Intro to Image Understanding (CSC420)

Assignment 1

Posted Sept 19th, 2017; Submission Deadline: *11.59pm ET Sept 27th* 2017

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: 12, max extra credit points: 3

[For problem 1 the supplementary material on convolution would be very helpful]

1. (a) [**0.5 point**] Using the concept of a 1D impulse $\delta(t)$ (centered at the origin), how would you extend this notation to write a 2D impulse which is a function of two variables u and v, and is centered at the 2D origin.(u and v represent the 2D spatial coordinates).

   (b) [**0.5 point**] The above impulse was located at the origin. Using the concept of shifting, how would you write a 2D impulse located at $u = m$, $v = n$?

   (c) [**2 points**] In the language of your choice, build a 100×200 image (i.e., height=100, width=200), such that the image has impulses at the following coordinates: (10,20), (20,40), (30,60), (40,80), (50,100), (60,80), (70,60), (80,40), (90,20). First coordinate represents the row index, while the second represents the column index. After building the matrix, visualize the image in 3D (In Matlab, see the function 'surf'; In Python see 'matplot3d'). Include both the code and plot in the assignment PDF.

   (d) [**1 point**] We have seen in class that a 1D function $f(t)$ of length L can be written as a sum of weighted (and shifted impulses), i.e., $f(t) = \sum_{l=1}^{L} f(t_l)\delta(t - t_l)$. How would you write a 2D image as weighted sum of shifted 2D impulses. Assume the size of the image to be $M \times N$.

2. (a) [**0.5 point**] Given a $n \times n$ image, I, and $m \times m$ filter, h, what is the computational cost of computing $h * I$ (the convolution)?

   (b) [**0.5 point**] What is the computational cost if h is a separable filter?

   (c) [**1 point**] Figure out if the filters specified by the following kernels are separable or not. If separable, write down the constituent horizontal and vertical filters.

$$\mathbf{F}_1 = \begin{bmatrix} 10 & 40 & 8 \\ 5 & 3 & 5 \\ 12 & 5 & 12 \end{bmatrix}; \quad \mathbf{F}_2 = \begin{bmatrix} 6 & 3 & 6 \\ 2 & 1 & 2 \\ 6 & 3 & 6 \end{bmatrix}$$

3. (a) [**Problem Statement**] Temperature sensing is a central component of modern processing plants in many industries. With the proliferation of Japanese electronics in the later part of the last century, many plants installed thousands of temperature sensors with (at the time) state of the art seven segment read outs (Many of you are perhaps not old enough to remember the calculators). You are in charge of renovating one such plant which processes milk to feed orphaned baby elephants! The plant is in tip-top working condition, all the sensors being rugged still work, and the readouts continue to provide accurate measurements.

   The problem is without dismantling the legacy sensors and displays you are required to post the data on the cloud for a data analytics engine. You being a seasoned computer vision expert, bring on the table an outrageously affordable solution using a less than 150 buck quadcopter drone equipped with a camera. The drone comes pre-programmed to periodically visit all the displays, take a picture, and then tweet it. You are given one such picture (thermometer.png). You are required to adapt the findWaldo function to get a list of digits in the display.

   Please notice that digits appear in 3 sizes in the display. For your convenience we have provided you with templates for all the digits at 3 different scales in the directory 'DIGITS', which has three sub-directories, i.e. there are in all 30 templates/filters. It will be easiest if MATLAB is used. It will be helpful if you read the steps very carefully. Several implementation tips are given in the following instructions. Good luck!

   Following are the steps you need to perform :

   (b) [**0.25 point**] Read in the templates. Code to read in templates is provided as the function readInTemplates, which returns a cell array containing 30 templates along with their dimensions.

   (c) [**1.5 points**] Compute normalized correlation (using normxcorr2) of the display image (thermometer.png) with each of the templates, and store it in a ($M \times N \times 30$) array (this will greatly simplify the later steps); $M \times N$ is the size of the input image. Let's name this array as 'corrArray'. Advice for this step follows. Please read carefully.

      i. Don't forget to convert the image and the templates into grayscale and cast them into double.

      ii. The output of the function normxcorr2 is bigger than the input image by offsets on both sides, and in both directions. If tH and tW are the height and width of the respective template, while M and N represent the height and width of the image, extract the central portion of the output as follows. Define two offset variables as offSetX = round(tW/2), and offSetY = round(tH/2). Then extract the portion (offSetY : offSetY + M - 1, offSetX : offSetX + N-1). This has to be done for the output of correlation with every template to ensure the output is always the same size, and the sizes of all the templates are not the same. It is for this reason that we stored the template dimensions separately while reading the templates.

(d) [**0.25 point**] The result of every correlation is an image itself. At every pixel check which template gave the maximum correlation. A naive implementation of this step will take ages. It is for this reason you were advised to maintain the outputs of correlation in a multidimesional array, so that you can make good use of operations along a specific dimension of the array. For instance if you had stored the results in corrArray, then you need to find the maximum along the third dimension i.e. use [maxCorr, maxIdx] = max(corrArray,[],3). Python users: please look up numpy function 'amax'.

(e) [**0.25 point**] By this time, you know at every pixel which template gave the maximum correlation. But it still might not be a digit, because obviously every pixel does not represent a digit. Find the pixels for which maxCorr exceeds a threshold T (use a threshold between 0 and 1, more on this later).

(f) These pixels are candidate digit locations. But still there would be multiple locations around the digit that exceed the threshold. So reject those pixels which are not a local maximum in terms of correlation. Specifically,

i. [**0.25 point**] Let's call the coordinates of candidate pixels (output of Step-e) as candX, candY. Loop through candX and candY. For every (candX, candY), check which template produced maximum correlation. This information can be retrieved from maxIdx(candY(i), candX(i)), where i is i-th candidate pixel. Note the order of x and y is reverse. Now you have the templateIndex for the template that gave the maximum correlation for the candidate location.

ii. [**0.25 point**] Extract the correlation matrix for the above templateIndex i.e. thisCorr = corrArray(:,:, templateIndex)

iii. [**2 points**] Then within 'thisCorr' matrix check if candX(i) and candY(i) is a local maximum in a 3 x 3 window. i.e. write a function isLocalMaximum(x, y, thisCorr), which checks if thisCorr(y,x) is equal to the maximum value in the window centered around x,y.

iv. [**0.25 point**] If candX(i), candY(i) pass the above test, draw a bounding box around x,y equal to the size of the template corresponding to template index. Code for this is provided as drawAndLabelBox(x,y,templateIndex , dimensions). Because you would be calling this function in a loop, it is a good practice to call the function 'drawnow' to refresh the display after every iteration, otherwise one gets an impression that the screen is frozen!

v. [**1 point**] Try out various values for the threshold T. You won't get perfect result, but your goal should be to set T such that (i) no non-digit is detected as a digit, (ii) No digit is labeled incorrectly, (iii) it is okay to miss a digit. Report your threshold, and a screenshot of detected bounding boxes. Report the number of correctly labeled digits.

vi. [**Extra Credit 3 points**]. ***Getting 1 of these 3 points is easy. Read carefully*** Can you improve the digit detection by using templates from the input image. i.e. try to crop your own templates for some of the digits from the image, and then re-run the code with new templates. Why is this expected to improve the performance? You will get 1 point if you correctly answer (in a sentence or two) why the performance is expected to improve, even if you don't implement it!