ENGN1610 & 2605 Image Understanding
Lab07: Image Formation and Essential Matrix

The goal of this lab is to

- Learn how an image is formed by projecting 3D points to 2D image plane.
- Learn how to estimate an essential matrix under a RANSAC scheme.

Camera Projection: Projecting a 3D scene point to a 2D point In order to know how an image is formed, we must know how a world scene is projected to an image plane. Suppose the rotation matrix $R \in \mathbb{R}^{3\times 3}$ and the translation vector $T \in \mathbb{R}^{3\times 1}$ of a camera in a world coordinate is known. Let a 3D point in a world coordinate be Γ_w; the transformation of this point from the world coordinate to the camera coordinate is then related by

$$\Gamma_c = R\Gamma_w + T. \tag{1}$$

The 3D point Γ_c under the camera coordinate can be represented as the 2D point scaled up by a depth ρ, Figure 1, *i.e.*,

$$\Gamma_c = \rho\gamma, \tag{2}$$

where $\gamma = [\xi, \eta, 1]^T$ is the 2D point in *meters* represented by the homogeneous coordinate. The transformation of the 2D point γ under the camera coordinate (black in Figure 1) to the 2D point $\gamma_{im} = [\xi_{im}, \eta_{im}, 1]^T$ on an image coordinate (red in Figure 1) is achieved by a *calibration matrix* K (sometimes it is also called the *camera intrinsic matrix*),

$$\gamma_{im} = K\gamma, \tag{3}$$

where γ_{im} is a 2D point in *pixels*. We will talk more about the calibration matrix in the next lecture. In summary, projecting a 3D point in a world coordinate to a 2D image plane can be captured by

$$\gamma_{im} = \frac{1}{\rho} K \left(R\Gamma_w + T\right). \tag{4}$$

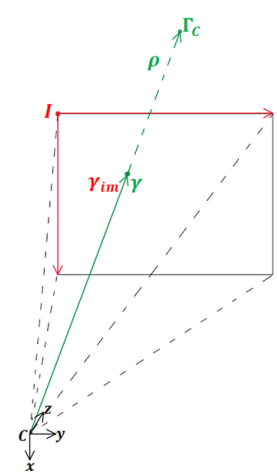


Figure 1

Problem 1. Image Formation Assume that the two cameras have the same resolution of 1200 × 1600 pixels (rows × columns) with the same calibration matrix capturing the same 3D scene, Figure 2. The 3D scene is represented by a dense cloud of 3D points storing $[X, Y, Z]$ coordinates together with their colors stored in (R, G, B), and the relative poses of the two cameras with respect to the world coordinate are represented by their rotation and translation matrices, *i.e.*, R_1, T_1, R_2, and T_2.

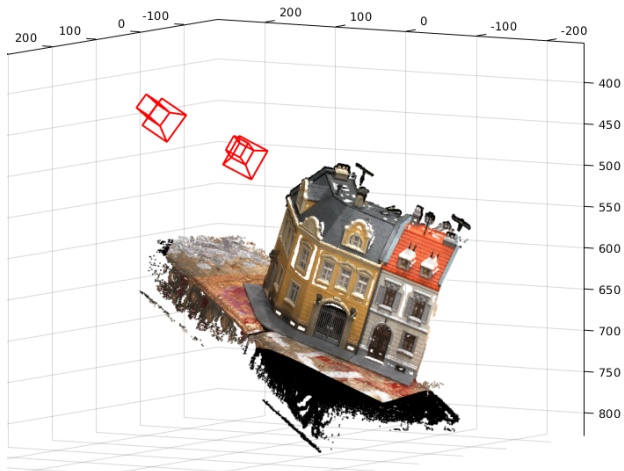


Figure 2: A 3D scene represented by a cloud of points is viewed by two cameras.

Use the provided scene points, camera calibration matrix, and relative pose encoded by `.mat` files, project the 3D points to the two image planes. Keep the following things in mind:

1. Projected points located outside the image boundaries should not be considered.
2. Direct projections on an empty image plane might give you an image shown in Figure 3(a). To construct a nicer and a rather compact image, Figure 3(b), you may need to make use of MATLAB's functions `meshgrid` and `griddata` to properly form the image.

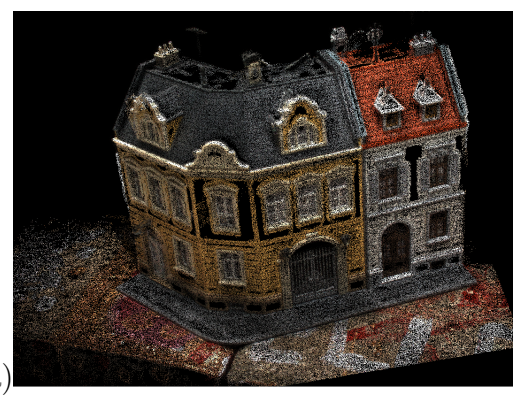
(a)  (b)

Figure 3: Examples of projecting 3D points directly on a blank image plane (a) and an image with nicer and compact formation (b).

Problem 2. Essential Matrix Constraint Let γ from an image I and $\bar{\gamma}$ from another image $\bar{I}$ be a pair of feature correspondence *in meters*, both represented by homogeneous coordinates. In the lecture, we learn that an essential matrix represents the relative pose between two cameras, and it satisfies $\bar{\gamma}^T E \gamma = 0$. The equation is often called the essential matrix constraint, or the *epipolar constraint*, meaning that given an essential matrix capturing the relative pose between two cameras, any pair of true correspondences in meters must follow that constraint.

Let $\gamma = [\xi \;\; \eta \;\; 1]^T$, $\bar{\gamma} = \begin{bmatrix}\bar{\xi} & \bar{\eta} & 1\end{bmatrix}^T$, the epipolar constraint is,

$$\begin{bmatrix}\bar{\xi} & \bar{\eta} & 1\end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} \xi \\ \eta \\ 1 \end{bmatrix} = 0. \tag{5}$$

Expand the expression, we have, for each $(\bar{\xi}, \bar{\eta})$,

$$A\bar{\xi} + B\bar{\eta} + C = 0, \tag{6}$$

where the coefficients A, B, and C are,

$$\begin{aligned} A &= e_{11}\xi + e_{12}\eta + e_{13} \\ B &= e_{21}\xi + e_{22}\eta + e_{23} \\ C &= e_{31}\xi + e_{32}\eta + e_{33}. \end{aligned}$$

Equation 6 is a line equation, meaning that given an essential matrix E, for each γ on image I, its correspondence $\bar{\gamma}$ must lie on a line on image $\bar{I}$. This line is often called an *epipolar line*. Figure 4 is an example, where the cyan squares are a pair of SIFT feature correspondence. The yellow line on the right image is the epipolar line, passing through the SIFT feature on the right image.

In this problem, you are asked to draw epipolar lines on an image pairs given two essential matrices: one from image 1 to image 2 and the other from image 2 to image 1. The camera intrinsic matrix is also provided as a `.mat` file. The necessary steps to follow are:

1. **Feature Detection:** Use VLFeat to detect SIFT keypoints and extract SIFT descriptors from two pairs of images in Figure 5.

2. **Find Feature Correspondences:** Find SIFT matches using the feature correspondences code you implemented in lab05. Rank order the matches by the SIFT descriptor distance in an ascending order. This gives you a rank-ordered list.

3. **Draw Epipolar Lines:** Pick top 20 from the rank-ordered list, visualize epipolar lines from the two images using the provided essential matrices in `.mat` files. You should also plot the feature points clearly to see whether they are close to the epipolar line. Note that it is okay if there are mismatches.

4. **Compute Reprojection Errors:** Because the SIFT feature locations are often noisy, a match point γ_{im} on an image plane does not necessarily pass through an epipolar line. The shortest distance from γ_{im} to the corresponding epipolar line is often called the *reprojection error*. Compute reprojection errors for **all** SIFT matches. Let 2 pixels be a threshold that distinguishes all matches from inliers to outliers, *i.e.* matches with reprojection errors less than 2 pixels on both images are inliers; otherwise they are outliers. What is the outlier ratio in the image pair?
(Please search on the internet if you forget the equation of the shortest distance from a point to a line.)

Figure 4: An example of an epipolar line.

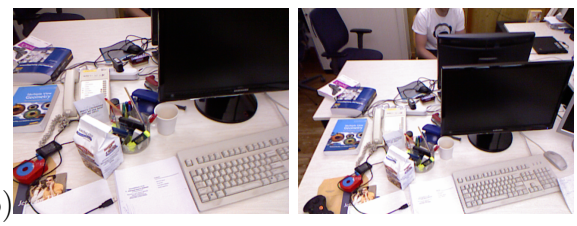
(a) (b) 

Figure 5

Problem 3. Compute Essential Matrices under a RANSAC Given two images, Figure 6, we are interested in finding the relative pose between these two images. This can be achieved by first estimating an essential matrix of image 2 with respect to image 1 in a RANSAC scheme, followed by recovering the relative rotation matrix and translation vector. The camera intrinsic matrix is provided in a `.mat` file. The necessary steps to follow are:

Figure 6

1. **Extract and Match SIFT Features:** Use VLFeat to extract and find SIFT feature correspondences. This is the same as you did in Problem 2.

2. **Estimate an Essential Matrix in a RANSAC loop:** Create the following function that returns an essential matrix `finalE` supported by the maximal number of inliers and the indices of inliers `inlierIndx`:

```
function finalE = Ransac4Essential(gamma1, gamma2, K)
```

   The inputs `gamma1` and `gamma2` are the feature correspondences, and `K` is the camera intrinsic matrix. You have the freedom to change the input names, but do not add additional input arguments.

   (a) In the lecture, we say that to estimate an essential matrix, at least 5 pairs of correspondences are necessary. Randomly pick 5 pairs of matches and feed them into the provided function file `fivePointAlgorithmSelf.m`. Follow the instruction in that file to properly structure your input.

   (b) The function `fivePointAlgorithmSelf.m` returns all (probably four or six) possible, valid essential matrices structured in a cell array. For each valid essential matrix, calculate the coefficients of the epipolar line as you did in Problem 2.

   (c) Using the epipolar line, count the number of inliers among all feature correspondences. A pair of correspondence is an inlier if the shortest distance between $\bar{\gamma}$ and the epipolar line is smaller than some threshold, *e.g.*, 2 pixels. Notation is borrowed from Problem 2.

   (d) Repeat steps (a) to (c) under an iterative loop with a predefined number of iterations. Set a large number of iterations to get an accurate essential matrix.

   (e) Finally, after the iterative loop ends, pick the essential matrix supported by the maximum number of inliers as your output `finalE` (final essential matrix).

3. **Recover R and T from E:** Decompose the essential matrix `finalE` returned from your `Ransac4Essential` function into relative rotation matrices R and relative translation vectors T. There are four sets of possible R and T. Refer to the Appendix for more information on how to do find R and T from an essential matrix.

4. **Evaluate Your Recovered R and T:** One out of the four R and T found by the previous step is valid. We will talk about how we can find the valid relative pose in our next lecture. In this lab, however, we can evaluate the recovered R and T using the ground truth relative pose which is provided in the `GT_pose.mat` file to find the valid set. By computing the errors between the recovered poses and the ground truth, the pose corresponding to the smallest error for both the rotation and the translation is the valid one. Let R_g and T_g be the ground truth rotation and translation, while R and T be the recovered pose.

(a) In the lecture, we say that a rotation matrix R has a property that $R^T R = I$. The error of your recovered R with respect to the ground truth can thus be

$$\Delta R\left(R_g, R\right) = \arccos\left(\frac{trace\left(R_g^T R\right) - 1}{2}\right). \tag{7}$$

(b) The recovered translation T is always a unit vector; thus, the error of your recovered T with respect to the ground truth can be

$$\Delta T\left(T_g, T\right) = \left|\langle T_g, T\rangle - 1\right|, \tag{8}$$

where $\langle T_g, T\rangle$ is the dot product of T_g and T.

Find the valid relative pose. What are the errors of the rotation and the translation?

Problem 4. Understanding the Variability of Poses (This part is mandatory for ENGN2605 students but optional for ENGN1610 students who will get 5 extra points if this problem is finished correctly.)

Given a set of noisy true correspondences and a camera calibration matrix, this question requests you to observe the variability of relative poses computed from essential matrix estimations. Follow the steps below:

1. **Load Noisy True Correspondences:** The correspondences can be found in the provided `Two_View_Correspondences.mat` file, where 62 point matches (ξ_{im}, η_{im}) in pixels are given.

2. **Pick 5 Random Matches:** Randomly pick 5 noisy matches, feed them to the function `fivePointAlgorithmSelf.m` to get all valid essential matrices. Keep the only one essential matrix supported by the maximal number of matches from the entire 62 point pairs. Note that this step is a bit different from the `Ransac4Essential` function: we have no outlier correspondences here, and thus we are not doing RANSAC. Just keep the essential matrix and we will use it in the next step.

3. **Recover R and T from E:** This step is identical to step 3 in Problem 3 which gives you R_1, R_2, T_1 and T_2.

4. **Collect and Cluster the Recovered R and T:** Repeat steps 2 to 3 at least N times to collect the recovered R_1, R_2, T_1 and T_2. Set N at least 40. Feed the collected rotations and translations into the provided function `getVeridicalGroupsOfPoses.m` to get a group of veridical rotation matrices and another group of veridical translation vectors. Refer to the code descriptions in the provided function to properly structure your data.

5. **Compute Pairwise Differences:** For the group of rotation matrices and the group of translation vectors returned by the `getVeridicalGroupsOfPoses.m` function, compute *pairwise* differences for each group. The difference between two rotation matrices R and $\overline{R}$ is $\Delta R\left(\overline{R}, R\right)$, Equation 7, while the difference between two translation vectors T and $\overline{T}$ is

$\Delta T\left(\overline{T}, T\right)$, Equation 8. N relative rotation/translation gives you $\binom{N}{2}$ pairwise differences. You are allowed to use `nchoosek` built-in function to generate all combinations of pairwise poses.

In the end, you should get two pools of pairwise differences from groups of rotations and translations. Plot the distribution for each pool. How does the variability look like?

Appendix: Recover R and T from E

1. **Method 1: Singular Value Decomposition (SVD)**
   Decompose an essential matrix E by SVD, *i.e.*, `[U,S,V]=svd(E)`, and define the matrix W as
   $$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{9}$$
   then two rotations are
   $$R_1 = UWV^T$$
   $$R_2 = UW^TV^T,$$
   and two translations are the last column of U with two different signs. Two translations and two rotations compose four set of solutions.
   Note that the rotation matrices R_1 and R_2 might be *improper*, *i.e.*, $det\,(R_1) = -1$ and $det\,(R_2) = -1$. In this case, simply reverse the sign of the essential matrix and redo the whole process again. Make sure to check the determinant of R_1 and R_2 to get *proper* rotation matrices.

2. **Method 2: Co-factor Matrix**
   From the constraint of an essential matrix,
   $$TT^T = \frac{1}{2}trace(EE^T)I - EE^T, \tag{10}$$
   the translation can be obtained by taking any row from the left hand side of the equation and normalize it, *i.e.*,
   $$\hat{T} = \frac{T}{||T||}. \tag{11}$$
   The two translations are $\hat{T}$ and $-\hat{T}$. The rotations can be subsequently computed from
   $$R = \frac{1}{||T||^2}cofactor(E)^T \pm \frac{1}{||T||^2}[T]_\times E, \tag{12}$$
   where the sign $\pm$ gives you two rotations. Two translations and two rotations compose four set of solutions.