HW1: Computer Vision (CS 600.361/600.461)

Instructor: Nicolas Padoy (padoy@jhu.edu)
TA: Maria Ayad (maria.ayad@jhu.edu)

Out on Friday, September 9, 2011
Due on Monday, September 20, 11:59pm.

Submission. Your submission should contain two parts. The first part is a document (electronic or paper format) containing your *short* explanations, comments or derivations for each question. The second part is your commented Matlab code. The electronic submission should be posted through Blackboard, using a single .zip file named using the format **jhed_hw1.zip** and containing a directory named using the format **jhed_hw1**. If the first part is submitted electronically, the explanation file should have the format **jhed_hw1.pdf** or **jhed_hw1.doc**, and be included in the directory *jhed_hw1*. A post box will be available in the CS department secretariat for paper submissions. Please write your full name within *each* document. Concerning the matlab code, each main problem should be answered in a different file (exo1.m, exo2.m, exo3.m). These files can make call to external function that you have defined; just don't forget to include the corresponding .m files!

Evaluation. Your solution will be evaluated with respect to 1) correctness of your responses; 2) correctness of the code execution; 3) conciseness and clarity of your solution

Integrity code. Please note that the assignments are strictly individual and that by submitting your solution, you abide by the CS department Academic Integrity Code.

---

1. **Separable filters** *[Solution file exo1.m; Please answer all questions by placing comments directly in the code.]*

   (a) *(1 pts)* Load the image *lena.jpg* and convert it to a greyscale image, called J in the following.

   (b) Let $g_h(\sigma)$ be a Gaussian filter of size $[3\sigma, 1]$, $g_v(\sigma)$ be a Gaussian filter of size $[1, 3\sigma]$ and $g_r(\sigma)$ be a Gaussian filter of size $[3\sigma, 3\sigma]$.

      i. *(2 pts)* Convolve the image J it with $g_r(2)$, using the matlab function *conv2*. [Method M1]

      ii. *(2 pts)* Convolve the image J successively with $g_h(2)$ and $g_v(2)$. [Method M2]

      iii. *(2 pts)* Convolve the image J with $g_h(2) \times g_v(2)$. [Method M3]

      iv. *(2 pts)* Compare the results visually. What do you conclude ?

(c) *(3 pts)* Compare the resulting images using the mean of the square differences ($s_x \times s_y$ is the size of the image):

$$e(I_1, I_2) = \frac{1}{s_x \times s_y} \sum_x \sum_y (I_1(x,y) - I_2(x,y))^2.$$

What do you conclude ?
What is the maximum intensity difference between two pixels ? What is a likely explanation for these differences ?

(d) *(4 pts)* Compare the speed of methods [M1] and [M2] for $\sigma \in \{1, \ldots, 10\}$. Plot together the computation times for each method with respect to σ. (Hint: check the matlab functions *tic* and *toc*)

(e) *(2 pts)* What do you conclude from the previous two experiments ?

(f) (600.461 only)

i. *(2 pts)* Perform the same experiment using the matlab function *imfilter*.

ii. *(3 pts)* What is the result concerning the computation times ? Can you explain the difference, compared to *conv2*?

2. **Circle detection** *[Solution file exo2.m]*

(a) *(3 pts)* Algebraic circle fitting
Let $\mathcal{C}$ be a circle defined by the equation

$$aP^{\mathrm{T}}P + B^{\mathrm{T}}P + c = 0, \qquad (1)$$

where $P = [x, y]^{\mathrm{T}} \in \mathbb{R}^2$ is a circle point and $(a, c) \in \mathbb{R}^2$, $B = [b_1, b_2]^{\mathrm{T}} \in \mathbb{R}^2$ are the circle parameters.
Express the center $C \in \mathbb{R}^2$ and the radius $r \in \mathbb{R}^+$ of $\mathcal{C}$ as functions of the parameters a, b_1, b_2, c.

(b) *(3 pts)* Let S be a set of points $\{P_i | 1 \leq i \leq n\}$ assumed to lie on the circle. Using equation 1, propose a linear least squares method to estimate the circle parameters. What is the condition on i for your method to work ?

(c) *(2 pts)* Write a matlab function *[Solution file points_circle.m]*

function [P] = points_circle(C,r,n)

that returns n points equally spaced on a circle of center C and radius r. P is assumed to be a matrix of size $2 \times n$, C of size 2×1.

(d) *(2 pts)* Write a matlab function *[Solution file noisy_points_circle.m]*

function [P] = noisy_points_circle(C,r,n,sig)

that returns n points equally spaced on a circle of center C and radius r to which Gaussian noise (with parameters $\mu = 0, \sigma = sig$) has been added.

(e) *(3 pts)* Write a function *[Solution file ls_circle.m]*

function [C,r] = ls_circle(P)

that returns the estimated center and radius of the circle going through the points P (vector of size $2 \times n$). [Hint: use the matlab functions *svd* or *pinv*]

(f) *(3 pts)* Using the previous functions, generate an image of size 512×512 containing 10 points from the circle $\{C = [130, 150]^{\mathrm{T}}, r = 50\}$ (circle points have intensity 1, other points have intensity 0).
Estimate the circle center and radius using the function *ls_circle* and compare to the original parameters. Compare also the results visually by plotting the estimated circle in the image.

(g) *(2 pts)* Perform the same experiment using 10 noisy points ($\sigma = 10$).

(h) *(2 pts)* Load the image *coins1.jpg*. Propose and implement an approach to process the image and obtain a resulting binary image containing the (noisy) circle contours of the coins.

(i) *(2 pts)* Write a function *[Solution file inliers_circle.m]*

[INL] = inliers_circle(C,r,P,d)

which returns the indexes of the vector of points P that correspond to points lying within a distance d to the circle.

(j) *(5 pts)* Propose and implement a method derived from RANSAC to detect all the circles successively: 1) use RANSAC to detect one circle and the corresponding inliers; 2) re-estimate the circle parameters using all the inliers; 3) remove the inliers of this circle from the list of image points; 4) go back to step 1) to detect the remaining circles
Describe how you choose your thresholds.

(k) *(2 pts)* Display on the image all the detected circles.

(l) *(3 pts)* Use the previous method to count automatically the number of coins. Try the complete approach on the image *coins2.jpg*.

(m) *(3 pts)* Propose and describe briefly the implementation of a different method to count the number of coins in the image.

3. **Corner matching** (600.461 only) *[Solution file exo3.m]*

(a) *(2 pts)* Load the two images *building1.bmp* and *building2.bmp*. The provided file *harrisCorner.m* provides a function that computes Harris corners in a greyscale image and returns a vector containing their locations.
Use this function to compute and display the corners in the two images.

(b) *(5 pts)* Implement a function *[Solution file match_corners_NCC.m]*

function [m1,m2] = match_corners_NCC(I1,I2,corners1,corners2,ws)

that use the NCC (normalized cross-correlation) to compare the corners in the two images and returns two vectors of matching corners (namely $m1(:,k)$ is a 2D point in $I1$ that should match the 2D point $m2(:,k)$ in the second image $I2$. For each pair of corners, perform the SSD comparison using a window of size $(2ws + 1) \times (2ws + 1)$. $corners1$ and $corners2$ are two vectors containing the original (unmatched) corners, as detected by the Harris detector.

(c) *(3 pts)* Display the images side-by-side within one figure and draw lines between the matched corners to evaluate visually the results using a chosen window size.

(d) *(4 pts)* Modify the previous function to weight the NCC score by a factor depending on the spatial distance between the two compared corners. This should exclude corners far from each other in order to reduce the number of incorrect matches, since the displacement between the two images is small.
Enforce also symmetry during the matching: select two corners c1 and c2 if and only if c1 is the best match for c2 and c2 is the best match for c1.