Computer Vision (600.461/600.661)
Homework 3: Line and Point Detection

Instructor: René Vidal

Due 10/02/2014, 11.59PM Eastern

1. **Line detection.**

   (a) Write a function `[line,inliers]=ransac(points,iter,thr,mininlier)` that implements the RANSAC algorithm for fitting a line with parameters `line` $\in \mathbb{R}^3$ to a collection of `points` $\in \mathbb{R}^{2\times P}$. Other input variables are the number of iterations (`iter`), the threshold (`thr`) on the distance from a point to the line used to determine if the point is an inlier, and the minimum number of inliers (`mininlier`) needed to declare a good fit. Other output variables are the set of `inliers` $\in \{0,1\}^P$, where $1 = inlier$ and $0 = outlier$. Write a script `hw3q1a.m` that does the following. Draw 100 points in $[-1,1]\times[-1,1]$ passing through the line $y = x$ and 100 points uniformly at random in $[-1,1]\times[-1,1]$. Add zero mean Gaussian noise to these points with $\sigma = 0.1$. Apply RANSAC to the noisy points. Plot each iteration of RANSAC with the estimated line and the inliers to the estimated line.

   (b) Suppose you are given data drawn from m models and corrupted with outliers. How would you modify RANSAC to fit m models to the data? If w is the fraction of inliers, n is the number of points needed to fit the model, k is the number of samples, and m is the number of models, how many samples k do you need to guarantee that you recover the m models with probability p? How does this number grow with n and m?

   (c) Write a function `[line,inliers]=ransac(points,iter,thr,mininlier,nlines)` that extends your function in 1a) to multiple lines so that the ith entry of `inliers` $\in \{0,1,\ldots,$`nlines`$\}^P$ is 0 if the ith point is an outlier, or a number in $1,2,\ldots,$`nlines` indicating the line to which the ith point belongs. Write a script `hw3q1c.m` that does the following. Draw 100 points in $[-1,1]\times[-1,1]$ passing through the line $y = x$, 100 points passing through the line $y = -x$ and 100 points uniformly at random in $[-1,1]\times[-1,1]$. Add zero mean Gaussian noise to these points with $\sigma = 0.1$. Apply RANSAC to the noisy points. Plot each iteration of RANSAC with the estimated line and the inliers to the estimated line.

   (d) Write a script `hw3q1d.m` that does the following. Apply the Canny edge detector to these images. Apply RANSAC to the detected edges. Plot the lines estimated by RANSAC superimposed on the image.

2. **Feature point detection and matching.**

   (a) Write a function `keypoints=corners(I,w,kappa)` that implements the Harris corner detector. Your function should a) Apply a `w`×`w` Gaussian filter to the image `I` to remove noise. b) Compute the gradients I_x, I_y of the image in x and y directions. c) Compute the Harris operator at each pixel (x,y)

$$K(x,y) = \frac{\det(H(x,y))}{\text{trace}(H(x,y))} \quad \text{where} \quad H(x,y) = \sum_{(u,v)\in W(x,y)} w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}_{(x+u,y+v)}, \qquad (1)$$

   where $w(u,v)$ is the Gaussian window function and $W(x,y)$ is a `w`×`w` neighborhood of (x,y). d) Find the set of pixels $\mathcal{H} = \{(x,y) : K(x,y) > \kappa\}$ such that the response of the Harris operator is above a threshold κ. e) Apply non-maximum-suppression in a `w`×`w` pixel neighborhood of each $(x,y) \in \mathcal{H}$. Store the pixel coordinates of the resulting `keypoints`. Write a script `hw3q2a.m` that tests your code on these images for different values of the window size w and the Harris corner threshold κ. For the best choice, plot the results of each step, as well as the Harris corners before and after non-maximum-suppression.

   (b) Write a function `descriptors=features(I,keypoints)` that extracts a descriptor centered around each keypoint in `keypoints`$\in \mathbb{R}^{2\times P}$. Use a simple vectorized image patch of 9×9 pixels as your descriptor so that `descriptors`$\in \mathbb{R}^{81\times P}$. **Hint:** Study the function `reshape`.

(c) Write a function `[matches,ssdvalues]=matching(descriptors1,descriptors2,tau)` that matches two sets of feature descriptors using the sum of squared differences (SSD) as a matching score. Your function should return for each descriptor in image 1 the index of the best match in image 2 provided that the SSD score is above a threshold τ. If a descriptor in image 1 has no match, then the corresponding entry of `matches` should be set to 0. Write a script `hw3q2c.m` that tests your code on these images and plots the resulting matches. Plot also the ROC curve and use it to select the threshold.

Submission instructions. Send email to vision14jhu@gmail.com with subject 600.461/600.661:HW3 and attachment `firstname-lastname-hw3-vision14.zip` or `firstname-lastname-hw3-vision14.tar.gz`. The attachment should have the following content:

1. A file called `hw3.pdf` containing your answers to each one of the analytical questions. If at all possible, you should generate this file using the latex template hw1-vision14.tex. If not possible, you may use another editor, or scan your handwritten solutions. But note that you must submit a single PDF file with all your answers.

2. For coding questions, submit a file called `README`, which contains instructions on how to run your code. Use separate directories for each coding problem. Each directory should contain all the functions and scripts you are asked to write in separate files. For example, for HW2 the structure of what you should submit could look like

   (a) `README`

   (b) `hw2.pdf`

   (c) `hw2q3`: `hw2q3c.m`, `hw2q3e.m`

   (d) `hw2q4`: `hw2q4b.m`, `hw2q4c.m`

   The TA will run your scripts to generate the results. Thus, your script should include all needed plotting commands so that figures pop up automatically. Please make sure that the figure numbers match those you describe in `hw2.pdf`. You do not need to submit input or output images. The output images should be automatically generated by your scripts so that the TA can see the results by just running the scripts. In writing your code, you should assume that the TA will place the input images in the directory that is relevant to the question solved by your script. Also, make sure to comment your code properly.