Computer Vision 2020
HW#4
Due April 15
Submission:

- *Please use your programming language of choice, MATLAB, C, python, …*
- *Please prepare a .pdf report file containing your answers.*
- *Compress your report file and your codes into a single zip file.*
- *Name the file as your last name followed by hw4 for example: shirani_hw4.zip*
- *Submit by uploading to Avenue (under Assessment/Assignment). Please do NOT email me your file.*

Problem 1: Image Resizing using Seam Carving

In this problem you will implement and evaluate seam carving for resizing an image using its content with (hopefully) minimal noticeable distortions. Write a `main.m` script file that reads a given color image file into MATLAB using `imread`, converts it to double so that pixels have values in the range [0..1], calls functions to implement seam carving, and then creates an output `jpg` image file (if you are using a programming language other than MATLAB perform equivalent steps). To do this write code to do each of the following steps:

1. **Compute the energy function**
   Implement a function `E = imenergy(I)` that computes the energy image `E` from an RGB image array `I`. You can use the MATLAB function `[DX DY] = gradient(I)` to compute the gradient and then sum the absolute values of `DX` and `DY` at each pixel (you can also use sum of squared values). Since `I` is a color image, first convert the image to grayscale (and double) using `rgb2gray` before computing the gradient. `E` should be a double 2D array of the same size as `I`, with a floating-point value at each entry.

2. **Compute the optimal horizontal seam**
   Implement a function `S = horizontal_seam(I)` that takes an image and finds the (one) optimal horizontal seam, returning a vector of length equal to the number of columns in `I` such that each entry in vector `S` is an integer-valued row number indicating which pixel in that column should be removed. For example, `S(10)=37` means that in the 10th column the pixel in row 37 is to be removed. The optimal seam can be found using dynamic programming to compute, left to right, the cumulative minimum energy array, `M`, as described in the lecture. Starting from the minimum value in the rightmost column of `M`, `S` is computed during the backward pass from the rightmost column to the leftmost column of `M`. Adjacent entries in `S` must be at most one row apart so that the seam found is a path of 8-adjacent pixel coordinates. (Note: you can plot the points in `S` on top of image `I` using `imshow, hold on, and plot`.)

3. **Remove 1 horizontal seam**

Implement a function `J = remove_horizontal_seam(I, S)` that removes *one* horizontal seam from image `I` (of size *m* x *n*) to produce new image `J` that has size (*m*-1) x *n*.

4. **Resize**
   Implement a function `J = shrink(I, num_rows_removed, num_cols_removed)` that takes an input color image `I` and computes an output color image `J` that has *`num_rows_removed`* fewer rows than `I`, and *`num_cols_removed`* fewer columns than `I`. This should find one seam, remove it, find the next seam, remove it, etc. Implement this function using only the horizontal seam detector and remover by (a) removing all horizontal seams, (b) rotating the image 90° using `J = permute(I, [2 1 3])`, (c) removing all vertical seams, and (d) un-rotating the image (using `J = permute(I, [2 1 3])`). Hence you do not need to implement a separate vertical seam removal function.

Experiments

1. Using the test image union-terrace.jpg from Avenue run your shrink function with the following values and create three images called `E1a.jpg` , `E1b.jpg` and `E1c.jpg` showing the results
   a. `num_rows_removed` = 0, `num_cols_removed` = 100
   b. `num_rows_removed` = 100, `num_cols_removed` = 0
   c. `num_rows_removed` = 100, `num_cols_removed` = 100
2. Create an image called `E3.jpg` showing the original image together with the first selected horizontal seam and the first selected vertical seam overlaid. Explain why these are reasonable optimal seams.
3. Find on the web or take a photo to use as an input image (that is not too big) and name it `E4a.jpg` where this image shows an ***interesting successful result*** of shrinking (either horizontally, vertically, or both) an image by some non-trivial amount. Save your result image as `E4b.jpg` Hand in both images and describe why it seems to work well.
4. Find on the web or take a photo to use as an input image (that is not too big) and name it `E5a.jpg` where this image shows an ***interesting poor result*** of shrinking a different image (either horizontally, vertically, or both) by some non-trivial amount. Save your result image with the name `E5b.jpg` Hand in both images and describe why it seems to work poorly.

Problem 2: Bilateral Filtering

Implement a function BilateralImage(image, sigmaS, sigmaI) to bilaterally blur an image. "sigmaS" is the spatial standard deviation in the spatial domain and "sigmaI" is the standard deviation in intensity/range. Hint: The code should be very similar to a Gaussian Blur that you have implemented in previous homework assignments. Take the "Yosemite.png" image from Avenue as input and try two different values for sigmaS and two different values for sigmaI for a total of 4 different combinations. Save the resultant image as "bilateraltask1.jpg" to “bilateraltask4.jpg”.