CSE 573: Computer Vision and Image Processing

HW 2: Homography and Fundamental Matrix Estimation

Instructor: Kevin Keane
TAs: Radhakrishna Dasari, Yuhao Du, Niyazi Sorkunlu

Due Date: October 18, 2017

Submitted by,
Muthuvel Palanisamy,
muthuvel@buffalo.edu
Person # - 50246815

Part I – Homography Estimation

(Using uttower images to mainly describe the parameters chosen)
(homgraphyEstimationMain())

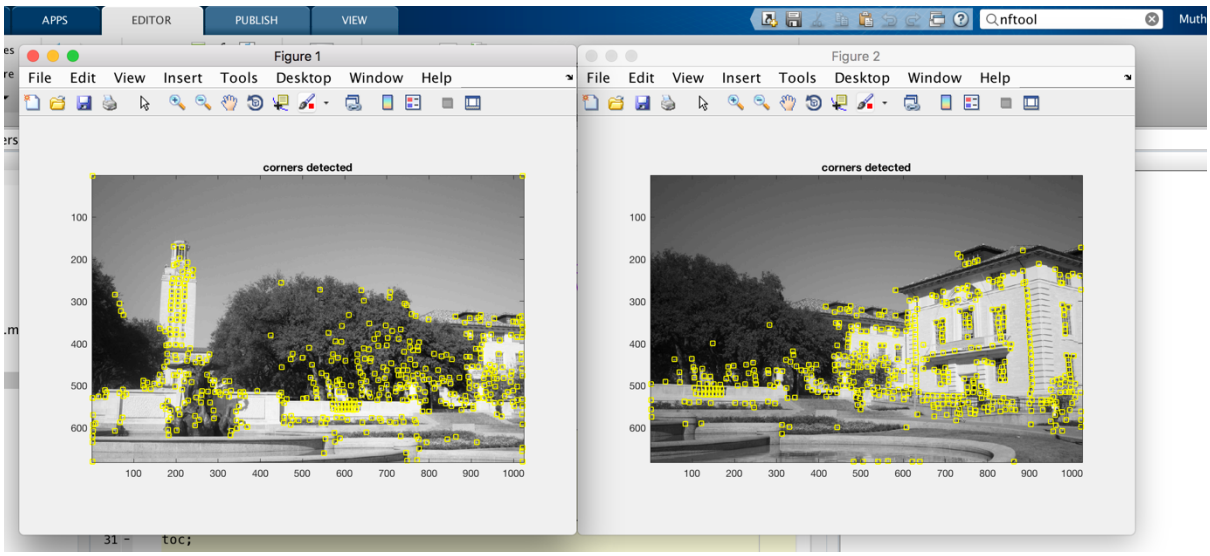
Algorithm Implemented

- Loaded two input images and converted them to grey scale and double.
- Used the provided **Harris Detector** to detect features (corners) in the image.
- Extracted neighbourhood around each keypoints to form **descriptors**
- The **putative matches** are obtained by comparing every corners on the left image with every corners on the right image using the provided **dist2()** function and selected the top **200 matches**
- Used **RANSAC** algorithm for (1000-20000 iteration) selecting the matching points on both images. **SVD** is used to estimate the homography in every iteration.
- Warped one of the images with the best homography obtained.
- Proceed with stitching of the other image with the warped image

1. Detecting Features – Harris Detector

- rgb2grey() and then im2double() on both images
- Used harris detector and extracted features
- On repeated testing it was found that sigma=3 and radius of non-max supp > 3 was and a threshold of 0.05 to 0.20 was the best input parameters that could detected many features
- Low radius size resulted in lot of unwanted corners. Also threshold higher than 0.5 was greatly inefficient in detecting many features

Detected corners on uttower images



2. Neighbourhood Extraction

- **Padarray()** is used to pad the image depending on the size of neighbourhood extracted to extract features of corners
- Extracted neighbourhood of every corners detected
- Descriptors were good when the size was frm 3*3 to 5*5
- Descriptor of size 5 was good in eliminating unwanted features after using dist2()

3. Pairwise Distance Estimation using dist2()

3.1 Initial approach tried:

- Passed each pair of descriptors one by one to dist2() in a loop
- Checked if the distance is less than a threshold value (if yes, they are chosen as putative matches
- Repeated the same procedure for all possible pairs

Drawbacks:

- It became hard to control the number of matches selected and a lot of experimenting was required
- Many features were detected on the uttowers which did not have any matching points
- Did not provide any way to control matches being selected from same area
- Since the intensity of pixels vary most features extracted were not from the matching points (ony on the left image)

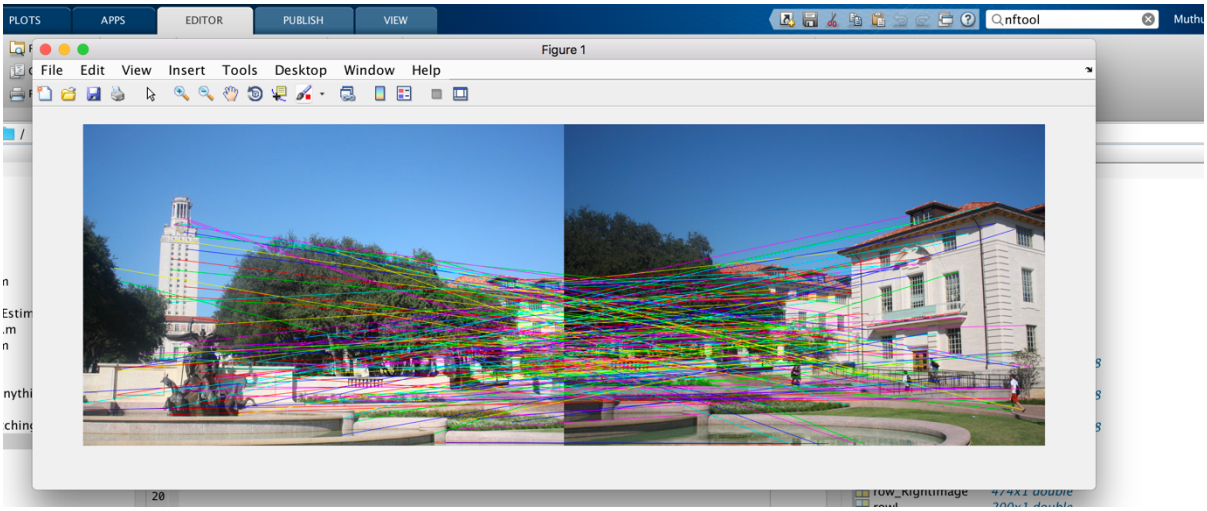
3.2 Final Approach: (credit to suggesstiong from piazza forum)

- Extracted distance between two points for every possible pair and stored them in an array in a loop - pairwiseDiff
- Created a loop iterating for the number of putative patches you want to be detected (eg: 200 patches = 200 times)
- Picked the minimum distance in the pairwiseDiff and extracted the matching points on both images accordingly
- Changed the row and column of minimum value selected to be a big value (say 100000) so that the same point is not detected in the next iteration
- Process in repeated for specified number of iterations

Adavantages:

- Controlling or changing the number of putative patches extracted is defined by the number of iteration
- **Setting the row and column** of detected points to maximum value helped in elimination of corners from same location. Upon experimenting it was found that this **had less incorrect patches** from towers on the left image
- Output had less inliers after RANSAC when compared to the earlier method

Visualization of output from 3.2 (second method) – 200 matches



Ref 1.

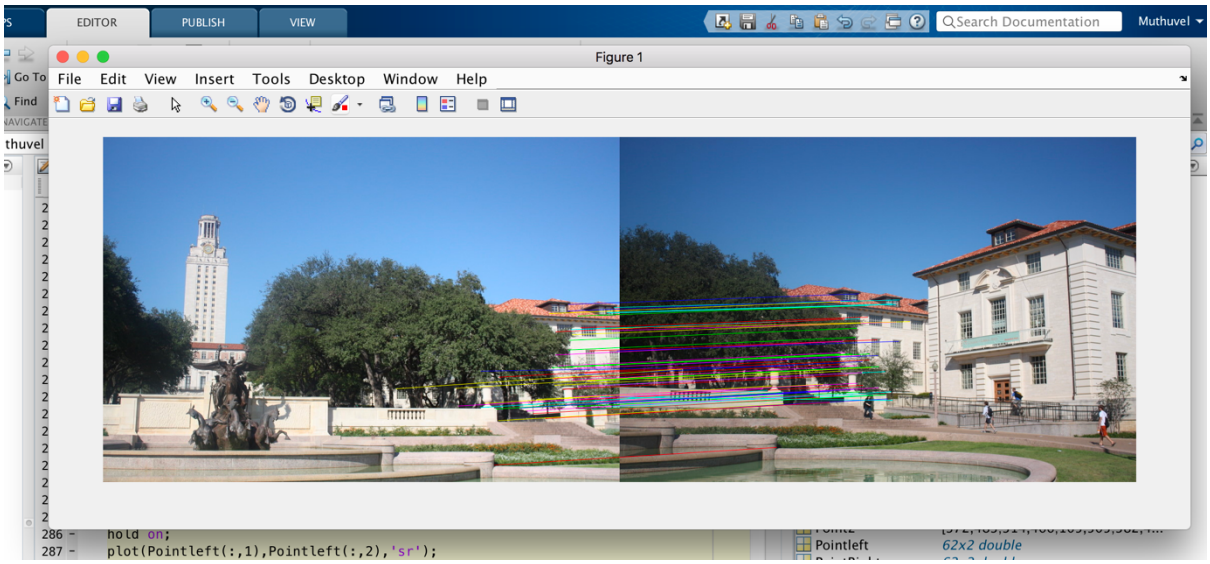
4. RANSAC (Used 10000 – 20000 iterations)

- Four points are chosen in random for every iteration
- **A** matrix is formed using the below formula (credit: https://math.stackexchange.com/questions/494238/how-to-compute-homography-matrix-h-from-corresponding-points-2d-2d-planar-homog)

$$PH = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' & x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y_1' & y_1y_1' & y_1' \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' & x_2' \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y_2' & y_2y_2' & y_2' \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3x_3' & y_3x_3' & x_3' \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3y_3' & y_3y_3' & y_3' \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4x_4' & y_4x_4' & x_4' \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4y_4' & y_4y_4' & y_4' \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ h8 \\ h9 \end{bmatrix} = 0$$

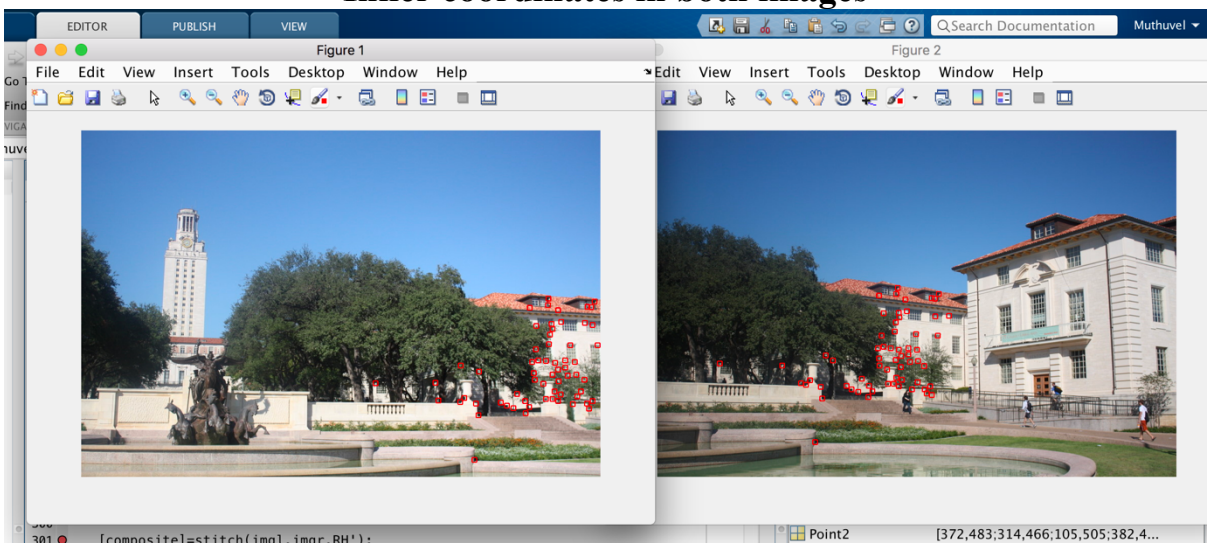
- **SVD** is then used to extract the Homography Matrix (H) of size 3*3
- **H** is multiplied with matching points from left image
- The number of inliers and outliers are calculated and stores by calculating the different of values in the above step from their corresponding matching points in right image. Anything below a threshold is chosen as inlier.
- The procedure is repeated for a defined number of iterations
- All the H matrix and their corresponding number of inliers are stores in arrays
- The best H matrix is chosen by finding the inlier matrix with maximum value and its corresponding H matrix
- Output visualized below shows the correct matched extracted from RANSAC

Inlier Visualized after RANSAC



1. (Inliers are denoted in colour lines)

Inlier coordinates in both images

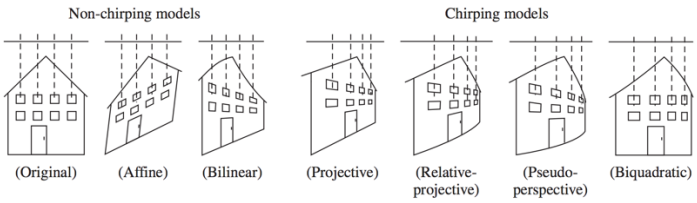


Numer of Inliers = 77
Residue = 763.33

5. WARPING IMAGE:

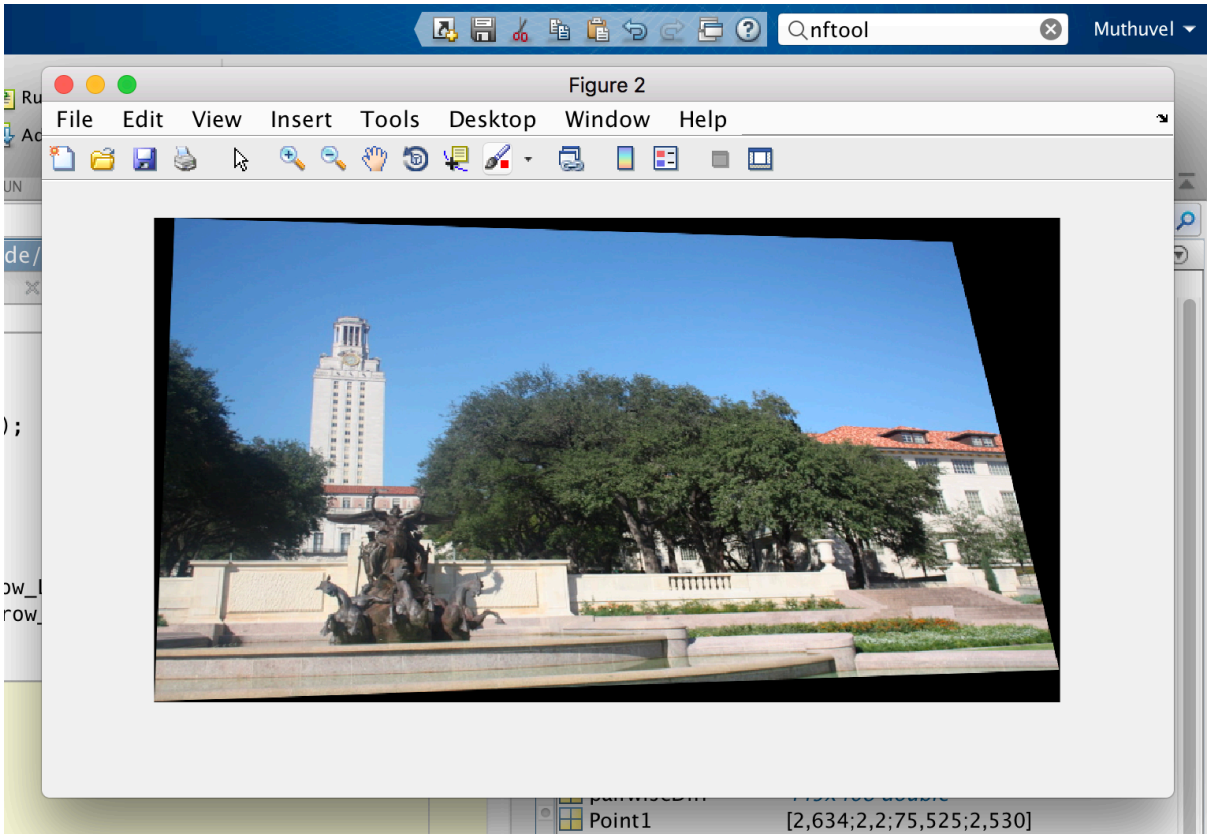
- Left image is warped into right image.
- Maketform calculated the tfrom from the best H matrix and ‘projective’ transformation is used since it best applies for panorama stitching as follows
- Thought the output was efficient there were instances where homography matrix was not perfect to make for ideal warping

Different Types of Mapping



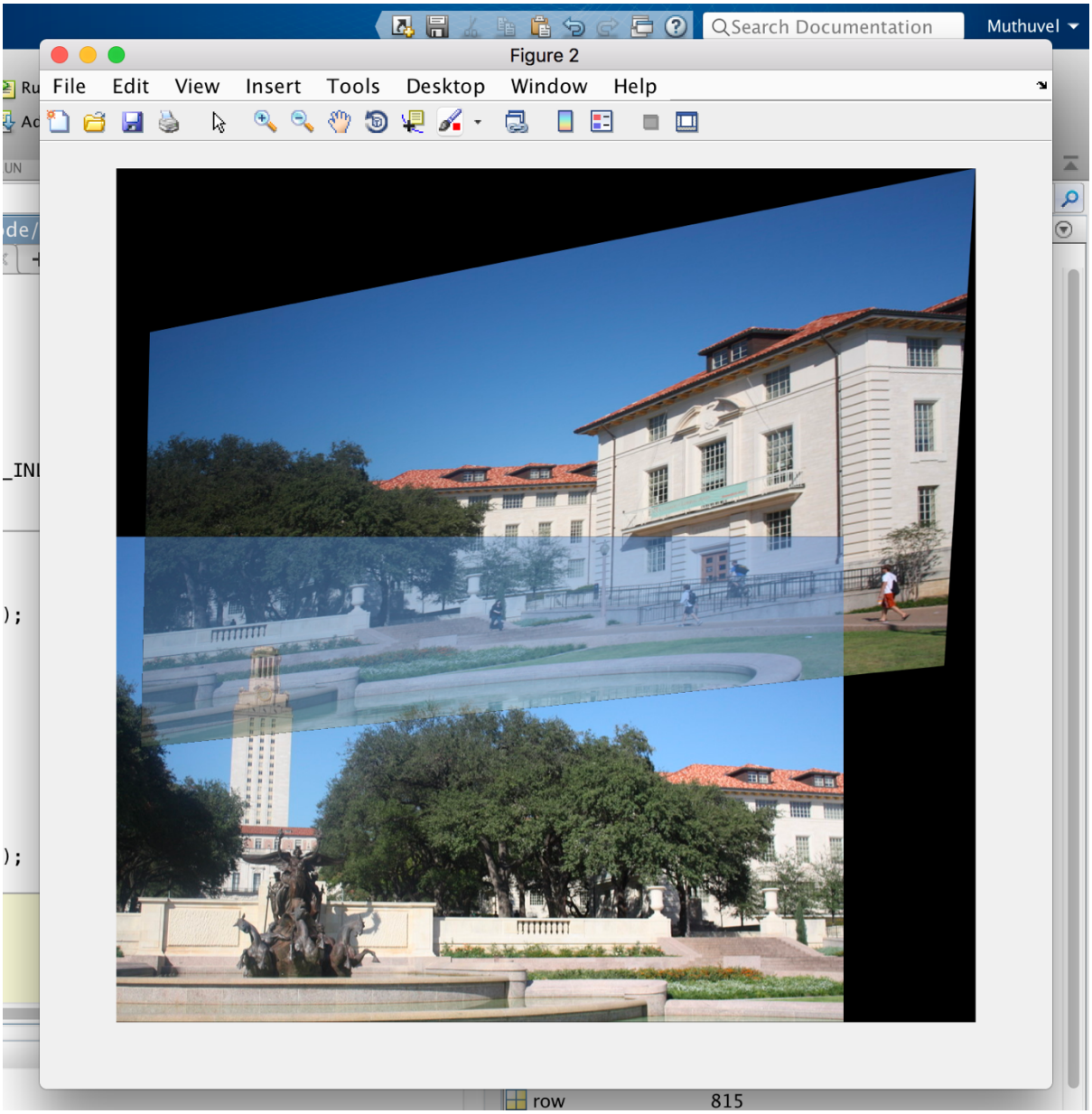
(credit : http://eeweb.poly.edu/~yao/EL5123/lecture12_ImageWarping.pdf)

Warped left image from uttower data



6. Stitching of two images

- Stitching did not produce desirable output
- The Best output that was derived is shown below



Part 2 : Fundamental Matrix Estimation

2.1 Part -1 : Fundamental Matrix Estimation using Matching Points

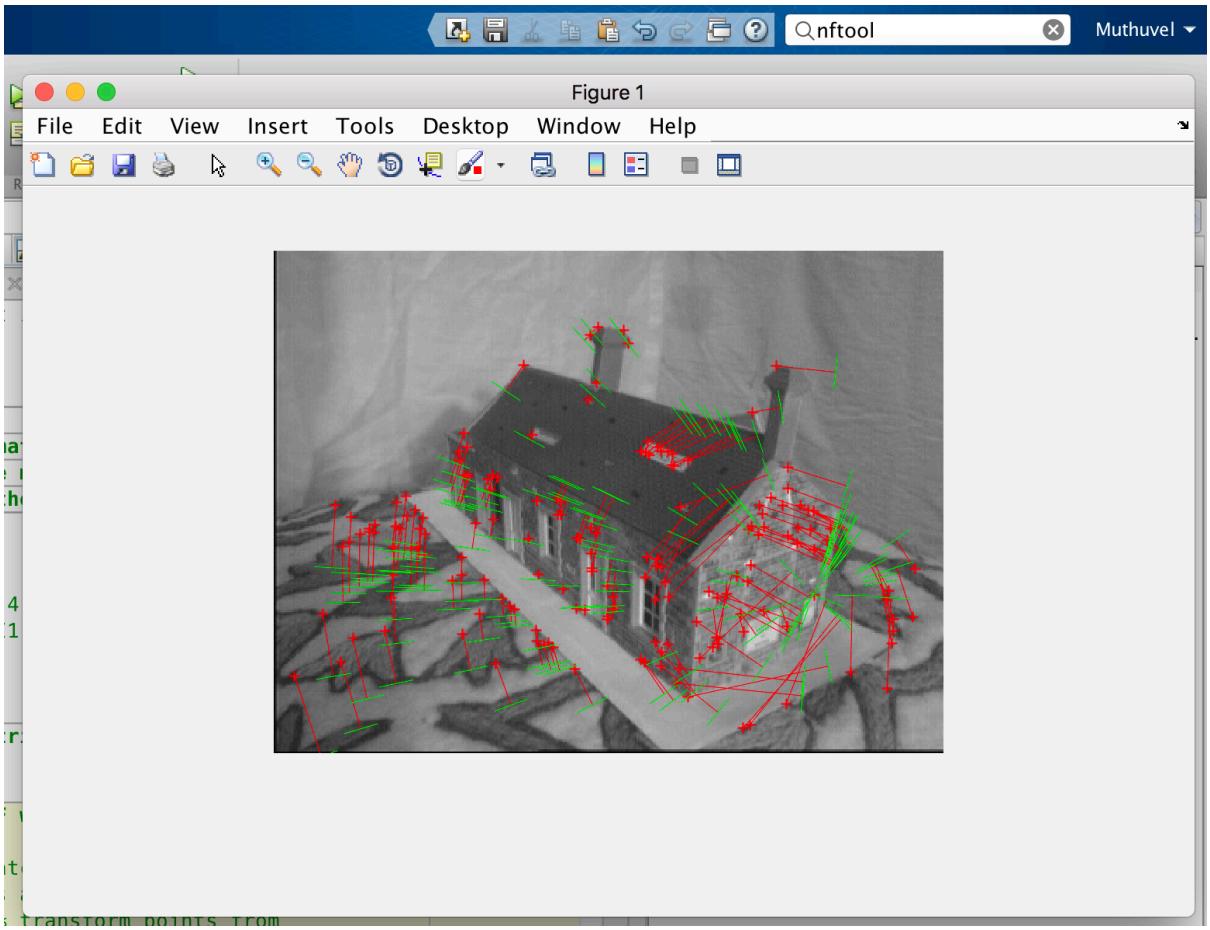
- **Fundamental_fit()** function is used to estimate the fundamental matrix for normalized and un-normalized matching points.
- If flag for normalization is set normalization is done on matching points.
- The below formula is used to find the A matrix for SVD

$$\begin{bmatrix} x'x & x'y & x' & y'x & y'y & y' & x & y & 1 \end{bmatrix} \mathbf{f} = 0,$$

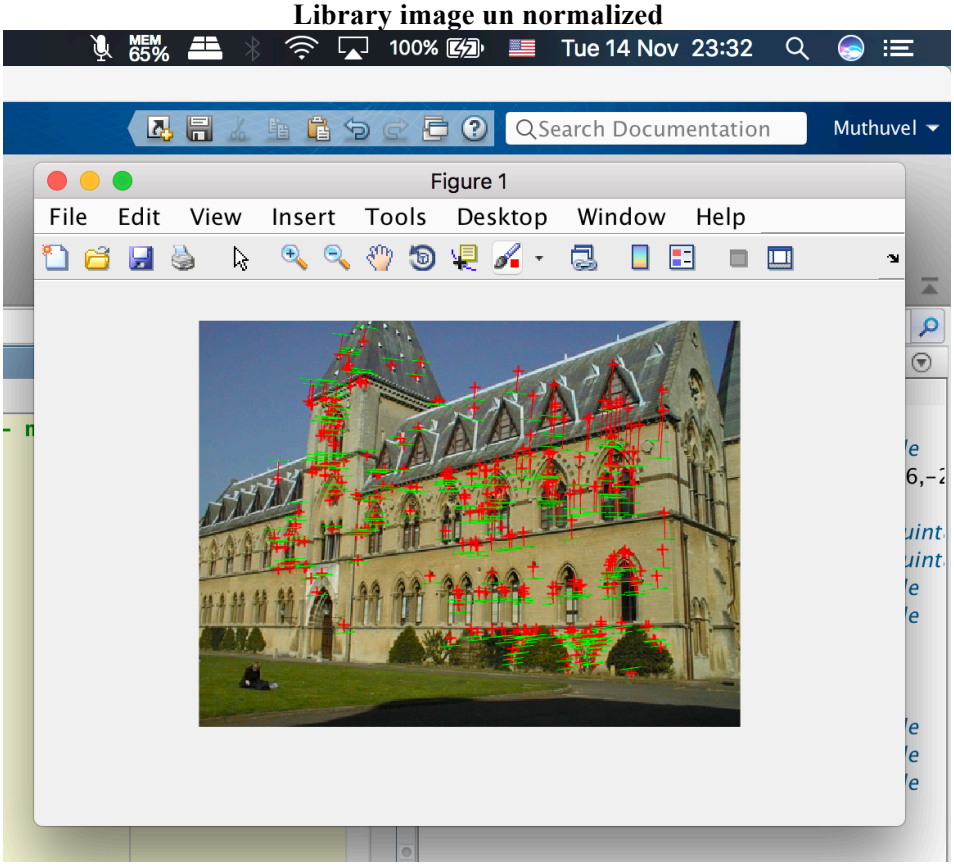
(referred recitation slides and https://www8.cs.umu.se/kurser/TDBD19/VT05/reconstruct-4.pdf)

- The above formula is repeated for every set of matches and SVD is applied on final A matrix
- The V co-ordinate is the Fundamental Matrix and to enforce rank-2, SVD is taken again on the fundamental matrix and the last singular value is set to zero. The values are multiplied again (U*S*V'=F ie inverse of SVD)to form F or rank-2.
- The given sample - code is used to visualize epipolar lines from Fundamental matrix.

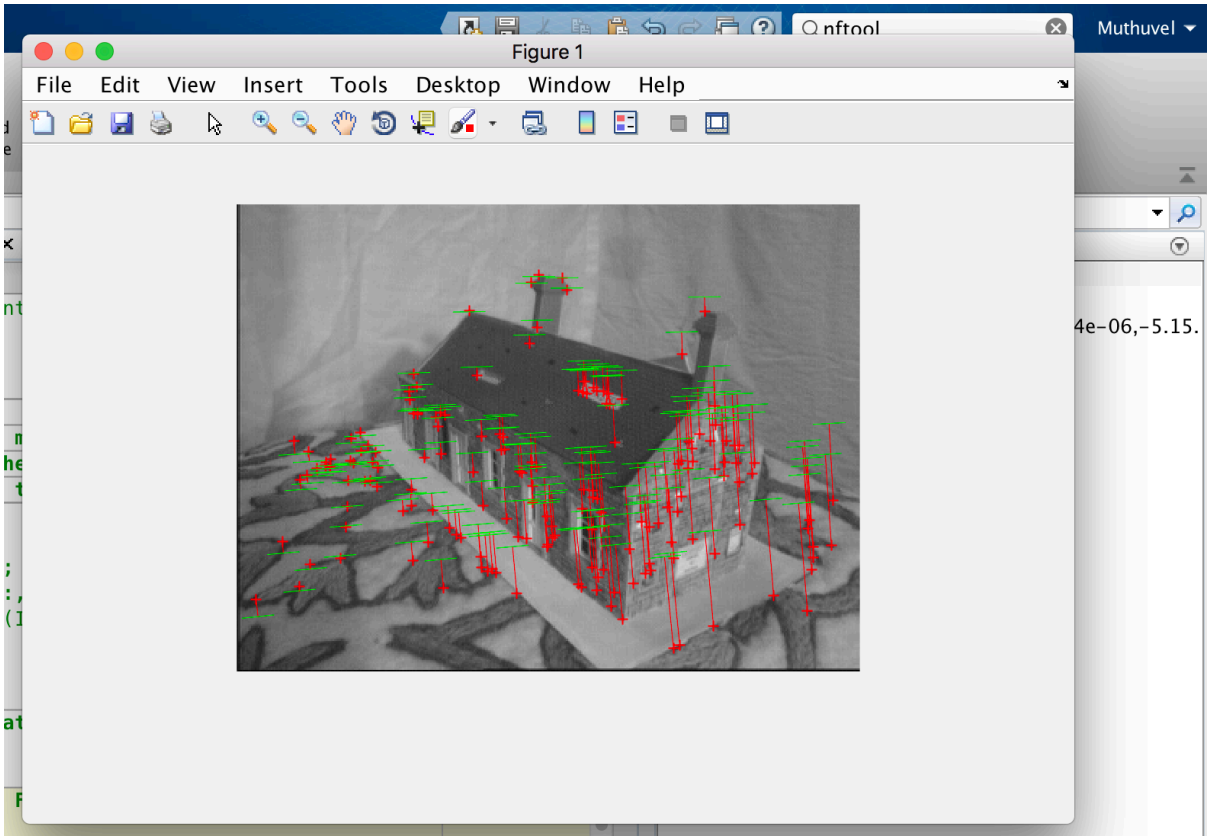
Ground truth- epipolar line (unnormalized)



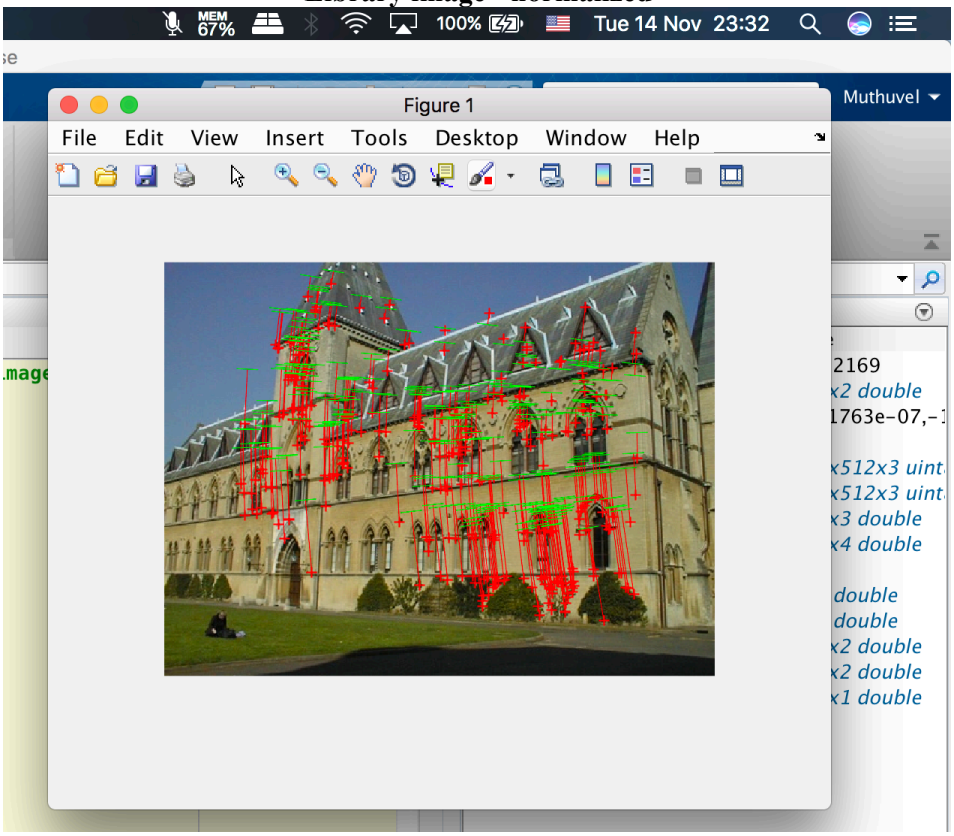
Library image un normalized



Ground Truth Epipolar lines – normalized



Library image - normalized



House image:
Average_Residue_Ground_Truth_Normalized = 20.5423
Average_Residue_ Ground_Truth_Unnormalized = 1.5719

Library Image
Average_Residue_Ground_Truth_Normalized = 39.2169
Average_Residue_ Ground_Truth_Unnormalized = 7.4697

2.2 Part 2 – RANSAC for estimation of Fundamental Matrix from images

- fundamentalMatrixRANSAC() – performs the task
- This RANSAC method used feature extraction similar to one documented in Homography Estimation
- Descriptor extraction from neighbourhood is done in the same way as Homography Estimation
- Putative patches are extracted in a similar way as Homography Estimation
- In RANSAC part– the extracted putative matches to used fetch eight random matches
- The matches are input to fundamental_fit() employed in the part above
- It returns the fundamental matrix which is then used to calculate the inliers using the code below given in sample_code

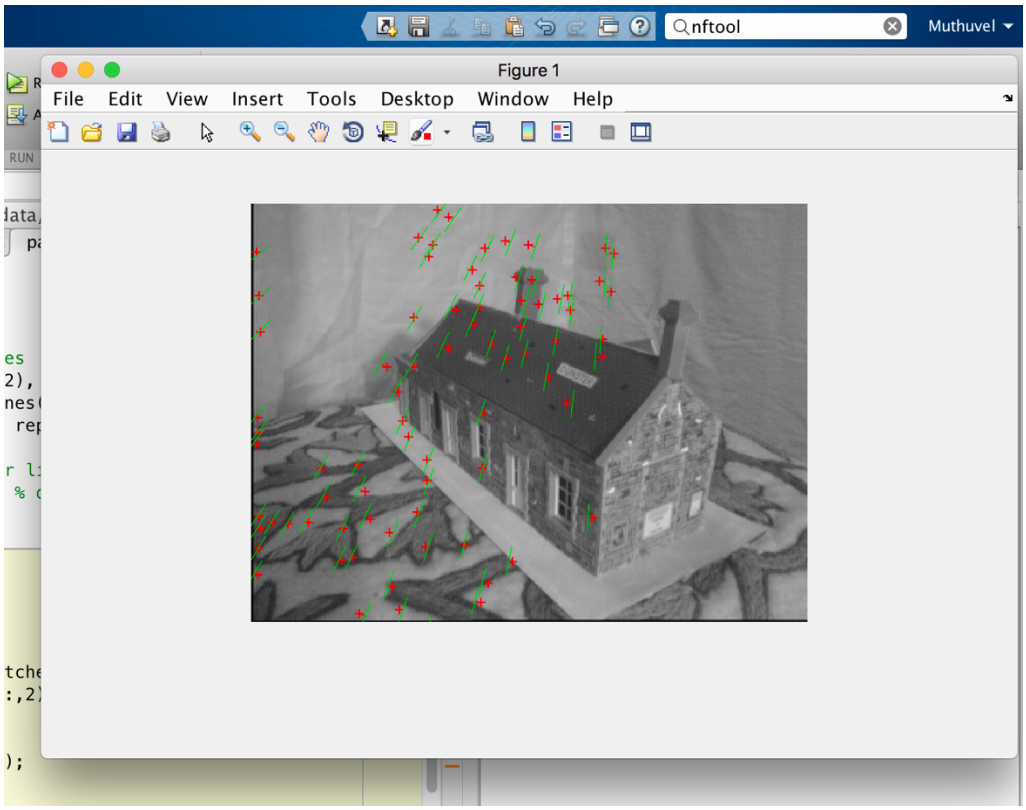
```
% Epipolar lines on image 2
L = (F * [matches(:,1:2) ones(N,1)]')';

% Finding epipolar lines closest to matches
L = L ./ repmat(sqrt(L(:,1).^2 + L(:,2).^2), 1, 3); % rescale the line
pt_line_dist = sum(L .* [matches(:,3:4) ones(N,1)],2);
```

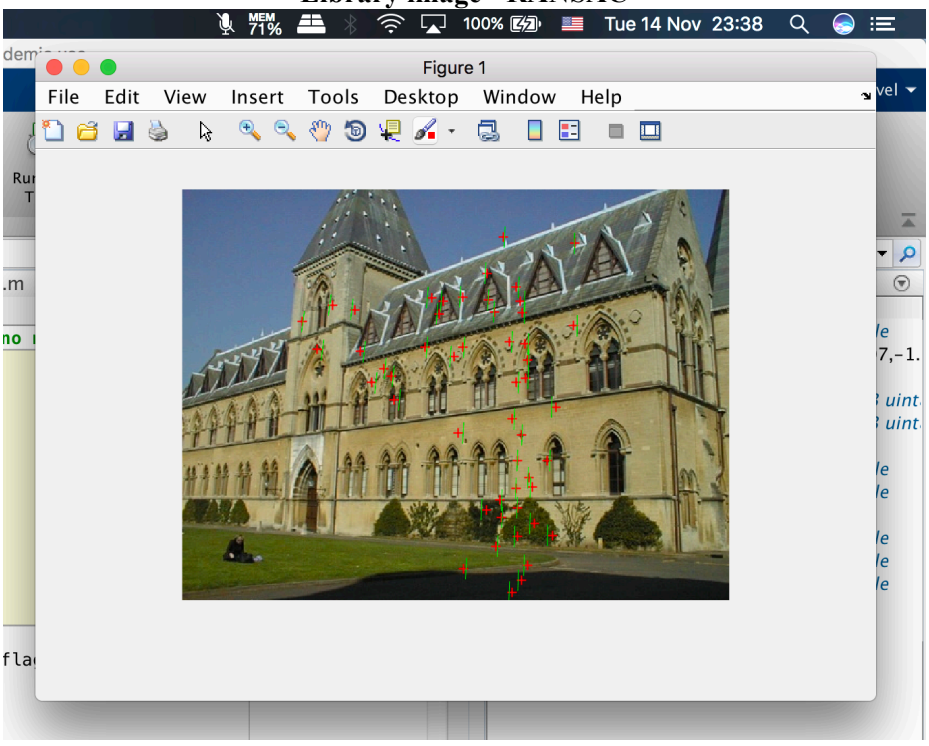
- The inliers are detected by using a threshold value

- The same procedure is repeated for n iteration (1000 to 10000) to get the best F matrix
- The epipolar lines are visualized using the code in sample_code

Epipolar lines from F derived from RANSAC (normalized) – displaying only inliers



Library image - RANSAC



House image:
Number of Inliers = 77
Average_Residue_RANSAC_Normalized = 0.2487

Library image
Number of Inliers = 70
Average_Residue_RANSAC_Normalized = 1.0037

Comparison

Ground Truth Based Estimation	RANSAC Estimation
Avg. Residue is high in normalized method	Avg. Residue is comparatively low in normalized method
All Inliers are found to point in same direction	Inliers tend to direct towards a point
Output is fixed, it does not matter how many times we run the program since all the input parameters are the same	Since this method is based on RANSAC, the output varies slightly most of the time as the input 8 points in RANSAC for estimation of fundamental matrix is randomly chosen. Very rarely does this method output that is grossly different from the previous one

2.3 Part – 3: Triangulation and Camera Center

- Uses triangulation1() function

Camera Center

- Camera center for both images is calculated by applying SVD on the inverse of the camera matrices of each images

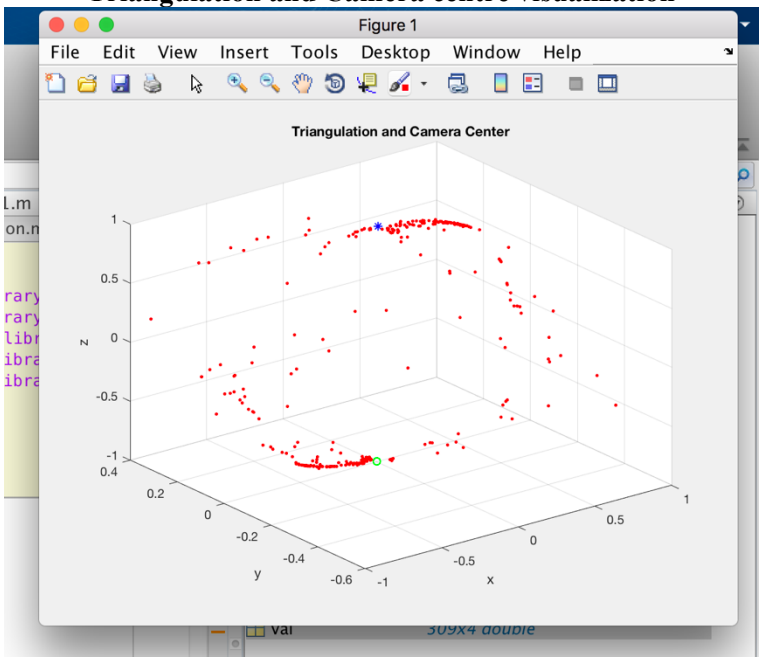
Triangulation

- Triangulation again uses SVD
- The A matrix is derived using the formula below

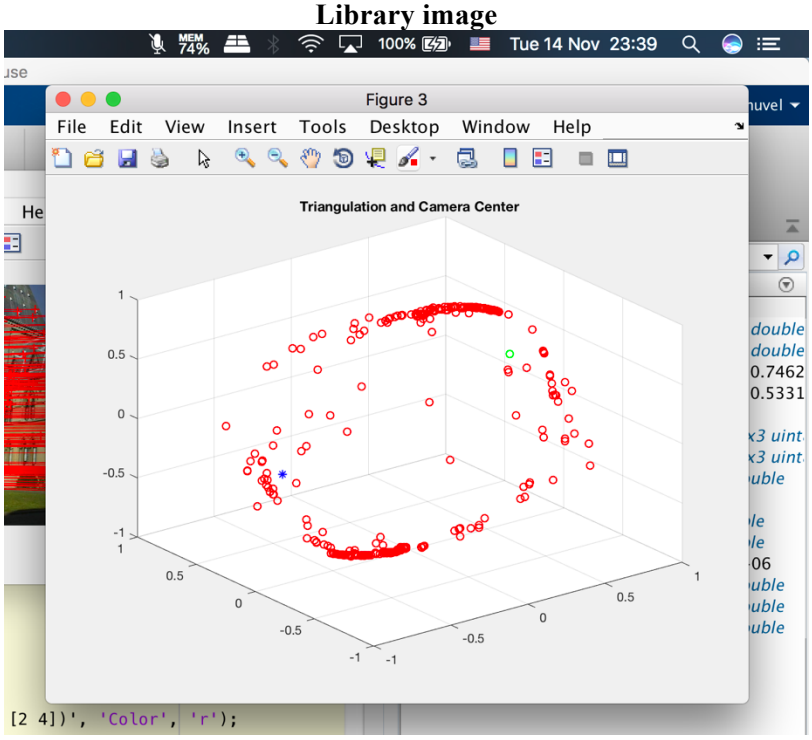
$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p}_3'^\top - \boldsymbol{p}_2'^\top \\ \boldsymbol{p}_1'^\top - x'\boldsymbol{p}_3'^\top \end{bmatrix}.$$

- **SVD** is imposed and the corresponding last vector in V contains the output
- The same process is repeated in a loop for every matching point sets and all the outputs are concatenated across loops
- The result is then multiplied with the camera matrix to get the triangulation
- The output is then plotted in 3D using the scatter3D() function

Triangulation and Camera centre visualization



Library image



House image:
Average Residual of 1st image 8.4806e+04
Average Residual of 2nd image 8.3131e+04
Library image
Average Residual of 1st image 1.5904e+06
Average Residual of 2nd image 1.4586e+06

Reference:

1. https://www.mathworks.com/matlabcentral/fileexchange/31144-match-plot?focused=5186593&tab=function ---- Used to visualize connecting lines between inliers in both images – present as match_plot in directory