# Midterm Research Project: Seam Carving for Content-Aware Image Resizing

**Rhythm Muhtadyuzzaman Syed**
Georgia Tech CS 6475: Computational Photography
msyed32@gatech.edu
Demonstration link: https://youtu.be/3CUu-XcNyuY

## Abstract

For the midterm project in CS 6475 this semester, a replication study was done on two papers by Avidan et al. and Rubinstein et al. The main objectives are to implement the algorithms and methodologies presented in the literature to learn more about the computational photography process. In this paper, the results and the implementation of the material is discussed in depth.

## 1 Description of Algorithms

In the 2007 paper *Seam Carving for Content-Aware Image Resizing* by Avidan et al., the algorithm presented at a high level is as follows: Based on the input image of the program, compute the energy function (1). The energy function is simply the summation of the gradients or partial derivatives of the input 2D image.

$$e_1(\mathbf{I}) = \left|\frac{\partial}{\partial x}\mathbf{I}\right| + \left|\frac{\partial}{\partial y}\mathbf{I}\right| \quad (1)$$

Next, compute the cumulative minimum energy matrix **M** for all possible seams that may be evident in the image. This is computed efficiently using dynamic programming (2).

$$M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1)) \quad (2)$$

The next step is to find a seam using the matrix **M** through backtracking while ensuring 8-connected path of pixels along the way. A vertical seam (3) is defined as follows:

$$\mathbf{s}^x = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, \text{ s.t. } \forall i, |x(i) - x(i-1)| \leq 1 \quad (3)$$

The seam will extend first row of the image all the way to the last. A similar function can be applied to the horizontal seam extending from the left of the image to the right. For image size reduction, the seam can be simply removed from the original image to generate a new image. For a specific reduction **k**, where k = original image size – new image size, k seams will be removed in a repetitive manner, recalculating the energy and cumulative minimum energy matrix along the way to find new seams that will need to be removed. For image enlargement, the found seam will need to be re-added to the original image. As explained in the paper, there are two methods for enlargement. Once a seam is found, an artificial seam can be created by taking the average of the left and right neighboring pixels. Unfortunately, this is not the most efficient way for enlargment as results often show a stretching effect in the output. The second method for enlargement includes following a process of *traversing back in time* to determine which seams would have been removed first then re-add them to the image whilst duplicating them in an iterative manner.

From the 2008 paper *Improved Seam Carving for Video Retargeting,* an improved algorithm for the cumulative minimum energy matrix is introduced. This algorithm is coined as *Forward Energy* (4), while the previous method is considered *Backward Energy*. The main difference in these two methods is that for Forward Energy, the cost of pixel boundaries are now taken into account when calculating the minimum energy. It is defined as the following:

$$M(i,j) = P(i,j) + \min \begin{cases} M(i-1,j-1) + C_L(i,j) \\ M(i-1,j) + C_U(i,j), \\ M(i-1,j+1) + C_R(i,j) \end{cases} \quad (4)$$

With the costs defined as:

$$C_L(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j-1)|$$
$$C_U(i,j) = |I(i,j+1) - I(i,j-1)|$$
$$C_R(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j+1)|$$

## 2 Implementation

The implementation for this project was focused around replicating the results of the two papers using the algorithms described in the previous section. For this purpose, a python script was created using Classes and Object Oriented Programming for seamless integration and implementation.

### 2.1 Program Entry Point

To initiate the program, the SeamCarving class is called based upon the input arguments of the script.

```
SeamCarving(input_filename, output_filename, dimension_change, energy_algorithm, display_seams)
python main.py 'images/fig5.png' 'fig5-test-2.png' '0, -100' 'backward' False
```

For this example, the main.py script is called with the cmd line arguments, **input_filename**, **output_filename**, **dimension_change**, **energy_algorithm**, and **display_seams.** The input_filename is the path for the input image needed to be processed. The output_filename is the

path where the output will need to be stored. The dimension_change specifies how much to change the aspect ratio of the image, with an example being (0, -100) referring to changing the height by 0 pixels and decreasing the width by 100 pixels. The energy_algorithm specifies whether to use the forward or backward algorithm method for the calculation of the cumulative minimum energy matrix. The display_seams argument is a boolean that determines whether or not to show the end user the physical seams in the image after calculation.

## 2.2 Brief Description of Functions

The program flow of this implementation follows the same method as described in the algorithms section:

```
class SeamCarving:
    # input_filename, output_filename, dimension_change, energy_algorithm
    def __init__(self, input_filename, output_filename, dimension_change, energy_algorithm, display_s
        self.input_filename = input_filename
        input_img = cv2.imread(input_filename, cv2.IMREAD_COLOR)
        dimension_change = dimension_change.split(', ')
        new_dimensions = (input_img.shape[0] + int(dimension_change[0]), input_img.shape[1] + int(dim
        self.energy_algorithm = energy_algorithm
        self.output_filename = output_filename
        self.seam_display = display_seams

        self.input_img = np.atleast_3d(input_img).astype(np.float)
        self.target_dimensions = new_dimensions
        self.new_image = self.input_img
        self.copy_image = self.input_img
        self.energy_map = np.ndarray(new_dimensions, dtype=float)
        self.cme_map = np.ndarray(new_dimensions, dtype=float)
        self.seam = np.ndarray(new_dimensions, dtype=float)
        self.seam_collection = np.ndarray(new_dimensions, dtype=float)
        self.seam_count = 0
        self.removal_seam_collection = {}

        print('Seam Carving Started!')
        self.aspect_ratio_change()
```

In the init function of the class, the cmd line arguments are first assigned to member variables. Afterwards, the energy_map and cme_map (cumulative minimum energy) are created. A seam_collection dictionary is also created to store all of the possible seams in the image manipulation process. The final function **aspect_ratio_change()** is the main function that runs a loop of finding a seam, removing or inserting the seam, checks if the desired new dimension of the image is achieved, and if not, regenerates a new energy_map and cme_map and loops again. At the end of the execution, cv2.imshow is used to display the generated images to the user.

# 3 Results

This section compares the results of this implementation to those presented in the research papers. Additional discussion of differences is also included.

## 3.1 Figure 5 from 2007 paper

Figure 5 from the 2007 paper (5) presents an output for seam removal using the backward energy algorithm.

(5)
Original

The following is the output of the implementation described in section 2 of this paper. The first image (6) presents a 50% reduction in width of the original image, from 700 pixels to 350 pixels. The second image (7) presents the same reduction in size using the improved forward energy algorithm.

(6)
Backward
Energy

(7)
Forward
Energy

These results are almost identical to the results presented in the paper, with the forward energy output being much improved to the backward energy. Some differences can be spotted around the sky region of the image where pixel boundaries were created by the seam removal process in the backward energy algorithm. The forward energy successfully eliminated this issue for more seamless blending of the new neighbor pixels.

## 3.2 Figure 8 from 2007 paper

Figure 8 from the 2007 paper (8) presents an output of seam insertion using backward energy. The images include 1 step 50% enlargement, display of seams, and 2 step 50% increase. The following are the results from the implementation in section 2. Image (9) is a 1 step 50% enlargement using an average seam insertion method. For every seam found, insert a new seam using the average of the left and right neighbor pixels. This causes a *stretching* effect. Image (10) shows the seams that were inserted. This is using backward energy.

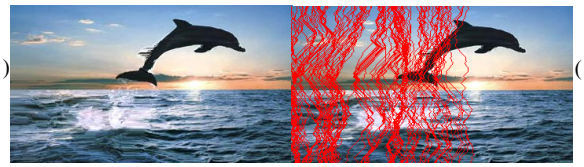
(8)
Original

(9)

(10)

Image (11) and Image (12) shows a 1 step 50% enlargement using a *back in time* seam insertion method. For a **k** increase in size, k seams are first found for removal, then duplicated.

(11)



(12)

Image (13) and (14) shows a *back in time* seam insertion for 2 step 50% enlargement.

(13)

(14)

These results matched the results from the 2007 paper for the 1 step 50% enlargement and the back in time method. Significant difference can be seen when conducting 2 step enlargement. The stretching effect can be seen on the left side of the image.

### 3.3 Figure 8 from 2008 paper

Figure 8 from the 2008 paper (15) presents a comparison of backward and forward energy. The following are results from the implementation in section 2.

(15)

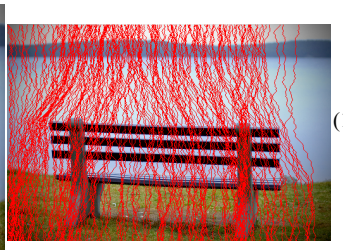
Image (16) shows vertical seam removal of 200 pixels from the width using the backward energy algorithm presented in the 2007 paper. Image (17) displays the physical seams that were removed from the image.

(16)

(17)

Image (18) and (19) shows the same removal method except using the forward energy algorithm.

(18)



(19)

These results are identical to those presented in the 2008 paper. The forward energy algorithm performed much better as compared to the backward energy.

### 3.4 Figure 9 from 2008 paper

Figure 9 from the 2008 paper (20) presents a method of seam removal and insertion by elongating a car in the image. The following are the results of the implementation presented in section 2.

(20)

Image (21) shows a condensed image using seam removal and backward energy. Image (22) shows seam removal using forward energy.

(21)

(22)

Image (23) shows seam insertion using backward energy and Image (24) shows the same using forward energy.

(23) (24)

The results from this section are similar to those in the paper. For the seam insertion and removal, forward energy performs much better, however, there are still visible artifacts in the image as compared to with the images in the paper.

## 4 Discussion

For this study, there were many things that made replication difficult from the literature. In the algorithm perspective, boundary conditions were not specifically talked about by the authors. For example, when calculating the cumulative minimum energy map **M**, if the process was at the left or right edge of the image, which neighbor pixels would be used for the dynamic programming portion of the calculation? My assumption was to only consider 2 neighbors instead of the 3 when faced with the boundary conditions. Another thing that made replication difficult was lack of information for the images provided. The authors should have included the aspect ratio and dimensions of the images used for calculation as well as the number of seams that were counted for insertion and removal. This would have helped debugged the algorithm I was trying to replicate. I overcame this by visually comparing the results with the images provided in the paper. Another assumption that was made for the algorithm was since the cme_map (cumulative minimum energy) was created by traversing the energy_map from the second row to the last, what would be stored in the first row of the cme_map? My assumption was to store the entirety of row=0 from the energy_map, however this was not clear from the paper. Overall, the project as a whole was a great learning experience and being able to read literature from the field of computational photography helped me understand the ongoing research and methods used today by many authors.

## 5 References


[Avidan *et al*., 2007] Shai Avidan, Ariel Shamir, Mitsubishi Electric Research Labs and The Interdisciplinary Center & MERL. *Seam Carving for Content-Aware Image Resizing.*

[Rubinstein *et al*., 2008] Michael Rubinstein, Ariel Shamir, Shai Avidan, Mitsubishi Electric Research Labs, Cambridg, The Interdisciplinary Center, Herzliya, and Adobe Systems Inc. *Improved Seam Carving for Video Retargeting.*