Zaryab Farooq
Kiranpreet Kaur
ECS 174
04/20/2018
Professor Yong

Problem Set 1

I. Short Answer Problems

1. Associative property of convolution: (f * g) * h = f * (g * h)
One example of how you can exploit the associative property of convolution to more efficiently filter an image is:
If **f** is the picture, then instead of applying 2 filters separately by first applying filter **g** then filter **h**, we can combine **g** and **h** and apply it to **f**. This will help if the picture is very large.
Example 1

A*f1	= A'
A'*f2	= Result

There are too many multiplications and additions occurring since there would be two operations of applying the filter on the image.

Example 2

(A*f1)*f2	= A*(f1*f2)
Let f3	= f1*f2
A * f3	= Result

There is only one filter being applied to the image which means less operations to perform

2. Given an input image: [1 0 1 1 1 1 1 1], the result of erosion with a structuring element [1 1 1] will result in am image: **[0 0 0 1 1 1 1 1].**

3. A possible flaw in the use of additive Gaussian noise to represent image noise is that the full covariance matrix found in the case of joint energy function in problems with Gaussian noise may be too large to compute and store. For example, in large structure from motion problems, a large sparse Hessian normally results in a full dense covariance matrix. In such cases, it is often considered acceptable to report only the variance in the estimated quantities or simple covariance estimates on individual parameters, such as 3D point positions or camera pose estimates.

In the real world, the noise is not random and sigmoid function that is used in the data type only has random noise.

4. First of all, I will feed an ideal image of how the part should look like from the camera in the system. A camera will then be perched above a conveyor belt at an automotive equipment manufacturer, the video data from which would be transmitted to the system. In the system, I would group video frames into shots because grouping in vision can be used to obtain an intermediate representation that compactly describes key image (video) parts.

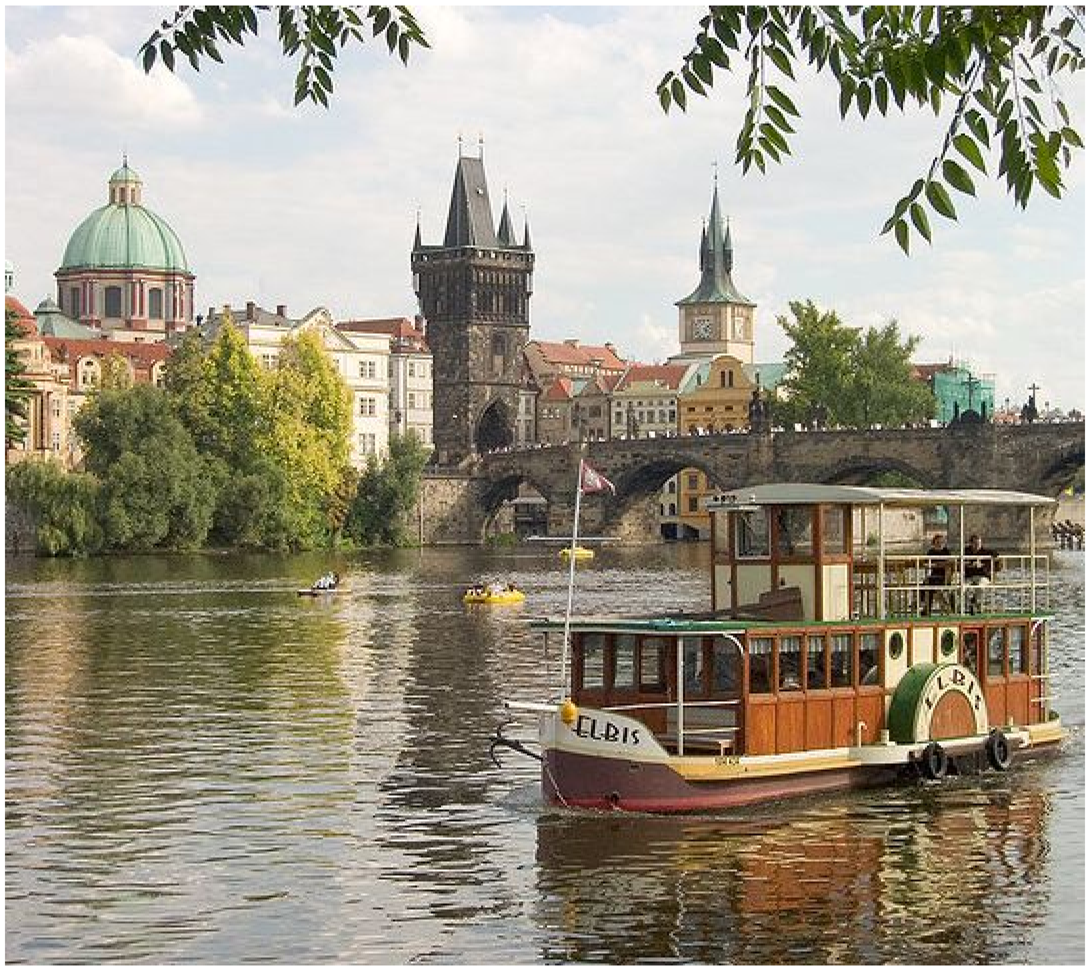
Here, I am essentially trying to compare two images: one from the video feed taken by the camera with an ideal image of the part in the system.
Process in the system:
I would separate image into "coherent objects" by using the process of segmentation. The goal of segmentation is to group together similar looking pixels for efficiency for further processing. I will use graph cut algorithm to separate objects in the picture from background. Then the edge detection will come in handy to find the edges of the objects and compare with the edges of the object in the ideal image. Also, we could do another form of comparison in the system here. We can display the assembly line image and the ideal image as matrices. And then we can use the **diff()** function in matlab to check for any differences that might exist in the elements of the matrices such as dimensions of the object as we have already separated the image using graph cut algorithm.
Now, we would compare the image one more time by calculating the gradient images. On the gradient image, we will use the same method as did in the view seam function in this code. That function will change the pixels of the image to show exactly where the difference lies.
The system will detect the flaw and display it as a warning on the screen. We can use an if statement in the code to display that warning whenever any mismatch is found. When we display this warning, we would also display the error image (with changed color of the pixels) on the screen.

Assumptions:
1) I would assume that the camera will capture images from different angles and these images could be easily captured from the video.
2) I would assume that the texture of the image captured by the camera is same as the texture of the test image already feeded into the program for comparison. Thus, the comparison is between similar textures of the images.
3) Any difference in these images reports a flaw in the the assembly of a part and not just a mere difference of the two images.

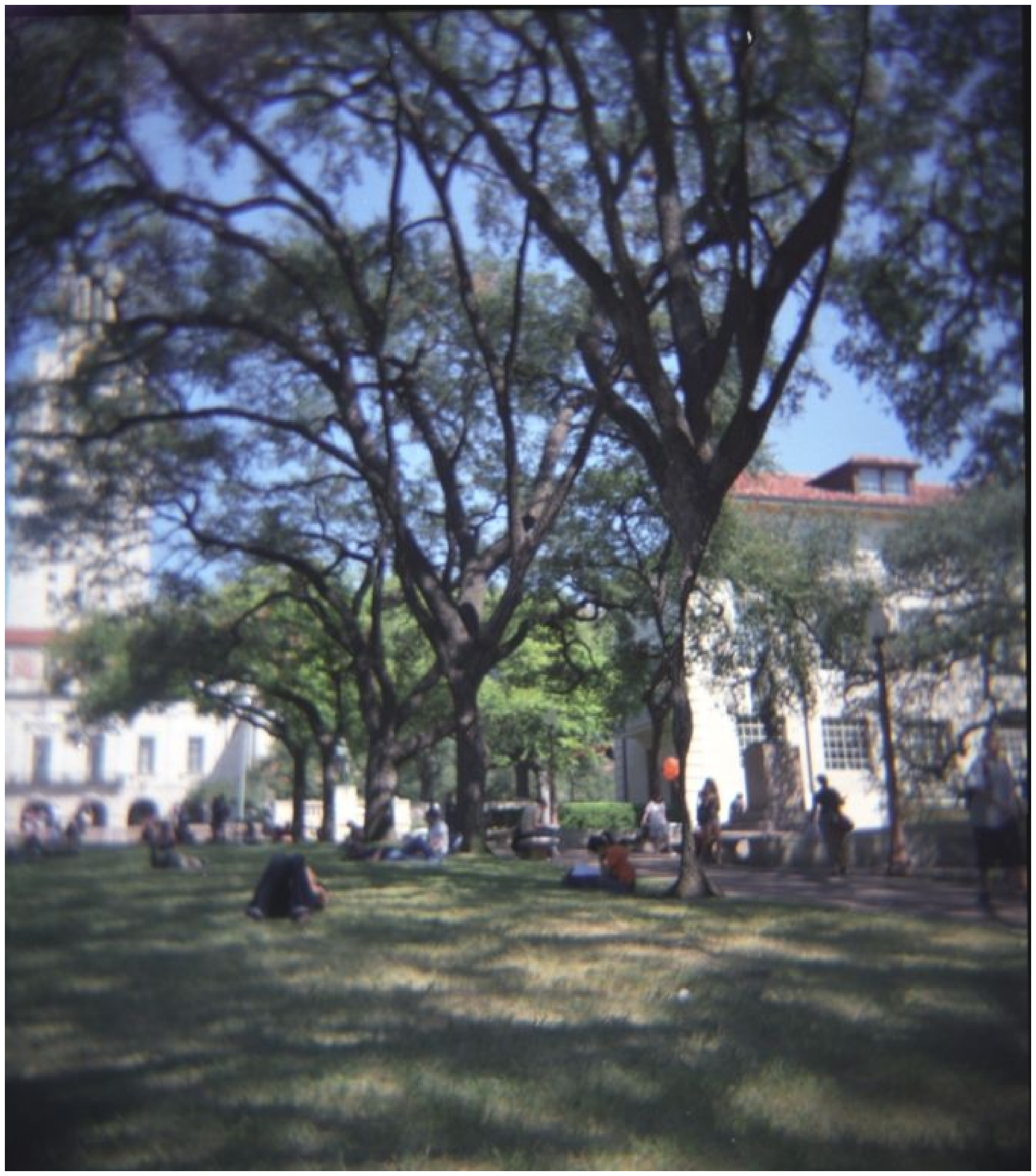
II. Programming problem: content-aware image resizing

1. Reduce the width of the image by 100 pixels.

The resulting output image is displayed as: outputReduceWidthPrague.png



The resulting output image is displayed as: outputReduceWidthMall.png

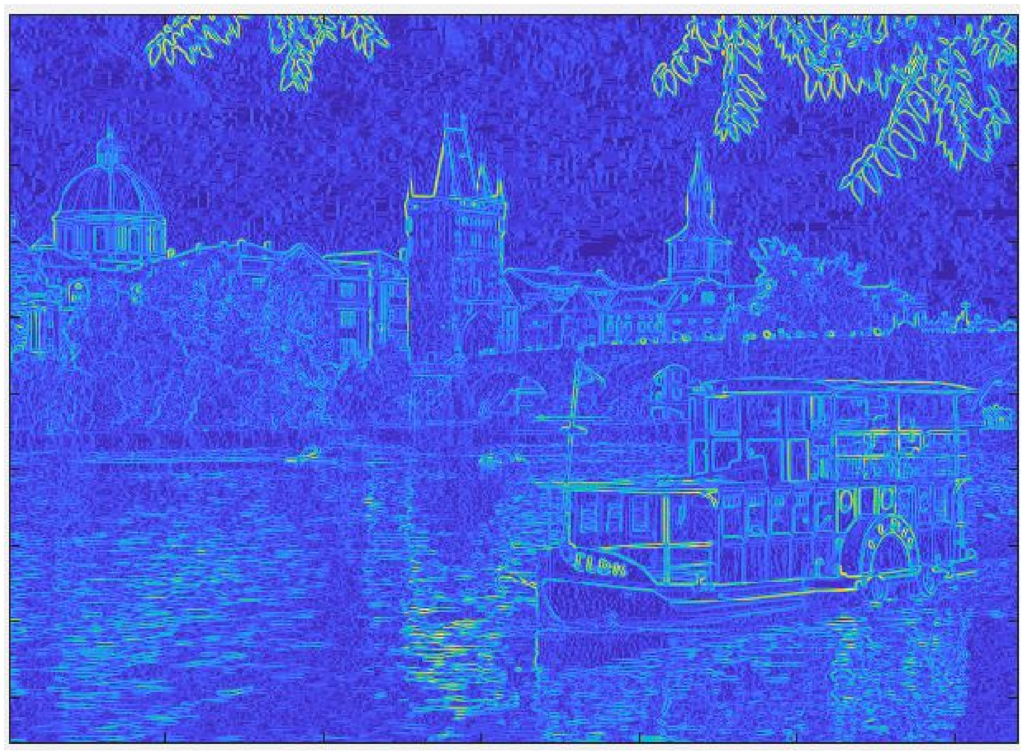


2. Reduce the height of the image by 50 pixels. SAVE SCRIPT

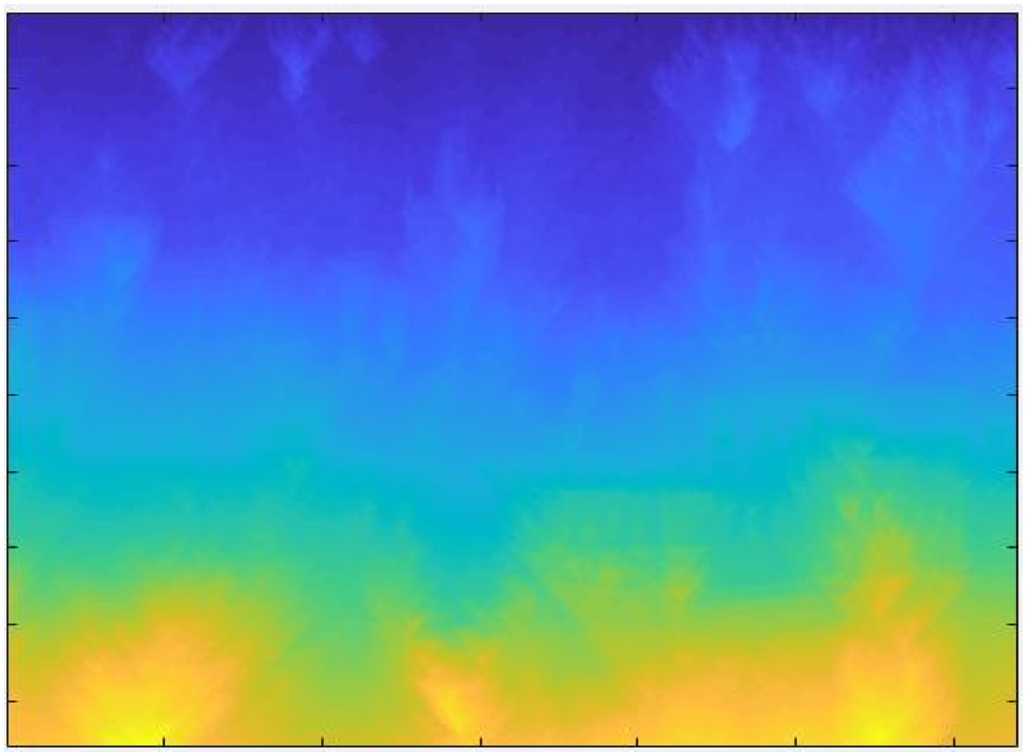
The resulting output image is displayed as: outputReduceHeightPrague.png

The resulting output image is displayed as: outputReduceHeightMall.png

3. (a) Display the energy function output for the provided Prague image:



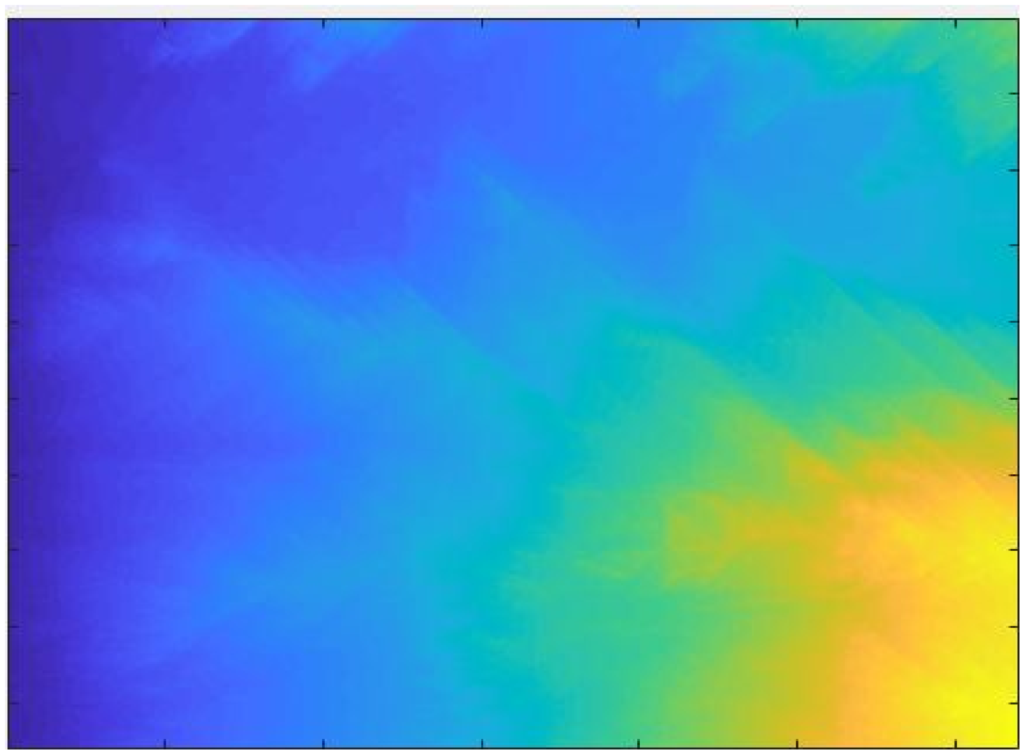
(b) **Vertical Energy**



The bright(yellow) area represent regions of high energy and similarly dark (blue) area represent low energy.

The reason we see a very high energy in the bottom is due to the fact that there is a lot of color changes in the water due to the reflections of the buildings and the boat.
We start at the top row and for each pixel add a lowest energy pixel from 3-connected pixels in the previous row cumulatively until the last row and repeat this for each column. We will get lo

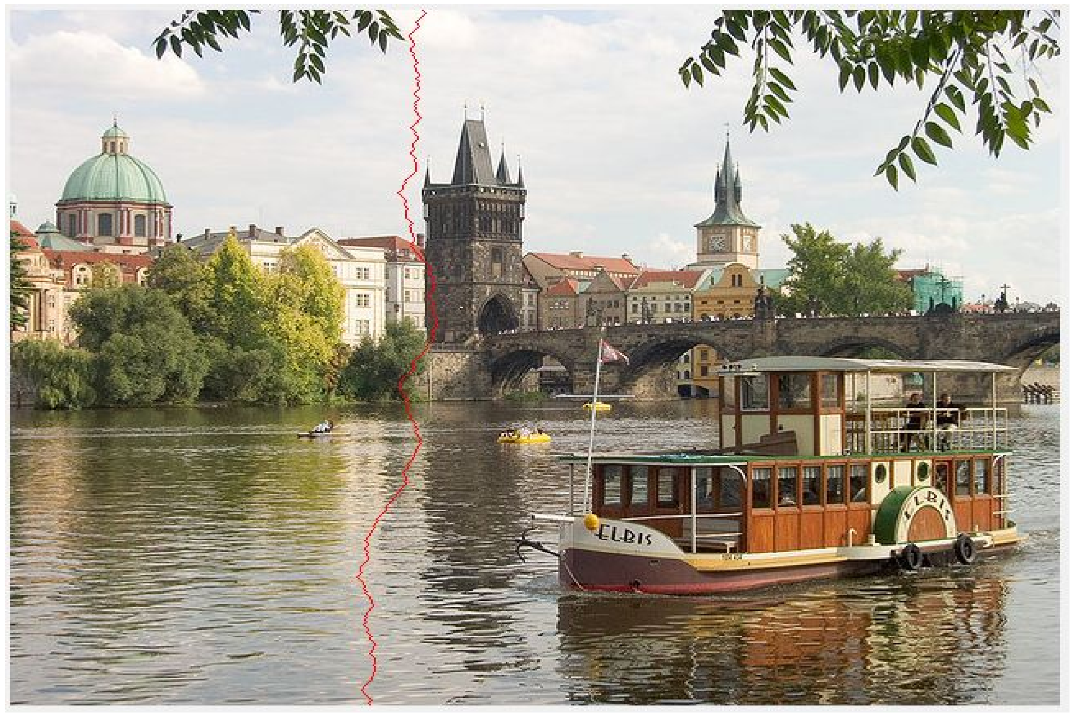
Horizontal Energy



Similarly, the horizontal seam has high energy regions on the water due to the reflections and slightly high energy region on the top right part of the image. The reason of high energy on top right is due to transition of colors between the dark leaves and the lighter blue sky.
The algorithm works as follows. We start at the top row and for each pixel add a lowest energy pixel from 3-connected pixels in the previous row cumulatively until the last row and repeat this for each column.

4. (a) **Original image:**

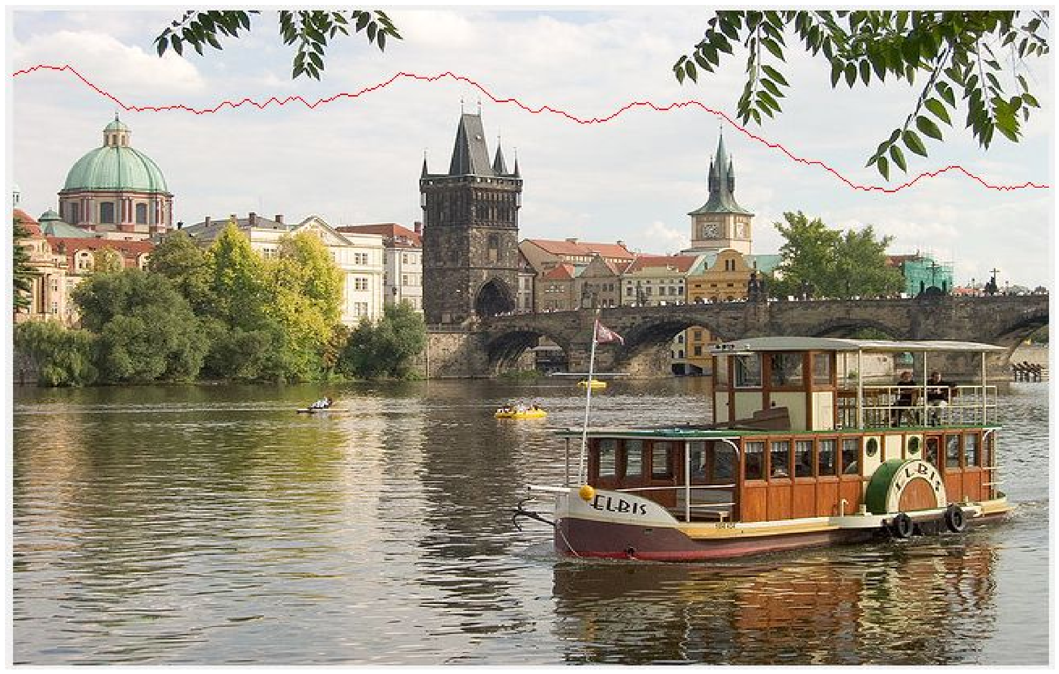
Vertical Seam:



The idea of vertical seam is to, first, find the pixel with the minimum energy at the bottom row. Then, keep looking at the immediate top 3 pixels and choose the minimum of them until the top most row. As we can see, the vertical seam in the picture does not really cut through the white building after the bushes but goes to the black tower since the energy was less as compared to between bush and the bright white building. Here, the seam is optimal because it starts from the minimum energy pixel at the bottom and keeps looking for the next minimum. The reason it does not passes through the boat shows it is optimal also.

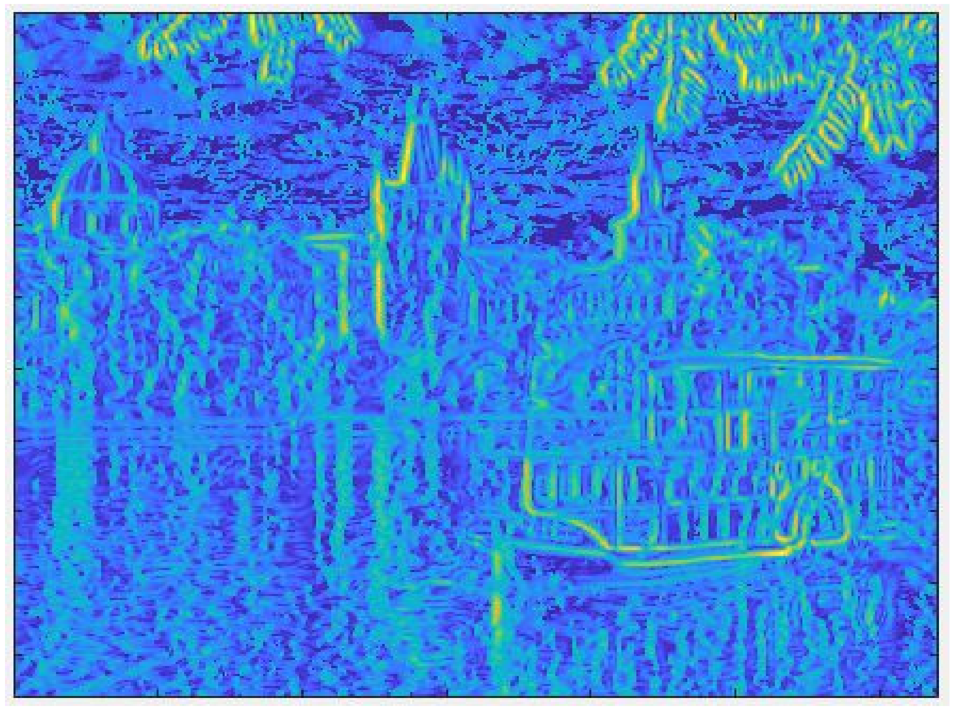
Original image:

Horizontal Seam:



We have the similar explanation for horizontal seam, as like the vertical seam, except that we start from the right most side rather than bottom. The sky was the only area where the intensity change was the minimum as even the water had many different intensities because of the reflection of the building.

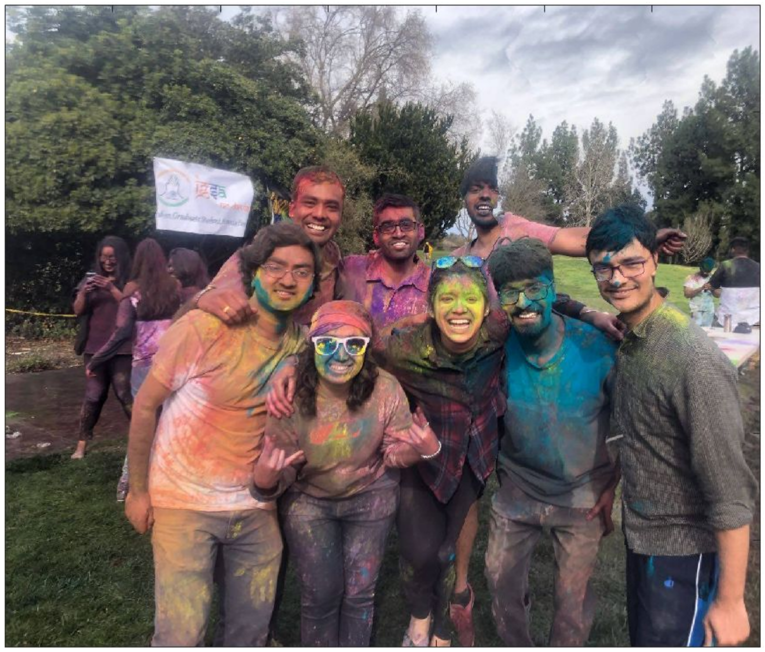
5.



So we increased the contrast of the image by using the **imadjust** and applied gaussian smoothing with sigma = 1 using **imguassfilt.**

__**Example**__

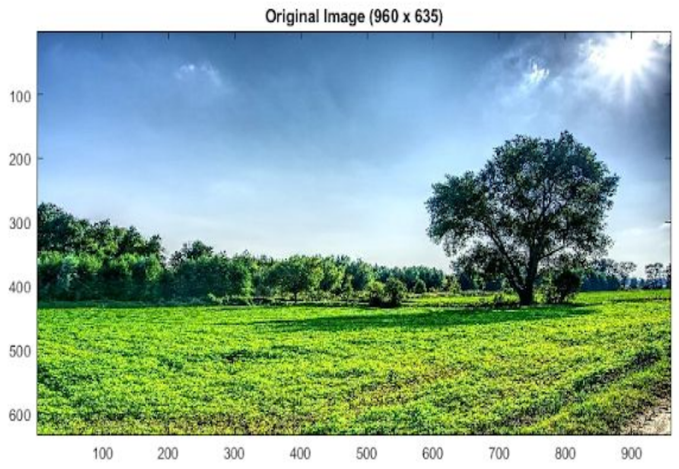
Resized Image

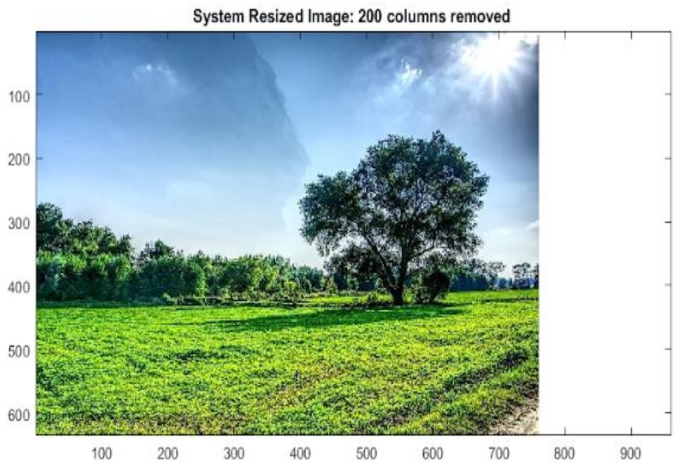


This one was not a really successful seam carving and it cannot be blamed on the energy image. We can clearly see a person's face chop of from the picture as our algorithm is not aware of the faces and cannot do much to protect them. But it did tries its best to refrain from carving a seam through an object. Since there was not much space for it to delete other than the group of people in the front.

6. <u>Image 1:</u> (https://www.pexels.com/search/landscape/)

As we can observe, this is a successful resizing of the image. Since there was only one main object in the picture, the algorithm recognized the useless background and deleted those 200 pixels to resize the image. The big tree in the focus does stays perfectly fine and no distortion can be seen on it.





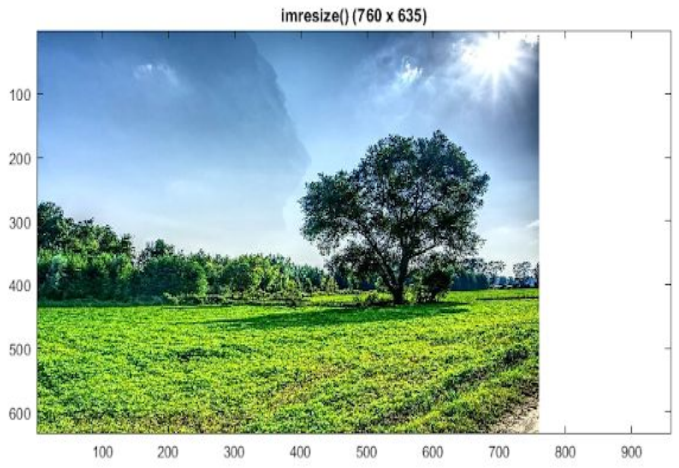
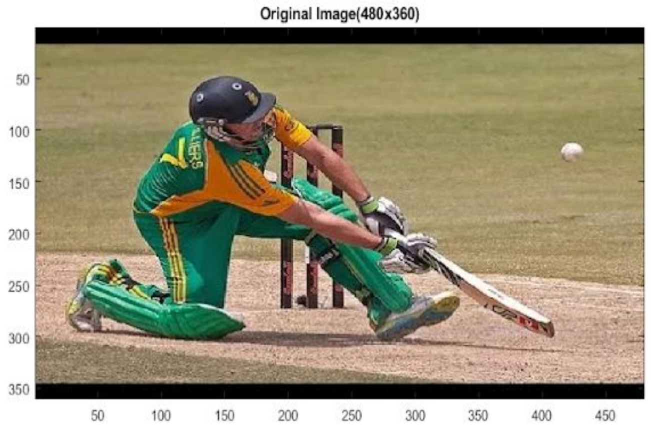
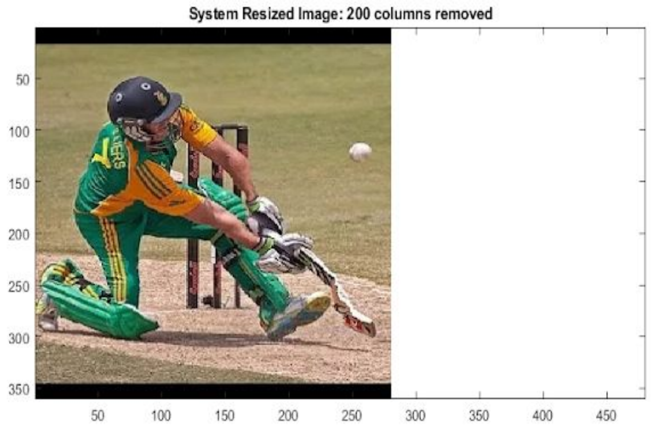
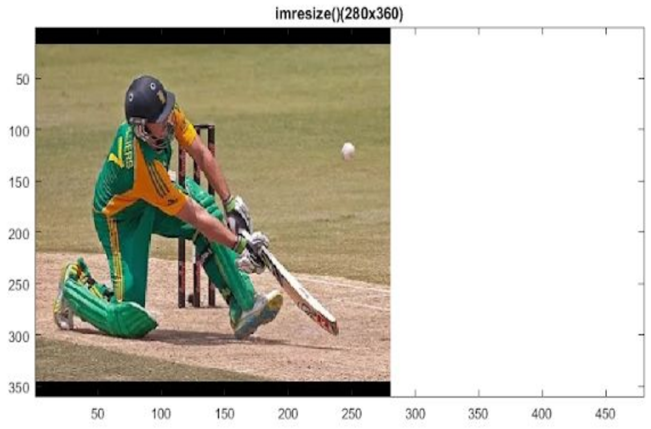


Image 2: (www.cricbuzz.com)

Now, if we delete the same number of pixels from this picture of a cricket player where the subject takes up a larger portion of the picture and the space around is minimum, the algorithm did not have choice than to delete pixels from the subject. Interestingly, instead of the player itself, the algorithm removed pixels through the bat since it took less image space. This was actually a good decision not to remove columns through the player but resizing still deformed the bat. We can fix this by resizing the height which will do less deformation to the pixels as there is plenty space above and below the subject which the algorithm can delete.

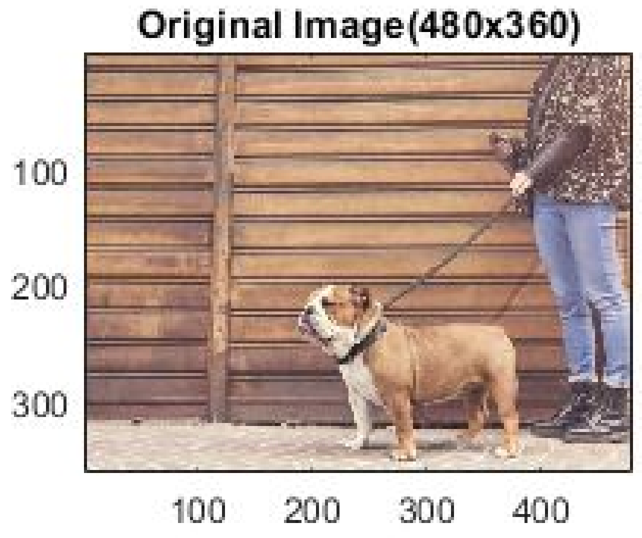


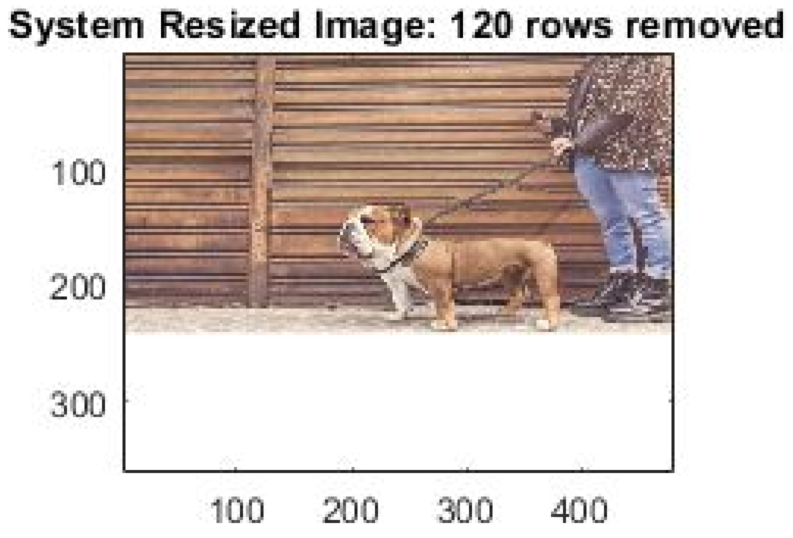


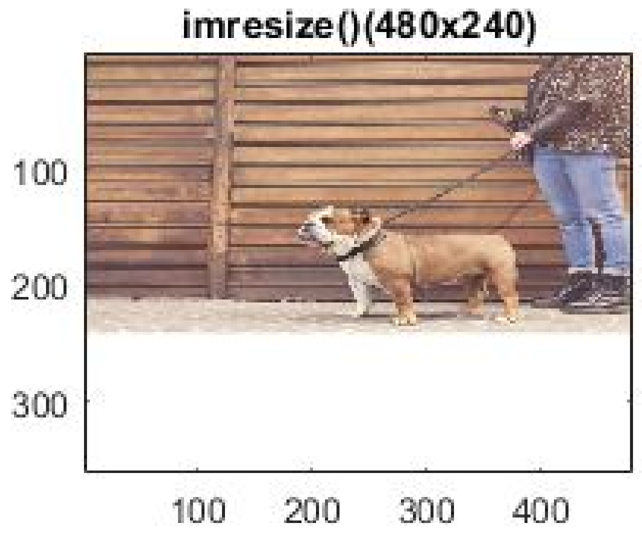


<u>Image 3:</u> (https://www.pexels.com/search/pet/)

This is a successful example of seam carving. We removed 120 pixels horizontally, and it did not deform any of the main subjects. The dog's and the woman's shape stayed quite good even though most of the seams went through the subject. It was interesting to see that if we did vertical seam carving on it, the dog and the woman would have stayed almost the same (except their height decreased too) and there would not have been interesting to look at it. Thus the reason we chose to do horizontal seam carving.



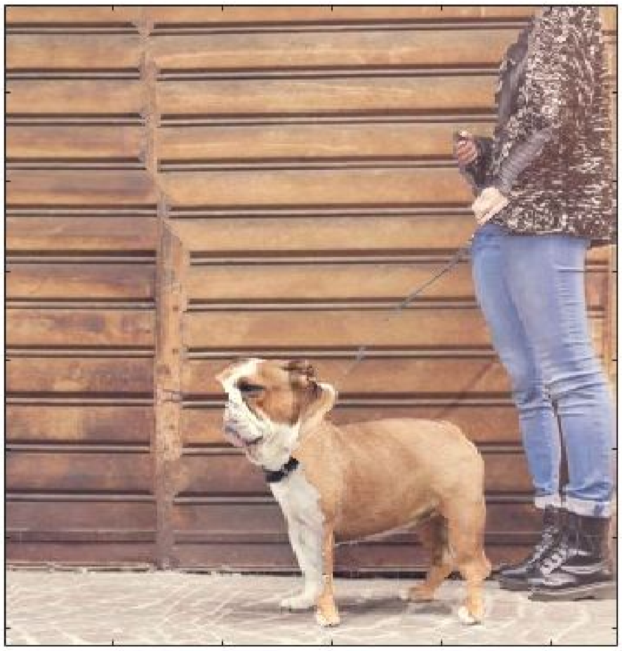




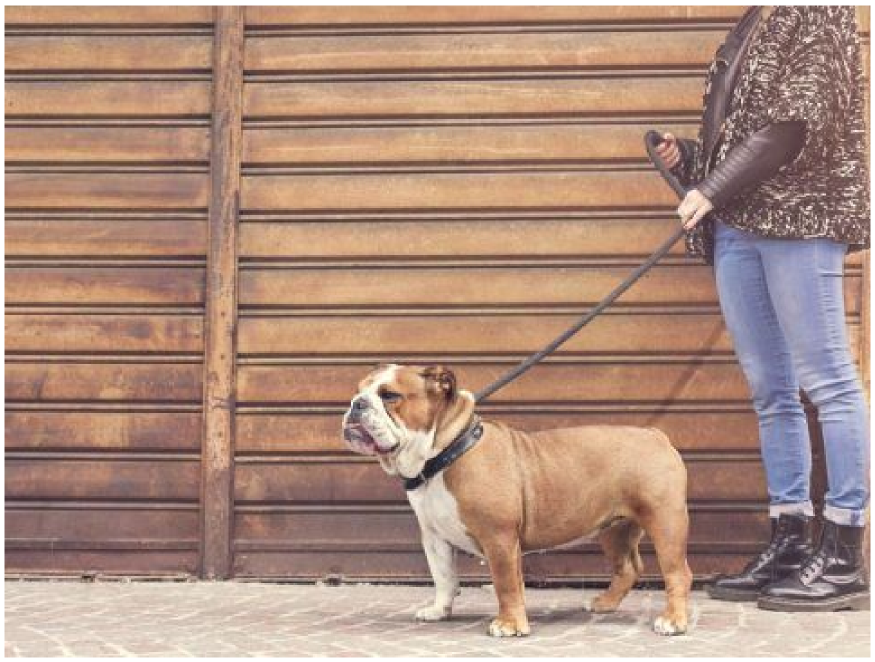
III. [OPTIONAL] Extra credit [up to 10 points each, max possible 10 points extra credit]

Use an existing extension or design your own extension and give an explanation of the extension as well as images displaying the results and a short explanation of the outcomes.
4- We did the fourth question as extra credit which is finding the **<u>Greedy solution</u>**

Greedy solution



Original Image



In this part, we created a new function similar to decrease width where instead of finding vertical seam on the energy image, we found the seam on the original image itself. What this does is that the algorithm is unaware of the low energy pixels and directly finds a minimum valued pixel. Due to this, prominent objects on the screen are really distorted and as we can see, the dog's face is almost deleted just like the woman's body.

To run the code, just run the script called **greedySolutionEC.m** and it displays the new image including the seams at each iteration.