CSCE448-500 Project 5

Lam Nguyen

April 7, 2023

1 Seam Carving

1.1 Overview

In this assignment, the task is to take a source image and scale it with a provided width and height factor between 0 and 1. Of course we could simply just half the image by taking every other pixel, but this would lead to artifacts as discussed in class. Instead what we could do is apply seam carving in order to achieve this in a manner that is clean and free of artifacts.

The general idea behind the implementation is that in order for seam carving to work is that every pixel's energy needs to be calculated. Then a list of seams need to be created and are ranked by energy, with low energy seams being of least importance to the content of the image. Seams can be calculated via the dynamic programming approach that will be detailed. Then the lowest energy seams will be removed, and the process will be repeated until the desired size is reached. Overall the entire process can be simplified to four steps seen below:

1. Find the sum of absolute gradients in x and y directions to get energy matrix M.
2. Find the best (lowest energy) seam using dynamic programming. The steps are:
    - Make a copy of your energy matrix M.
    - Set the values of every entry in the matrix except for the first row by adding to it the minimal value in any of the cells above it in the seam: $M(x, y) = E(x, y) + \min[M(x\ \ 1, y\ \ 1), M(x, y\ \ 1), M(x + 1, y\ \ 1)]$
    - Find the minimal value in the bottom row of the scoring matrix.
    - Trace the seam from the bottom by following the smallest value in any of the positions above it in the seam.
3. Remove that seam in the image
4. Repeat steps 1-3 until desired size is reached

The following images in the following subsection are my results from running the given first three images and my own personal example of a failure case. I did not do the extra credit as an undergraduate so those images are not attached. One thing to note is that some images have been purposefully resized or else the raw size of them would make the report very hard to read. I hope you do not mind.

1.2 Results

(a) Original Image

(b) Half Width Image

(c) Half Height Image

Figure 1: Result 1

(a) Original Image

(b) Half Width Image

(c) Half Height Image

Figure 2: Result 2

(a) Original Image

(b) Half Width Image

(c) Half Height Image

Figure 3: Result 3

(a) Original Image

(b) Half Height Image

Figure 4: Failure Case (Resize 50%)

1.3 Analysis

For the most part, using seam carving can be effective way to resize images as seen in Figures 1-3, it does have drawbacks as seen in Figure 4. To explore why Figure 4, we can view its energy map of the image rotated below:

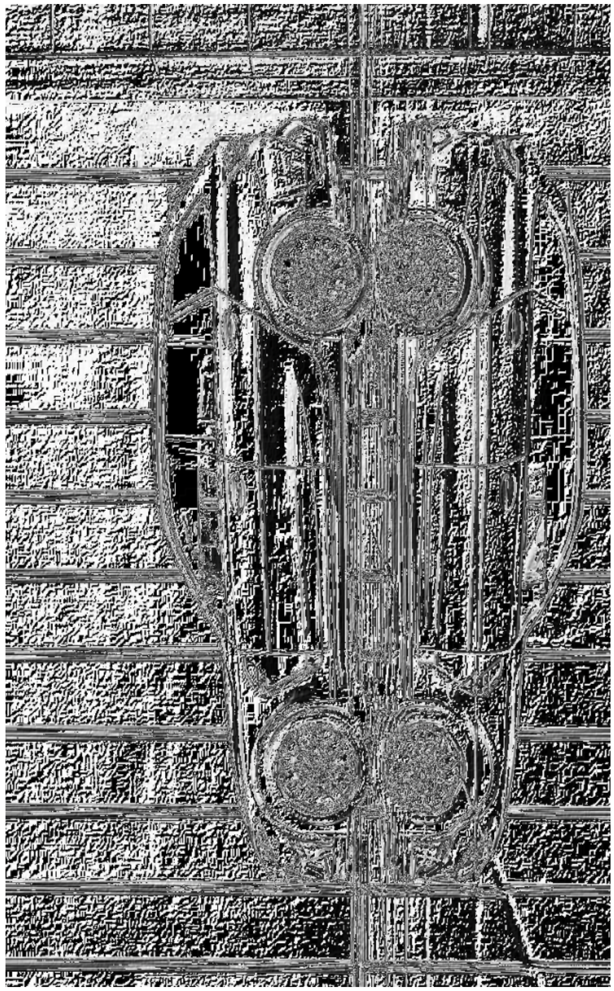

Figure 5: Energy map of Figure 4

Looking at Figure 5, we can notice on the windows and doors of the vehicle, there is a fairly low amount of energy; the reflection of the same features also have low energy but not as crazy since the reflection does not capture all of the details of the car. This is important to keep in mind as the point of seam carving is that it ranks the energies and removes the least important ones (denoted by darker color) in order to reduce the size. Since the windows and doors have the lowest energy, we can expect that the reduced image will not have much of it as well, which is evident in the result. Some possible reasons why the doors and windows have low energy is because they have a uniform color and texture throughout, which can make them less visually important than other parts of the car with more intricate details or patterns like the wheels that are largely preserved.

In regards to any challenges faced during the implementation, I think the biggest challenge I faced early on was that matplotlib.pyplot reads images as RGB but cv2 wants BGR. This was insanely confusing early on when trying to convert to grayscale to capture the energy, but some quick documentation read up helped by just being consistent. I also removed dividing by 255 on the Read function. Besides that, this was fairly straightforward.