Intro to Image Understanding (CSC420)

Assignment 1

Posted Sept 23rd, 2016; Submission Deadline: Oct. 4th 11.59pm 2016

Instructions for submission: Please write a document (either pdf, doc, etc) with your solutions (include pictures where needed). Include your code inside the document, and submit through **MarkUS**.

Max points: 13, max extra credit points: 3

1. (a) **[2 points]** Write your own code for computing convolution of the 2D (grayscale) image and a 2D filter. Make the output matrix be the same size as the input image. Be careful to correctly deal with the border of the image. The easiest way to do this is to "zero-pad" the image prior to convolution.

   (b) **[1.5 points]** Is it possible to write a convolution in one pixel as a dot product between two vectors? Is it possible to write the full convolution between the image and the filter via matrix multiplication?

2. (a) **[1 points]** Given a $n \times n$ image, I, and $m \times m$ filter, h, what is the computational cost of computing $h \times I$ (the convolution)? What is the computational cost if h is a separable filter?

   (b) **[1 points]** If I first convolve an image with a Gaussian filter with $\sigma = 3$, and then convolve the output with a Gaussian with $\sigma = 8$, this gives an equivalent result as if I just convolve the image with a Gaussian with what σ?

   (c) **[1 points]** Write your own function that creates an anisotropic Gaussian filter with σ_x and σ_y as an input parameter.

   (d) **[1 points]** Convolve the attached cat.jpg with a (2D) Gaussian filter with $\sigma_x = 15$ and $\sigma_y = 2$ and visualize the result (display the result of the convolution). You can use built-in functions for convolution.

   (e) **[1 points]** The Gaussian filter in question 2.d is separable. How can you use this fact to speed up convolution? What are the vertical and horizontal filters? No need to write code.

3. (a) **[1.5 points]** Compute magnitude of gradients for the attached images templateNoise.png and waldoNoise.png using convolution at three different scales.

   (b) **[1 points]** Write a function that localizes the template (templateNoise.png) in the image waldoNoise.png based on the magnitude of gradients. You can help yourself using **findWaldo.m**.

4. (a) **[1 points]** Run the Canny edge detector on tennisCourt.jpg. Play with the parameters so that you get rid of low-contrast edges. In Matlab you can help yourself with function **edge**.

   (b) **[1 points]** Any idea how you could find the bounds of the court in the image? No need to write code, just brainstorm!

5. **[3 points] total** (this is an optional exercise) Implement seam carving using **Dijkstra's algorithm** for shortest paths:

   (a) Compute magnitude of gradients of an image

   (b) Build a directed image graph with a source and a sink

   (c) Find the connected path of pixels that has the smallest sum of gradients. A path is valid if it is connected (the neighboring points in the path are also neighboring pixels in the image), it starts in the first row of the image and in each step continues one row down. It finishes in the last row of the image. *Hint: It is easier to restrict your graph, than to modify Dijkstra's to exhibit this property.*

   (d) You do not have to write your own Dijkstra's algorithm. In Matlab you can use **graphshortestpath**.

   (e) Remove the pixels in the path from the image. This gives you a new image with one column less.

   (f) **[2 points]** Remove a few paths with the lowest sum of gradients. Create (fun!) examples with a few of your own images.

   (g) **[1 points]** Could you use this algorithm (with minor modifications) to segment the bull in cattle.png? A segmentation is a closed path following the boundary of the object. You can use mouse-clicks, e.g. **ginput**, to specify initial points on the object boundary. You will need at least two points, try to get it working with as few as possible.

1 Sources

1. cat.jpg Osamu, Uchida, `https://www.flickr.com/photos/13975275@N00/`

2. tennisCourt.jpg Michael, Coghlan, `https://www.flickr.com/photos/mikecogh/`