Homework 3 - Write up
M1522.001800 Computer Vision (2019 Spring)

2014-10469 InSeoung Han

Date: April 19 Friday

1 Programming Part

1.1 Part1 : Line detection using RANSAC

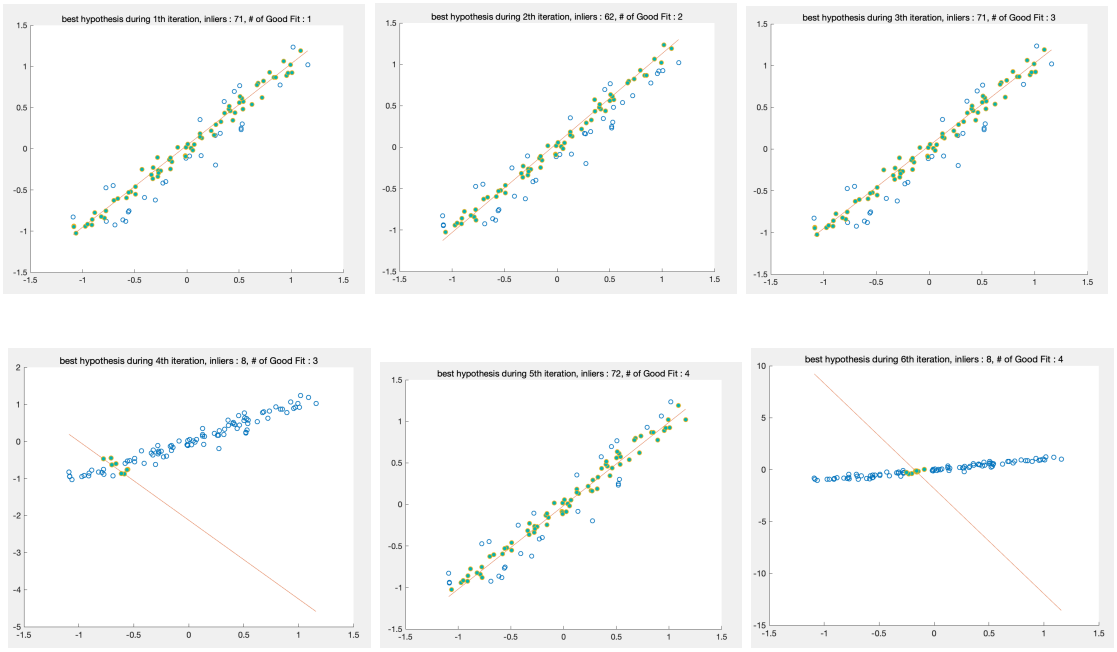

Figure 1: ransac iteration 1 to 6

To implement ransac, firstly, I randomly selected two points in input points. Then calculated inliers according to the line formed by the two points. At this time, we can compute distance between any point and the line by projecting the point to the normal vector of the line. After acquiring inliers, I compared the number of inliers with mininliers. And add one to GoodFit if it is bigger than mininliers. Also, if it is bigger than the number of inliers in previous best fit, I updated best inliers which will be the result of 'inliers'. At last, I made hypothesis of inliers using 'polyfit' matlab function, then ploted original points inliers hypothesis. I did all this again for 'iter' times.

1.2 Part2 : Estimating transformations from the image points

I used 1-2-(d) method in theory question.

$$Ah = \begin{bmatrix} {x_2}^1 & {y_2}^1 & 1 & 0 & 0 & 0 & -{x_1}^1{x_2}^1 & -{x_1}^1{y_2}^1 & {x_1}^1 \\ 0 & 0 & 0 & {x_2}^1 & {y_2}^1 & 1 & -{y_1}^1{x_2}^1 & -{y_1}^1{y_2}^1 & {y_1}^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ {x_2}^N & {y_2}^N & 1 & 0 & 0 & 0 & -{x_1}^N{x_2}^N & -{x_1}^N{y_2}^N & {x_1}^N \\ 0 & 0 & 0 & {x_2}^N & {y_2}^N & 1 & -{y_1}^N{x_2}^N & -{y_1}^N{y_2}^N & {y_1}^N \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ ... \\ 0 \end{bmatrix}$$

After making A, I figured out the eiganvector of A^TA with smallest eiganvalue in V by using 'svd' function in matlab. and reshape it to make H matrix. 'computeHnorm' was easy to implement because all I have to do was normalize p1, p2 and pass these to 'computeH' function to get H. normalization can be done by using 'normalize' function in matlab.

1.3 Part3 : Mosaicing

first I found correspondent points between two images. and save it in 'points.mat'. I compute warped image by using inverse warping method. if there is some interpolation, I just rounded off the float coordinates to make it discrete. Then I merge it to 'Iref' image by firstly making big screen to include both images and drawing the warped image first and drawing the reference image overlapping the warped image.

1.4 Part3 : Rectification

it was similiar to warping. I firstly selected four corner points in 'HTC.png' and decided target corner locations. Then using inverse warping, I fill out all pixels in new image.