CS 543

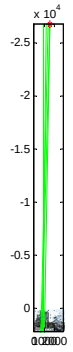
Homework 3

Ning Xu

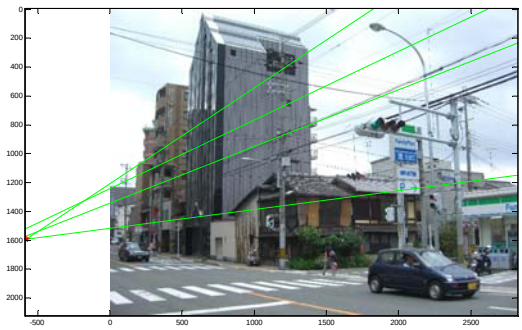
1. Single-View Metrology

A. VPs and lines used to estimate them

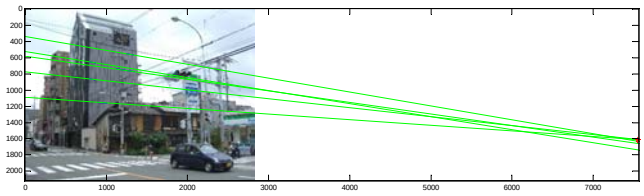
I use the criteria that VP minimizes the angles between itself and center points of all the lines to choose VP.



Vp1 = [377543.94122; -6550331.0717; 243.94552912];



Vp2 = [2712475.1094; -7489637.248; -4709.2093448];



Vp3 = [-2562895.779; -547588.75174; -339.4024753];

Horizon line can be computed by the cross product of Vp2 and Vp3, which is [-0.0028259407; 0.99999601; -1592.0449]. Therefore a = -0.0028259407, b = 0.99999601, c = -1592.0449.

B. Focal length and optical center

Since the Intrinsic matrix K = $\begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix}$, and for vanishing points, $X_i^T X_j = 0$. We can use the three finite vanishing points to solve for the focal length f and optical center (u_0,v_0).

$$X_i^T X_j = vp_i^T K^{-T} K^{-1} vp_j = 0 \qquad \text{i}\neq\text{j} \tag{1}$$

$$K^{-T}K^{-1} = \begin{pmatrix} 1/f^2 & 0 & -u_0/f^2 \\ 0 & 1/f^2 & -v_0/f^2 \\ -u_0/f^2 & -v_0/f^2 & \frac{u_0^2}{f^2} + \frac{v_0^2}{f^2} + 1 \end{pmatrix} \tag{2}$$

Using MATLAB “solve” function, we can get the focal length f equals to 3498.5996, u_0 equals to 1468.4995 and v_0 equals to 1159.2203.

C. Rotation Matrix

Since $w\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = KR\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$, it’s easy to conclude that three orthogonal VPs can provide columns of Rotation matrix. Besides, by setting the unit length of columns of R, we can solve the ambiguity of w.

Therefore, using the three orthogonal VPs, we have

$$w1 * vp_3 = KR\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = Kr_1$$

$$w2 * vp_1 = KR\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = Kr_2$$

$$w3 * vp_2 = KR \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = Kr_3$$

where r_1, r_2 and r_3 are the columns of Rotation matrix. In fact, according to the right hand rule, the leftmost VP direction should be –Z direction, which meas r_3 should be $-r_3$. But I didn't modify it in the computation.

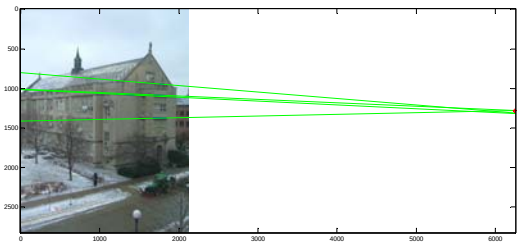
Since we already have the Intrinsic matrix K, it's easy to get the Rotation Matrix such that R = [r_1 r_2 r_3].

$$R = \begin{pmatrix} -0.8650 & 0.0028 & 0.5017 \\ -0.0646 & -0.9923 & -0.1058 \\ -0.4975 & 0.1239 & -0.8585 \end{pmatrix}$$

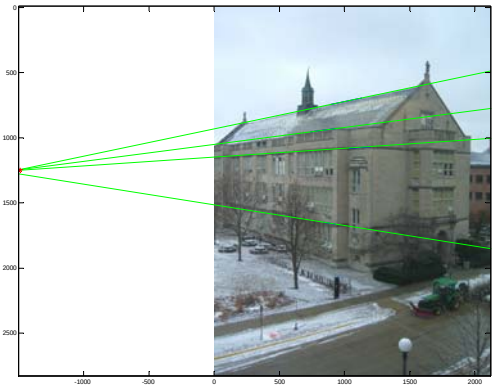
It's easy to get that $R*R^T = I$ by MATLAB.

D. Height estimation

First, we estimate the horizon line,

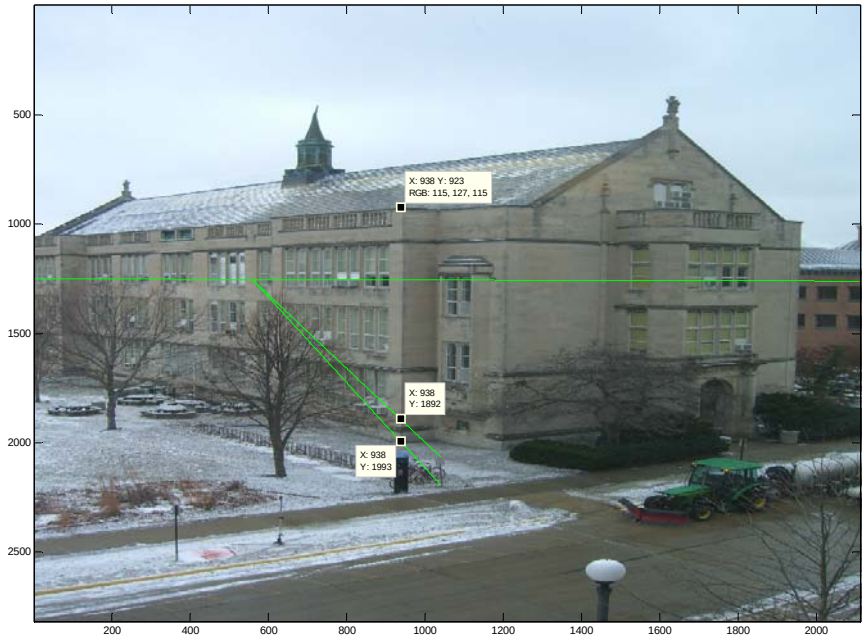


vp1 = 1.0e+007 * [1.2799; 0.2623; 0.0002];

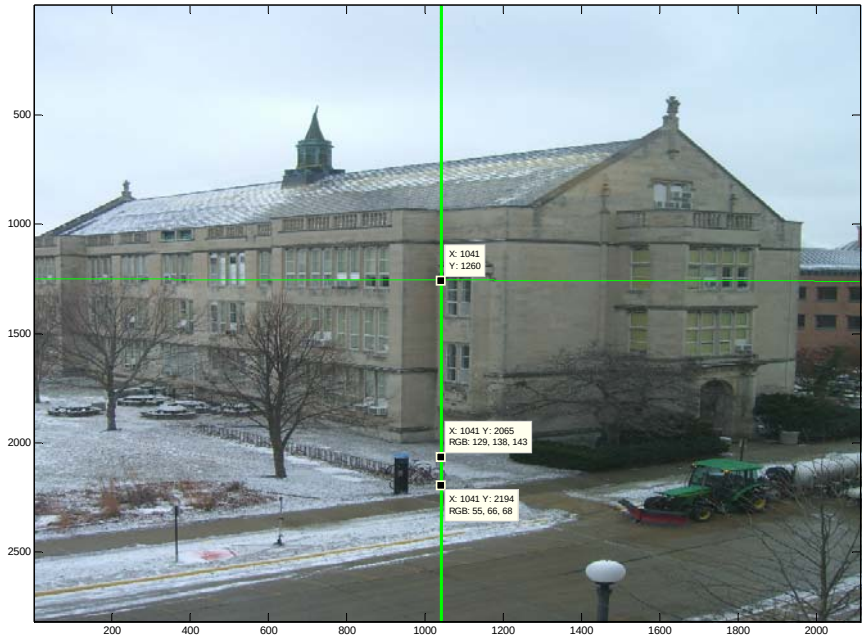


vp2 = 1.0e+006 * [-5.6778; 4.7466; 0.0038];

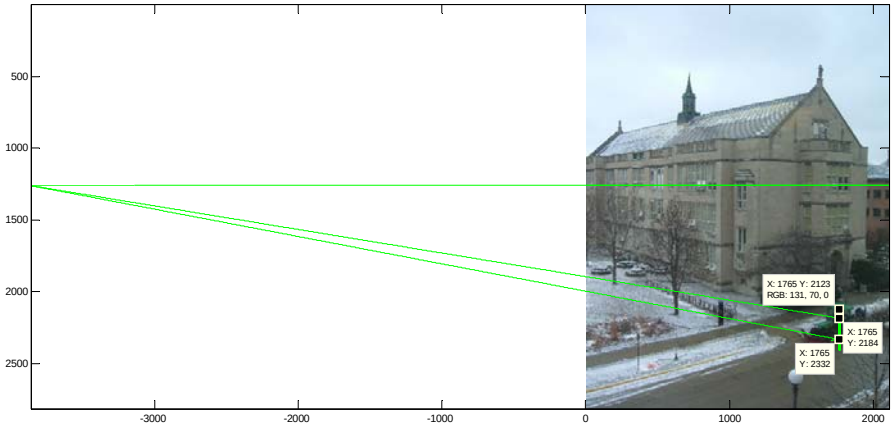
By cross product of vp1 and vp2, we get the horizon line = 1.0e+013 * [0.0000; -0.0060; 7.5649];



Then we draw a line between the bottom of the sign and the building and get the corresponding VP. From the VP, we draw a line go through the upper of the sign and we will know the proportion according to the indicated points in the above figure (their x coordinates are the same). According to the cross ratio invariant principle, the height of the building $h_{building}$ = (1993-923)/(1993-1892)*h_{sign} = 17.4802m;



Since the height of the camera is equal to the vertical distance between the horizon line and the ground. Therefore from the three indicated points in the above figure (their x coordinates are the same) and the cross ratio invariant principle, the height of the camera h_{camera} = (2194-1260)/(2194-2065)*h_{sign} = 11.9465m;



Similarly, the height of the truck h_{truck} = (2332-2123)/(2332-2184)*h_{sign} = 2.3301m; Note that I find the bottom of the truck by first drawing a line going through the bottom of the two wheels and drawing another vertical line going through the upper of the truck. Then the intersection of the two lines is the bottom point of the truck. This is a more precise way to get the bottom of the truck.

2. Mission Possible?

3. Yes, the intuition is that we need to display different views in the projection screen according

to the rotation of the security camera, so that to make the scene as pseudo 3D. To achieve this, we need to know the intrinsic and extrinsic matrix of the camera, the world coordinates X, Y and Z of objects as well as updating the rotation matrix of the camera as it rotates. Then we can update the scene showed on the projection screen.

4. Yes, the idea is similar as the security camera. We need to know the intrinsic and extrinsic matrix of the guard's eyeball and the world coordinates of objects. Since the guard can move around, the difference from the camera is that we need to update both the rotation matrix and the translation matrix.
5. No, we can't display two different views in one projector screen at the same time. But we may fool them both by using the technology like 3D movie.

6. Epipolar Geometry

a)

1. Indicate what test you used for deciding inlier vs. outlier.

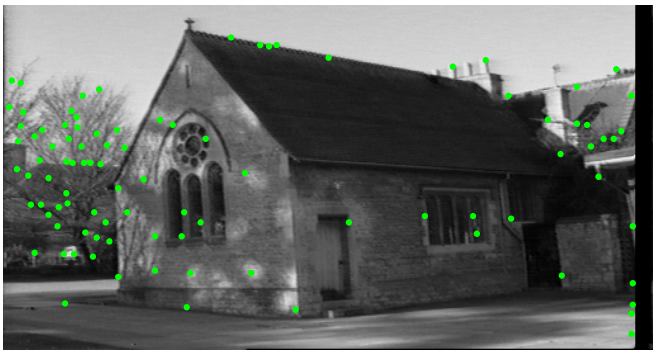
After obtaining the Fundamental Matrix F, I can calculate the corresponding epipolar lines for each pair of matched points in two images. If the distance between point x and the epipolar line L in the first image is less than a threshold (for example 1 pixels) and the distance between point x' and epipolar line L' in the second image is also less than the threshold, then the matched pair is regarded as inlier. Otherwise, they are regarded as outlier. I create a m file called "getInliers" to identify inliers.

The distance between a point and a line can be formulates as $\frac{||au+bv+c||}{\sqrt{a^2+b^2}}$

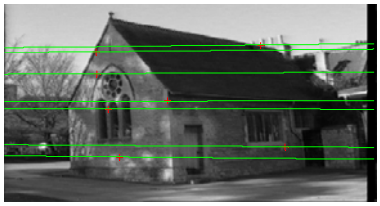
2. Display F

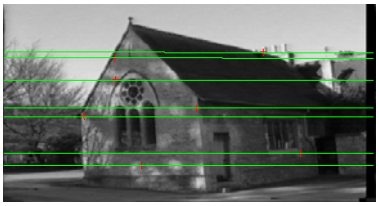
$$F = \begin{pmatrix} 2.7908e-007 & 2.3789e-006 & 0.0013284 \\ -5.6402e-005 & 5.529e-006 & -0.32241 \\ 0.0067154 & 0.33742 & 0.88439 \end{pmatrix}$$

3. Plot the outliers



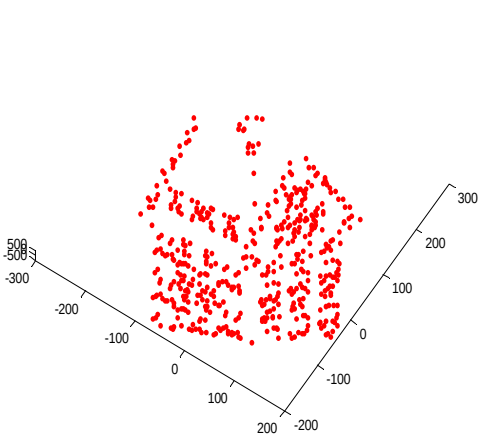
b)

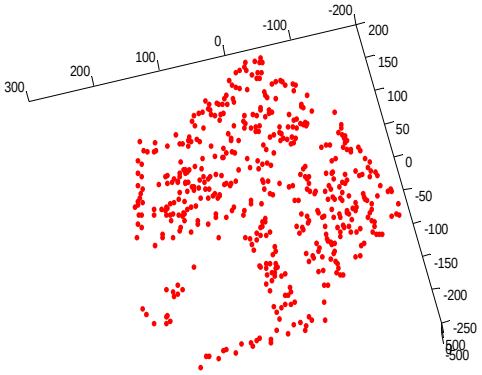


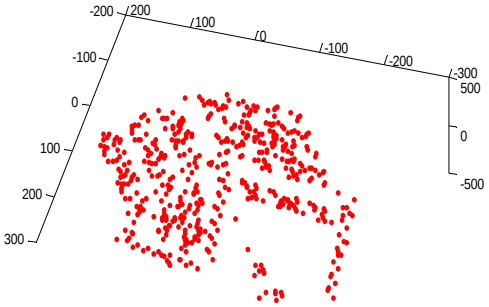


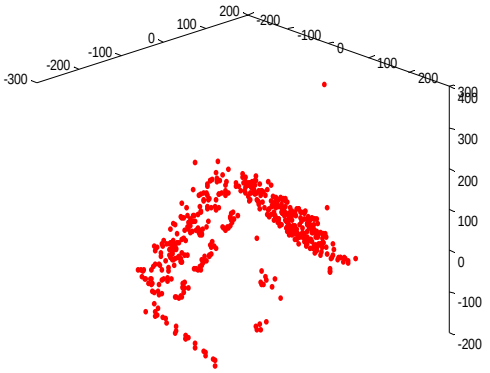
7. Affine Structure from Motion

1. Viewpoints for 3D structure

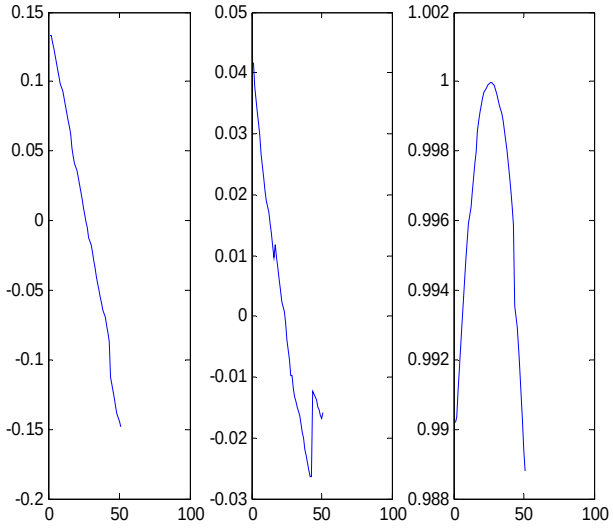








2. 3D path of camera



Pseudo Code:

```
function [K,X,A] = ASM(Xs,Ys)
%inputs:  Xs and Ys are the tracked positions of keypoints through all
frames, there are n keypoints and m frames;
%outputs: K is the 3D path of camera through all frames;
%         X is the 3D position of all keypoints;
%         A is the Affine Projection Matrix;
```

- For each frame, move the center of all key points to zeros;
- Construct a 2m*n measurement matrix D:
  --Column j contains the projection of point j in all views;
  --Row i and m+i contains coordinates of the projections of all key points in image i;
- Compute the SVD of matrix D;
- Create U_3 by taking the first 3 columns of U;
- Create V_3 by taking the first 3 columns of V;
- Create W_3 by taking the upper left 3*3 block of W;
- Create the Affine matrix $A = U_3 * W_3^{0.5}$ and shape matrix $X = W_3^{0.5} * V_3^T$;
- Apply the orthographic constraints on A to eliminate the affine ambiguity:
  --Solve for $L = CC^T$ by least square method; (MATLAB “pinv”);
  --Recover C from L by MATLAB “chol”;
- Update A = A*C and $X = C^{-1} * X$;
- Compute the 3D path of the camera k_i = cross(A(i,:),A(m+i,:)) and normalize;
- Plot;

```
end
```

8. Extra Problem – Missing Track Completion

Output Image:

Fig. Missing Path drawn on the first frame

Pseudo Code:

```
function [X1 Y1] = MissingMatrixCompletion(X,Y)
```

% Input: X and Y are the coordinates of all key points (including missing ones) through all frames;

% Output: X1 and Y1 are the completed matrices after performing Missing Track Completion;

- Points which are not visible in three frames cannot be measured therefore are discarded;
- Use the three frames where all key points exist to perform Affine Structure from Motion algorithm which is described in the problem 4;
- In the remaining each frame i, use all the existed key points to perform Affine Structure from Motion Algorithm:

  --Center X and Y image coordinates;

  --Center X, Y and Z world coordinates;

  --Use $D_i' = a_i * X_i'$ to estimate the affine matrix of frame i, where D_i' and X_i' are the matrices only containing existed key points;

  --Use $D_i * \eta_i = (a_i * X_i + T_i * e^T) * \eta_i$ to estimate the translation T_i, where D_i is the unnormalized X and Y image coordinates containing both existed and missing key points, e^T is the 1*size(D,2) all one vector, η_i is the size(X,2)*1 vector of which the element is 1 if the corresponding key point exists otherwise is 0;

  --Use $D_i = a_i * X_i + T_i * e^T$ to compute the missing elements of D, which are also the track of missing key points;
- Plot;

```
end
```

Note that for one missing key point, we only need as least as three visible key points to estimate its missing track. Therefore, we can estimate each key point in one time when we don’t have three frames where all key points exist, which means we can’t get the X, Y and Z world coordinates of all key points in one time.