ASSIGNMENT 3 – COMPUTER VISION

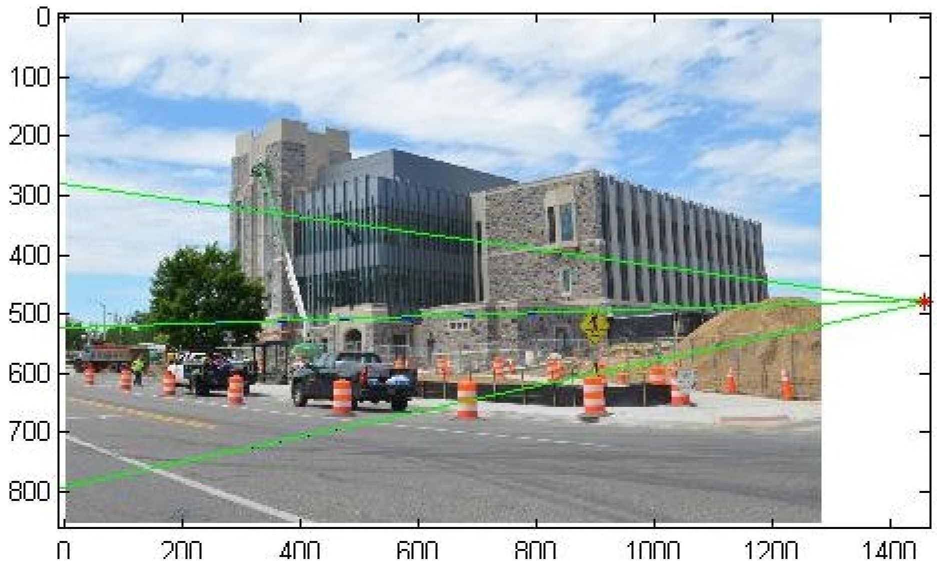
1. Single-View Metrology

1.The vanishing points were obtained by randomly plotting 3 lines.
2. The line parameters were substituted in the equation given below:

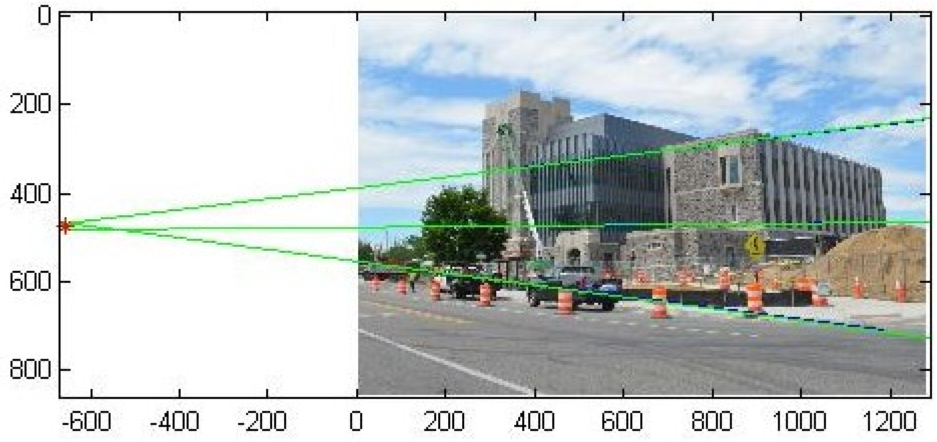
$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

3.The values of x and y were deduced .

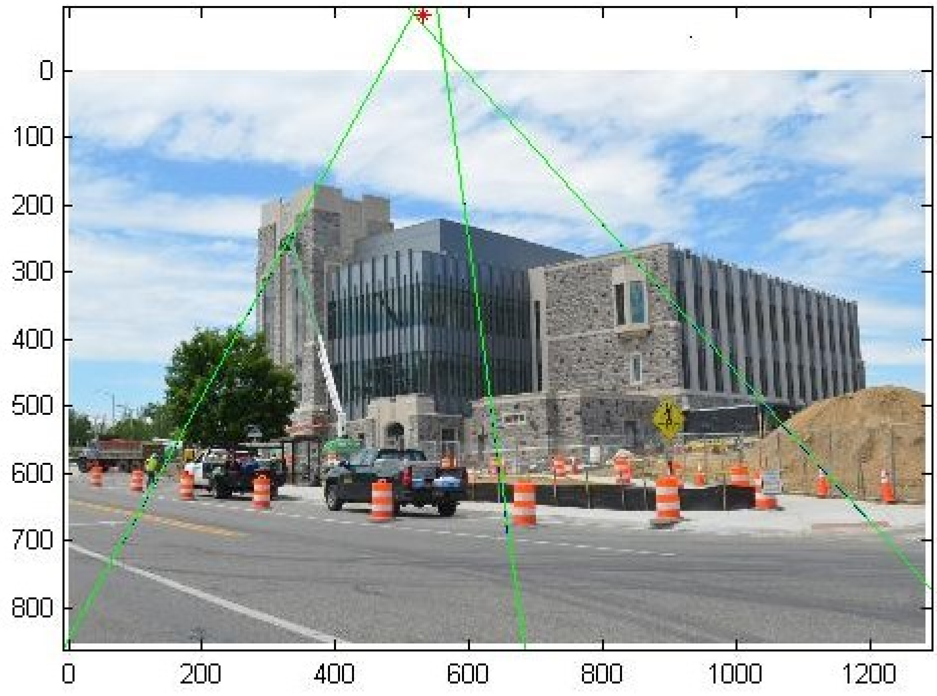
```
A=[lines(1,1),lines(2,1)];
B=[-lines(3,1)];
for i=2:(size(lines,2))
    A=vertcat(A,[lines(1,i),lines(2,i)]);
    B=vertcat(B,-lines(3,i));
end

size(A)
size(B)
vp=A\B;
```



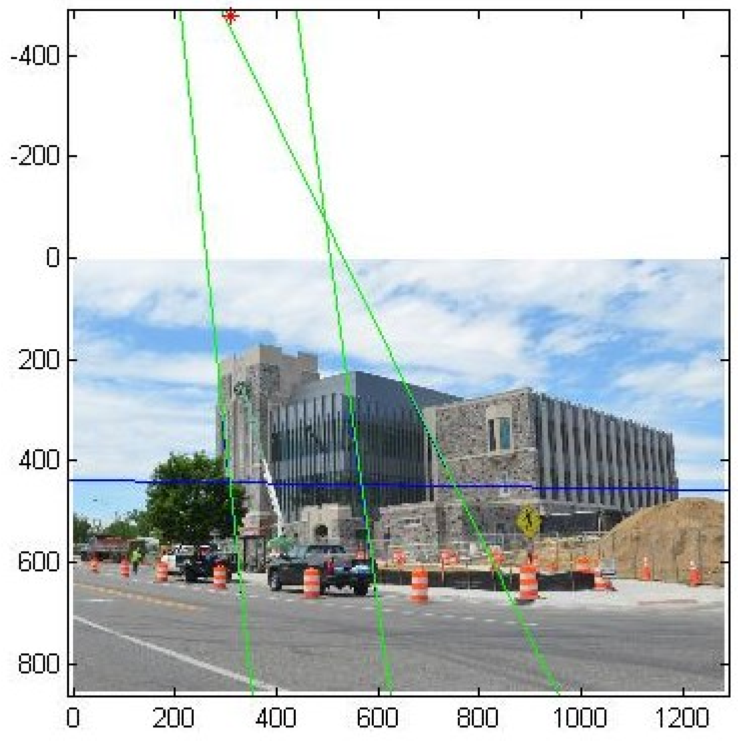
The right most vanishing point in the x direction



The Left Most Vanishing point



The Top most vanishing point in the Z direction



The Blue line gives the horizon line

The left most VP was =[1481.97,465.54,1]
The Right most point = [-53.4,5.66,1]
Top Most VP = [529.76,829.74,1]
The equation parameters of horizon line = [0.01,-1,460.92] This is normalized and hence $\sqrt{(a^2 + b^2)}$=1.
b. Intrinsic Parameter Caliberation:
Method:

1. The three vanishing points were calibrated as [x1,y1](VP1), [x2,y2](VP2), [x3,x3](VP3)
   VP1=[2298,1550,1];VP2=[-349.71,1495.99,1];VP3=[1025,-46.21,1]
2. Now these values were substituted in the equation given below:

   Eqn (1) – Eqn (2) $\Rightarrow (x_1 - u_0)(x_2 - x_3) + (y_1 - v_0)(y_2 - y_3) = 0$
   Eqn (2) – Eqn (3) $\Rightarrow (x_3 - u_0)(x_1 - x_2) + (y_3 - v_0)(y_1 - y_2) = 0$

```
eqn1=((Vx(1)-x).*(Vx(1)-Vz(1)))+((Vx(2)-y).*(Vy(2)-Vz(2)));
eqn2=((Vz(1)-x).*(Vx(1)-Vy(1))+((Vz(2)-y).*(Vx(2)-Vy(2))));
[A,B] = equationsToMatrix([eqn1, eqn2], [x, y])
X = linsolve(A,B)
f=sqrt(((-(Vx(1)-X(1)).*(Vy(1)-X(1))))-((Vx(2)-X(2)).*(Vy(2)-X(2))))
```

The uo=-224.292 vo=-20.1119

Solve for u_0, v_0

$$f = \sqrt{-(x_1 - u_0)(x_2 - u_0) - (y_1 - v_0)(y_2 - v_0)}$$

F=1168.21 in pixels

F(mm)=F(pixel)*36/1286≅33mm

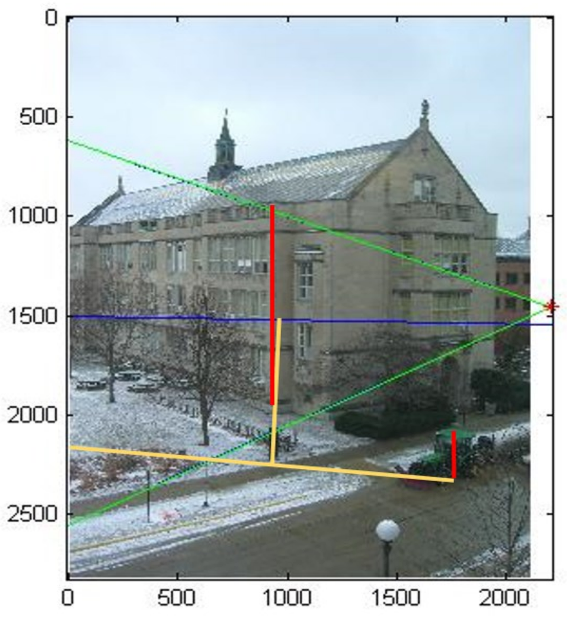
c.Extrinsic Parameter calibration:

1. The K matrix was identified using uo and vo values.
2. This was substituted in the second set of equations to find r1,r2,r3.
3. R=[r1 r2 r3]
4. Inv(R) was made equal to R^T by normalizing r1,r2,r3. The obtained R was :

```
R =

      0.81          0.56         -0.17
      0.23          0.02          0.34
      0.55          0.83          0.92
```

```
K=[f,0,u;0,f,v;0,0,1];
p1=[1481.97;465.54;1];
p2=[554.66;5.34;1];
p3=[-438.81;412.15;1];

r1=K\p1;
r2=K\p2;
r3=K\p3;

r1=r1/norm(r1);
r2=r2/norm(r2);
r3=r3/norm(r3);
```

3. Metrology:



The Method used:
1. Initially the three orthogonal vanishing points were found using the code in the first part.
2. The base points of the building, tractor are joined with that of base of sign post. These lines are shown by yellow lines. In MATLAB , the cross product of these base points are taken to find the line joining the bases. The cross product of the line joining the bases and the horizon is taken to find the intersection point. This is v.
3. The r is the top point of the building , tractor. The t is obtained by the cross product of lines(joining top and bottom of building) and line(joining v and top of sign post). Thus the values can be substituted in the formula given below where H would be 1.65. The sample code for finding height of building is given below.
4. The height of camera is found by finding the distance of sign post bottom to horizon.(perpendicular)

$$\frac{|t-b||V_z-r|}{|r-b||V_z-t|}=\frac{H}{R}$$

The tabulated result:
1. Height of building was= 15.2m
2. Height of tractor = 2m
3. The height of camera=11.6m

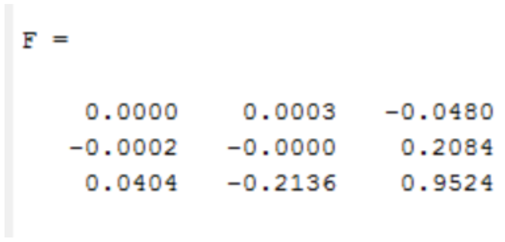
```
c1=(cross(temp,bo));
    c2=cross(vp_n_x,vp_n_y);
    v=cross(c1,c2);
    c3=cross(v,to);
    c4=cross(r,temp);
    t=cross(c3,c4);
    A=[bo';to'];
    B=[temp';r'];
    C=[t';temp'];
    D=[r';temp'];
    E=[vp_n_z';r'];
    F=[vp_n_z';t'];
    d1 = pdist(C,'euclidean');
    d2 = pdist(D,'euclidean');
    d3 = pdist(E,'euclidean');
    d4 = pdist(F,'euclidean');

    fac=((d1*d3)/(d2*d4))
    H=1.65*fac
```

2. Epipolar Geometry

Method:
1.Normalize the points . Randomly select 8 points. Compute the fundamental matrix.
2. The inliers is decided by calculating the distance of the points in the first image and the epi-polar lines of the second and the distance between the points in the second image and epi-polar lines in first image. If this distance is less than 2 it is classified as inlier.

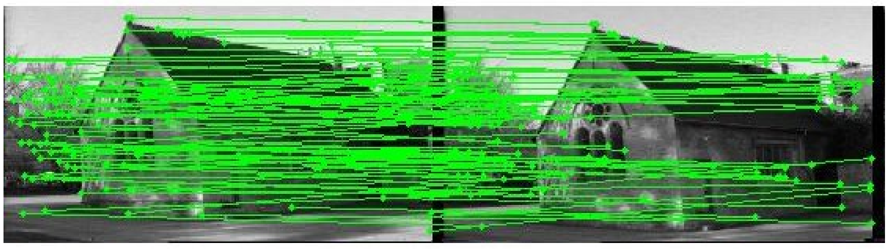
3. The RANSAC is performed 30 times computing the fundamental matrix. The fundamental matrix with maximum number of inliers is returned.

```
for i = 1:30

    ind=randsample(size(matches,1),8)
    idx = matches(ind,:);
    F{i} = fund_comp(idx,c1,r1,c2,r2);
    x1 = c1(matches(:,1));
    y1 = r1(matches(:,1));
    x2 = c2(matches(:,2));
    y2 = r2(matches(:,2));

    p2 = F{i}*[x1';y1';ones(1,size(x1,1))];
    norm2 = sqrt(p2(1,:).^2 + p2(2,:).^2);
    p1 = F{i}'*[x2';y2';ones(1,size(x2,1))];
    norm1 = sqrt(p1(1,:).^2 + p1(2,:).^2);
    %finding inliers by checking the distance of points in first image to
    %epipolar line in second image. Similarly the distance of keypoints in
    %second image and epi-polar
    dist1 = abs(sum([x1';y1';ones(1,size(x1,1))].*p1))./norm1;
    dist2 = abs(sum([x2';y2';ones(1,size(x2,1))].*p2))./norm2;
    inliers = (dist1<2 & dist2<2);

    if (sum(inliers)>no_inliers)
        inliers_most=inliers;
        outliers_less=~inliers;
        no_inliers = sum(inliers);
        F_new=F{i};
    end

end

F=F_new;
inliers=inliers_most;
outliers=outliers_less;
```

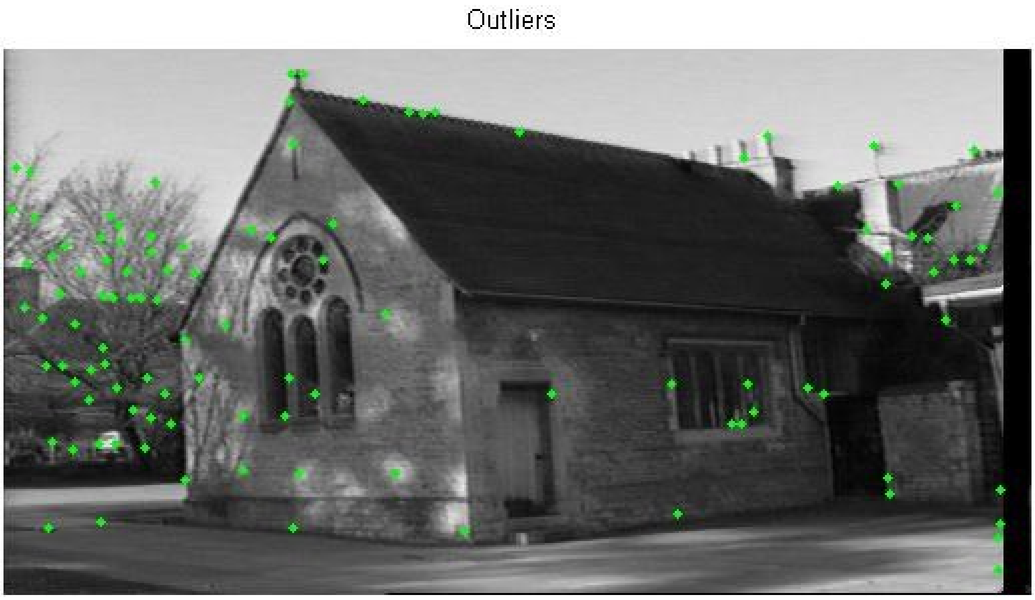


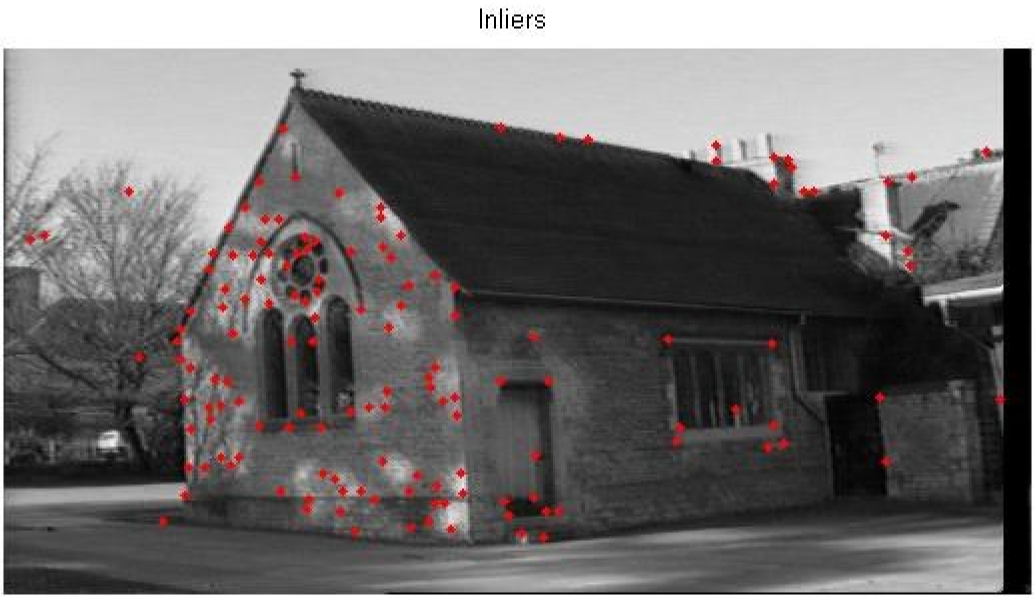
```
F =

    0.0000    0.0003   -0.0480
   -0.0002   -0.0000    0.2084
    0.0404   -0.2136    0.9524
```

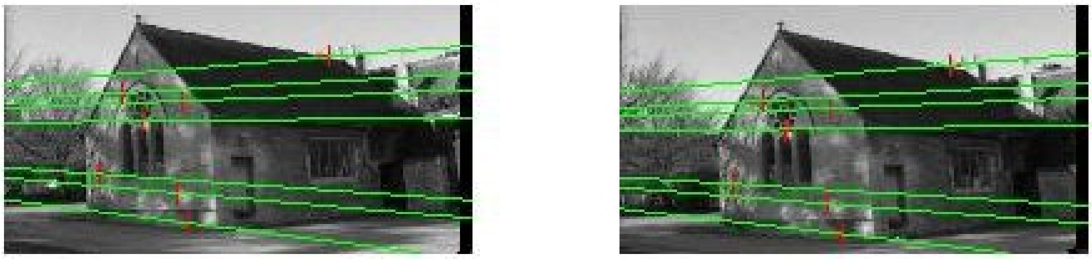
Estimated Fundamental Matrix



The Matches






The Plot of Epi-Polar Lines

3. Affine Structure from Motion

Method:

1. The tracked X and Y key points are taken for all frames. Those keypoints which are present in all frames alone are considered.
2. The singular value decomposition of the combined x, y points is done.

```
[U D V] = svd((W));
U3=U(:,1:3);
V3=V(:,1:3);
V3_n=V3';
D3=D(1:3,1:3);
A=U3*sqrt(D3);
S=sqrt(D3)*V3_n;
```

'A' gives the affine motion matrix. 'S' gives the shape matrix.

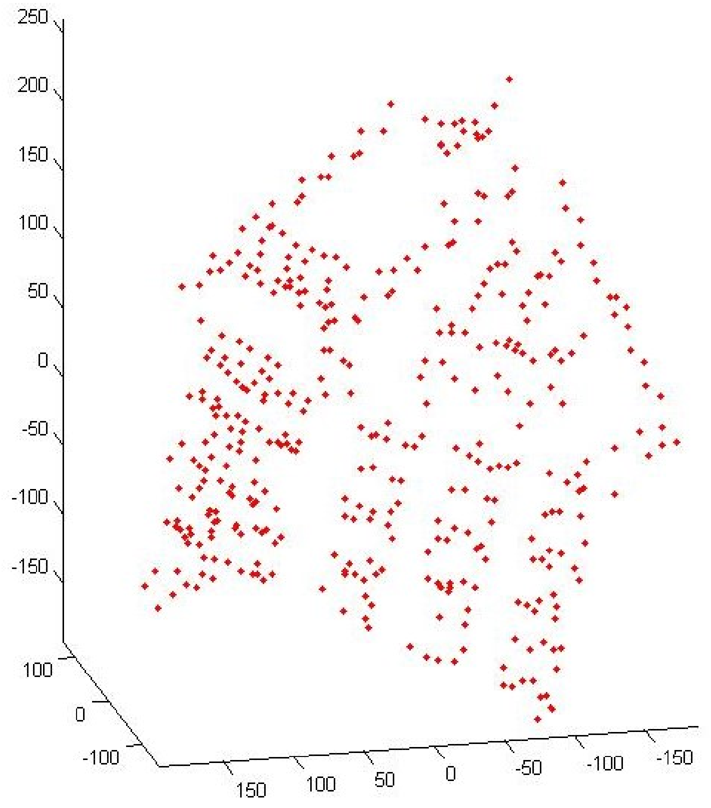
3. The I matrix is found by the least square solution between the metric matrix and a matrix containing a column of zeros and ones. This is done to solve the following equation.

$$\tilde{\mathbf{a}}_{i1}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i1} = 1$$
$$\tilde{\mathbf{a}}_{i2}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i2} = 1$$
$$\tilde{\mathbf{a}}_{i1}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i2} = 0$$

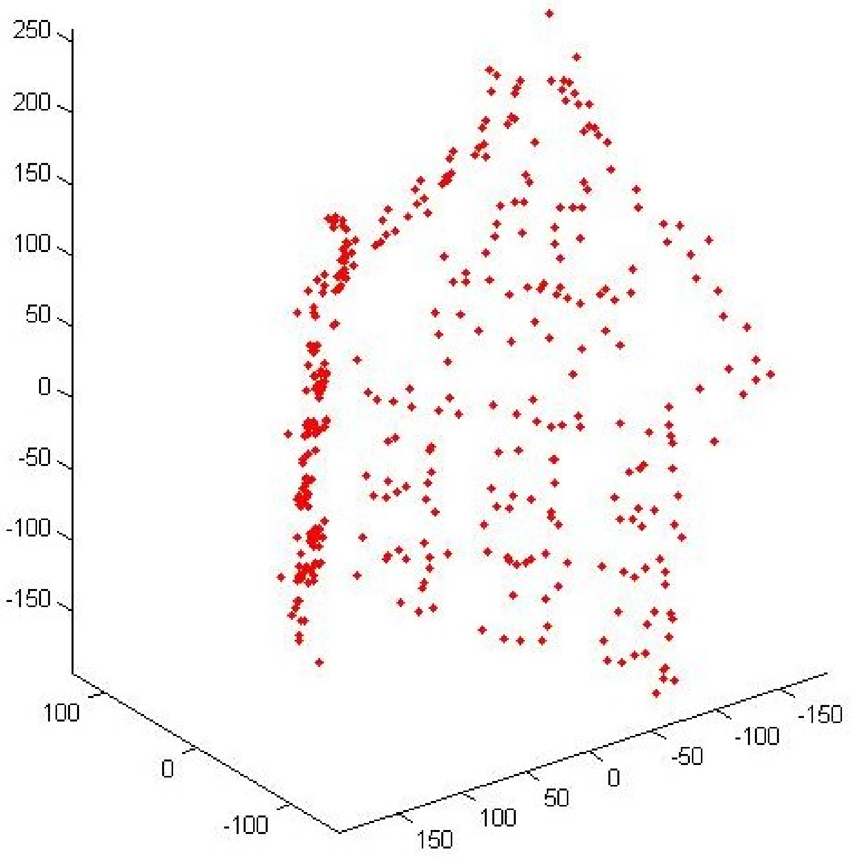
4. The L matrix is comoposed from I .The L matrix is made positive definite by using spdMat function. The cholskey decomposition of L gives C. The A and C are updated by the equation :

```
A=A*C;
S=C\S;
```

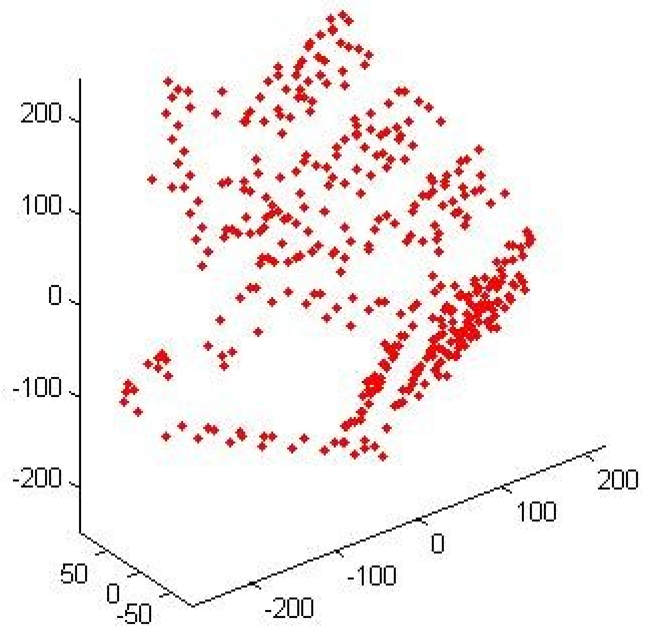
RESULTS:



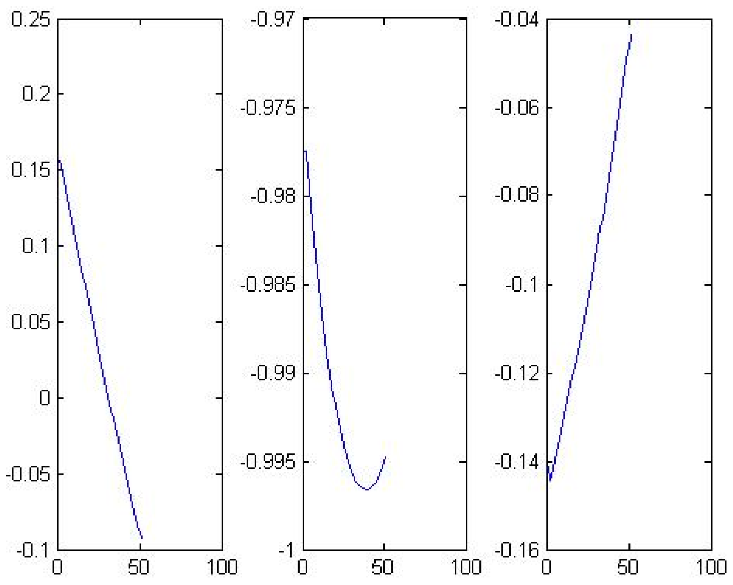
The 3D plot of the projected 3D positions of the keypoints over 50 frames



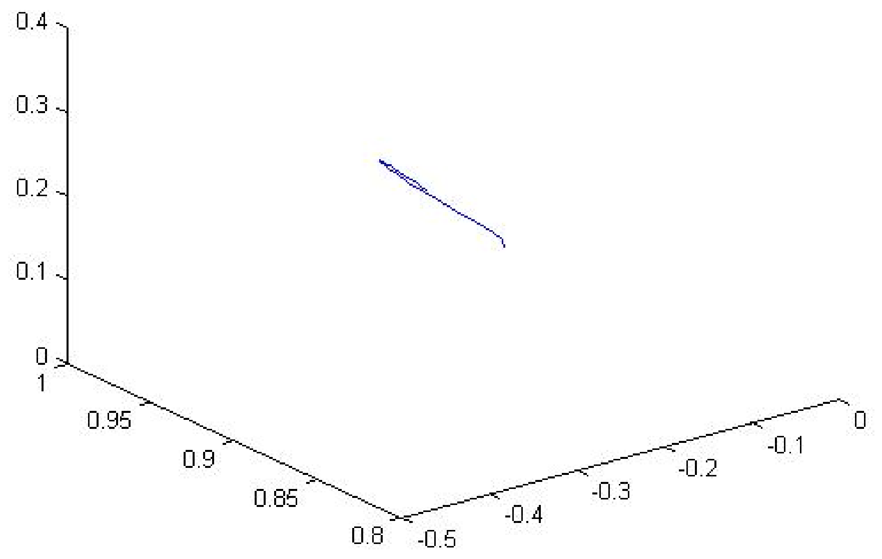
Another view of the 3D points



Third view of the 3D points



The camera position plot in the xy,yz,zx axes



The 3D-Plot of recovered camera positions

EXTRAS

1. The Automatic Vanishing point detection:

I had implemented the method to get the prefMat However I did not get results, as the JK clustering assigned all the lines to the same vanishing point instead of grouping them into three categories.

The Method used:

a) The lines obtained were RANSAC ed for two random lines. The point of intersection was found.
b) The angle between this point and the mid point of 100 lines are found. If the angle is less than 30, the corresponding line element for that vanishing point hypothesis in the prefMat is set 1.
   Thus the dimension of prefMat is 200*100.

```
    for i=1:200

        p1=rand_vp(lines); %This function returns the intersection point
        for j=1:100

            mp=[(line(j,1)+line(j,3))/2;(line(j,2)+line(j,4))/2;1];
            ep=[line(j,1);line(j,2);1];
            l1=cross(mp,ep);

            vp=p1';
            l2=cross(vp,mp);
            v1=[l1(2),-l1(1)];
            v2=[l2(2),-l2(1)];
            angle=rad2deg(acos( dot(v1,v2)/(norm(v1)*norm(v2)) ));
            if(angle>(180-angle))
                angle=(180-angle);
            end

            if (angle<2)
                ang=angle
                PrefMat(j,i)=1;
            else
                PrefMat(j,i)=0;
            end
        end
    %     if(sum(sum(PrefMat))>sum_in)
    %         PrefMatf=PrefMat;
        end
    %     PrefMat=logical(PrefMat)
    %     size(PrefMat)

    PrefMat=logical(PrefMat);
```

2.The normalization was applied to the points and finally denormalized as a part of the normal problem.

```
 %DENORMALISATION

    T1 = [nd1 0 0;0 nd1 0;0 0 1]*[1 0 -mean(x1);0 1 -mean(y1);0 0 1];
    T2 = [nd2 0 0;0 nd2 0;0 0 1]*[1 0 -mean(x2);0 1 -mean(y2);0 0 1];
    [U1,S1,V1] = svd(F);
    S1(3,3) = 0;
    F = U1*S1*V1';
    F = T2'*F*T1;
    F = F*(sqrt(1/sum(sum(F.^2))));
 %the x and y coord were    centered   by subtracting mean. These coordinates
were then divided by their norm - NORMALIZATION


x_cent = [(x1-mean(x1)) (y1-mean(y1))];
    y_cent = [(x2-mean(x2)) (y2-mean(y2))];
    nd1 = 1/(std(sqrt(sum(x_cent.^2,2))))*sqrt(2);
    nd2 = 1/(std(sqrt(sum(y_cent.^2,2))))*sqrt(2);
    x_cent = x_cent*nd1;
    x1_norm = x_cent(:,1);
    y1_norm = x_cent(:,2);
    y_cent = y_cent*nd2;
    x2_norm = y_cent(:,1);
    y2_norm = y_cent(:,2);
```

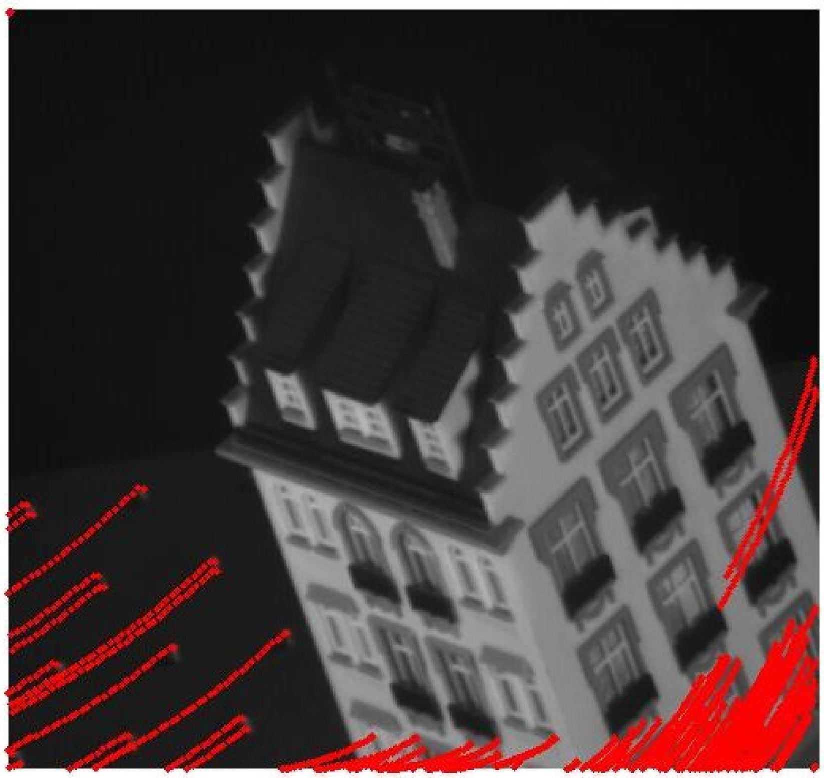
3.The Prediction of Disappearing Points:

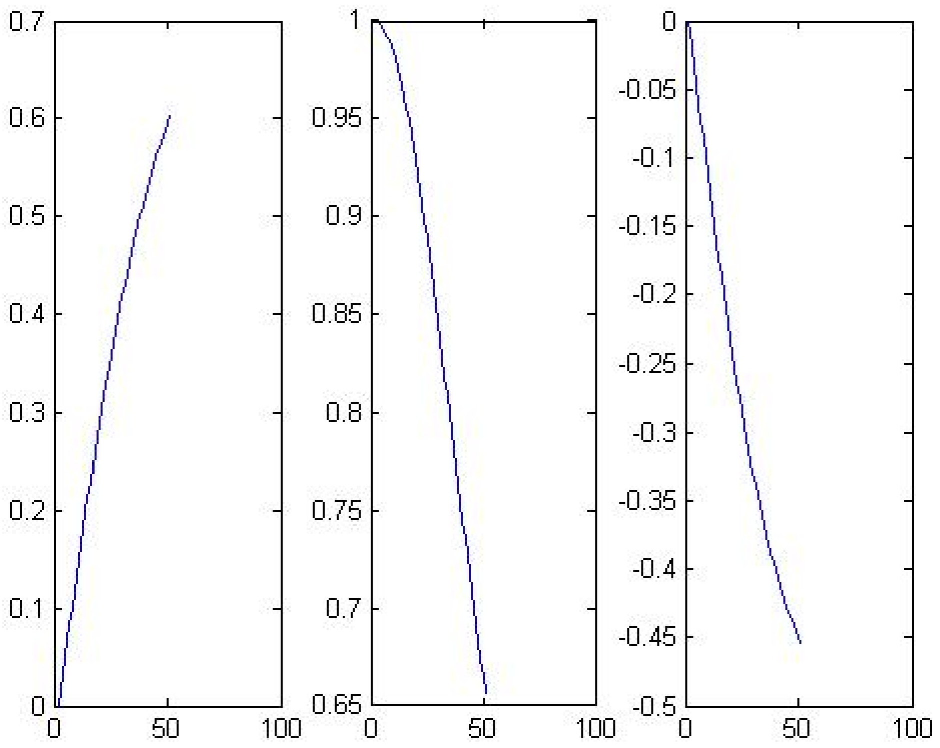
Method Used:

1. The Affine matrix was found for the keypoints that were available in all the frames.
2. Every keypoint (Kp)that went missing in some of the frames were considered. The frames that had the keypoints for each of them in Kp is taken as frames. The affine matrix corresponding to these frames are chosen and constructed as Anew.
3. Ptsnew comprises of the x and y coordiates of the points available in those ‘frames’. Thus the 2D position of the point belonging to Kp is found as Dneeded. This Dneeded has the point positions corresponding to all frames. This procedure is repeated for each point in kp and thus the Final Dneeded gives the x and y coodinates of all points in KP.
4. This Dneeded is used as the new track_x and track_y points and the procedure used in the third part is repeated to find the 3D position and camera position path for the disappeared points.

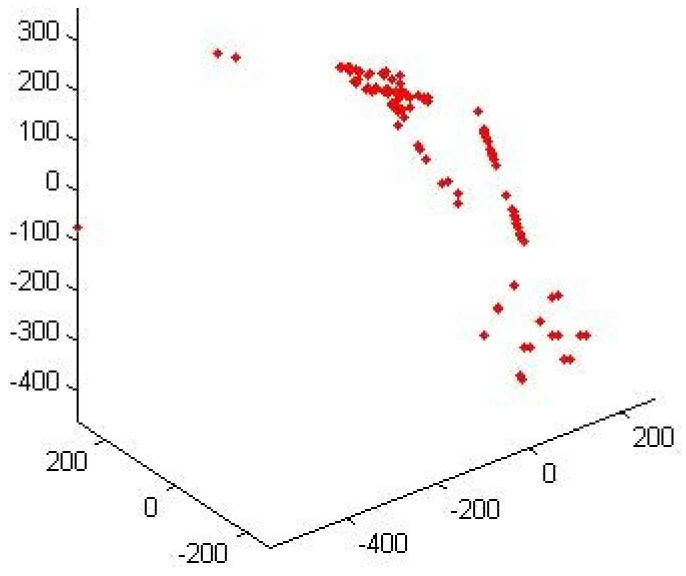
```
    [~,mis_pts]=find(isnan(Xorig)); % finding those points that had isNaN
    a=unique(mis_pts);
    tch=size(a,1)
    Dneeded=[];
for k=1:100
        Anew=[];
        ptsnew=[];
        Snew=[];

        [frames,~]=find(~isnan(Xorig(:,a(k)))); %Finding those frames that
had isNan corresponding to a point
        Anew=zeros(size(frames,1),3);
        req=size(frames,1);
        for j=1:req
            ptsnew(j,1)=Xorig(frames(j),a(k))
            ptsnew(j+req,1)=Yorig(frames(j),a(k))
        end
        for i=1:req
            Anew(i,1:3)=A(frames(i),:);
            Anew(i+req,1:3)=A(frames(i)+F,:);
        end
        tn1=size(Anew)
        tn2=size(ptsnew)
        Snew=Anew\ptsnew;
        Dneeded(:,k)=A*Snew; %Dneeded computed by multiplying original affine
matrix and new shape vector.
    end
```

RESULTS:
The predicted 3D plot of the keypoints in all frames and the camera positions is given in the following.

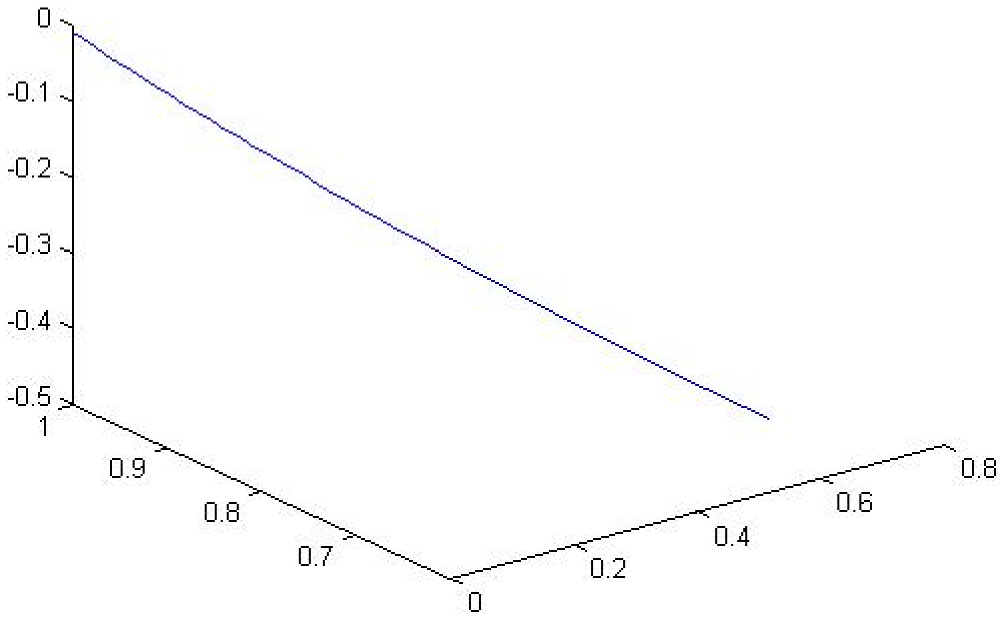

The disappeared point plots for different frames on 2D



Plot of 3D Camera positions for all the frames of those keypoints that went missing in some of the frames



The predicted 3D position plot of the disappeared keypoints



3D camera position plot