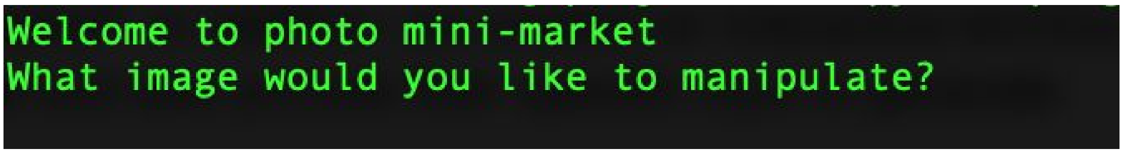
CS6161 - Algorithms - Final Project Report
Photo Mini-Market

Parima Sahbai (ps4ww), Daniel Perez (dep7sc), Zoya Yeprem (zy8ez)



Introduction
In this project, we aim to implement two different image manipulation algorithms: Voronoi and Seam-Carving. Due to the nature of image manipulation, in honor of photoshop we decided to call our packaged algorithms “Photo Mini-Market”. In addition, we created an interactive terminal UI to allow the user to choose between them and also provide their specific input to generate the desired output.

Implementation details
As described in the previous section, we implemented 2 algorithms as follows:

1. **Seam carving**

   By definition, “Seam Carving is a simple image operator that supports content-aware image resizing for both reduction and expansion”. A seam is an optimal 8-connected path of pixels on a single image from top to bottom, or left to right, where optimality is defined by an image energy function. Our seam carving approach makes use of the python library Pillow that allows us to manipulate images. Originally when looking into this concept we attempted a recursive approach that recursively goes through the image to find the lowest energy function. This approach took way to long for a simple image, henceforth we switched our approach to that of a dynamic programming algorithm that allows us to find a seam in the picture that has the lowest energy. An image is composed of a 2D-Matrix($m * n$). There is preprocessing involved. We have to first go through all of the images and build a matrix of the pixels red, green, and blue intensities. From there we are able to run our seam carving. Seam carving starts by building an energy list where it goes through the image and for every single point it calculates its energy. Which the distance formula of the surrounding pixels in accordance with all of the red, green, and blue intensities. Using the dynamic programming method makes the program more efficient because it does not have to recalculate a lot of the same pixels, hence improving the runtime. The program outputs a list of points that compose the lowest energy seam. We then use this seam to demonstrate the most unimportant part of the image. Where if deleted would not cause any distortion to the image.

The red line in the duck image is the lowest energy seam. On the right, we have the weight of the seam and the pixels that compose the lowest energy seam.

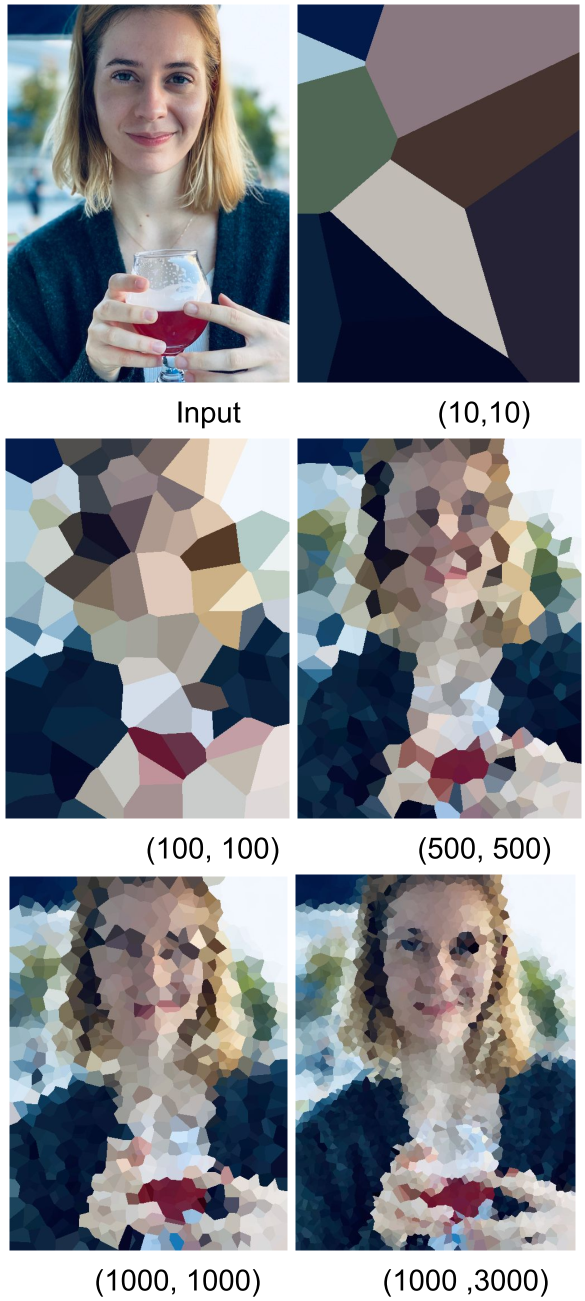
2. **Voronoi Diagram**

Voronoi algorithm was introduced in the class and we chose to implement its algorithm as part of our project. In our case, we had a few options in terms of how to approach the diagram. There were many online explanations and algorithms for random voronoi points, however algorithms offering explanation on turning an image into a voronoi diagram were minimal. After research, the plan was to try to find the points with the most entropy, making it important to the picture and to create a diagram with the voronoi diagram algorithm after. After trying this approach, we realized that for a picture like this using random points would work just as well, and even better as it would not abandon points the algorithm considered unimportant by mistake and thereby randomizing and introducing less errors in picking the correct points due to the randomization. Using this approach, we were able to first, generate and pick random pixels within the image, and saving their color to a third array to keep track of. The last step is to draw the triangles which we do by going over the new canvas we created (same size as the old) and calculating the hypotenuse of two given points similar to k-means++ clustering and using it to calculate the points closest to each site (cell) and group them.[1]

Results

Cell size = c
Points = p
Format (c,p)

[1] Reference: https://rosettacode.org/wiki/Voronoi_diagram#Python


Input
(10,10)
(100, 100)
(500, 500)
(1000, 1000)
(1000 ,3000)