Seam Carving and Seam Carving Detection
Group 16

Ashish Sharma	Chinmaya Pancholi	Ken Kumar
13CS30043	13CS30010	13CS30044

1 Objective

In this project, we aim to perform object removal using seam carving. We also aim to detect if an image has tampered using seam carving without using the original image.

2 Seam Carving

Seam Carving [1] is a technique targeting image compression and content-aware image resizing based on detection of seams from the energy function of the image. The method aims at finding seams(threads) of minimum energy and manipulating the image using them.

2.1 Seams

Seams can be either vertical or horizontal. A vertical seam is a path of 8 connected pixels from top to bottom in an image with one pixel in each row. It is formally defined as follows:

$$s^v = \{(i, col(i)), \quad s.t. \forall i, |col(i) - col(i-1)| \leq 1\}$$

where i and col(i) denote row coordinates and the corresponding column coordinates of an image I, respectively. A horizontal seam is similar with the exception of the connection being from left to right. The importance/energy function values a pixel by measuring its contrast with its neighbor pixels.

2.2 Application of Seam Carving

Effective resizing of images should not only use geometric constraints, but consider the image content as well. Conventional image resizing consists of cropping or evenly down sampling that lead to loss of important features or distortion. This method enables us to remove pixel from uninteresting parts of the image while preserving important content. Scaling does not

always preserve the aspect ratio (height to width ratio) of an image so the content of an image must be modified to fit the new dimensions. Naive resizing techniques include stretching and cropping. Neither of these options are ideal as stretching distorts the image and cropping reduces the image content. "Seam carving" provides a generally superior method. At a high level, seam carving is an algorithm that preserves the sizes and shapes of "important" objects, while resizing "less important" parts if the image.

Hence, seam carving can be used for distortion free image expansion by inserting least energy seams in the image. This can also be extended to object removal and object protection in an image.

2.3 Design and Implementation

Reducing the size of an image is accomplished by removing pixels that will go unnoticed. The pixels to be removed are determined by finding the path across the image with the lowest total energy, which is the sum of the energy of each pixel along the path. Consider making the image shown below one pixel narrower. One pixel in each row of the image is removed, reducing the width of the image by one pixel. The red line near the right edge shows the minimum energy path from the top to the bottom of the image. Each pixel in the path is removed from the image for an overall reduction in width. To further reduce the width of the image, the pixels from the next lowest energy path will be removed. This process can continue until the desired width is reached.

There are various methods for generating the image energy map.

1. **Gradient Magnitude:** There are various methods to extract the unnoticeable pixel from an image. First and most basic method is to assign energy to each pixel by using a gradient operator.(Sobel, Prewiit, Robert or Laplacian) to compute the gradient in both x and Y direction.

2. **Entropy:** The basic gradient function doesn't give defined outputs. A local entropy filter can be applied on the image to improve the basic gradient energy map.

3. **History of Gradients:** Another method to compute the gradient energy map is Histogram of Gradients (Hog) method. $Hog(I(x, y))$ is taken to be a histogram of oriented gradients at every pixel.

We next describe the algorithm we have used for seam removal, insertion and object removal.

2.3.1 Seam Removal

For removing vertical seam, we follow the following steps:

1. **Calculate energy map:** As the first step, we calculate energy by summing the absolute value of the gradient in both x direction and y direction for all three channel (B, G, R). Energy of each pixel in the image I is measured as:

$$e(I) = |\frac{\partial I}{\partial x}| + |\frac{\partial I}{\partial y}|$$

From the energy of a pixel, energy of a vertical seam $E(s)$ is defined as follows:

$$E(s) = \sum_{i=1}^{n} e(i, col(i)), \quad s.t. \forall i, |col(i) - col(i-1)| \leq 1$$

2. **Build cost matrix:** Next, we need to find vertical seam with minimum energy path. In order to do that, we first build an accumulated cost matrix M using dynamic programming. The value of each pixel (i, j) is equal to its corresponding value in the energy map added to the minimum new neighbor energy introduced by removing one of its three top neighbors (top-left, top-center, and top-right). That is,

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

3. **Find vertical seam with minimum energy path:** Using the cost matrix M, we Backtrack from the bottom edge to the top edge in order to find seam with minimum energy path. This is the seam to be removed. All the pixels in each row after the pixel to be removed are shifted over one column to the left if it has index greater than the minimum seam.

4. Step 1 to 3 are repeated until targeting width is achieved.

For removing horizontal seam, image is rotated 90 degree counter clockwise and same steps are repeated.

2.3.2 Seam Insertion

For seam insertion, we follow the following steps:

1. Find the cumulative energy map of the image using steps 1 and 2 of seam removal.

2. Find the minimum energy path (optimal seam) in the desired direction, i.e. top to bottom or left to right.

3. Copy the optimal seam and extend the image at the point of the optimal seam.

4. Step 1 to 3 are repeated until targeting width is achieved.

2.3.3 Object Removal

For removing an object from an image using seam carving, we follow the following steps. The pixels of the object to be removed are provided by user:

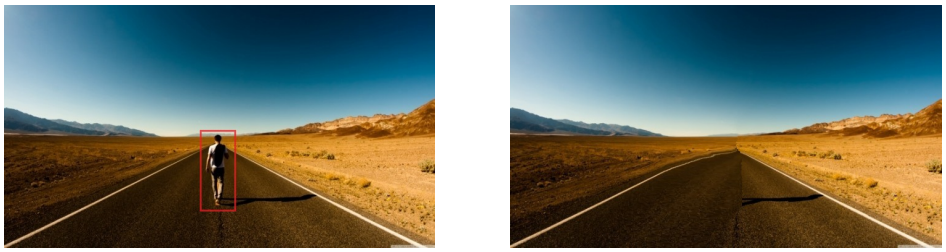
1. Find the cumulative energy map of the image using steps 1 and 2 of seam removal.
2. Provide the minimum energy values to M(i,j) if pixel (i,j) is selected for removal by the user.
3. Find the minimum energy path (optimal seam) in the desired direction, i.e. top to bottom or left to right. Copy the optimal seam and extend the image at the point of the optimal seam.
4. Steps 1 to 3 are repeated until all the pixels specified by the user have been removed.
5. Finally, seam insertion is performed on the output, so as to match the input dimensions.

2.4 Test Data and Results

In this project, our major aim is to remove objects from images using seam carving. Here, we test the algorithm described above on some test images and present the results in Figure 1,2 and 3. The object to be removed has been highlighted by a red square.

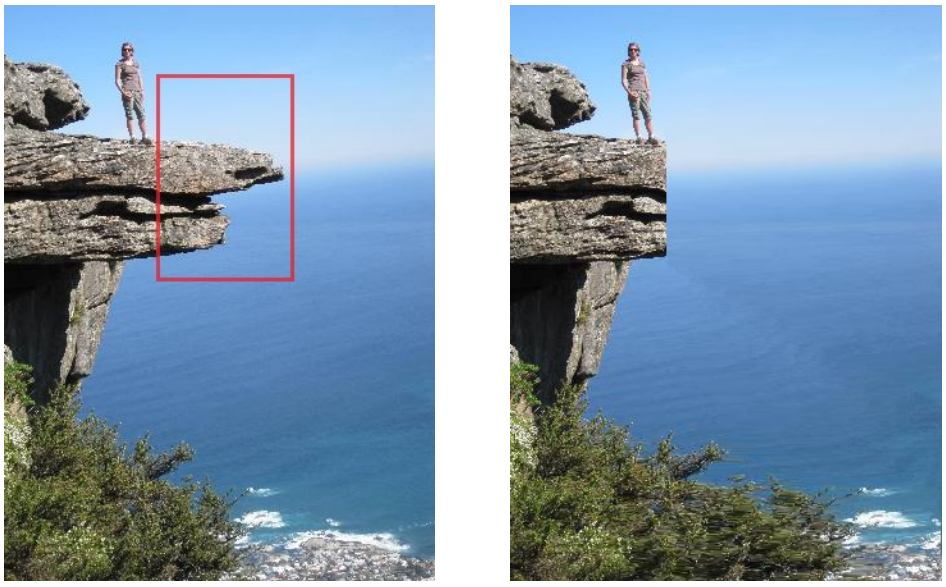
(a) Input Image (b) Output Image

Figure 1: Test Image 1



(a) Input Image (b) Output Image

Figure 2: Test Image 1



(a) Input Image (b) Output Image

Figure 3: Test Image 3

As we can clearly see, the algorithm has been successfully able to remove the objects. Here, we had specified the objects to be removed as square blocks. More precision can be obtained by providing exact set of pixel values corresponding to an object.

3 Seam Carving Detection

As we saw in the previous sections, Seam Carving allows us to remove objects from an image without degrading image quality due to the optimal seam choices. An object in an image generally carries a semantic meaning with itself and removing it can potentially alter the semantic content of the image.

This can lead to malicious acts of *image tampering* and *forgery* and thus, there is a need to design algorithms for detecting whether an image has undergone seam carving. In this section, we design an algorithm based on discrete cosine transform (DCT) for detecting image tampering caused by seam carving. Please note that we are mainly interested in detecting cases where a complete object has been removed from the image as those are the cases where the meaning of an image gets altered. Since seam carving is designed for *content-aware image resizing*, other cases are less likely to be malicious and are thus of lesser interest.

3.1 Challenges Involved

The modifications to an image caused by seam carving are difficult to identify due to the following reasons:

1. The insertions and deletions caused by seam carving spans the entire image rather than a part or a portion of an image.
2. As opposed to other image tampering techniques like image-inpainting, seam removal and insertion does not necessarily bring visually annoying artifacts such as shadow or blurriness in an image.
3. In most of the cases, seams carving only changes the local texture and the energy distribution of an image.

Therefore, the inherent nature of the seam carving process should be taken care while designing tampering detection algorithm.

3.2 Design of our Algorithm

In this project, we utilize discrete cosine transform (DCT) for detecting image tampering. We first divide the image into fixed-size blocks. We then apply DCT on each of those blocks. In order to reduce the dimension, 4 features are extracted from each block which are finally utilized for finding tampered portions of the image. The detailed steps of our algorithm are as follows:

1. **Dividing into blocks:** Given an image of size MxN, we first divide it into overlapping blocks of size BxB. The blocks adjacent to each other differ only by either one row or one column. Thus, the total number of blocks, $N_{blocks} = (M - B + 1) * (N - B + 1)$.
2. **Discrete Cosine Transform:** DCT is applied on each block B_{ij} (where i,j denote the starting indices of the block). After applying DCT, each block is represented as a matrix of DCT coefficients of the same size as of the block.

3. **Extracting Features:** The energy of DCT coefficients focuses largely on low frequency components. Thus, all the B^2 elements of the DCT coefficient matrix are not important and our focus should be on the ones with low frequency. We extract the low frequency DCT coefficients in a zigzag order. We use a circle block to represent the coefficients matrix and divide it into four parts - $C_1, C_2, C_3, andC_4$ in order to represent the whole image as a four part energy. This helps in reducing the computational complexity without sacrificing a lot on efficiency. For $C_1, C_2, C_3, andC_4$, we obtain the corresponding features $f_1, f_2, f_3 and f_4$ as follows:

$$f_i = \frac{\sum DCT(x, y)}{area(C_i)}$$

for i = 1,2,3 and 4, where $(x, y) \in C_i$. Thus, for each BxB block, we obtain a 4-dimensional feature vector $f = \{f_1, f_2, f_3, f_3\}$.

4. **Detecting Tampering:** After representing each block as a feature vector, we next detect if the image has been tampered or not. Under the assumption that object was significantly large in size, we aim to figure out if an object has been removed or not. Since seam insertions and deletions span the whole image, it is difficult to detect the exact portion in the image where the object was previously located just using DCT coefficients. Thus, we restrict ourselves to the problem of identifying image tampering on the whole.

   In order to achieve this, we make use of 8-neighborhood of each block . We compute the average feature vectors of the blocks in the 8-neighborhood of (x, y) and check if it is similar to the feature vector of (x, y) block. That is,

$$\sqrt{\sum_{i=1}^{4}(f_i^{x,y} - f_i^{nbr(x,y)\_avg})^2} \leq SIM\_TH$$

   We check this for each block B(x,y) and find the set of candidate blocks in which pixels have potentially been inserted using seam carving. Let A_{cand} denote the set of candidate blocks identified from the above step. We call that an image has been tampered using seam carving if

$$\frac{|A_{cand}|}{N_{blocks}} \geq CAND\_TH$$

   where $|A_{cand}|$ denotes cardinality of set A_{cand} and $N_{blocks} = (M - B + 1) * (N - B + 1)$ for an image of size MxN and Block size B.

$CAND\_TH$ is a threshold chosen based on expected size of objects which might have been removed. In our experiments, we choose $B = 8$, $SIM\_TH = 0.005$ and $CAND\_TH = 0.3$. Please not that all feature vectors are unit normalized.

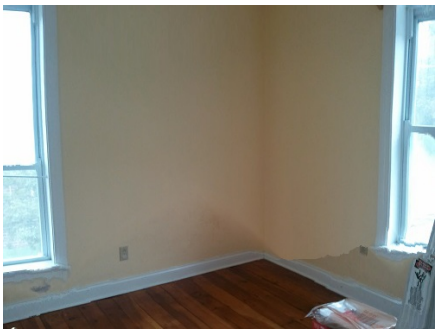
3.3 Implementation

We use python for implementing the above stated algorithm. *open-cv* and *numpy* are the 2 major library used. open-cv allows us to read and write images using *cv2.imread()* and *cv2.imwrite()* functions. It also helps in computing DCT coefficients of different blocks with it's function *cv2.dct()*. We use numpy for dealing with feature vectors and inequalities described in our algorithm. Please refer *tampering-detection.py* for the code.

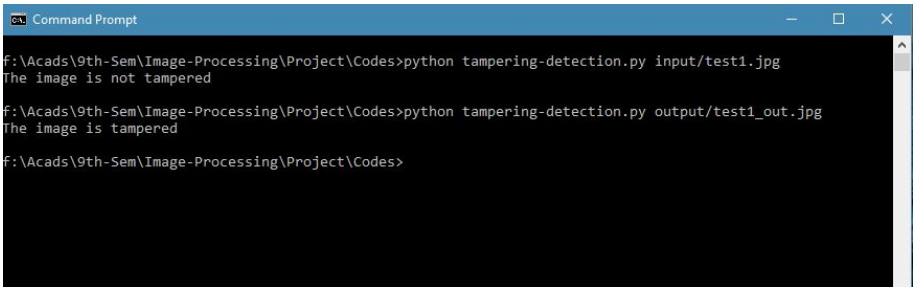
3.4 Test Data and Results

In order to test this algorithm, we give the original as well as object removed image as input to the algorithm. The algorithm should ideally output "not tampered" for the original image and "tampered" for the seam-carved image. We check this on a significant number of images, and consistently observe the same with a few failures mainly due to perfection of seam carving algorithm. Here, we show the output for one of the images in Figure 4.

(a) Input Image



(b) Output Image



(c) Detection Algorithm Output

Figure 4: Tampering Detection

4 Discussion

In this project, we worked on implementing a content-aware resizing approach called *Seam Carving*. We used it for removing objects from an image and thus, saw how can it be easily used in image tampering and forgery. In order to counter it's malicious use case, we designed an algorithm for detecting whether an object has been detected using analysis of DCT coefficients.

As a next step, it would be interesting to design algorithms for finding the locations where the removed object was located in the original image and use this information for reconstructing the image. Incorporating machine learning techniques and advanced computer vision approaches might be the way ahead.

There are several works which try to achieve this. Sarkar et al [3] extract markov features in the quantized DCT domain and then use SVM for detecting seam carving and seam insertions. Wei et al [4] divide the image into small patches called mini-squares, and then searched for one of nine types of patches that is likely to recover a mini-square from seam carving. [2] created a forensic hash of the original image and then check for seam insertions and seam deletions. Most of these approaches require training on large datasets. However, work on detecting seam carving rather than

object deletion using seam carving for which generating the dataset is difficult. Models focusing on the more interesting problem of detecting object deletion using seam carving and using supervision of large dataset forms an interesting future work.

References

[1] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *ACM Transactions on graphics (TOG)*, volume 26, page 10. ACM, 2007.

[2] Wenjun Lu and Min Wu. Seam carving estimation using forensic hash. In *Proceedings of the thirteenth ACM multimedia workshop on Multimedia and security*, pages 9–14. ACM, 2011.

[3] Anindya Sarkar, Lakshmanan Nataraj, and Bangalore S Manjunath. Detection of seam carving and localization of seam insertions in digital images. In *Proceedings of the 11th ACM workshop on Multimedia and security*, pages 107–116. ACM, 2009.

[4] Jyh-Da Wei, Yu-Ju Lin, and Yi-Jing Wu. A patch analysis method to detect seam carved images. *Pattern Recognition Letters*, 36:100–106, 2014.