ECE 4984/5554: Computer Vision, Fall 2015
PS1

Instructor: Devi Parikh (parikh@vt.edu)
TA: Jiasen Lu (jiasenlu@vt.edu)

Due: Wednesday, September 16th, 11:55 pm

Instructions

1. Answer sheets must be submitted on Scholar. Hard copies will not be accepted.
2. Please submit your answer sheet containing the written answers in a file named: FirstName_LastName_PS1.pdf.
3. Please submit your code and input/output images in a zip file named: FirstName_LastName_PS1_mat.zip if using Matlab or FirstName_LastName_PS1_py.zip if using Python. Please do not create subdirectories within the main directory.
4. You may collaborate with other students. However, you need to write and implement your own solutions. Please list the names of students you discussed the assignment with.
5. For the implementation questions, make sure your code is bug-free and works out of the box. Please be sure to submit all main and helper functions. Be sure to not include absolute paths. Points will be deducted if your code does not run out of the box.
6. If plots are required, you must include them in your answer sheet (pdf) and your code must display them when run. Points will be deducted for not following this protocol.

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.
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]`?
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:* Assymetric filters must be flipped prior to convolution.)
4. Name two specific ways in which one could reduce the amount of fine, detailed edges that are detected with the Canny edge detector.
5. Describe a possible flaw in the use of additive Gaussian noise to represent image noise.
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.

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

For this exercise, you will implement a version 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". The paper is available off the course website. The goal is to implement the method, and then examine and explain 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.
Write Matlab or Python functions as below. Save each of the functions in a file called `<function-name>`.m(py) and submit all of them.

- `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). The input `im` should be a `MxNx3` matrix of datatype `uint8`. (It can be the output of `imread` on a color image.) The output should be a 2D matrix of dataype `double`.
- `cumulativeEnergyMap = cumulative_minimum_energy_map(energyImage,seamDirection)` - to compute minimum cumulative energy. The input `energyImage` should be a 2D matrix of datatype `double`. (It can be the output of `energy_image` function defined above.). The input `seamDirection` can be the strings '`HORIZONTAL`' or '`VERTICAL`'. The output must be a 2D matrix of datatype `double`.
- `verticalSeam = find_optimal_vertical_seam(cumulativeEnergyMap)` - to compute the optimal vertical seam. 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 must be a vector containing the column indices of the pixels which form the seam for each row.
- `horizontalSeam = find_optimal_horizontal_seam(cumulativeEnergyMap)` - to compute the optimal horizontal seam. 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 must be a vector containing the row indices of the pixels which form the seam for each column.
- `displaySeam(im, seam, type)` - to display the selected type of seam on top of an image. The input `img` should be an image of type `jpg`. `type` can be the strings '`HORIZONTAL`' or '`VERTICAL`'. `seam` can 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. *Hint:* The origin of the plot will be the top left corner of the image.
- 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 `MxNx3` matrix `im` of datatype uint8. The input `energyImage` can be the output of the `energy_image` function. The output must return 2 variables: `a)` a 3D matrix same as the input image but with its width or height reduced by one pixel; `b)` a 2D matrix of datatype `double` same as the input `energyImage`, but with its width or height reduced by one pixel.

Answer each of the following as indicated:

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.

(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.

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`

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

4. 10 points. For the same image `inputSeamCarvingPrague.jpg`, display the original image together with `(a)` the first selected horizontal seam and `(b)` the first selected vertical seam in your answer sheet. Explain why these are the optimal seams for this image.

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.

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.

   (b) your system's resized image,

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

   (d) the input and output image dimensions,

   (e) the sequence of removals that were used

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

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).

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.

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.

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.

5. Implement the greedy solution, and compare the results to the optimal Dynamic Programming solution.

Matlab hints:

1. Useful functions: `imfilter`, `im2double`, `fspecial`, `imread`, `imresize`, `rgb2gray`, `imagesc`, `imshow`, `subplot`.

2. To plot points on top of a displayed image, use `imshow(im);` followed by `hold on;` followed by `plot(...);`.

3. Be careful with `double` and `uint8` conversions as you go between computations with the images and displaying them.

Python hints:

1. Useful modules: `numpy`, `scipy`, `matplotlib`, `skimage`

2. `scipy` has different convolution functions: `scipy.ndimage.filters` and `scipy.signal.convolve`, see the difference and choose the appropriate one.

3. To plot points on top of a displayed image, use `matplotlib.pyplot.hold(True)`.

4. Be careful with `double` and `uint8` conversions as you go between computations with the images and displaying them.

---

1. This assignment is adapted from the PS1 assignment from Kristen Grauman's CS 376: Computer Vision at UT Austin.
2. Image acknowledgements: Thanks to the following Flickr users for sharing their photos under the Creative Commons license: inputSeamCarvingMall.jpg is provided by hey tiffany! inputSeamCarvingPrague.jpg david.nikonvscanon.
3. Be sure to credit any photo sources.