CS 6476
PS 1

1.1 The associative property is that f*(g*h) = (f*g)*h. For images, we can think of filters as f, g and image as h. Sometimes it will be more efficient to perform 1D convolution twice instead of 2D convolution. First we convolve with the input and M*1 kernel in vertical direction, and then convolve again with the previous result and N*1 kernel in horizontal direction. (But we should do computation depending on the case.) Let f = $[1\ 2\ 3]^T$, g= [3 2 1] and h =, `[[1, 2, 3], [1, 4, 1], [2, 4, 3]]` then f*g = `[[3, 2, 1], [6, 4, 2], [9, 6, 3]]`, and (f*g)*h = 85, which involves two 3x3 matrices. On the other hand, if we computer g *h first: `[[10], [12], [17]]`: this is only 3x1 matrix, and then f*(g*h) = 85. Clearly the number of operations are saved to a great extent.

1.2 [0 1 1 1 1 1 1 1]. Only the first pixel has no foreground pixel in its neighbors.

1.3 The second derivative can be obtained by applying the first derivative twice to the original picture: I' = I * f', for image I and first derivative f', then I'' = (I * f') * f'. By the associative law it is equivalent to I * (f' * f') = I * f''. So we can calculate f'' by convolving f' with itself but flipped: f'' = [0 1/4 0 -1/2 0 1/4 0].

1.4 (1) Increase σ for Gaussian blur filter to smooth out the image and reduce the fine lines. (2) Increase the value for lower/higher threshold. (3) We can try opening on the result image, if we don't directly adjust the algorithm.

1.5 Additive Gaussian noise only draws noises from normal distribution. In the cases where we need extreme noises such as salt and pepper, additive Gaussian noise cannot be applied. So it can only generate some certain type of noise (with some mean and standard deviation).

1.6 Assume that the camera is installed at a fixed position above the conveyor belt, we can (1) detect the shape of the parts. All parts should have the exact shape when they are being transferred on the belt. If not, then we know that a particular part is poorly manufactured. First we use the canny edge detector to extract the contours of the manufactured parts and a standard part. Then we use Chamfer matching algorithm: we have the distances and the contour from the standard part, we will compare them with summed distances of each contour of the manufactured parts. The distance should be small so that we know they are the same shape; otherwise we may have a bad part.

(2) We can also detect the texture of the parts. The process is similar to (1): we will compare texture of the standard part contour with the captured part contour. However, in this case we will be comparing D(a, b) for d-dimensional features instead of 1-d distance in (1).

2.1

(Reduced Width by 100 pixel, Output 540x480)

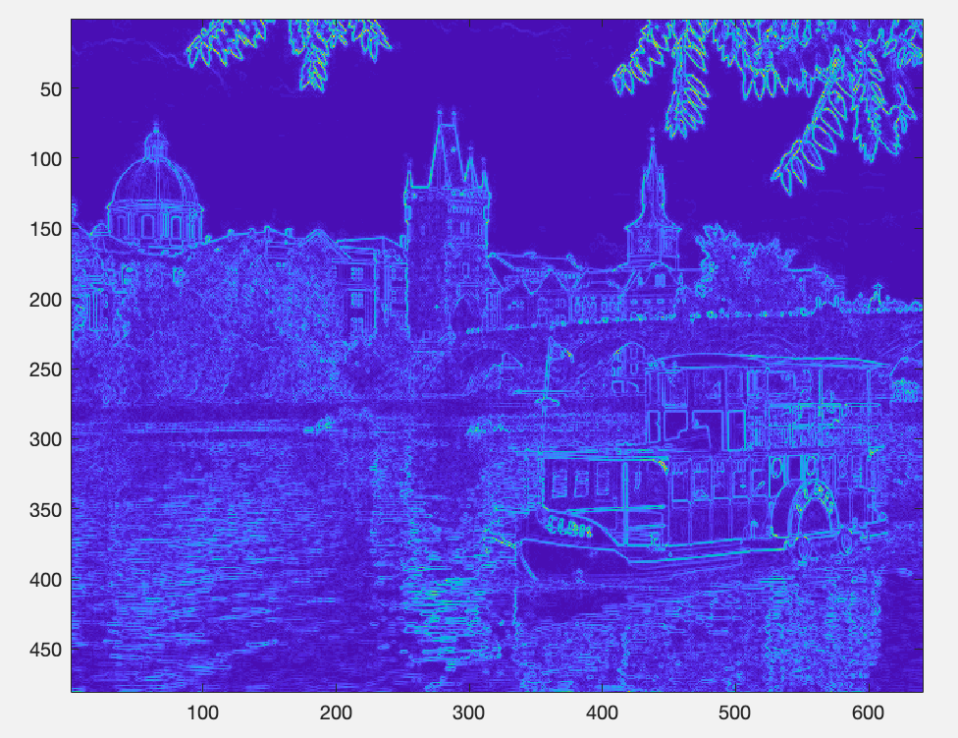
(Reduced Width by 100 pixel, Output 675x769)

2.2

(Reduced Height by 100 Output 640x380)

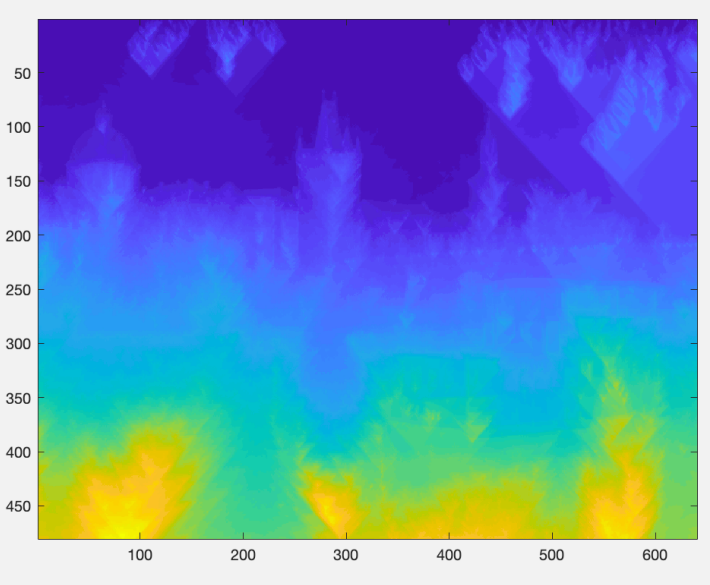
(Reduced Height by 100 pixel, output 775x669)

2.3 (a) Energy Function

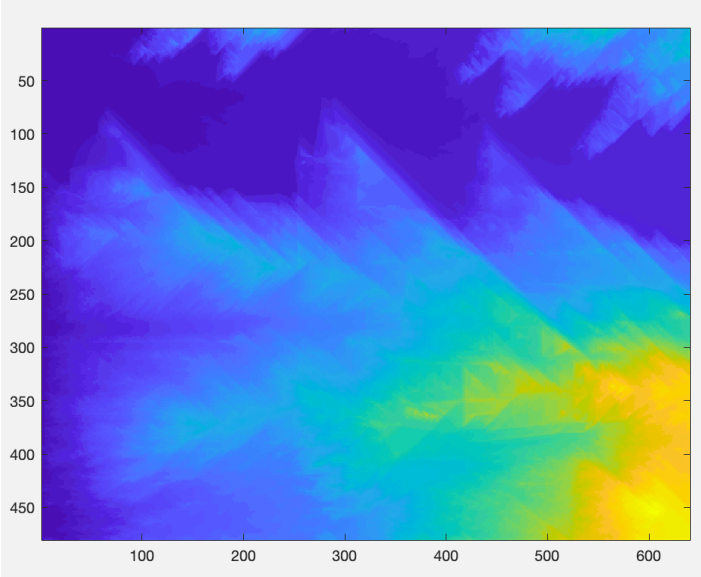


Explanation: The energy map looks like the edge/contour of the image. It is because the “energy” we calculated is actually gradient and by definition it captures edge information, and so it looks like the result of edge detection. We calculate the gradient strength using d/dx and d/dx, so the orientation of the energy does not simply point horizontally or vertically, instead it shows strengths in all directions. The brighter the “edge” looks like, the higher the energy is around that edge (contrast, sudden change in contour, etc.), whereas darker regions indicates smoother area: the sky has no outstanding edges in it.

2.3(b)



Vertical



Horizontal

Explanation: We are calculating the cumulative minimum energy from top to bottom (vertically) or from left to right (horizontally). Therefore, the more to the left/top, the darker the area – because cumulative minimum energy is relatively low, and the more to the bottom/right, the brighter the area is. And also by examining this energy map, we can see bright areas with higher energy and so those are where the algorithm will try to avoid when seaming the image.

4. original image:

Original Image

(a) First selected horizontal seam

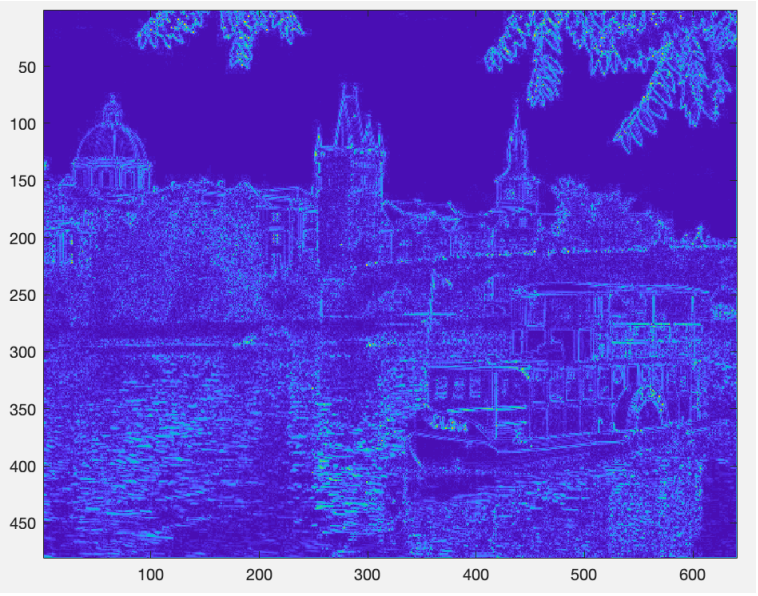
(b) First selected vertical seam

Explanation:
For the first horizontal seam, it is somewhat obvious by just observing it even if we don't know how seaming works: it is the path with least changes. Along that line, the sky is generally blue/white and there are no other objects. If we look at the energy map in 3(a), we can see that the sky area is indeed the place with lower energy, and thus the first line across that area would the optimal choice.

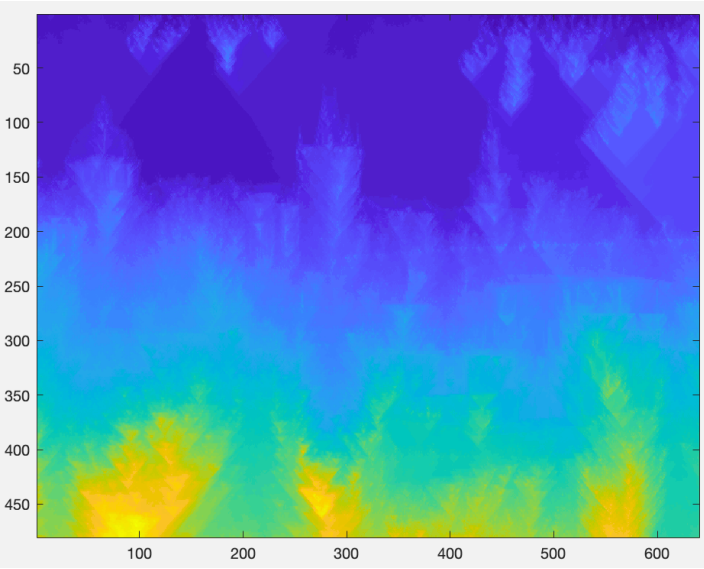
For the vertical seam, we see that there's hardly a place with no or minimum changes in edges. And thus it is not that obvious for us to observe the result. However, referring to the energy map, we can pick out a least obvious path, crossing the waves and between buildings.

5. I applied Laplacian filter to calculate the energy map.

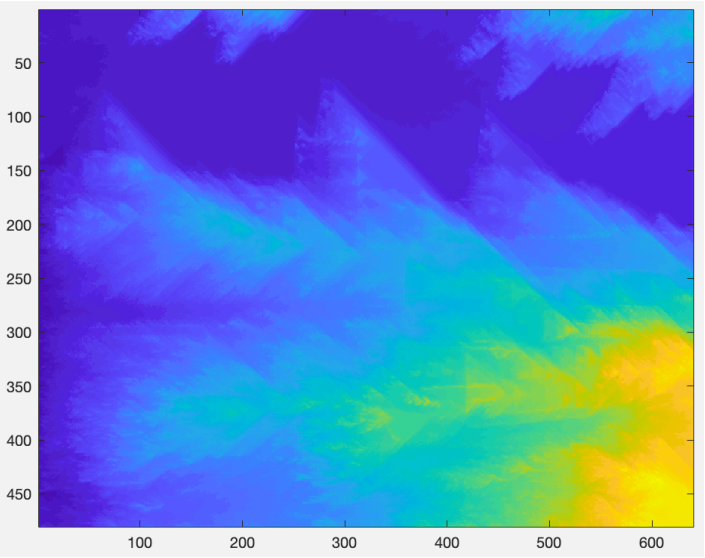
Reduced by using lapacian filter



Energy map



Cumulative minimum energy in vertical direction



Cumulative minimum energy in horizontal direction

First selected vertical

The reduced image with Laplacian filter is not too much different from the reduced image without, the most distinctive difference is the size of the boat and the dark Gothic building. This may be because the gradient magnitude for edges are different. The first selected vertical seam is also different.

6. Image 1


Original (600x550)

Reduced in both height and width by 80 pixels, output size 520x470

Resized using Matlab resampling method, output size 520x470

Explanation: Matlab resizing performs better than seaming. Here I only take out 80 pixels from height and width, and also we can see that the algorithm takes out mostly the white blank on top of image and some black spaces on the right and bottom of image (see the leaves), because obviously the sushi rolls have higher energy values. As a result, the black sushi on the right is squished up.

Image 2

Original (800x533)

Reduced in width by 150 pixels. Output size 650x533

Resized using Matlab. Output size 650x533

Explanation:

Seaming algorithm performs better than Matlab built-in resizing. Since there are a lot of vertical spaces between the two cardinals, the energies are low in those area. The algorithm conveniently takes out those pixels and the result is pretty good.

Image 3

Original (500x333)

Reduced in height by 100 pixels. Output size 500x233

Resized using Matlab. Output size 500x233

Explanation:

Since to top part of the image has lower energy, the top part is removed a lot compared to other area. As a result, Iron Man, Thor and Captain America's heads don't look natural anymore, but the rest of the Avengers look generally fine. In this case, the more ideal way is to probably remove less important features like clothing on the bottom part, light effects, but try to preserve faces.