Seam Carving

CS 4670: Final Project Report

Adam Sorrin (acs73), Cari Reiche (car56), & Garrett Bernstein (gsb29)

1. Introduction

Conventional image shrinking methods consist of evenly downsampling to achieve the desired size. This method can lead to distortion and also means that every object in the image is scaled down proportionally. Ideally when choosing pixels to remove, those belonging to more interesting objects in the image would be given preference while uninteresting parts of the image are removed. To achieve this desired effect we implement Seam Carving on the Nokia N900, which consist of three parts: Calculation of pixel interest score, determination of minimum interest seam spanning the image, and removal of that seam. We discuss the implementation of those three steps, along with the results of running the algorithm on the cell phone.

The Seam Carving algorithm is introduced and discussed by Avidan and Shamir [1]. Our extension to that paper is to implement and optimize the algorithm for the Nokia N900.

2. Methods

Seam carving is accomplished by a 3-step process:

1. Calculating the energy of each pixel. This is done by smoothing the image and then computing the first x- and y-derivative at each point using a 3x3 Sobel filter.

$$Energy = |I_x| + |I_y|$$

2. Using dynamic programming to find a seam from top to bottom (or left to right) with the least total energy. A seam is defined as a continuous series of pixels. Thus, if you have a vertical seam, it consists of one pixel per row, with pixels in adjacent rows in either the same or an adjacent column.
3. Removing the pixels along the lowest-energy seam to create a smaller image. This will remove a less-interesting part of the image, while still preserving continuity in the image.

These steps are repeated until the image has shrunk to the desired size.

Our program allows the user to change both the width and height of the image. We alternate between vertical and horizontal seam carving until we reach the desired dimensions (or stop carving in one direction if that dimension has already been achieved).

Our first attempt at this functionality allocated a new image for each subsequently smaller image and recalculated the energy function for the entire new image. This approach proved to be very slow, so we redesigned the algorithm to compute the new blur, x- and y- derivatives, and the new shrunken images all in place. We then realized that recalculating the blur and derivative of most of the pixels was redundant and that only pixels in the vicinity of the previously removed seam had altered properties. We were able to improve our algorithm by eliminating the redundant calculations and only

recalculating the areas around the removed seam.

These optimizations gave us significant speed-up on the computer, reducing computing time from 35-40 seconds to 17 seconds (depending on the size of the image and the number of rows and columns to be removed). However, the runtime on the phone was barely affected, still taking around 2 minutes to compute a carved image.

An additional optimization we considered was avoiding resizing any of the blur, derivative, or original images until the very end as the process of shifting all to the right or below a seam is computationally expensive. Instead we would mark each pixel removed in a previous iteration with a flag and take that into account when recalculating blur and derivatives. Then at the very end we would simply iterate through the image once, copying over unflagged pixels into the new shrunken image. We did not have time to implement this, as many of the implementation details are non-trivial.

OpenCV functions were only used to calculate the initial blurring of the input image and the initial Sobel filters (the first derivatives in both the x and y directions). After the initial calculation we computed these "by hand" only on pixels in the image that were affected by removal of a neighboring pixel.

The nature of this algorithm made it pretty easily divisible among our group members. Cari focused on the energy calculation, getting the program to run on the Nokia N900 phone, and setting up the GUI. Garrett worked on the dynamic programming implementation to obtain an optimal seam along which to carve. Adam focused on obtaining the new image once an optimal seam has been selected. This involved moving away from the initial implementation of simply allocating a new image and instead altering the image in place. Additional improvements were made to each of these steps by all three members.

3. Results

Below are three examples of our algorithm shrinking images. The first was executed on the computer and the last two were executed on the phone.

Image 1a (top): Picture of a bird

Image 1b (bottom left): Bird image resized via Seam Carving (on computer)

Image 1c (bottom right): Bird image resized via simple compression

Image 2a and 2b: Picture of Duffield and the resized image. Algorithm executed on the phone.

Image 3a and 3b: Picture of Robotics Lab and the resized image. Algorithm executed on the phone.

4. Discussion

As evidenced in the figures above we were able to to implement Seam Carving on the N900 with some degree of success. Figure 1a shows a picture of a bird against a blurry background. Figure 1b and 1c show the same image shrunken horizontally by ~100 pixels via Seam Carving and simple compression respectively. Seam carving chooses to remove seams mostly to the left and right of the bird, though some seams were removed from the bird itself. Regular compression however, removes evenly spaced

columns of pixels and thus significantly distorts the bird.

Figure 2 illustrates both success and a limitation of Seam Carving. Shrinking from Figure 2a to Figure 2b preserves interesting objects in the picture, such as the banner and most of the tables, but the straight diagonal line of the window sill becomes broken. This is because the line spans most of the image and thus it is very hard to find a vertical seam that will not break up the line.

Figure 3 shows a very busy picture of a corner of the Robotics Lab. Seam Carving removes chunks of the white wall as expected and desired. You may notice it also removes the side of the white shelf which is a single solid color (and thus very low energy) but spares the interesting objects on the shelves themselves.

In addition to the one mentioned above, there are a few other limitations of our Seam Carving algorithm implementation. While faces are considered interesting objects, skin is generally a continuous swath of color. Therefore the algorithm deems skin pixels to have low energy and picks seams to go through them. Removing even a single row of pixels from a face will noticeably distort it to the human eye. To avoid this problem we could implement face detection and then specify the dynamic programming to only pick seams that avoid the pixels of the faces.

Another limitation is that the image needs to have uninteresting parts to remove. If the image has interesting features across the entire breadth, such as one consisting entirely of tree branches and leaves, then any seam the algorithm deems to be of low interest will still contain interesting objects. Ideal images for running this program on should contain "boring" areas.

4.1 Future Work

Currently, our system only allows images to be shrunk. Possible future work would include image expansion. Upon finding a seam of minimal interest we would interpolate to determine the intensity between the seams neighbors and then add an adjacent seam.

Another interesting feature that could exploit this function is object removal. This would require some user input to define the area or object to be removed. Then the algorithm would simply do the seam carving while making sure to select pixels inside the area to be removed.

To improve the the quality of the shrunken carved image we could change the ordering of seam removal. Instead of simply alternating between removing vertical and horizontal seams would could implement implement a second dynamic programming algorithm to determine the exact ordering of seams to remove that would minimize the energy removed.

5. References

[1] Avidan, Shamir, "Seam Carving for Content-Aware Image Resizing." http://cs.nyu.edu/~fergus/teaching/comp_photo/readings/AvidanShamir2007-SeamCarving.pdf.