ECE 4984/5554: Computer Vision, Fall 2015

PS1

Christopher Mobley

Due: Monday, 17 SEP 15

1 Short answer problems [30 points]

1. Give an example of how one can exploit the associative property of convolution to more efficiently filter an image.
    a. The associative property of convolution allow us to more efficiently filter an image, due to the fact we can convolve the desires filters together before convolving them with an image(s). In addition, it allow us to use the resulting filter on multiple images. Resulting in code that is much less computationally expensive. The example of this giving in class was the convolving the filters [1 2 1] and [[1], [2], [1]] to get [[1, 2, 1], [2, 4, 2], [1, 2, 1]] before convolving it with the matrix [[2, 3, 3], [3, 5, 5], [4, 4, 6]] to get 65. Instead of having to convolve [1 2 1] with [[2, 3, 3], [3, 5, 5], [4, 4, 6]] to get [[11],[18],[18]] and then convolve this with [[1], [2], [1]] to get 65. This is more computationally expensive and requires continually storing [1 2 1] and [[1], [2], [1]] instead of just the new convolved filter.
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]?
    a. [0 1 1 1 1 1 1 1]
3. The filter f' = [0 -1/2 0 1/2 0] gives 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.)
    a. [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.
    a. Two specific ways in which one can reduce the amount of fine, detailed edges detected with Canny edge detector are using non-maximum suppression, which thins wide "ridges" down to single pixel width, as well as linking and thresholding (hysteresis), in which you define both a low and high threshold. The high threshold is used to start and edge while the low threshold is used to continue an edge.
5. Describe a possible flaw in the use of additive Gaussian noise to represent image noise.
    a. The noise in an image is contingent on many different factors, of which Gaussian noise only represents one type.
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.

a. Important Assumptions
    i. The automotive assembly and parts have sharp edges that won't be missed by Canny edge detection
    ii. All possible orientations and distances templates have been recorded
    iii. The computer running the detection method has enough computational power to easily match all templates in the time the part is in optional view.
b. Steps
    i. Take template images of complete parts/assemblies in a wide assortment of distances and orientations
    ii. Use Canny edge detection using non-maximum suppression, as well as linking and thresholding, with optimally determined thresholds, to draw an outline of each of part in the assembly and the assembly in general.
    iii. Use Chamfer Distance based on templates to both find the specific part and determine if any defects are present.

2 Programming problem: content-aware image resizing [70 points]

1. 10 points. Write a script called SeamCarvingReduceWidth.m(py) which does the following by using the functions defined above:
    a. Loads a color input image called inputSeamCarvingPrague.jpg. Download the image from here (http://filebox.ece.vt.edu/~F15ECE5554ECE4984/resources/images/inputSeamCarvingPrague. jpg)
    b. Reduces the width of the image by 100 pixels using the above functions.
    c. Saves the resulting image as outputReduceWidthPrague.png. Submit it. Display this output in your answer sheet. Submit the script.

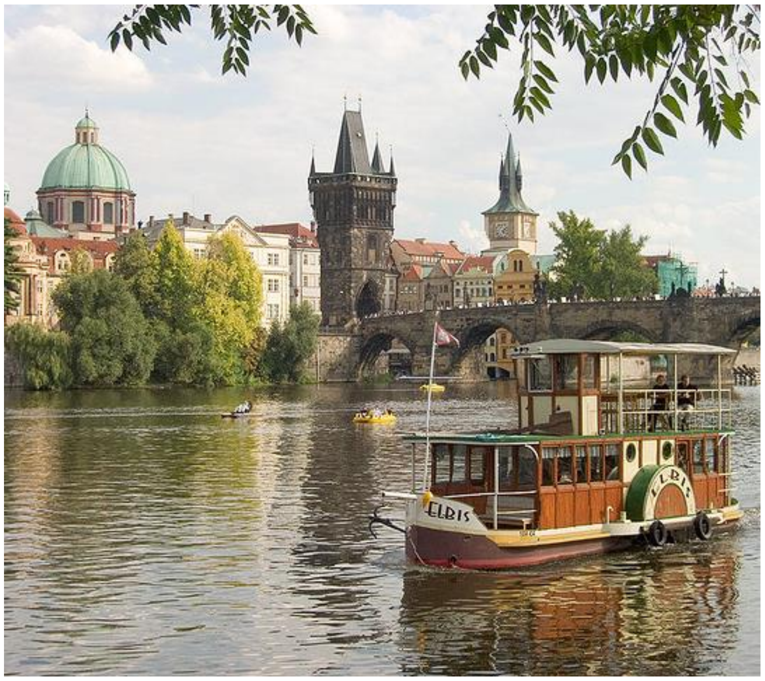

Figure 1: Prague image having had width reduced by 100 pixel through seam carving

d. Repeat the steps for an input image called inputSeamCarvingMall.jpg. Download the image from here (http://filebox.ece.vt.edu/~F15ECE5554ECE4984/resources/images/inputSeamCarvingMall. jpg). Save the output as outputReduceWidthMall.png. Display the output in your answer sheet.

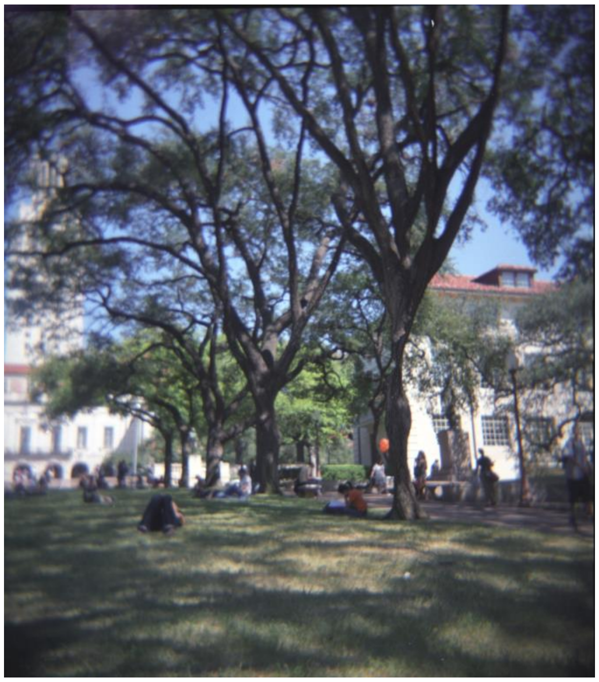

Figure 2: Mall image having had width reduced by 100 pixel through seam carving

2. 10 points. Repeat the above steps for both the input images, but reduce the height by 100 pixels. Call the script SeamCarvingReduceHeight.m(py), and save the output images as outputReduceHeightPrague.png and outputReduceHeightMall.png respectively. Display both the outputs in your answer sheet. Submit the script which loads the image inputSeamCarvingPrague.jpg

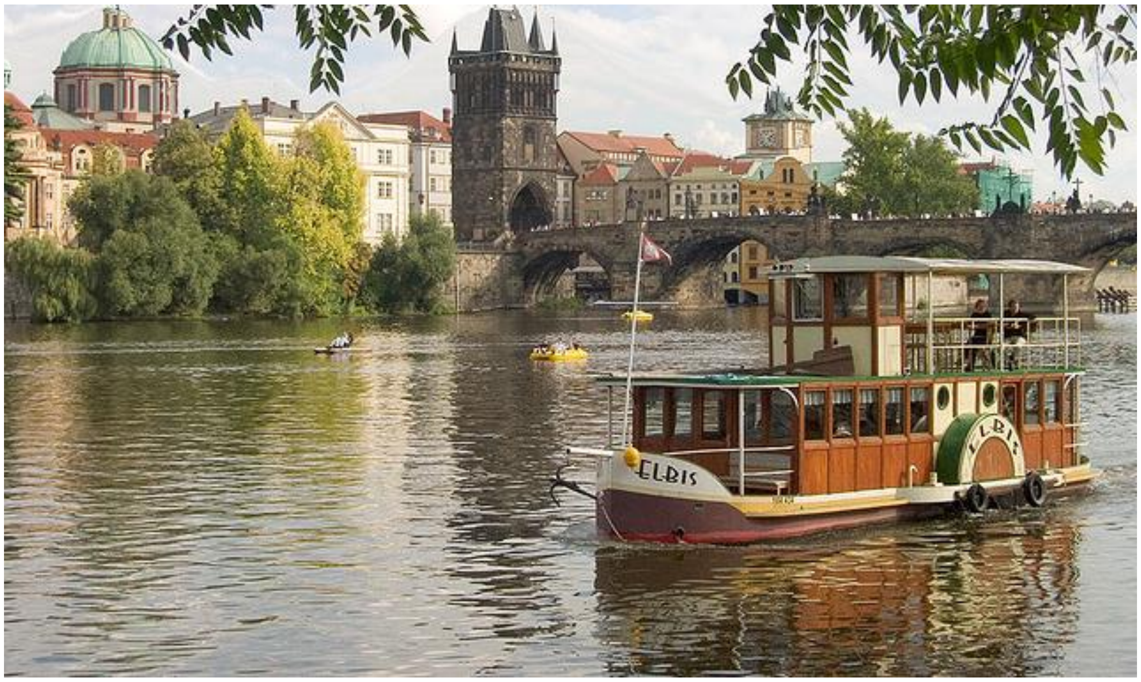

Figure 3: Prague image having had height reduced by 100 pixel through seam carving

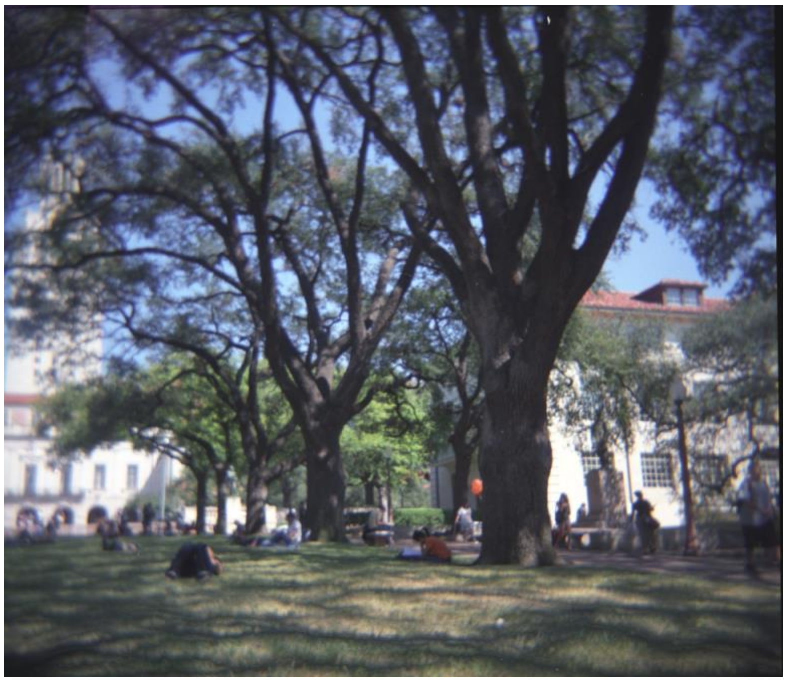

Figure 4: Prague image having had height reduced by 100 pixel through seam carving

3. 10 points. Display in your answer sheet:
    a. the energy function output for the provided image inputSeamCarvingPrague.jpg, and

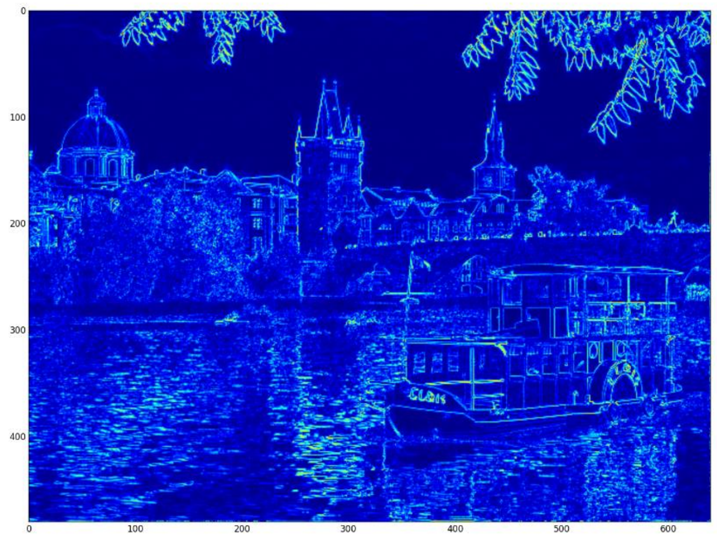

Figure 5: Prague image energy map

    b. the two corresponding cumulative minimum energy maps for the seams in each direction (use the Matlab's imagesc or Python's matplotlib.pyplot.imshow). Explain why these outputs look the way they do given the original image's content.

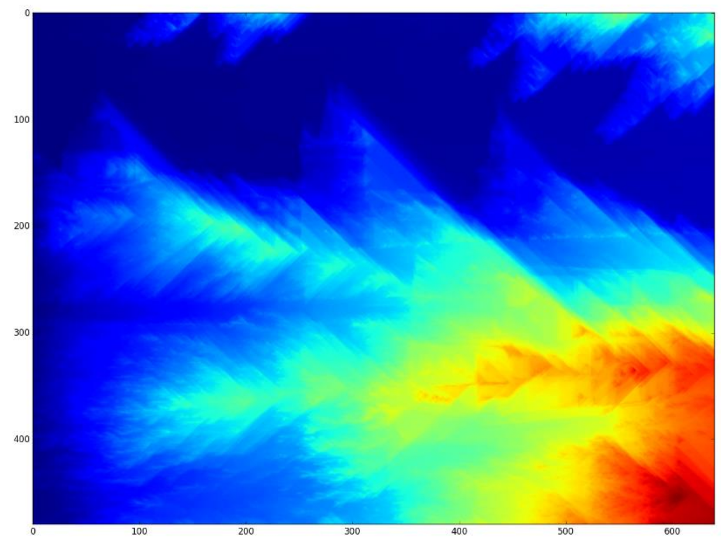

Figure 6: Prague image horizontal cumulative energy map

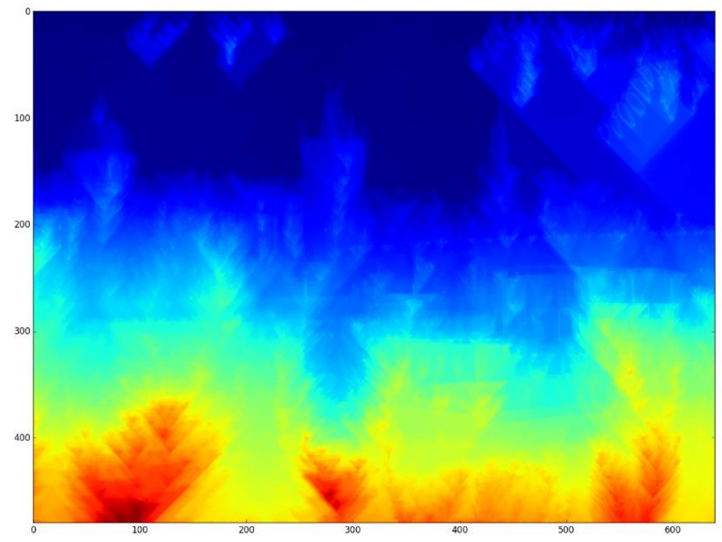

Figure 7: Prague image vertical cumulative energy map

Figure 5, represent the gradient magnitude or increase/decrease in the magnitude of a pixel's RGB value in relation to the pixels around it. As a result, regions which have very little change in RGB value, such as the sky and forest, tend to be dark blue while the rippling water ranges from light blue to teal. Figures 6 and 7 represents the cumulative sum of the magnitude of the gradient of each minimum connected pixel across the image from the left to right side of the image, in the horizontal case, and the top to bottom of the image, in the vertical case. As a result, regions with higher RBG values in Figure 5 will cause the regions to the right of or bottom of to have a higher RGB value. As a result, Figure 6 has large RGB values at the bottom right of the picture which result from the high gradient magnitude of the rippling water, in contrast to the sky which remains dark blue. Figure 7 has large RGB values at the bottom due to the gradient magnitude of the reflection of the forest in the rippling water and the ferry.

4. 10 points. For the same image inputSeamCarvingPrague.jpg, display the original image together with
   a. the first selected horizontal seam and

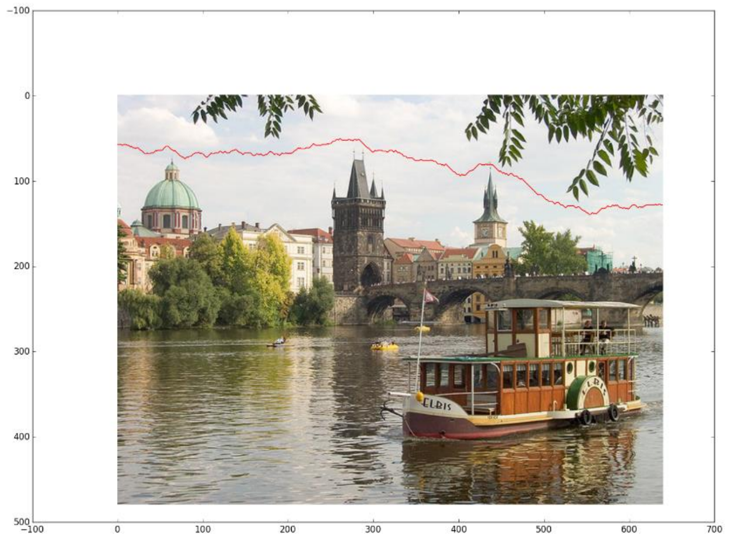

Figure 8: Prague image with the first horizontal seam drawn

   b. the first selected vertical seam in your answer sheet. Explain why these are the optimal seams for this image.

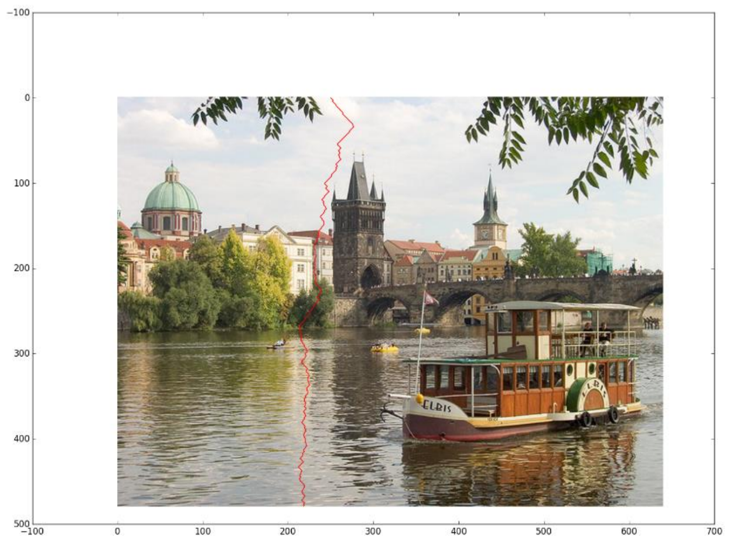

Figure 9: Prague image with the first vertical seam drawn

Figures 8 and 9, show the optimal horizontal and vertical seams. The optimal seems are those which are not likely to be noticed if they are carved out. The seams shown follow the path of least cumulative energy, or least cumulative gradient magnitude. As a result, the absence of these seams is less likely to be noticed than those of higher cumulative gradient magnitude.

The following are the optional seams

5. 10 points. Make some change to the way the energy function is computed (i.e., filter used, its parameters, or incorporating some other prior knowledge). Display the result and explain the impact on the results for some example in your answer sheet. You need not submit this code.

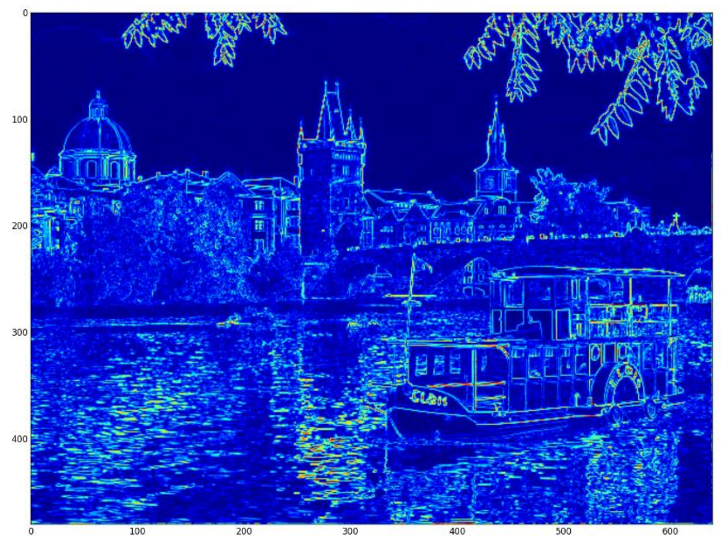

Figure 10: Prague image energy map with Robert's finite difference filter

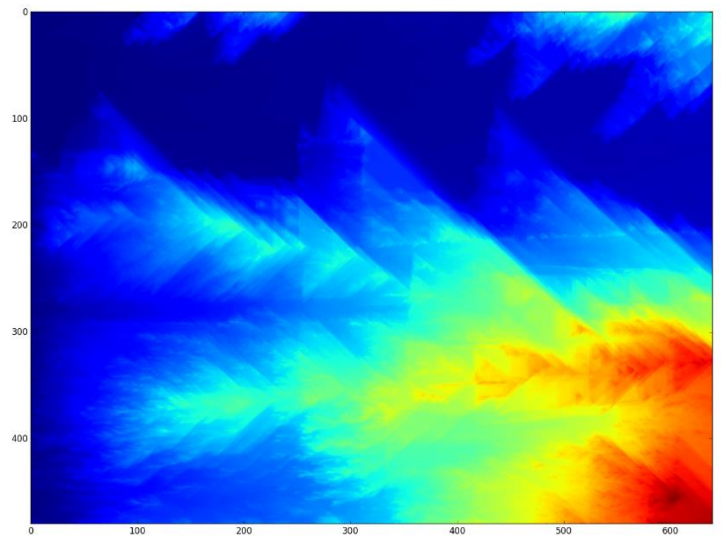

Figure 11: Prague image horizontal cumulative energy map with Robert's finite difference filter

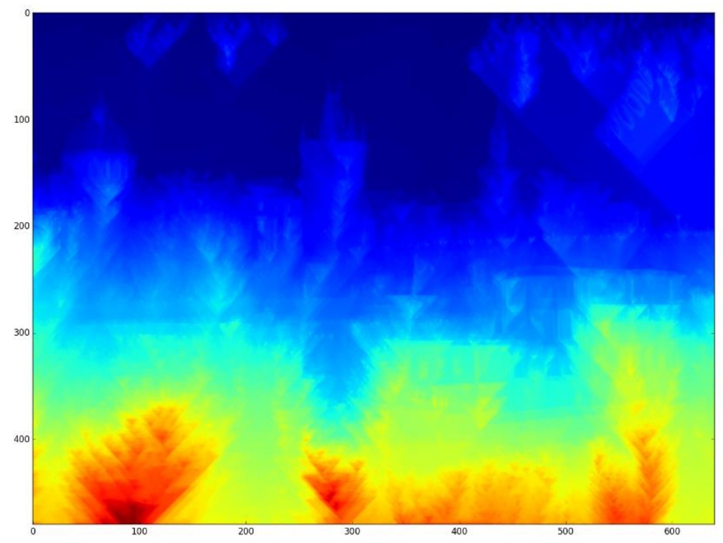

Figure 12: Prague image vertical cumulative energy map with Robert's finite difference filter

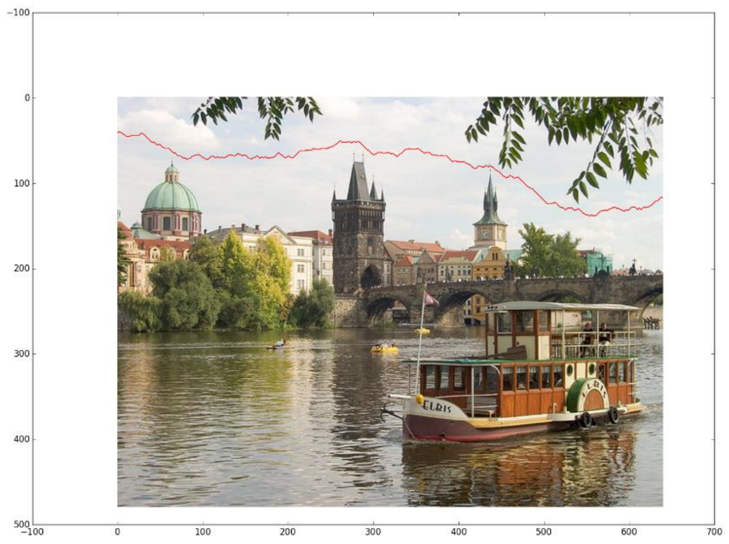

Figure 13: Prague image with the first horizontal seam drawn with Robert's finite difference filter

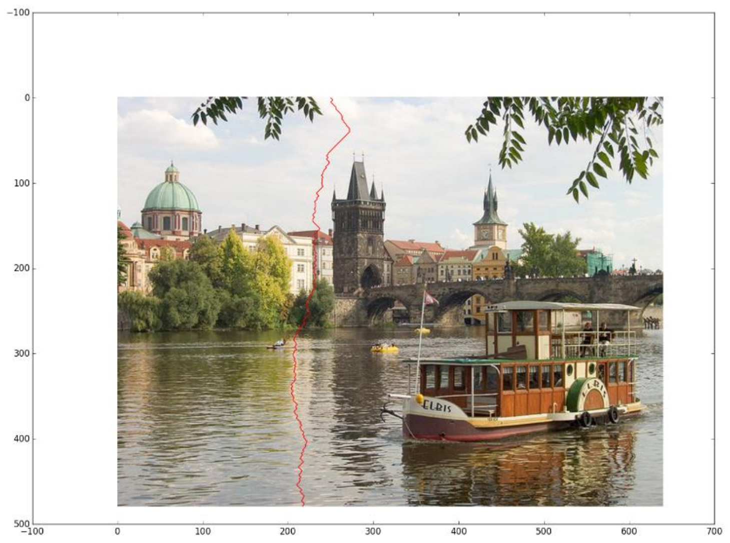

Figure 13: Prague image with the first vertical seam drawn with Robert's finite difference filter

Instead of calculating the graduate magnitude by convolving [1,-1] with the image to the partial derivate in the x direction, and [[1],[-1]] to get the partial derivative in the y direction. The Roberts finite difference filters were used to obtain these partial derivatives. Figure 10 through 12, show an average increase in the gradient magnitude when calculated using the Roberts finite difference filter. However, due to the fact that this increase appears to be similar for each pixel, the resulting seam are approximately the same.

6. 20 points. Now, for the real results! Use your system with different kinds of images and seam combinations, and see what kind of interesting results it can produce. The goal is to form some perceptually pleasing outputs where the resizing better preserves content than a blind resizing would, as well as some examples where the output looks unrealistic or has artifacts. Include results for at least three images of your own choosing. Include an example or two of a "bad" outcome. Be creative in the images you choose, and in the amount of combined vertical and horizontal carvings you apply. Try to predict types of images where you might see something interesting happen. It's ok to fiddle with the parameters (seam sequence, number of seams, etc) to look for interesting and explainable outcomes. For each result, include the following things, clearly labeled:

   a. the original input image. Image Credit: www.piercepioneer.com

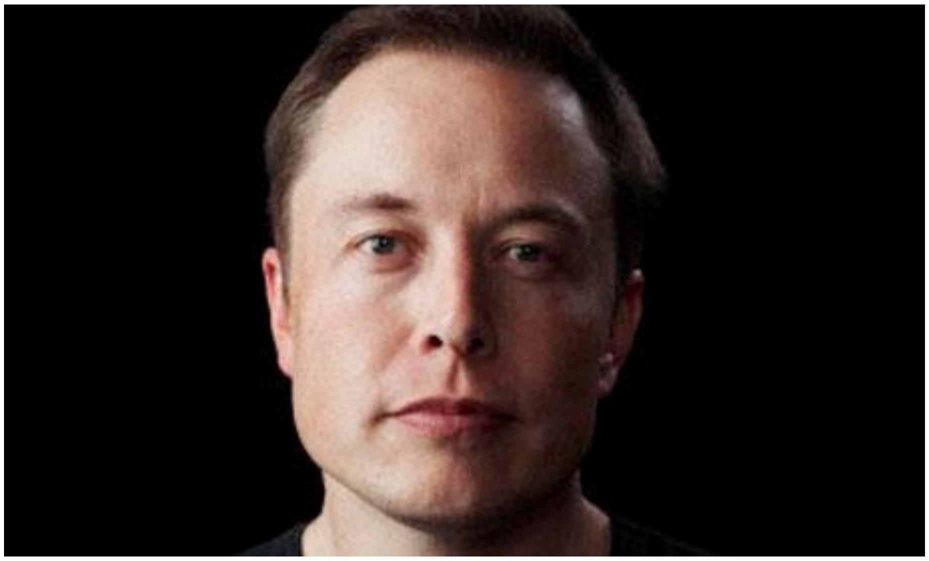

Figure 14: Original Elon Musk image

b. your system's resized image,

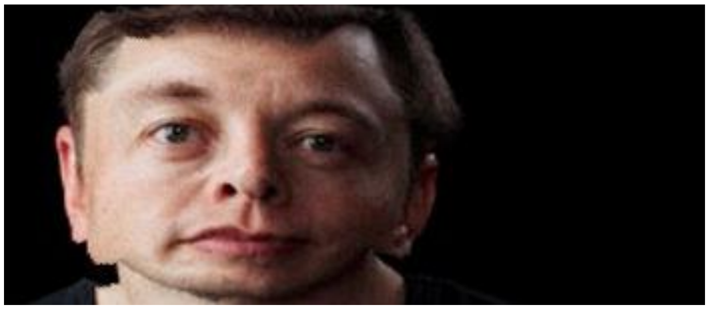

Figure 15: Elon Musk image resized with seam carving

c. the result one would get if instead a simple resampling were used (via Matlab's imresize or Python's scipy.misc.imresize)

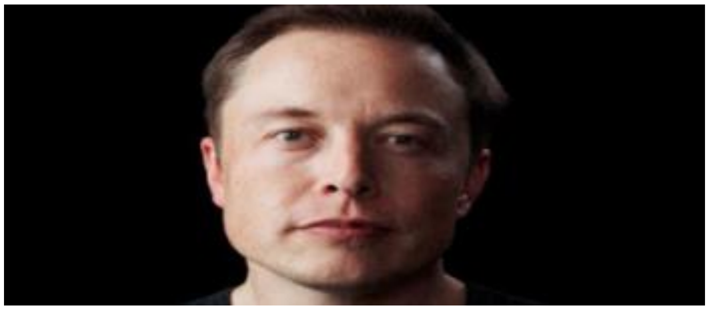

Figure 16: Elon Musk image resized with traditional imresize

d. the input and output image dimensions,

Original Image: (276L, 460L, 3L)

Output Image: (150L, 350L, 3L)

e. the sequence of removals that were used

First, 110 pixels were removed from the width. After which, 126 pixel were removed from the height.

f. a qualitative explanation of what we're seeing in the output.

Figures 15 and 16 show the limitations of the seam carving technique. It determine the importance of each pixel based on the magnitude of their gradient. While this work well for many pictures, it does not tend to work well in pictures in which we want to preserve human features due to the fact that our skin color does not vary significantly and thus has a low energy level which will be removed by seam carving. In this picture a traditional resize of the image leave Elon Musk face intact unlike the one produced by seam carving.

a. the original input image. Image Credit: http://www.farm3.static.flickr.com

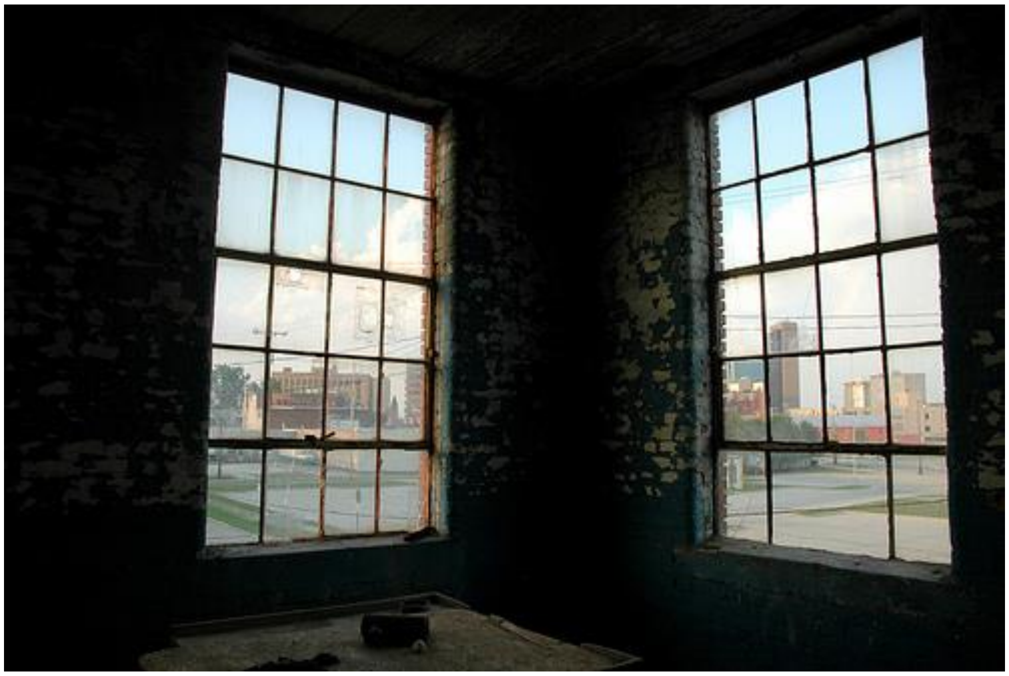

Figure 17: Original warehouse windows image

b. your system's resized image,

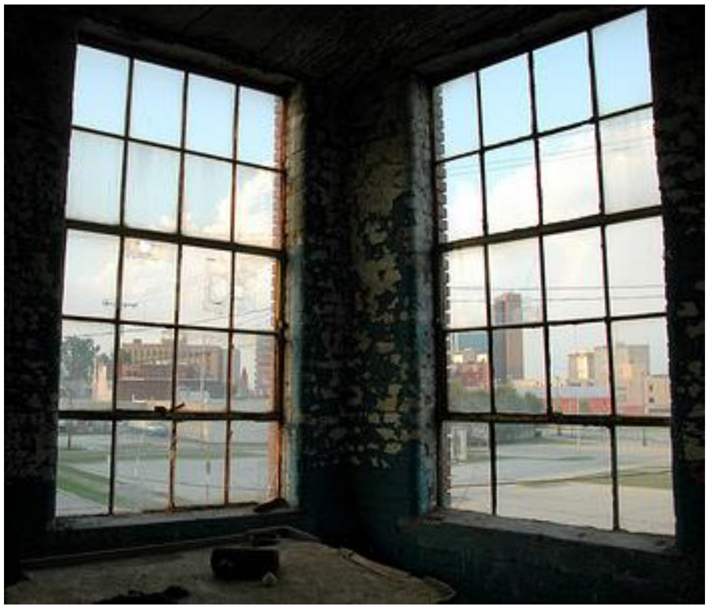

Figure 18: Warehouse windows image resized with seam carving

c. the result one would get if instead a simple resampling were used (via Matlab's imresize or Python's scipy.misc.imresize)

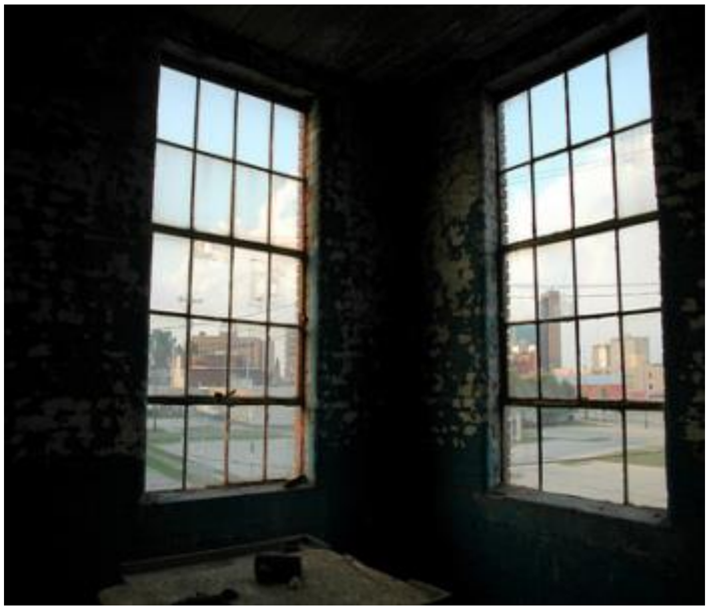

Figure 19: Warehouse windows image resized traditional imresize

d. the input and output image dimensions,

Original Image: (333L, 500L, 3L)

Output Image: (300L, 350L, 3L)

e. the sequence of removals that were used

First, 150 pixels were removed from the width. After which, 33 pixel were removed from the height.

f. a qualitative explanation of what we're seeing in the output.

Figures 18 and 19 show the power of seam carving. Since seam carving determines the importance of each pixel based on the magnitude of their gradient, places with high gradient magnitudes are left intact while places with low gradient magnitudes are carved out. While the traditional resize lost information regarding what lay outside the window, our seam carving image left these intact while carving out portion of the long brick wall, which would not be noticed by or important to most observers.

a. the original input image. Image Credit: http://www.spotlight-online.de/

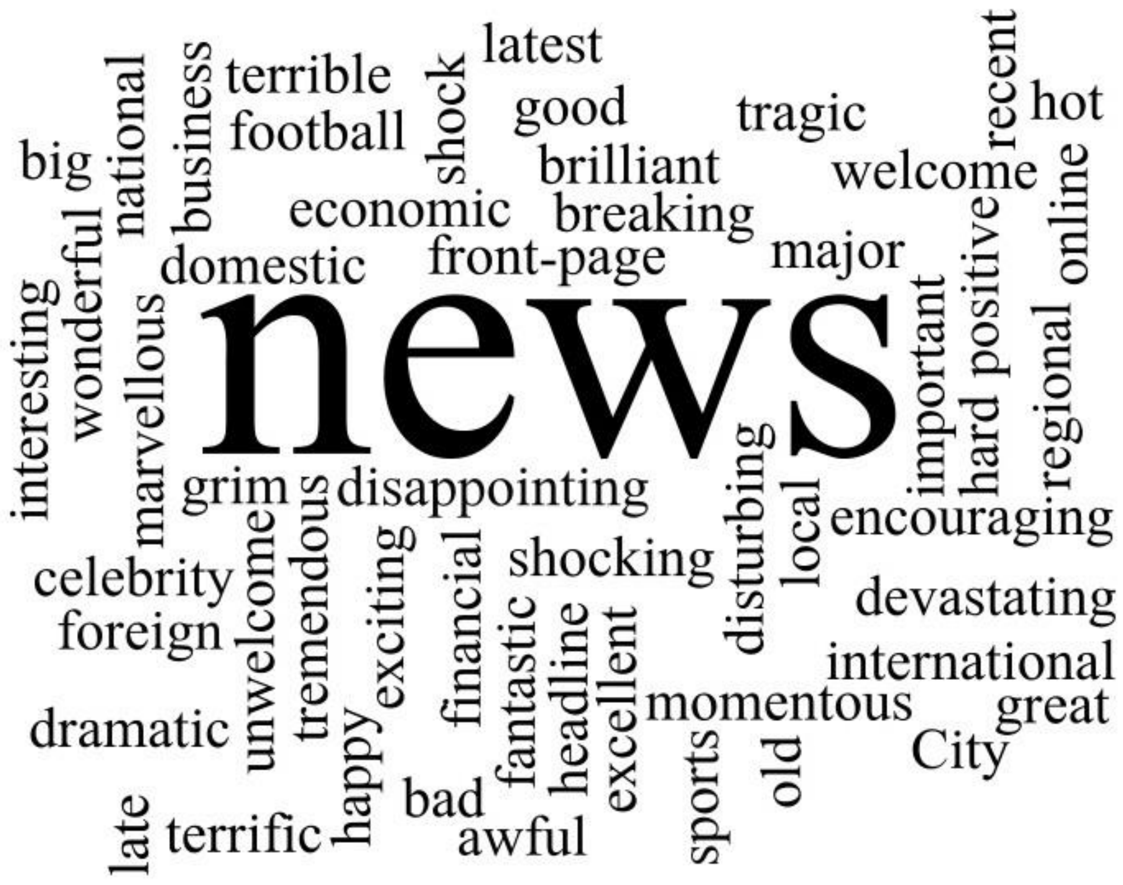

Figure 20: Original word art image

b. your system's resized image,

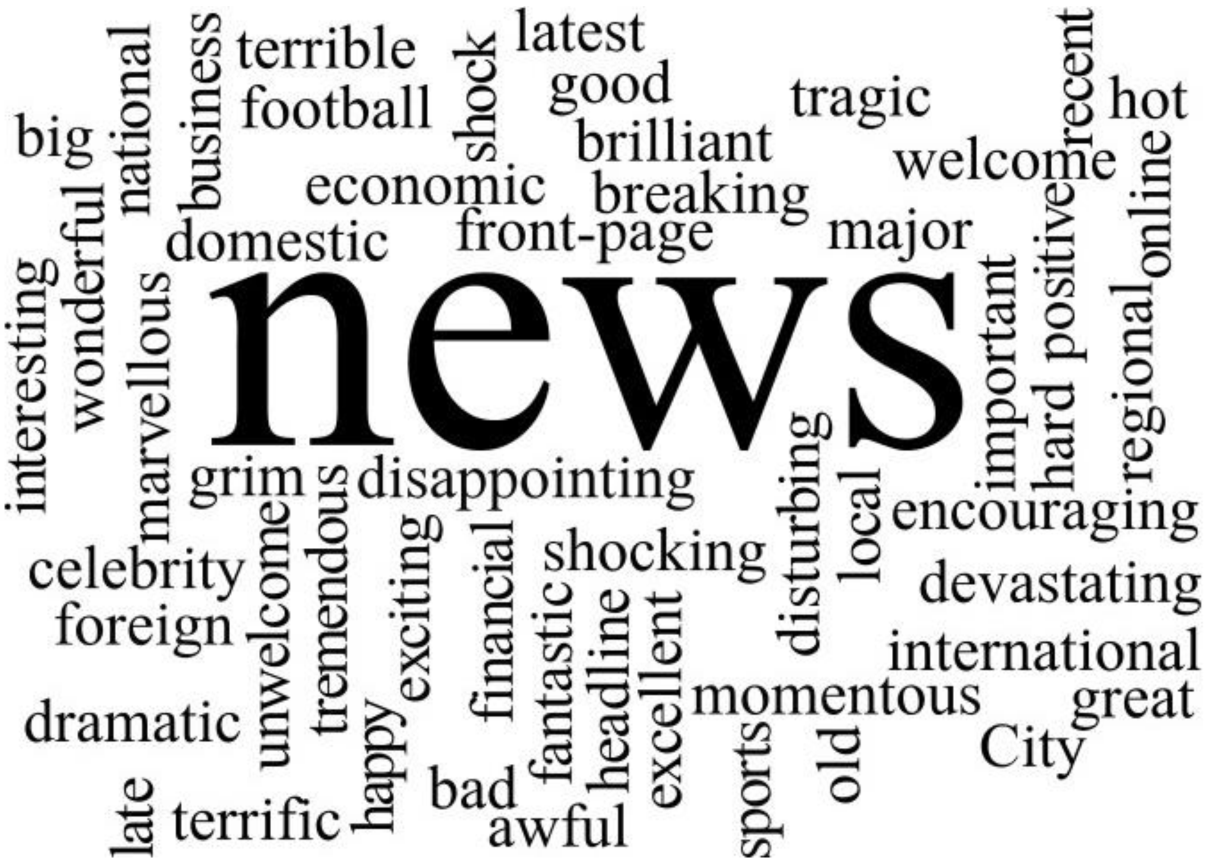

Figure 21: Word art image resized with seam carving

c. the result one would get if instead a simple resampling were used (via Matlab's imresize or Python's scipy.misc.imresize)

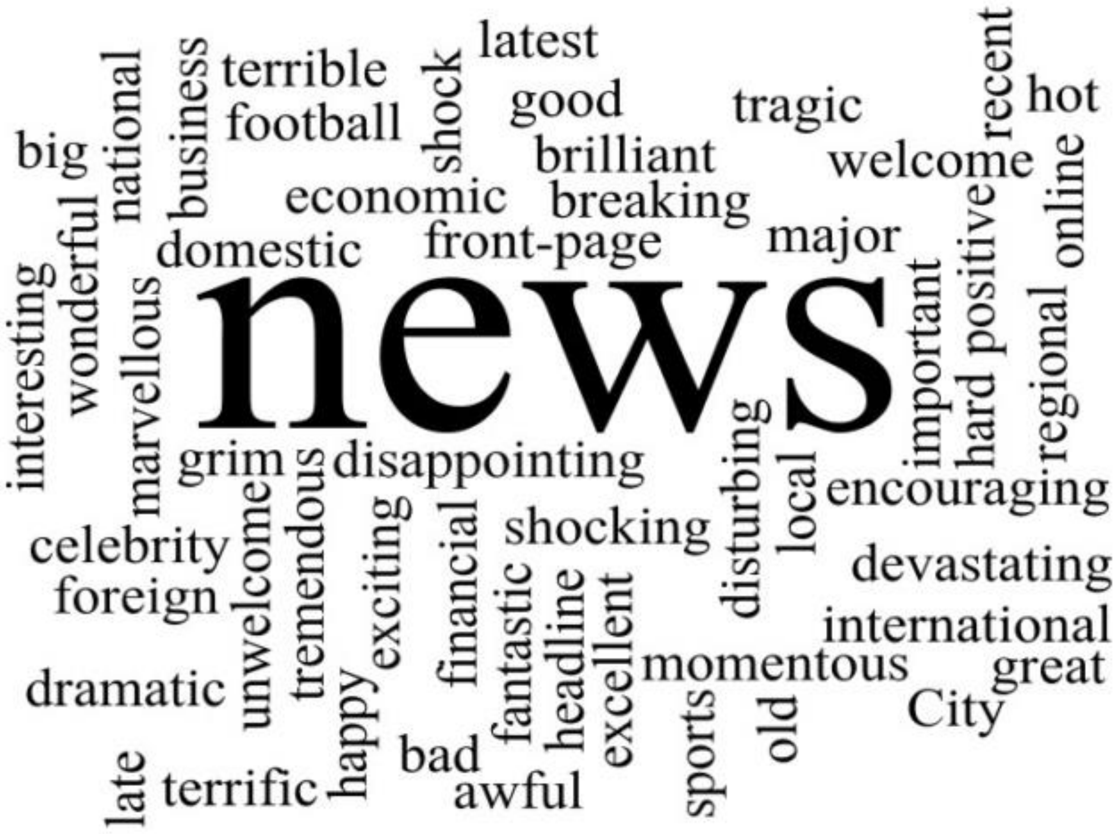

Figure 22: Word art image resized with traditional imresize

d. the input and output image dimensions,

Original Image: (546L, 759L, 3L)

Output Image: (477L, 700L, 3L)

e. the sequence of removals that were used

First, 69 pixels were removed from the height. After which, 159 pixel were removed from the width.

f. a qualitative explanation of what we're seeing in the output.

Figures 21 and 22 again show the power of seam carving. Since seam carving determines the importance of each pixel based on the magnitude of their gradient, places with high gradient magnitudes are left intact while places with low gradient magnitudes are carved out. While the traditional resize reduced the size of the text, making it more difficult to read. The seam carving image preserved this size by cutting out portion of unnecessary white space. Thus making the image the desired size, as well as maintaining it readability.

3 [OPTIONAL] Extra credit [up to 10 points each, max possible 20 points extra credit]

Below are ways to expand on the system you built above. If you choose to do any of these (or design your own extension) include in your answer sheet, an explanation of the extension as well as images displaying the results and a short explanation of the outcomes. Also include a line or two of instructions telling us what needs to be done to execute that part of your code and submit your code.

1. Allow a user to mark an object to be removed, and then remove seams until all pixels on that object are gone (as suggested in section 4.6 of the paper). Either hard-code the region specific to the image, or allow interactive choices (Matlab's ginput or impoly, Python's pylab.ginput functions are useful to get mouse clicks or draw polygons).

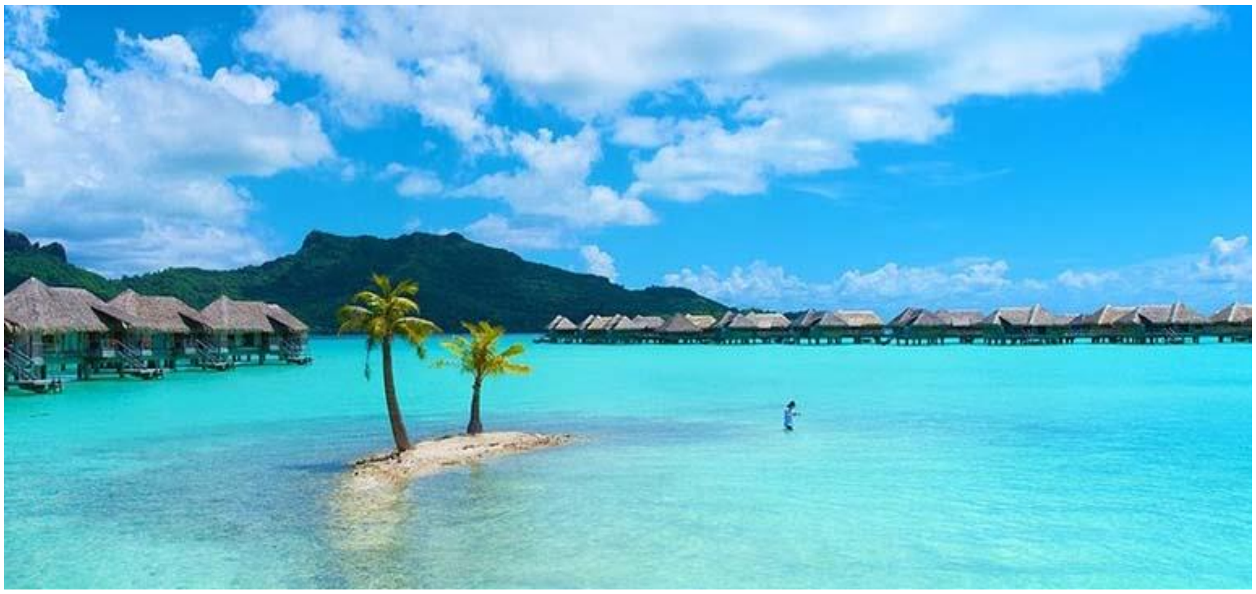
Figure 23: Original Beach Image Image Credit: http://www.quitsmokingmessageboard.com/

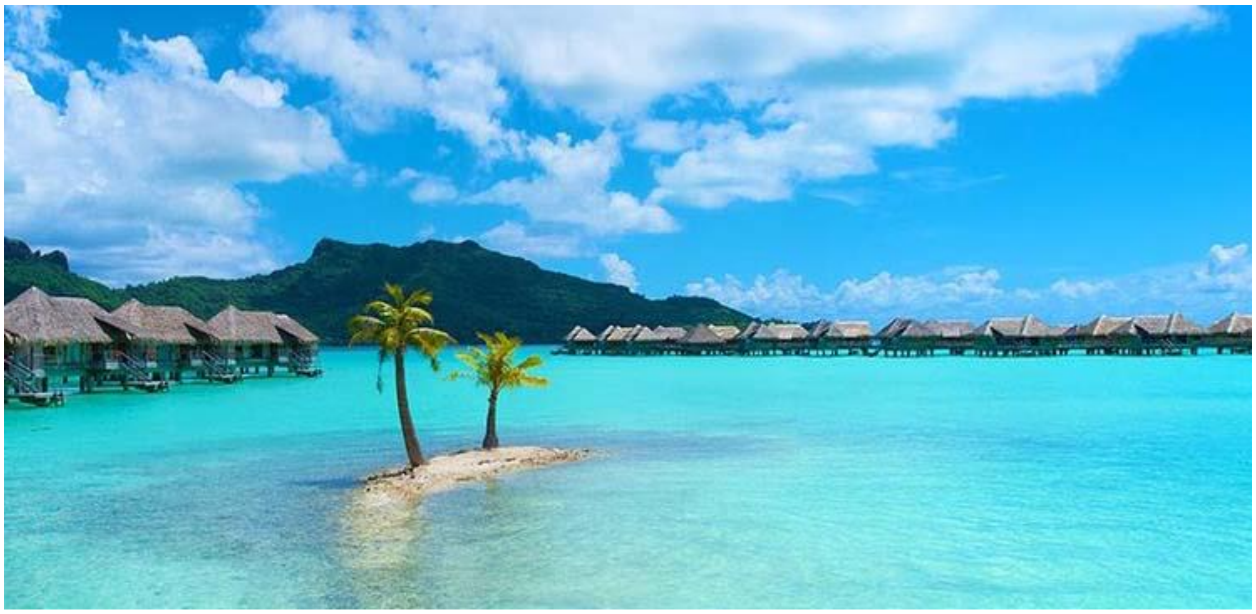

Figure 24: Beach image resized with seam carving to remove far right person

Figure 23 show an image of a beach with an isolate person. Figure 24 shows this same beach with the person removed. By adding 1000 to the RBG values of the energy image and then setting all pixels to 0 around the object we want to delete, we are forcing our optimal seam carving algorithms to remove the selected pixels first, in this case the person.

In order to run this approach, run remove_marked_object.py after uncommenting the statements outside the functions or run the following f from remove_marked_object import removed_marked_object and run removed_marked_object(im, desired_reduction_width, desired_reduction_height).

2. Design an alternate energy function, instead of the gradient magnitude. Explain your choice, and show how it can influence the results as compared to using the gradient magnitude. Choose an image or two that illustrates the differences well.
    a.
3. To avoid warping regions containing people's faces, have the system try to detect skin-colored pixels, and let that affect the energy map. Try using the hue (H) channel of HSV color space (For Matlab, see rgb2hsv function to map to HSV color space. For Python, see skimage.color.rgb2hsv module). Think about how to translate those values into energy function scores.
    a.
4. Implement functions to increase the width or height of the input image, blending the neighboring pixels along a seam. (See the Seam Carving paper for details.) Demonstrate on an image that clearly shows the impact.
    a.
5. Implement the greedy solution, and compare the results to the optimal Dynamic Programming solution.

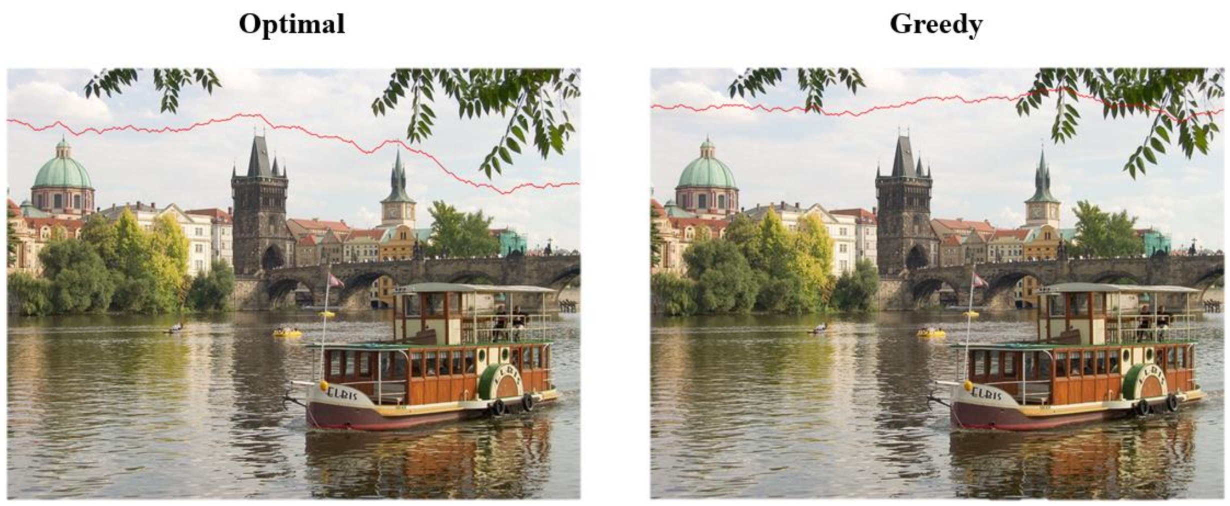

Figure 25: First horizontal seam comparison for optimal versus greedy

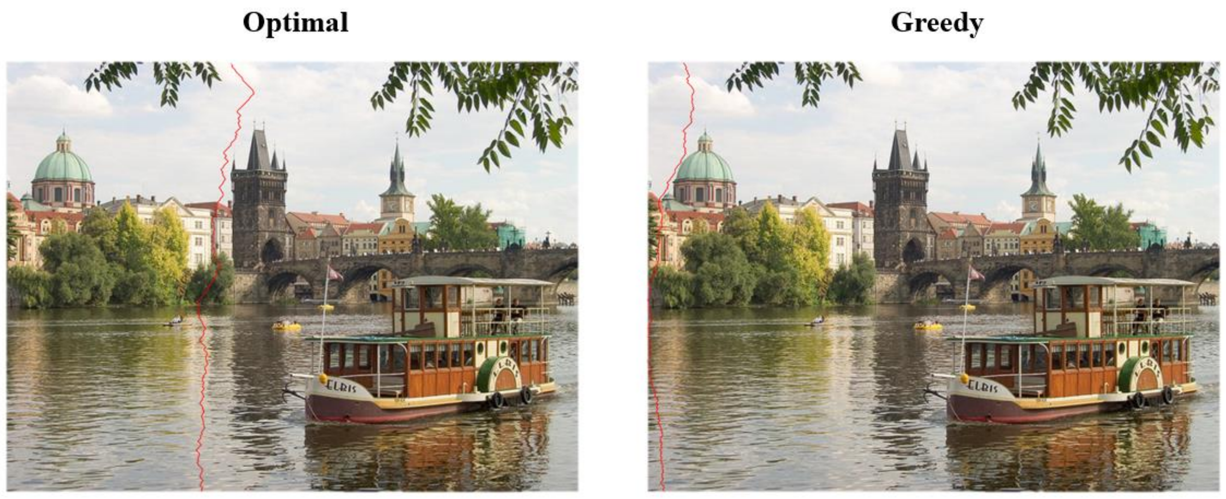

Figure 26: First vertical seam comparison for optimal versus greedy

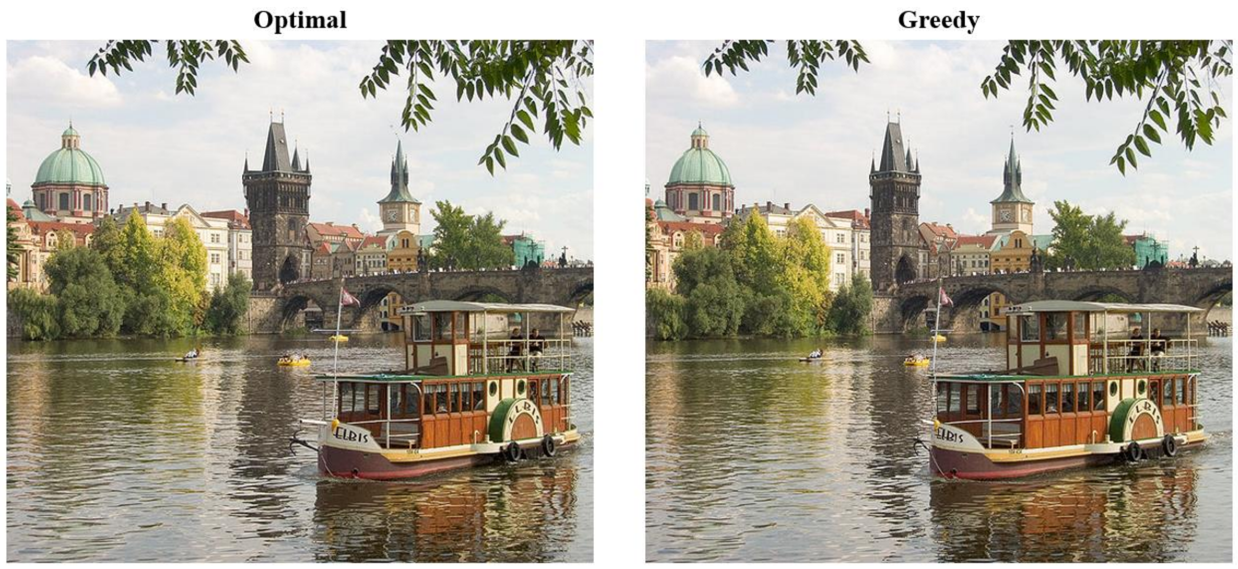

Figure 27: Reduce width by 100 pixel comparison for optimal versus greedy

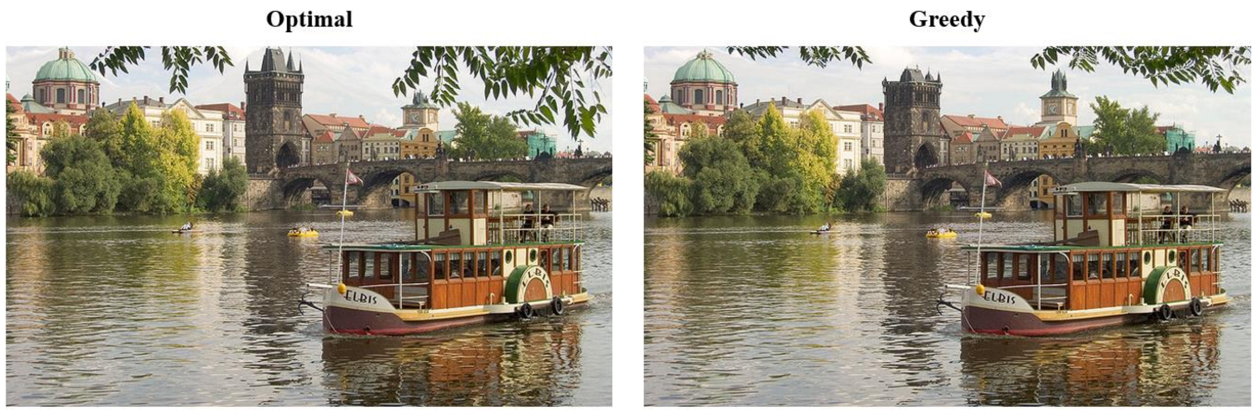

Figure 28: Reduce height by 100 pixel comparison for optimal versus greedy

Figures 25 through 28 show a comparison of the optimal versus the greedy approaches. Unlike the optimal method, which dynamically follows the path of lowest cumulative energy, the greedy approach starts at the pixel with the least gradient, regardless of whether this is the best path or not hence the name greedy. As a result, it is significantly more likely to carve out important features and/or leave artifacts. Figure 28 most clearly shows the differences between the two methods, as the optimal method leave the foliage unchanged, the greedy method deletes much of it from the image.

In order to run the greedy approach, run either seamCarvingGreedyReduceHeight.py or seamCarvingGreedyReduceWidth.py after uncommenting the statements outside the function or run the following from SeamCarvingGreedyReduceHeight import SeamCarvingGreedyReduceHeight and run SeamCarvingGreedyReduceHeight(im, desired_reduction) or from SeamCarvingGreedyReduceWidth import SeamCarvingGreedyReduceWidth and run SeamCarvingGreedyReduceWidth(im, desired_reduction).

Note: I discussed this assignment with Murat Ambarkutuk, Evan Smith, Orson Lin, and Yi Tien