CS 131 Computer Vision: Foundations and Applications (Fall 2014) Problem Set 1

Due Friday, October 24, 5 PM

1 Normalized Cross-correlation

In class, we covered cross-correlation, in which a template image is multiplied with sections of a larger image to measure how similar each section is to the template. *Normalized cross-correlation* is a small refinement to this process. More specifically, before multiplying the template with each small section of the image, the section is scaled and offset so it has zero mean and variance of 1. This increases accuracy by penalizing image sections which have high intensity but do not match the pattern of the template. In MATLAB, normalized cross-correlation is implemented by the function `normxcorr2`.

(a) Use the provided `crossCorrelation.m` file to load the provided photo and template. Read the MATLAB documentation for `normxcorr2`, and use it to perform cross-correlation to find the section of the image that best matches the template. Include your MATLAB code in your writeup, and describe why the peak occurs where it does. Also explain the straight-line artifacts you observe in the cross-correlation *(Hint: look at the template and the original image)*. You don't need to include the cross-correlation image itself in your report, as it may not print well.

(b) In `crossCorrelation.m`, fill in "part (b)" to perform cross-correlation using the larger template. Note that the larger template does not exactly match the image. Describe your results, and why they are different from part (a). What does this tell you about the limitations of cross-correlation for identifying objects in real-world photos? Notice that the code we provide auto-scales image brightness so that it covers the full range. Please remember that the color white doesn't denote the same value in the two cross-correlation images.

(c) Above, we saw that cross-correlation can be fragile. One way to make it less fragile is to perform cross-correlation using many templates to cover the different ways an object may appear in an image. Suppose we wish to search for N_R possible rotations of an object at N_S possible sizes. Assume the image is size $n \times n$ and the template is roughly size $m \times m$. How many mathematical operations will the entire search require? Here, we're looking for a "Big-O Notation" estimate. In other words, you may neglect constant factors, such as the effects of image edge padding and smaller terms.

$O(N_R N_S n^2 m^2)$

2 Linear Filters

In class, we introduced 2D discrete space convolution. Consider an input image $I[i, j]$ and a filter $F[i, j]$. The 2D convolution $I * F$ is defined as

$$(I * F)[i, j] = \sum_{k,l} I[i - k, j - l]F[k, l] \tag{1}$$

(a) Convolve the following I and F. Assume we use **zero-padding** where necessary.

$$I = \begin{bmatrix} 2 & 0 & 1 \\ 1 & -1 & 2 \end{bmatrix} \qquad F = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \tag{2}$$

Important: Please DO NOT use Matlab for this question. You must show all necessary steps here. It will also be helpful for answering question (d).

(b) Note that the F given in (2) is *separable*; that is, it can be written as a product of two 1D filters: $F = F_1F_2$. Here, we have

$$F_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad F_2 = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

Compute $(I * F_1)$ and $(I * F_1) * F_2$, i.e. first perform 1D convolution on each column, followed by another 1D convolution on each row. *Important: Please DO NOT use Matlab for this question. You must show all necessary steps here. It will also be helpful for answering question (d).*

(c) Prove that for any separable filter $F = F_1F_2$,

$$I * F = (I * F_1) * F_2.$$

Hint: Expand equation (1) directly.

(d) Carefully count the *exact* number of multiplications (multiplications only, including those multiplications due to zero-padding) involved in part (a) and part (b). Which one of these requires fewer operations? *Important: We are asking for two exact numerical values here. We will not accept any approximations. You may find the computation steps you wrote down for (a) and (b) helpful here.*

(e) Consider a more general case: I is an M_1 by N_1 image, and F is an M_2 by N_2 separable filter.

(i) How many multiplications do you need to do a direct 2D convolution?

(ii) How many multiplications do you need to do 1D convolutions on rows and columns?

Hint: For (i) and (ii), we are asking for two functions of M_1, N_1, M_2 and N_2 here. We will not accept any approximations.

(iii) Use Big-O notation to argue which one is more efficient in general: direct 2D convolution or two successive 1D convolutions?

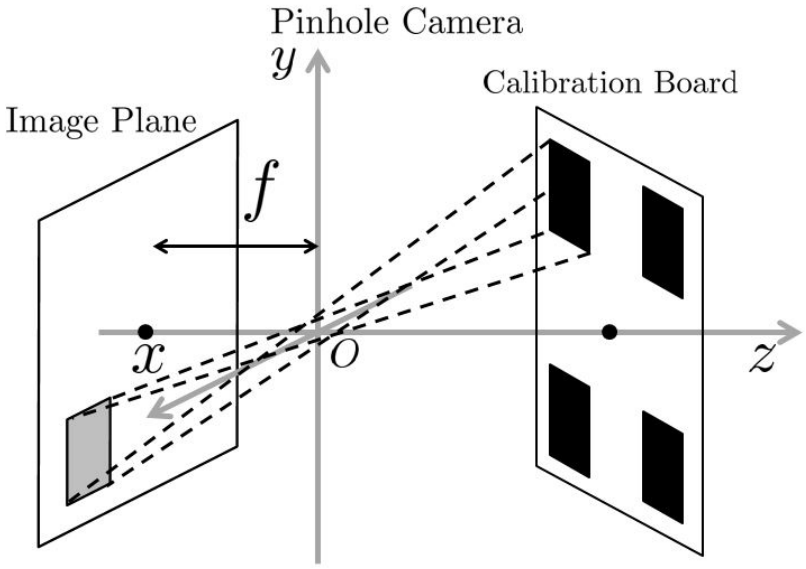

Figure 1: Pinhole camera model.

3 Pinhole Camera Model

In class, we discussed the pinhole camera model. In this problem, we will calibrate a pinhole camera. We have provided an illustration for pinhole camera model in Figure 1, where O is the location of the pinhole. The *focal length*, i.e. the distance from the image plane to the pinhole, is f.

(a) Suppose that we have a point (x, y, z) in the coordinate system defined by the pinhole and that the image plane origin is aligned to the pinhole. What's the corresponding point in the image plane? *Hint: You can use f, x, y and z to represent the corresponding point.*

(b) Suppose we have a calibration board (or a checkerboard) as in Figure 2. Each black square on the checkerboard has an area of S. Assume the image plane and the calibration board are parallel, face each other, and the distance between the calibration board and the pinhole is L. What is the area of each black square in the image?

(c) Now suppose that the image plane origin is no longer aligned to the pinhole, which is (c_x, c_y). Also, suppose we have a point (x, y, z) in the coordinate system defined by the pinhole. What's the corresponding point in the image plane? *Hint: You can use c_x, c_y, f, x, y and z to represent the corresponding point.*

(d) Continue from part (c). We have our pinhole camera parameters f, c_x and c_y. Suppose these parameters are unknown and we want to measure them by taking photos of our calibration board. The calibration process works as follows:

(i) Place the checkerboard parallel with the image plane at some distance L.

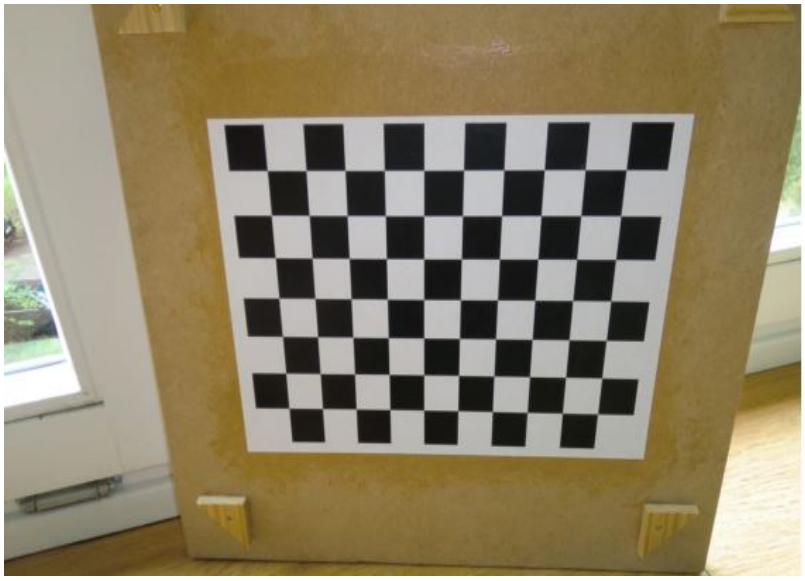

Figure 2: A sample calibration board.

(ii) Mark some corners (or some distinctive points on the checkerboard), measure each point's location (x, y, L) on the checkerboard, and then take a photo.

(iii) Find the corresponding points in the image and measure their location in the image.

What's the minimal number of points to measure in order to get the pinhole camera parameters f, c_x and c_y? Explain your reasoning.

(e) Continue from part (d). Is it possible to get the pinhole camera parameters f, c_x and c_y with our checkerboard fixed (i.e. without moving the checkerboard during the calibration process)? Justify your answers.

4 Canny Edge Detector

(a) Suppose the Canny edge detector successfully detects an edge. This detected edge (shown as the red horizontal line in Figure 3) is then rotated by θ, where the relationship between a point on the original edge (x, y) and a point on the rotated edge (x', y') is defined as

$$x' = x\cos\theta \qquad y' = x\sin\theta$$

Will the rotated edge be detected after the rotation using the same Canny edge detector? Provide either a mathematical proof or a counter example. *Hint: The detection of an edge by the Canny edge detector depends only on the magnitude of its derivative. The derivative at point (x, y) is determined by its components along the x and y directions. Think about how these magnitudes have changed because of the rotation.*

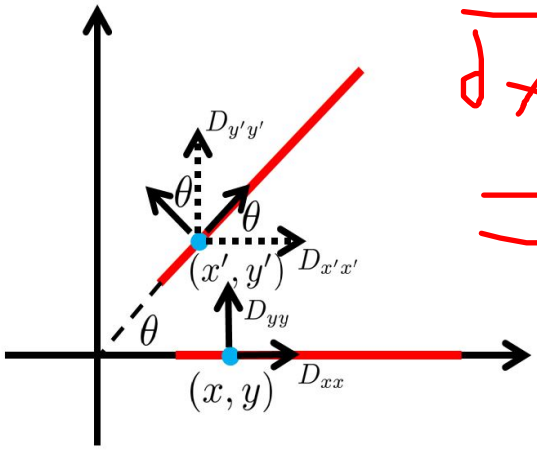

Figure 3: A detected edge and its rotation.

(b) After running the Canny edge detector on an image, you notice that long edges are broken into short segments separated by gaps. In addition, some spurious edges appear. For each of the two thresholds (low and high) used in hysteresis thresholding, state how you would adjust the threshold (up or down) to address both problems. Assume that a setting exists for the two thresholds that produce the desired result. Explain your answer very briefly.

5 RANSAC for Fitting Circles

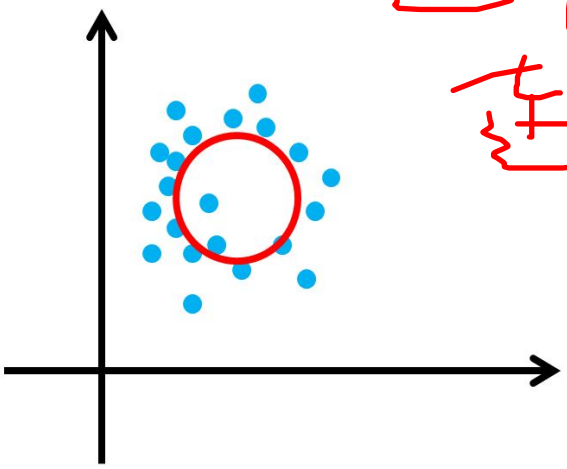

Figure 4: Fitting a circle to a group of 2D points.

In class, we discussed how to fit a line to a series of points using RANSAC. In this problem, you are going to develop an algorithm that uses RANSAC for fitting a circle to a group of points in two dimensional space $\{(x_i, y_i)\}_{i=1}^n$ (see Figure 4). *Important: Please submit your code and plots for this problem.*

(a) A circle $\mathcal{C}$ in 2D space is given by its center (c_x, c_y) and its radius R. Before we implement RANSAC, we need a way to fit a circle (c_x, c_y, R) to a chosen set of points (we will need at least 3 points to specify a circle, and with more points we can get an "average circle" that provides a better fit). Read the provided handout on least-squares model fitting. We will use least-squares to fit a circle to 3 or more points. Remember that, for each point i, a circle should satisfy the simple scalar equation:

$$R^2 = (x_i - c_x)^2 + (y_i - c_y)^2.$$

From basic algebra, this can be rewritten as

$$\begin{aligned} R^2 &= x_i^2 - 2x_i c_x + c_x^2 + y_i^2 - 2y_i c_y + c_y^2 \\ 2x_i c_x + 2y_i c_y + R^2 - c_x^2 - c_y^2 &= x_i^2 + y_i^2 \end{aligned}$$

Note that we our equations contain squares of unknown variables (R, c_x and c_y). Least squares can only handle linear equations, so this is a problem. However, we can define a new variable $q = R^2 - c_x^2 - c_y^2$. Using this trick, we have:

$$2x_i c_x + 2y_i c_y + q = x_i^2 + y_i^2$$

This is indeed a linear equation (note x_i^2 and y_i^2 are constants), so a set of these equations can be solved by least squares! Afterward, we can recover the value of R^2 from q.

Your task is to edit the provided `FitCircle.m` to set up a system of the above equations (one equation per input point) and solve the system using least squares. Run the provided `TestFit.m` to check your solution. The fit should be good on the first plot, but may be sensitive to outliers for the second plot. Include your code and the plots from a run of `TestFit.m` in your submitted solution.

(b) Using your `FitCircle()` function as a subroutine, complete the `RANSAC.m` function. Run the provided `TestRansac.m` to test your function. Include your code and the plots from a run of `TestRansac.m` in your submitted solution. *Hint: review the lecture slides on RANSAC and read through `RANSAC.m` from the beginning. Be sure to understand the meaning of each input argument before coding. Use the given functions to help you. Each "Your Code Here" spot should only take one or a few lines of code. Look at our default value for each variable to understand the format expected.*

(c) Above, we applied RANSAC to $N = 10$ points. Edit `TestRansac.m` to set $N = 1000$ and run it a few times to see the results. Briefly explain how the results are different and why. What RANSAC parameters could you change to improve the solution? You can modify the parameters in `TestRansac.m` to test your answer.

6 Difference-of-Gaussian (DoG) Detector

(a) The 1-D Gaussian is given by

$$g_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(-\frac{x^2}{2\sigma^2}\Big)$$

Calculate its 2nd derivative with respect to x, and use Matlab to plot it. (assume $\sigma = 1$).

(b) Use Matlab to plot the difference of Gaussians in 1-D given by

$$D(x, \sigma, k) = \frac{g_{k\sigma}(x) - g_\sigma(x)}{k\sigma - \sigma}$$

using $k = 1.2, 1.4, 1.6, 1.8$ and 2.0. State which value of k gives the best approximation to the 2nd derivative w.r.t. x. Again, you may assume that $\sigma = 1$, and we recommend that you plot the results from parts (a) and (b) on the same graph. *Important: for this problem, please make sure you submit both your Matlab code and your plots.*

(c) The 2D equivalents of the plots above are rotationally symmetric. To what type of image structure will a difference of Gaussian respond maximally?