Homework #3: Image Resizing using Seam Carving

Assigned: Monday, October 3
Due: Friday, October 14

In this assignment you will implement and evaluate an algorithm for resizing an image that uses image content to do this with (hopefully) minimal noticeable distortions. The method is called **Seam Carving**. It is implemented in Photoshop as a feature called "content aware scaling." First, read the paper "Seam Carving for Content-Aware Image Resizing" by S. Avidan and A. Shamir, with emphasis on Section 3. 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. To do this write code to do each of the following steps:

1. **Compute the energy function**
   Implement a function `E = imenergy(I)` (in a file called `imenergy.m`) that computes the energy image `E` from an RGB image array `I` as defined in Equation 1 in the paper. 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. 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 10^{th} 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 paper. 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.)` See the Wikipedia page on "seam carving" to see how dynamic programming works here (though the example shown is for a vertical seam). NOTE: It is hard to vectorize the dynamic programming algorithm in MATLAB so you are free to use loops.

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])` (do not use `transpose`), (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 , 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 of the energy function output called `E2a.jpg` and an image called `E2b.jpg` of the cumulative minimum energy array M for the first horizontal seam. Use `imagesc` and `saveas` to create these images. Explain why these images look as they do.
3. 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.
4. 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.
5. 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.

Extra Credit (up to 10%)

Implement one or more of the following or your own extension. Show and describe results. Hand in in a separate ExtraCredit folder your code and images.

1. Write a function `J = expand(I, num_rows_expanded, num_cols_expanded)` that expands the image by adding horizontal and vertical seams to create an output image that has `num_rows_expanded` more rows and `num_cols_expanded` more columns.
2. Implement one or more alternate energy functions and compare them experimentally.