CS543: Computer Vision **Spring 2017**

Problem Set 3

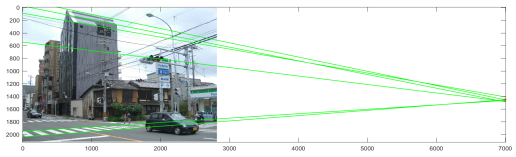
Chongxin Luo (cluo5) *Handed In: March 22, 2017*

1. Single-View Metrology

(a) In part A, for the given image, we used manually selected lines to compute three vanish points in the image. The vanish points were determined according to the building and street orientation in the image, and for each vanish point, several parallel lines in the same direction were selected. Because all selected lines might not intersect at one single point, we compute all intersect points between all selected line pairs, and compute the score for all points, and chose the point with highest score. The score for each intersect point is calculated as :

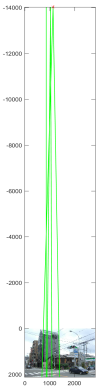
$$s_j = \sum_i |l_i| exp(-\frac{|\alpha_i - \theta_i|}{2\sigma^2})$$

Three vanishing points were calculated, where one vanish point located to the right of image, one located to the left of image, and one located to the top of image. The three vanishing points coordinates are top VP (1249, -15436.5), right VP (6809, 1415), and left VP (-706.7, 1602.8). Figures below shows the thee vanishing points plotted on the original picture.



(a) Right vanish point

(b) Left vanish point



(a) Top vanish point

Figure 2: Three vanishing points for given image

The ground horizon line can be computed as the line connected left VP and right VP. To obtain the line equation connected two points, we perform cross product of the two points in homogeneous coordinate. We computed the horizon line equation $au + bv + c = 0$, and normalized the parameter so that $a^2 + b^2 = 1$. The obtained horizon line equation is : $-0.0155u - 0.9999v + 0.0006 = 0$. Figure below shows the horizon line plotted on the original image.

Figure 3: Ground horizon line

(b) Using the fact that the vanishing points are in orthogonal directions. We can compute the focal length (f), and optical center (u_0, v_0) by solving the following system of equations.

$$P_1 * K^{-T} * K^{-1} * P_2 = 0$$
$$P_1 * K^{-T} * K^{-1} * P_3 = 0$$
$$P_2 * K^{-T} * K^{-1} * P_3 = 0$$

Where K^{-1} is the inverse of intrinsic matrix, and K^{-T} is the transpose of inverse of intrinsic matrix, and the intrinsic matrix K is:

$$K = \begin{bmatrix} f & 0 & u \\ 0 & f & v \\ 0 & 0 & 1 \end{bmatrix}$$

The calculated values are: $u_0 = 1655.4$, $v_0 = 823.5$ and $f = 3422.3$.

(c) In part C, we compute the camera rotation matrix with computed three vanishing points from part A. We assume the top VP is towards Y direction, right VP is towards X direction, and left VP is towards Z direction. We define the rotation matrix as:

$$R = [R_1, R_2, R_3]$$
$$R_1 = kinv * VP_right$$
$$R_3 = kinv * VP_left$$
$$R_2 = R3XR1$$

The rotation matrix that computed for the given image is

$$R = \begin{bmatrix} 0.8293 & 0.0245 & -0.5583 \\ 0.0952 & 0.9783 & 0.1842 \\ 0.5507 & -0.2059 & 0.8089 \end{bmatrix}$$

(d) In part D, we use calculated vanish points and estimated horizon line to estimate objects' height inside an image. First of all, we apply the same algorithm from part A to find the left and right vanish points, and draw the horizon line in the image. Figure below shows the horizon line of the image.

Figure 4: Caption

With the horizon line determined, we can estimate the height of object A with known height of object B according to following method:

i. Draw a line connecting the bottom point of object A with bottom point of object B.

ii. Extend the "bottom" line to intersect to with the horizon line, calculate the intersection point.

iii. Connect the intersection point with top point of object A, extend the line to intersection with object B.

iv. Find the intersect point on object B.

v. Using following ratio to compute object B's height: $\frac{\text{intersect B height - Bottom B height}}{\text{object A height}} = \frac{\text{Top B height - Bottom B height}}{\text{object B height}}$.

The algorithm described above was used to estimate the height of building, and height of tractor, given the height of sign is 1.65 meters. The figure below illustrated the application of the algorithm. For the final estimate, the building's height is 17.82m and the tractor's height is 2.23m.

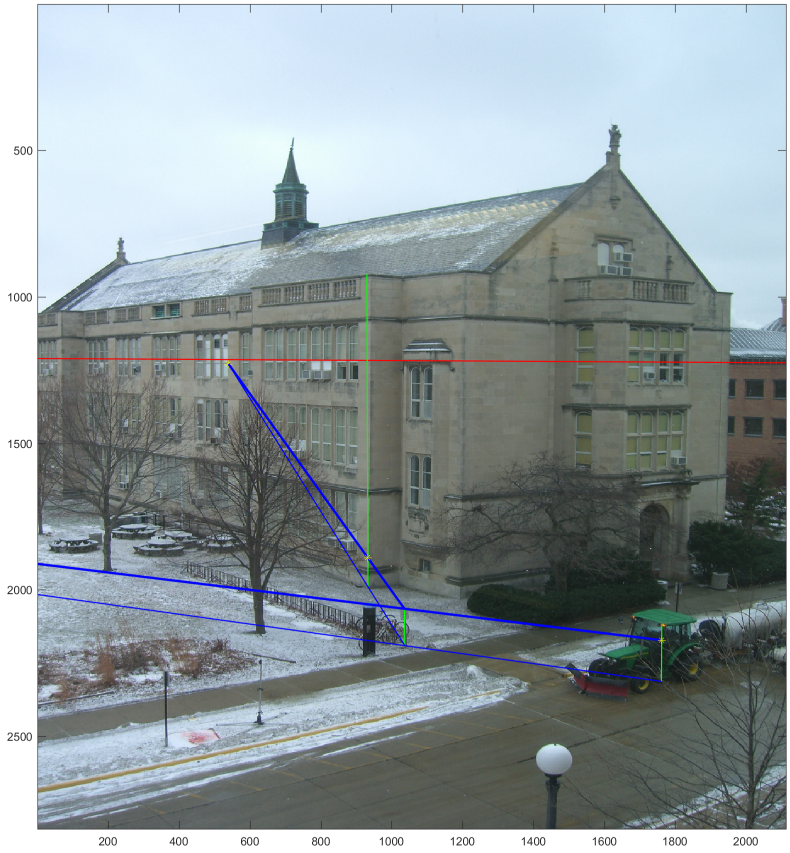


Figure 5: Height Estimation

2. **Mission Possible?**

   (a) For a single camera with rotation only, the projection screen can achieve undetectable, as long as it knows the intrinsic matrix and rotation matrix of the camera, it can produce a perfect seen to trick the camera.

   (b) For a guard with only 1 eye, the projection screen can achieve undetectable, as long as it knows the eye's intrinsic matrix, rotation matrix, and translation matrix, it can produce a perfect seen to trick the guard.

   (c) For both security camera and one-eyed security guard, the projection can no longer trick both of them at once, because for two different locations in the world coordinate, they will have two different intrinsic matrices, rotation matrix, and translation matrices, and the image projection will have two different solutions. This means the projection screen cannot strick both of them at the same time.

3. **Epipolar Geometry**

   (a) We use the matched points provided in prob3.mat, and apply RANSAC and normalized 8-point algorithm to find the Fundamental matrix. Therefore total of 252 pair point matches that provided for this problem, first of all, we normalize all points by compute matrix multiplication $T * x$, where T is the normalize matrix, and x is the homogeneous coordinates of the point.

$$T = \begin{bmatrix} 1/std & 0 & -mean1x/std \\ 0 & 1/std & -mean1y/std \\ 0 & 0 & 1 \end{bmatrix}$$

   Each image is being normalized individually, and two normalized matrix were build, where T1 respond to normalize matrix for image 00, and T2 respond to normalize matrix for image 01.
   Then, with normalized matched points and normalize matrix T1 and T2, we apply RANSAC and normalized 8-point algorithm to solve the Fundamental matrix by apply following algorithm:

```
EightpointAlgo:
  threshold ← 1.1
  GoodPercentage ← 0.5
  currPercentage ← 0

  while currPercentage < GoodPercentage:
      inliers = []; outliers = [];
      pointSet ← randPerm(252, 8)
      A = zeros(8,9)

      for ii from 1 → 8:
          A(ii:;) = [u*up, u*vp, u, v*up, v*vp, v, up, vp, 1];
      end for

      (solve f using SVD – code from lecture)
      (resolve det(F) = 0 using SVD – code from lecture)
      normF = transpose(T1)*F*T2

      for ii from 1 → totalpoints:
          point1 ← [c1(matches(ii,1)); r1(matches(ii,1)); 1]
          point2 ← [c2(matches(ii,2)); r2(matches(ii,2)); 1]
          line1 = normF * point2
          line2 = normF' * point1
          distance1 = computeDistance(line1, point1)
          distance2 = computeDistance(line2, point2)
          if distance1 ≤ threshold || distance2 ≤ threshold
              inliers = [inliers; ii]
          else
              outliers = [outliers; ii]
          end if
      end for

      currPercentage = inliers/totalpoints
      if currPercentage ≥ GoodPercentage:
          bestF = normF
          bestinliers = inliers
          bestoutliers = outliers end if
  end while
```

We used the distance from match point to epipolar lines to determine if a point is inlier or outlier. The threshold was set to 1.1 pixel size, and for points with distance smaller than 1.1 pixel size, they were being categorized as inlier, and for points that had larger than 1.1 pixel size, they were being categorized as outlier. The best Fundamental matrix was calculated and normalized which is shown

below, and the outliers were plotted as green dots shown in the figure below.

$$normF = \begin{bmatrix} 0.0 & -0.0 & 0.0008 \\ 0.0 & -0.0 & -0.0037 \\ -0.0006 & 0.0036 & 0.0245 \end{bmatrix}$$

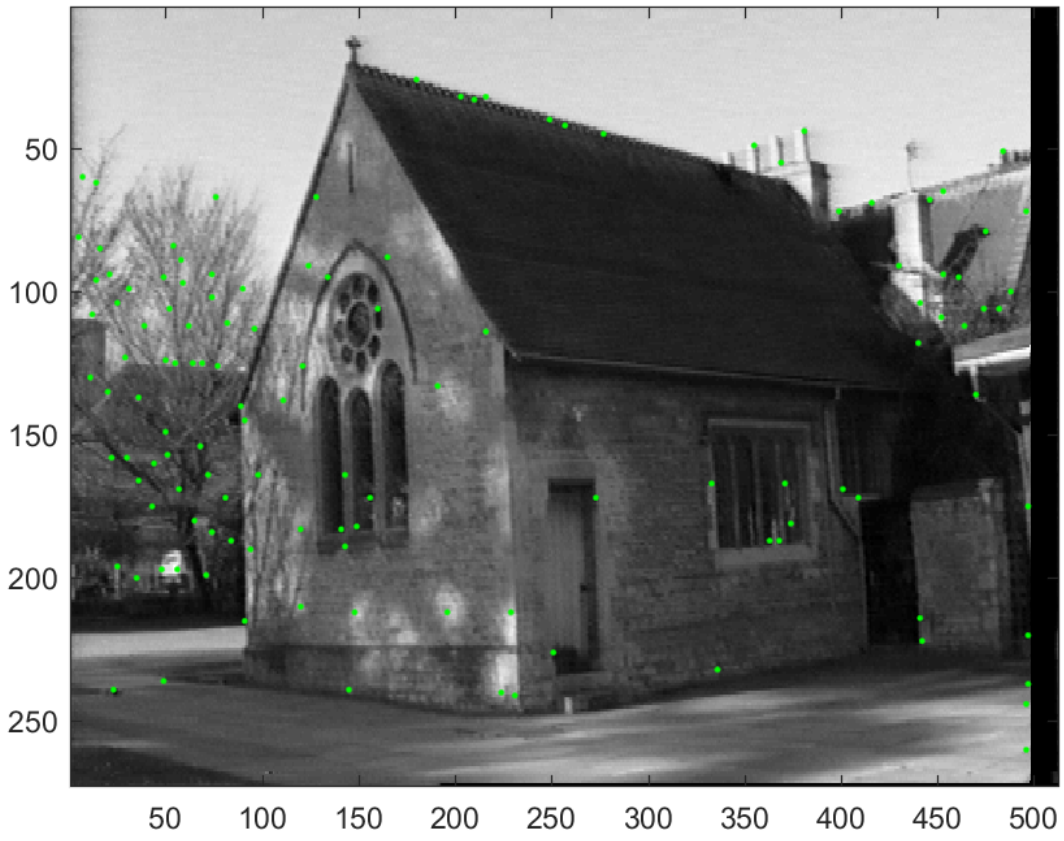
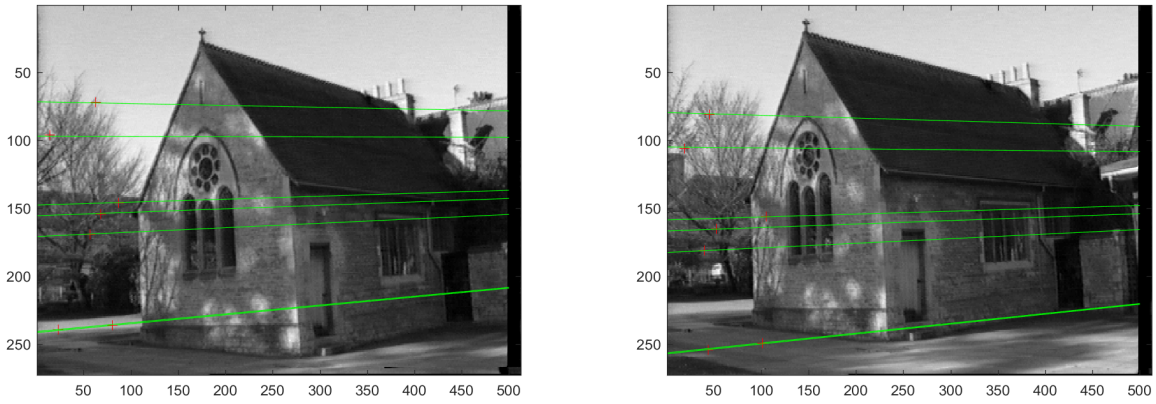


Figure 6: Epipolar outliers

(b) With Fundamental matrix computed from part A, we randomly selected 7 sets of matching points, and plotted the corresponding Epipolar lines in green and the points in read for each image, shown in the figure below.



(a) Epipolarline left

(b) Epipolarline Right

Figure 7: Epipolarline comparison

4. **Affine Structure from Motion**

   (a) In this part, we predicted 3D locations of the tracked points for 3 different views. Given tracking points over 50 frames, and reconstruct the 3D structure. We use Affine structure from motion to reconstruct the 3D point by applying the following steps:

      i. Normalize both x value and y value of critical points. We construct the 2m x n D matrix, where m is the number of frames and n is the number of tracking points, and each line alternates between x and y values.

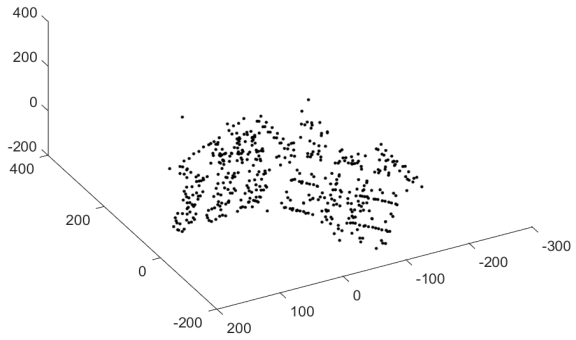
      ii. Compute the first guess of matrix A, and matrix X using Singular Value Decomposition. Where A is the transition matrix, and X is the 3D shape matrix.

      iii. Solve for orthographic constraints by solving following system of equations

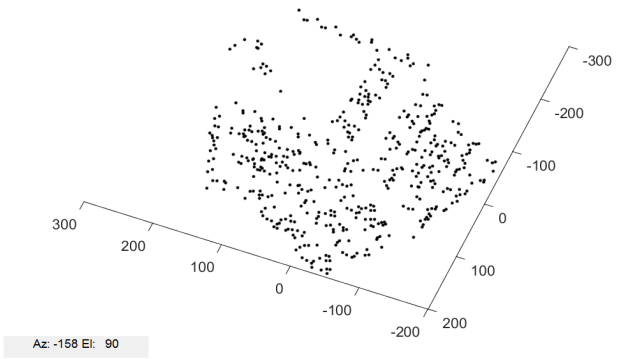
$$a_{i1}^T * L * a_{i1} = 1$$
$$a_{i2}^T * L * a_{i2} = 1$$
$$a_{i1}^T * L * a_{i2} = 0$$

      iv. Recover C from L by Cholesky decomposition, and update A and X by $A = AC, X = C^{-1}X$

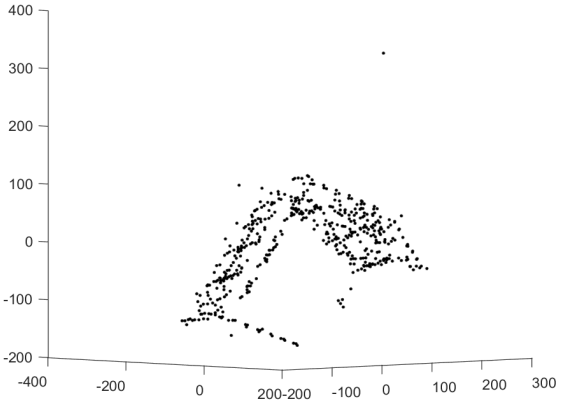
   The 3D points were being reconstructed and three images from three different angles are shown in the figure below.



(a) 3D points 1



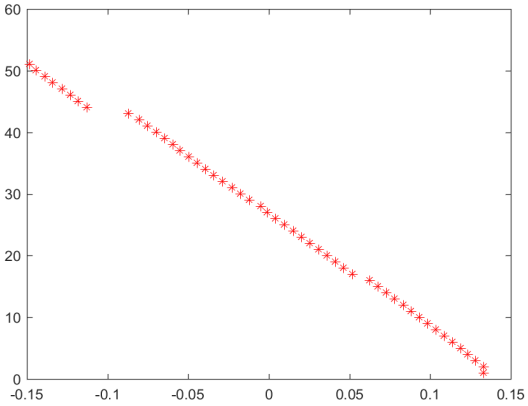
(b) 3D points 2



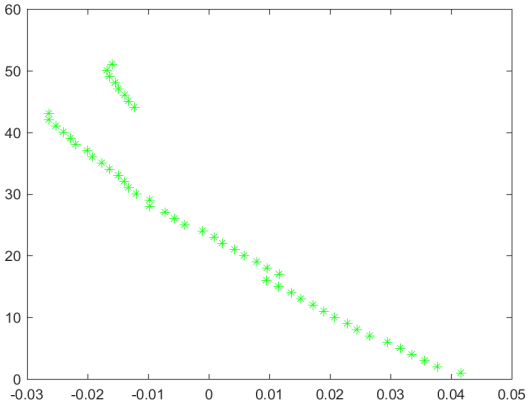
(c) 3D points 3

Figure 8: 3D points reconstruction

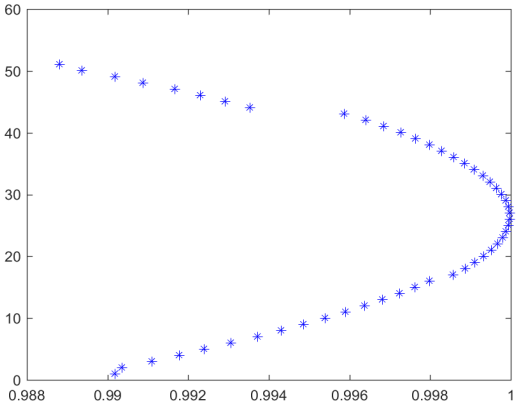
(b) Using the calculated motion matrix A from part 1, we can recompute the camera motion by matrix cross product $k_f = i_f X j_f$. The three following figures are the camera translations in three axies.



(a) Camera Rotation 1



(b) Camera Rotation 2



(c) Camera Rotation 3

Figure 9: Camera Rotations Reconstruction