An Implementation and Exploration of Seam Carving

Logan Donaldson
Jeremy Wang

May 15, 2022

1 Introduction

Problem Statement: The resizing of images is a problem frequently encountered across many domains including in the creation of fixed format presentations such as slideshows and papers. The two most commonly employed approaches, cropping and interpolation, each have drawbacks. Cropping is only capable of shrinking an image by removing pixels in the image's periphery. Interpolation, the resizing method used by programs such as Microsoft PowerPoint when a user clicks and drags an image's corner, squashes and stretches the image contents. Furthermore, for low-resolution images, the interpolation method will produce increasingly blurry results as image size grows.

Solution: One alternative approach which seeks to address these drawbacks is seam carving, introduced by Shai Avidan and Ariel Shamir in their 2007 paper titled "Seam Carving for Content-Aware Image Resizing" [1]. As the paper's title suggests, seam carving alters the dimensions of an image while taking into consideration the image's composition. Furthermore, seam carving is capable of removing (or adding) pixels from the image's interior and alleviates many of the distortion issues which plague interpolation. See **Figure 1** for a practical comparison of resizing methods.

2 Shrinking Algorithm

For simplicity we will initially focus our attention on image shrinking. The seam carving algorithm takes as input an image and the desired dimensions of the output image from which the number of seams to be removed can be computed.

Algorithm 1 Seam Carving for Image Shrinking

```
1: $j \leftarrow 0$
2: $n \leftarrow$ number of seams to remove
3: $I \leftarrow$ input image
4: while $j < n$ do:
5:     $E \leftarrow$ compute_energy_function($I$)
6:     $s \leftarrow$ find_minimum_seam($E$)
7:     $I \leftarrow$ remove_seam($I$, $s$)
8:     $j \leftarrow$ j+1
   return $I$
```

3 Energy Functions

It is through the energy function that the algorithm takes into consideration the content of the image. An energy function accepts an image as input and outputs a saliency map of the same dimension. For our purposes we will consider energy functions of the form $e : R^{m\times n} \rightarrow R^{m\times n}$. Note that our energy functions are evaluated on 2D images so color images must be converted to gray scale prior to function evaluation. A good energy function will take on large values at positions corresponding to the "interesting" regions of the input image and small values elsewhere. Thus, edge detectors and corner detectors are reasonable choices. Unless otherwise noted, all examples of seam carving included in this paper use the L^1-Norm of the image's gradient as the energy function. That is, $e(I) = \left|\frac{\partial I}{\partial x}\right| + \left|\frac{\partial I}{\partial y}\right|$ where the partial derivatives of the image are computed via convolution with the usual Sobel filters,

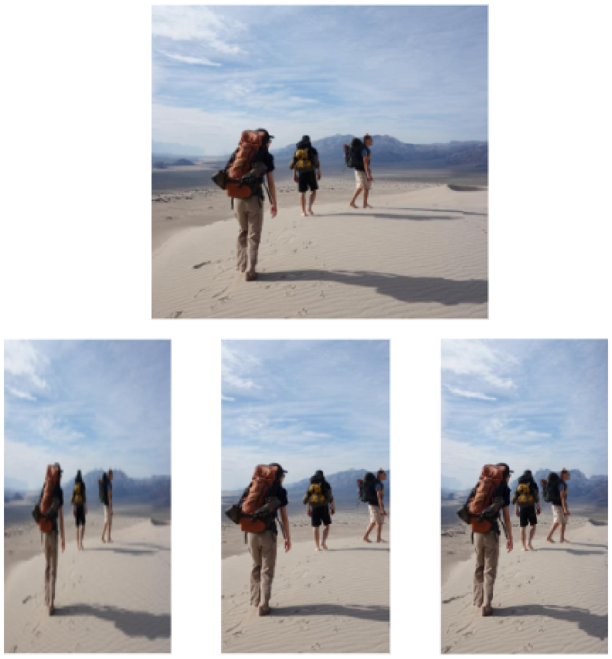


Figure 1: (From Left to Right, Top to Bottom) Original Image; Interpolation Resizing; Cropping; Seam Carving

$$h_x = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \; h_y = \frac{1}{8}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Three additional energy functions were also tested,

L^2-Norm : $(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2$

Harris Corner Detector: $\det(\bar{M}(x,y)) - \alpha[(\mathrm{Tr}(\bar{M}(x,y))]^2$

Histogram of Oriented Gradients (HOG): $\dfrac{|\frac{\partial I}{\partial x}| + |\frac{\partial I}{\partial y}|}{\max(\mathrm{HOG}(I(x,y)))}$

The partial derivatives in the L^2-Norm are computed in an entirely equivalent manner to those in the L^1-Norm. The matrix $\bar{M}$ is equivalent to the definition provided in class, namely

$$\bar{M}(x,y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} w(k-x, l-y) \cdot \begin{bmatrix} (\frac{\partial I}{\partial x}(k,l))^2 & \frac{\partial I}{\partial x}(k,l)\frac{\partial I}{\partial y}(k,l) \\ \frac{\partial I}{\partial x}(k,l)\frac{\partial I}{\partial y}(k,l) & (\frac{\partial I}{\partial y}(k,l))^2 \end{bmatrix}$$

where w is the window function. We use a window size of 5 and $\alpha = 0.04$. An explanation of the HOG is given in **Section 4**.

In our albeit limited testing we have observed that the L^1-Norm, L^2-Norm, and HOG Scaling produce similar results, whereas the Harris Corner Detector removes objects of interest and thus performs noticeably worse. See **Figure 2** for a visualization of the energy functions and **Figure 3** for a comparison of their carving abilities.

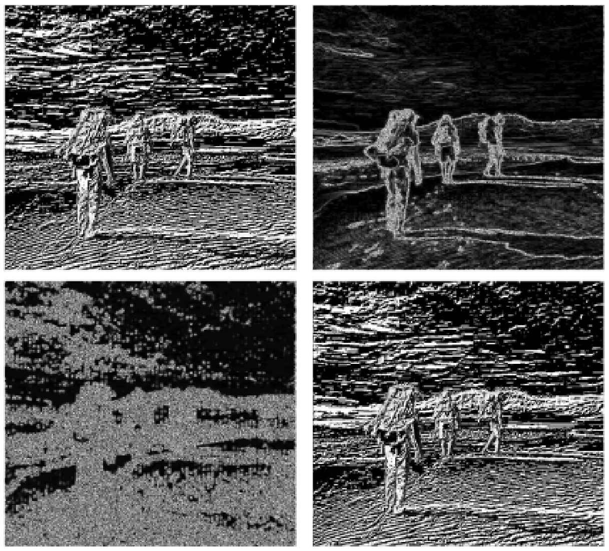


Figure 2: (From Left to Right, Top to Bottom) L^1-Norm, L^2-Norm, Harris Corner Detector, HOG Scaling

4 Histogram of Oriented Gradients (HOG) [2], [3]

The Histogram of Oriented Gradients algorithm was first introduced in a 2005 paper titled "Histograms of Oriented Gradients for Human Detection" by Navneet Dalal and Bill Triggs [2]. The paper used an MIT database comprised of photos purposed for human detection. Though the paper primarily applied this technique to humans, it has since become popular for object detection in general.

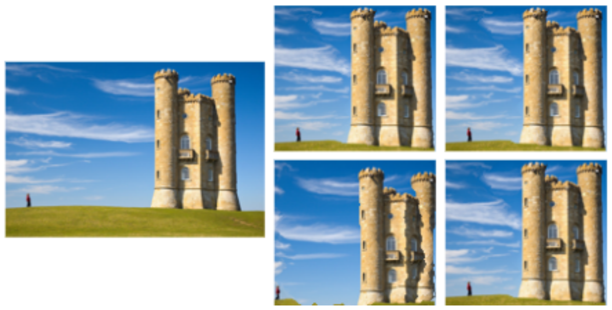


Figure 3: (Left) Original Image, (Right) Result of carving with various energy functions. Energy functions are in same order as in **Figure 2**

The Histogram of Oriented Gradients method can be separated into 4 steps:

1. Generate image gradient
2. Calculate histogram of gradients within $M \times N$ cells and X bins
3. Normalize across blocks of cells
4. Develop an optional feature vector

The gradient of the image is calculated using Sobel filters where,

$$I_x = \frac{\partial I}{\partial x} \text{ and } I_y = \frac{\partial I}{\partial y}$$

The image is then split into $M \times N$ cells. Within each cell, a histogram of gradients is calculated. Each gradient can be defined by its magnitude and direction,

$$\text{Magnitude} = \mu = \sqrt{I_x^2 + I_y^2}$$

$$\text{Angle} = \theta = |\tan^{-1}(I_y/I_x)|$$

The bins in the histogram are defined by

$$\text{Step size} = \Delta\theta = \frac{180°}{\#\text{ of bins}}$$

Note that unsigned gradients are used so the numerator of $\Delta\theta$ is 180° rather than the full 360°.

For each pixel, the magnitude of the gradient at that index is placed proportionally into the bins based on the respective angle of the gradient at the same index. As an example, take the scenario at an index where the magnitude is 85, the angle is 145°, and the closest histogram bins are 140° and 160°. The magnitude is proportionally split between the two bins, with 63.75 going into the 140° bin and 21.25 going into the 160° bin. In the code, the final histogram is stored as an X length array. We use an 11×11 cell with an 8 bin histogram.

Normalization of the histogram is then completed across a block of $H \times G$ cells to account for lighting variations since dark regions of an image will have smaller pixel values and thus smaller gradients. After normalization all histogram values will lie in $[0, 1]$. The code stores the resulting normalized vector as a $X \cdot H \cdot G$ array. We use a 1×1 block of cells, which amounts to just one cell.

Finally a feature vector can be calculated by flattening the array of block-normalized arrays into a single vector. This is primarily used as input into a classifier such as a support vector machine or neural network. For our purposes this is unnecessary.

The denominator of the energy function

$$\frac{\left|\frac{\partial I}{\partial x}\right| + \left|\frac{\partial I}{\partial y}\right|}{\max(\text{HOG}(I(x,y)))}$$

will be large when the gradients face the same direction and small otherwise. In general, the gradients will be oriented in the same direction near edges and will be oriented in differing directions in background areas and areas with unstructured texture. Thus, the denominator term in fact makes the energy function smaller near "interesting" regions. The authors of the initial seam carving paper provide the following intuition as to why rescaling the L^1-Norm with the max of HOG may aid in reducing seam carving artifacts, "taking the maximum of the [HOG] at the denominator attracts the seams to edges in the image, while the numerator makes sure that the seam will run parallel to the edge and will not cross it" [1]. In our limited testing, this rescaling had little effect on the result of the shrinking operation.

5 Seams

A seam is a connected path of pixels which runs across the entirety of one dimension of the input image. Two pixels are said to be connected if they share a corner or edge. Each seam contains exactly one pixel in each of the image's rows or columns depending on the seam's orientation. The orientation of the seam, that is whether it spans the width or length of the image, is determined by the selection to alter either the vertical or horizontal dimension of the image, respectively.

Associated with each seam is a cumulative energy value that is equal to the sum of energy values of all pixels comprising the seam. The seam with the smallest cumulative energy value is removed, the intuition being that high energy seams contain "interesting" parts of the image and thus would be more noticeable if removed. Seams are created using the following procedure: let C be the cumulative energy value of the seam, let E be the energy map, let M be the number of rows in the image, and let N be the number of columns in the image. Then beginning in the second row from the top we traverse through the rows updating C as follows,

$$C(x, y) = E(x, y) + \min\{C(x-1, y-1), C(x-1, y), C(x-1, y+1)\}$$

where x denotes the row index and y the column index. Note that if $y = 0$ or $y = N - 1$ then the minimum function will only have two arguments. In this way,

$$\min_i \{C(M-1, i)\}_{i=0}^{i=N-1}$$

corresponds to the ending point of the seam with the lowest cumulative energy among all seams. The pixels comprising this seam can be recovered by backtracking to the top row. See **Figure 4** for a visualization of seams.

It may not be entirely obvious why the pixels to be removed at each iteration of the algorithm must form a seam. One could imagine, for example, removing the pixels in each row with the lowest energy regardless of whether they are connected. Alternatively, the column with the lowest cumulative energy could be removed. Ultimately, seams are chosen because they simultaneously have the flexibility to accommodate "interesting" regions with organic shapes and the structure to reduce the number of artifacts. In algorithmic terms removing single disconnected pixels will preserve the most energy, but destroy the image content in the process. On the other hand, removing columns fails to preserve as much energy as seam removal while also producing noticeable artifacts in practice. See **Figure 5** for a concrete comparison.

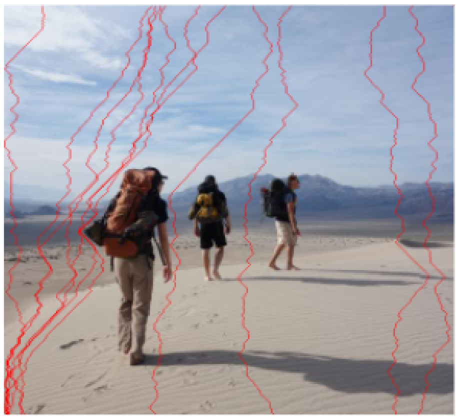
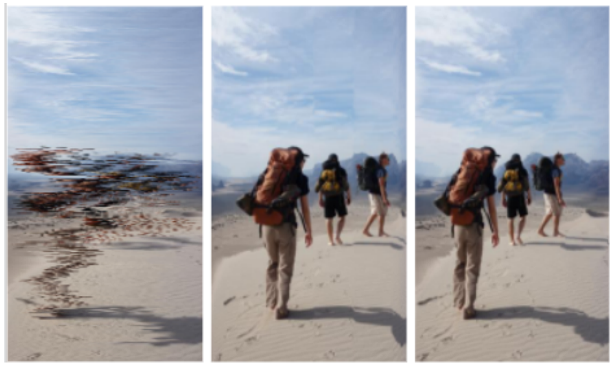


Figure 4: Visualization of 11 low-energy seams

6 Enlarging Algorithm

The idea of seam carving can also be applied to enlarging images. To add n seams of pixels to our horizontal dimension we find the first n seams for removal using the procedure previously described, then add pixels adjacent to these seams by copying the neighboring pixel values over.

The intuition is similar to that of the shrinking algorithm; adding pixels to less "interesting" regions of the image is likely to be less noticeable than say adding pixels in the middle of a person's face. One may reasonably wonder why it is required that we must first find n seams for removal. Why do we not simply add pixels along the seam with the lowest energy every time? Unfortunately, the same seam is often chosen repeatedly with this procedure. The same



Removal Method	Average Energy
Unconnected Pixels	12.15
Columns	7.20
Seams	7.66

Figure 5: From Left to Right: Removal of unconnected pixels, columns, seams

pixel values would be copied over and over again resulting in a smearing effect. See **Figure 6** for a comparison between the correct and naive implementations.

7 Seam Carving Shortcomings and Solutions

Seam carving does not perform well in four scenarios,

1. When the "interesting" regions span the the entire width (or height) of the image
2. When the "interesting" regions do not correspond to large values in the energy map
3. The image is too full of "interesting" regions so the energy map is uniformly large in value
4. When image resizing needs to be completed extremely quickly (i.e. in real time)

The second issue can be addressed with a different choice of energy function. For example, a face detector could be employed for portrait type images. Another solution would be to allow the user to indicate pixels which should not be removed. In algorithmic terms, this could be accomplished by setting the energy values of the indicated pixels to be $+\infty$ or some very large number.

Algorithm 2 Seam Carving for Image Enlarging

```
1: $j \leftarrow 0$
2: $n \leftarrow$ number of seams to add
3: $I \leftarrow$ input image
4: $seams \leftarrow [\,]$
5: while $j < n$ do:
6:     $E \leftarrow$ compute_energy_function($I$)
7:     $s \leftarrow$ find_minimum_seam($E$)
8:     $I \leftarrow$ remove_seam($I$, $s$)
9:     $seams[j] \leftarrow s$
10:    $j \leftarrow$ j+1
11: $Ia \leftarrow$ input image
12: while $k < n$ do:
13:    $s \leftarrow seams[k]$
14:    $Ia \leftarrow$ extend_along_seam($Ia$, $s$)
    return $Ia$
```

While the fourth issue can not be entirely solved because seam carving is inherently more computationally expensive than many alternatives, such as cropping. However, there are a number of programming tricks one can employ to reduce the run time. For example, dynamic programming can be employed to create the seams; this is the method described in **Section 5**. Furthermore, rather than copying over the pixel values to a new array each time a seam is removed, one could simply mark the pixels for removal so copying need only occur once. Depending on the energy function it may not be necessary to recompute the energy map over the entire image each time a seam is removed. For example, if the energy function only takes into account local pixel values such as the L^1-Norm described in **Section 3**, then only the energy values of pixels near where the seam was removed need to be recomputed.

8 Potential Improvements and Extensions

Forward Energy One improvement proposed by the original authors in a follow-up paper published

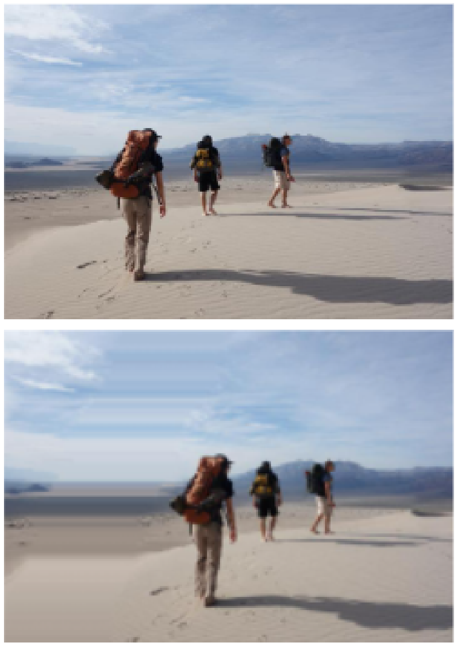


Figure 6: (Top) Correct Enlarging Algorithm, (Bottom) Naive Enlarging Algorithm

in 2008 is forward energy [4]. Many of the artifacts resulting from the standard seam carving algorithm occur because the choice of seam depends only on the current image's energy and does not take into account of the energy of the image resulting from the seam's removal. Whenever a seam is removed two pixels which were previously separated become adjacent. If the pixel values of these newly adjacent pixels differ wildly than lots of energy is inserted into the image after removing the seam. These new high energy regions often appear as artifacts in the resulting image. With this intuition, the authors developed forward energy which aims to minimize the following at every iteration,

$$\Delta E_{t=i+1} = e(I_{t=i+1}) - [e(I_{t=i}) - e(s_i)],$$

where E is the total energy, e is the energy function, I is the image, s is the seam to be removed, and the subscripts denote the iteration.

In terms of algorithm implementation the authors provide guidance beginning with **Figure 7** which illustrates the new edges created for each case of seam shape. They then go on to enumerate a cost function for each of the 3 cases which are shown below using the notation from the paper,

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

which take into account the difference in the pixel values across the new edges. The cumulative energy of a seam M at point (i,j) is then,

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

where P is an optional additional energy function.

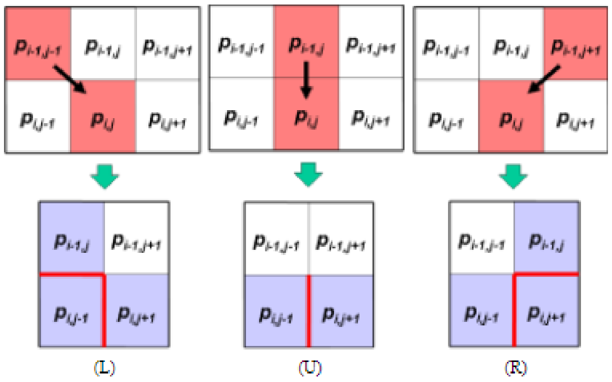


Figure 7: Fig. originally from [4]. Top row shows the pixels in the seam in red, Bottom row shows the new edges created after the seam is removed in red.

Seam Carving for Video In the same follow-up paper the authors also use the new forward energy criterion to carve videos. A video can be viewed as a 3D rectangular prism with two spatial dimensions and a time dimension. The seams are then 2D surfaces which span the time dimension and a single spatial dimension. An illustration of a 2D seam is shown in **Figure 8**. Notably, the authors abandon the dynamic programming used for images and instead formulate the problem in terms of finding the min-cut in a directed graph.

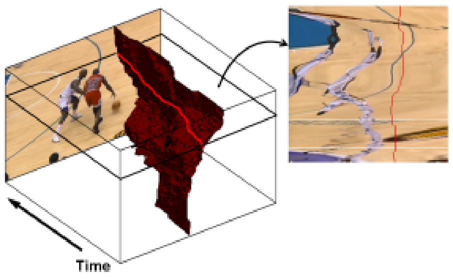


Figure 8: Fig. originally from [4]. Illustration of a 2D seam in the rectangular prism representation of a video.

9 References

[1] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," in *ACM SIGGRAPH 2007 papers*, pp. 10–es, 2007.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, Ieee, 2005.

[3] S. Mallick, "Histogram of oriented gradients explained using opencv," Nov 2021.

[4] M. Rubinstein, A. Shamir, and S. Avidan, "Improved seam carving for video retargeting," *ACM transactions on graphics (TOG)*, vol. 27, no. 3, pp. 1–9, 2008.

10 Note

All functions and algorithms described in this paper, save for two exceptions, were implemented from scratch in Python without consulting any existing code repositories. The exceptions are that we used the scikit-image implementation of the Histogram of Oriented Gradients and we did not implment the techniques described in **Section 8**. See the attached pages for our code.