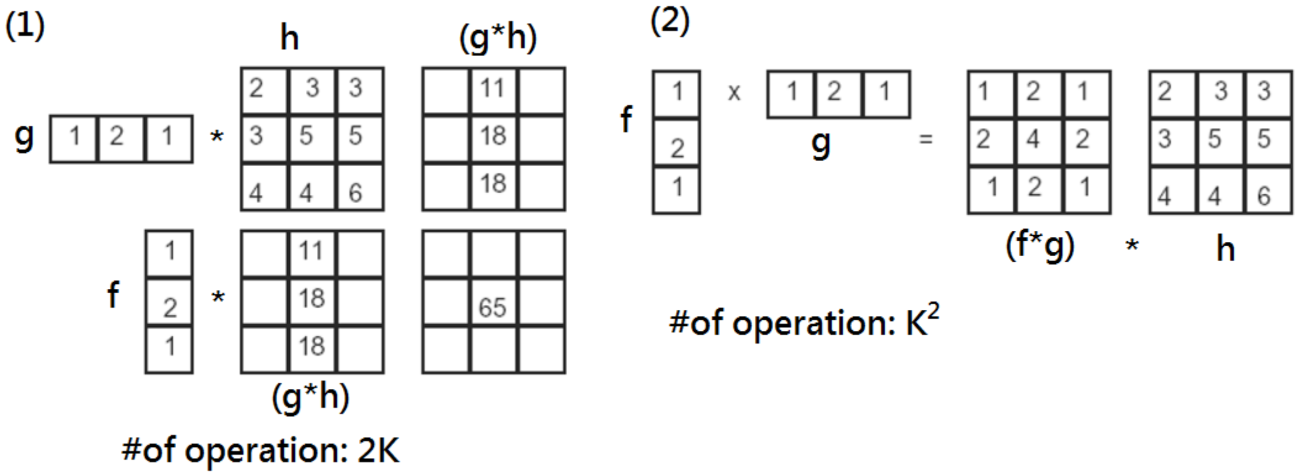
ECS189G Computer Vision Problem Set 1

WEI-CHIH CHEN

ID: 912448776

I. Short programming example

1. Associative property is (f*g)*h=f*(g*h). We can compute smaller matrices first and use the result to compute larger matrices to reduce number of operations. In lecture 2 page 63 example, we compute g*h first and then compute f*(g*h). The number of computation is 2K. But if we compute f*g first and then (f*g)*h, the number of computation is K^2.



2. [0 1 1 1 1 1 1 1]
3. To derive f", we can use convolution which f"=f'*f'. To do convolution, flipped f' which = [0 1/2 1 -1/2 0] first and then convolved by following formula: $y[n] = f[n] * g[m] = \sum_{m=0}^{M-1} f[n-m]g[m]$. So the result of convolution is f"=[1/4 0 -1/2 0 1/4].

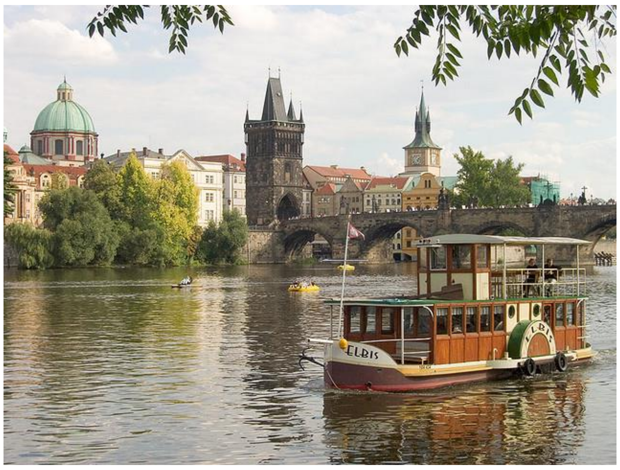
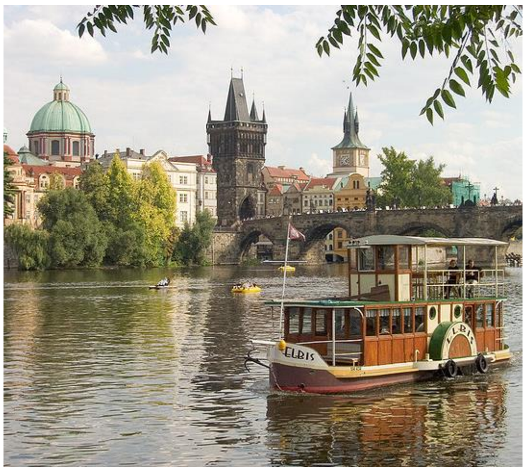
4. We can use Additive Gaussian noise to simulate thermal noise. But it is normal distribution. It's hard for as to simulate extremely energy such as Salt and Pepper Noise. So it can only simulate some certain type of noise.

5. We assume that flaws of the assembly of a part is (1) different shape: shape of that part is not exactly the same as other; and (2) material of that part should keep the same quality, if the quality changes, we assume the texture or color change.

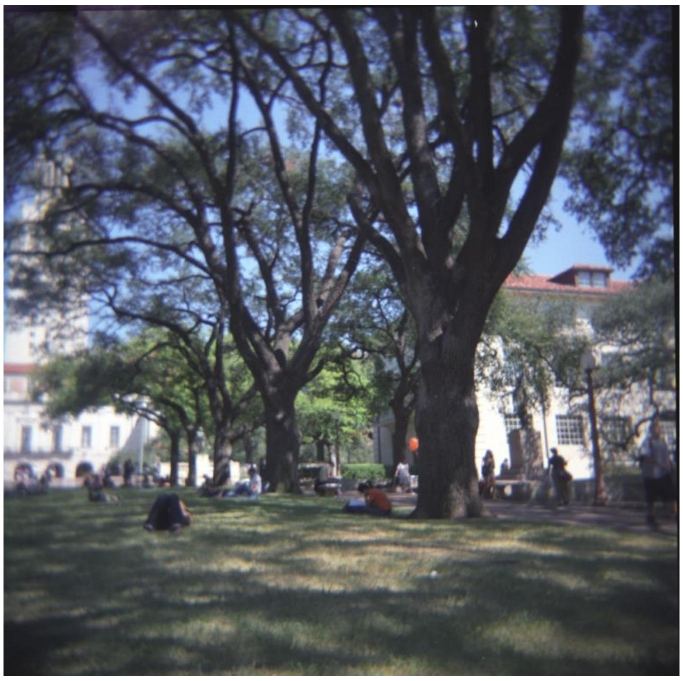
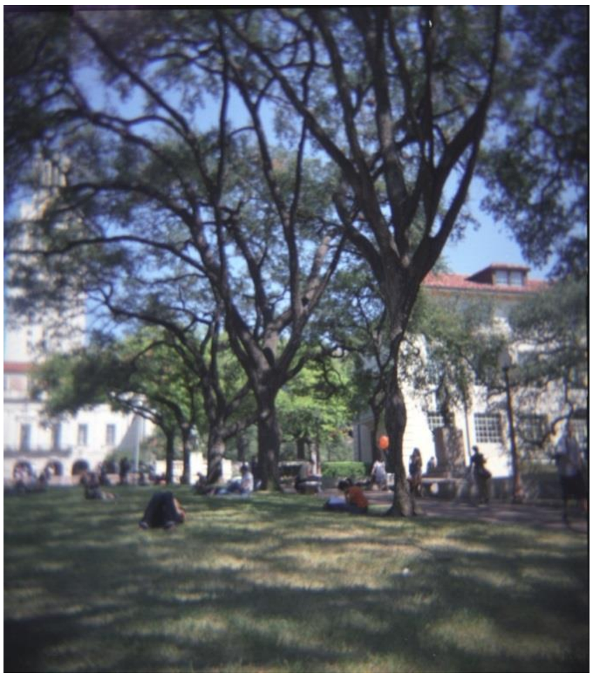
   If we fix camera on a same position, the size of part should be exactly the same for each part. Therefore, in (1), we use canny edge detector to detect target part, and using chamfer distance as a passing criteria. We have idea shape contour as standard shape and compare it with target part from camera. Not only $D_{chamfer}$ value should be small, but it should be small enough to distinguish some abnormal shape.
   For the (2) is similar to compare color and texture of the standard one with target part. But the criteria for texture becomes D(a,b) (d-dimensional features).

II. Programming problem: content-aware image resizing

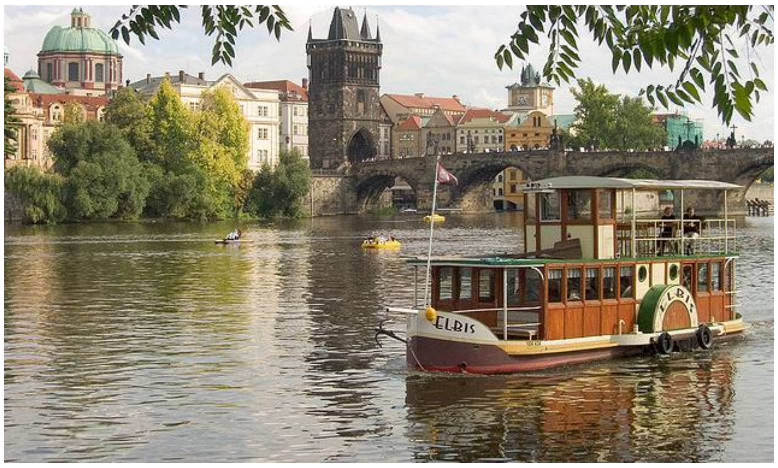
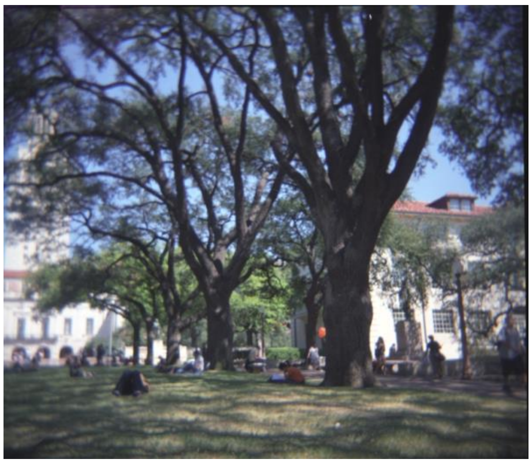
1. Original inputSeamCarvingPrague (480X640) outputReduceWidthPrague (480X540)

Original inputSeamCarvingMall(769X775) outputReduceWidthMall(769X675)

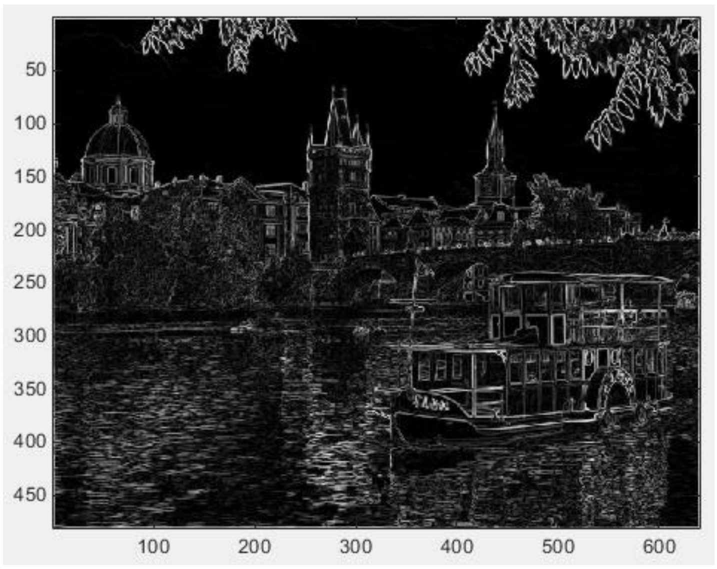



2. outputReduceHeightPrague(380X640) outputReduceHeightMall(669X775)

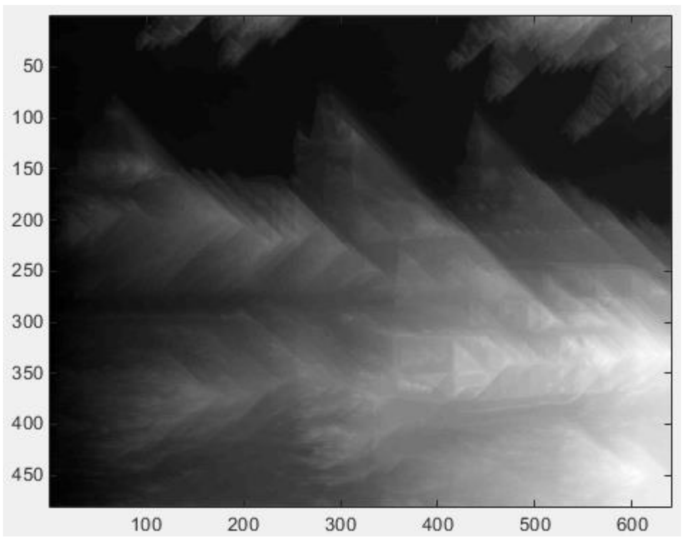
3.

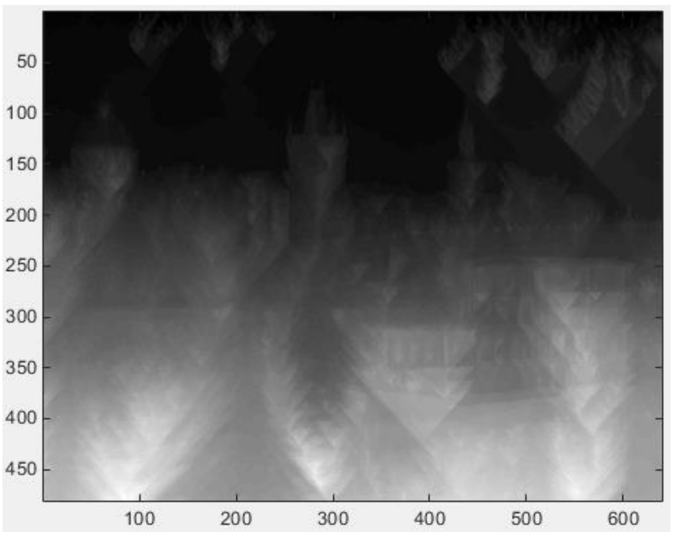
(a) Energy function



This graphic obtains by three processes:
(1) transferring to gray image;
(2) passing though filter ”sobel”; and
(3) calculating edge strength.
In second step, we can create gradient in x and y direction which calculate differences in these two direction. And if there is an edge in the picture, there has relative high energy difference. So in third step, sqrt(dx^2+dy^2) outline edges in the graph which is shown as left picture.

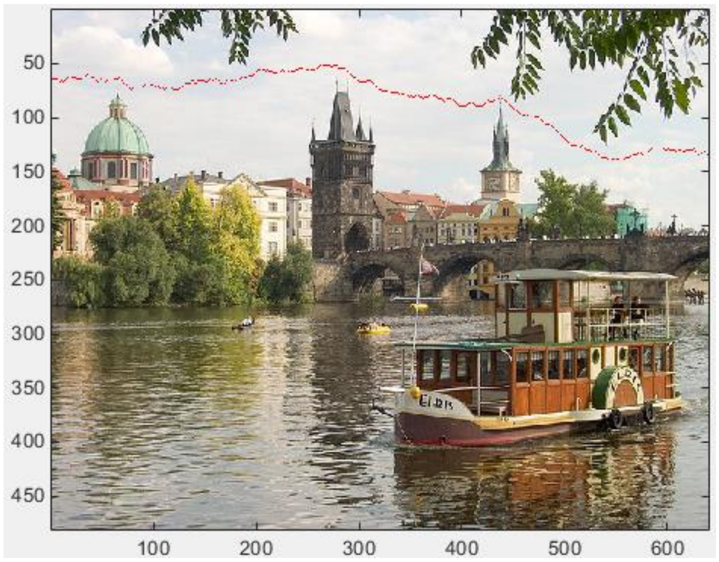
(b) Cumulative energy map(horizontal) (vertical)





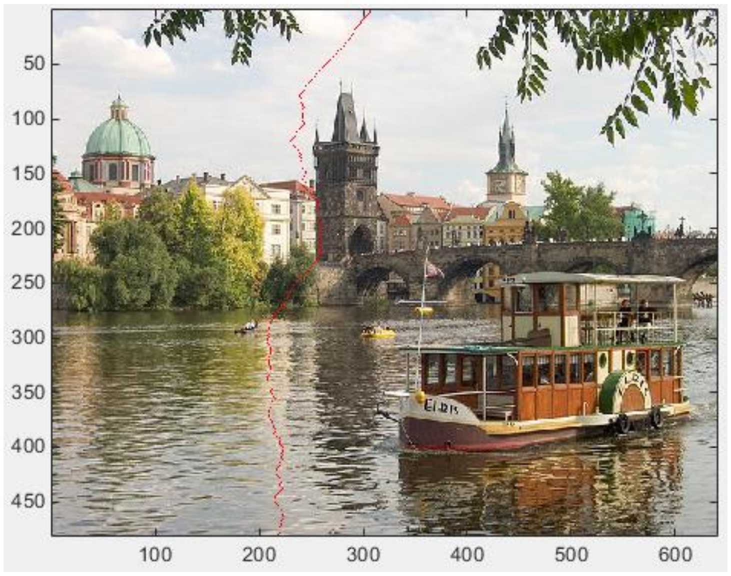
Horizontal:
This graph is cumulative energy in horizontal (left to right direction) by dynamic programming that we learned in class. So we can tell that buildings’ shadow is shift horizontally.
Vertical:
Since this graph is cumulative energy in vertical direction, we can see limited leaves shadow on top of the picture because of limited edges of leaves above. And more energy increases by more edges in the middle and bottom of the graph.

4. (a) horizontal seam



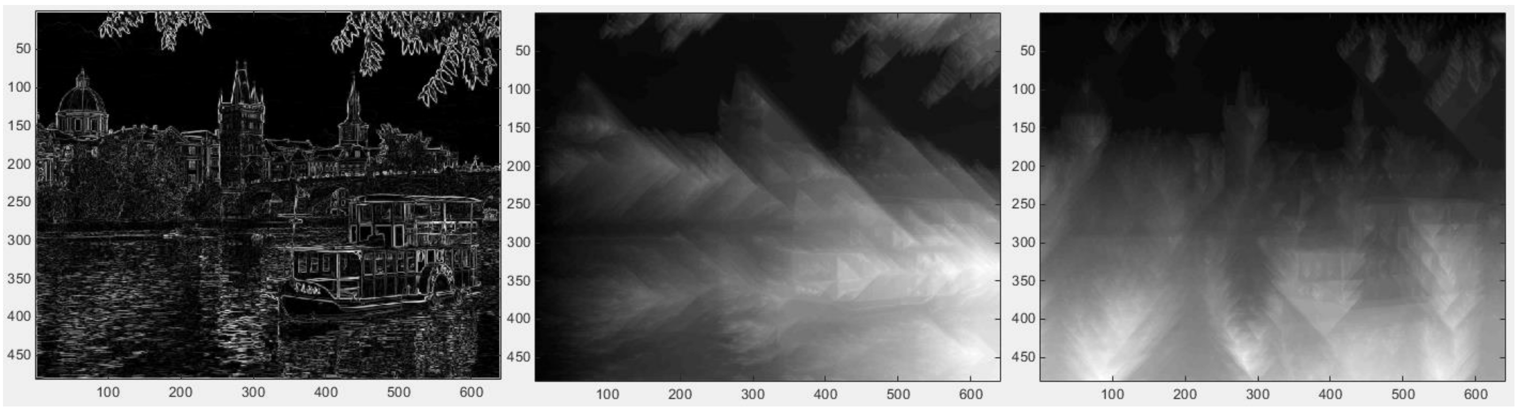
The black region of horizontal cumulative energy map above means the lowest energy. And we select a seam from the least energy in horizontal or in vertical which means least noticeable by human. In horizontal seam, it is easy to distinguish that the red line region is the least energy in the cumulative energy map.

(b) vertical seam



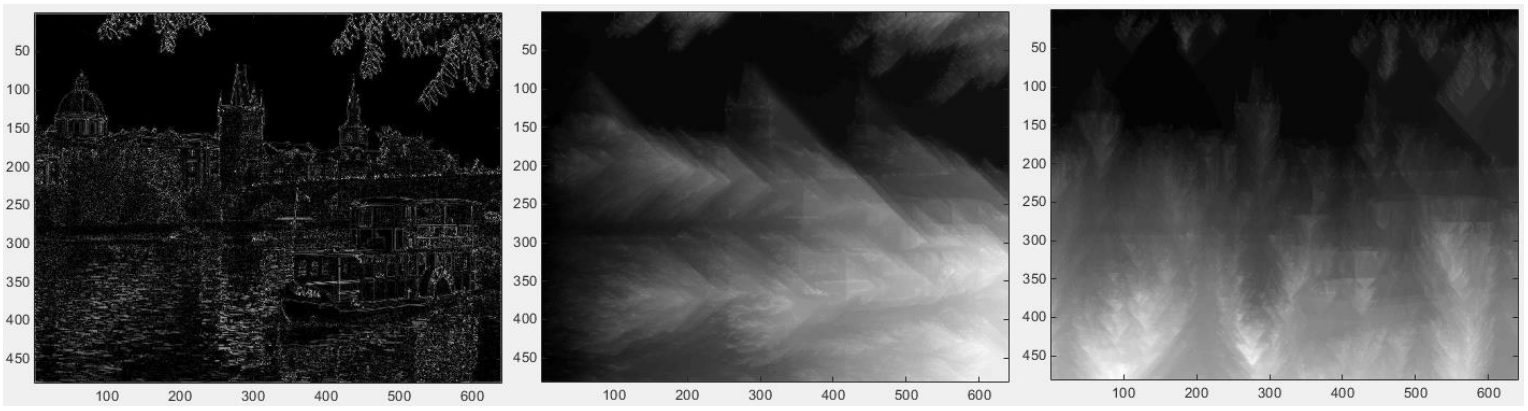
In vertical seam, it’s not easily to visually tell which vertical seam is the least energy. In cumulative energy map function, we find minimum energy from the bottom of cumulative energy map and trace back to top of picture.

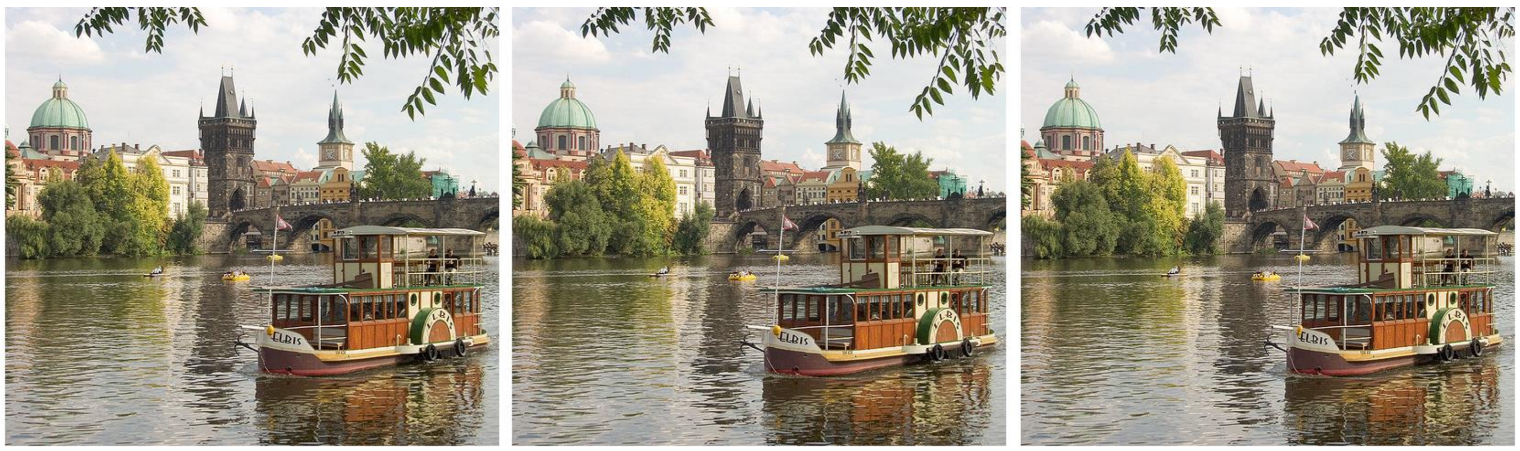
5. (a)



This one is change parameter of fspecial in energyImage. Original is sobel. In this one is **Prewitt**. This is some different matrix to calculate derivatives of image.

(b) This one change parameter of fspecial to **laplacian**.

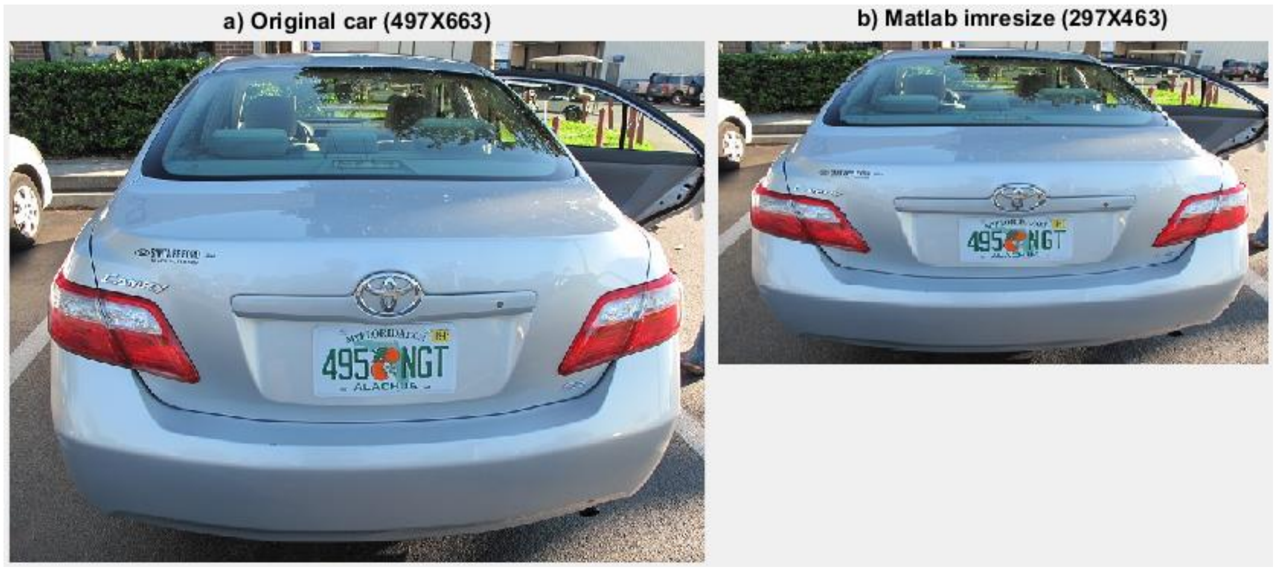


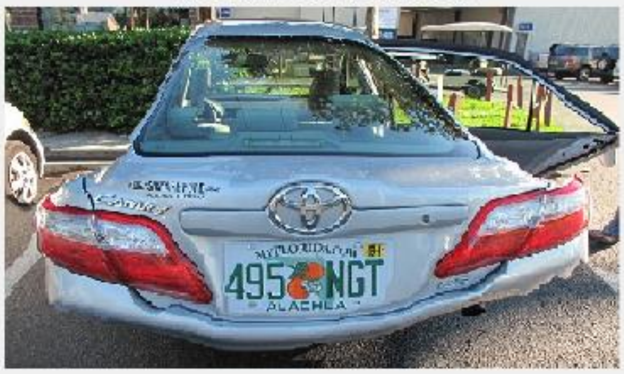


This is the result of reduce width by 100 with three different filter parameters which are **sobel**, **prewitt** and **laplacian** from left to right. If we merge them into one graph, we may find that outputs of sobel and prewitt are pretty similar although there are some different by selecting seams. Laplacian can tell little different than other two. The boat and buildings are smaller than other two. This may be caused by unsuitable gradient magnitude for edge.

6.

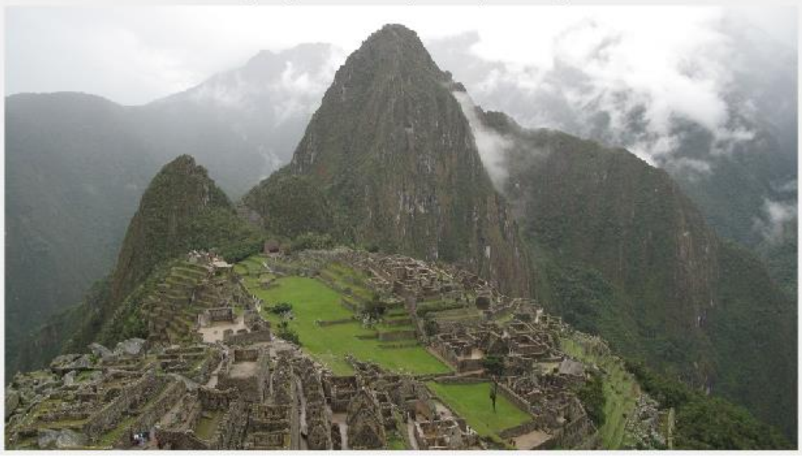
(1)

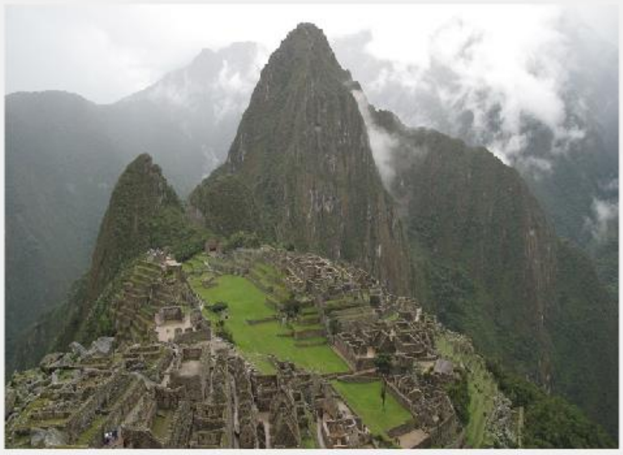


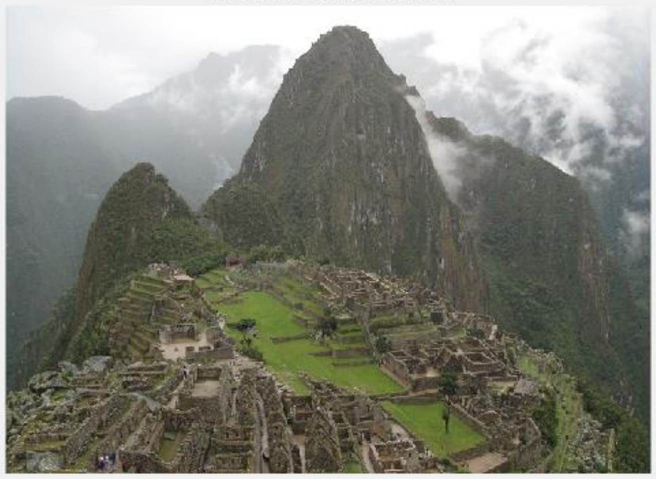

c) Seam resize (297X463)

In this picture, I resized 200 pixels on width and 200 pixels on height. But, overall my resized image got a bad outcome. Since words, like a car plate and “CAMRY” have most edges, large amount of energy is on these parts of the cumulative energy. Therefore, comparing to other part, words remain while other parts changing a lot even causing deform.

(2)

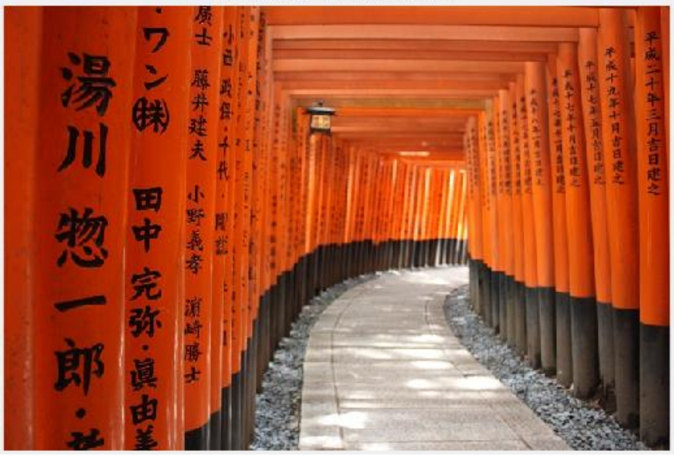

a) Original machu picchu (496X884)

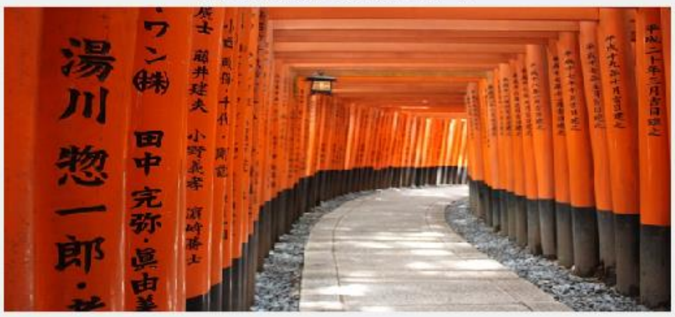

b) Matlab imresize (496X684)

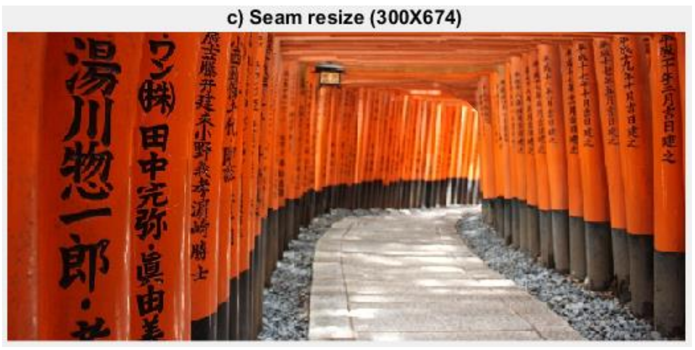

c) Seam resize (496X684)

I tried to reduce width by 200 pixels in original image. In this picture, right side of the original macho picchu picture can be removed. But in b) Matlab imresize, it just compressed the picture horizontally. So we can easily tell that main mountain is compressed in b) while main mountain remains unchanging in c).

(3)


a) Original Torii (450X674)


b) Matlab imresize (300X674)



I tried to reduce height by 150 pixels in original image. In b) the image is compressed vertically. However, in c) it removed empty space between words and remain words unchanging (words have most edges and therefore have most energy in cumulative energy map).

NOTE: All three of images was shoot by me.

III. Extra credit

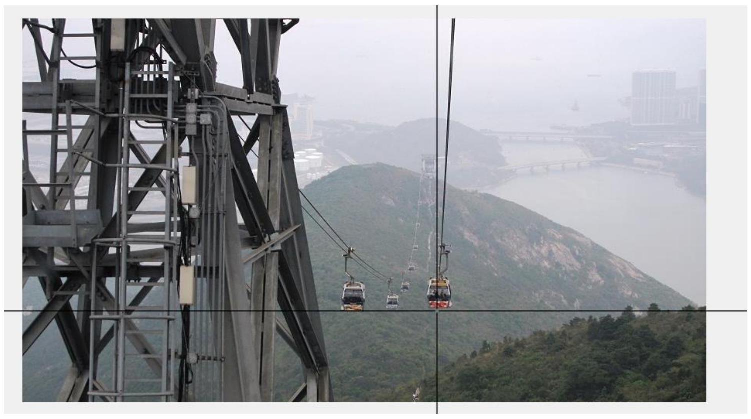
(1) I propose two methods to do this: Reduce select region directly and Set energy to zero.

Reduce select region directly:

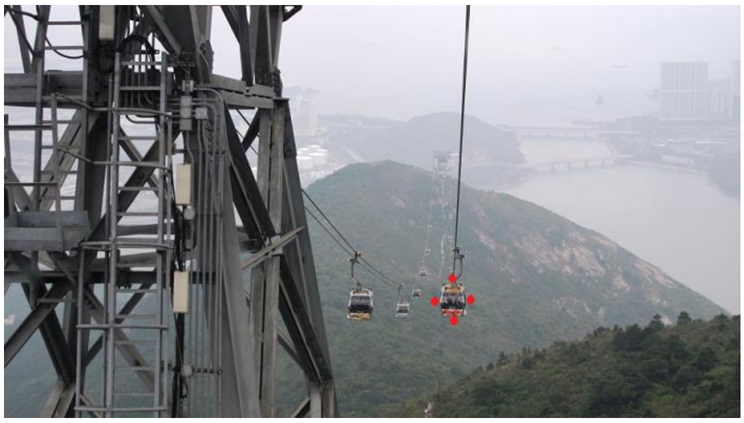
In this question, we should separate it into three parts:

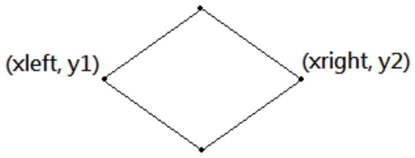
a. Select object: If we try to select an object to remove, we can use `ginput` function in matlab which is shown below. If n=4, then after we clicking four points by mouse to outline an object on the image, matlab stores four coordinates to [x,y]. If we want to click more points, we can change n or use `impoly` function.

```
figure;
imshow(image);
[x,y] = ginput(n);
```



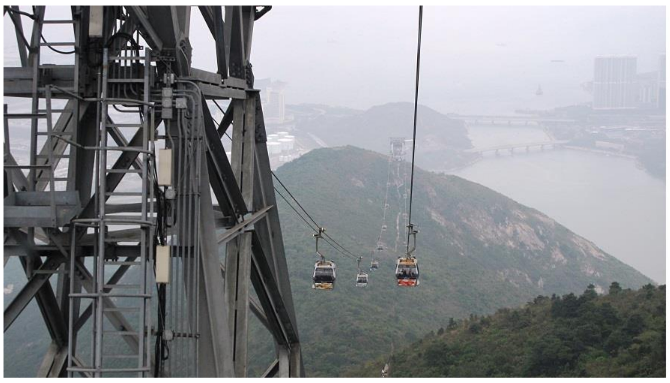
b. Calculate vertical seam: My strategy to remove seam is that remove all the seams passing though the widest distance among four points. In this graph, we let x-coordinate of left most point is xleft, and right most point is xright. And I removed all seam passing through (xleft, y1) to (xright, y1). That means we force seam passing through this xleft-xright line.

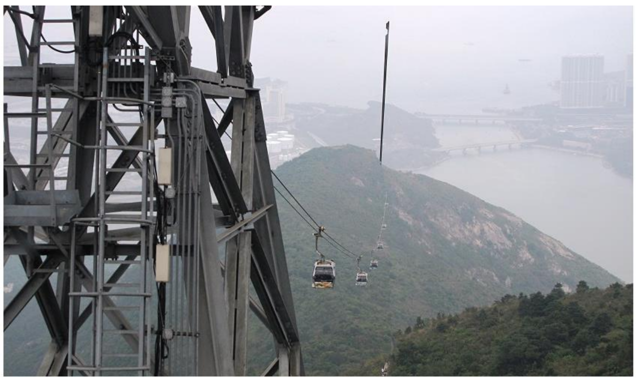




c. New algorithm: Since original seam is to remove a seam with lowest energy in the bottom of image, we should change it to y1 axis. I create another `OPTIONAL1_find_optimal_vertical_seam` function to select a seam passing to a specific point. And I also create another `OPTIONAL1_reduce_width` function to reduce one pixel corresponding to selected vertical seam. In the end, removed it xright minus xleft times under same y1 axis, we can remove all the point passing through (xleft, y1) to (xright, y1).

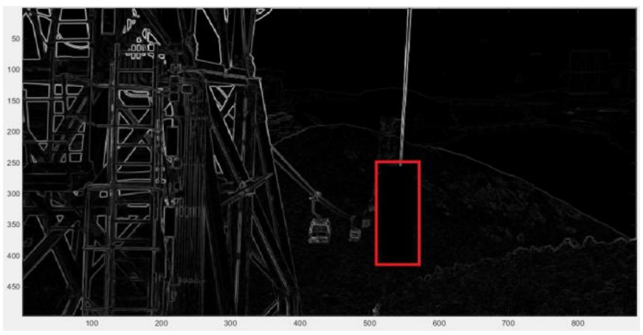
In conclusion, we can remove an object that we selecting. Below left is the original image and right is removed cable car. Since we removed several seams, so output image size (496X841) is less than original one (496X884). However, this algorithm is assuming that all the selected seams can cover the object completely. If it's a tall object, we may not cover some part of the object. Then we need to make sure all the pixels inside polygons should be removed. It needs some conditions and loop to achieve that. The concept is the same, so I do not do this part here.

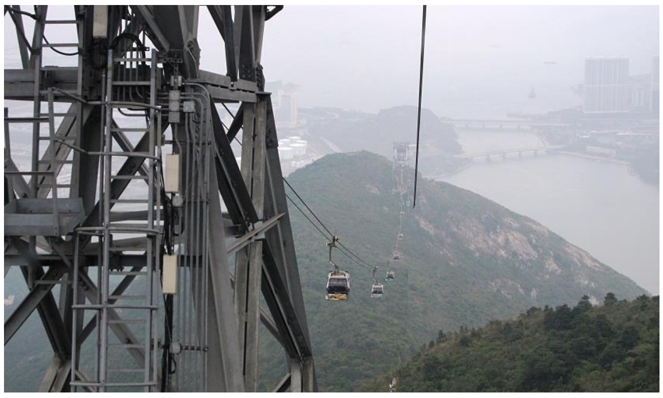




<u>Set energy to zero:</u>

There is another way to remove select object that is **to set energy to zero of marked region**. However, since there may be some similar low energy in image, it may not complete reduce the marked region actually. Therefore, I increased 20% area of x axis and 90% area of y axis of marked region. Since energy cumulate vertically, setting more y axis energy to zero causes more change to be lowest cumulative energy in image. **Comparing two methods, to set energy to zero seems get better result.**





[NOTE] Code of Reduce select region directly: OPTIONAL1.m

Code of Set energy to zero: OPTIONAL12.m