CS376: Computer Vision: Assignment 1 Due: Sept. 16th, 11:59 PM

**Format for writeup:** You may use any tool for preparing the assignment write-up that you like, as long as it is organized and clear, and figures are embedded in an easy to find way alongside your descriptive text. **Submission:** See the end of this document for submission instructions.

**Assignment questions:** Please see Piazza for questions and discussion from the class.

# I. Short answer problems [30 points, 6 points each]

The answer to each question is within 3 to 5 sentences.

1. What are the three applications of image filtering?

Image filtering is used to enhance images by reducing noise, resizing, increasing level of detail, etc. It is also used to extract information from an image by detecting properties such as textures, edges, features, etc. Finally, it is used for template matching, which enables pattern detection. These applications allow filtering to be a building block for neural networks.

1. What is the difference between the mean filtering and the median filtering?

Mean filtering averages the input pixels to get the pixel value that all the other pixels in the input set will be reassigned to. Median filtering picks the median pixel (the pixel in the middle when they are ordered by value) to be the value that the input pixels will be reassigned to.

1. In class, we talked about image smoothing followed by computing image gradients. Is it identical to computing image gradients first and then perform image smoothing on the resulting image gradients?

No, the order of smoothing and computing gradients matters. If an image is smoothed before the gradient is computed, rates of change (derivatives) of pixel intensities in any direction will be reduced (sharp lines become blurrier). This would result in a gradient with a smaller magnitude than a gradient calculated before smoothing the image.

1. How to take the advantage of the separability of a filter for fast image filtering calculation?

If a filter is separable, separate convolutions can be run in each dimension for each pixel. In the case of two dimensional images, this means that instead of running a two dimensional convolution, which would take K^2 operations per pixel (where k is the width or height of the kernel), we can run two 1-dimensional convolutions, resulting in a total of 2K operations per pixel.

1. In non-maximum suppressing, we detect the maximum pixel along the image gradient direction. Provide examples where this approach is sub-optimal. You can draw illustrations or provide results on real examples. Please provide a short justification (2-3 sentences) on why this is the case.

In cases where the image has objects with broad edges or borders, non-maximum suppressing will thin them to an undesirable degree and possibly even create two edges for a single border. Consider the rectangle below. It has a frame of a different color than the content on the inside.

Without non-maximum suppression, we would get a single rectangle as the detected edges:

With non-maximum suppression, we would likely end up with two thinner, concentric rectangles representing the inner and outer edges of the frame.

**Extra credit (5points).** So far we have covered filtering and edge detection for images. Please mention



Figure 1: A typical result of seam carving. (Left) Input image. (Right) Output image.

how to extend the idea to videos. Please discuss how to de-noise in both the spatial and/or temporal domain, how to compute gradients in the spatial and/or temporal domain, and how to detect ”edges” in the spatial and/or temporal domain.

The de-noising, gradient calculations, and edge detection techniques remain the same in the spatial domain for both photos and video—just apply the same techniques to each individual frame. To handle the temporal domain, we can take the results of the calculations on each individual frame and use techniques similar to those used in the spatial domain compare adjacent frames to each other. We can smooth in the temporal domain by averaging the intensity of every pixel with the intensities of the pixels in the same location in the next frame and gradients can be computed by examining the changes in intensity of each pixel with the pixels in the same locations in the next frame. Edge detection would follow the gradient calculations, which would allow them to factor in both the spatial and temporal domains.

# II. Programming problem: content-aware image resizing [70 points]

For this exercise, you are asked to implement a version of the content-aware image resizing technique de- scribed in Shai Avidan and Ariel Shamir’s SIGGRAPH 2007 paper, ”Seam Carving for Content-Aware Image Resizing” (See Figure [1](#_bookmark0) for one result). The goal is to implement the method, and then examine and explain its performance on different kinds of input images.

As we discussed in this class, the key idea of this approach is to use dynamic programming to select the optimal seams to insert/delete to modify an input image.

Write Matlab code with functions that can do the following tasks:

Compute the energy function at each pixel using the magnitude of the x and y gradients (equation 1 in the paper)

*•*

* + Compute the optimal vertical seam given an image
  + Compute the optimal horizontal seam given an image
  + Reduce the image size by a specified amount in one dimension (width or height decrease)
  + Display the selected seam on top of an image
  + Functions with the following interface:

[output] = removeVertical(im, numPixels) [output] = removeHorizontal(im, numPixels)

These functions take an input image im, and a parameter specifying how many seams to carve, from

the width or height,respectively. The image im will be a *h w* 3 uint8 matrix, which is what imread returns for a color image.Put these functions in file named removeVertical.m and removeHorizontal.m

*× ×*

Set up scripts so that you can play with the seam removal and specify different combinations of horizontal and vertical removals. Apply your system to the provided images. View the results in color, but note that the gradients should be computed with the grayscale converted image.

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–filtering should be done with doubles.

*•*

Use toy examples to debug! For example, create an image with all Gaussian noise, while leaving three vertical seams right. Your result shall return that vertical seam. As another example,

*•*

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

(10 points) Run your removeVertical function on the provided ut.jpg with numPixels = 100 (in other words, shrink the width by 100 pixels). Run your removeHorizontal function on the provided river.jpg with numPixels = 100 (in other words, shrink the height by 100 pixels). Display the outputs.

*•*

Graphical user interface, website

Description automatically generated

(10 points) Display (a) the energy function output (total gradient magnitudes *e*1(*I*)) for the provided image ut.jpg, and (b) the two corresponding cumulative minimum energy maps (M) for the seams in each direction (vertical and horizontal, use the imagesc function) for the image ut.jpg. Explain why these outputs look the way they do given the original image’s content.Graphical user interface

Description automatically generated

*•*

Since the energy function considers both the vertical and horizontal rates of change at each point, edges (at which these rates of change are generally greatest) tend to be outlined. The vertical cumulative energy calculation accumulates the energy as it examines the image downward, so areas under the greatest rates of change, i.e. under the most concentrated collections of edges are marked more vibrantly. The same applies to the horizontal cumulative energy moving left to right—areas to the right of the most concentrated collections of edges are marked more vibrantly.

(10 points) For the same image ut.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.

*•*

A picture containing outdoor, sky, building, people

Description automatically generated

These are the optimal seams for this image because they traverse parts of the image that have the fewest and the least sharp edges. The energy function is smallest along these paths, which indicates that there is very little change in colors and intensity along them. This means that removing them is less likely to affect the content of the image than other parts of it.

(10 points) Make some change to the way the energy function is computed (i.e.,filter used, its param- eters, or incorporating some other a priori knowledge). Display the result and explain the impact on the results for some example.

*•*

A picture containing timeline

Description automatically generated

Changing the energy function from the sum of the derivatives to the sum of the squares of the derivatives reduced the amount of sharp lines in it. Notably, in comparison to the original energy function calculation, the clouds in the top right corner of the ut.jpg image are no longer outlined. The dramatic reduction in the amount of relatively high energy lines causes a in the amount of relatively high cumulative energies as well, while the areas of high energy accumulation are more dramatical higher energy than their surroundings.

(30 points) Now, for the real results! Use your system with different kinds of images and seam combi- nations, 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 creative when generating the results. Look into the original seam carving paper for inspiration. Note that you are allowed to hand-craft the feature map, e.g., by removing an object or prevent pixels of an object from removing.

*•*

For each result, include the following things, clearly labeled (the title may be useful here):

* (a) the original input image,
* (b) your system’s resized image,
* (c) the result one would get if instead a simple resizing was used (this is considered the baseline comparison, see MATLAB function: imresize),
* (d) the input and output image dimensions,
* (e) the sequence of enlargements and removals that were used (i.e. the inserted or removed seam(s) you computed), and
* Graphical user interface, application, Word

  Description automatically generated(f) a brief qualitative explanation of what we’re seeing in the output.

These figures show how my seam carving system works with extreme cases of resizing, i.e. removing about half or more of the pixels from the original image. The left column shows these removals on the ut.jpg image. These changes show how some of the problems with how the system decides what the “content” in “content preserving image resizing” is. Backgrounds/foregrounds that may not necessarily be the focus of the image may end up getting prioritized over the focus of the image because these unimportant sections have more edges. The people in the foreground and the clouds in the top right corner seem to be taking high priority and preserving them distorts the tower. This system doesn’t remove objects in the scene, it simply distorts them to fit them in the image. The imresize, on the other hand, does not make any attempt at distinguishing between parts of the image to find what the objects are. It simply distorts every part of the image the exact same way, thus preserving all the content with the image along with all the unimportant parts of it. This can be very useful when the ratio of the dimensions remains locked.

A picture containing text

Description automatically generated

I was also curious to see how my system handles text. It seems that text is always “preserved” in that regardless of the absurd amount of removals performed, an outline for every single letter remains. Because the English language reads horizontally, cumulative energy calculated horizontally is much greater around the text than cumulative energy calculated vertically. This means that text is more likely to undergo greater distortion when vertical seams are removed as opposed to horizontal seams.

# Submission Instructions

The files you need to submit:

* + A **PDF** write-up that contains answers for Part I. and Part II.
  + MATLAB code (.m file)

Compress all the files (code, PDF writeup with images) into a **zip** file and upload the **zip** file to Canvas.