16-822: Geometry Based Methods
Assignment 2

Andrew ID: sajaved

Question 1

1.1

1.1.1

The points given are p_1 and p_2. In order to join on these points, we need the following relation

$$p_1 \vee p_2 = p_1 \times p_2 \tag{1}$$

Similarly, we can apply the same concept to lines as well. We have two lines as l_1 and l_2. Thus the meet operation becomes

$$l_1 \wedge l_2 = l_1 \times l_2 \tag{2}$$

1.1.2

The dual of a line can be represented as planes in the 3D space as follows

$$P = \Pi_1 \wedge \Pi_2 \tag{3}$$

1.1.3

A plane in 3D can be constructed using 3 non-collinear points. We can use the equation of join in points to get the plane. This is given as follows :

$$\Pi = M_{\Pi}^{+} p_1 \vee M_{\Pi}^{+} p_2 \vee M_{\Pi}^{+} p_3 \tag{4}$$

1.2

1.2.1

Writing the camera equation and the homography relation:

$$p_1 = M_1 P$$
$$p_2 = M_2 P$$
$$p_2 = H p_1$$

We also know that the epipole is a projection of the camera center C:

$$e' = M_2 C$$

If we take a point on the line joining the camera center C and the 3D point P, we can rewrite any point on this line as linear combinations of these two, which in turn means that we can rewrite the Hp_1, p_2, e' as

linear combinations of each other. (as can be seen from the above equations). More formally, let P' be the linearly combined 3D point.

$$\begin{aligned} Hp_1 &= M_2 P' \\ Hp_1 &= M_2((1-\lambda)C + \lambda P) \\ Hp_1 &= M_2(1-\lambda)C + M_2\lambda P \\ Hp_1 &= (1-\lambda)e' + \lambda p_2 \end{aligned}$$

Thus the three points are aligned.

1.2.2

Using the previous result:

$$\lambda p_2 = Hp_1 - (1-\lambda)e'$$

This if of the same form as $p_2 = Hp_1 + re'$

1.2.3

Let F be the fundamental matrix relating image 1 and image 3.

$$p_3^T F p_1 = 0$$

We also know that the two planes are related by a homography H. This gives us a relation from Hartley and Zisserman relating the fundamental matrix.

$$F = [e]_\times H$$

Rewriting the first equation using the result computed in the previous part for relating p_1 and p_3:

$$\begin{aligned} p_3^T [e]_\times H p_1 &= 0 \\ p_3^T [e]_\times (p_2 - re') &= 0 \\ p_3^T [e]_\times p_2 - p_3^T [e]_\times re' &= 0 \end{aligned}$$

Since the two epipoles lie on the same line, the second term is zero because of the cross product. Thus:

$$p_3^T [e]_\times p_2 = 0$$

Also, we know that for $K = I$, the fundamental matrix can be written as $F = [t]_\times R$. Therefore the rotation matrix $R = I$ in the above equation relating p_3 and p_2 and the two images are related by pure translation.

1.2.4

We have shown in the above parts that for image planes related by epipolar geometry (p_1, p_2), the homography based relation contains an additional projective depth term which is contains the epipole in the second image. Also, we can show that the epipole in the third image is at zero since the two planes, plane 1 and plane 3 are only related by a homography $p_3 = Hp_1$ and $e' = 0$ (ie, epipole in the image 3). We also showed that the translation vector between image 2 and image 3 is equal to the epipole position. Thus the translation between image 1 and image 3 is zero.

1.3

1.3.1

We know that:

$$\begin{aligned} P' &= HP \\ P_0 &= QP \\ P_0' &= QP' \\ P_0' &= H_0 P_0 \end{aligned}$$

The eigen equation for H_0 is given by:

$$H_0 v = \lambda v$$

Rearranging the terms:

$$\begin{aligned} P_0' &= QP' = QHP = H_0 QP \\ H_0 &= QHQ^{-1} \end{aligned}$$

The eigen equation for the above matrix H_0 is given by:

$$\begin{aligned} QHQ^{-1}v &= \lambda v \\ HQ^{-1}v &= \lambda Q^{-1} v \end{aligned}$$

Thus $Q^{-1}v$ is an eigenvector of H with the same values

1.3.2

$$\begin{aligned} T^* &= T^{-T} \\ T^{-\top} &= \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}^{-\top} = \begin{bmatrix} R^\top & 0 \\ t^\top & 1 \end{bmatrix} \end{aligned}$$

We know that the plane at infinity π_∞ is invariant under the transformation T^*.

$$\begin{aligned} T^* \pi_\infty &= \lambda \pi_\infty \\ \begin{bmatrix} R^\top & 0 \\ t^\top & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} &= \lambda \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

After multiplication, only the plane at infinity remains. Thus the plane at infinity is an eigenvector of the given transformation.

1.3.3

We know from the previous section that:

$$T^* \pi_\infty = \lambda \pi_\infty$$

Since H_0 is a Euclidean transformation, we can write,

$$H_0^{-T}\pi_\infty = \lambda\pi_\infty$$

Now since Q is given to be a correction matrix which does projective to Euclidean transformation, we can write the following expression:

$$\begin{aligned} H_o^{-\top}Q^{-\top}\pi_\infty &= Q^{-1}\lambda\pi_\infty \\ QH_o^{-\top}Q^{-\top}\pi_\infty &= \lambda\pi_\infty \\ \left(Q^{-1}H_oQ\right)^{-\top}\pi_\infty &= \lambda\pi_\infty \end{aligned}$$

We showed in the previous questions that:

$$\begin{aligned} H_0 &= QHQ^{-1} \\ H &= Q^{-1}H_0Q \end{aligned}$$

Substituting this value in the above equation:

$$H^{-\top}\pi_\infty = \lambda\pi_\infty$$

Thus the eigenvector of H^{-T} is the plane at infinity π_∞ and the eigen values are real.

1.4

1.4.1

We need to find the expression for $[p]_\times^2$. Using the expression for double cross product:

$$\begin{aligned} [p]_\times^2 x &= p \times (p \times x) \\ &= \left(p^T x\right) p - \left(p^T p\right) x \\ &= \left(pp^T\right) x) - \left(p^T p\right) x \\ [p]_\times^2 &= pp^T - p^T p \end{aligned}$$

1.4.2

We know that

$$S = [e']_\times F$$

Pre-multipying by e' and using the above derived expression:

$$\begin{aligned} [e']_\times S &= [e]_\times^2 F \\ [e']_\times S &= (e'e'^T - e'^T e')F \end{aligned}$$

We know that $e'^T F = 0$.

$$\begin{aligned} [e']_\times S &= e'e'^T F - e'^T e' F \\ [e']_\times S &= -e'^T e' F \end{aligned}$$

Since F is defines up to scale, we get:

$$[e']_\times S = F$$

1.4.3

We know that F is singular and of rank 2. Hence, S is also singular and of rank 2 using the expression above.

Question 2

2.1

Method

To find the vanishing point, we follow the process done in the lecture slides provided with the assignment. An overview of the pipeline is as follows:

1. **Annotate all edges of the image:**
   The edge detection is based on Canny edge detector

2. **Use the edges to estimate line segments over the edges:**
   The line segments are found using Probabilisitic Hough lines from skimage library.

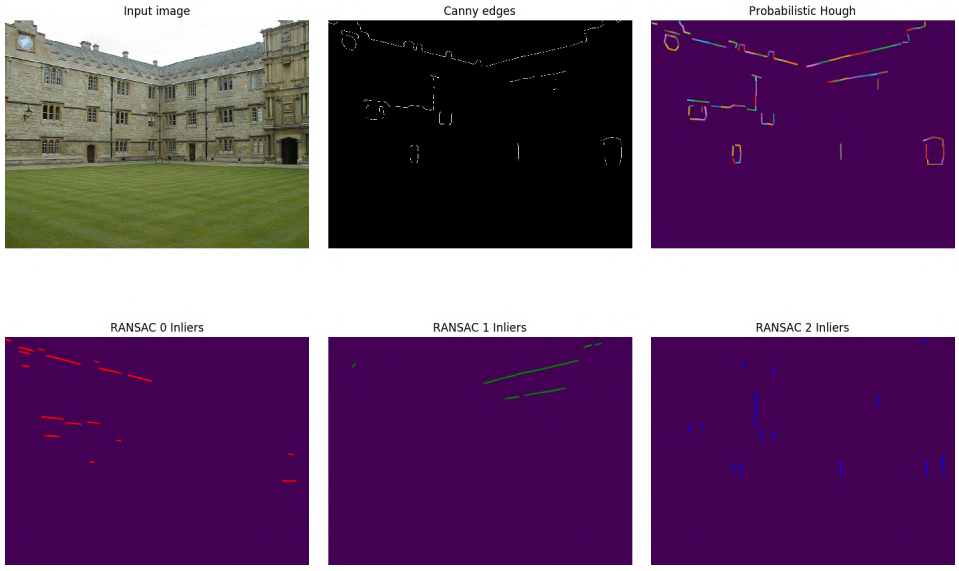
3. **Cluster parallel line segments pointing in the same direction:**
   Instead of clustering the line segments based on their angles with a fixed origin, a RANSAC is run to find the dominant set of line segment directions. This is done iteratively, 3 times, each time removing the inlier lines found for the subsequent RANSAC. The RANSAC on line segments is run with the metric as the cosine of the angles of each line segment and the line obtained by joining one end of the line segment and the hypothesis vanishing point which is under consideration at a particular iteration. This is similar to the method implemented in the 1st reference paper. Instead of using line strength to augment RANSAC votes, the Hough line parameters are changed to ensure that only long lines are chosen. The details of the parameters used are present in the argument parser in the code.

4. **Using the set of multiple lines, find the best fitting vanishing point in each of the three directions for each cluster:**
   After the clustering, each set of inliers for the 3 clusters are used for computing the final vanishing point. This best hypothesis is returned by the function.

Results

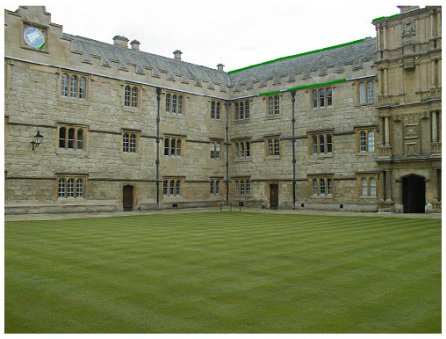
The intermediate results and the vanishing points for the required images are shown below:

Image 1



(a) Intermediate Results

(a) Vanishing Pt 1

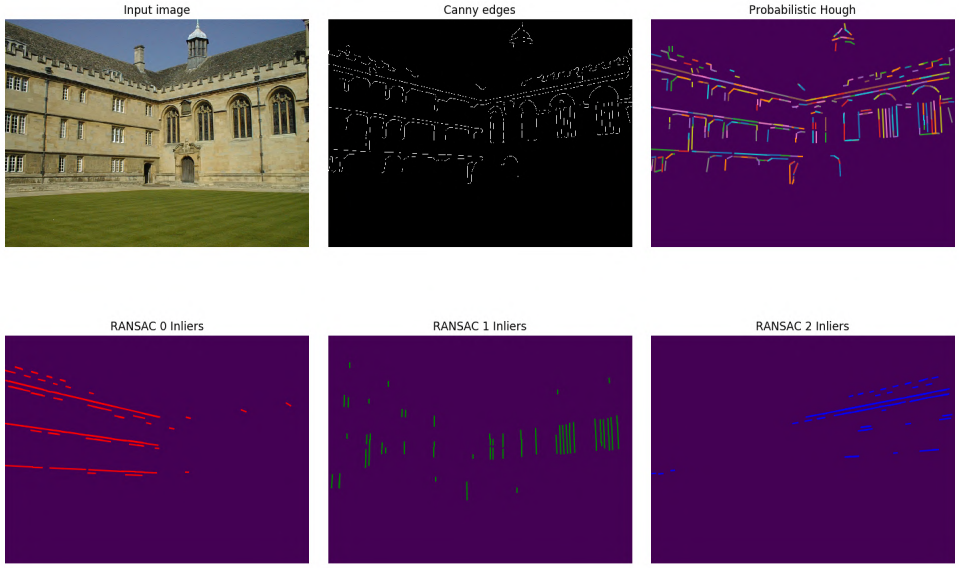


(b) Vanishing Pt 2

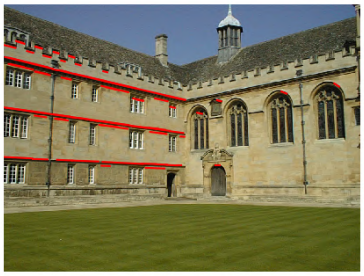
(a) Vanishing Pt 3

(b) All Vanishing Pts

Image 2

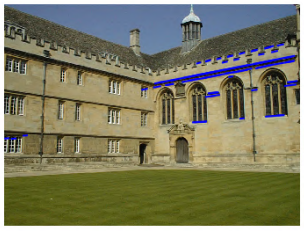


(a) Intermediate Results



(a) Vanishing Pt 1

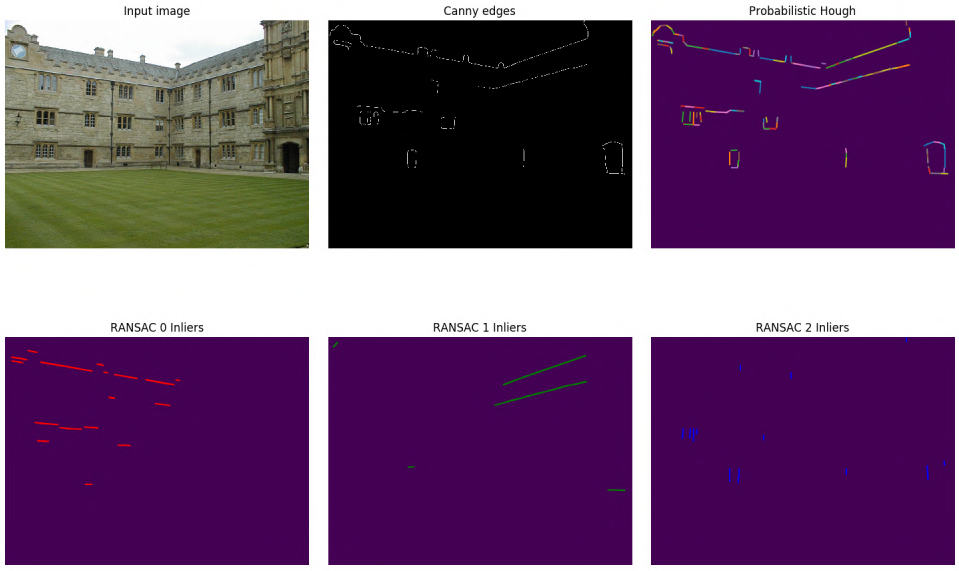
(b) Vanishing Pt 2



(a) Vanishing Pt 3

(b) All Vanishing Pts

Image 3



(a) Intermediate Results

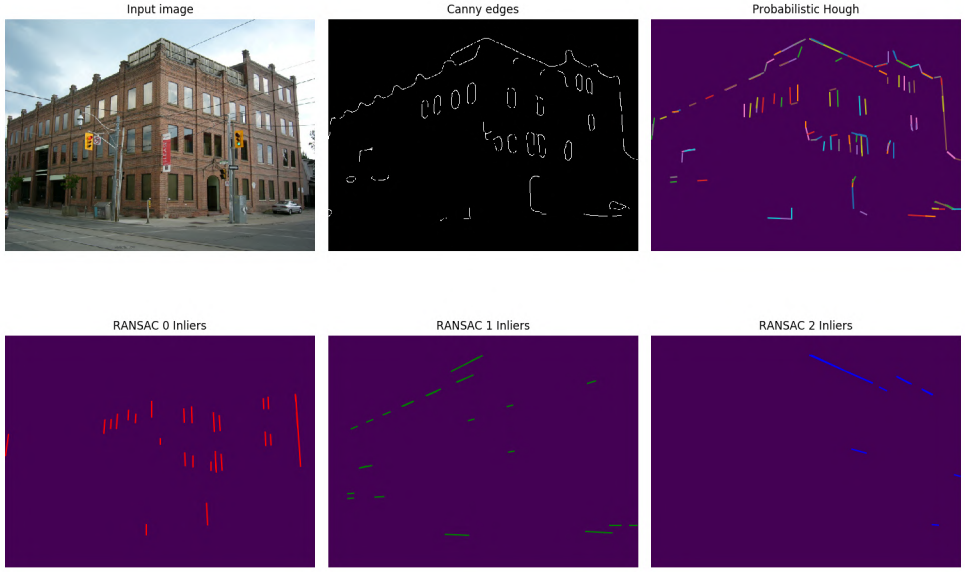
(a) Vanishing Pt 1

(b) Vanishing Pt 2

(a) Vanishing Pt 3

(b) All Vanishing Pts

Image 4



(a) Intermediate Results

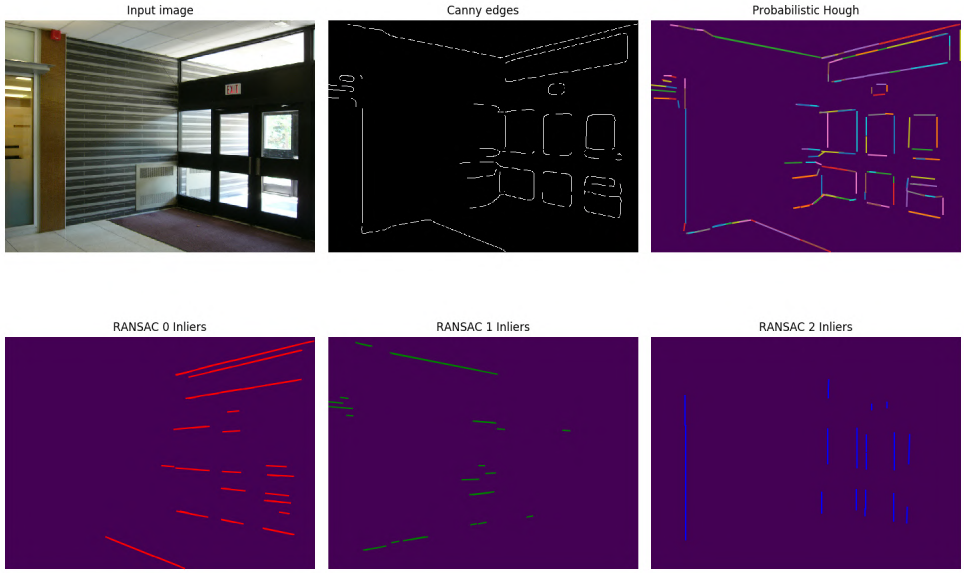
(a) Vanishing Pt 1

(b) Vanishing Pt 2

(a) Vanishing Pt 3

(b) All Vanishing Pts

Image 5



(a) Intermediate Results

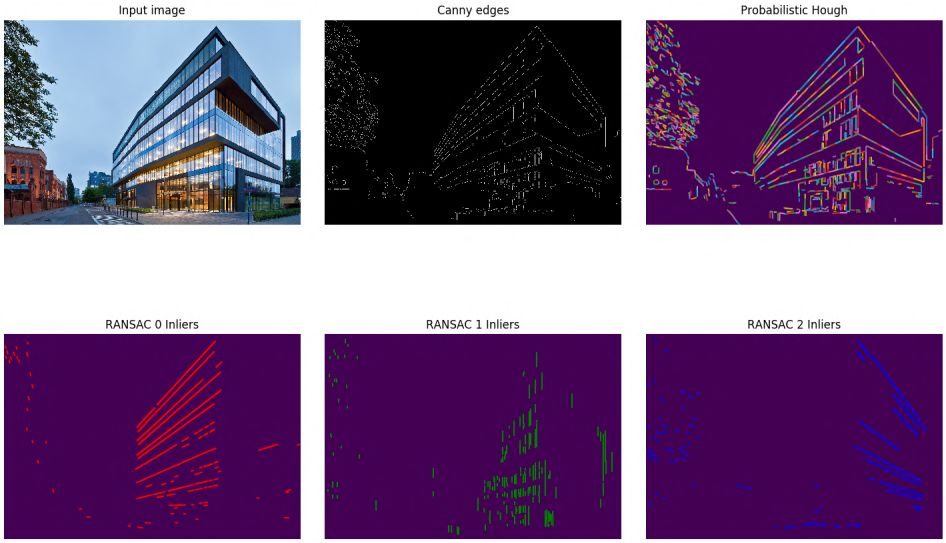
(a) Vanishing Pt 1

(b) Vanishing Pt 2

(a) Vanishing Pt 3

(b) All Vanishing Pts

Image 6



(a) Intermediate Results

(a) Vanishing Pt 1

(b) Vanishing Pt 2

(a) Vanishing Pt 3

(b) All Vanishing Pts

Difficulties Encountered

The following are some of the difficulties encountered during this question:

- The formulation of line segment RANSAC can be done in multiple ways like using distance of the line from the hypothesis vanishing point as the metric or using the difference in the angles between the line segment and the one subtended by its end point and the hypothesis vanishing point as the metric. The use of distance from the point yielded incorrect results although it's a better formulation.
- For certain scenes with multiple sources of parallel lines, the noise was too much for RANSAC to handle and either not enough inlier points were found or the ones found were incorrect, thus affecting the vanishing points.

Executing the Code

The following commands can be used to replicate the results:

- pip install -r requirements.txt
- python python_code/vanishing_point_detection.py -f images/input/1_001.jpg

The argument parser gives the flags available for the code.

2.2

Method

To find the planar homographies of the different segmented planes in the stereo pair and the 3D reconstruction of the scene planes, the following pipeline is used:

1. **Getting SIFT features and stereo correspondences:**
   SIFT features are extracted for the stereo pair and matches are established using a KNN based matcher.

2. **Autocalibration using vanishing points:**
   The 3 vanishing points found in the previous section are used to estimate the camera intrinsic matrices for both images. The function in the code supports both manual labelling and using the first question's code to get the vanishing points. The method implemented is as follows:

   We know that given a vanishing point v, the camera equation is given by:

   $$p = KRv$$
   $$v = R^T K^{-1} p$$

   where p is the projected 2d points, R is the rotation matrix and K are the intrinsics which are to be estimated. If 2 orthogonal vanishing points can be found, we have:

   $$v_1^T v_2 = 0$$
   $$p_1^T K^{-T} R R^T K^{-1} p_2 = 0$$
   $$p_1^T K^{-T} K^{-1} p_2 = 0$$

   The term $K^{-T}K^{-1}$ or KK^T is denoted as ω and we can solve for the linear system using SVD and taking the last null space fo the formed system. In order to retrieve K from ω, we can decompose it using Cholesky decomposition.

3. **Estimating fundamental matrix and essential matrix from correspondences and camera intrinsics:**
   A RANSAC using the seven point algorithm is used to get the fundamental matrix estimate from the point matches. After getting the best set of inliers, the final estimate is obtained using the eight point algorithm and refining using an non-linear optimization. The essential matrix is then computed using the intrinsics and the F estimate using $E = K^T F K$.

4. **Computing the camera matrices for the stereo pair from the essential matrix:**
   The stereo extrinsics are obtained by decomposing the essential matrix into R and t ($E = [t]_\times R$) using the method described in the Hartley and Zisserman book. This is used to construct the camera matrix for both cameras.

5. **Triangulating the 3D points using the correspondences and the camera matrices:**
   The two sets of sparse correspondences are used along with the camera matrix to get sparse 3D points using triangulation. This is effectively done by solving a system of linear equations for each point using SVD.

6. **Running iterative RANSAC to find dominant planes in 3D:**
   A plane RANSAC is run to find the plane equation in 3D and dominant inliers are iteratively removed to find subsequent plane equations. The metric used for this RANSAC is the distance of the points form the hypothesis plane equation and the details of the parameters are present in the code.

7. **Projecting 3D points to 2D and finding planar homography for each identified plane:** The 3D points are next projected to the two images using the camera matrices. Since we have segmented inlier points lying on the plane in 3D, we can now segment the sparse points in both images in 2D. A homography using 4 matching points is estimated for each plane. This too is done using RANSAC.

8. **Using a convex hull on the projected points to get all the pixels on each plane in 2D:** To obtain dense correspondences, a convex hull is created in the first image for each plane individually. This gives dense points for each individual plane.

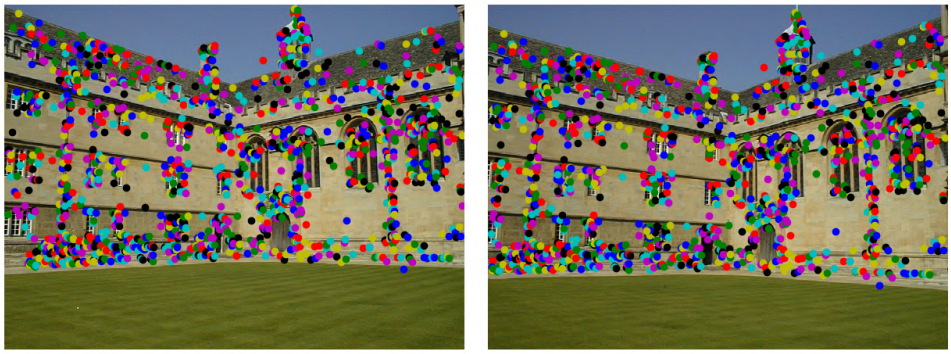
9. **Finding dense correspondences using the convex hull points and the planar homography for each plane:** The convex hull is used to identify each plane's dense points in one image and then the corresponding points in the second image is calculated using the plane homography computed before. This gives dense correspondences for all pixels within the convex hull of each plane.

10. **Triangulating dense 3D points using the dense 2d correspondences for each plane:** The two set of dense correspondences are then used to reconstruct the 3D points.

Results

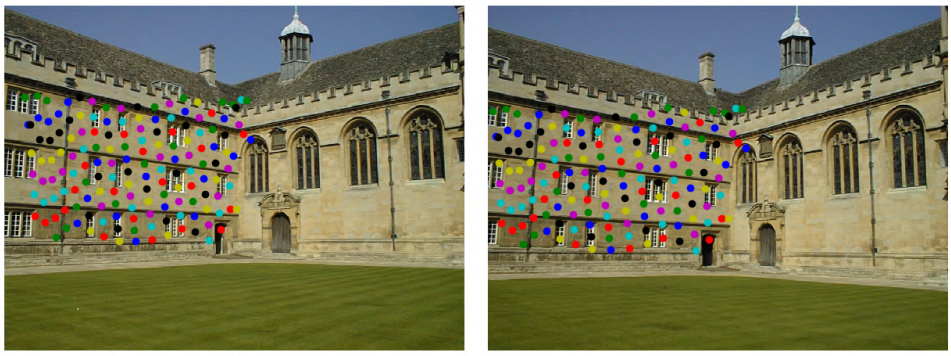
The results show some visualization for some intermediate steps followed by the plane detection and 3d reconstruction of the planes.

The feature correspondences are shown for the stereo pair. The matches are color coded.



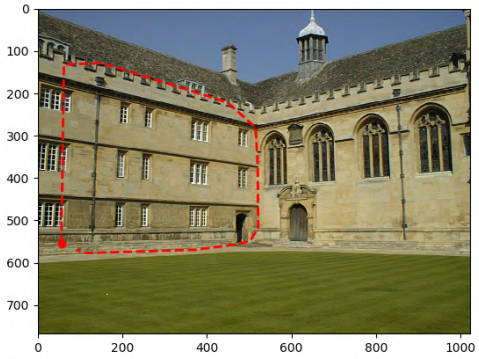
(a) SIFT Feature matches

The feature correspondences after plane segmentation are shown for the stereo pair. The matches are color coded.

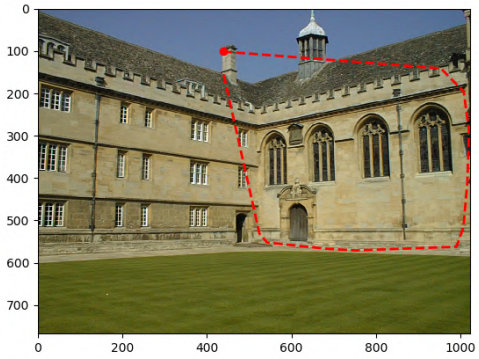


(a) SIFT Feature matches for a detected plane

The convex hull intermediate results are shown below for one sample image.

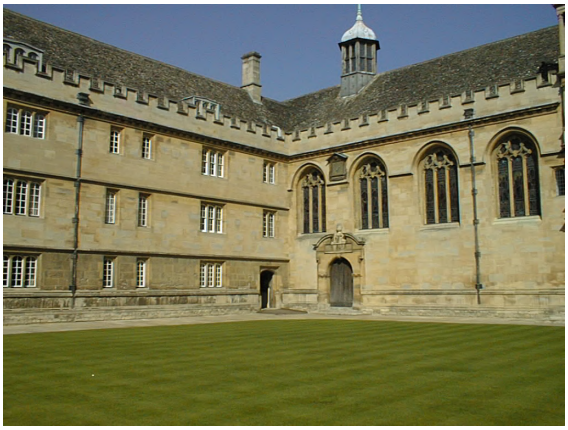


(a) Convex Hull- Plane 1

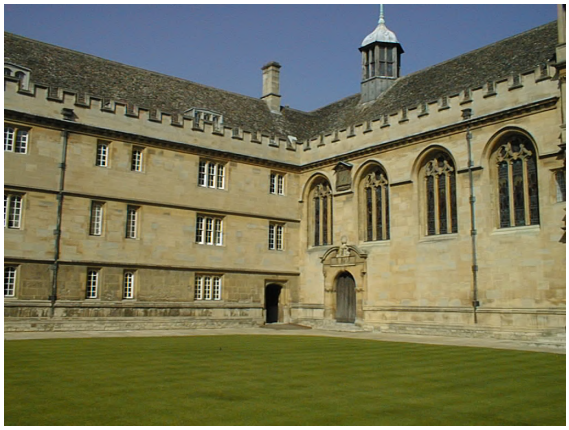


(b) Convex Hull- Plane 2

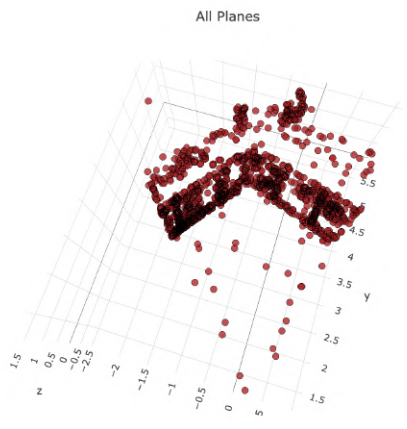
Image 1



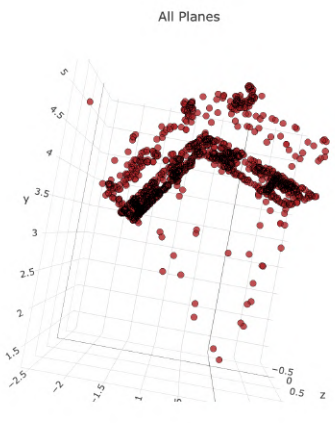
(a) Input Image 1



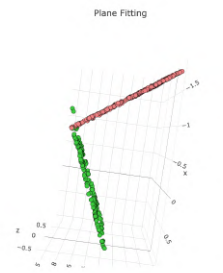
(b) Input Image 2



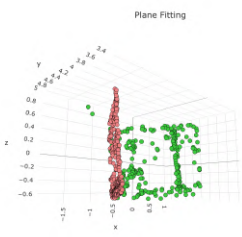
(a) 3D points



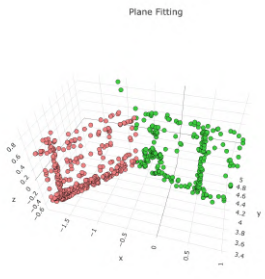
(b) 3D points



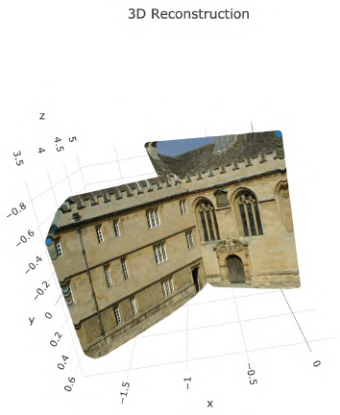
(a) Plane Segmentation



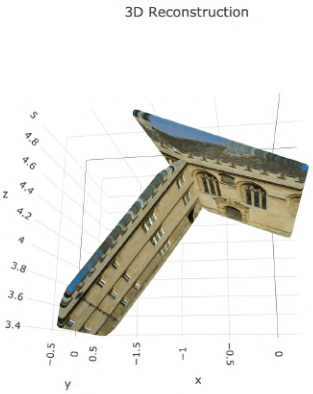
(b) Plane Segmentation



(c) Plane Segmentation

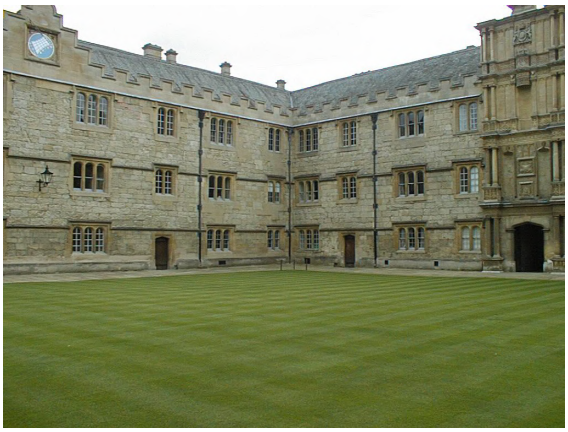


(a) Dense 3D planes

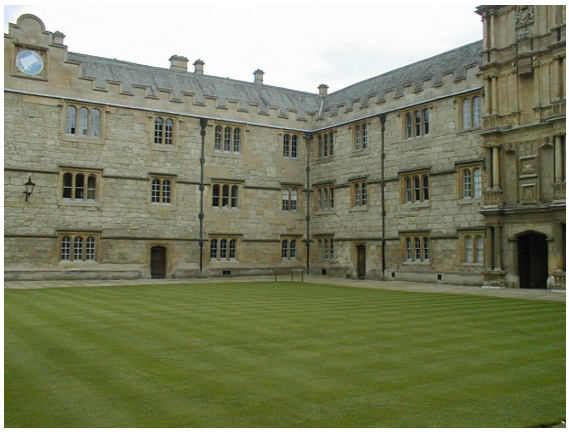


(b) Dense 3D planes

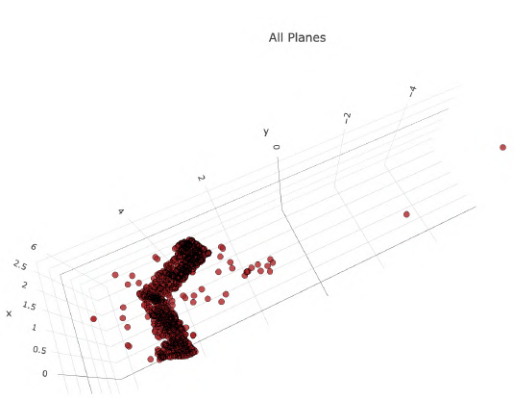
Image 2



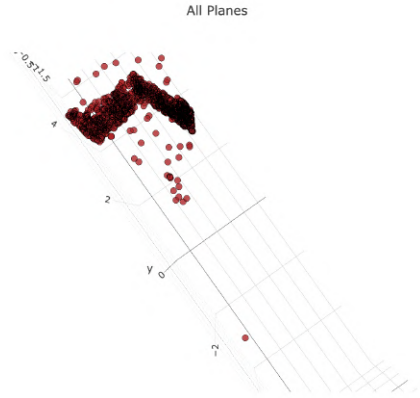
(a) Input Image 1



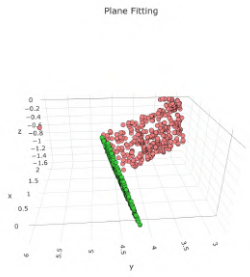
(b) Input Image 2



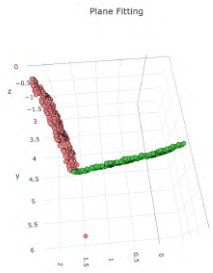
(a) 3D points



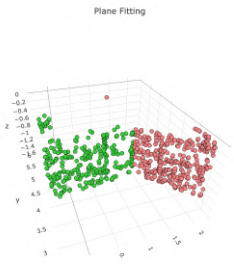
(b) 3D points



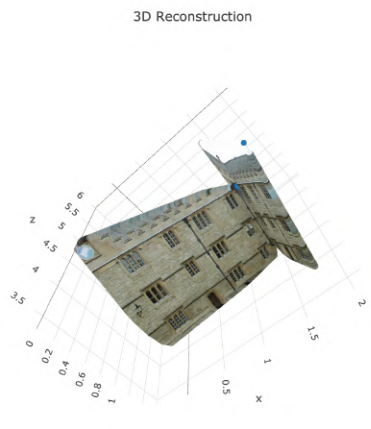
(a) Plane Segmentation



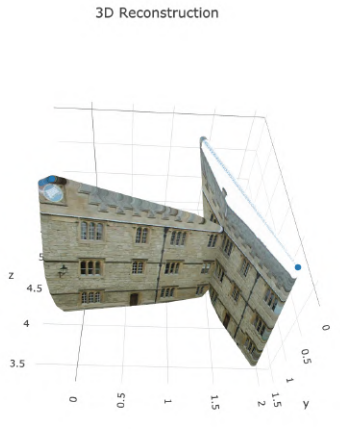
(b) Plane Segmentation



(c) Plane Segmentation

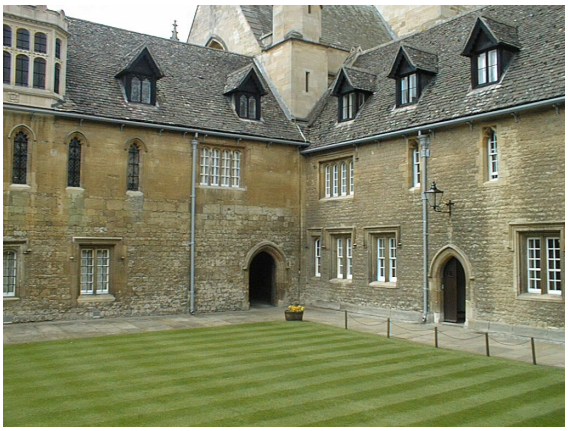


(a) Dense 3D planes

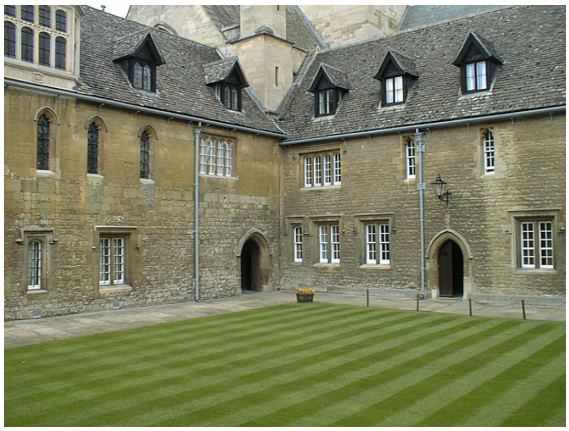


(b) Dense 3D planes

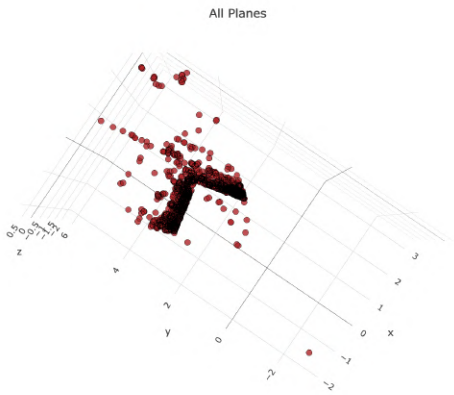
Image 3



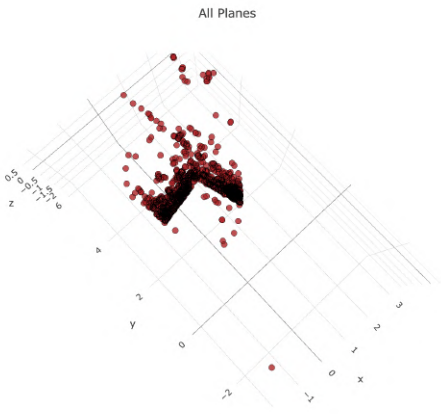
(a) Input Image 1



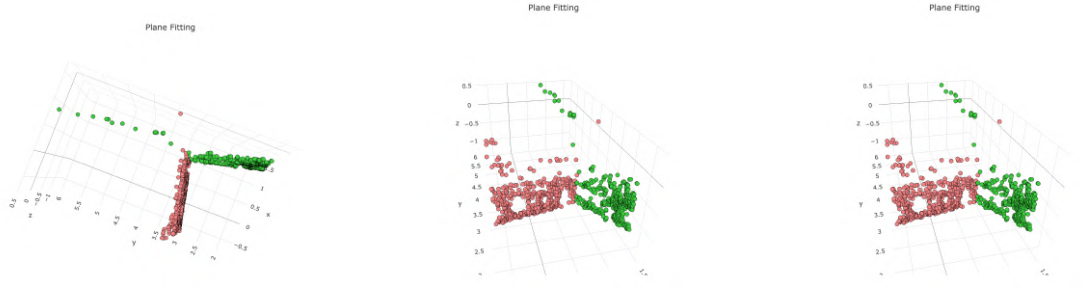
(b) Input Image 2



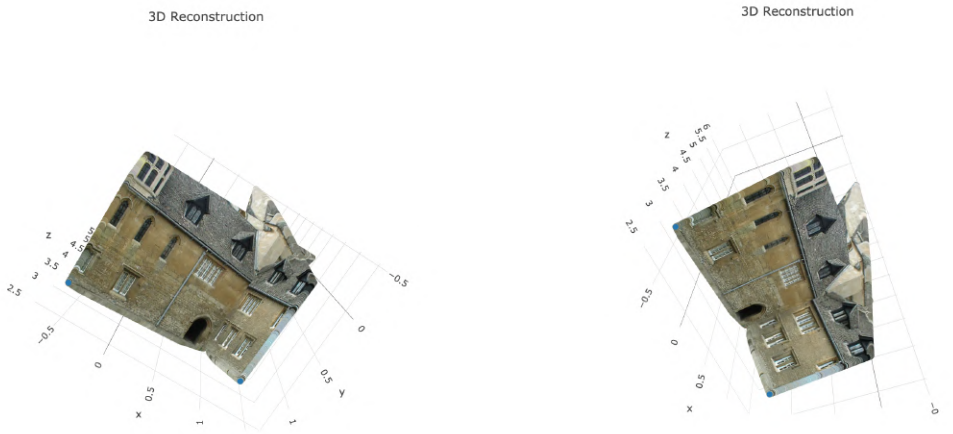
(a) 3D points



(b) 3D points



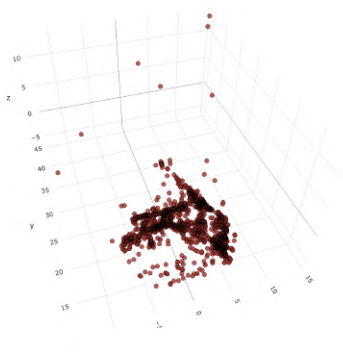
(a) Plane Segmentation (b) Plane Segmentation (c) Plane Segmentation



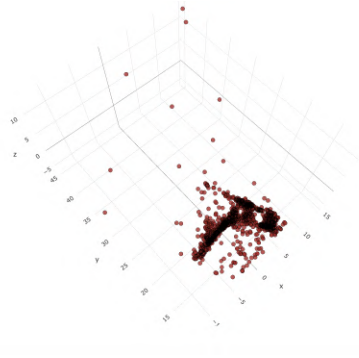
(a) Dense 3D planes (b) Dense 3D planes

Image 4

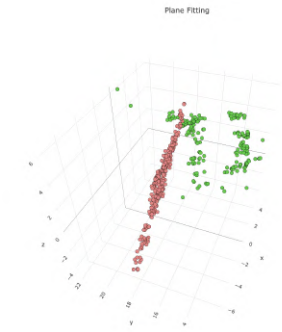
(a) Input Image 1 (b) Input Image 2



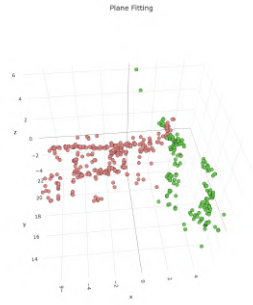
(a) 3D points



(b) 3D points



(a) Plane Segmentation



(b) Plane Segmentation

Difficulties Encountered

The following are some of the difficulties encountered during this question:

- The autocalibration step often computes vanishing points, which when fed to the system of equations doesn't give positive semi-definite matrix which throws an error in Cholesky decomposition. Finding the correct points needs multiple runs.
- The whole procedure involves computing multiple estimates of different quantities which adds to the total error of the system. For example, estimating the 3d plane equation itself is a function of various estimations of the fundamental and camera matrices and errors in the computation result in 3D reconstruction which is pretty off (for the last image, the method couldn't get a good reconstruction of the planes).

Executing the Code

The following commands can be used to replicate the results:

- pip install -r requirements.txt
- python python_code/reconstruct_planes.py -f1 images/input/1_001.jpg -f2 images/input/1_002.jpg

The argument parser gives the flags available for the code.

Collaboration and online resources used

Some approaches were discussed with Talha Ahmad Siddiqui and Rishi Madhok. A few online resources were consulted during the derivation:

- Vanishing point detection
- Canny edge detection and Hough lines
- Plane Segmentation
- Epipolar Geometry and the Fundamental Matrix from Multi-View Geometry textbook