Advanced Algorithms Fall 2015

Voronoi Stippling

Prof. Jyh-Ming Lien

The goal of this assignment is to deepen your understanding on various implementations of 2-d Voronoi Diagram. In this exam, you are given two implementations of 2-d Voronoi Diagram: the Fortune's algorithm and an algorithm using well-known image-based wave propagation method. You task is to (1) understand the implementations, (2) compare the results from these two implementations, and (3) finally improve the image-based method.

What to submit: You need to turn in a report in LaTeX (see the template in report folder). Your report should include three sections: a summary of what the code does (for both implementations), your discovery of the differences in output, and your improvement. In the second and third sections, you should include all the example outputs (visual and/or statistical results). In the last section you should report known bugs, and known limitations.

How to submit: You should use github to maintain your code and the report. Email me your github clone command to my email address *jmlien@cs.gmu.edu* before the deadline.

Due: Nov 26, 2017. At 11:59 pm.

1 Part 1: Understand the implementations (30 pts)

1.1 What should you do?

Your goal is to get a full grasp of what the code does in both implementations: hedcuter and voronoi. Your summary should provide algorithms for computing Voronoi diagram, Centroidal Voronoi tessellation (CVT), and stippling methods from both implementations.

Hint 1: Both methods are based on the paper by Secord, Adrian. "Weighted voronoi stippling." Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering. ACM, 2002. It is highly recommend that you read the paper first.

Hint 2: To compile hedcuter code, please use the solution file in folder *hedcuter/code/vc_files*. This code requires OpenCV. By default, it requires 64 bits installation of OpenCV on Windows. The solution file also uses two environment variables called "OPENCV_INCLUDEDIR" and "OPENCVX64_LIBRARYDIR" the point to the include and library folders on the system. Therefore, make sure that you have those variable defined before you compile. You can consult OpenCV documents (*docs.opencv.org*) if you encounter problems.

Hint 3: To compile the second code, you will need boost (*www.boost.org*) and define two boost related environment variables: BOOST_INCLUDEDIR and BOOST_LIBRARYDIR before you compile. Each of these variables should have the full paths to the include and library folders on the system. This implementation is obtained from *http://www.saliences.com/projects/npr/stippling*. Some details of the code can be found there.

2 Part 2: Compare the Outputs (30 pts)

2.1 What should you do?

Your goal is to compare the output of these two implementations. Use the images in folder hedcuter/images or use images of your own, please show the differences visually (e.g. circle the areas of difference) and also in writing. You should also provide some discussion of where these differences are from. Here are some questions for you to consider. You are free to explore beyond these questions.

1. Do you get the same results by running the same program on the same image multiple times?
2. If you vary the number of the disks in the output images, do these implementations produce the same distribution in the final image? If not, why?
3. If you vary the number of the disks in the output images, is a method faster than the other?
4. Does the size (number of pixels), image brightness or contrast of image increase or decrease their difference?
5. Does the type of image (human vs. machine, natural vs. urban landscapes, photo vs. painting, etc) increase or decrease their difference?
6. Are the outputs of these stippling methods different the hedcut images created by artists (e.g. those from the Wall Street Journal)?

3 Part 3: Improve "hedcuter" code (40 pts)

3.1 What should you do?

Provide at least two improvements (each will worth 20 points) to the hedcuter code. Below are some possible improvements that you can do. In your report, you should show the improvement either visually or/and statistically for timing/performance results using images of your own, i.e. do not use the images in hedcuter/images. Extra bonus of 20 points will be given for an additional improvement.

1. Improve the distribution of the disks to avoid unnatural clustering of the disks. One idea is to use higher image resolution (using subpixels) for computing the centroids of Voronoi cells.
2. Improve the computation efficiency. One way of doing this is via GPU. You can try the method by Hoff III, Kenneth E., et al. "Fast computation of generalized Voronoi diagrams using graphics hardware." Proceedings of the 26th annual conference on Computer graphics and interactive techniques, 1999. The implementation should be pretty simple if you know OpenGL.
3. Add functionality to generate colorful disks. For example, you can implement functions that are not available in hedcuter code but provided in the voronoi code.