CV ASSIGNMENT 2- SUBHASHREE RADHAKRISHNAN

1.Feature Tracking:

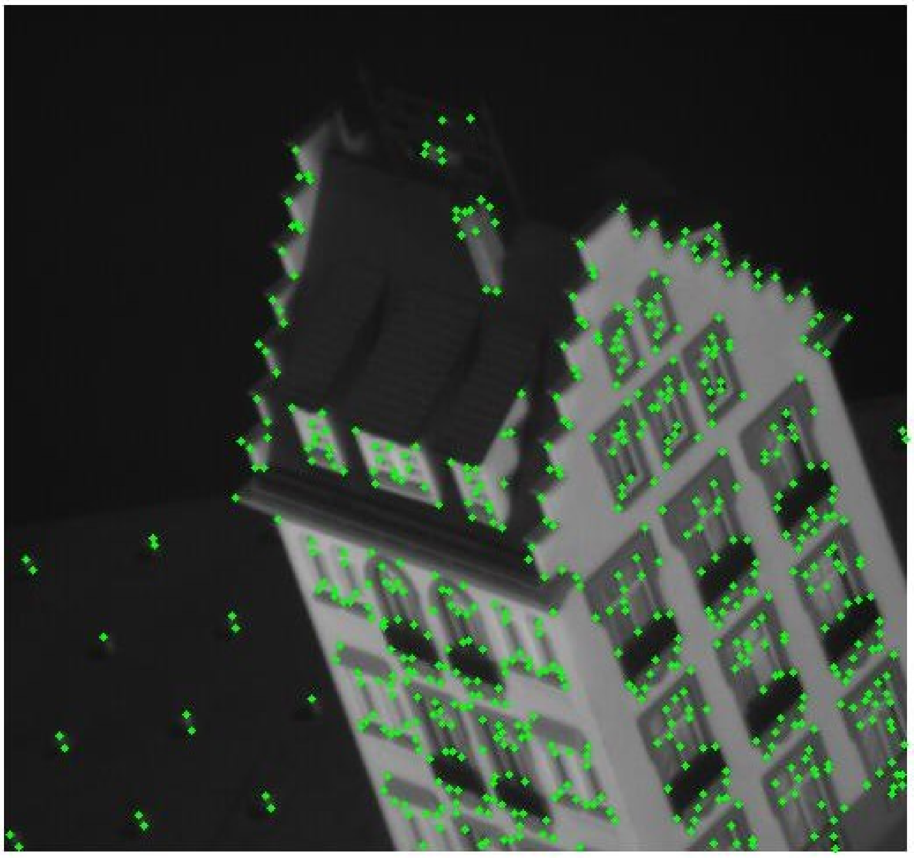
1.1Keypoint Detection:

Pseudo Code:

a)Input the image and do Gaussian filtering

b)Find the Ix and Iy gradients of the image

c)The har(cornerness function) is calculated using the equation which will give a score of the variations of the intensity at different pixels. `har=(IxIx.*IyIy)-((IxIy).^2)-(0.04*((IxIx+IyIy).^2))`
d) The points that are greater than zero and which are greater than a threshold(checking the intensity variation) from har. Lesser the threshold , greater will be the number of obtained points. `har = har.*double(har ==ordfilt2(har, N^2, true(N))) > .0001;`. The indices of the resulting har gives keypoints.
d)The keypoints found are plotted with the plot function.



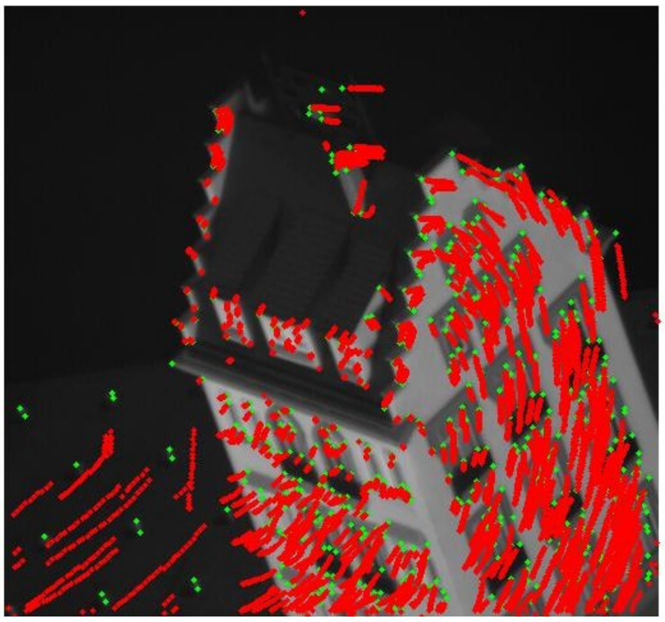
2. Tracking of the points:

Method:

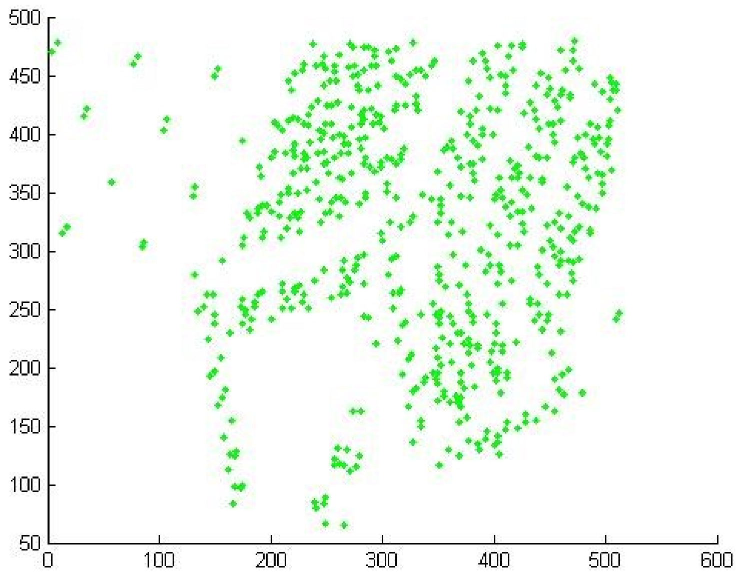
1. Lucas kanade iteration was performed for every keypoint on frame1 that was found using keypoint function and the respective keypoints were found for all the frames of the sequence. LKT basically

computes the displacement vector to be added to the old key points. This is done by solving the equations I(x,y,t)= I(x+u,y+v,t+1). When the displacement becomes smaller than a threshold we stop the iteration.(chosen was 0.5)

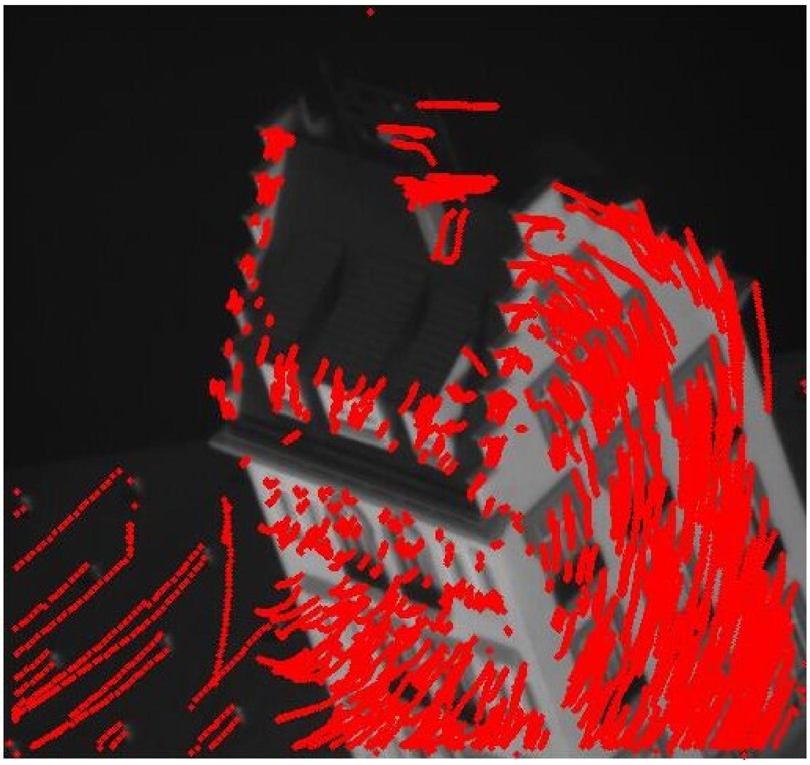
2. The lost points shown in the figure were tracked by plotting those points that had crossed the limits of the image in consecutive frames. The following plot gives the disappearing points on the first image.



The tracked points of the drifting key points over 50 frames



The resulting plot of matches



Plotting of Lost key points

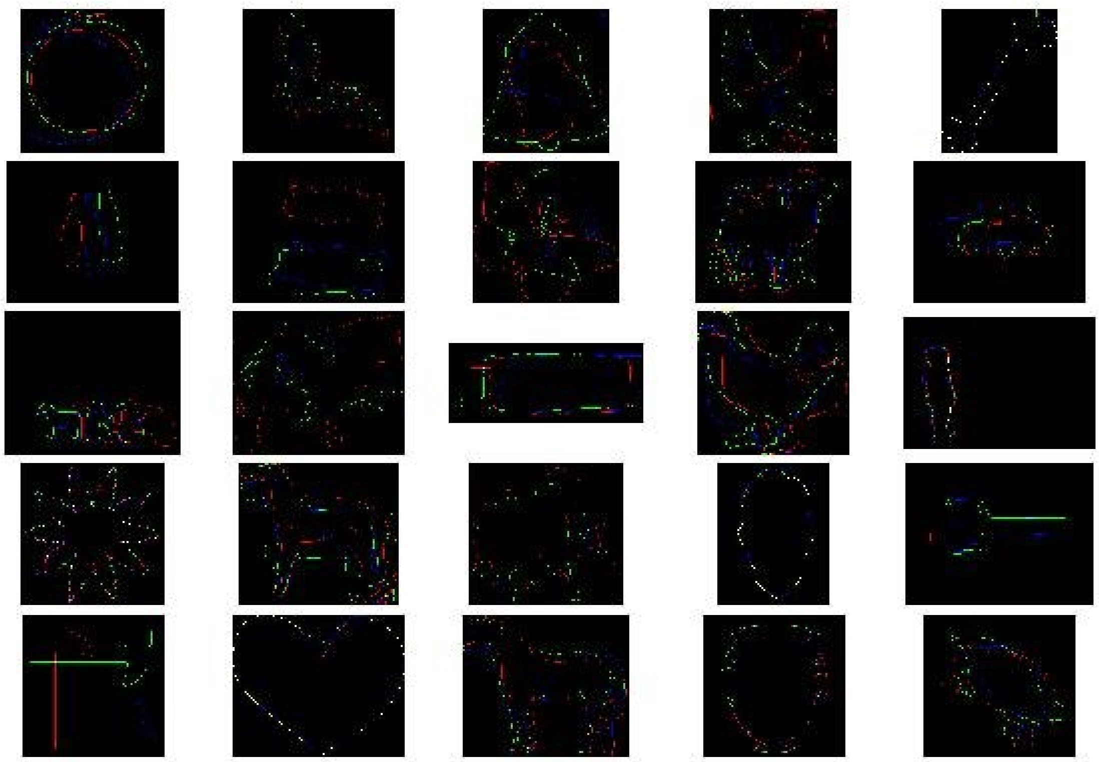
2.**Shape Alignment**

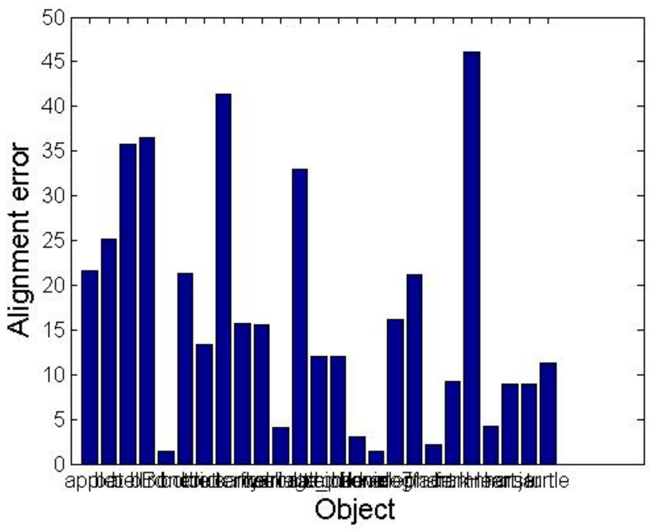
The shape alignment was achieved using affine transformation. The points greater than 0 were first obtained from the 2 images. The shortest distance and the indices corresponding to the short distances between the images were computed using bwdist. The initial transformation matrix was found with dx and dy which were calculated from mean of the set of points in each image.

Thus with the points in the data image and transformation, the new set of x,y points were found. Now the transformation matrix can be found by multiplying the inverse of matrix formed with data points in first image and the matrix containing the x',y'(points of reference image). This transformation is again applied to data points to get new points. The smallest distance from these data points are again deduced to get xi' and yi' and the loop continues. When the Euclidean distance between old transformation matrix and the new transformation matrix becomes less than a threshold(implying no transformation required) the iteration is stopped. The chosen threshold was 7.

Affine Transformation matrix=[m1 m4 0; m2 t1 0; m3 t2 1]

Where m1,m2,m3,m4 gives the parameters of the rotation matrix and t1 ,t2 gives the translational parameters.





The average error was found to be :18.61

The time elapsed was found to be 13.4 sec

3.**Feature Matching**

First Object: x1,y1,w1,h1,theta1

Second Object: x2,y2,w2,h2,o2

Given:[(u1,v1,s1,theta1);(u2,v2,s2,theta2)]

O2=theta2-theta1

h2=(s2/s1)*h1

w2=(s2/s1)*w1

we know that,

x+u=x’

y+v=y’

where x’,y’ corresponds to center points in the second image and u and v are the displacement vectors

x=x1-u1; y=y1-v1

where x and y corresponds to the actual positions of the keypoints in the first image.

X1,y1 corresponds to center points of bounding box of first image.

Now applying rotation and scaling to this keypoint gives the center points of the bounding box of second image.

Rot=[cos(theta2-theta1),-sin(theta2-theta1);sin(theta2-theta1) cos(theta2-theta1)]

Scale=s2/s1

Using the relation of affine transformation,

[x2;y2]=[u2;v2]+Rot*Scale*[(x1-u1);(y1-v1)]

x2=u2+(s2/s1)*(cos(o2)*(x1-u1)-sin(o2)*(y1-v1))

y2=v2+(s2/s1)*(sin(o2)*(x1-u1)+cos(o2)*(y1-v1))

Verification:

Only Rotation:

Scaling is 1 and u1, v1 becomes zero. Hence the resulting coordinates will be x2,y2 itself without translation.Thus verified in this case.

Only Scaling:

O2=0

U1=v1=0

Substituting in equations,

X2=u2+(s2/s1)*x1;   y2=v2+(s2/s1)*y1   which proves this equation is right as the x and y corrdinates alone are scaled.

Only Translation:

O2=0

S2/s1=1

Plugging in,

X2=u2+(x1-u1); y2=v2+(y1-u1)

Thus the equation holds true special cases (involving only rotation, translation and scaling).

b)

Method:

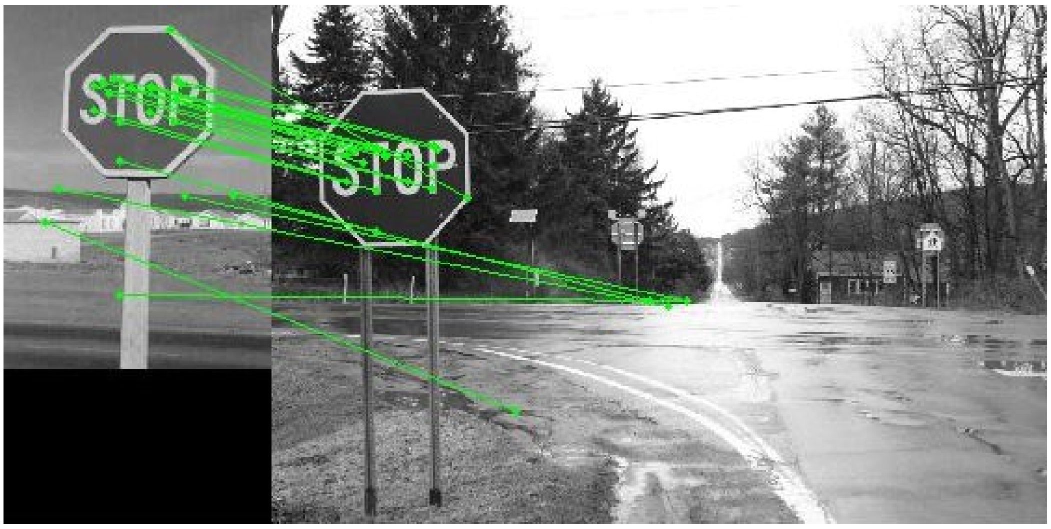
1)Take the two descriptors and find the minimum distance between the features using dist2 function which gives . This finds the nearest feature in descriptor 2 for every feature in descriptor 1.

2)Compare the computed distances for a feature in every column of descriptor 1 with a threshold. If it is lesser than the threshold, we take that as a matched keypoint. This is done so that minute movements can be easily tracked.

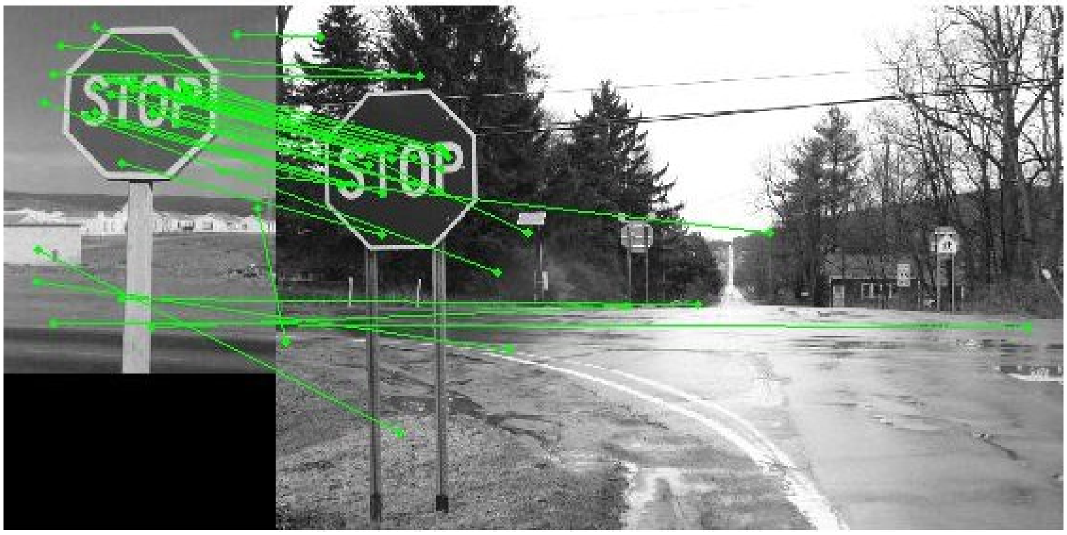
1. Thresholding with distance:
   Here the distances are compared with a threshold directly. The value chosen was 80000.
2. Thresholding with distance ratio :
   Here the distance ratio is compared with a threshold. Thus the ratio of minimum distance to second minimum distance is compared with a threshold of 0.7. This code is in matchKeypoints_new1.

As it can be seen from the below images, the thresholding with distance ratio gives better results with more number of matching points.

We take the corresponding index in frame 2 for the matched keypoint in image 2.



Key point Matching with thresholding using distance ratio



Key point Matching with thresholding using distance ratio

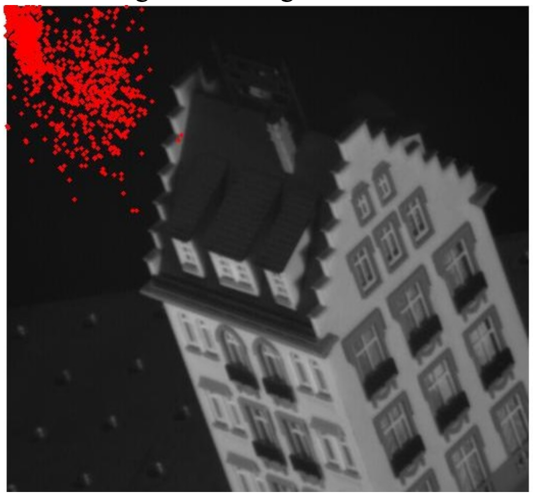
3)We parse the indices of the matched key points to plot matches which will plot lines between the keypoints of the two images.

Thresholding Without_distance ratio – Time elapsed= 0.43348 sec

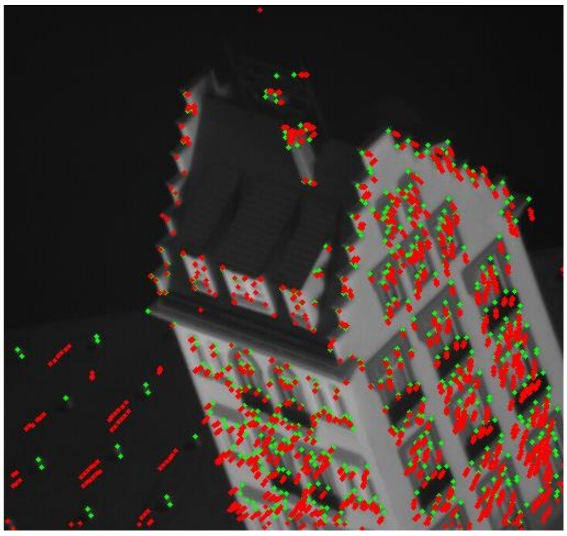
Threshoding with distance ratio-Time elapsed= 0.47126 sec

Extras:

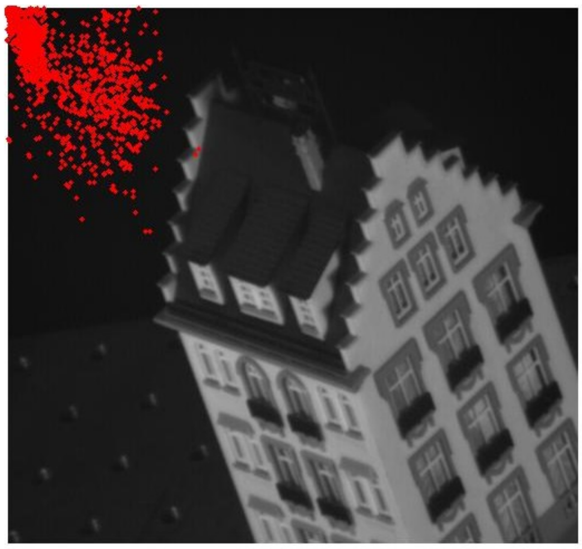
1. Iterative Lucas Kanade (Coarse To Fine Registration):
   These codes are given in LK_Optical flow. Here the drifting of keypoints is found in the last level of Gaussian pyramid. Thus the larger movements can be traced. The Lucas iteration was performed for the last level of the Gaussian pyramid by subscaling the initially found keypoints twice.The results of this at different frames are shown below. Its clear from the results shown below that the drift shown by coarse to fine registration is greater than the normal scale lucas iterative method.



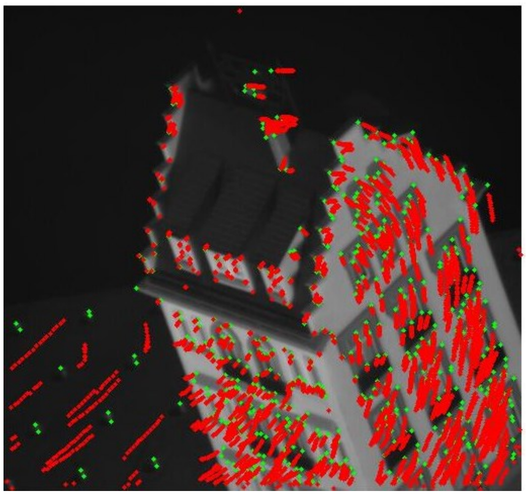
LKT done for 10 frames



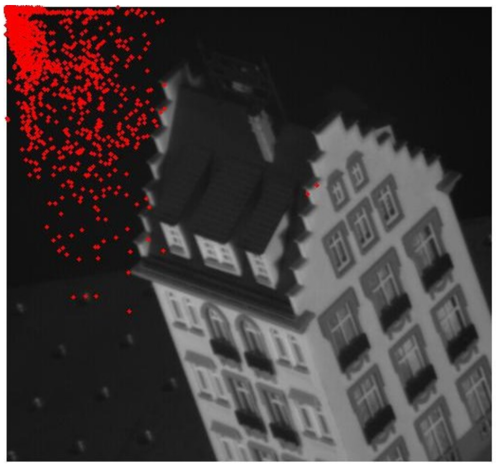
Single scale LK for 10 frames

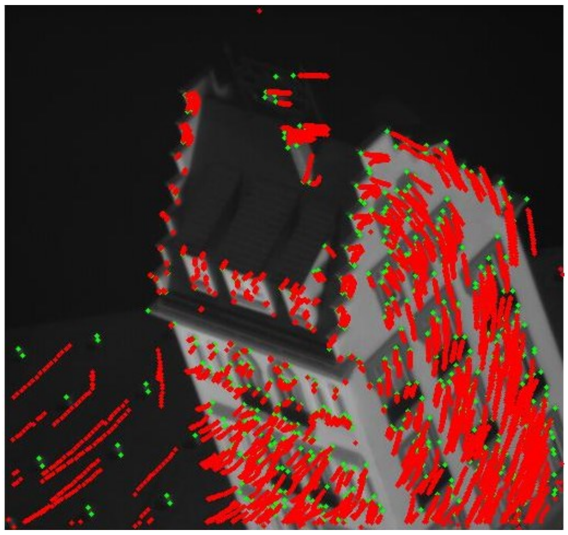


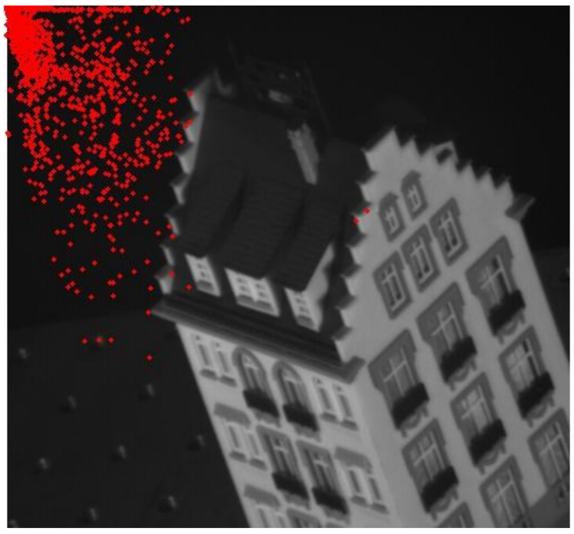
LKT done for 20 frames

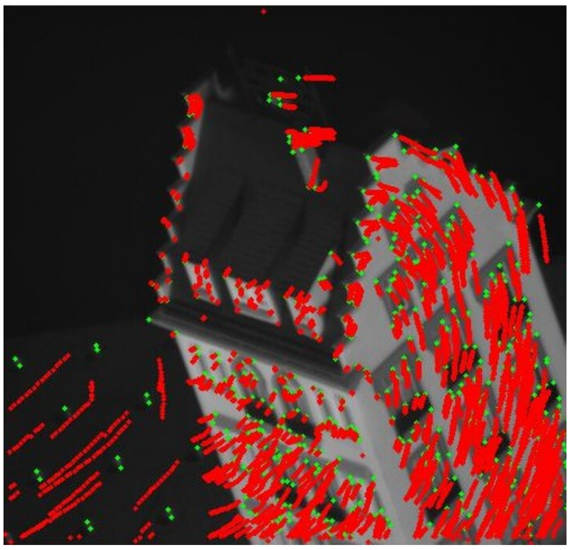


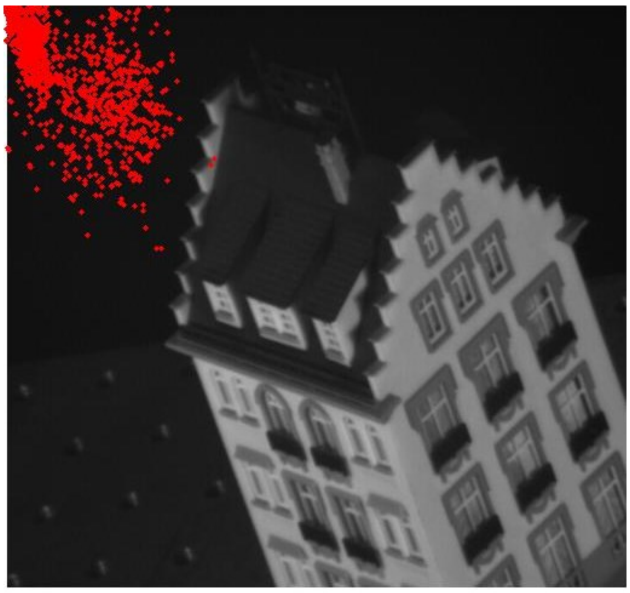
Single scale LK for 20 frames


LKT done for 30 frames


Single scale LK for 30 frames


LKT done for 40 frames


LKT done for 40 frames


LKT done for 50 frames

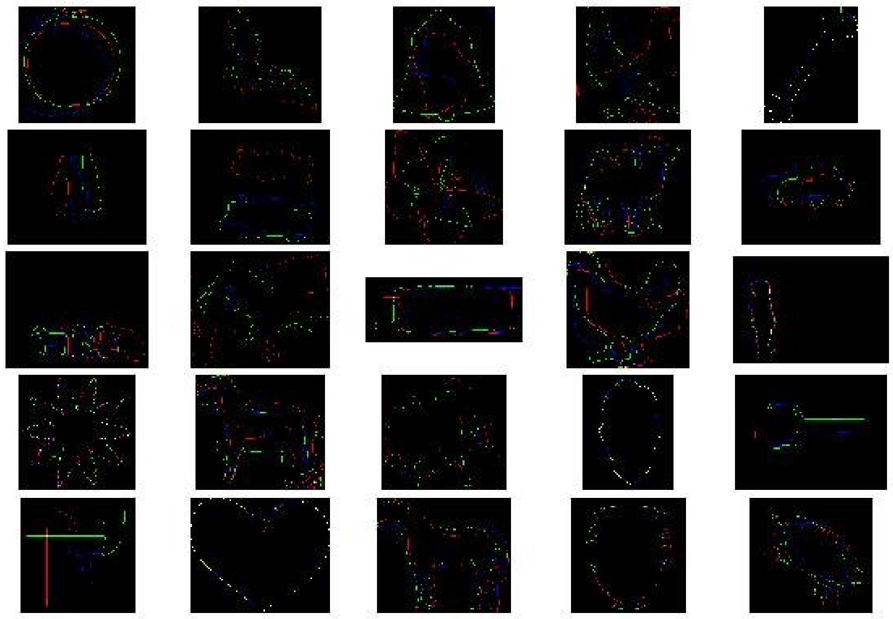
2. Lucas Kanade Multi scale Algorithm
   The u,v corresponding to previous level of Gaussian filter was mutilplied by 2 and was subjected to bilinear interpolation. This code is given in multi_scale folder. Basically lucas kanade iteration is performed for every Gaussian pyramid level , the u and v are deduced.

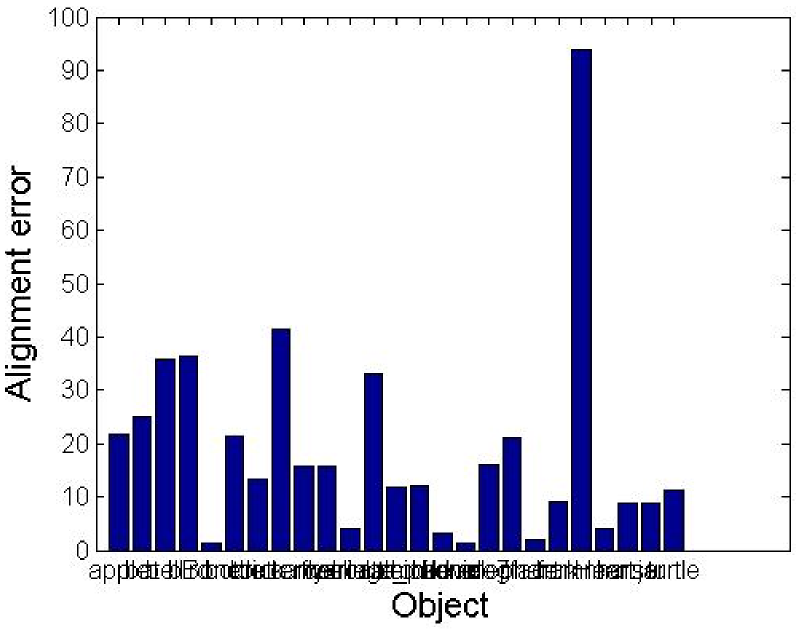
   Structure:
   The main code call the translation to find the new points. This function calls predictLK which perform iterative LKT for given gausian level and return u,v. This u and v are added to the parsed coodinates to find the new keypoints in the next frame.

   max flow: 0.1867 flow range: u = -0.007 .. 0.000; v = -0.187 .. 0.000

2.Alignment Improvisation:

The error reduced from 18.7 to 16.8 when some rotation techniques were implemented to the image initially. The SVD function was used to calculate the rotation of each of the images. The theta was then calculated by taking the difference of the two orientations. The image was rotated according to the orientation found. The hammer and the cock were properly flipped and rotated.





3.a) The additional search tree used was knn- search.

The minimum distances and their indices were found using knnsearch(double(dis2_new), double(dis1_new), 'k', 2, 'NSMethod', 'kdtree')

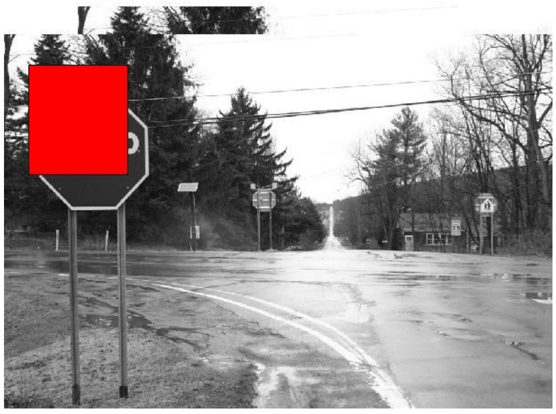
This function finds the neighbouring points by parsing through the adjacent nodes.

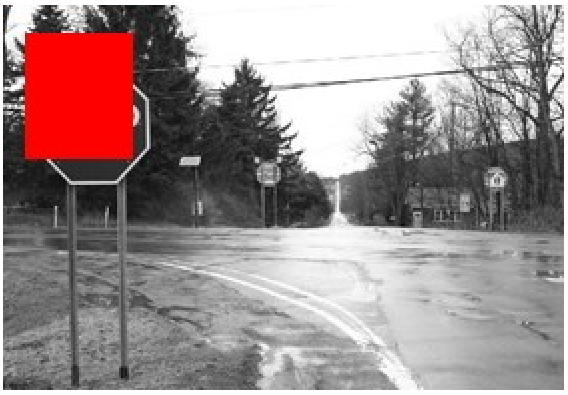
This code is present in matchKeypoints_new2.m

The total elapsed time was found to be : 410298 seconds.

As it is seen the time elapsed for the knn search was found to be the least compared to other two methods.

b)





As it can be seen from the results:

The bounding box was generated by identifying the affine transformation for different pairs of matched keypoints in the two frames. Randomly generated 3 keypoints from the obtained matches, were used for finding affine transformation, it gave good result of bounding the stop board. This code is present in matchKeypoints_new1_bbn.m.

Method:

1. Choose random 3 matched keypoints and their respective coordinates from both the frames. The frame contains positions, scale and orientation. The relation between coordinates in one image and the corresponding in the other used for finding was similar to that of second problem (affine transform equation).

2. Using these 3 keypoints, the affine transformation matrix and their 6 parameters can be deduced.
3. This transformation is then applied to a bounding box of the given coordinates ,transformed and displayed in the second frame using rectangle. The best 3 points to form affine transformation were deduced by comparing the limits of each matched point coordinate with

the boundary box dimensions. Thus those points which lie within the stop board were taken to train and form affine matrix.