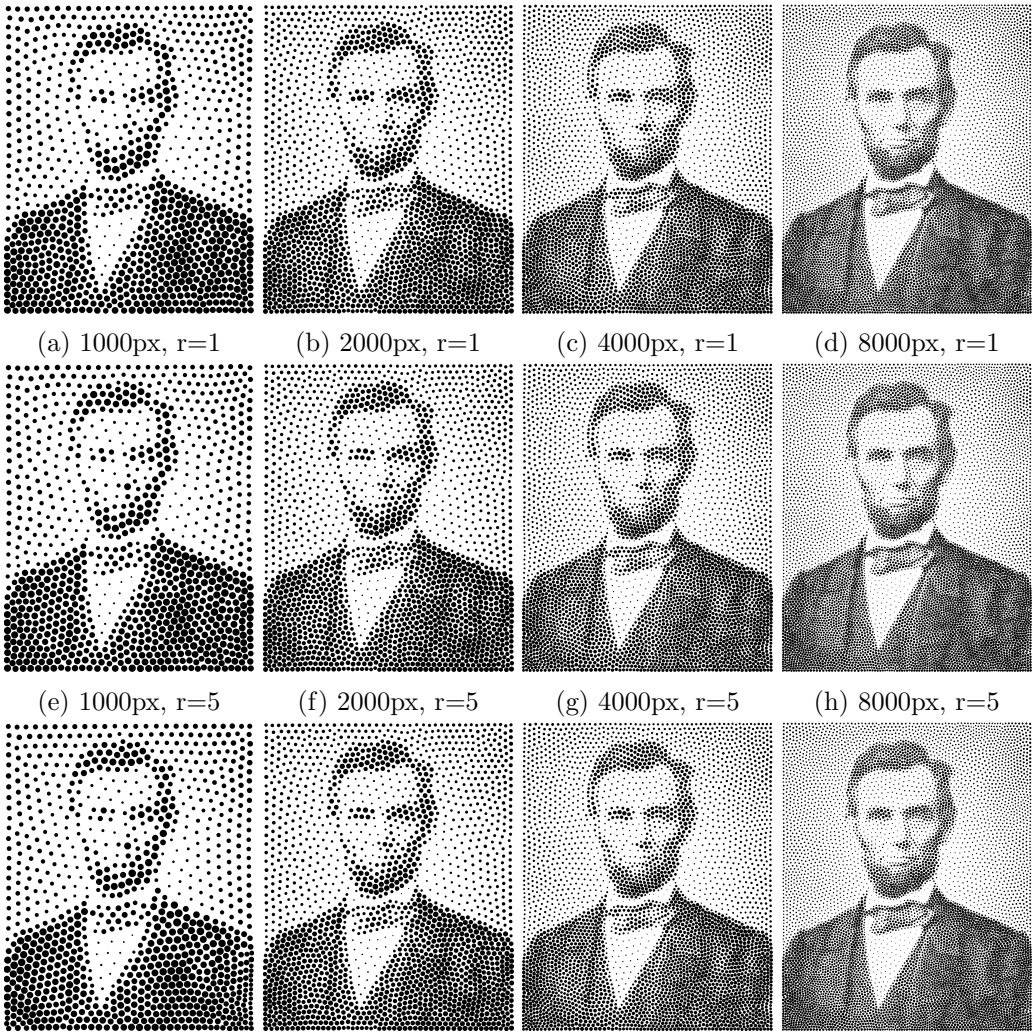
(a) 1000px, r=1 (b) 2000px, r=1 (c) 4000px, r=1 (d) 8000px, r=1

(e) 1000px, r=5 (f) 2000px, r=5 (g) 4000px, r=5 (h) 8000px, r=5

(i) 1000px, r=10 (j) 2000px, r=10 (k) 4000px, r=10 (l) 8000px, r=10

Figure 3: Hedcuter Method images varying number of disks and radii.

The first row of images output from HCM in Figure 3 at first appears to contain no images, when in reality we have simply selected a disk size that is so small as to not be visible at the scale presented. However, in subsequent images, the stippled images may be seen progressing from least-densely to most-densely filled.



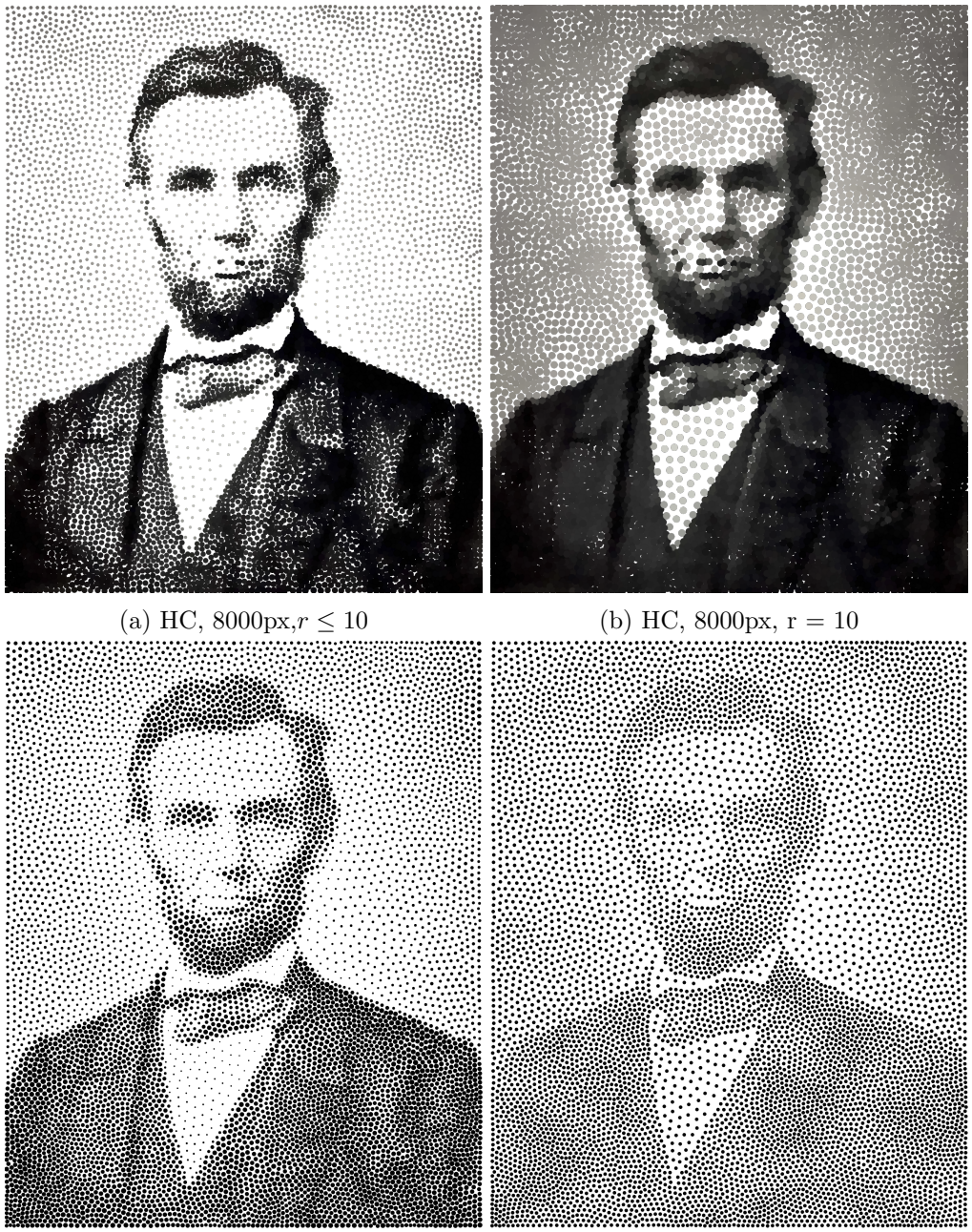
(a) 1000px, r=1 (b) 2000px, r=1 (c) 4000px, r=1 (d) 8000px, r=1

(e) 1000px, r=5 (f) 2000px, r=5 (g) 4000px, r=5 (h) 8000px, r=5

(i) 1000px, r=10 (j) 2000px, r=10 (k) 4000px, r=10 (l) 8000px, r=10

Figure 4: Voronoi Method images varying number of disks and radii.

The images produced by VR, and seen in Figure 4, demonstrate the constrain of disk size based on localized grayscale value. Regardless of the limitation of the tile size of centroid computations, the variations are iteratively removed as the algorithm progresses. The variably sized stipple points allow finer details such as edges or textures to be preserved.



(a) HC, 8000px,$r \leq 10$ (b) HC, 8000px, r = 10

(c) VM, 8000px, $r \leq 10$ (d) VM, 8000px, r = 10

Figure 5: Variations between fixed and uniform disk radius between methods. 8000 disks.

The images in Figure 5 are variations of an 8000 point stipling of the sampled image. The top row utilizes HC, while the bottom row uses VR. The left two images have variable stipple sizes, while the images on the right have a fixed stipple radius of 10 pixels. Image a) maintains many of thestrong edge detais of the original at the expense of some of the finer details, such as the lips. Figure c) mainains much of the finere details, but the contrasts between lighter and darker areas lose much of their distinction. Of the four images, d) is the poorest quality as the fixed stipple size causes many of the subptle shades of gray to be completely lost.

Table 1: Sampling of HC and VR run times versus disk quantities.

Disks	hedcuter (s)	voronoi (s)
1000	384.581	121.74
2000	417.722	70.80
4000	411.796	50.76
8000	732.592	38.16
16000	1325.6	27.91
32000	1381.59	20.31
64000	2052.16	16.59
128000	3961.65	15.68

2.3 Variations on Speed with Number of Disks

As the two methods used in this assignment differ signicantly, the time to calculate a solution does vary within each implementation. This variation is dependant upon the number of disls selected to stipple the given image. As seen in the following table, run times for HC increase with increased number of disks. The VM, however, has the opposite trend ... becoming faster as disk count increases. This is due to the larger number of individual pixels that must be handled and calculated even at smaller quantities of disks.

2.4 Differences in image type

2.5 Differences from WSJ Hedcuts

The images produced by HC and VM differ from the traditional Wall Street Journal hedcuts is a couple of areas. First, and simplest, are background areas. The subject mater for a WSJ hedcut is usually a person, and just the head and upper torso are included. For a grayscale image, the background would be included to some extent - even when unwanted. This can be avoided somewhat by avoiding darker backgrounds.

Another differene is the us of random samples. When an artist creates a hedcut, there is purpose behind the stipples produced. From a programmatic standpoint, there will always be "artifacts" - unwanted pixels in the image as a result of the processing. These, too, may be eliminated though careful image selection and image quality. Random Samples

3 Improvement of hedcuter method

3.1 Distribution of Disks

Stictly random selection of initial points. How does this affect the edges?

3.2 Improve efficiency

GPU use - i never could get this to work on my Macbook Pro.

3.3 Colorful Disks

Addition of "colorful" disks.

(a) Original Image: 800x596, RGB

(b) HC, 4000px, native colored stipples

(c) VM, 4000px, colorful disks

Figure 6: Variations of applying random colorization to output disks.

As seen in Figure 6, the underlying pixel color is used when the image selected for stippling is itself

a color image. Adding a few randomizers behind the selction of the color of each indivifual pixel yield an extremely colorful stippling.

The following shows the Original and the Modified code snippettes which produced the colorful stipplings in Figure 6.

Original:

```
r = floor(r / cell.coverage.size());
g = floor(g / cell.coverage.size());
b = floor(b / cell.coverage.size());
```

Moddified:

```
r = (int)floor(255*rng_uniform.uniform(0.f, 1.f));
g = (int)floor(255*rng_uniform.uniform(0.f, 1.f));
b = (int)floor(255*rng_uniform.uniform(0.f, 1.f));
```

References

[1] Adrian Secord. Weighted voronoi stippling. In *Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering*, NPAR '02, pages 37–43, New York, NY, USA, 2002. ACM. 1.2