MSML640: Computer Vision
Assignment 2

Instructions

1. Answer sheets, code and input/output images must be submitted on Canvas.
2. Please provide a pdf version of your answer sheet named: FirstName_LastName_ASN2.pdf.
3. Please use '.py' as file extension for Python scripts to submit.
4. Please put all your code, input/output images and answer sheets in a folder (no subdirectories). 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.
5. 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.
6. Your code and plots should use the same filenames mentioned in the question (if present). Variables in your code should use the same names that are mentioned in the question (if present).
7. Please make sure that the folder is named: LastName_FirstName_ASN2_py.
8. Zip the above folder and name the zipped file LastName_FirstName_ASN2_py.zip. Submit only the zip file.

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]` is used as the filter to compute 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 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 Python functions as below. Save each of the functions in a file called `<function-name>`.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.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.
   (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. 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.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 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 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.