Seylom AYIVI **HW3**
CS 543 – Computer Vision

1 – Single-View Metrology.

We compute all possible vanishing points obtained by first intersecting all pairs of parallel lines. The candidates vanishing points obtained for these lines are then sorted increasingly by the sum of the angle between a specific line and the segment that join the vanishing point to the line center.

- We use the cross product or two lines to determine their intersection if there exists one.
- We choose the vanishing point that results in intersecting lines with the least angle deviation
- We divide the resulting point by its third coordinates in order to obtain our vanishing points in homogenous coordinates.

VP1: (-464.00, 1572.00)

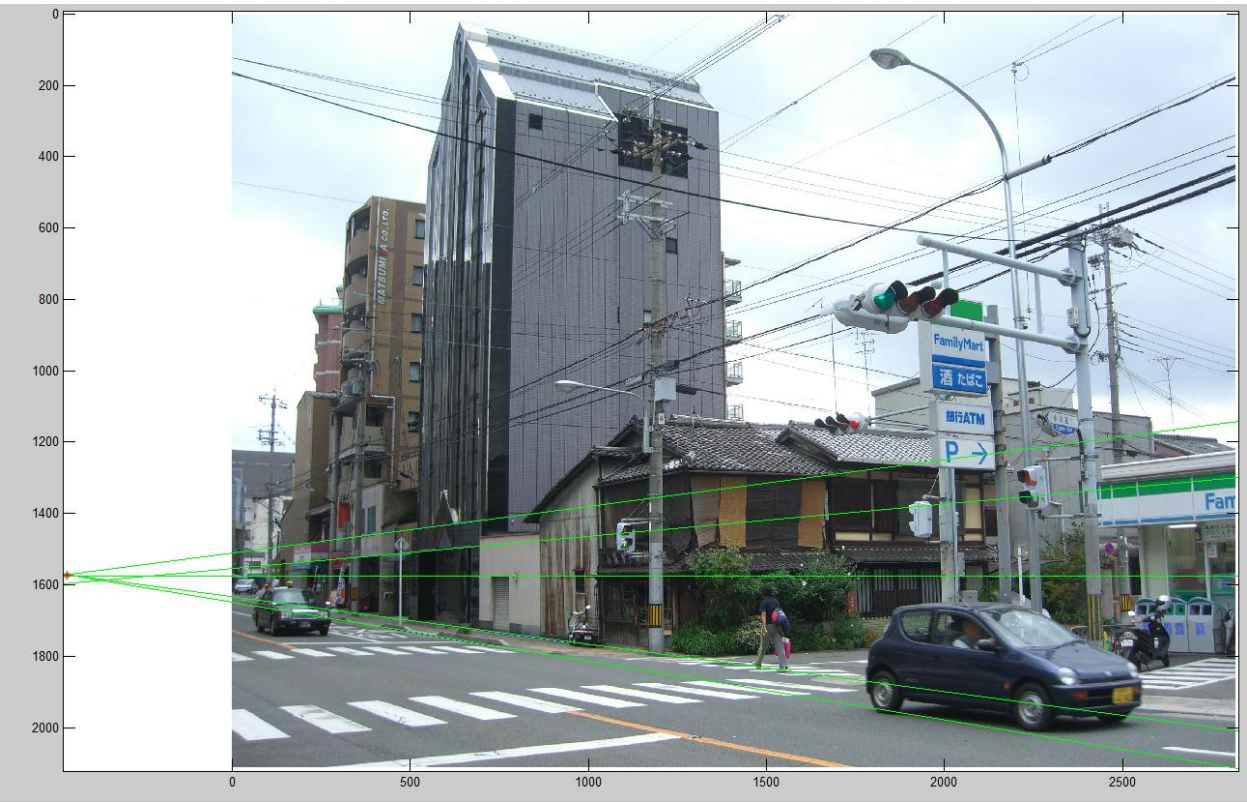


Figure 1 First vanishing point 1 estimation with 5 lines

VP2: (1058.00, -13457.00)

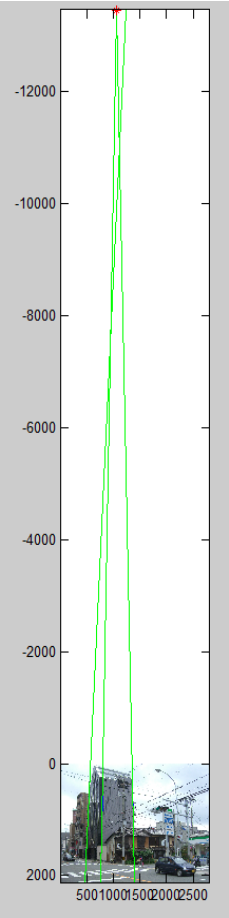


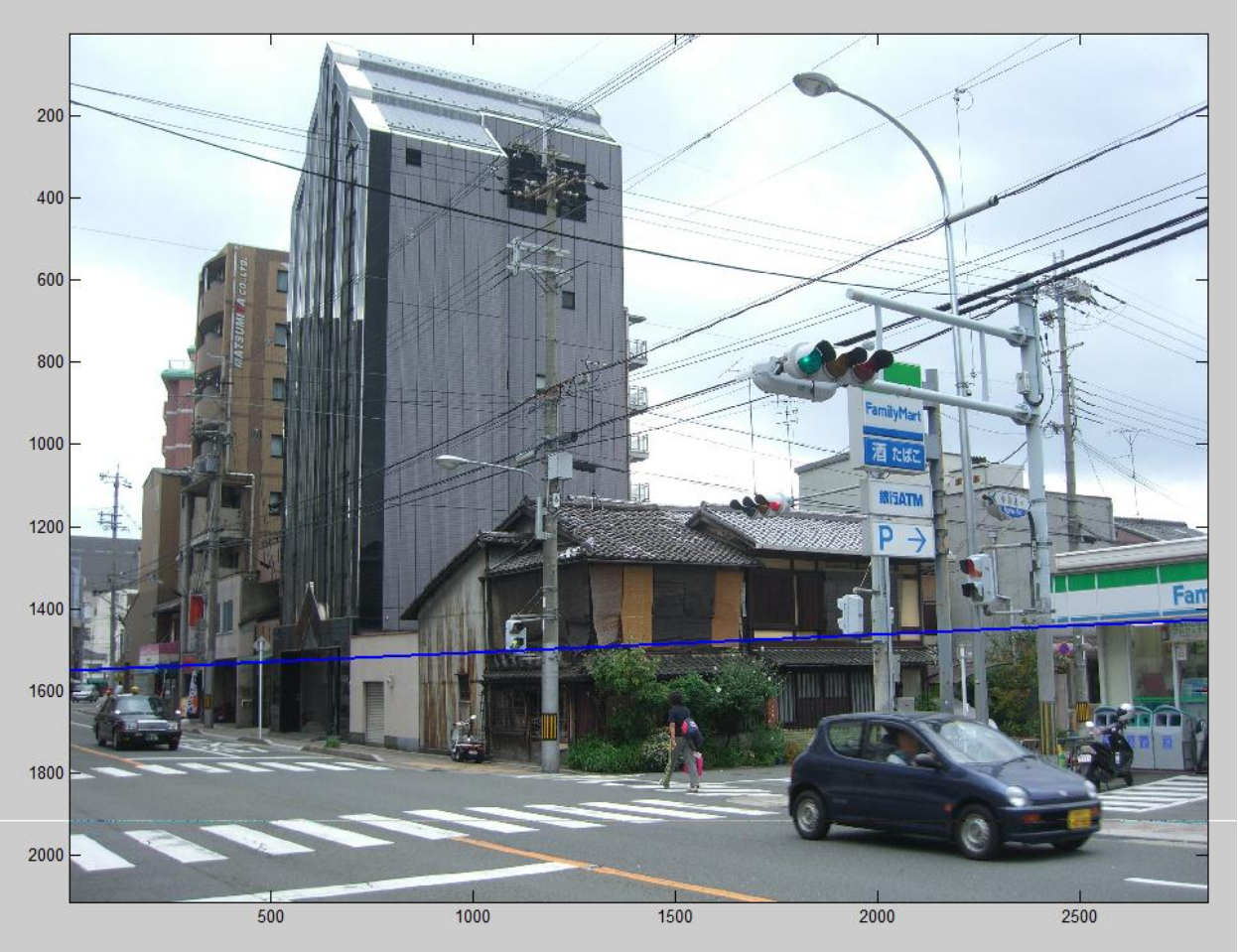
Figure 2: Vanishing point 2 estimation with 3 lines

VP3: (5942.8, 1286.9)

Figure 3 vanishing point 3 estimation with 6 lines

The ground horizon is obtained by computing the cross product of VP1 and VP3

VP1: (-464.00, 1572.00, 1)
VP3: (5942.8, 1286.9, 1)



We obtain a1 = 300, b1 = 6400 and c1 = -9.9392 * 1.0e+006
By dividing these parameters by **sqrt (a1^2 +a2^2)** we obtain a =~ 0 (??), b = 1, c = -1551.3

B –

If **K** is the camera matrix we can establish a relation between the projection of our vanishing point in the plane and their 3D coordinates. We have the relation

$$VP = K * R * \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \qquad (1)$$

Where R is a rotation matrix that maps the real world coordinates to the 2d projection coordinates and $K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}$

(The translation component has been removed since it does not impact our vanishing points).

$$K^{-1} = \begin{bmatrix} 1/f & 0 & -U_0/f \\ 0 & 1/f & -V_0/f \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Since all pairs of our vanishing point are perpendicular in the 3D world we can choose them to be image projection of perpendicular vectors [1 0 0] [0 1 0], [0 0 1].
Replacing these particular vectors in (1) yields:

$$(K^{-1} * VP_i)^t * (K^{-1} * VP_j) = 0 \qquad \{i, j \in \{1,2,3\} \; and \; i \neq j\} \quad (3)$$

Let's $VP_i = \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$

Replacing (1) and (2) in (3) yields:

$$\frac{(x_i - u_0)(x_j - u_0)}{f^2} + \frac{(y_i - u_0)(y_j - u_0)}{f^2} + 1 = 0$$

$$\begin{aligned} (x_1 - u_0)(x_2 - u_0) + (y_1 - u_0)(y_2 - u_0) + f^2 &= 0 \\ (x_2 - u_0)(x_3 - u_0) + (y_2 - u_0)(y_3 - u_0) + f^2 &= 0 \\ (x_1 - u_0)(x_3 - u_0) + (y_1 - u_0)(y_3 - u_0) + f^2 &= 0 \end{aligned}$$

Expanding these equations and replacing f^2 yields:

$$\begin{aligned} (x_3 - x_2) * u_0 + x_1 * (x_3 - x_2) + (y_3 - y_2) * v_0 - y_1 * (y_3 - y_2) &= 0 \\ (x_3 - x_1) * u_0 + x_2 * (x_3 - x_1) + (y_3 - y_1) * v_0 - y_2 * (y_3 - y_1) &= 0 \end{aligned}$$

Solving the system yields

$$u_0 = 1702, v_0 = 854$$

P = (1702, 854) is our Optical Center.

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

f = 2941.8

C –

We can use the relation P = K[R |t] X where K is the camera matrix we just computed. P is the projection of X on the plane and is expressed in homogenous coordinates like X. We can then use our vanishing point to compute the matrix.
Since translation doesn't impact our vanishing points we can remove the translation component and keep P = KRX (X is no longer expressed in homogenous coordinates but P is)

For each of our vanishing points we have the relation

$$w * \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = K * \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} * \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

For our three vanishing point we can use orthogonal unit vectors [1 0 0],[0 1 0] and [0 0 1] to compute these parameters.
We obtain:

$$\begin{pmatrix} w.x_i \\ w.y_i \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} r_{11} \\ r_{21} \\ r_{31} \end{pmatrix} \quad (1)$$

$$w * x_1 = f * r_{11} + u_0 * r_{31}$$
$$w * y_1 = f * r_{21} + v_0 * r_{31}$$
$$w = r_{31}$$

These previous equations yield

$$\frac{r_{11}}{r_{31}} = \frac{(x_1 - u_0)}{f} \quad (2)$$
$$\frac{r_{21}}{r_{31}} = \frac{(y_1 - v_0)}{f} \quad (3)$$

We can use (2) and (3) with the relation

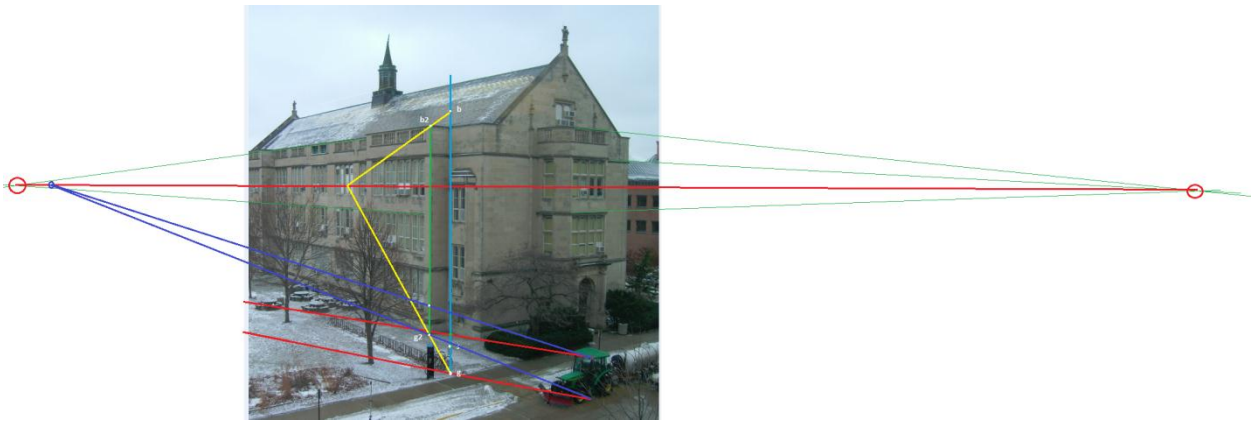
$$r_{11}{}^2 + r_{21}{}^2 + r_{31}{}^2 = 1 \quad \text{or} \quad r_{11}{}^2/r_{31}{}^2 + r_{21}{}^2/r_{31}{}^2 + 1 = 1/r_{31}{}^2 \quad (4)$$

Replacing (2) and (3) in (4) yields $\boldsymbol{r_{11} = 0.82, r_{21} = 0.08, r_{31} = 0.57}$

Using a similar approach with the two remaining points we obtain
$\boldsymbol{r_{12} = -0.04, r_{22} = -0.98, r_{32} = 0.20}$ And
$\boldsymbol{r_{13} = -0.57, r_{23} = 0.24, r_{33} = 0.78}$

$$R = \begin{pmatrix} 0.82 & -0.04 & -0.57 \\ 0.08 & -0.98 & 0.24 \\ 0.57 & 0.20 & 0.78 \end{pmatrix}$$

D-

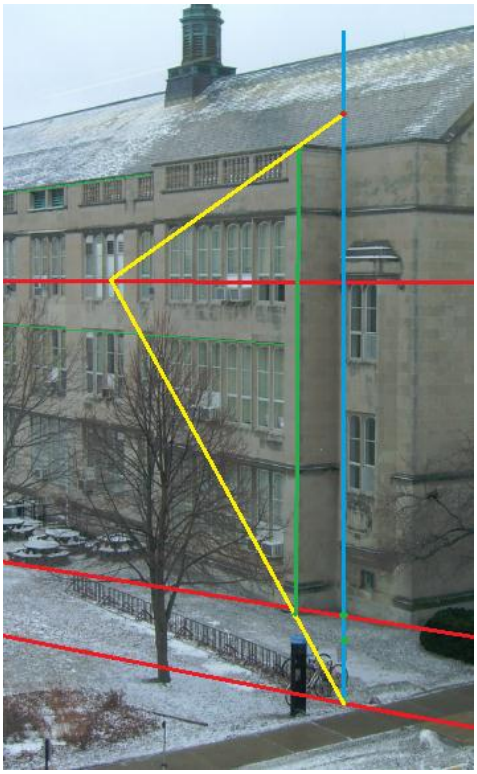


A larger resolution is available with my submission, the file is called building.png

Let's denote by C, B, S, T the height of the camera, the building, the sign and the tractor (respectively), c, b, s and t the corresponding points on the image and g the ground at the base of the sign. Using vanishing points and cross ratio, we have the following relations:
For the camera's height we need to estimate the distance between the base of the sign and the horizon. We have the relation:

$$\frac{C}{S} = \frac{(\mathrm{g} - \mathrm{c})(\mathrm{Vz} - \mathrm{s})}{(\mathrm{g} - \mathrm{s})(\mathrm{Vz} - \mathrm{c})}$$

Where C denotes the height of the camera and S the height of the sign.



The expression simplifies to

$$\frac{C}{S} = \frac{\mathrm{g} - \mathrm{c}}{\mathrm{g} - \mathrm{s}}$$

The height of the sign is about 44 pixels, and the height of the horizon from the bottom of the sign is 319 pixels, S= 1.65m

➔**Camera Height = 11.96 m**

Similarily for the Height of the building we have

$$\frac{B}{S} = \frac{\mathrm{g} - \mathrm{b}}{\mathrm{g} - \mathrm{s}}$$

The building height is about 446 pixels

Using the previous relation yields:

➔**Building height** = 446/44*1.65 = **16.72 m**

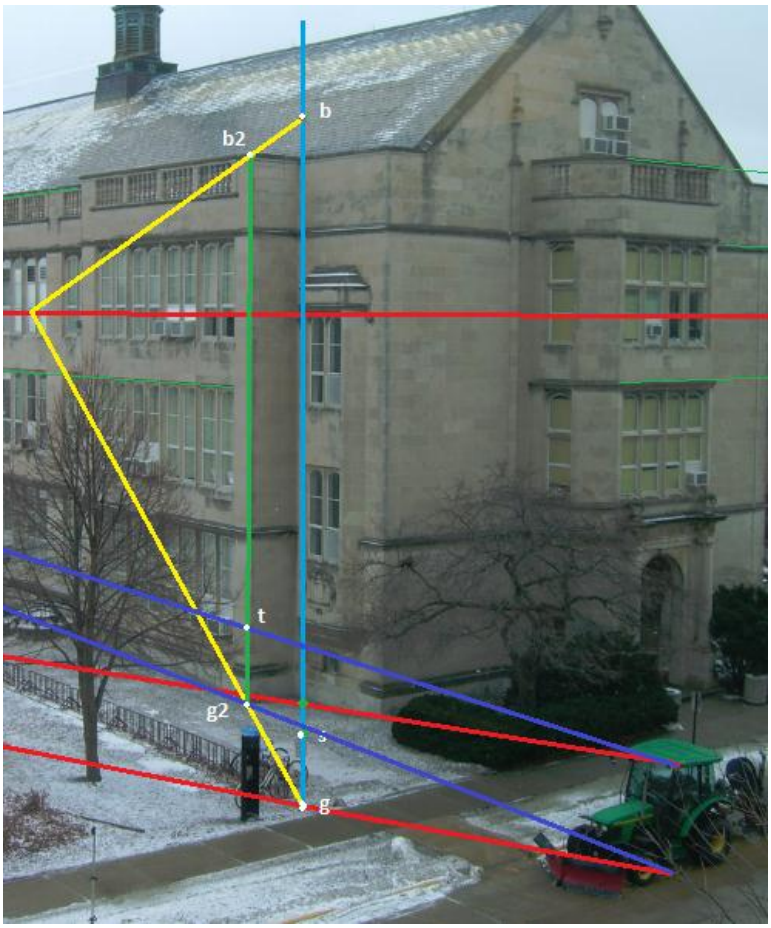


Figure 4 Using blue lines to estimate the height of the tractor

The height of the tractor is estimated using the height of the building (blue lines)

$$\frac{T}{B} = \frac{\mathrm{g2} - \mathrm{t}}{\mathrm{g2} - \mathrm{b2}}$$

The height of the building in pixels is 356 and the height of the tractor in pixels is 50
➔**Tractor height = 2.34 m**

2 – Mission Possible

a) We cannot recover the depth from a camera that only translates. Using two pictures could help us recover the homography H between them but not the depth information.

b) The guard can recover depth with only one eye, or it will be very difficult. If he can move though we could keep track of his distance to the screen to keep him away from it.

c) If should not be possible to fool them both at the same time since they have two different viewpoint. We may need a second projector system to create two separate images for camera and guard.

3 - Epipolar Geometry

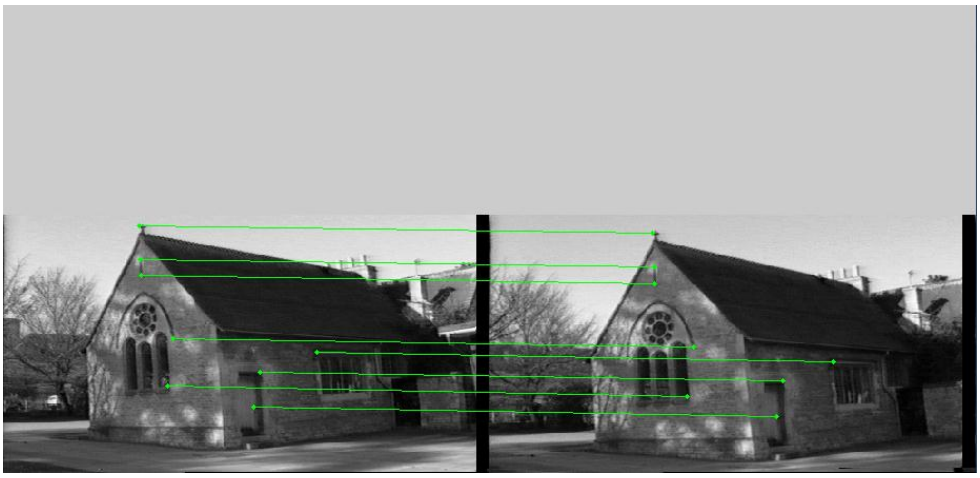


Figure 5 Set of 8 good matches used to compute F at first

a) Manual point selection:
I chose the above 8 points to solve for the Fundamental matrix. I used the normalized 8-point algorithm to estimate it. Below are the steps followed:

- Retrieve the matches coordinates from the data file
- Compute the centroids of both set of points and shift the points in our dataset by the centroid values
- Normalize the data. I followed the definition provided by the Wikipedia article on the normalized 8 point algorithm to compute the normalized points : mean distance from the origin to a point is equal to $\sqrt{2}$
- Express the transformation matrix mapping the original coordinates to the normalized ones. T1 and T2 will represent the results, mapping respectively the coordinates of image1 and image2 to their normalized counterparts.
- Compute the matrix F as defined in class using SVD and enforcing $\det(F) = 0$
- Compute F using the relation F = T2'*F*T1.

With the 8 selected points I obtained:

$$F = \begin{pmatrix} 0 & 0 & 0.0026 \\ 0 & 0 & 0.0072 \\ 0.0032 & -0.0085 & 0.0538 \end{pmatrix}$$

We can use any of the selected matching point to compute the epipolar lines in the other image using the relation $l = F * p$ where p is a matched point in the first image expressed in homogenous coordinate.

Using the relation $ax + by + c = 0$ and $l = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$

We have $y = (-c - ax)/b$

We can then plot the epipolar lines by choosing $x = (1 : n)$ and plot(x, y, …) where n is the number of column in the image.

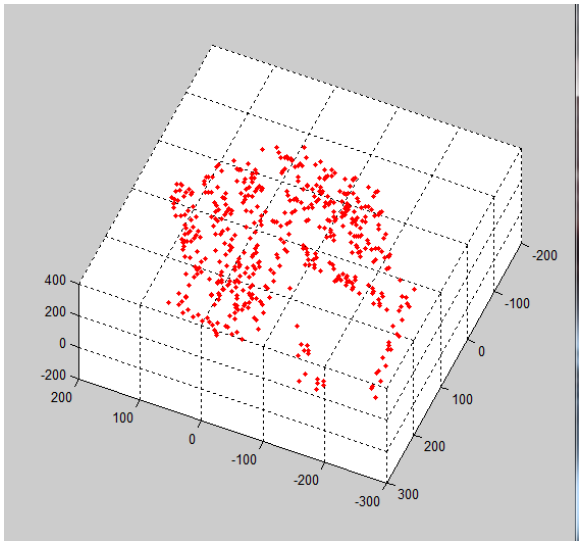
I was unable to obtain satisfactory results with the plotting. I couldn't obtain acceptable results with the obtained fundamental matrix. After normalizing the matrix to have all column vectors add up to 1, the position of the tracked feature in the second image does not match the computed value.

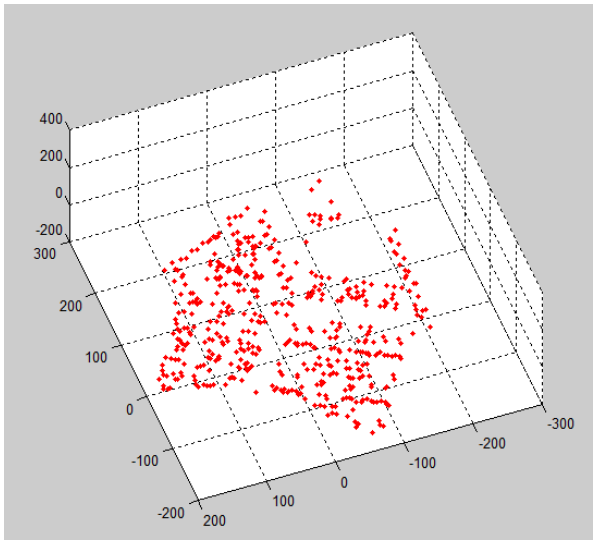
Ransac solution:

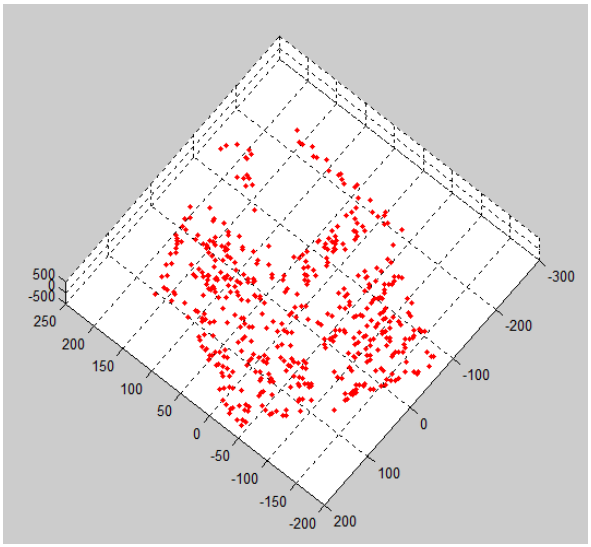
My initial idea was to use the reprojection error as a measure of the error (a threshold)
I couldn't figure out why the previous matrix listed above is unable to
I was unable to make my ransac algorithm work, but only after spending some time wondering what might have been wrong with my previous fundamental matrix. I made available the code in **runthis_ransac.m**. This function in turn calls another function **computeransac** to evaluate the fundamental matrix.

4 - Affine Structure form Motion.

I included a file called *runthis.m* with my submission that makes use of the tracked points provided by the homework material. I initially implemented the algorithm using my own data but when trying to recover the camera using the Cholesky decomposition, Matlab produces an error. After receiving the grade and report from HW2 I realize that there may have been a slight convergence issue with some of the points I tracked.







The 3D point cloud is computed using the algorithm described in class

- I am using the provided tracked points in the supplemental material
- we create a 2m*n matrix named D in my code
- We factorize our matrix D
- Compute the motion and shape matrices
- Eliminate affine ambiguity using the orthographic constraints and Cholesky decomposition to recover C

2- Camera path

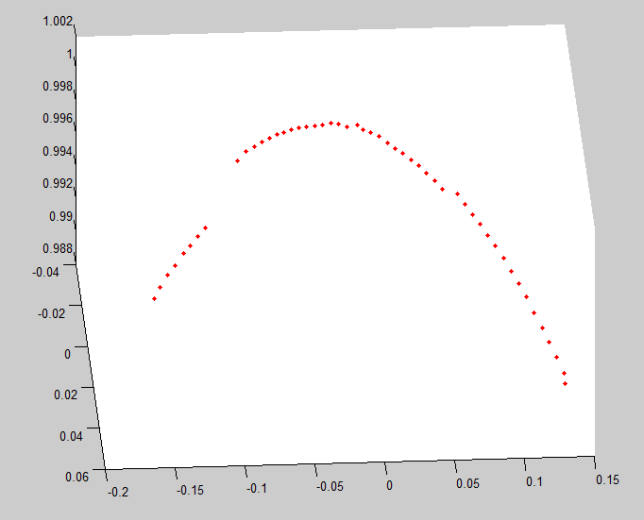


Figure 6 Camera path in 3D

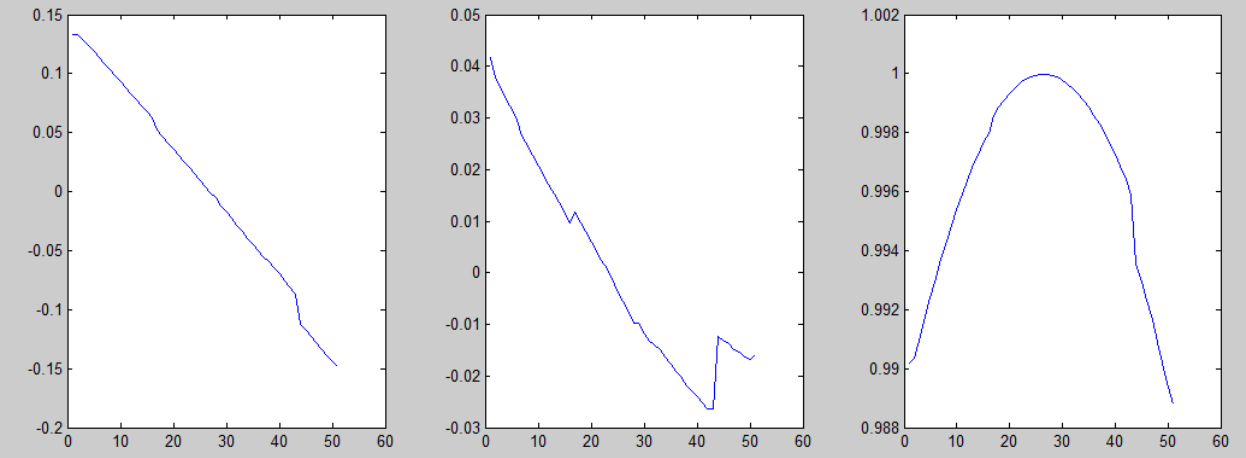


Figure 7 camera path along all 3 directions

For each image we extract from the D matrix i_f and j_f (even and odd rows) and compute their cross product. We can then normalize the result (by dividing the obtained vector by its norm) and store it as the camera components for the frame we processed.