CS 4476 - Intro to Computer Vision
PS2 worksheet
Jingbo Wang

Short answer problems

1. **Give an example of how one can exploit the associative property of convolution to more efficiently filter an image.**

Given an image *f*, two filters *g* and *h*. If the filters are separately applied to the image *f*, with many pixels in *f*, each filter needs to go through the entire image once, which can be a long runtime with just one filter. With two filters, this long process will be doubled. By the same reasons, with *k* of these filters, this long process will be increased by *k* times, which can easily result in an extremely long runtime. However, with the associative property of convolution, the original process of (f * g) * h is equivalent to the process of f * (g * h). When the filters are first convolved with each other and combined to one filter, which takes a trivial amount of time comparing to the runtime of a filter convolve the entire image because a filter normally has way less pixels than an image, this combined filter can just go through the entire image once to reach the final result, which is a lot more efficient by applying filters one by one to the image. With the same reasoning, when there are *k* filters to be applied to an image, these filters can first convolve and combine into one filter, and this final filter can just be applied to the image once to get the final image.

2. **This is the input image: [0 0 1 1 0 0 1 1] . What is the result of dilation with a structuring element [1 1 1] ?**

Result: [0 1 1 1 1 1 1 1]

3. **The filter f' = [0 -1/2 0 1/2 0] is used as the filter to compute an estimate of the first derivative of the image in the x direction. What is the corresponding second derivative filter f" . (Hint: Asymmetric filters must be flipped prior to convolution.)**

Given image h, apply the filter f' to h, the result is the first derivative in the x direction h'. Then apply another first derivative f' to h', the result is the second derivative in the x direction h''. The h'' is also the result of image h filtered with the second derivative filter f'', which makes f'' = f' * f'. [0 -1/2 0 1/2 0] * [0 -1/2 0 1/2 0] = [1/4 0 -1/2 0 1/4]

Another way to prove it:
Image:

x_0	x_1	x_2	x_3	x_4	— apply filter f' —>	$\frac{x_1}{2}$	$\frac{x_2}{2}-\frac{x_0}{2}$	$\frac{x_3}{2}-\frac{x_1}{2}$	$\frac{x_4}{2}-\frac{x_2}{2}$	$-\frac{x_3}{2}$

$\frac{x_1}{2}$	$\frac{x_2}{2}-\frac{x_0}{2}$	$\frac{x_3}{2}-\frac{x_1}{2}$	$\frac{x_4}{2}-\frac{x_2}{2}$	$-\frac{x_3}{2}$	— apply filter f' —>

$\frac{x_2}{4}-\frac{x_0}{4}$	$\frac{x_3}{4}-\frac{x_1}{2}$	$\frac{x_4}{4}-\frac{x_2}{4}-\frac{x_2}{4}+\frac{x_0}{4}=\frac{x_4}{4}-\frac{x_2}{2}+\frac{x_0}{4}$	$-\frac{x_3}{2}+\frac{x_1}{4}$	$\frac{x_2}{4}-\frac{x_4}{4}$	

This also shows that:

x_0	x_1	x_2	x_3	x_4

— apply filter f" —>

$\frac{x_2}{4} - \frac{x_0}{4}$	$\frac{x_3}{4} - \frac{x_1}{2}$	$\frac{x_4}{4} - \frac{x_2}{4} - \frac{x_2}{4} + \frac{x_0}{4} = \frac{x_4}{4} - \frac{x_2}{2} + \frac{x_0}{4}$	$-\frac{x_3}{2} + \frac{x_1}{4}$	$\frac{x_2}{4} - \frac{x_4}{4}$

If I put the filter f" centered at x_2, I should be able to get $\frac{x_4}{4} - \frac{x_2}{2} + \frac{x_0}{4}$, which gives a filter that looks like: [1/4 0 -1/2 0 1/4]. This result is exactly the same as that one I got above. Thus the answer is [1/4 0 -1/2 0 1/4].

4. Name two specific ways in which one could reduce the amount of fine, detailed edges that are detected with the Canny edge detector.

1) Use the high thresholding to filter out the too fine and detailed edges.
2) Not applying non-maximum suppression can makes the edges blurry and obscure, which, in another word, makes the edges less fine and detailed.

5. Describe a possible flaw in the use of additive Gaussian noise to represent image noise.

Addictive gaussian noise has a strong assumption about the perfect gaussian distribution, where the noise will be added evenly according to the gaussian distribution. However, such assumption is never valid for real world scenarios, which can make the perfect distributed gaussian noise inaccurate. There are also many other types of noises that the addictive gaussian noise cannot model, such as salt and pepper and impulse noise.

6. Design a method that takes video data from a camera perched above a conveyor belt at an automotive equipment manufacturer, and reports any flaws in the assembly of a part. Your response should be a list of concise, specific steps, and should incorporate several techniques covered in class thus far. Specify any important assumptions your method makes.

1) First take an image of the ideal part
2) Use binary image analyze to recognize shapes and features on this ideal part image
3) For each part in the assembly, use some of the edge detection methods, such as canny edge detection with hysteresis threshold with higher high thresholds to extract out the strongest edges
4) Use the chamfer distance to compare strong edges extracted to the recognized features and give an overall score of how well they match
5) Use a threshold value to determine is this overall score indicates a flaw in the part

Important assumptions:

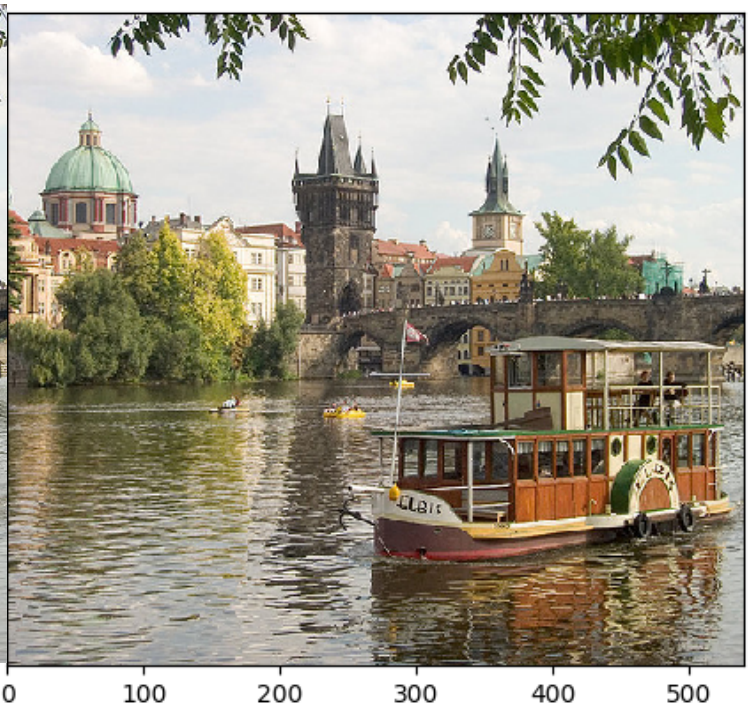
- there is an ideal image available
- this image has apparent features and edges
- each part in the assembly is in the same size and the same orientation
- there is already some testing previous to the deployment to determine the best threshold value to use

Programming problem: content-aware image resizing

1. Reduce Width by 100 pixels

Original Image

Reduced Width



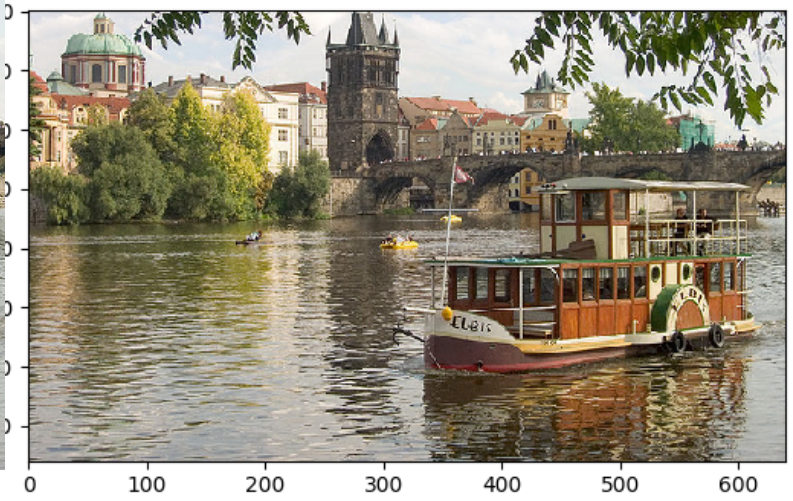
Original Image

Reduced Width

2. Reduce Height by 100 pixels

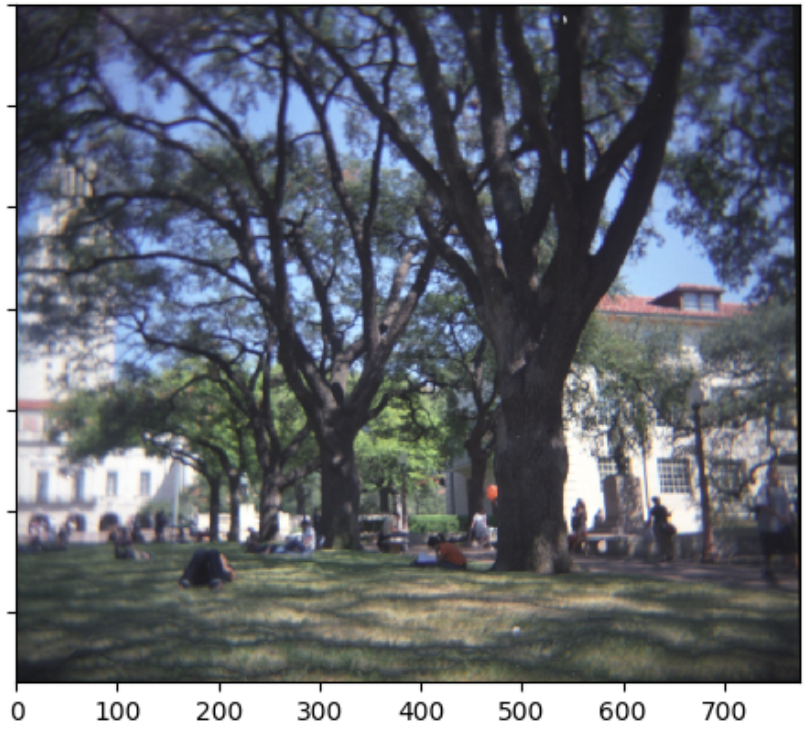
Original Image

Reduced Height



Original Image

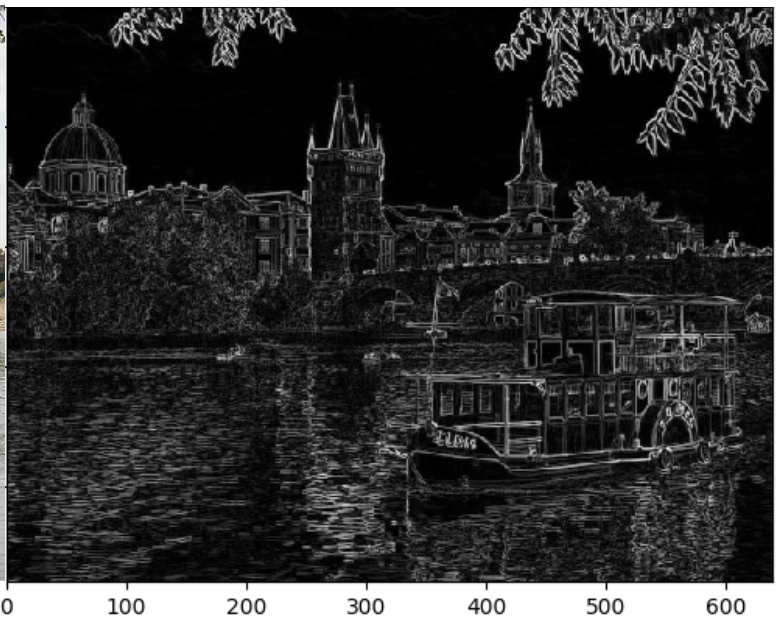
Reduced Height



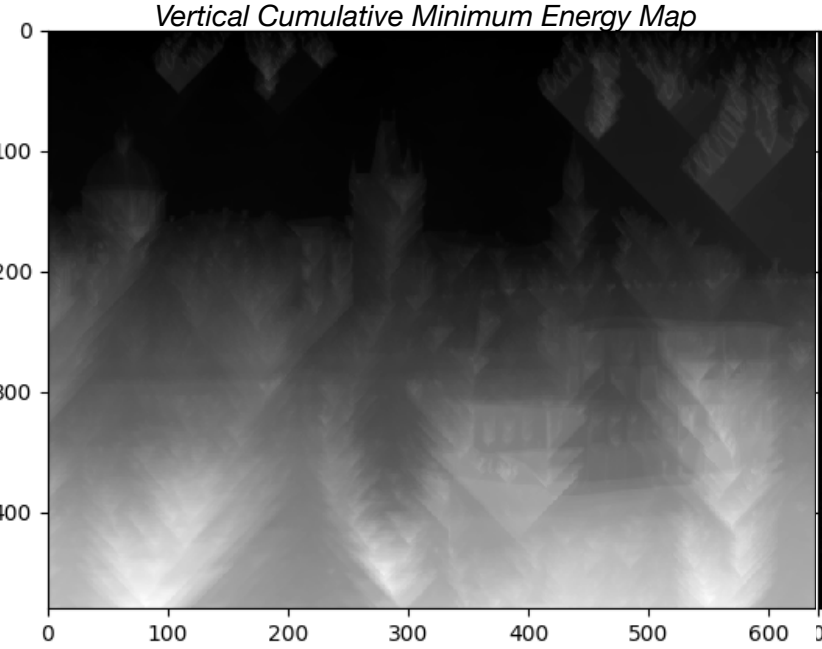
3. Energy Image and Cumulative Minimum Energy Maps

Original Image

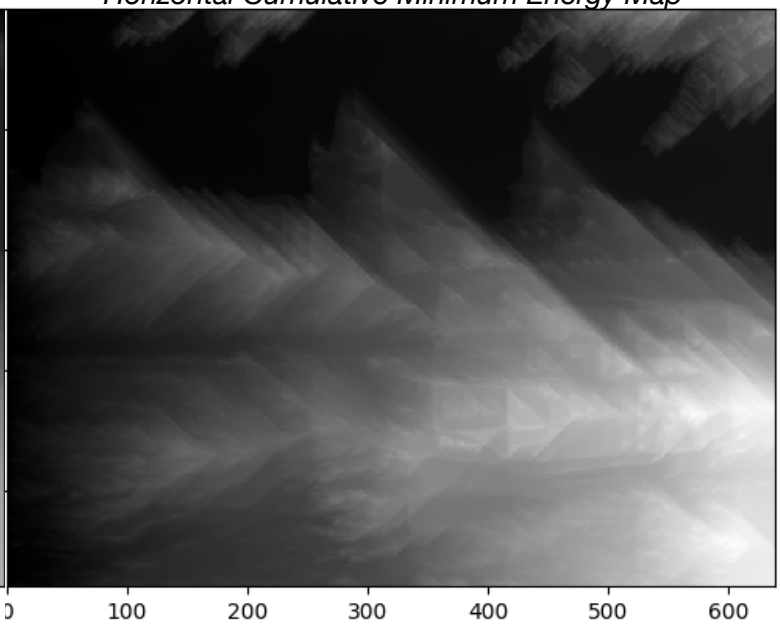
Energy Image



Vertical Cumulative Minimum Energy Map



Horizontal Cumulative Minimum Energy Map



The gradients in the energy image are calculated using Prewitt masks. When the intensities of neighboring pixels have a noticeable change, a high intensity value is put at that pixel's coordinate in the energy image, indicating an edge. In the energy image, all the edges are clearly extracted, and even the leaves in the trees on the left hand-side and ripples in the water.

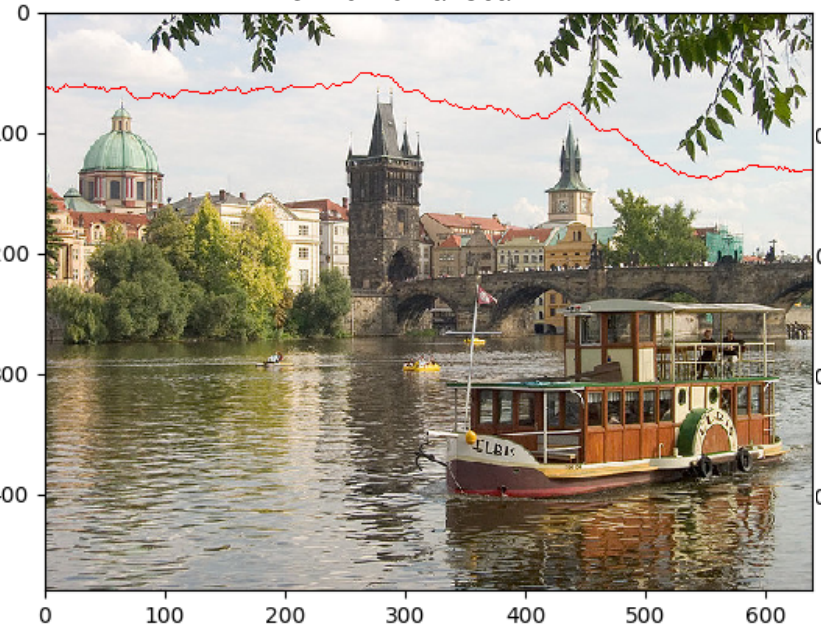
In the vertical cumulative minimum energy map, the intensity of a pixel basically means how many edges the path/seam, at least, has to cross to get to this pixel from the top

(the higher intensity, the more edges). The leaves on the top, the top of the houses and towers are clearly depicted in this map because they are reasonably the first high energy edges on the way from the top. The lower right region has a high cumulative energy because of the bridge, the boat and the reflection of the boat. The lower left region also has a high cumulative energy mainly because of the reflection of the trees and the ripples. In the lower center, it shows the reflection of the tower as well.

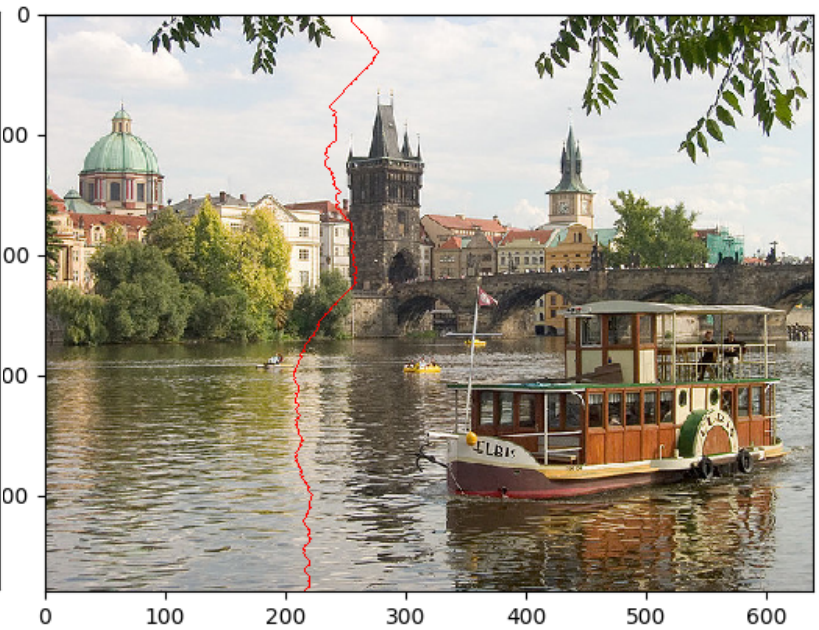
In the horizontal cumulative minimum energy map, the intensity of a pixel basically means how many edges the path/seam, at least, has to cross to get to this pixel from the left (the higher intensity, the more edges). The leaves on the upper left, the left side of the first building from left and the ripples are the first high energy edges the path/ seam encounters, which are also clearly depicted in the energy map. The lower right region shows a cluster of high cumulative energy pixels because of all the ripples, boats, and the reflections.

4. Display Seams

First Horizontal Seam



First Vertical Seam



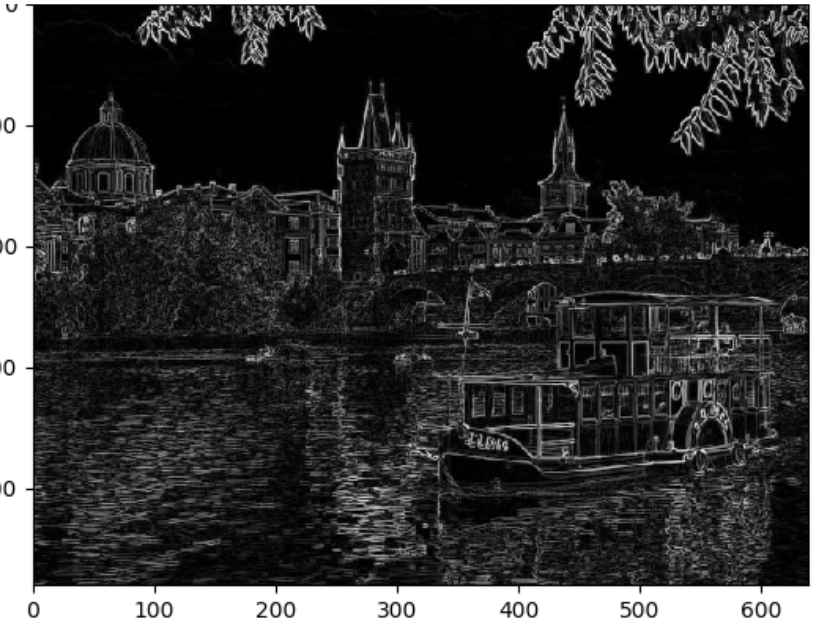
The horizontal seam should be the optimal horizontal seam in this image because from the horizontal cumulative minimum energy map above, we can see that this seam traces through the obviously darkest region in this image, which clearly avoids the cloud, the top of the towers, and the leaves.

The vertical seam should be the optimal vertical seam in this image as well because in the vertical cumulative minimum energy map above, we can see that the darkest region in the bottom of this map is around the 200 mark, which is also where the bottom of the seam is located. From the image, the seam avoid the leaves and the top of the tower, cuts in between the buildings on top of the constant brown pixels. The part where the seam goes in the water is also a part in the water with the least amount of visible ripples due to the lighting in the image.

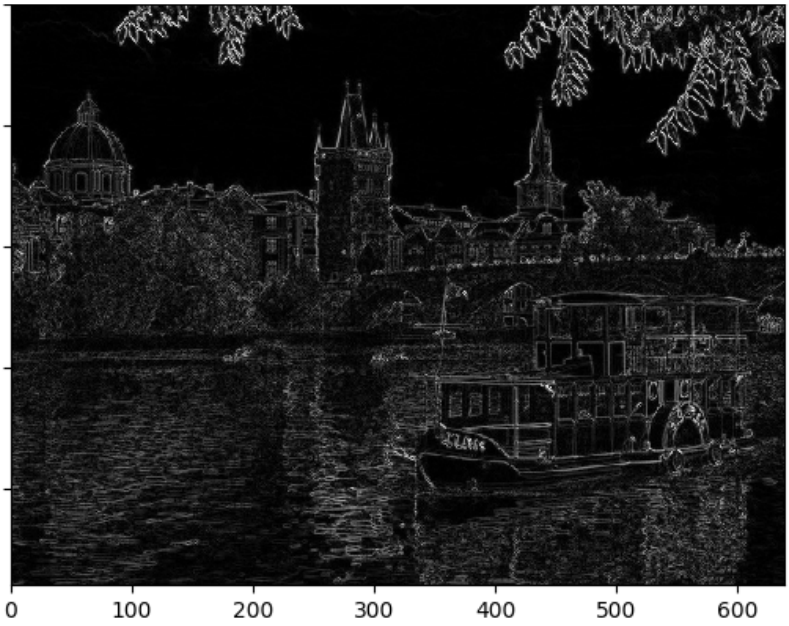
5. Modify Energy Function

a) a simpler derivative filter that just looks at two neighbors in the appropriate direction

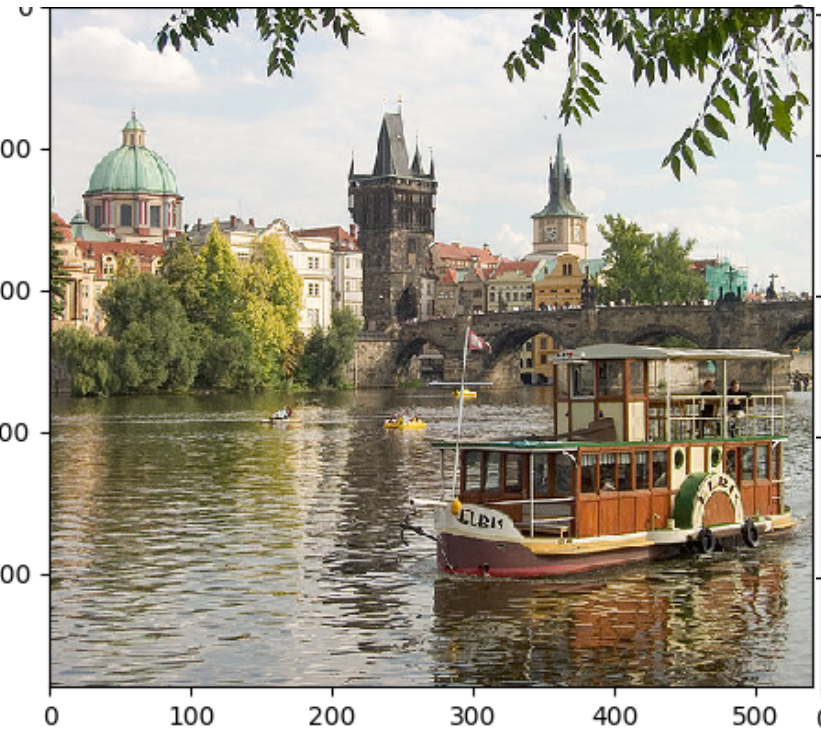
Original Energy Image



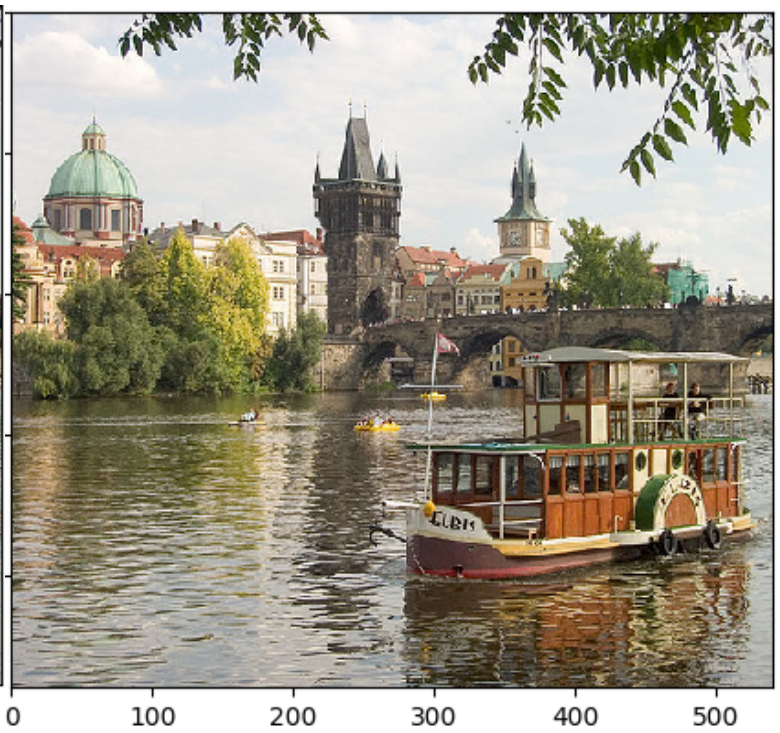
Modified Energy Image



Original Reduced Image



Modified Reduced Image



For this modification, the derivative filter only looks at two neighbors in the desired direction. For example, the x-direction filter used to be [[1, 0, -1], [1, 0, -1], [1, 0, -1]], which also looks at the rows above and below the current row, and for this one, the x-

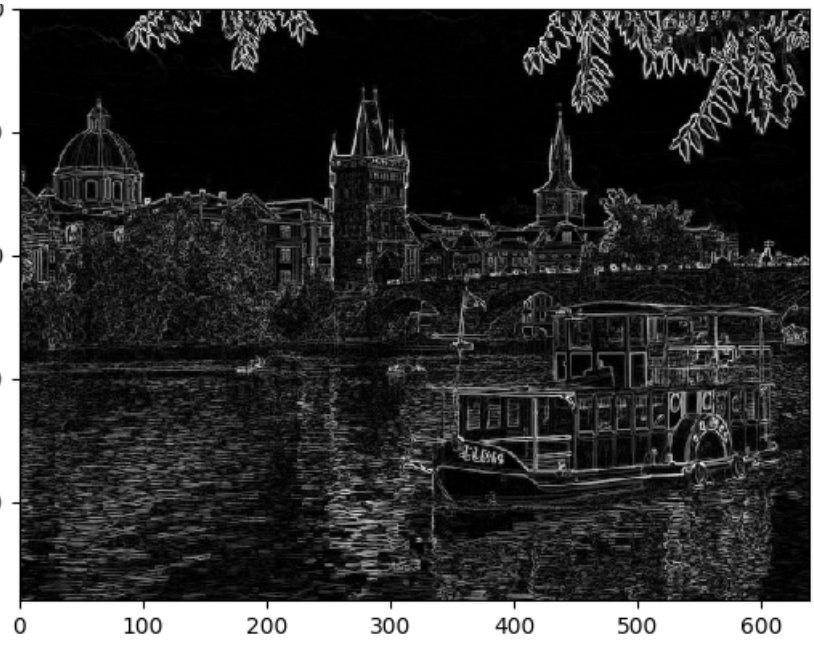
direction filter is [[-1, 0, 1]], which only looks at the two neighbors in the current row. Same idea applies to the y-direction derivative as well.

After this modification is applied, the edges seem a lot more ambiguous and weaker, which makes sense because the modified energy function is supposed to be weaker and assign smaller values to edges. When there is an apparent edge, the original derivative function would take differences from two more rows and sum all of them together to assign to the center pixel; however, the modified function would only take the difference from the current row without summing to any other values, which would reasonably be a smaller value. Furthermore the edges (if zoomed in closer) in the modified energy map also seems more granular than the ones in the original energy map. In the modified energy map, edges appear to be formed by little particles; whereas the edges in the original energy map seem a lot more smoother. This can be explained by the scope by the derivative filters. The original one takes in consideration of all the neighbors, which is 9, surrounding the center pixel, but the modified one only takes in consideration of 2 of the neighbors, which provides a much less scope, thus breaking up the connectivity of the edges.

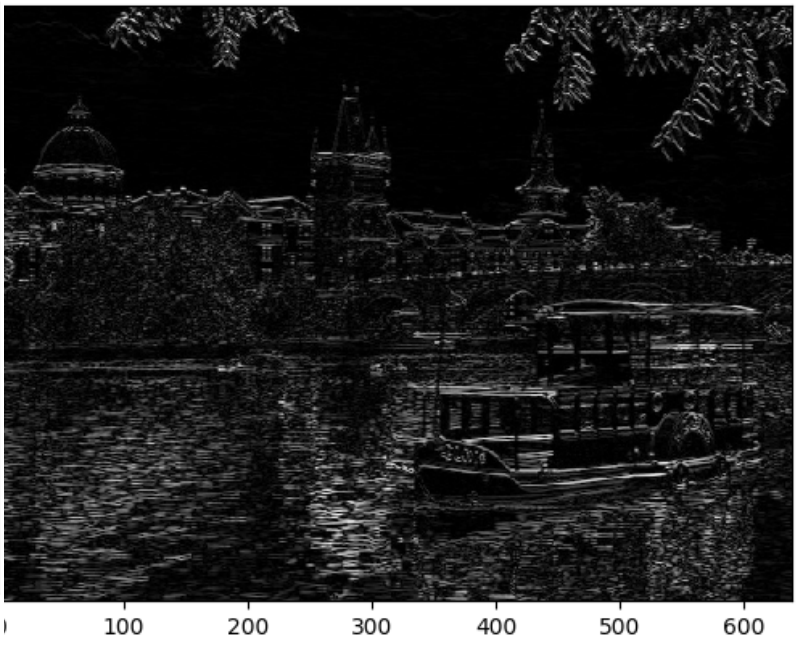
In the final resized image using the modified energy function, the boat's size is not well preserved. This is caused by the clearer and cleaner edges given by the original energy map.

b) use a threshold to cutoff some ambiguous edges in order to clean up the image

Original Energy Image

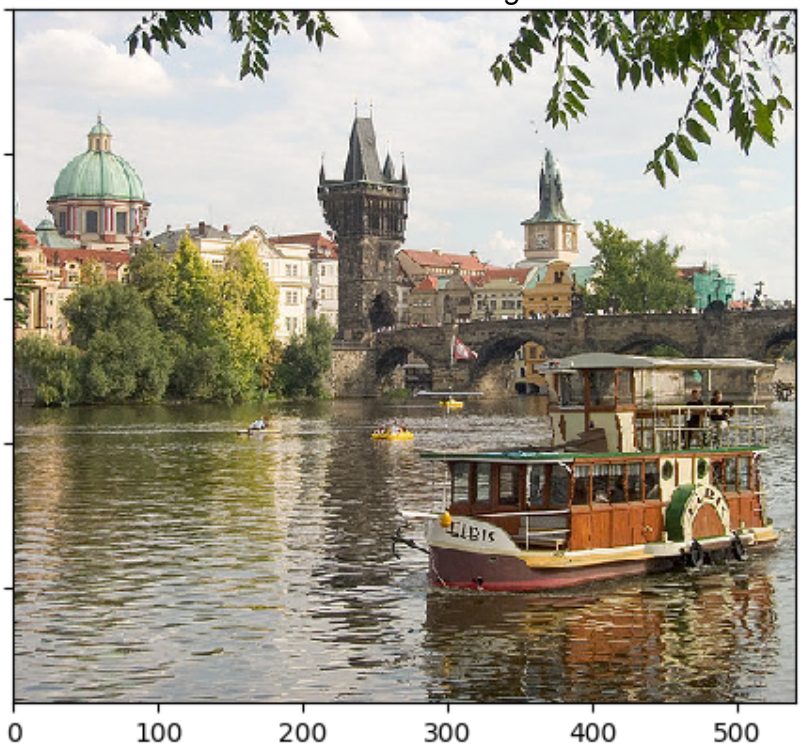


Modified Energy Image



Original Reduced Image

Modified Reduced Image



For this modification, the original derivative filters are kept; however, after the energy map is calculated, the mean of the intensities acts as a threshold to subtract all the values in this energy image. Basically, any pixels with intensity less than this value are not considered and colored black as all the other pixels will get weaker by the amount.

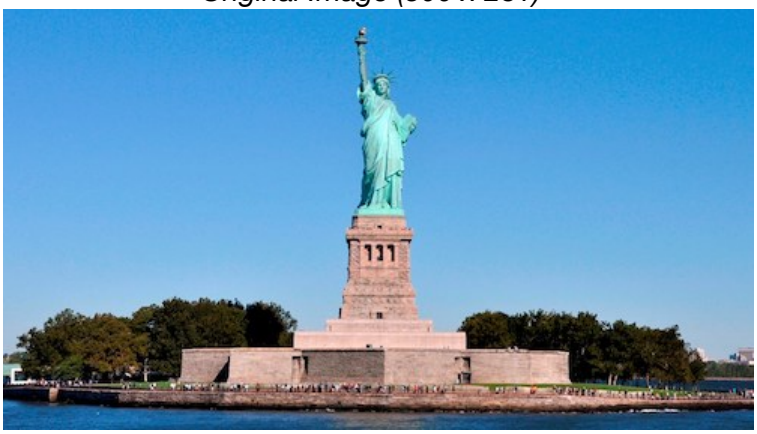
After the threshold is applied, the noise, such as the leaves in the trees in the background, is significantly decreased, but many of the necessary edges are also significantly weaker, such as the top of the towers and some parts of the boat. The ripples don't really have much effect because they are actually perceived as very strong edges due to the lighting and the reflections. The intention of this modification is to remove the weaker edges for cleaner edge detection, but it didn't turn out like expected because due to the lighting and some color contrast, many of the human perceived edges are recognized as weak edges by the algorithm.

The final resized image using the modified energy function has a pretty bad result. The size of the boat isn't very well preserved at all, and the back of the boat is also distorted.

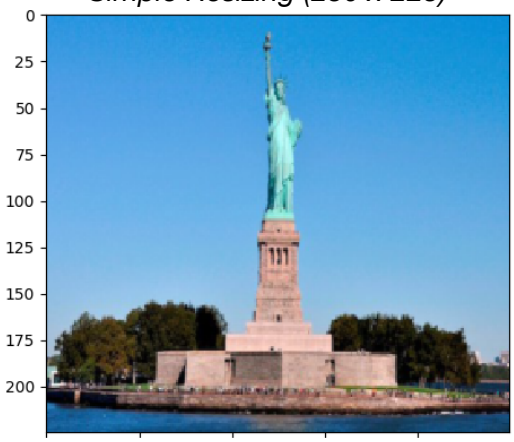
6. Different Images and Operations

a) Landmark with an important feature to preserve

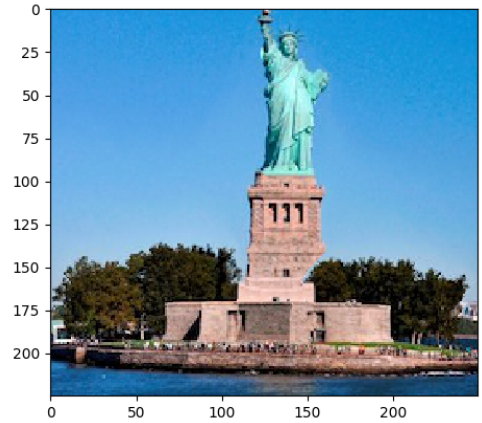
Original Image (500 x 281)



Simple Resizing (250 x 225)

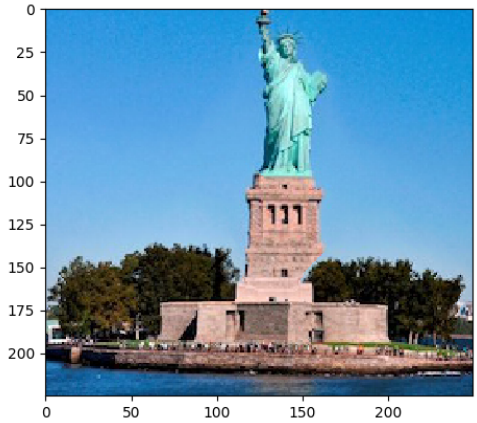


Seams: horizontal, then vertical



Seams: vertical, then horizontal

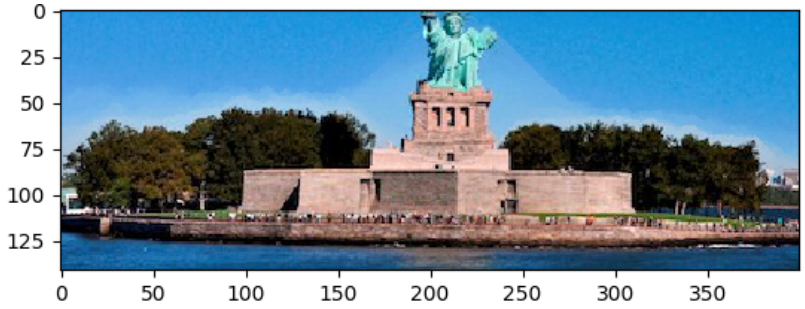
Seams: alternating



Simple Resizing (400 x 141)

Seams: horizontal, then vertical

Seams: vertical, then horizontal



Seams: alternating

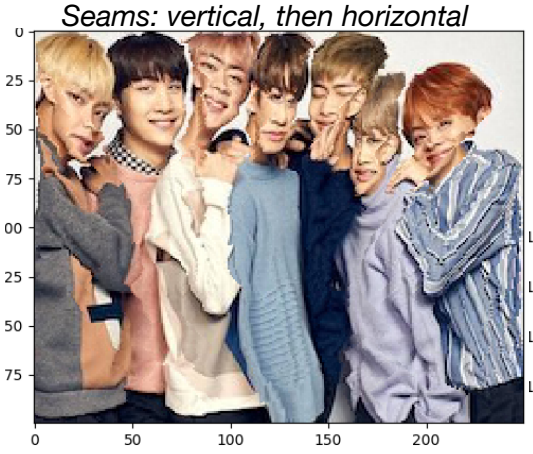
b) Group picture

Original Image (500 x 250)

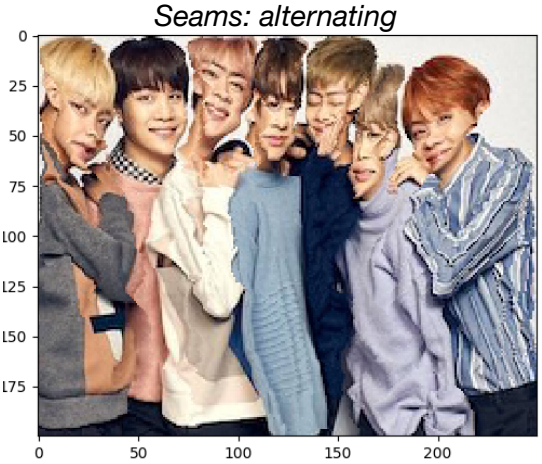
Simple Resizing (250 x 200)

Seams: horizontal, then vertical

Seams: vertical, then horizontal



Seams: alternating



Simple Resizing (400 x 125)

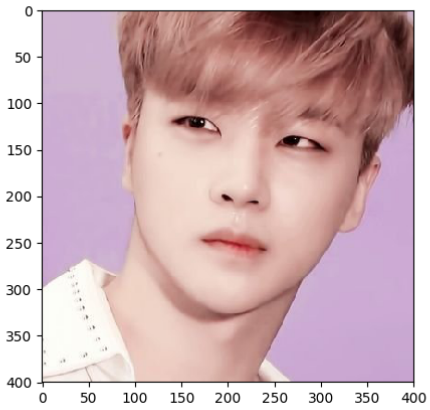
Seams: horizontal, then vertical

Seams: vertical, then horizontal

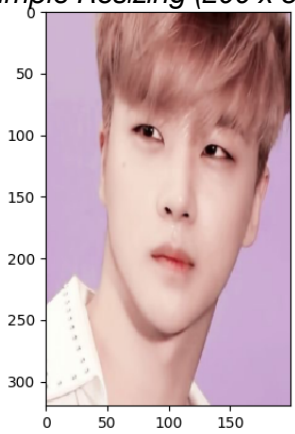
Seams: alternating

c) Face picture

Original Image (400 x 400)



Simple Resizing (200 x 320)



Seams: horizontal, then vertical

Seams: vertical, then horizontal

Seams: alternating

Simple Resizing (320 x 200)

Seams: horizontal, then vertical

Seams: vertical, then horizontal

Seams: alternating

Explanation:
I used three images in total. The first image in an image of statue of liberty, which has one main feature in the image I want to preserve, the statue. The second image is an image of a group of people, and the third one is an image of someone's face. For each of these images, there are two resizing operations. The first one is reducing the width by half and the height by 1/5 of the original height, and the second one is reducing the width by 1/5 of the original width and the height by half. For each of these operations, there are also three seam sequences tested. The first one is all horizontal seams first, and then vertical seams. The second one is all vertical seams first and then horizontal seams. The last one is alternating between horizontal and vertical seams. For all of the images, the different seam sequence doesn't really make any visible differences to the output images.

For the first landmark image, when the image's width is the main one getting reduced, the content-aware resizing does an excellent job at preserving the statue, which stayed nearly the same after resizing. When the images height is the main one getting reduced, the content-aware resizing had to cut out part of the statue's body to reach the size because the intensity contrast of the statue body and the sky is much weaker than the intensity contrast of the status base and the trees, which had many edges/noise.

For the second group image, when the image's width is the main one getting reduced, the content-aware resizing tries to squeeze people closer together, which causes significant deformation to the people in the image because there have to be seams going through people to reach the target image size, which causes distortion in the image. When the image's height is the main one getting reduced, the content aware resizing has to have seams that go through people even more because the people's heads are already very close to the top of the frame. Having almost all the seams going through people, most seams actually choose to go through clothes rather than face because clothes generally have less edges than the faces. Therefore, people's heads are almost staying in the same size, but the body is extremely shortened.

For the third face image, when the image's width is the main one getting reduced, the content-aware resizing tries to squeeze all the facial features to the middle horizontally because it is recognizing all the facial features as the features that need to be preserved. When the image's height is the main one getting reduced, for the same reason, the content-aware resizing tries to squeeze all the facial features to the middle vertically. Whatever number of seams are done, this one always tries to avoid the facial features, which perfectly makes sense.

Extra Credit:

1. Greedy solution for finding optimal seam (run *python2 greedySolution.py* can reproduce the answers)

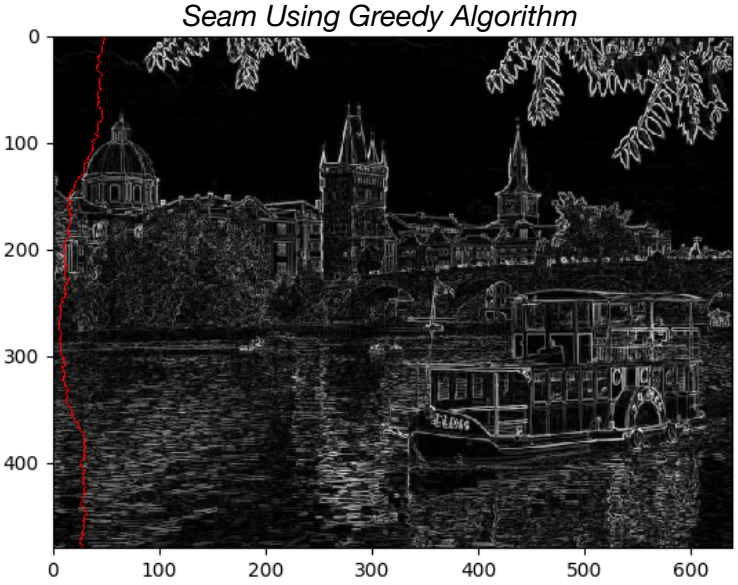
The greedy algorithm basically starts from the least intense pixel from the first row or column and keep going down to the next least intense pixel that are reachable from the current pixel.

Reduce Width

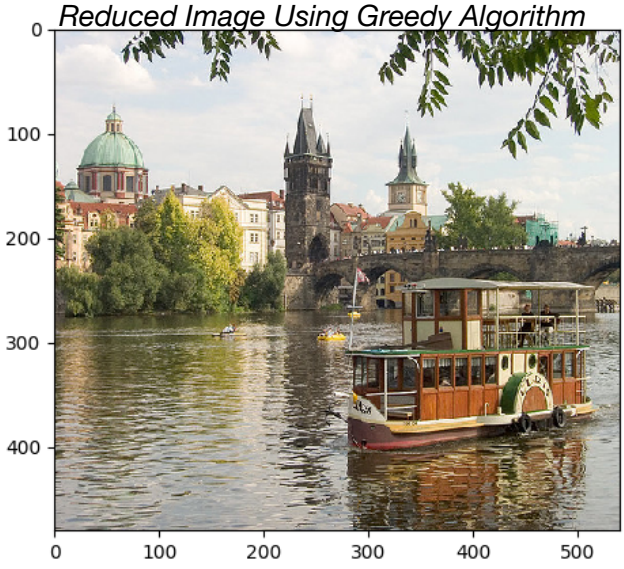
Seam Using Dynamic Programming

Reduced Image Using Dynamic Programming

Seam Using Greedy Algorithm



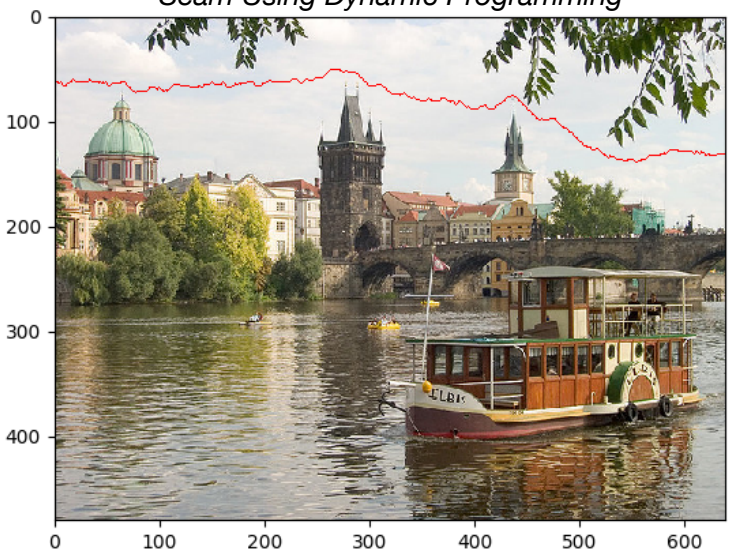
Reduced Image Using Greedy Algorithm



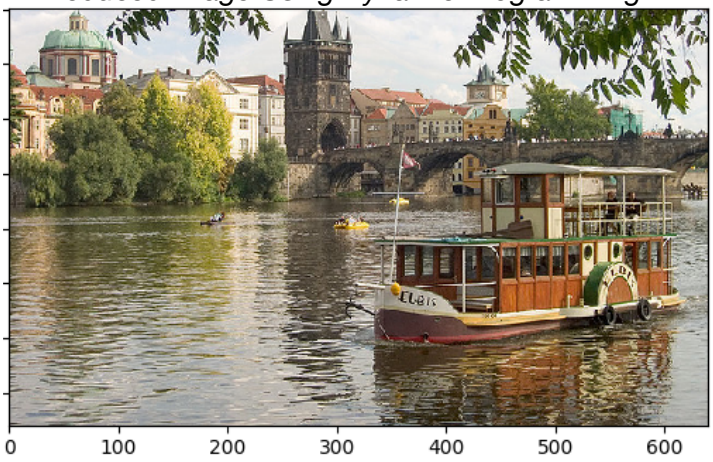
The seam using greedy algorithm might be able to start from the most optimal pixel but since it has no knowledge about the entire image, it might run into a cluster of higher pixels and can't get out. The seam in the '*Seam Using Greedy Algorithm*' shows that it started from the most optimal point, but as it goes down more it has to go through all the higher intensity pixels because they are the only ones reachable at that point. The comparison of the reduced images also shows that the one using dynamic programming is way better because the one using greedy algorithm has the head of the boat distorted

Reduced Height

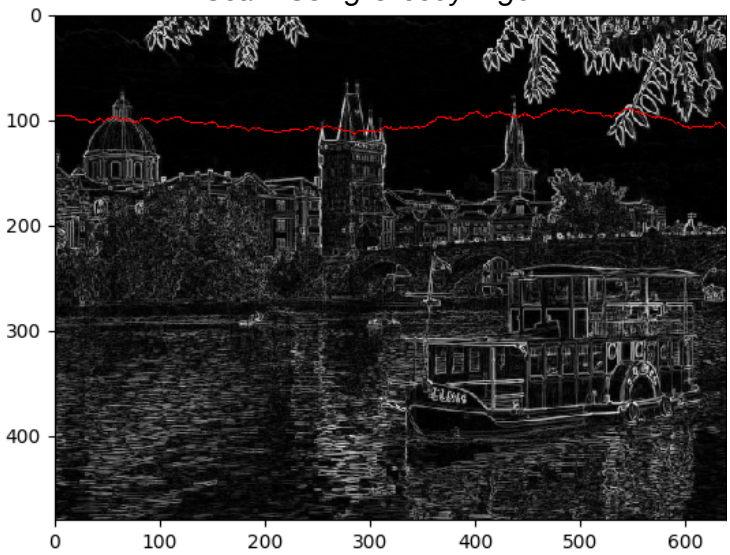
Seam Using Dynamic Programming



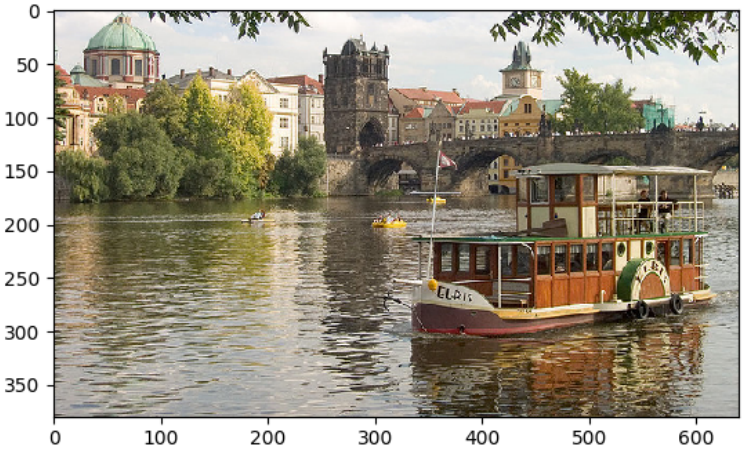
Reduced Image Using Dynamic Programming



Seam Using Greedy Algorithm



Reduced Image Using Greedy Algorithm



The seam using greedy algorithm is again starting from the least intense pixel, but it is obvious that, when the seam encounters the towers and the leaves, since it has no prior knowledge about them, it is to late for it to go around the, so it has to just cut through them, which results in a suboptimal solution. Comparing this with the seam using dynamic programming, since the seam using dynamic programming has knowledge about what it will encounter, it goes around the towers and the leaves. The reduced image using greedy algorithm actually removes part of the brown tower in the middle, which wouldn't be ideal at all. Thus the dynamic programming is a better approach to calculate the seam than the greedy approach.

Other Files:

main.py

- run '*python2 main.py*' to recreate the part of the answer sheet including the energy image, cumulative minimum energy map and displaying seams

run '*python2 SeamCarvingReduceWidth.py*' and '*python2 SeamCarvingReduceHeight.py*' to recreate the answers for reducing width and height by 100 pixels for two images

modified_energy_function.py

- run '*python2 modified_energy_function.py*' to recreate the part of the answer sheet relating to modifying energy function

otherImages.py

- edit the 'filename' variable in the file and run '*python2 otherImages.py*' to recreate the part of the answer sheet relating to using different images, number of seams and seam sequence

greedySolution.py

- run *python2 greedySolution.py* to recreate the part relating to extra credit

progressBar.py (all usage commented out)

- helps to print a progress bar in command line as the warping process is pretty slow, so the user knows how the process is going
- very buggy when running on Mac small-size terminal, but seems fine when running in PyCharm terminal or maximized screen Mac terminal

/more images

- a folder that contains all the new images used, and all the output images from the algorithms