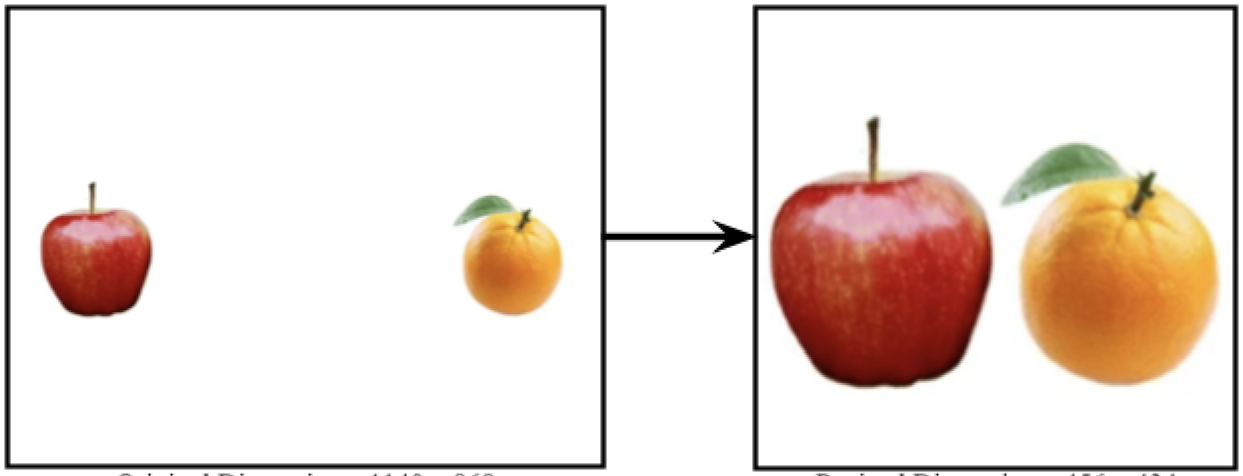
Trisha Menon (tmenon3)

Prashant K Naganaboyina (pkn3)

CS 445

Final Project - Seam Carving



Original Dimensions: 1140 x 868 Resized Dimensions: 456 x 434

Motivation and Impact:

As we continue into a digital world, sharing images across so many different mediums becomes much easier. A creative aspect of that is resizing to create collages or convert horizontal images for vertical mediums, like various social media platforms. Although cropping is an easy way to quickly change the dimensions of an image to meet your needs, it can be difficult if meaningful parts of your image lie along the borders or edges. To solve this problem, we wanted to implement a method to resize images using the seam-carving algorithm to identify and discard trivial "seams" along images without having to sacrifice important content and details. This "content-aware" technique allowed us to learn more about the versatility of graphs and dynamic programming, as well as apply our knowledge of energy functions.

Approach:

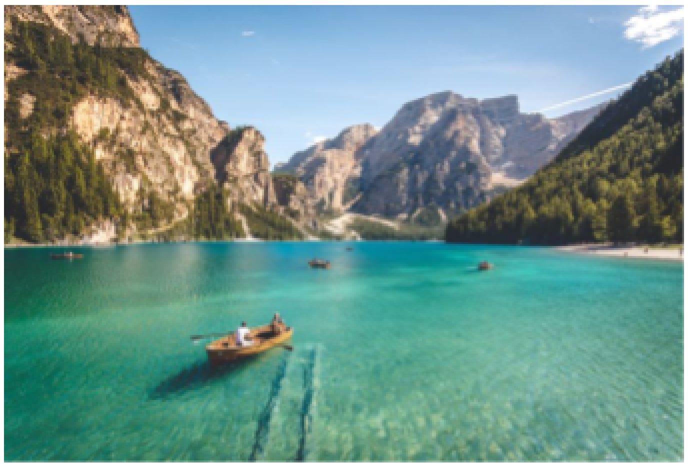
Our approach to implementing this algorithm begins with the calculation of the energy map for the image that we wish to remove a seam from. This energy map holds the “energy” of all the pixels, computed by summing the absolute value of the gradients in the x direction and y direction, which then is used to determine the significance of each pixel. It is calculated based on the pixel values of the image in grayscale. Once that map is fully calculated, we used a dynamic programming approach to backtrack and find a “seam” of pixels from the first row to the last that holds the lowest cumulative cost and therefore the least amount of meaning. The corresponding seam is then found on the image and those pixels are discarded, which also entails filling the gap with the surrounding pixels. Each seam removal decreases one of the dimensions by 1. Because the selection of the seam is dynamic, it will only find a path from the top of the image to the bottom with the lowest total energy values out of all the possible path options.

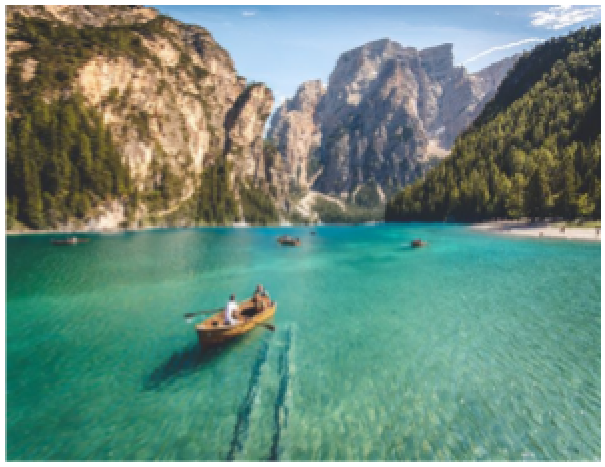
Both of these computations are performed by functions in our *utils.py* file. The main Python notebook that faces the user prompts them to specify what percentage of both the height and width of the original image they would like to keep. We calculate the number of seams that must be taken off each image side, and iteratively call our seam finding and removing functions to achieve the given dimensions. We first find seams from the top of the image to the bottom, to decrease the width of the original image, and then find seams from the right of the image to the left to decrease the height. We allow the user to save each frame of the seam removal process to combine into a video later on, as well as save and display both the original and resized images for the users to see the changes relative to each other.

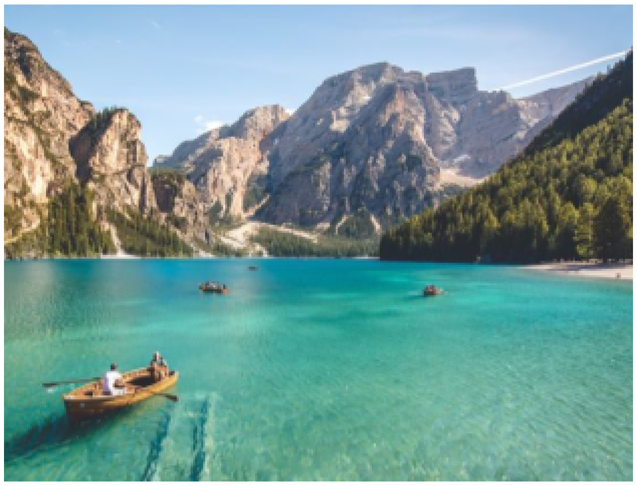
Results:

We ran our algorithm on several different images and found that it was substantially better at resizing images than simply cropping. Using the energy maps of the images, the algorithm was able to differentiate between important objects and trivial features of the images and specifically chose the least meaningful seams to be eliminated, thereby keeping the objects in view.

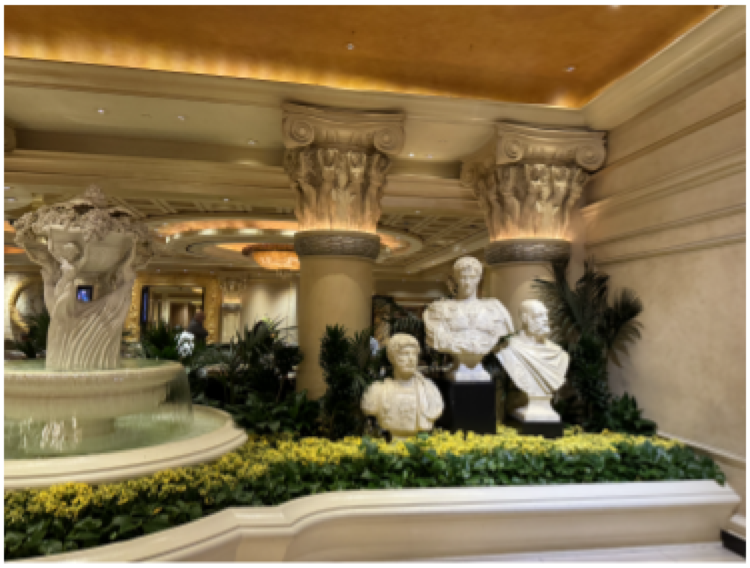
Here are just a few samples of inputs and outputs produced by our project:

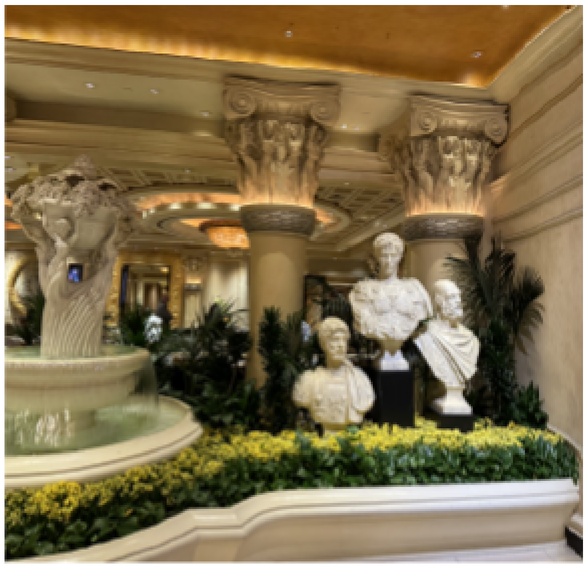

Original Dimensions: 800 x 533


Resized Dimensions: 560 x 426

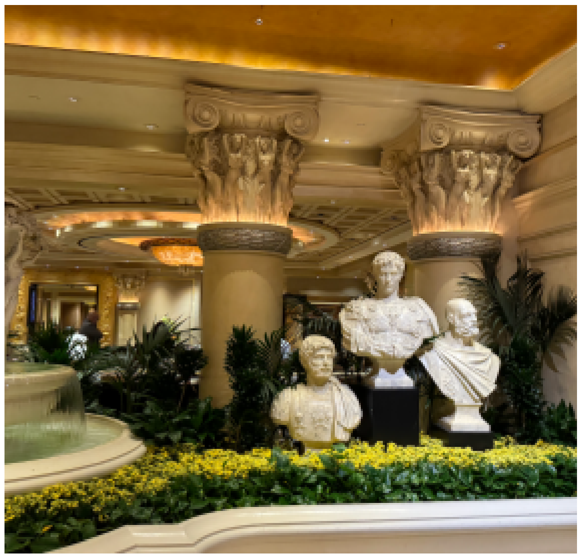

Cropped Dimensions: 560 x 426

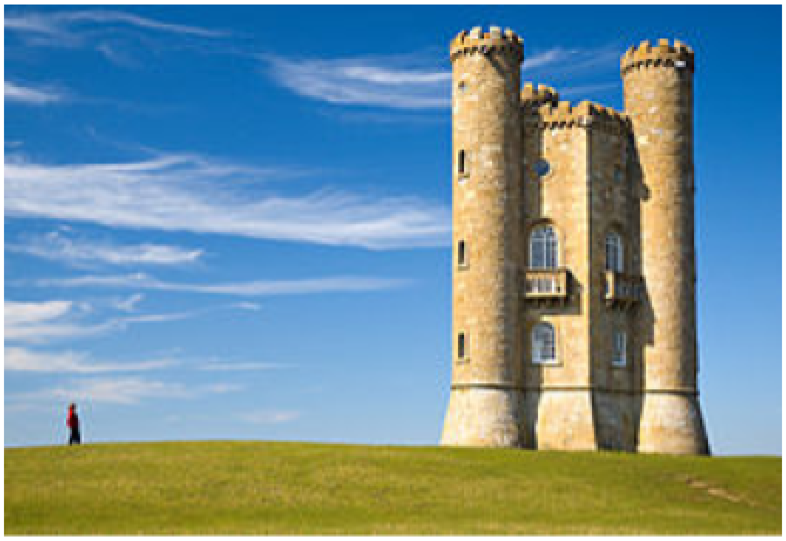
If you look at the cropped version of the original image, with the same dimensions as the resized image, you can see that a lot of the meaningful features along the borders have been cropped out. For example, the cloud in the top-right corner and a significant portion of the mountainside on the left have all been cut out of the image.

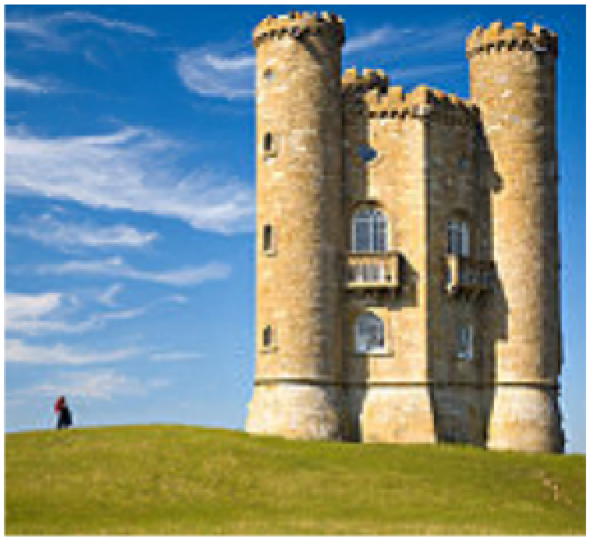

Original Dimensions: 1850 x 1380

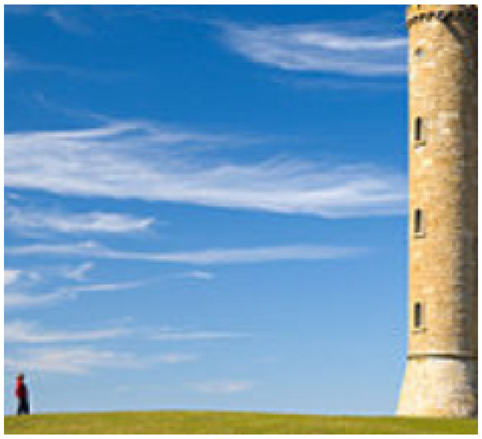

Resized Dimensions: 1295 x 1242

Once again, the cropped version of the image, made to have the same dimensions as the resized version, cuts out significant features of the original image. In this case, almost all of the fountain is not visible in the cropped version, while the seam-carved version maintains the fountain by removing seams along the right-hand wall and in the space between the busts and the fountain.


Cropped Dimensions: 1295 x 1242


Original Dimensions: 274 x 186


Resized Dimensions: 164 x 149


Cropped Dimensions: 164 x 149

Finally, for our last example, it is clear that less than a third of the castle is visible in the cropped version. The seam carving algorithm for image resizing allows the viewer to see all the meaningful aspects of the image, without having to compromise. In the naively cropped image, for the person to be visible, the castle would have to be cropped, and for the entire castle to be visible, the person would have to be cut.

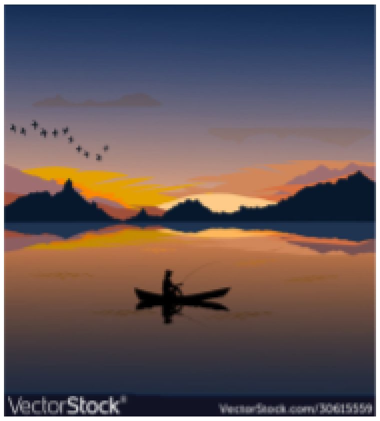
Here is a video of the seams being found and removed in real-time on an image of Vincent Van Gogh's 'Wheatfield': https://www.youtube.com/watch?v=-DyiftGUCCY

Another video of real-time seam carving on a picture of the Eiffel Tower:

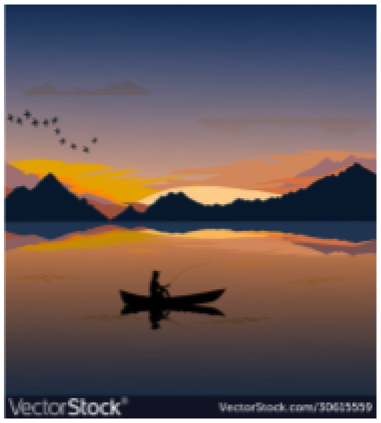
https://www.youtube.com/watch?v=Ib6XV-ObM2I

We tested our various outputs against other publicly available implementations of resizing and seam-carving, which can all be seen below:

VectorStock
VectorStock.com/30615559
Original Dimensions: 1000 x 830

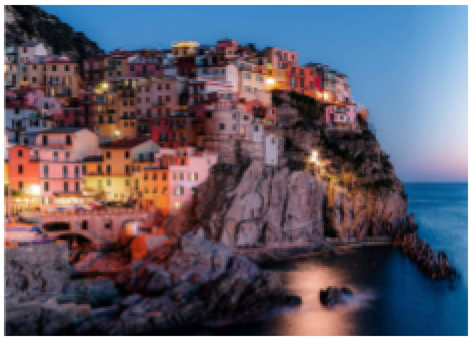


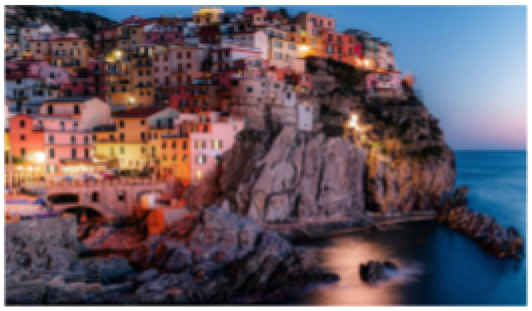
Our Implementation Resized Dimensions: 600 x 664

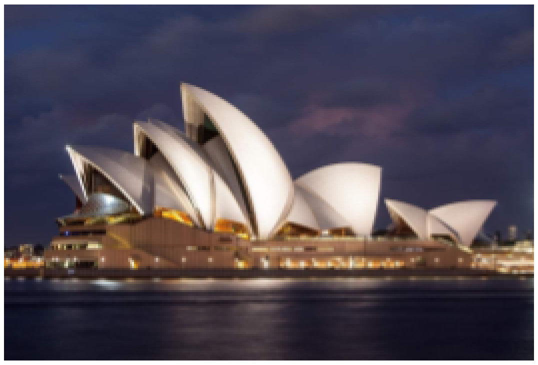


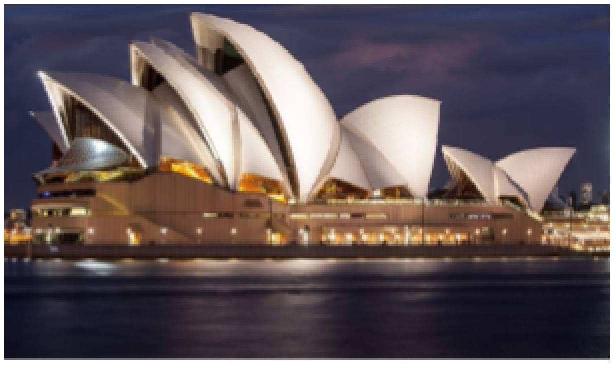
Public Implementation Resized Dimensions: 600 x 664

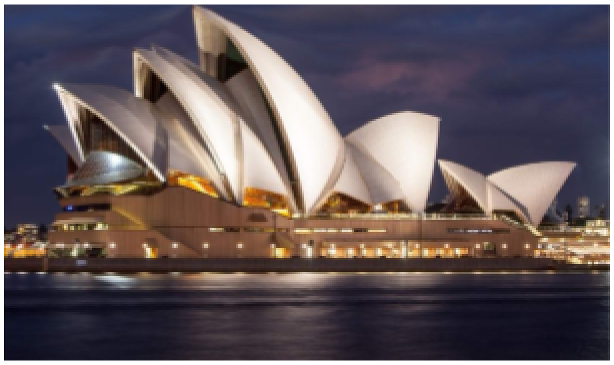
Original Dimensions: 2160 x 1080


Our Implementation Resized Dimensions: 1512 x 1080


Public Implementation Resized Dimensions: 1512 x 1080


Original Dimensions: 1200 x 800


Our Implementation
Resized Dimensions: 960 x 560


Public Implementation
Resized Dimensions: 960 x 560

The public implementation of the seam-carving algorithm that we checked ours against can be found here: https://trekhleb.dev/js-image-carver/.

Implementation Details:

For our implementation, we chose to use Python and the number of packages that it has to offer, specifically cv2, numpy, random, matplotlib, and ipywidgets. Numpy and cv2 were primarily used for the actual image transformations and calculations, while matplotlib was used for displays. Ipywidgets was used for user interactivity like uploading images to be resized and inputting dimensions parameters. Ipywidgets is a package that offers web-based controls and features specifically for Jupyter Notebooks. For our interface, we used Jupyter Notebook for easy flow and interactivity.

Challenge / Innovation:

As we went through our implementation, there were a couple of times when we faced challenges and had to innovate ways to overcome them. One such example was when we were trying to figure out how to create the energy paths after deriving the maps. We realized we didn't just want to find the path of pixels from either side of the image that individually had the lowest energy values, but instead wanted to consider the total sum of energy values of the pixels in the total path – even if that meant choosing a high-energy pixel to get to a lower-energy pixel. We realized that we would have to find this path dynamically, so we employed a backtracking algorithm to significantly reduce the running time of finding the right seam to eliminate.

Another challenge that we faced was how to remove the horizontal seams after we had figured out the vertical seam removal. We had begun with a naive approach to this by creating a whole new method of finding the path with the lowest cumulative energy from the left to the right of the image. However, this was quite cumbersome and was a little more confusing to conceptualize than the top-to-bottom approach. After refraining the problem, we realized we could rotate the image and then apply our initial top-to-bottom seam-finding algorithm, to find horizontal seams from left to right. This saved us quite a bit of time and made our final source code a lot more concise and easy to follow.

All in all, we learned a lot from completing this project and although there were only 2 of us working together, we were able to present a working algorithm that effectively and accurately resizes images using seam-carving and energy functions. We hope to receive full marks for the innovation and challenge portion of this project because we found that there weren't a lot of public implementations that resized images in both the height and width dimensions and by providing percentages instead of fixed pixel values. On top of this, we implemented a neat trick to rotate the image and then apply the same algorithm, and we also allowed the user to save the image frame by frame after each seam removal, so they can make their own videos or gifs.

Contributions:

Group members contributed similarly

Trisha: Worked on creating the energy map, finding seam algorithm, the report, testing algorithm

Prashant: Worked on the jupyter notebook, finding seam algorithm, removing seams, testing algorithm