CS1674: Homework 2 - Programming

Due: 9/14/2016, 11:59pm

This assignment is worth 50 points.

Please post on Piazza if you have any questions, rather than emailing the instructor directly, unless you have a concern you wish to keep private. I will monitor Piazza daily.

You will implement one part of the content-aware image resizing technique described in Shai Avidan and Ariel Shamir's SIGGRAPH 2007 paper, "Seam Carving for Content-Aware Image Resizing", available here. The goal is to implement the method, and then examine its performance on different kinds of input images.

First read through the paper, with emphasis on sections 3, 4.1, and 4.3. Note: choosing the next pixel to add one at a time in a greedy manner will give sub-optimal seams; the dynamic programming solution ensures the best seam (constrained by 8-connectedness) is computed. Use the dynamic programming solution as given in the paper and *explained in class*.

You will need the following Matlab functions. Some of them will be provided for you, and some you will have to write. Put each function you write in a separate file called [function-name].m and submit all of them (excluding ones that were provided with the assignment).

- [5 points] `energyImage = energy_image(im)` - to compute the energy at each pixel using the magnitude of the x and y gradients (equation 1 in the paper; sqrt((dI/dx)^2+ (dI/dy)^2)). There are at least two ways to compute the gradient in each direction, i.e. the dI/dx and dI/dy. One of these is to use two of the filters we discussed in class. You can also explicitly compute the difference between pixels in the x and y direction by looping over pixels. However, don't use Matlab's `imgradient` function.
  The input `im` should be a nrows-x-ncols-x-3 matrix of datatype uint8, e.g. the output of `imread` on a color image. However, you need to convert it to grayscale before computing the gradients, using `rgb2gray`. The output should be a 2D matrix of datatype double.
- [provided] `M = cumulative_minimum_energy_map(energyImage,seamDirection)` - to compute minimum cumulative energy. The input `energyImage` should be a 2D matrix of datatype double. (It should be the output of the `energy_image` function defined above.). The input `seamDirection` should be the strings 'HORIZONTAL' or 'VERTICAL'. The output is a 2D matrix of datatype double.
- [provided] `verticalSeam = find_optimal_vertical_seam(M)` and `horizontalSeam = find_optimal_horizontal_seam(M)` - to compute the optimal vertical and horizontal seams. The input should be a 2D matrix of datatype double. (It can be taken from the output of the `cumulative_minimum_energy_map` function defined above). The output is a vector containing the row indices (column indices, respectively) of the pixels belonging to a horizontal (vertical, respectively) seam. For a horizontal seam, you have as many seam points as you have image columns, i.e. one row location for the seam per image

column.
- [5 points] `displaySeam(im, seam, seamDirection)` - to display the selected type of seam on top of an image. The input `im` should be the result of an `imread`. `seamDirection` should be the strings 'HORIZONTAL' or 'VERTICAL'. `seam` should be the output of `find_optimal_vertical_seam` or `find_optimal_horizontal_seam`. The output should display the input image and plot the seam on top of it. Use one plot color for horizontal and another for vertical seams. To plot points on top of a displayed image, use `imshow(im);` followed by `hold on;` followed by `plot(...)`. The origin of the plot will be the top left corner of the image. Note that for the plot, rows are the y axis and columns are the x axis.
- [10 points] Functions with the following interface:
  `[reducedColorImage,reducedEnergyImage] = reduceWidth(im, energyImage)`
  `[reducedColorImage,reducedEnergyImage] = reduceHeight(im, energyImage)`
  These functions should take as inputs a) a 2D matrix `energyImage` of datatype double and b) a nrows-x-ncols-x-3 matrix `im` of datatype uint8. The input `energyImage` should be the output of the `energy_image` function. The output must return 2 variables: a) a 3D matrix `reducedColorImage` same as the input image, of datatype `uint8`, but with its width or height reduced by one pixel; b) a 2D matrix `reducedEnergyImage`, of datatype `double`, the same as `energyImage`, but with its width or height reduced by one pixel. Edit: One way to do this, for a vertical seam, is by creating a new image, then going row by row, reading in the row from the original image, removing one pixel (which is part of your seam) from it, and putting that pixel-removed row in the new image. The solution is analogous for a horizontal seam. You can remove pixels in a vector by setting them to the empty vector `[]`.

Matlab hints:

- Useful functions: `imfilter, im2double, fspecial, imread, imresize, rgb2gray, imagesc, imshow, subplot;`
- Use `saveas(gcf, '[filename].png')` to save your images. You can also save images directly without displaying them, by turning the 'visibility' of the figure in which they're shown 'off'.
- Be careful with double and uint8 conversions as you go between computations with the images and displaying them -- filtering should be done with doubles.

Answer each of the following, and include image displays where appropriate. Save your code for each of the parts below as a separate .m script. Name the scripts `part_a.m`, `part_b.m`, `part_c.m`, and `part_d.m`. Submit the scripts along with the functions above. Also submit the image results, as indicated below. *Your grader should be able to run your scripts and get the image results you submitted*.

a. [5 points] Run your `reduceHeight` function on the provided `prague.jpg` and shrink the height by 100 pixels. Then run your `reduceWidth` function on the provided `mall.jpg` and shrink the width by 100 pixels. Also show what standard image resizing would do (use `B = imresize(A, [numrows numcols])`). Display the outputs, save them, and submit

them.

b. [5 points] Display, save and submit (i) the `energy_image` function output for the provided images `prague.jpg` and `mall.jpg`, and (ii) the two corresponding cumulative minimum energy maps (M) for the seams in each direction (use the `imagesc` function).
c. [5 points] For the same two images, display, save and submit the original image overlaid with (a) the first selected horizontal seam and (b) the first selected vertical seam.
d. [15 points] 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 ("failure cases"). 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. If you have a large image, try `imresize` to resize the image to a small size to make the computations faster and allow you to try more ways of transforming the image. For each result, include the following things, clearly labeled (see `title` function):
    - the original input image,
    - your system's resized image,
    - the result one would get if instead a simple resampling were used (via Matlab's `imresize`),
    - the input and output image dimensions,
    - the sequence of ~~enlargements and~~ removals that were used (just the steps, e.g. remove 10 rows -> remove 20 columns -> remove 30 rows, not the intermediate image results), and
    - a qualitative explanation of what we're seeing in the output.

Acknowledgement: The author of this exercise is Kristen Grauman.