EE4-62 Machine Learning for Computer Vision
Image Matching Coursework

Biyuan WANG
CID 00821345

Hiu Yan Cecilia KAN
CID 00828315

Imperial College London

1. Introduction

In this coursework, we explore image matching for two different cases of multiple view geometry. We implemented methods for acquisition, matching of interest points from images, evaluation of the match, using the provided Boat and Tsukuba sequences for tuning, then discuss results of their application on testing photos from camera.

2. Interest Point Detection

To perform image matching, we first need to select distinctive corresponding features for the images. We constructed a Harris corner detector (Harris and Stephen, 1988), which chooses points with significant intensity gradient in local neighbourhood for all directions [1], then represent these with SIFT descriptors, both rotationally invariant methods. In the corner region detection, a corner response function measures the classification quality as:

$$R = detM + \alpha(traceM)^2 \quad (1)$$

where M is the auto-correlation matrix, R is corner response, $\alpha = 0.04$ is previously empirically-determined constant. Points with high positive R values indicates corner region, and the mean of the top R values is used for thresholding to add adaptiveness. The corner pixels are the local maxima of R, which we obtained through non-maximum suppression. The SIFT descriptors of these interest points are matched in feature space with Lowe's ratio test (Lowe, 2004) [2]. As some interest points may not have a perfect match, for each SIFT descriptor from one image, the ratio L of its Euclidean distances with the two closest SIFT descriptors from the other image to be matched is used to discard false matches. The R threshold needs to be low enough such that enough number of relevant interest points are chosen to allow sufficient qualifying matches, whereas the L threshold needs to deliver lowest trade off between tolerance to transformation and false match rejection.

The algorithm is very robust against pure translation, with clear overlapping interest points detected in image pairs that are correctly matched (App. N Fig. 19). This allows a comparatively low R threshold. L is very close to 0 (readings on *MATLAB* is 0) for all matches for R thresholded at mean of between 1000 and 3000 highest values for this image set. In this work we use 2000 and 0.6 for R and L thresholds respectively in Section 4. The corner detector is theoretically rotational invariant but not scale invariant, since features detected must be in comparable scale to the window size used. Detection in rotated images is also less robust (App. B). Therefore less corresponding interest points are detected in the boat image series (App. N Fig. 9), leading to poorer performance as the rotation and field of scaling increases as in App. N Fig. 18, which cannot be compensated with threshold settings when the distortion is large. The thresholds for homography estimation is decided using matches between boat images 1 and 2 (App. A Fig. 9 *top* and *middle*), and App. B, which by observation has limited number of mismatch hence serves as meaning reference. We settled with using mean of top 3000 R values for thresholding R, above which larger number of mismatches are introduced, and 0.87 for thresholding L to allow generalisation as using 0.9 affects match quality but no match is drawn for some images using 0.85 (App. F).

3. Homography

We performed homography estimation from interest points correspondence, using Direct Linear Transform [3]. To solve for the homography matrix H, we require the eigenvector associated with the smallest eigenvalue of AA^T, where coordinates of each pair of matched interest points $\mathbf{p}$ and $\mathbf{p}'$, $\mathbf{p}' = H\mathbf{p}$, contributes two rows to

A in the form of:

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & \frac{x'x}{z'} & \frac{x'y}{z'} & \frac{x'}{z'} \\ 0 & 0 & 0 & -x & -y & -1 & \frac{y'z}{z'} & \frac{y'y}{z'} & \frac{y'}{z'} \end{bmatrix}$$

where $(x, y, z)^T$ and $(x', y', z')^T$ are the coordinates of $\mathbf{p}$ and $\mathbf{p}'$ respectively. This eigenvector, comprised of entries of H, can be found in the singular value decomposition (SVD) of A. 4 matches should be sufficient for generating H but more are needed for robustness. We evaluate the HA error as the average absolute distance in pixels between $H\mathbf{p}$ and $\mathbf{p}'$. Matches are sorted to prioritise those with low L values.

We investigate our homography estimation pipeline on the HG image series (App. C) acquired with settings listed in Appendix E. We verify that Harris corner detector is not scale-invariant by comparing interest points detected from HG1 in its original size to that from resize by factor of 2. Different interest points are detected as shown in Fig. 1. Only 7 matches are identified, and the HA error for the points is $[191.0, 469.4]$, which is largely due to obvious mismatches present (App. I).

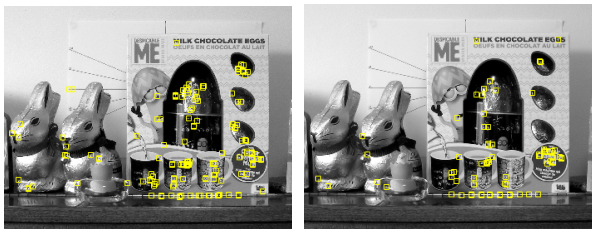

Figure 1. Details in interest points detected by Harris corner detector for HG1 *Left*: in original size; *Right*: scaled down by factor of 2.

Next, we compare the pipeline with interest points input from manual selection from image against those from automatic detection. For projection from HG1 to HG2, the HA error vary significantly when estimated using the minimum number of correspondence $n = 4$ depending on the points used, but is consistently low when $n = 15$ for both input types (App. K). There is no visible mismatch for the detected corners (App. O) hence the performance is similar, with HA error of $[2.655, 3.557]$ and $[1.259, 0.8963]$ for the two sources in interest points respectively, and very similar derived H matrices (App. G), that describes rotation of around 15°and scaling of factor of 1.2 - 1.4. We are largely able to recover HG2 rotated in the opposite direction by projecting from HG1 using H derived from automated pipeline, with rounding defects.

Figure 2. Image projected from HG1 using H derived from transformation to HG2, using Harris detector.

Manual selection produces clearly more accurate results when the correspondence drawn automatically include false matches. For matching between HG1 and HG3, since the distortion is large we are constrained to use L threshold = 0.9, and error from false matches propagates down the pipeline (App. O and Q. The HA error for manual selection is $[3.579, 4.918]$ in contrast to $[567.4, 509.7]$ for automated. The respective derived H matrices are:

$$manualH = \begin{bmatrix} 1.331 & 0.5077 & -1207 \\ -0.5855 & 1.311 & 474.1 \\ -7.968e^{-6} & -3.450e^{-5} & 1 \end{bmatrix}$$

$$autoH = \begin{bmatrix} -0.5456 & -0.3384 & 1378 \\ -0.7229 & -0.3753 & 1778 \\ 4.060e^{-4} & 2.193e^{-4} & 1 \end{bmatrix}$$

Both H matrices describe perspective transformation that is close to affine, with the bottom row close to [0 0 1], more in the case of $manualH$. From row 1, column 2 element of $manualH$, the deduced angle of rotation is 30.5°clockwise, but from row 2, column 1 the angle is 35.8°. Dividing $cos(30.5°)$ through the first two elements in the diagonals yield scale factor of 1.54 and 1.52 respectively. The results are close to the ground truth of approximately 20°rotation with 1.59 zoom factor, taking into account possible noise from camera calibration. The rotation deduced from corresponding elements in $autoH$ is 19.7°anticlockwise or 46.3 °clockwise, which are contradictory, therefore $autoH$ is likely to comprise of shearing as well. The scale factor deduced assuming 46.3°rotation is 1.93 and 1.89. $autoH$ does not describe the transformation correctly. While automatic detection cannot compete with manual selection for large distortion, for small transformation it performs well with calibration.

One parameter that needs adjusting is n. Since our pipeline filters out false match in the beginning, small increase in n from 4 increases robustness for estimation of H from HG1 to HG2 projection, so the HA error reduces in general for increasing n (App. J), as well as actual error in projection (Fig. 3). Increasing n does not improve projection accuracy when large number of correspondences are false, as in the case for HG1 and HG3 (App. L, highlighting the need for outlier detection at this stage. We implemented an outlier detector that takes the histogram of hue of patch around coordinates of both actual and projected points in the actual image as a descriptor, and large distances between descriptors of projected and actual points in feature space is associated with outliers. The colour information has not been included in the pipeline and we hypothesise that wrong projection would differ in appearance to actual point. The detection is not sophisticated or precise enough to achieve high accuracy (App. R).

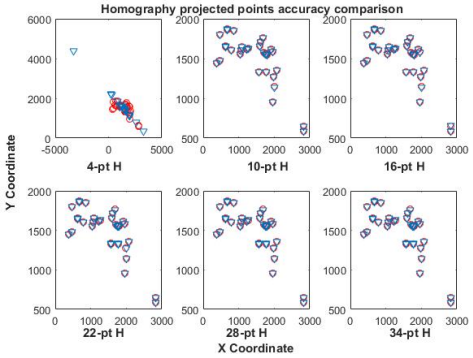

Figure 3. Interest point coordinate comparison between projected points from HG1 using H (*blue triangles*)and HG2 (*red circles*) for increasing n in H estimation.

4. Epipolar Geometry

Using correspondence for stereo images, we look into pipeline for recovery of depth information for FD images. We start with estimating the Fundamental matrix, defined as $\mathbf{u}'^T F\mathbf{u} = 0$ where $\mathbf{u}'$ and $\mathbf{u}$ are coordinates of points from a pair of stereo images. This is achieved using the normalised 8-point algorithm (Hartley, 1997), which follows similar steps to H estimation, with the set of equations to solve being linear so 8 correspondences are required to solve for 8 unknowns in F [4]. For matches between FD1 and FD2,

$$F = \begin{bmatrix} 0.000 & 0.000 & 0.0002 \\ 0.0000 & 0.0000 & 0.0014 \\ -0.0002 & -0.0014 & 0.0507 \end{bmatrix}$$

We project epipolar lines with F' and F on FD1 and FD2 respectively (Fig. 4) which converge to epipoles outside the images, indicating the images are not on the same plane, which may be due to camera calibration error. The epipolar lines through matching points have no observable offset, from that we assess F as reasonably accurate.

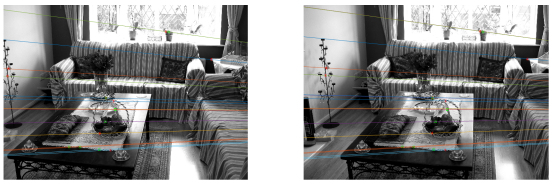

Figure 4. Epipolar lines projected with F' and F derived from interest points marked with red asterisks on FD1 and FD2. Epipolar lines for interest points (green squares) are also shown.

Figure 5. Disparity map for FD1 and FD2, in pixel difference

The disparity map generated for FD1 and FD2 is shown in Fig. 5. There is substantial error in the foreground, with the disparity D of pixels from the table among the lowest even though it is closer to the camera than the sofa. We do not expect the cushions on sofa to have a much lower disparity compared to the sofa either. The region of low disparity on the left may be due to handling of missing correspondence, as well as on the left of the table. However we do see the objects on the table, the sofa at the back, and parts of the window behind correctly differentiated, together with gradual decrease in disparity on the floor on the left of the map. We used a median filter to remove some noise in the map, constrain its size to $D \in [150700]$, then apply the formula

$$depth = \frac{fb}{D}$$

, where depth is the distance from the view point in -z direction, f is the focal length in mm, b is the baseline of the system in mm, to retrieve the depth information of the scene in FD and reconstruct the scene in 3D, shown in Fig 6. We can vaguely make out the table, the black cylinder on the rightmost corner (see App M for depth map with texture map from original image), but the scene may have been too much texture and too little difference in depth for the pipeline.

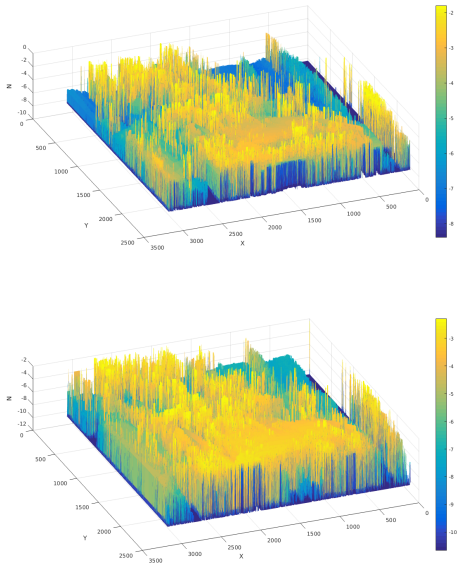

Figure 6. Depth maps established from disparity maps between FD1 and FD2 taking with camera focal length at *Top:* 6.3mm; *Bottom:* 8.0mm

We compare the depth map of FD taken with focal length 8.0mm against that with 6.3mm. The range of values of depth has expanded as the increase in apparent movement of each points in the scene is greater for objects in foreground. The lowest depth values are lost but from Fig. 5 the top disparity values seems to be from noise. To see the effect of noise, we added Gaussian noise to the disparity map for FD taken with focal length 6.3mm and generated the depth map, shown in Fig 7. For our case the Gaussian noise does not alter the map significantly as it is small in magnitude and the disparity map is already noisy. We visualise the effect of noise with the Tsukuba images (Fig. 8) and as expected the noise roughens the surface of the reconstructed 3D scene.

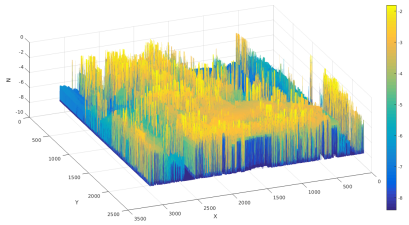

Figure 7. Depth maps established from noisy disparity maps between FD1 and FD2 taking with camera focal length at 6.3mm

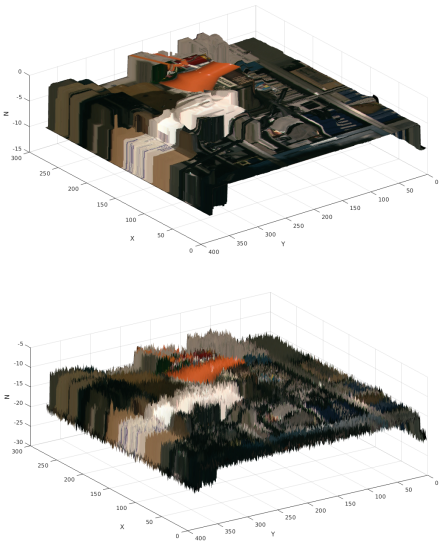

Figure 8. Depth maps established from disparity maps between Tsukuba images *Top:* without noise; *Bottom:* with added Gaussian noise to disparity map

Reference

[1] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[2] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[3] Elan Dubrofsky. *Homography estimation*. PhD thesis, UNIVERSITY OF BRITISH COLUMBIA (Vancouver, 2009.

[4] Richard I. Hartley. In defense of the eight-point algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(6):580–593, June 1997.

Appendices

A. Interest point detected for boat image series

Figure 9. Interest points detected by corner detector for image series with rotation and scaled field of view. Top image is matched with the other two. R threshold is mean of top 1000 R values

B. Interest points detected for rotated image

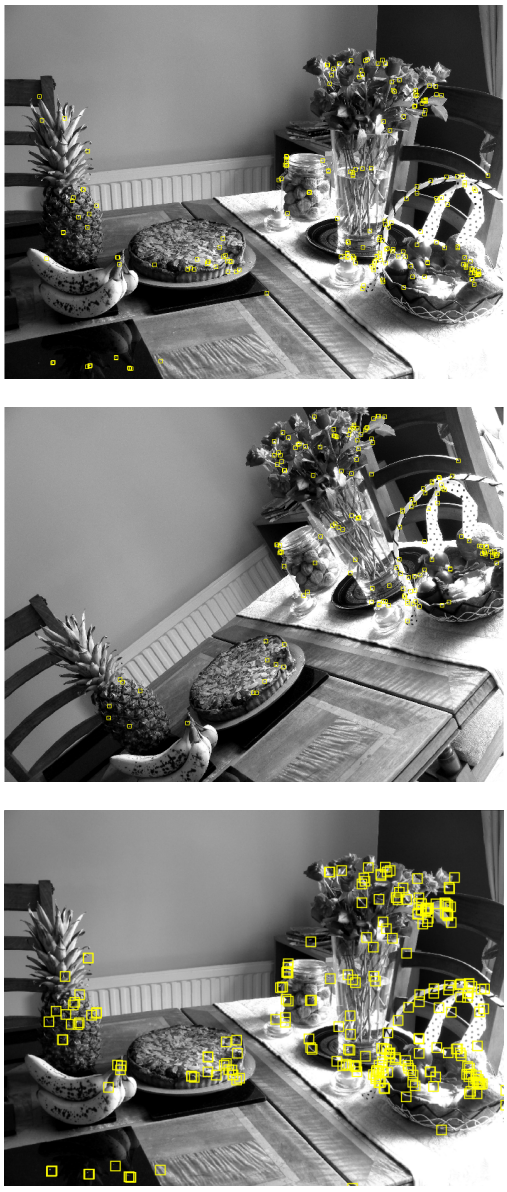

Figure 10. Interest points detected for rotated images *Top*: Original scene; *Middle*: Rotation with camera calibration; *Bottom*: Rotation by MATLAB function for 45 °. R threshold is mean of top 3000 R values. Corner detector may be sensitive to noise, for instance from camera calibration or distortion after warping, for rotated image.

C. Interest points in HG images series

Figure 11. Interest points in HG image series *Top*: HG1; *Middle*: HG2; *Bottom*: HG3

D. Interest points in FD images series

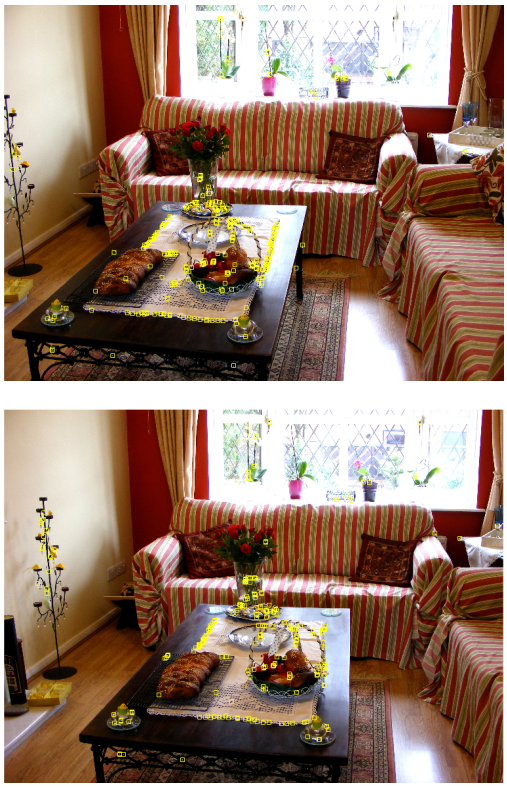

Figure 12. Interest points in FD image series *Top*: FD1; *Bottom*: FD2

E. Information on HG image series acquisition settings

Image	HG1	HG2	HG3
Angle of tilt / °	0	approx -10	approx -20
Focal length / mm	6.3	7.6	10.0
Zoom factor	/	1.21	1.59

Table 1. Information on HG image series acquisition settings

F. Distribution of L with different R and L threshold values

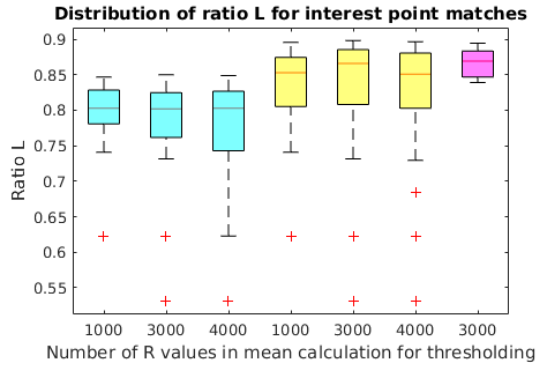

Figure 13. Distribution of L for matches between Fig. 9 *top* and *middle* images, with different R and L threshold values, and for Fig. B *(magenta)*. Highest L value of each box plot is the L threshold value.

G. Derived H matrix from manually and automatically selected interest points

$$manualH = \begin{bmatrix} 1.229 & 0.3017 & -613.0 \\ -0.2559 & 1.246 & 151.6 \\ 6.451e^{-6} & 2.737e^{-5} & 1 \end{bmatrix}$$

$$autoH = \begin{bmatrix} 1.162 & 0.2476 & -522.5 \\ -0.2499 & 1.1506 & 199.9 \\ 9.599e^{-7} & -4.163e^{-6} & 1 \end{bmatrix}$$

H. Homography estimation testing results

Derived H for transforming image points from boat image 1 to boat image 2:

$$\begin{bmatrix} 0.8631 & 0.2153 & 9.345 \\ -0.2110 & 0.8592 & 131.2 \\ 8.340e^{-6} & 2.274e^{-7} & 1 \end{bmatrix}$$

$HA = [0.6473, 0.3991]$

Ground truth for transforming image points from boat image 1 to boat image 2:

$$\begin{bmatrix} 0.85828552 & 0.21564369 & 9.9101418 \\ -0.2115844 & 0.8587636 & 130.47838 \\ 2.0702435e^{-6} & 1.288611e^{-6} & 1 \end{bmatrix}$$

I. Interest point matches for HG1 and resized HG1

Figure 14. Interest points matching for HG1 image in its original size to that from its resize by factor of 2

J. HA of Homography computed from different number of correspondences

No. Correspondences	X error	Y error
4	571	207
10	1.2	0.82
16	1.26	0.93
22	0.81	0.67
28	0.74	0.66
34	0.71	0.66

Table 2. HA error between projected points from HG1 to HG2 and HG2, using H computed from different numbers of correspondences

K. HA error for different number and set of correspondence

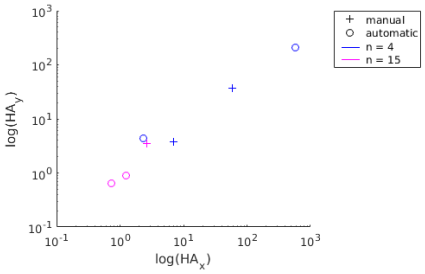

Figure 15. HA error of H derived from HG1 to HG2 transformation, with different number of manually or automatically matched interest points

L. Projection error for H derived from HG1 to HG3 transformation

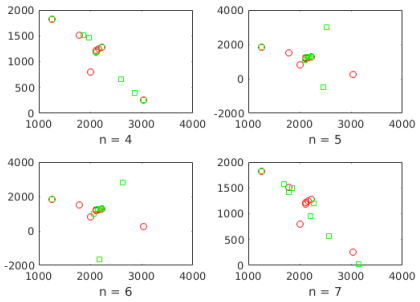

Figure 16. Pixel coordinates of projected points from HG1 to HG3 (*green squares*) and HG3 (*red circles*), using H computed with different n

M. Depth map from disparity between FD1 and FD2 with image colours

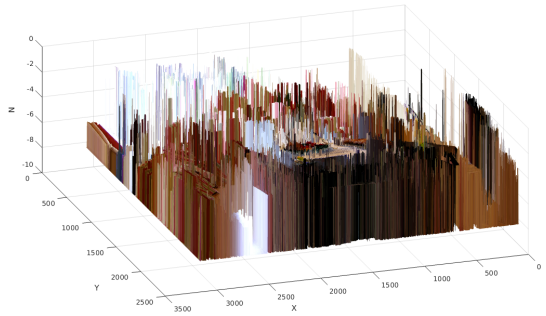

Figure 17. Depth map generated from disparity map between FD1 and FD2, with image colour as colourmap

N. Interest point matching visualised

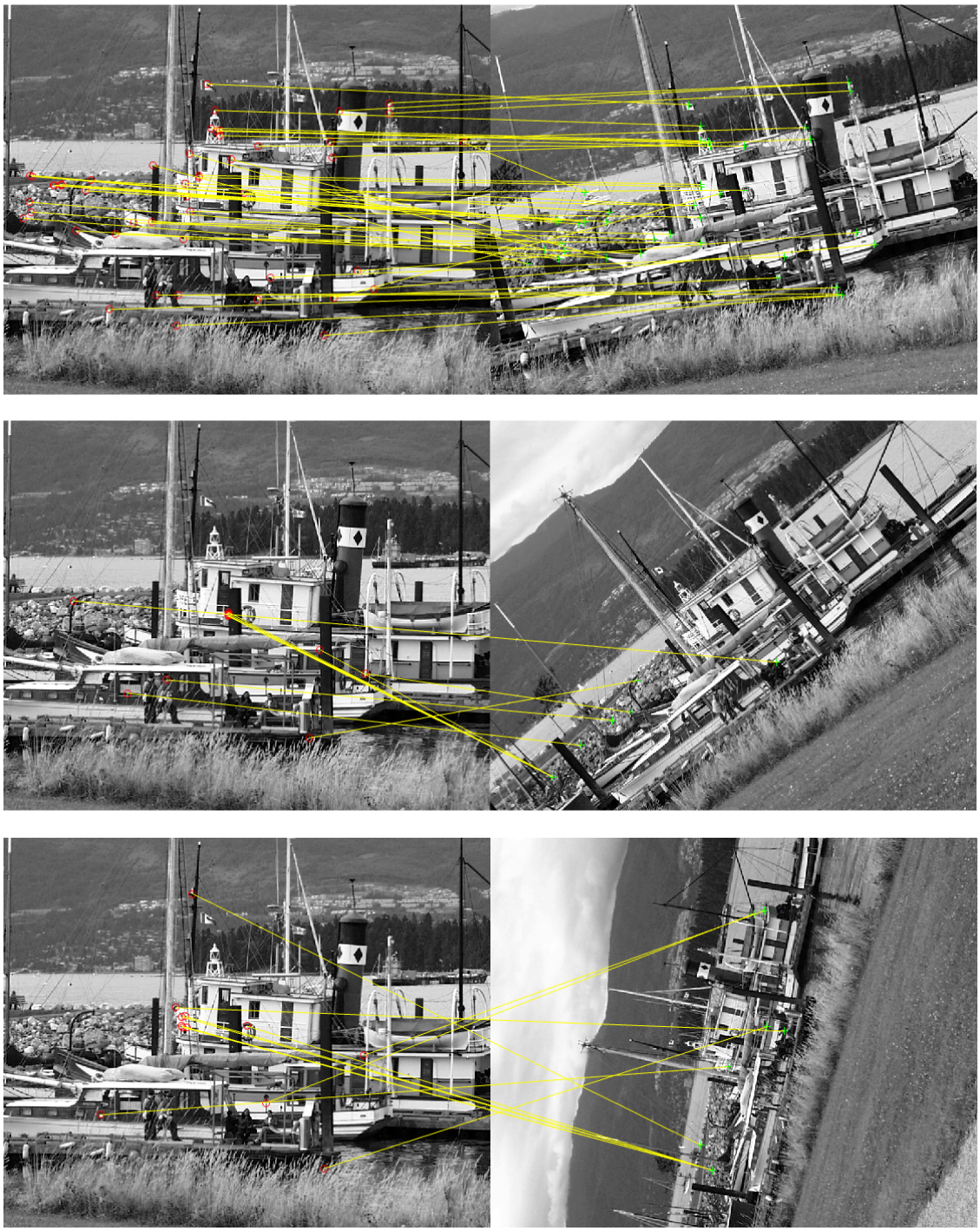

Figure 18. Interest points matching for boat image 1 to *Top:*2, *Middle:* 3, and *Bottom:* 4. R threshold = mean of top 3000 R values, L threshold = 0.90.

Figure 19. Interest points matching for Tsukuba images. R threshold = mean of top 2000 R values, L threshold = 0.60.

O. Interest point matching for HG images

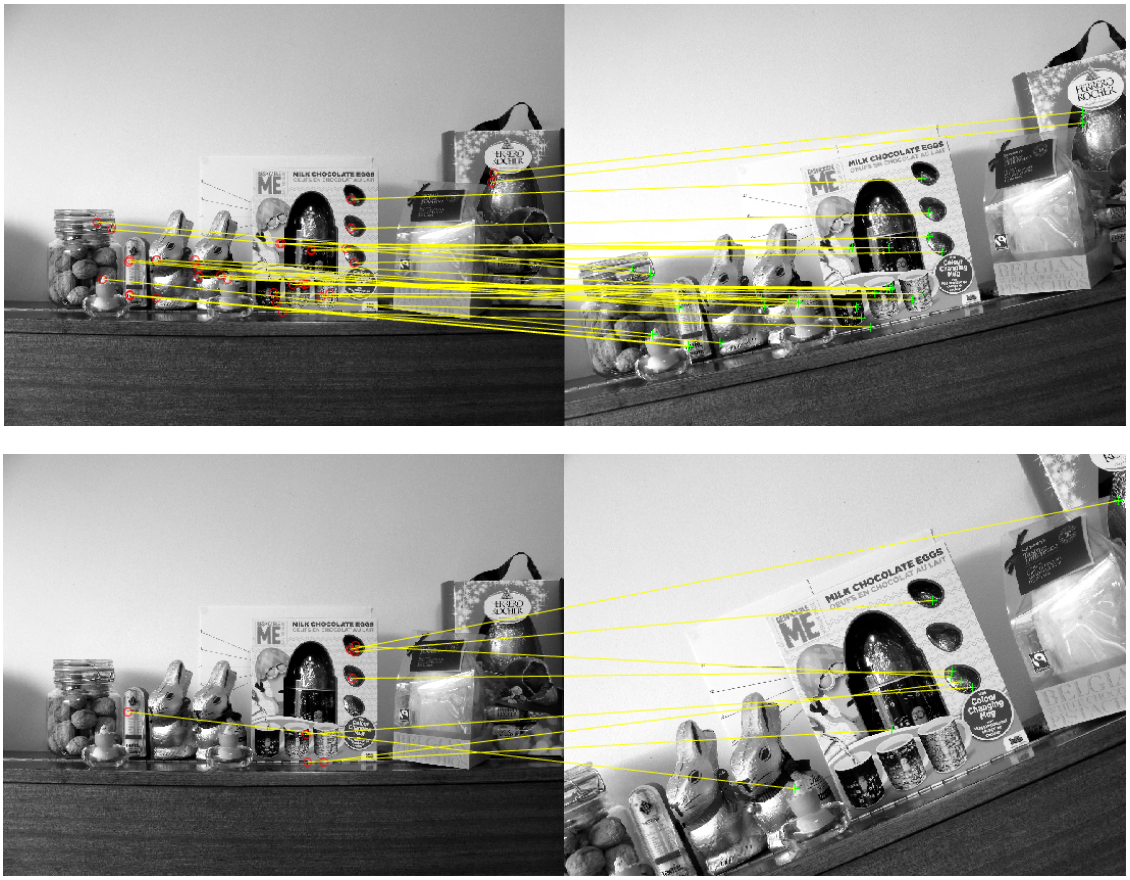

Figure 20. Interest points matching between HG1 and *Top:* HG2, R threshold = mean of top 3000 R values, L threshold = 0.87.; *Bottom:* HG3, R threshold = mean of top 3000 R values, L threshold = 0.90.

P. Interest point matching for FD images

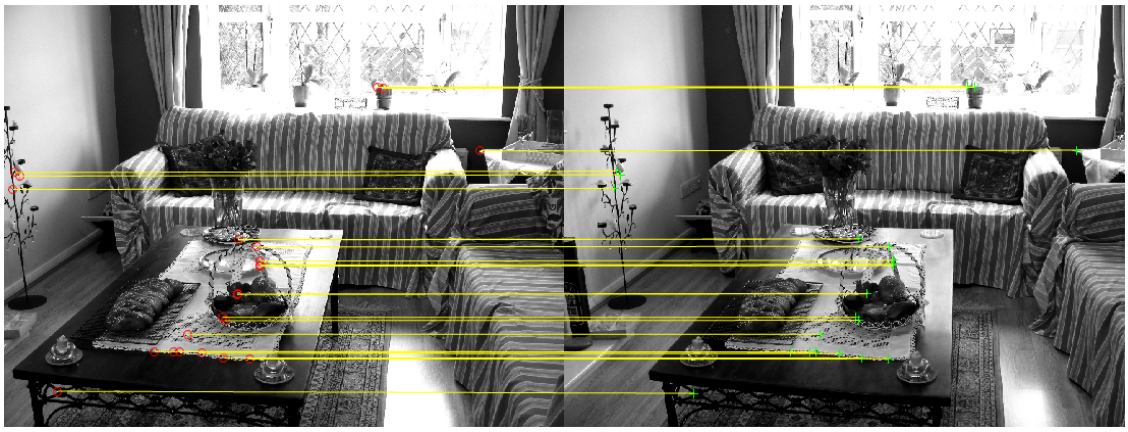

Figure 21. Interest points matching between FD1 and FD2, R threshold = mean of top 2000 R values, L threshold = 0.60.

Q. Matching projected points from HG1 to HG3 with HG3

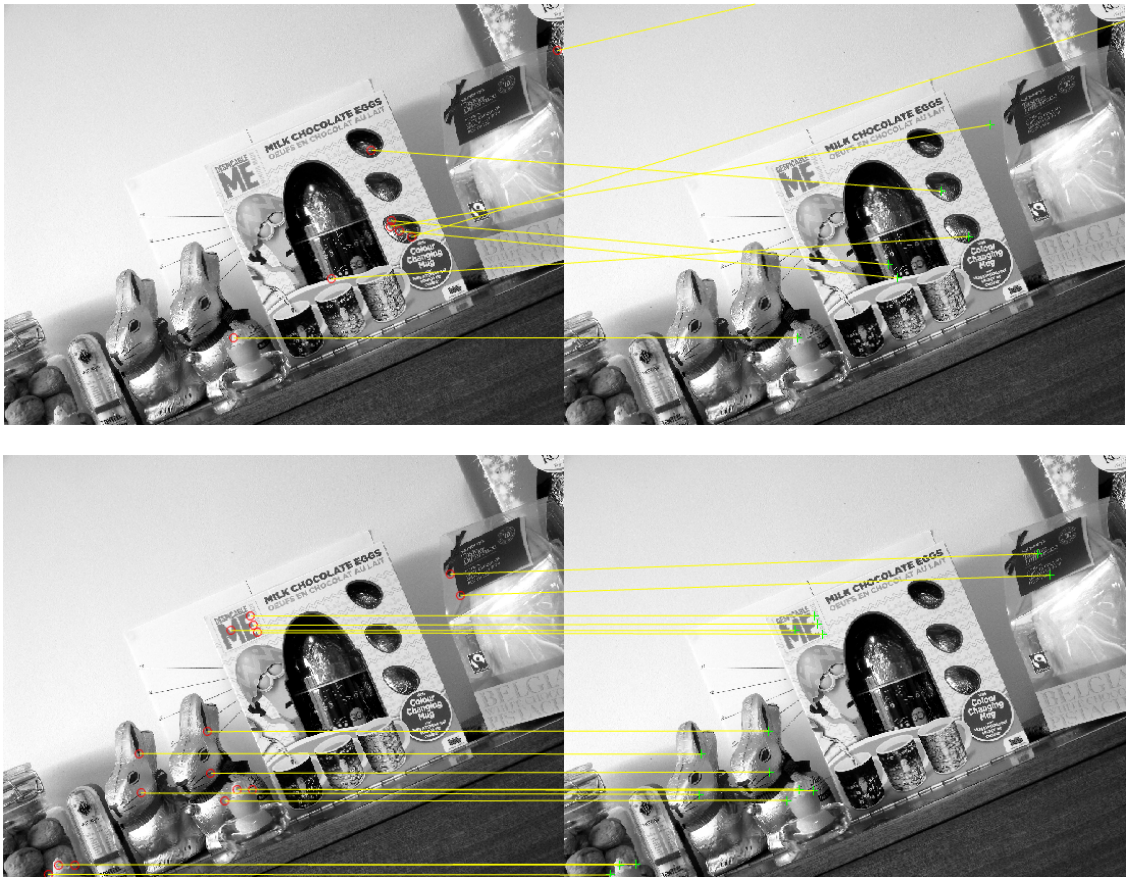

Figure 22. Interest points are projected from HG1 using H derived from transformation to HG3, then matched with HG3 (*left*). *Top:* Interest points selected with Harris corner detector; *Bottom:* Interest points selected manually from image.

R. Outlier detector performance

Figure 23. Outlier detection based on hue histogram of, for HG2 and projection from HG1 using H matrix. *Top:* Inliers *Bottom:* Outliers

S. Code for Harris corner detector

```
function intere_pt = get_interePt(img, patch_size,Rthresh)
% img: image path, must be string.
% Rthresh = 2000 - 3000

blurMask=[0.03 0.105 0.222 0.286 0.222 0.105 0.03];

%% 2) Automatic
% a) Harris points detector
[dx, dy]= meshgrid(-1 :1,-1 : 1); % x & y derivative matrix

% compute derivative of img (correlation between kernel and img)
Ix = imfilter(img, dx);
Iy = imfilter(img, dy);

% autocorrelation matrix
```

```
Ix2 = Ix.^2;
Iy2 = Iy.^2;
Ixy = Ix.*Iy;

%applying gaussian filter on the computed value (window thing)
% h = fspecial('gaussian',max(1, fix(6*sigma)),sigma);
h = blurMask;
Sx2 = imfilter(Ix2, h);
Sy2 = imfilter(Iy2, h);
Sxy = imfilter(Ixy, h);

alpha = 0.04;
R = (Sx2.*Sy2 - Sxy.^2)-alpha*(Sx2 + Sy2).^2;
R_sort = sort(reshape(R,[],1),'descend');

threshold = mean(R_sort(1:min([Rthresh,length(R_sort)])));

% Remove low gardients, graythresh makes the threshold adapt to image
highR = R>threshold;

% find local max: Non-maximum suppression, finds next highest point in
% neighborhood and checks whether this point in R is greater than that
nearest_max = ordfilt2(R, 8, [1 1 1;1 0 1;1 1 1]);
local_maxima = R>nearest_max;

% get interest points: satisfy both high gradient and is local maximum
% around the neighborhood
output = highR & local_maxima;

% remove interest points near image boundary
boundary = patch_size+1;
output(1:boundary,:)=0;
output(end-boundary:end,:)=0;
output(:,1:boundary)=0;
output(:,end-boundary:end)=0;

[r, c]=find(output);
intere_pt=[c';r']; % return interest points matrix
figure; imshow(img);
hold on;plot(c,r,'ys') % c-xaxis; r-yaixs
end
```

T. Code for SIFT descriptor generation

```
function [features] = get_features(img, x, y, feature_width)
% Returns a set of SIFT feature descriptors for points(x_i,y_i).
% 'x' and 'y' are n*1 vectors of x and y coordinates

features = zeros(length(x), 128);
```

```

gaussian = fspecial('gaussian', feature_width, 1);
[Gx, Gy] = gradient(gaussian);

cellSize = feature_width / 4;

Ix = imfilter(img, Gx);
Iy = imfilter(img, Gy);

% calculate orientation of pixels
orientation = atan2(-Iy,Ix)*180/pi;
mag = sqrt(Ix .* Ix + Iy .* Iy);

reduce_g = fspecial('gaussian', feature_width, feature_width/2);
for i = 1:length(x)
    currentX = x(i) - 2*cellSize;
    for j = 1:4
        currentY = y(i) - 2*cellSize;
        for s = 1:4
            currentCell = orientation(currentY:(currentY + cellSize - 1), currentX:(currentX
                 + cellSize - 1));
            currentMag = mag(currentY:(currentY + cellSize - 1), currentX:(currentX +
                cellSize - 1));
            currentMag = imfilter(currentMag, reduce_g);
                for k = -4:1:3
                    mask = currentCell >= k*45 & currentCell < (k+1)*45;
                    mag_masked = currentMag(mask);
                    histo = nnz(mag_masked);
                    features(i, (j - 1) * 32 + (s - 1) * 8 + k + 5) = histo;
                end
            currentY = currentY + cellSize;
        end
        currentX = currentX + cellSize;
    end
    total = sum(features(i,:));
    features(i, :) = features(i, :)/total;
end
end
```

U. Code for nearest neighbour matching of interest points

```

function [match, confidence, dist, ratio] = knn_match(featureA, featureB, threshold)

dist = zeros(size(featureA,1),size(featureB,1));
match = [];
confidence = [];
k = 1;
for i = 1:size(featureA,1)
    for j = 1:size(featureB,1)
```

```
        dist(i,j) = sqrt(sum((featureA(i,:)-featureB(j,:)).^2));
    end
    [dist_sort, idx_sort] = sort(dist(i,:), 'ascend');
    % Nearest Neighbor Distance Ratio Test
    ratio = dist_sort(1)/dist_sort(2);
    if ratio < threshold
        match(k,1) = i; % match point for img A
        match(k,2) = idx_sort(1); % match point for img B
        confidence(k) = ratio;
        k = k+1;
    end
end

end
```

V. Code for homography estimation with Direct Linear Transform

```
function H = get_homography(corre1,corre2)
% input points vector must be 2xN size and at least contain 4 points

n = size(corre1,2);
A = zeros(2*n,9);

for i = 1:n % for loop to create A matrix where Ah=0
    A(2*i-1,:) = [corre1(:,i)',1,0,0,0,-corre1(:,i)'.*corre2(1,i),-corre2(1,i) ];
    A(2*i,:) = [0,0,0,corre1(:,i)',1,-corre1(:,i)'.*corre2(2,i),-corre2(2,i) ];
end

if n>=4 % minimum number of equations to solve 8 unknows in H
    [~,~,V] = svd(A,0);
    H = reshape(V(:,end),[3 3])';
else
    error('Not enough points');
end
H = H./H(end,end);
end
```

W. Code for outlier detector for points projected with H

```
% Outlier detection
% Compare histogram of hue of interest point areas in feature space
% Threshold distance to determine inlier/outlier

colourB = im2double(imread(imgB));
[numR, numC] = size(colourB(:,:,1));
shift = patch_size/2; % Use patch size to determine area size
bin = 360; % Hue as 360 degrees
```

```
colourDist = zeros(size(testPoints2,1), 1);

for i = 1 : size(testPoints2,1)
    % patch around actual pixel
    left_boundB = testPoints2(i,1)-shift;
    right_boundB = testPoints2(i,1)+shift;
    up_boundB = testPoints2(i,2)-shift;
    low_boundB = testPoints2(i,2)+shift;

    % patch around projected pixel coordinate in actual image
    left_boundProj = proj_ptB(i,1)-shift;
    right_boundProj = proj_ptB(i,1)+shift;
    up_boundProj = proj_ptB(i,2)-shift;
    low_boundProj = proj_ptB(i,2)+shift;

    % reduce patch size if area extend outside image
    if left_boundB < 1 || up_boundB < 1 || left_boundProj < 1 || up_boundProj < 1
        reduce = min([left_boundB, up_boundB, left_boundProj, up_boundProj]);
        left_boundB = left_boundB + reduce;
        right_boundB = right_boundB - reduce;
        up_boundB = up_boundB + reduce;
        low_boundB = low_boundB - reduce;

        left_boundProj = left_boundProj + reduce;
        right_boundProj = right_boundProj - reduce;
        up_boundProj = up_boundProj + reduce;
        low_boundProj = low_boundProj - reduce;
    end

    if right_boundB > numC || low_boundB > numR || right_boundProj  > numC || low_boundProj
          > numR
        reduce = max([right_boundB - numC, low_boundB - numR, right_boundProj - numC,
            low_boundProj- numR]);
        left_boundB = left_boundB + reduce;
        right_boundB = right_boundB - reduce;
        up_boundB = up_boundB + reduce;
        low_boundB = low_boundB - reduce;

        left_boundProj = left_boundProj + reduce;
        right_boundProj = right_boundProj - reduce;
        up_boundProj = up_boundProj + reduce;
        low_boundProj = low_boundProj - reduce;
    end


    colourPatchB = colourB(up_boundB : low_boundB, left_boundB : right_boundB, :);
    colourPatchProj = colourB(up_boundProj : low_boundProj, left_boundProj: right_boundProj,
         :);
    HSV_B = rgb2hsv(colourPatchB);
    HSV_proj = rgb2hsv(colourPatchProj);
    HueB = reshape(HSV_B(:,:,1),1,[]);
```

```
    HueProj = reshape(HSV_proj(:,:,1),1,[]);
    H_histB = histcounts(HueB * 360 , bin); % Histgram of hue
    H_histProj = histcounts(HueProj * 360 , bin); % Histgram of hue
    colourDist(i,1) = pdist2(H_histB, H_histProj); % Eucledian distance
end

C_thresh = 3;

outliers = find(colourDist > C_thresh);
inliers = find(colourDist < C_thresh);

figure;showMatchedFeatures(colourB, colourB, b(inliers,:), proj_ptB(inliers,:),'montage');
figure;showMatchedFeatures(colourB, colourB, b(outliers,:), proj_ptB(outliers,:),'montage');
```

X. Code for normalised 8-point algorithm

```
function [F,e1,e2] = get_fMatrix(pt_l, pt_r)
% compute fundatmental matrix
% (x_l)'F(x_r) = 0 where F is the 3x3 fundamental matrix
% F = [a b c; d e f; g h m];
% 9 components; Rank 2; 7 degrees of freedom
% F captures the relationship between the corresponding points in two views
% require at least 8 points (altho 9 unknows but its homogeneous system)

% input arg:
% pt_l & pt_r are 2xN matching points matrix
% N at least be 8

% output:
%          F      - The 3x3 fundamental matrix such that x2'*F*x1 = 0.
%          e1     - The epipole in image 1 such that F*e1 = 0
%          e2     - The epipole in image 2 such that F'*e2 = 0

if size(pt_l,2)<3
    pt_l = [pt_l ones(size(pt_l,1),1)]';
    pt_r = [pt_r ones(size(pt_r,1),1)]';
end
% Normalise each set of points so that the origin
% is at centroid and mean distance from origin is sqrt(2).
% normalise2dpts also ensures the scale parameter is 1.
[pt_l, T1] = normalise2dpts(pt_l);
[pt_r, T2] = normalise2dpts(pt_r);
% by expanding (pt_l)'*F*(pt_r), rewrite to A*f
A = [pt_r(1,:)'.*pt_l(1,:)'   pt_r(1,:)'.*pt_l(2,:)'  pt_r(1,:)' ...
    pt_r(2,:)'.*pt_l(1,:)'   pt_r(2,:)'.*pt_l(2,:)'  pt_r(2,:)' ...
    pt_l(1,:)'            pt_l(2,:)'           ones(size(pt_l,2),1) ];

[~,~,V] = svd(A,0);
F = reshape(V(:,end),[3,3])';
```

```
34
35 % Enforce constraint that fundamental matrix has rank 2 by performing
36 % a SVD then reconstructing with the two largest singular values.
37 [U,D,V] = svd(F,0);
38 F = U*diag([D(1,1) D(2,2) 0])*V';
39
40 % Denormalise
41 F = T2'*F*T1;
42
43 [U,~,V] = svd(F,0);
44 e1 = hnormalise(V(:,3));
45 e2 = hnormalise(U(:,3));
46 end
```

Y. Code for evaluation and projection of epipolar line

```
1  % calculate Epipolar line from F for every point in the image
2
3  close all
4  pt_1 = [a(1:n,:)';ones(1,n)]; % matched points from imgA
5  pt_2 = [b(1:n,:)';ones(1,n)]; % matched points from imgB
6  figure; imshow(imgA); hold on
7  plot(matchedPoints1(1,:), matchedPoints1(2,:),'r*');
8  figure; imshow(imgB); hold on
9  plot(matchedPoints2(1,:), matchedPoints2(2,:),'r*');
10
11 for i = 1:size(pt_1,2)
12     line1 = F'*pt_2(:,i); % Epipolar lines in image1: projection from imgA to B'
13     line1 = line1 ./ sqrt ( repmat ( line1 (1 ,:).^2 + line1 (2 ,:).^2 ,[3 1]));
14     line2 = F*pt_1(:,i); % Epipolar lines in image2: projection from imgB to A
15     line2 = line2 ./ sqrt ( repmat ( line2 (1 ,:).^2 + line2 (2 ,:).^2 ,[3 1]));
16     point_x1 = [0 size(imgA,2)];
17     point_y1 = [(-line1(1)*point_x1(1)-line1(3))/line1(2)...
18         (-line1(1)*point_x1(2)-line1(3))/line1(2)] ;
19     point_x2 = [0 size(imgB,2)];
20     point_y2 = [(-line2(1)*point_x2(1)-line2(3))/line2(2)...
21         (-line2(1)*point_x2(2)-line2(3))/line2(2)] ;
22     figure(1);hold on
23     plot(point_x1,point_y1);
24     figure(2);hold on
25     plot(point_x2,point_y2);
26 end
```