CS 577 – Introduction to Computer Vision

Homework – 7

Name: Sridhar Mocherla

PART – A:

1. We use the homogeneous least squares method in RANSAC to compute the 3x3 homography matrix. The root mean square error between the inlier points and the actual points is 0.9466.

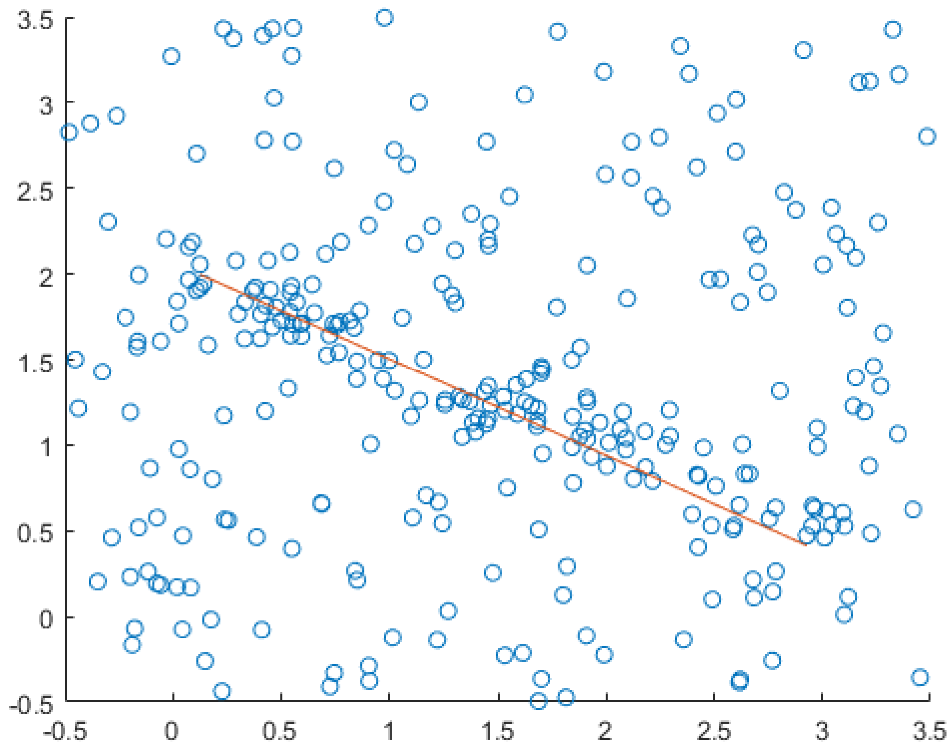

Figure 1: Line fitting using RANSAC

The equation of the line obtained is give as

$$0.2388x + 0.4227y - 0.8743y = 0$$

There is a total of 79 points in the inliers. We use d=75 and distance threshold as 0.3 and we perform K=100 iterations.

PART – B:

1. We manually click on 8 keypoints on each of the slide/frame image pairs and record the coordinates in slide/frame.txt (for each pair separate set of files).

   We take 4 points from these 8 points and compute the homography matrix using two methods.

   In the first method, we create the 8x9 (n=4) matrix A and find the minimum eigen value of A'*A and its corresponding eigen vector which is 9x1 unit vector.

   We show the results from Figure 2 to Figure 4 for the first method. The points in the slide image are indicated using red asterisks (*). These are the points manually clicked.

   The manually clicked points in the frame image are shown using red diamonds. The frame points computed using the homography matrix are indicated using yellow circles and some of them overlap with the red diamonds (especially the ones used to compute the H matrix).

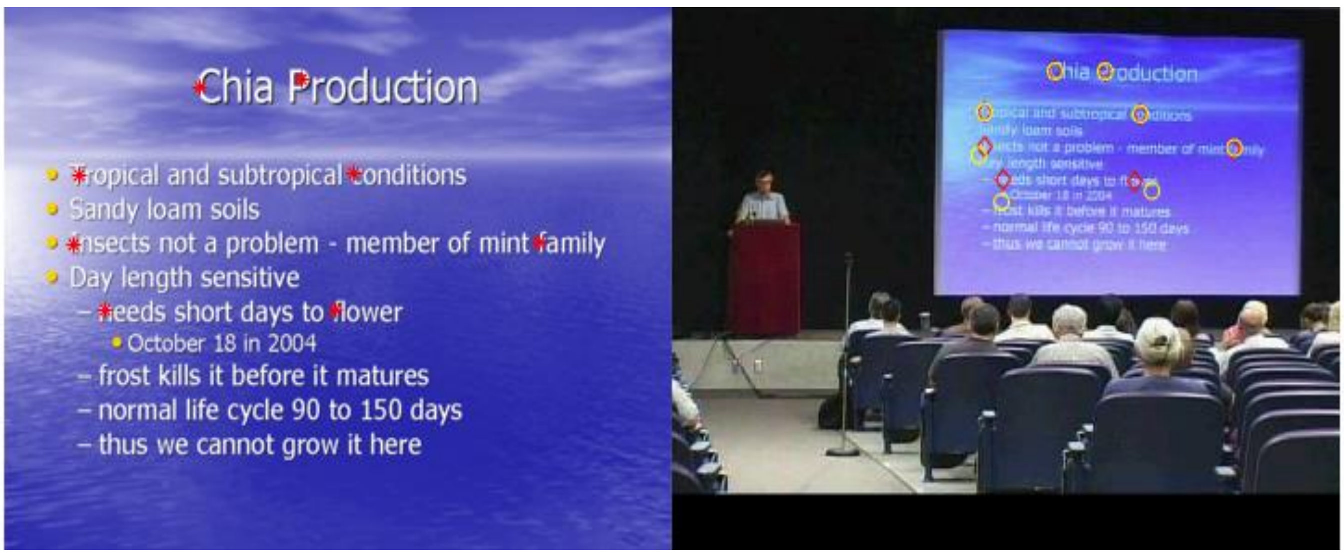

Figure 2: Slide/Frame 1 – Homography using first method

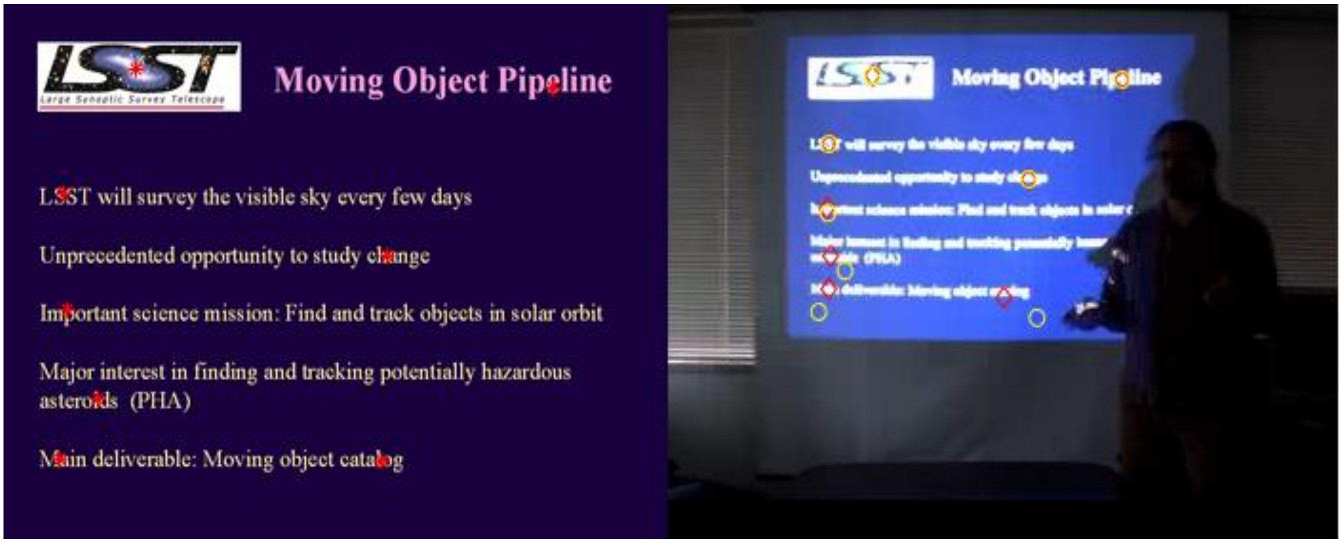

Figure 3: Slide/Frame 2 – Homography using first method

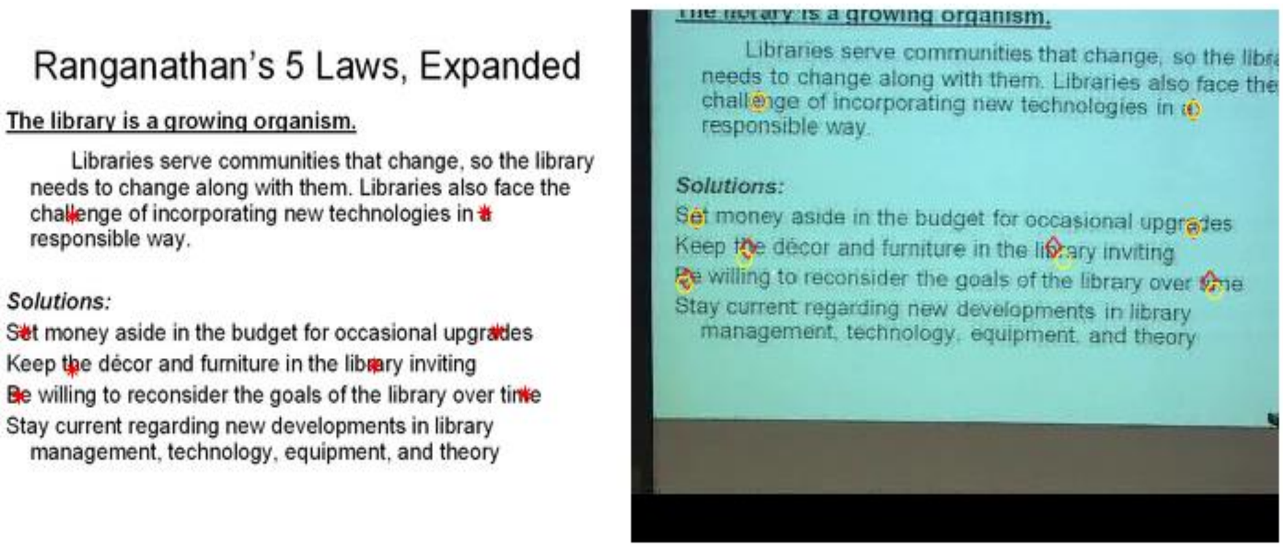

Figure 4: Slide/Frame 3 – Homography using first method

Figure 5-7 shows the results obtained when the homography matrix is computed using Direct Linear Transformation (DLT) method. In MATLAB, we use SVD (Singular Value Decomposition) to decompose the 8x9 matrix and obtain the 3x3 homography matrix.

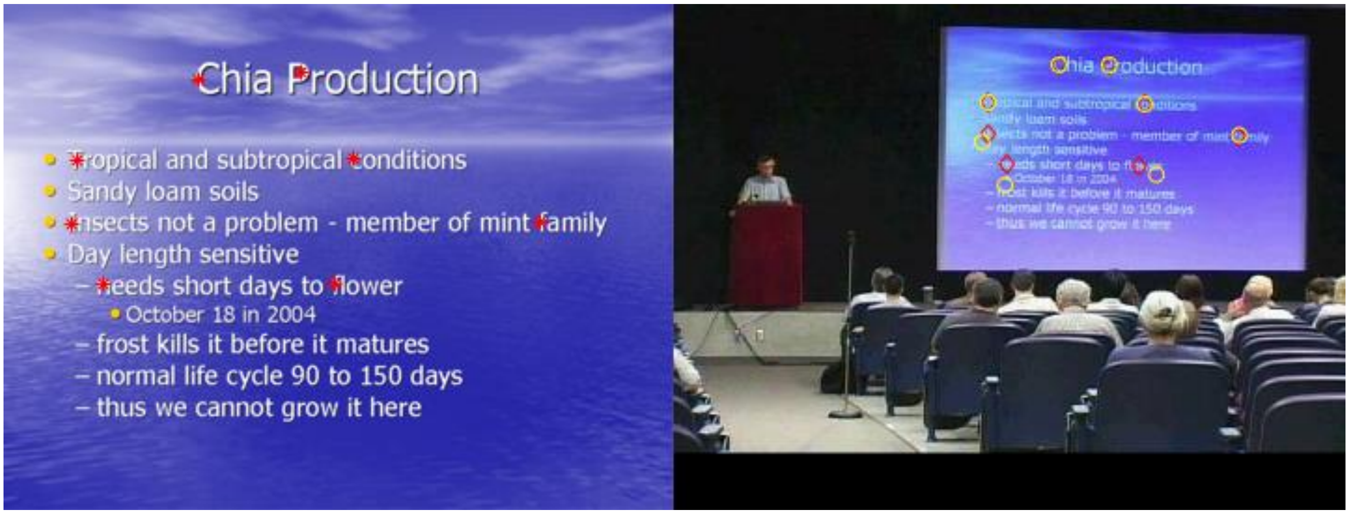

Figure 5: Slide/Frame 1 – Homography using DLT

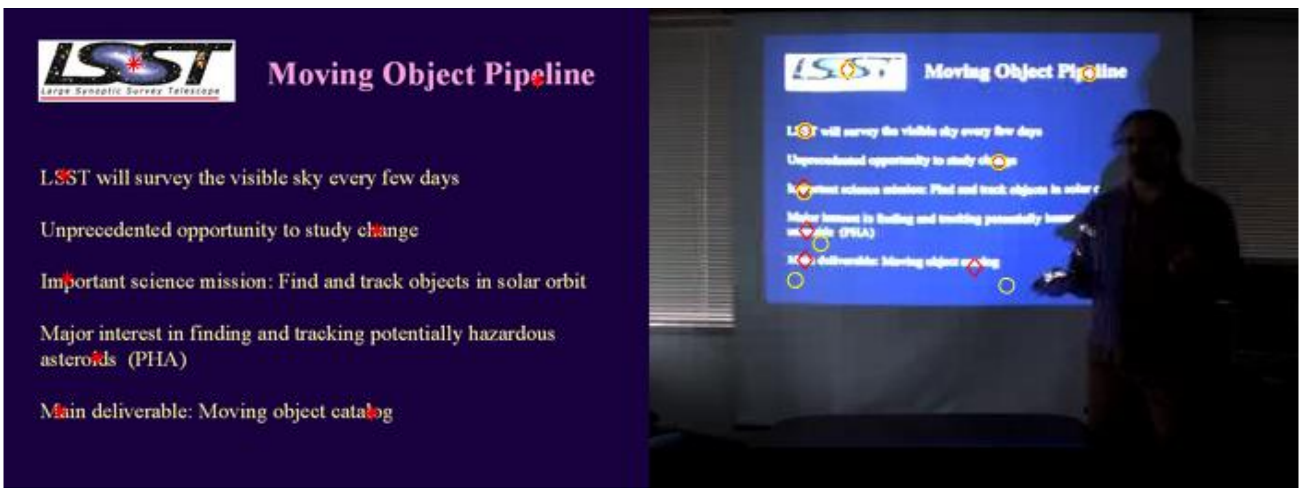

Figure 6: Slide/Frame 2 – Homography using DLT

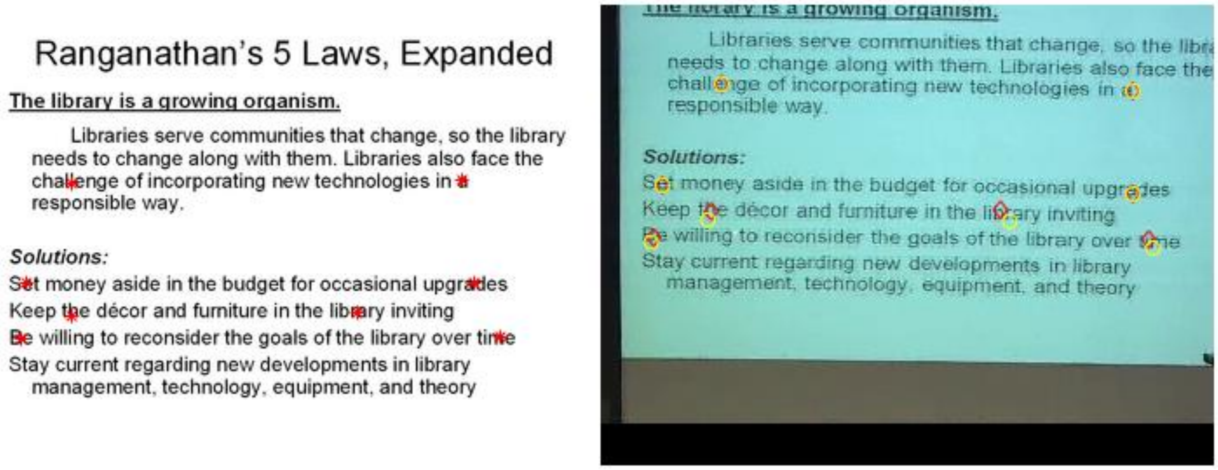

Figure 7: Slide/Frame 3 – Homography using DLT

PART – C:

SIFT Keypoint matching using RANSAC and Homography

For each slide/frame image pair, we retrieve the coordinates for the keypoints using the VLFeat library. We then compute all the matches using Euclidean distance with threshold based on Nearest Neighbor/Second Nearest Neighbor ratio greater than 0.7 for matches. From these matches, we perform RANSAC by taking 4 random points and computing homography using both the methods possible. We repeat this for a certain number of iterations. In each iteration, we also compute the error between actual frame keypoints and estimated frame keypoints (obtained using homography matrix) and points with error less than 49(threshold) are considered inliers.

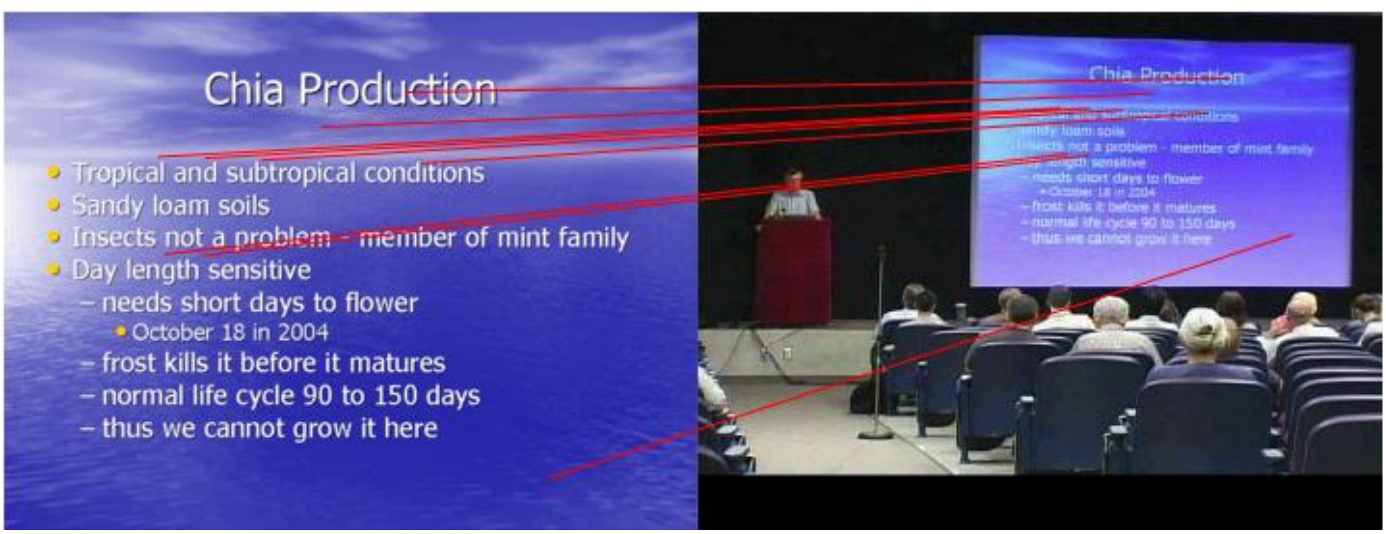

Figure 8: Best KP matches using RANSAC/Homography on SIFT matches for slide/frame 1 using first method. (31 matches)

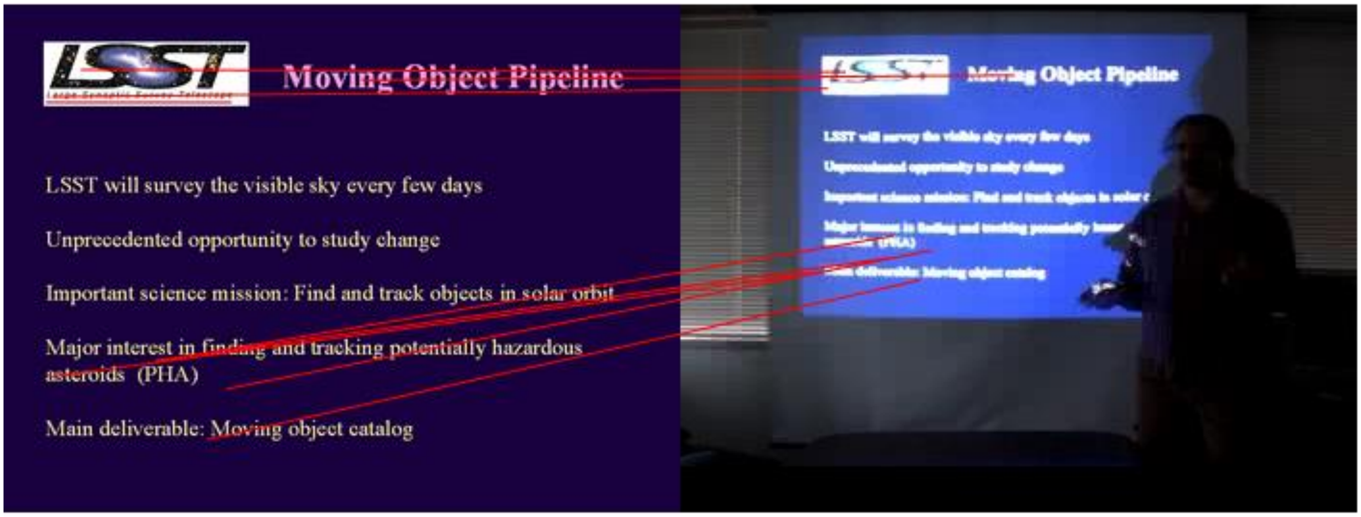

Figure 9: Best KP matches for slide/frame 2 pair based on homography computed using first method. (8 matches)

Figure 10: Best KP matches for slide/frame 3 pair based on homography computed using first method.

In Figure 8-10, we show the best matches obtained by performing RANSAC on matches which agree on homography computed using the first method (eigen vectors).

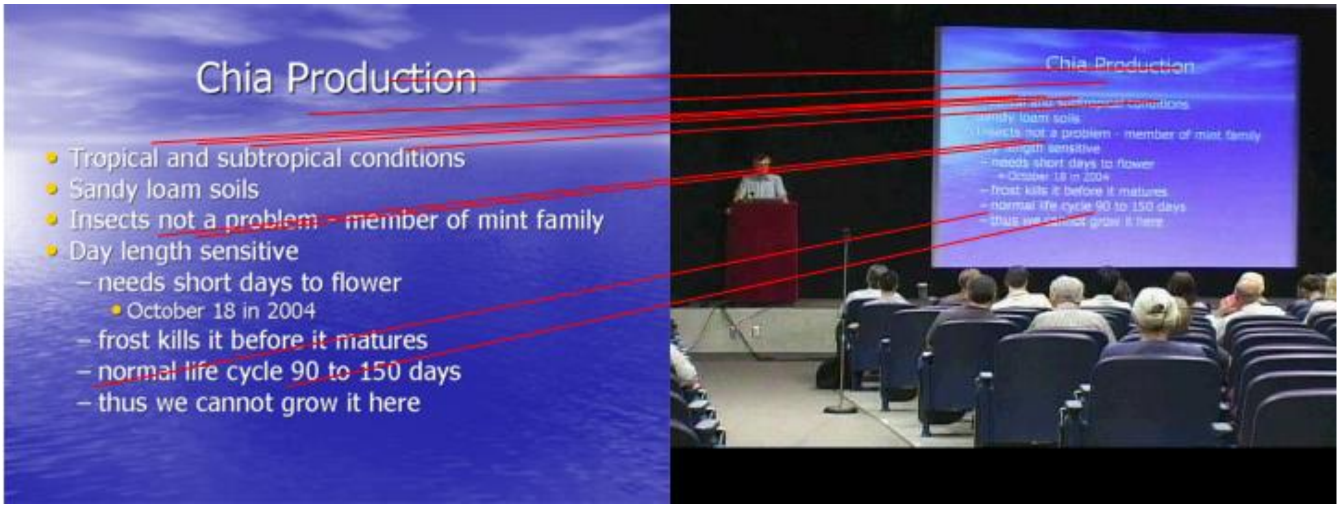

Figure 11: Best KP matches for slide/frame 1 based on homography computed using DLT

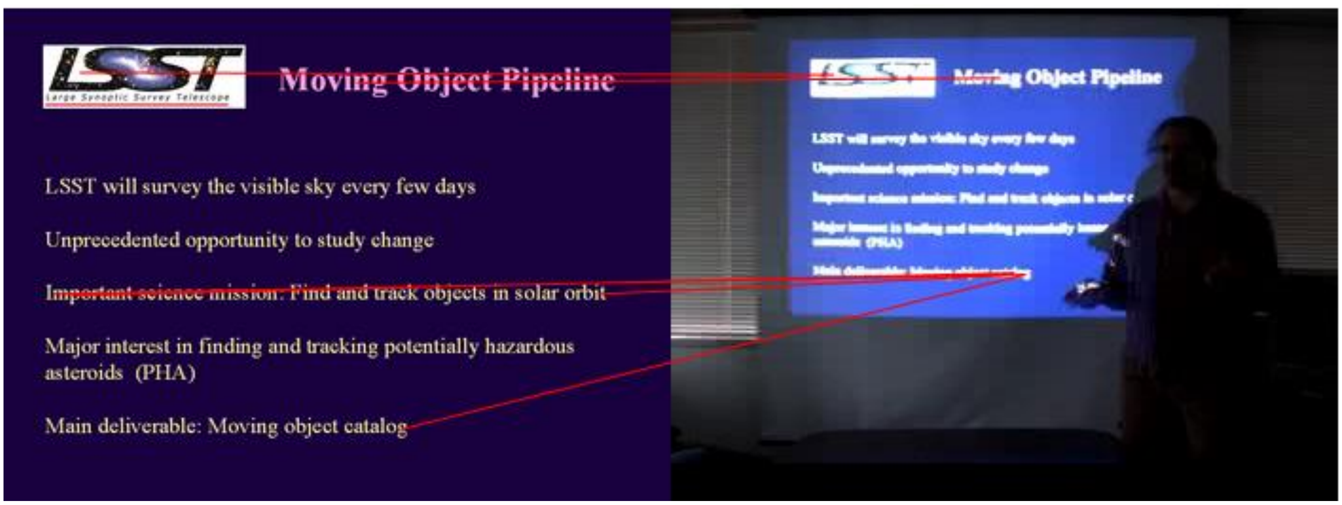

Figure 12: Best KP matches for slide/frame 2 based on homography computed using DLT

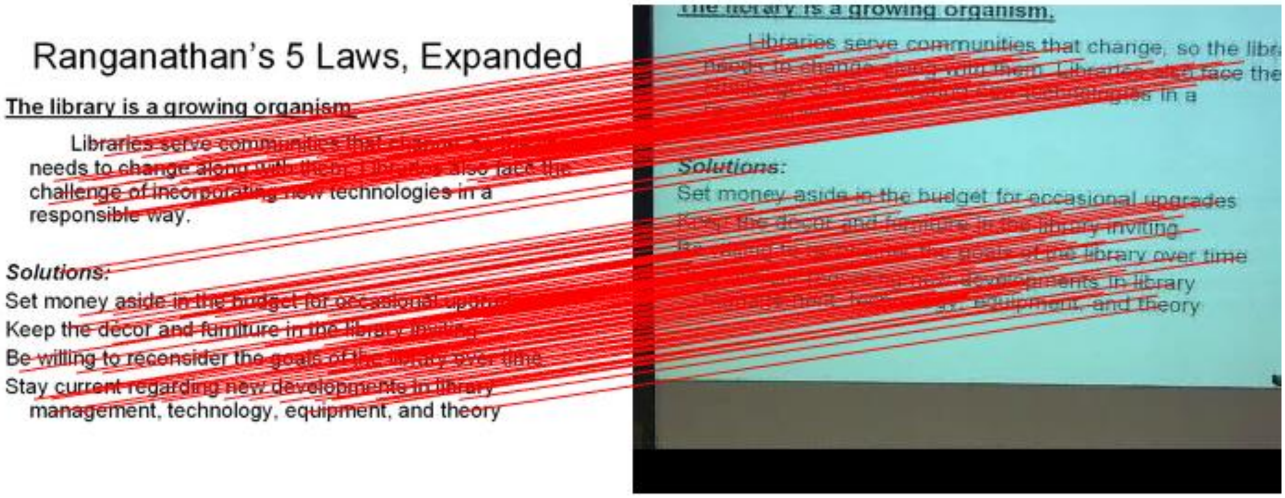

Figure 13: Best KP matches for slide/frame 3 based on homography computed using DLT

Figure 11-13 show the results obtained using homography computed using DLT method. In the case of Slide/Frame 1, DLT performs better with accurate matches. As we can see, the matches are much more accurate than observed in the previous assignment and also more in number for each pair. But that does not seem to be the case when it comes to Slide/Frame 2 pair, where DLT performs worse. In the case of slide/frame 3, the matches are mostly same in terms of number and accuracy.

In terms of difference in solving for homography, in the first method we use the homogeneous least squares (eigen vector corresponding to least eigen value) whereas in DLT, we use singular value decomposition (SVD) to decompose the 2nx9 matrix into three 3 matrices, one of which contains the homography matrix.

Appendix:

Code snippets:

Line fitting using RANSAC: (PART A)

```
%Fit set of points to a line using RANSAC -
% data - set of points
% n - sample size
function[inliers] = fit_line_ransac(data,n,k,threshold,d)
sample = zeros(n,2);
indices = zeros(n,1);
points = data;
close_points = [];

index = randi([1 size(data,1)]);
sample(1:2,:) = data(1:2,:);
indices(1:2) = 1:2;
```

```
for i=1:k

    [m,c] = fitLine(sample);
    for j=1:size(data,1);
        if ismember(j,indices)==0
            point = data(j,:);
            distance = findDistance(m,c,point);
            if distance <=threshold
               close_points = [close_points;point];
            end
        end
    end
    if size(close_points,1) >=d
        data = [];
        data = close_points;
        inliers = close_points;
        close_points=[];
        indices=[];

        indices = zeros(size(data,1),1);
    else
        data = points;
        close_points = [];
        indices = zeros(size(data,1),1);
    end
    for j=1:n
           index = randi([1 size(data,1)]);
           sample(j,:) = data(index,:);
           indices(j) = index;
    end
    for j=1:n
           index = randi([1 size(data,1)]);
           sample(j,:) = data(index,:);
           indices(j) = index;
    end
end
```

Computing homography (both methods): (PART B)

```
function[] = compute_homography(i,slide_image,frame_image,method)
fid = fopen(strcat('slide',int2str(i),'.txt'));
        slide_points=[];
        line = fgetl(fid);
        while ischar(line)
           vals = strsplit(line)
           slide_points =
[slide_points;str2double(vals(1)),str2double(vals(2))];
           line = fgetl(fid);
        end

        fid = fopen(strcat('frame',int2str(i),'.txt'));
        frame_points=[];
        line = fgetl(fid);
        while ischar(line)
```

```
            vals = strsplit(line)
            frame_points =
[frame_points;str2double(vals(1)),str2double(vals(2))];
            line = fgetl(fid);
        end
        slide_subset = slide_points(1:4,:); %Take 4 points
        frame_subset = frame_points(1:4,:); %Take 4 points

        A=[];
        for j=1:4
            slide_p = slide_subset(j,:);
            frame_p = frame_subset(j,:);
            val1 = [slide_p(1) slide_p(2) 1 0 0 0 -slide_p(1)*frame_p(1) -
slide_p(2)*frame_p(1) -frame_p(1)];
            val2 = [0 0 0 slide_p(1) slide_p(2) 1 -slide_p(1)*frame_p(2) -
slide_p(2)*frame_p(2) -frame_p(2)];
            A =[A;val1];
            A = [A;val2]; %form 2nx9 matrix
        end
        if strcmp(method,'LS')
           [E,V] = eig(A'*A); %Least square solution
           H = E(:,1);
        else
            [U,S,V] = svd(A);%DLT method
            H = V(:,9);
        end
        H = reshape(H,3,3);
        fclose(fid);

        actual_frame_points = zeros(8,2);
        for j=1:8
           slide_p = slide_points(j,:);
           X = [slide_p(1);slide_p(2);1];
           F = H'*X;
           F = (F./F(3));%compute estimated points
           actual_frame_points(j,:) = [F(1) F(2)];
        end
        figure;
        imshowpair(slide_image,frame_image,'montage');
        hold on;
        plot(slide_points(:,1),slide_points(:,2),'*r');
        hold on;

        plot(frame_points(:,1),frame_points(:,2),'dr');
         plot(actual_frame_points(:,1),actual_frame_points(:,2),'oy');
end
```