Fall 2017

Assignment3 Report

Genqian Hu *Computational Geometry*

1 Summary of the two methods

1.1 hedcuter method

1.1.1 Voronoi diagram and Centroidal Voronoi tessellation

First, the algorithm resizes the input image to a larger virtual image. Then, it scans the cells and stores all sites in a heap. After, it pops a cell to compute dx and dy so to get the centroidal voronoi diagram. When all cells are visited and the heap is empty, it collects cells from (0,0) rightwards and then remove empty cells. Finally, it moves each cell to its center of its coverage.

1.1.2 stippling

It reads the image and the centroidal voronoi tessellation computed before to stipple. It retrieves each cell from CVT and its grayscale information. Then, the location of each disk is defined by the x and y coordinates of each sites of the cells retrieved. The color of it is determinded by grayscale value. If it is required a colorful output, then the rgb value is from the original image's rgb value at that pixel.

1.2 voronoi method

First, it computes the range of the coordinates of the sites. Then do the plane sweep, if the sites' y coordinate is bigger than the lowest intersection's, it determines which half edge the new site intersects with and will triangulate it. If the sites' y is smaller, it creates a new bisector between the left and right half edges. Finally, it scans the edges and eliminates the invalid edges.

1.2.1 Centroidal Voronoi tessellation

It adds clip lines to the voronoi digrams one by one. If the density is higher there, the x and y there will also get larger weight when computing the new centroid. If no points are outside the clip planes,the centroid is computed by the sum of coordinates and the density there.

1.2.2 stippling

It read the coordinates of all the centroids created before. Then it outputs them to the output file. And it assigns each of them the right color as needed.

2 Comparison of the two methods

Figure 1: Different outputs from hedcuter

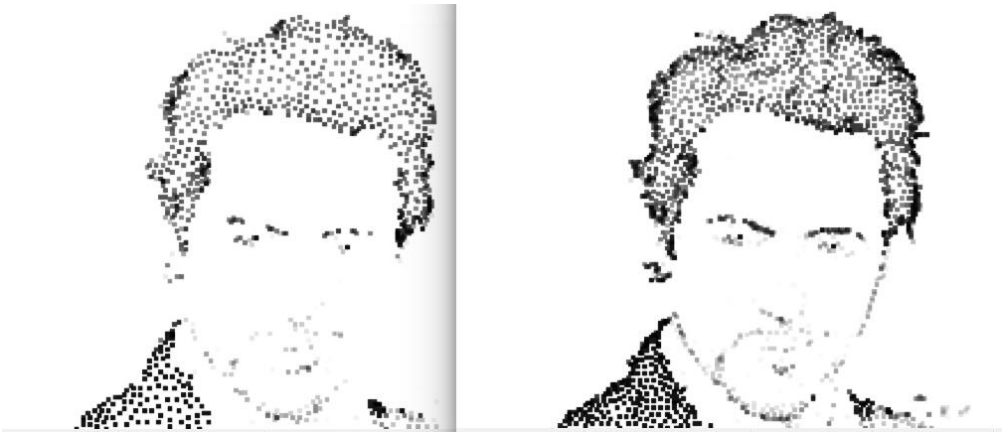

Figure 2: Outputs from hedcuter

1. The output images from hedcuter are different as shown in Figure 1. But the output images from voronoi are the same. Because hedcuter uses rng_uniorm and rng_gaussian methods to randomly initial points, the details is normal to be different during each run.

2. The overall distribution will be the same for both methods. As Figure 2 and Figure 3 shows, the distribution remains the same. The difference here is that the one with more disks seems to be darker, because its density of disks is higher. And following

Figure 3: Outputs from voronoi

```
tex@tex-virtual-machine:~/hedcut/hedcuter/code/build$ ./hedcuter timg.jpg
Total time: 15.3056
tex@tex-virtual-machine:~/hedcut/hedcuter/code/build$ ./hedcuter -n 2000 timg.jp
g
Total time: 23.1563
tex@tex-virtual-machine:~/hedcut/hedcuter/code/build$ ./hedcuter -n 3000 timg.jp
g
Total time: 31.097
```

Figure 4: Outputs from hedcuter

```
tex@tex-virtual-machine:~/hedcut/voronoi$ ./voronoi_stippler -s 2000 timg.png 11
.svg
Generating 2000 stipples.
Options: Black stipples, overlapping stipples, Variable radius, Subpixel density
 of 5
14.94% Complete
34.47% Complete
53.34% Complete
70.91% Complete
87.59% Complete
100.00% Complete
Completed in 0.57 seconds.
tex@tex-virtual-machine:~/hedcut/voronoi$ ./voronoi_stippler -s 3000 timg.png 11
.svg
Generating 3000 stipples.
Options: Black stipples, overlapping stipples, Variable radius, Subpixel density
 of 5
18.86% Complete
43.01% Complete
65.39% Complete
85.67% Complete
100.00% Complete
Completed in 0.50 seconds.
```

Figure 5: Outputs from voronoi

these 2 methods, darker places deserve more disks. So the distribution should always be the same.

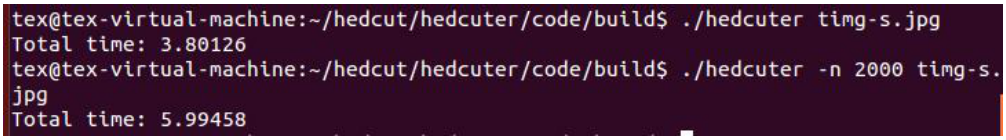

Figure 6: Outputs from hedcuter

Figure 7: Outputs from hedcuter

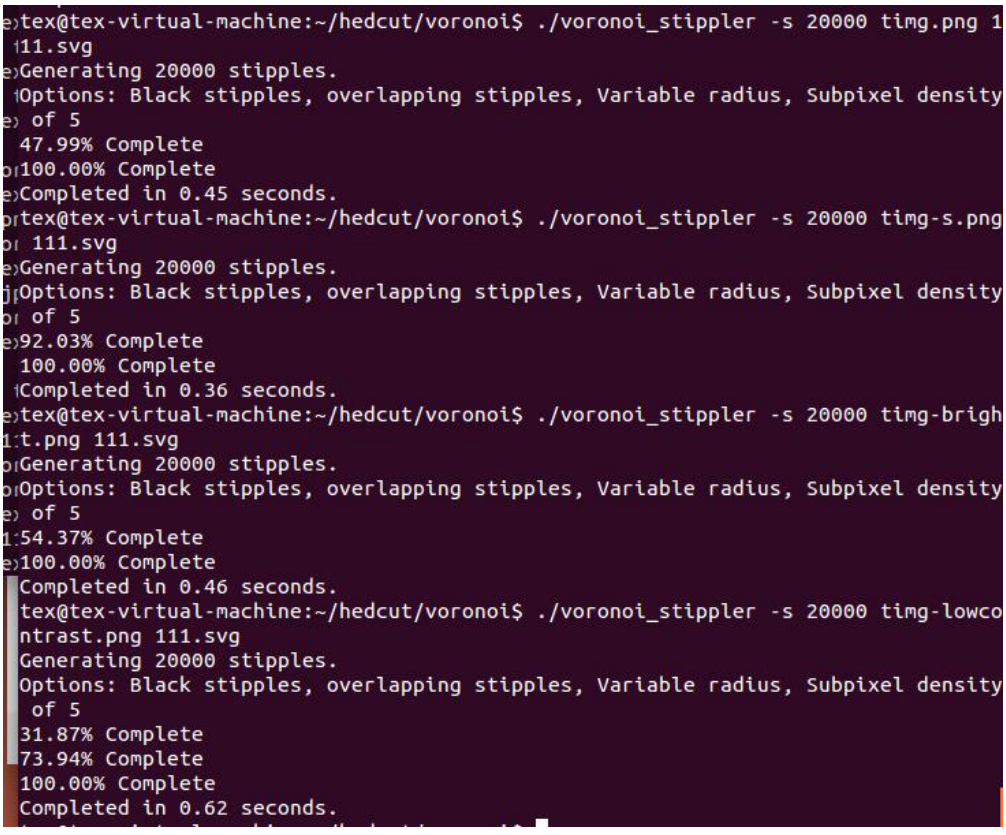

Figure 8: Outputs from hedcuter

3. As Figure 4 and 5 show, the voronoi method is always much faster than the other one when parameters are the same. And I think the reason hedcuter method is slower that much is because it generates the same virtual image repetitively. So,

I take it as one of my improvement and the result is not bad. And other reasons are voronoi precomputes the sets of distribution and merge them at runtime. But hedcuter method is doing plane sweep from points with small x coordinates which is time consuming.

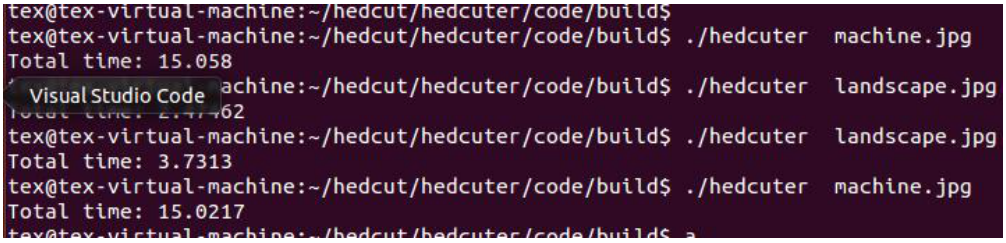

Figure 9: Outputs from hedcuter

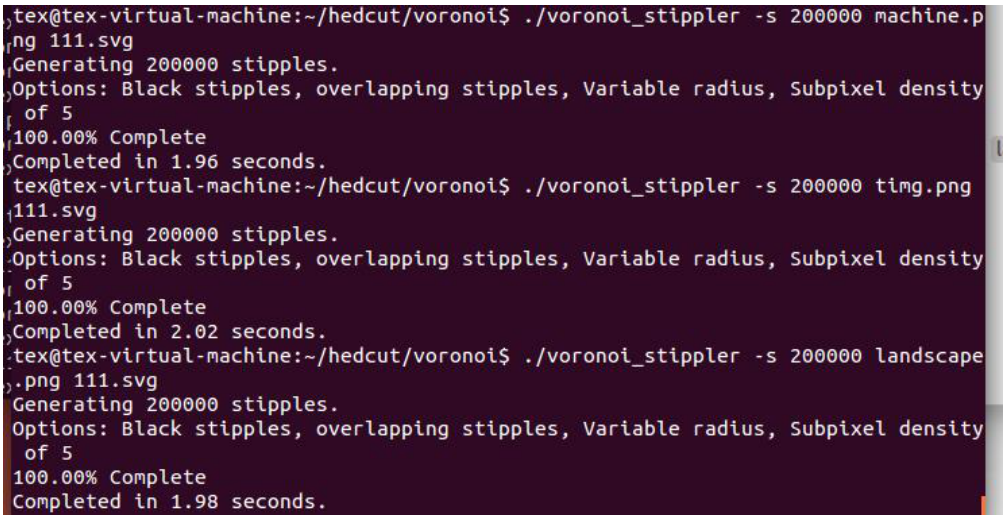

Figure 10: Outputs from hedcuter

4. As figure 6, 7 and 8 show, for both methods, the same picture with smaller size consumes less time, because less pixels mean less iteration. For hedcuter method, it runs much faster with the low contrast picture. Because lowering contrast makes the picture darker. And when the picture is getting darker and darker, its distribution of disks becomes more uniform. Thus, the propagation task is achieved faster.

5. From Figure 9 and 10, different types of images have the approximate total time for voronoi method. But for hedcuter method, it performs better on landscape picture. I think it's because the landscape picture I chose is relatively darker than other types of pictures. So the reason is like for the pictures with low contrast value.

6. I think hedcut images from the Wall Street Journal are better than images generated by these 2 methods. As shown in Figure 11, it has smooth boundaries and more artistic distribution of disks. Figure 12 is generated by voronoi method from

Figure 11: hedcut image from Wall Street Journal

a 200x200 picture with 2000 disks. Although viewers are able to tell what's in the hedcut picture, its distribution of disks is not as natural as the one from Wall Street Journal.

3 Improvement of hedcuter method

The first improvement is that I reorganized same codes so that the workload for vor and move_sites methods are reduced. According to the original code, it seems like each loop these two methods will generate a virtual high resolution image with the same parameters which means there is no need to do it repetitively. And it's also time consuming. Then, I changed that part of code so that the virtual image generating will be done only once. Figure 13 shows the total time the original program used to output the result. Figure 14 shows the total time the improved program used. The total time has reduced.

Figure 12: Outputs from hedcuter

```
tex@tex-virtual-machine:~/hedcut/hedcuter/code/build$ ./hedcuter -n 3000 -black
true pi.png
Total time: 28.2317
```

Figure 13: output from original method

```
tex@tex-virtual-machine:~/hedcut/hedcuter2/code/build$ ./hedcuter -n 3000 -black
 true pi.png
Total time: 0.136199
```

Figure 14: outputs from new one

The second improvement is that the program now can treat png files which contain transparent alpha channel correctly. The original version treats the transparent background as black which makes the output file weird. I modified main.cpp to make the input file lay on a white image. Fiqure 15 shows the comparison of the original program against the improved program with the same parameters. But when outputing colorful disks, the color of each disk is not always right.

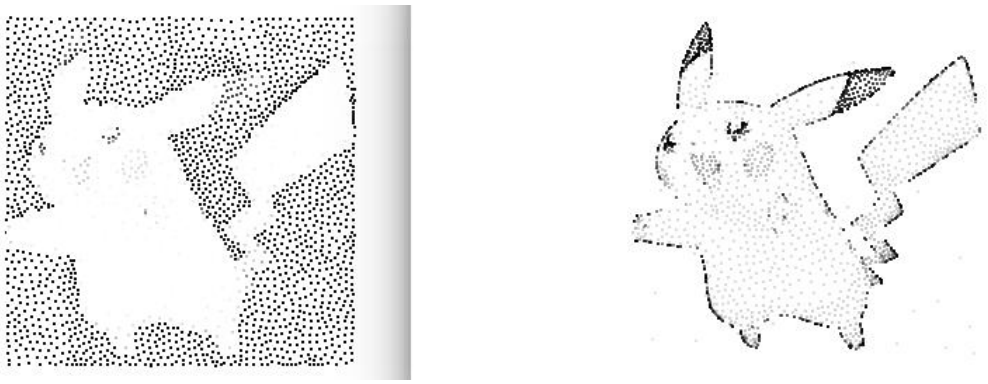

Figure 15: comparison

Folder hedcuter2 has the codes for the first improvement. And folder hedcuter3 has the second improvement. And the output image makes more sense now.