# *Content-Aware Image Resizing*

EE368 Project Report

Parnian Zargham
Stanford University
Electrical Engineering Department
Stanford, CA
pzargham@stanford.edu

Sahar Nassirpour
Stanford University
Electrical Engineering Department
Stanford, CA
saharnp@stanford.edu

***Abstract*— The purpose of this project is to implement a content-aware image resizing method. The algorithm is optimized and then applied to a database of 100 images. The results are evaluated and different features are added to the method. Finally, a graphical user interface is devised.**

***Keywords: image resizing, Seam, Seam Carving***

## I. INTRODUCTION

As today's technology-driven world advances, creating adaptive processing tools has become of more and more interest. One of the applications that has brought much attention to itself recently is the use of intelligent image resizing techniques in order to fit the images into the variety of display tools that exist such as smartphones, monitors, different page layouts, etc. Conventional image resizing techniques consist of cropping or evenly downsampling the image without taking into account the content of the image. These methods can lead to loss of important image features or distortion. A better method would remove pixels from uninteresting parts of the image while preserving the important content.

An elegant and efficient method that has been proposed recently is called "Seam Carving" [1]. Basically this method uses a kind of metric to determine the importance of each pixel of the image. Different metrics include gradient magnitude, entropy or visual saliency [1]. By using this importance map, the algorithm finds the optimum seam, which is defined as the path of pixels with the lowest cumulative energy, and removes it from the image. This way, because the interesting features in the image usually have more energy, the method automatically preserves the important contents and removes pixels from the uninteresting parts.

There are a number of other interesting things that can be done using this method. For example, the seam carving method can be run reversely to insert interpolated seams along the optimum seams to enlarge the image. Alternatively, by manually assigning large negative or positive energy values to certain pixels in a region of interest, the algorithm can be manipulated to remove or preserve the region.

In this project, after exploring the existing literature on this algorithm, we have implemented an interactive graphical user interface, which enables the user to run this algorithm on the image, test the implemented features and compare different methods for image resizing. We have used this user interface to carefully study the effect of algorithm on different types of images and identify on which images it works the best.

## II. METHODOLOGY

As mentioned before, the main idea behind seam carving algorithm is to remove unnoticeable pixels while preserving the significant content of the image. The pixel importance can be measured by some metric such as gradient operator, entropy operator, etc. Hence, we have to generate an energy map and cumulative energy matrix of the input image then find the optimal seam which is defined as a path of pixels having the minimum cumulative energy and finally remove the seam from the image and repeat these steps until the image is of desired size. In this section we describe each step in details.

### *A. Generating the energy map*

There are a number of ways to extract the unnoticeable pixels from an image. First and most straightforward is to assign energy to each pixel by using a gradient operator (e.g. Sobel) to compute the gradients in both $x$ and $y$ directions, The energy function is defined for each pixel as follows [1]:

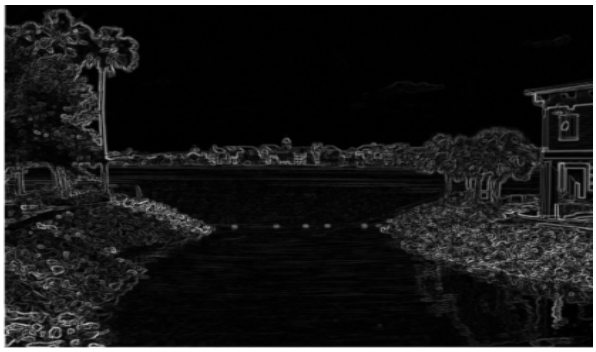
$$e(x,y) = \left|\frac{\partial I}{\partial x}\right| + \left|\frac{\partial I}{\partial y}\right| \quad (1)$$

This is a metric for measuring the variation of the image in both directions and then assigning a number to each pixel.

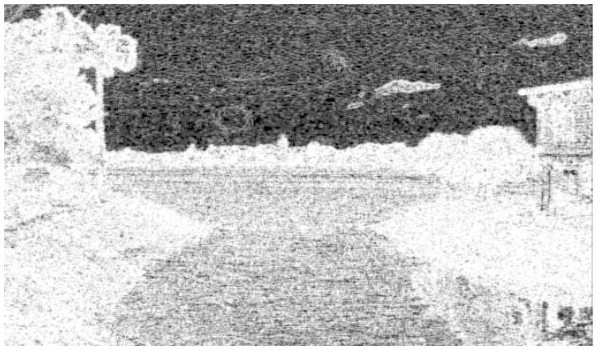
Instead of energy function, a local entropy filter can be applied on image, where each output pixel would be the entropy value of the 3-by-3 neighborhood around the corresponding pixel in the original image [8]. This metric measures the information that each pixel contains. Figure 1 shows an example of an image along with its energy map (figure 1b) and its local entropy map (figure 1c) .

Later, the two methods are compared in the experimental results section.

(a)



(b)



(c)

Figure1-(a) original image, (b) gradient energy map, (c) entropy map.

*B. Optimal seam detection*

A vertical seam for a $n \times m$ image is defined as [1]:

$$\boldsymbol{S^x} = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, where\ \forall i, |x(i) - x(i-1)| \leq 1 \quad (2)$$

Likewise a horizontal seam is defined as:

$$\boldsymbol{S^y} = \left\{s_j^y\right\}_{j=1}^m = \{(y(j), j)\}_{j=1}^m, where\ \forall j, |y(j) - y(j-1)| \leq 1 \quad (3)$$

In other words, a vertical seam is a path from top to bottom (left to right for horizontal) containing only one pixel from each row.

The cost $C$ of the seam $s$ is defined as the sum of the energies of the pixels along the seam path [1],

$$C(s) = \sum_{i=1}^n e(I(s)) \quad (4)$$

An optimal seam would have the minimum seam cost. In order to find this seam, we first define the cumulative minimum energy map $M$ for the second row to last row as the following [1]:

$$M(i,j) = e(i,j) + \min\left(M(i-1,j-1), M(i-1,j), M(i-1,j+1)\right) \quad (5)$$

Then the optimal seam would be found by taking the minimum pixel value in the last row of $M$ which would be the end of the optimal seam path. We then track the optimal seam path by going upwards in matrix $M$ and finding the minimum value among the three adjacent pixels right above the first one.

The above calculations were all described for vertical seams. For horizontal seams all the implementation can simply be done on the transposed version of the image. The cumulative energy map for the same image of figure 1 along with the first optimal seam is shown in figure 2.

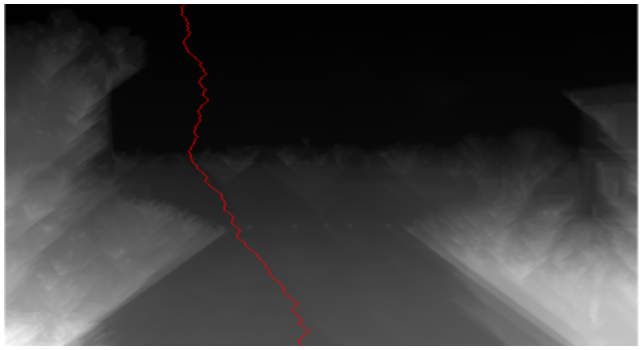


Figure2- cumulative energy matrix of figure1, with the first optimal seam superimposed

## III. IMAGE RESIZING

*A.Image shrinking*

In many applications we want to change the aspect ratio of an image or resize it to fit in a smaller display. This can be accomplished by finding the optimal seams in horizontal and vertical directions and removing them from the image. One way is to first remove all the horizontal optimal seams and then remove all the vertical ones. But a more efficient way is to remove the seams in an optimal order, where we compare the cost energy of the horizontal optimal seam to the vertical optimal seam and first remove the one with the minimum cost

energy. By repeating this process and recalculating the new energy map in each iteration, the image can be converted to the desired size.

After comparing the optimal order seam removal with the simple way of removing first horizontal seams and then vertical seams over a sample of 50 test images, we did not see considerable improvement for the efficient method over the simple method. So, in order to avoid the speed cost, we chose the first method.

An example of image shrinking is shown in figure 3. For comparison purposes, we have also shown the result of imresize() function of MATLAB on the same image in figure3c.

(a) (b)

(c)

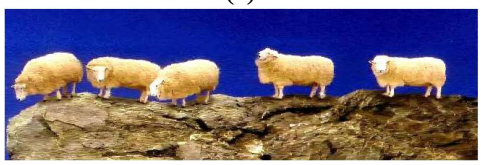
Figure3-An example of image shrinking, (a) the original image, (b) the resized version, size reduced by %30 in horizontal direction, (c) same resizing done using MATLAB imresize()

### *A. Image enlarging*

In order to enlarge the image, we can add pixels to regions of image that do not have a lot of variation. Again, we can use the optimal seams to identify these regions, and insert interpolated seams by averaging the two adjacent seams. This way we can ensure that there is a smooth transition between the added pixels. To avoid picking the same optimal seam over and over again, each time we detect an optimal seam we assign a large positive value to the corresponding pixels in the energy map.

An example of image enlarging is shown in figure 4.

(a)

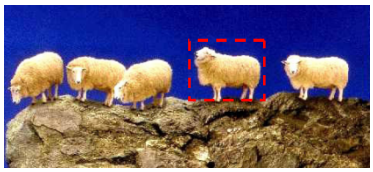


(b)

Figure4-an example of image enlarging, (a) the original image, (b) the enlarged version, size increased by %30 in horizontal direction.

## IV. OBJECT REMOVAL

Another application of seam carving is object removal where you can select an undesirable object from the image, and use seam carving to remove it smoothly. Here, we manually assign large negative energy to the region of interest to make sure that the optimal seams pass through the region. However, since we are removing several seams from the same region this may cause distortion. To solve this problem, Seams can be inserted where the object used to be. An example of object removal is shown in figure 5.



(a)

(b)

Figure 5- an example of object removal, (a) the original image, (b) the dashed sheep removed

## V. OBJECT PRESERVATION

Sometimes there are features in an image that we need to preserve, because the human eye is very sensitive to the symmetry and proportion of the object. A clear-cut example of this would be a human face or body in an image where symmetry plays a key role, but by passing through the seam carving algorithm even removing a few seams can distort the symmetry. In this case, we can manually select the region of interest and preserve it by manually assigning large positive

energy values to the corresponding pixels and prevent the optimal seams from passing through that region.

An example of object preservation is shown in figure 6.

Figure 6- An example of object preservation, (a) the original image, (b) size reduced by %20 horizontally, the human figure is distorted, (c) same size reduction, but the human figure is preserved

## VI. GRAPHICAL USER INTERFACE

A GUI was implemented to provide a better interface for visualizing different features and comparing different methods. This was created in MATLAB's GUI environment.

The GUI has several features including:

*1)* Importing an image to the GUI environment
*2)* Aspect ratio changing (both enlarging and shrinking) with the option of showing the seams superimposed on the image while removing them.
*3)* Selecting a region of interest by dragging the mouse cursor over the region.
*4)* Object removal.
*5)* Object Preservation
*6)* Pan and zoom in the image.
*7)* Selecting different energy functions for comparizon

Figure7 shows a screenshot of the GUI environment.

## VII. EXPERIMENTAL RESULTS

### A. *Testing the algorithm on different images*

In order to carefully study, the algorithm was tested on a database of 100 images, classified in 10 different categories: natural scenes, sports, Art, buildings, Objects, animal, shapes and patterns, texture, human face and medical images .

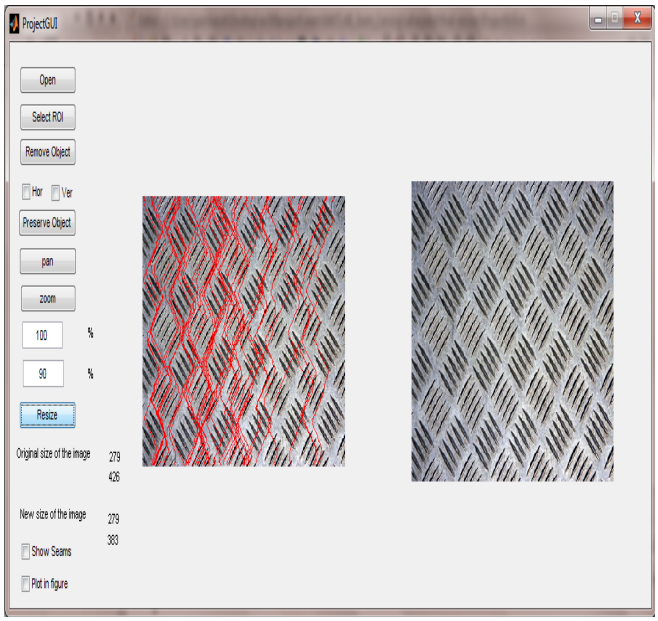



Figure 7- a screenshot of the GUI

The original Images are all $600 \times 800$ pixels and all resized %60 vertically and %80 horizontally .A sample of each category is shown in figure 8.

It can be observed that the algorithm works best on images that have a lot of low energy seams such as natural scenes. One way to automatically detect if the algorithm would work on a given image would be to count the number of seams which have energies below a certain threshold. As expected, the algorithm does not work well when applied to images of shapes. Unless they are preserved, the shapes look much distorted in the resulting image. Also the algorithm does not properly work on images where symmetry is important such as images including human faces or bodies. The algorithm seems to work well on texture images.

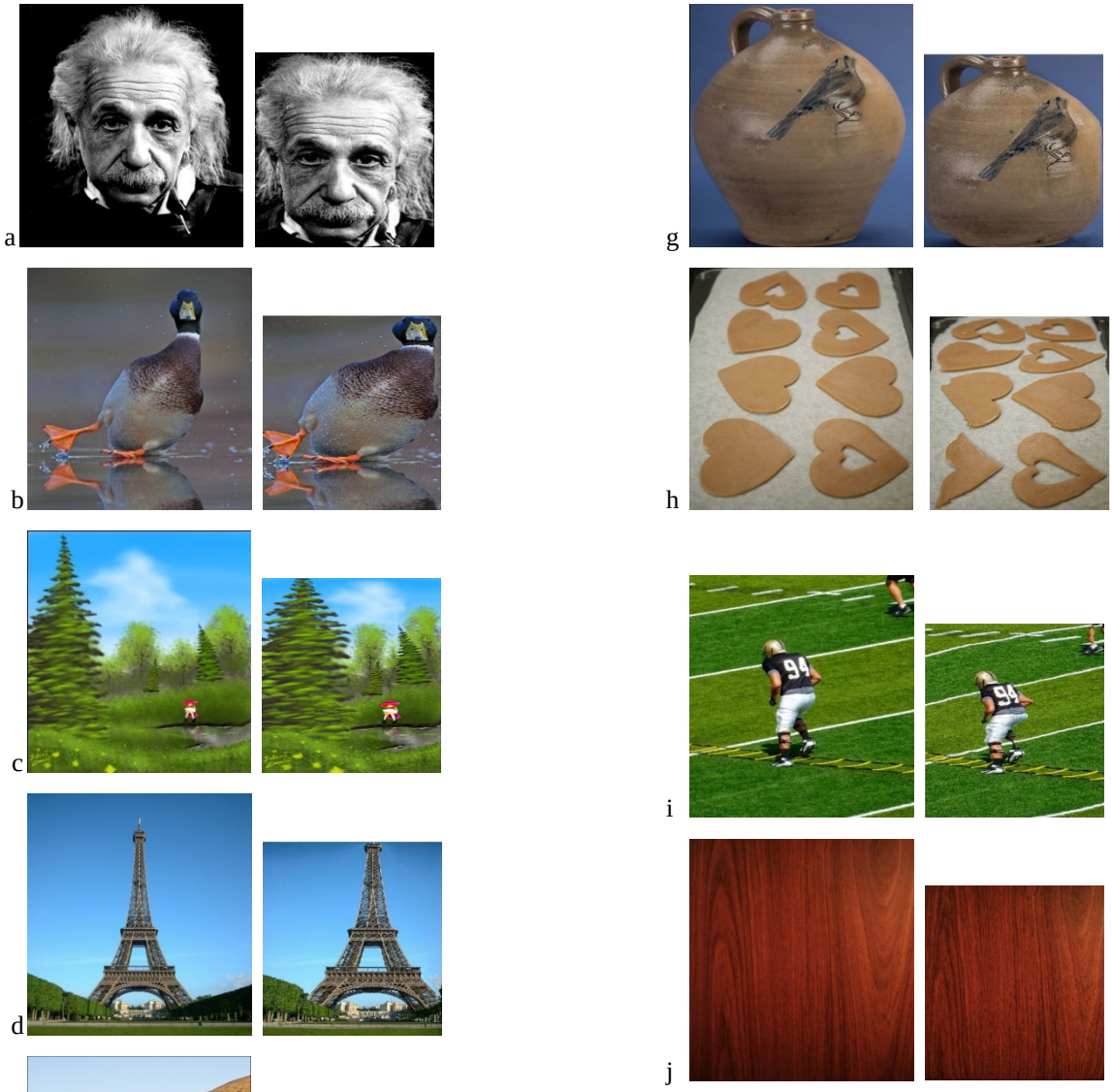


Figure 8- Original and resized samples of 10 categories, (a) human face,(b) animal,(c)Art,(d)building,(e) natural scenes,(f) medical images,(g)Objects,(h) shapes and patterns,(i)sport,(j)texture.
]

*B. Testing the algorithm for different resolutions*

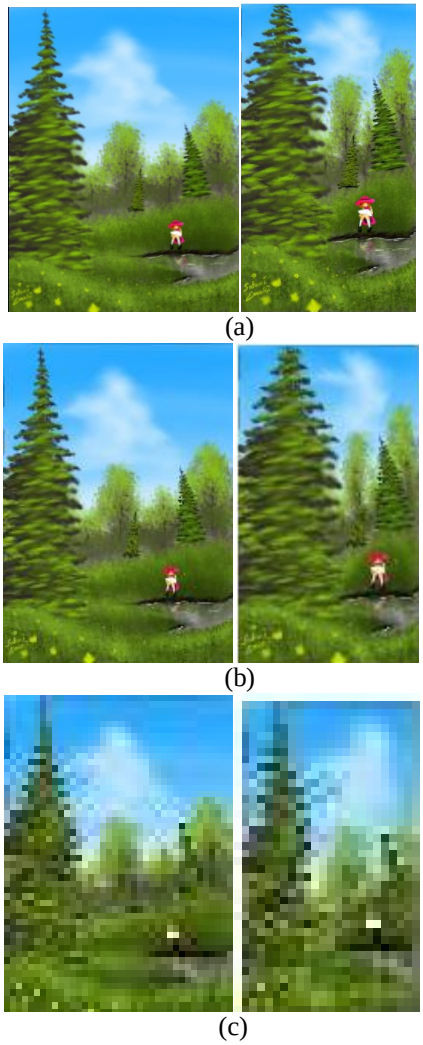
In order to study the effect of applying the algorithm on different resolutions, we did an experiment in which we took 10 images from our database, originally of size $600 \times 800$, and downsampled them by a factor of 4 and then again by a factor of 4. Then we applied seam carving on the three resulting images. Figure 9 shows the results.

By comparing the result with original image in each case, we can see the result of lower resolution version of the image looks better to human eye and abrupt changes between the regions that the seams are removed from and those that remain intact are more noticeable in high resolution images compared to lower resolution ones.

### C. *Gradient operator VS. entropy*

We have applied the entropy and gradient energy operators on different images. From our observation, sometimes the entropy operator works better as shown in figure 10, and sometimes the other way around.

We think that by devising a hybrid algorithm that would use both metrics to assign scores to pixels we can improve the algorithm.

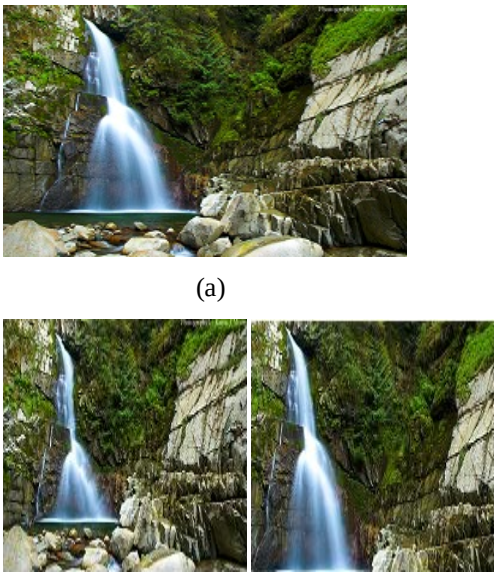


(a)

(b)

(c)

Figure 9- Applying the algorithm to different resolutions, (a) the original image along with the size reduced one, (b) the downsampled image by a factor of 4 along with the size reduced one, (c) the downsampled image by a factor of 16 along with the size reduced one



(a)

(b) (c)

Figure10-An example for comparing energy operator with entropy operator, (a) the original image, (b) the resized version, size reduced by %40 in horizontal direction using energy operator, (c) the resized version, size reduced by %40 in horizontal direction using entropy operator.

### D. *Seam carving VS. MATLAB imresize function*

We have seen an example of aspect ratio changing using MATLAB imresize() function compared to Seam Carving in figure 3. We can see that MATLAB evenly downsamples all the features but the Seam carving method intelligently removes pixels while preserving important features.

### E. *Run time*

When we first implemented the algorithm in MATLAB, it was working properly, but the run time was longer than desired. By looking into the profiler viewer of MATLAB, we found the bottleneck, which was the seam removal (SeamCut.m) and minimum cumulative energy matrix generating function (findSeams.m). This was firstly due to multiple nested for loops which were optimized by vectorizing the code and secondly due to multiple min function calls which was resolved by assembling separate vectors in a matrix and calling the min operator once on all the rows.

After optimization run time was reduced by a factor of 20.

## VIII. CONCLUSIONS

In this project, we have explored and evaluated the seam carving algorithm. While the original method only allows for content-aware image resizing, it can be used to develop different interesting features including object removal or preservation. We have concluded that the algorithm is

performing properly on landscapes and in general images with a lot of flat regions.

In summary, the algorithm when paired with the object preservation feature can be used on display systems to allow for content-aware image resizing instead of evenly downsampling or croping.

## IX. Appendix

Parnian:

- Writing MATLAB functions SeamCut.m , SeamInsert.m , CalculateEnergy.m and entropy calculation.
- Adding interpolation and image enlarging features
- Literature review
- GUI design and coding
- Poster preparation
- Report preparation
- Code speed up
- Running experiment on 50 images of database

Sahar:

- Writing MATLAB functions findSeams.m and OptSeam.m and embedding the region selection feature.
- Adding object removal and object preservation features
- Literature review
- GUI design and coding
- Poster preparation
- Report preparation
- Code speed up
- Running experiment on 50 images of database

## X. References


[1] S. Avidan, A. Shamir, "Seam carving for content-aware image resizing" ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH, Volume 26 Issue 3, July 2007.

[2] W. Dong, J. Pauls, "Adaptive content-aware image resizing", Eurographics, vol. 28. No. 2, 2009.

[3] K. Thilagam, S, Karthikeyan, "An analysis of hybrid techniqes of seam carving," IJCSE, vol. 3, No.4, April 2011.

[4] C. Fillion, G. Sharma, "Detecting content adaptive scaling of images for forensic applications", Proceeding of SPIE-IS&T, Vol 7541,.

[5] D. Conger, M. Kumar, L. Miller, J. Luo, H. Radha, "Improved Seam Carving for Image resizing," IEEE Workshop on ISPS, San Fransisco, 2010.

[6] R Achanta, S. Susstrunk, "Saliency Detection for content-aware image resizing", 16th IEEE conference on image processing, Cairo, 2009.

[7] R Marin,"Seam carving implementation", 2007.

[8] http://www.mathworks.com/help

*Photos:*

[Figure1] Webshot by Jane Chen

[Figure2 through 9] From google images

[Figure10] Webshot by Jane Chen