Implementing Seam Carving for Content-Aware Image Resizing

Russell Jones, rjk2@williams.edu, 17rjk2
Julia Goldman, jsg2@williams.edu, 18jsg2
Zijie Zhu, zz4@williams.edu, 17zz4,
Yiheng Zhang, yz4@williams.edu, 18yz4

1. Introduction

The tremendous variety and flexibility of display devices in our modern world has put pressure on digital media producers to be able to readily adapt to changing specifications. In this paper, we tackle the problem of image resizing in relation to these issues. Common resizing methods today include scaling and cropping, but these methods do not operate at optimal efficiency as neither takes into account the content of the image (Figure 1). To resize images while effectively preserving the original image content, one can use a different approach called seam carving. This resizing method was first introduced by Shai Avidan and Ariel Shamir [Avidan and Shamir 2007] and our paper aims to reproduce and implement their seam carving method.

(a) (b) (c)

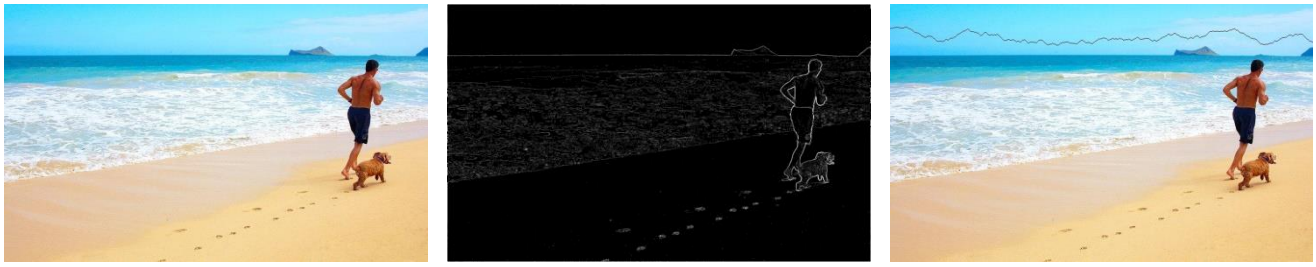
Figure 1: Scaling versus Seam Carving. (a) is the original image at 400 x 400 pixels; (b) is scaled to 200 x 400 pixels; (c) is seam-carved to 200 x 400 pixels. Note how the seam-carved image conserves the boat

2. Methodology

2.1 Approaching the Problem

Seam carving operates by removing pixels with low importance from the image, thereby conserving its key features while also changing the display specifications. The importance of a pixel is measured by its energy, which is calculated with an energy function. There are many different energy functions based on different criteria. For our project we chose a function similar to the Sobel operator and found that it worked well for most images [Krishnamurthi 2012].

To balance the goals of removing the lowest energy pixels and of conserving the overall structure of the image, we remove seams of low energy pixels from the image. A seam is a vertical or horizontal path of connected pixels that crosses the image as seen in figure 2(c). Horizontal seams have one pixel in each column of the image, and vertical seams have one pixel in each row of the image. With dynamic programming, we can calculate the seam with the lowest total energy. By then adding or removing vertical and horizontal seams with low energy, we can reduce or enlarge images while preserving their important parts.



(a) (b) (c)

Figure 2: (a) shows the original image, (b) shows the energy map of the image, and (c) shows the horizontal seam with lowest energy based on the energy map

2.2 Energy Mapping

The first step in the seam carving process is to transfer the data in an image file into data that we can manipulate. After the image data is retrieved, the energy of each pixel is calculated using an energy equation [Krishnamurthi 2012] that looks for the difference in RGB values between a pixel and its 8 neighbors.

$$Energy(x,y) = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$$

As seen in figure 2(b) the energy values can then be mapped in grayscale to create an energy map of all of the pixels.

2.3 Seam Carving Sum Algorithm

Once the energy map is calculated, dynamic programming is used to calculate the energy sums of the seams. The dynamic programming algorithm works by finding the minimum energy path to get to each pixel on a row by row basis. For the vertical seams, the vertical sum fields of the top row of pixels are set to their energy value. Then for the next row of pixels, the vertical sum field is set to the pixel's energy value added to the minimum of the energy sum of the three possible "parent" pixels that are above that pixel. This process is continued until the vertical sums for the last row of pixels are set. By tracing that path of pixel back up the image finding smallest energy sum each time, we can locate the vertical seam to remove. An almost identical process happens during horizontal sum calculation, with the seams starting on the left edge of the image and moving right.

The above sum calculation method is a "backward" method of finding seams, as it removes a seam only according to the current energy distribution of the image. We also implemented a "forward" sum calculation [Rubinstein et al. 2008] that calculates sum fields based on what will happen if the pixel is removed. The algorithm works in a very similar way to the backward one, but instead of looking at the

energy values for each pixel, the forward approach is concerned with the energy that is introduced to the image when a seam is removed due to pixels having new neighbors. We once again use a dynamic programming algorithm to calculate the forward sums for each pixel, and in our command-line interface, the user can choose to use either forward or backward seam carving.

2.4 Seam Removing and Adding

Once the sum fields have been calculated, seams can either be removed or added to the image. For seam removal the pixels are simply taken out of the image one seam at a time. After every seam removal the energy and sum fields of the remaining pixels are reset to create a better final image. For vertical seams, this is done by erasing the individual pixels in the seam one at a time, but for horizontal seams we bring the seam of pixels to be removed down to the bottom row of the image before we delete them due to our underlying data structure. For seam addition, the lowest energy seam is found, and then a new parallel seam is added to the image directly below it for horizontal seams, and to the right of it for vertical seams. The color for the added pixels is then set to an average of the color of the pixels directly above and below those pixels for horizontal seams and to the left and the right of them for vertical seams. For added seams, the energy of the new seam is then artificially increased using a special energy field, as this design makes it less likely that new seams will congregate in one area, and thus create artifacts in the image.

2.5 Marking Areas for Conservation and Deletion

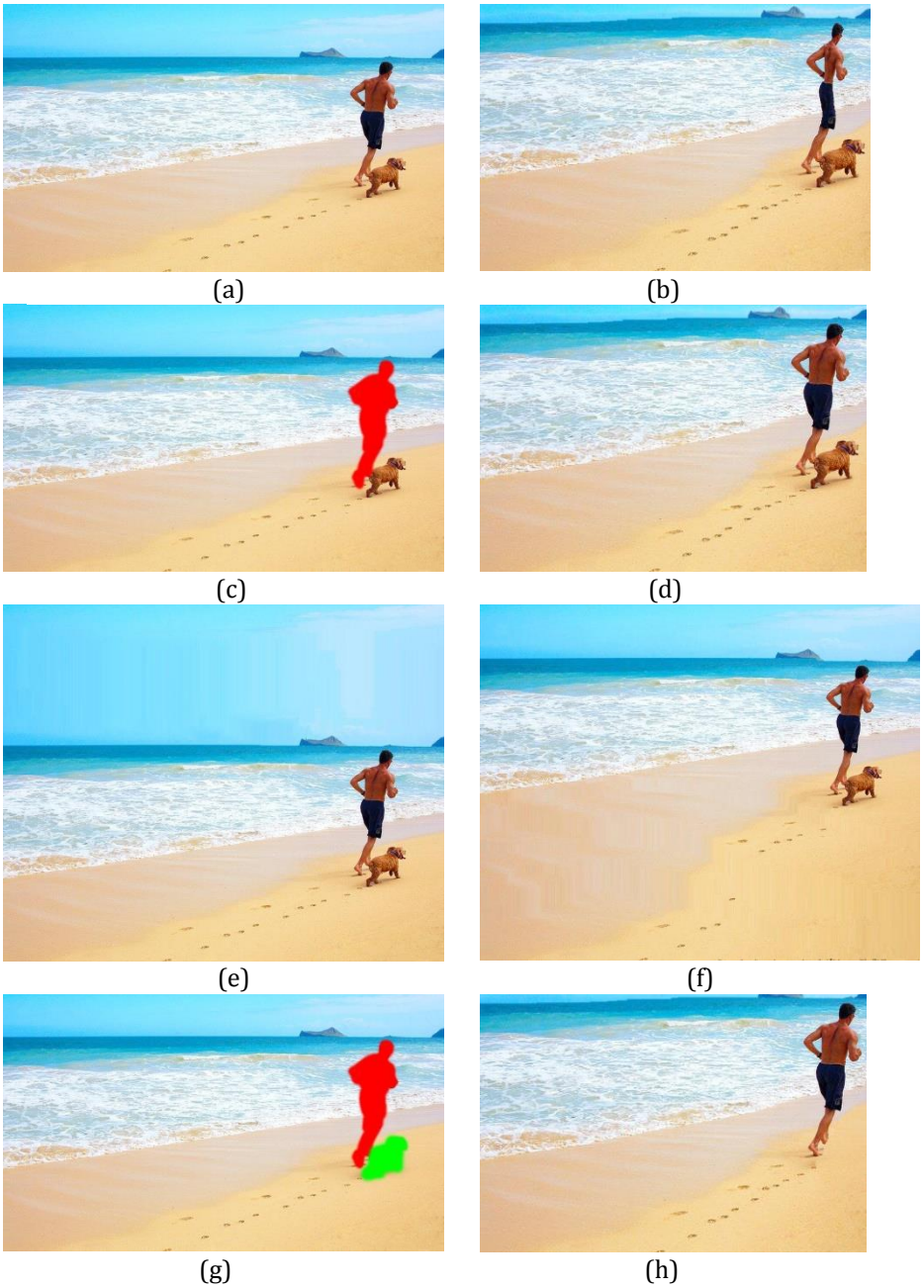
In our implementation we also give the user the ability to mark areas of the image to be conserved or erased. This is accomplished by using the special energy field mentioned above. If the user marks an area to be conserved, then the special energy field of those pixels will be kept artificially high, making sure that seams do not pass through those areas. If the user marks an area to be erased, then the special energy field of those pixels will be set artificially low, giving incentive for low energy seams to pass through those areas.

3. Results

3.1 Resulting Images

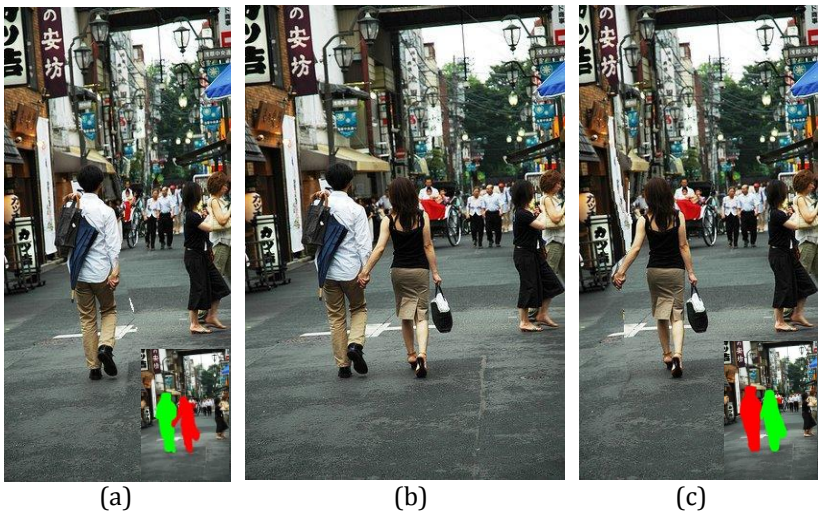
(a) (b)

Figure 3: (a) is the original image with 1428 x 968 pixels, and (b) is resized using backward seam carving to 1128 x 968 pixels



(a) (b) (c) (d) (e) (f) (g) (h)

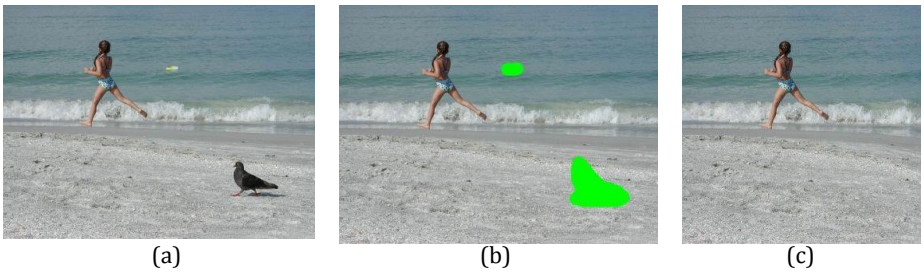
Figure 4: (a) is the original picture at 900 x 548 pixels. As seen in figure 2 (b), our energy function does not identify the man's back as a region of high importance. Therefore, in (b) when we resized the image to 750 x 548 pixels using the backward method, the man's back got distorted. In (c) and (d), we added a high-energy mask to conserve the man's body so that when we resized the image again, his back did not become distorted. In (e) and (f), we added horizontal seams to the image using the forward and backward methods respectively to increase the image ratio to 900 x 750. As seen in the images, minor artifacts were introduced. In (g) and (h), we used a mask to mark the dog with low energy, and when we ran our seam carve down to 750 x 548, the dog was removed from the image and the man was conserved.



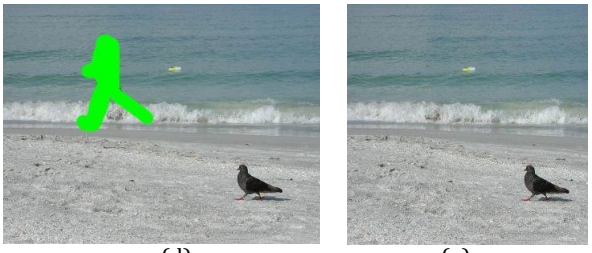
(a) (b) (c)

Figure 5: (b) is the original image at 333 x 500 pixels. For (a) and (c), we removed the woman and the man respectively using the masks at the bottom-right corners of the images.

Figure 6: (a) is the original image at 400 x 300 pixels. In (c), we removed the pigeons and the Frisbee using the mask shown in (b). In (e), we removed the girl using the mask shown in (d).



(a) (b) (c)



(d) (e)

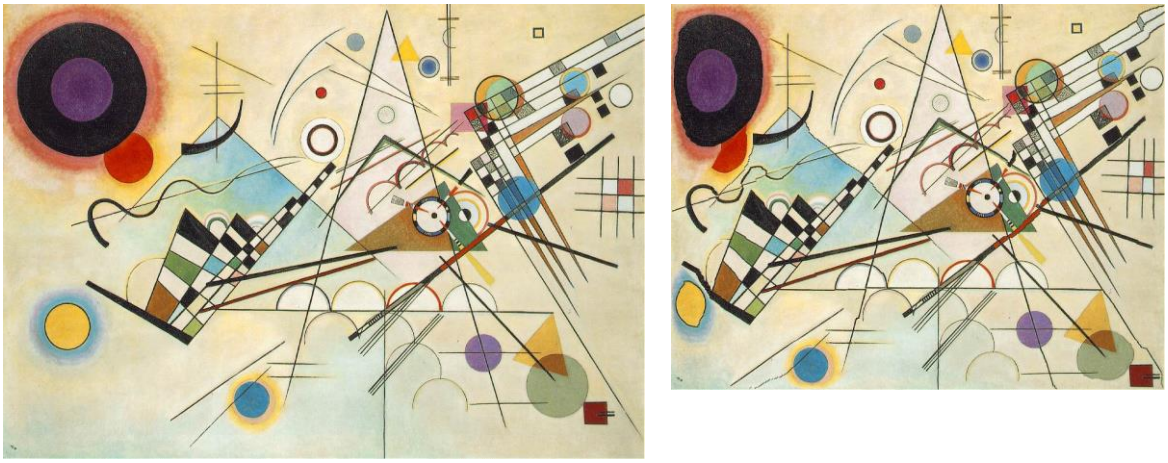
Figure 7: (b) is the original image at 600 x 396 pixels. In (a) and (c), we reduced the image to 400 x 396 pixels using the forward and backward methods respectively. As seen in the tree trunks and the mountains on the right, the forward method does a better job of resizing the image. For (d) and (e), we increased the image size to 900 x 496 pixels using the forward and backward methods respectively. There is little difference between the two methods in this case.

(a) (b) (c)

(d) (e)

Figure 8: Highlights some of the limitations of seam carving. (a) is a complex image with text. When we attempted to seam-carve this image (b), the letters and some of the berries became distorted.

(a) (b)



(a) (b)

Figure 8: Depicts further limitations of seam carving. The complex artwork of Kandinsky is significantly distorted when seam-carved. Specifically, the seam carving algorithm does not perform well in preserving straight lines and geometric shapes.

4. References

Avidan, S., Shamir, A.: "*Seam carving for content-aware image resizing*," In: *SIG-GRAPH* (2007)

Krishnamurthi, Shriram. "*Programming with Data Structures and Algorithms*," *Seam Carving – CS19 Assignments* (2012)

Rubinstein, Michael, Ariel Shamir, and Shai Avidan: "*Improved Seam Carving for Video Retargeting*," *ACM Transactions on Graphics* 27.3 (2008)

*Image class used created by Sam Donow and Morgan McGuire