# ECS 174: Computer Vision, Spring 2019
# Problem Set 1

Instructor: Yong Jae Lee (yongjaelee@ucdavis.edu)
TA: Xueyan Zou (zxyzou@ucdavis.edu)
TA: Yangming Wen (ymnwen@ucdavis.edu)
TA: Wei-Pang (Tyler) Jan (wjan@ucdavis.edu)

Due: Wednesday, May 1st, 11:59 PM

**Instructions**

1. Answer sheets must be submitted on Canvas. 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.zip. Please do not create subdirectories within the main directory.
4. You may complete the assignment individually or with a partner (i.e., maximum group of 2 people). If you worked with a partner, provide the name of your partner. **We will be using MOSS to check instances of plagiarism/cheating.**
5. For the implementation questions, make sure your code is documented, 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.

## I. Short answer problems [24 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: `[1 0 1 1 0 0 0 1]`. What is the result of dilation with a structuring element `[1 1 1]`?
3. Describe a possible flaw in the use of additive Gaussian noise to represent image noise.
4. 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.

## II. Programming problem: content-aware image resizing [76 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 code** with functions that can do the following tasks. Save each of the below functions in a file called `<function-name>.m` and submit all of them.

- `energyImg = energy_img(im)`- to compute the energy at each pixel using the magnitude of the x and y gradients (equation 1 in the paper; sqrt(dx^2+dy^2)). The gradients should be computed with the grayscale converted image. You do not need to smooth the image before computing the gradients. The input `im` should be an MxNx3 matrix of datatype uint8. (It can be the output of `imread` on a color image.) The output `energyImg` should be a 2D matrix of datatype double.

- `cumulativeEnergyMap = cumulative_min_energy_map(energyImg,seamDirection)`- to compute minimum cumulative energy. The input `energyImg` should be a 2D matrix of datatype double (the output of `energy_img` function defined above). The input `seamDirection` must be the strings 'HORIZONTAL' or 'VERTICAL'. The output `cumulativeEnergyMap` must be a 2D matrix of datatype double.

- `verticalSeam = find_vertical_seam(cumulativeEnergyMap)` - to compute the optimal vertical seam. The input `cumulativeEnergyMap` should be a 2D matrix of datatype double (the output of the `cumulative_min_energy_map` function defined above). The output `verticalSeam` must be a vector containing the column indices of the pixels which form the seam for each row.

- `horizontalSeam = find_horizontal_seam(cumulativeEnergyMap)`- to compute the optimal horizontal seam. The input `cumulativeEnergyMap` should be a 2D matrix of datatype double (the output of the `cumulative_min_energy_map` function defined above). The output `horizontalSeam` must be a vector containing the row indices of the pixels which form the seam for each column.

- `view_seam(im,seam,seamDirection)`- to display the selected type of seam on top of an image. The input `im` should be an MxNx3 matrix of datatype uint8. `seam` can be the output of `find_vertical_seam` or `find_horizontal_seam`. `seamDirection` can be the strings 'HORIZONTAL' or 'VERTICAL'. The function 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:
  `[reducedColorImg,reducedEnergyImg] = decrease_width(im,energyImg)`
  `[reducedColorImg,reducedEnergyImg] = decrease_height(im,energyImg)`
  These functions should take as inputs (a) an MxNx3 matrix `im` of datatype uint8 and (b) a 2D matrix `energyImg` of datatype double. The input `energyImg` can be the output of the `energy_img` function. The output must return 2 variables: (a) a 3D matrix `reducedColorImg` same as the input image but with its width or height reduced by one pixel; (b) a 2D matrix `reducedEnergyImg` of datatype double same as the input `energyImg`, but with its width or height reduced by one pixel.

**Answer each of the following**, and include image displays where appropriate:

1. [11 points] Write a script called `seam_carving_decrease_width.m` which does the following by using the functions defined above:
   a) Loads a color input image called `inputSeamCarvingPrague.jpg`. Download the image from http://web.cs.ucdavis.edu/~yjlee/teaching/resources/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 http://web.cs.ucdavis.edu/~yjlee/teaching/resources/inputSeamCarvingMall.jpg. Save the output as `outputReduceWidthMall.png`. Submit it. Display the output in your answer sheet.

2. [11 points] Repeat the above steps for both input images, but reduce the height by 50 pixels. Call the script `seam_carving_decrease_height.m`, and save the output images as `outputReduceHeightPrague.png` and `outputReduceHeightMall.png`, respectively. Display both outputs in your answer sheet. Submit the script for image `inputSeamCarvingPrague.jpg`

3. [11 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 `imagesc` function). Explain why these outputs look the way they do given the original image's content.

4. [11 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. Explain why these are the optimal seams for this image.

5. [11 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 first horizontal and vertical seams for `inputSeamCarvingPrague.jpg` and explain the impact. You need not submit this code.

6. [21 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 (be sure to credit any photo sources). Include an example 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 (see `title` function) and using the `subplot` function for displaying the images:

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`),
d) the input and output image dimensions,
e) the sequence of enlargements and removals that were used, and
f) a qualitative explanation of what we're seeing in the output.

## III. [OPTIONAL] Extra credit [up to 10 points each, max possible 10 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 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` functions are useful to get mouse clicks or draw polygons).

2. 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 (see Matlab's `rgb2hsv` function to map to HSV color space). Think about how to translate those values into energy function scores.

3. 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.

4. Implement the greedy solution, and compare the results to the optimal DP solution.

**Matlab hints:**

- Useful functions: `imfilter, im2double, fspecial, imread, imresize, rgb2gray, imagesc, imshow, subplot`.
- To plot points on top of a displayed image, use `imshow(im);` followed by `hold on;` followed by `plot(…);`.
- Be careful with double and uint8 conversions as you go between computations with the images and displaying them.[1]

---

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 by david.nikonvscanon.