ENGN2560 Computer Vision

Lab02: Feature Correspondences, Camera Relative Pose, and RANSAC

The goal of this lab is to:

- Camera calibration from a cube of checkerboard.
- Explore image feature correspondences.
- Learn how to estimate camera relative pose from two images under a RANSAC scheme.
- Learn how to evaluate an estimated relative camera pose.

Problem 1. Camera Calibration from a Cube of Checkerboard In our previous lab, a sheet of checkerboard renders the calibration matrix unsolvable. This is due to metric ambiguity, see reading materials for more information. One solution to such an issue is to use a *cube* of checkerboard, Figure 1. In this lab, we aim to set the calibration problem right by providing a list of 3D world point locations from a cube of checkerboard and their corresponding 2D points on an image. Follow the steps below.

1. Solve the camera intrinsic and extrinsic matrix as you did in the previous lab. In the end, you will get one intrinsic matrix and four sets of possible solutions for rotation and translation.
2. To find the veridical rotation $\mathcal{R}$ and translation $\mathcal{T}$, project the 3D points to 2D image. The set $(\mathcal{R}, \mathcal{T})$ is a veridical camera extrinsic matrix if the *reprojection error* is the minimal, *i.e.*, the sum of distances between the projected points and the 2D points is the least.

Code in `P1_main.m` and report your calibration results.

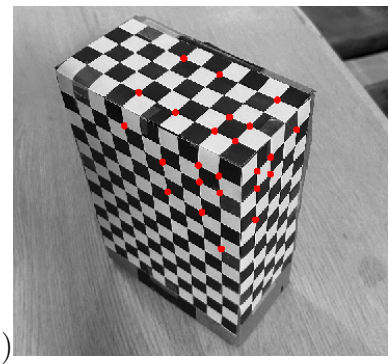
(a)  (b) 

Figure 1: (a) A cube of checkerboard can be effectively used for solving the camera calibration problem from (b) a set of 3D points (red) and their corresponding 2D points.

Problem 2. Feature Correspondences and Rank-Ordered List In this problem, you will continue to explore image features from the previous lab to finding correspondences of SIFT features across two views. Before finding feature matches, each feature is first encoded by a *descriptor* which provides information of that feature, *e.g.*, feature orientation, scale, *etc.* The difference between two descriptors tells how similar these two features are. SIFT feature descriptor is a 128 dimension vector (You can confirm this by the second returned value of `vl_sift` function), and the similarity of SIFT features is the euclidean distance of the two 128 dimension vectors.

Use VLFeat `vl_ubcmatch` function [1] to match SIFT features of an image pair, Figure 2. Use the default parameters for `vl_sift` and `vl_ubcmatch`. Construct a rank-ordered list of feature correspondences. Superimpose the top 30 feature correspondences on the image pair using the provided function `draw_Feature_Matches`. A `P2_main.m` file is provided. Show your results in the report.

Figure 2: A pair of images for SIFT feature correspondences.

Problem 3. Measure Outlier Ratio from Known Relative Camera Pose How to evaluate the quality of your feature correspondences, namely, measure the outlier ratio? From the lecture, we know that the relation of a veridical point correspondence can be captured by the epipolar constraint. Thus, an inlier feature correspondence must have small distance from points to the epipolar line. In this problem, we assume that the intrinsic matrix, the relative rotation $\mathcal{R}$, and the relative translation $\mathcal{T}$ are known, and measure the outlier ratio from the SIFT feature correspondences you had in Problem 1. The known intrinsic/extrinsic matrices are loaded in the `P3_main.m` file. Denote two images as image 1 and 2, respectively. Follow the steps below:

1. Calculate an essential matrix E from $\mathcal{R}$ and $\mathcal{T}$. Since E lives in the metric space, a tip is to calculate the fundamental matrix $F = (K^{-1})^T EK^{-1}$ which lives in the pixel space.

2. For each SIFT feature on image 1, find the epipolar line coefficients from F.

3. Calculate the shortest distance from the SIFT features on image 2 to the epipolar line you found in the previous step.

4. If the distance is smaller than some threshold, *e.g.* 2 pixels, mark this feature correspondence as an inlier.

[1] https://www.vlfeat.org/matlab/vl_ubcmatch.html

5. Loop over all SIFT features on image 1, calculate the outlier ratio, *i.e.*, the number of outliers over the total number of feature correspondences.

Report the outlier ratio for two pairs of images in Figure 2. Pick three inliers and plot epipolar lines. An example result is given in Figure 3.

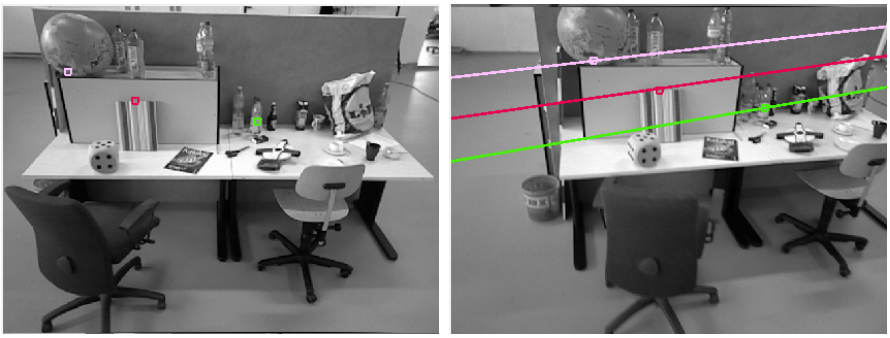


Figure 3: Three example SIFT features on the left image constraint their corresponding SIFT features on the right image by respective epipolar line.

Problem 4. Estimate Relative Pose Under a RANSAC Scheme In this problem, assume that we are given only two images in Figure 2. The goal is to estimate the relative pose ($\mathcal{R}$ and $\mathcal{T}$) under the RANSAC scheme from feature correspondences, with known camera calibration matrix. Code in the provided `P4_main.m` file by following the steps below.

1. **SIFT Feature Detection and Matching:** Use VLFeat to detect and find SIFT feature correspondences. This is the same as you did in Problem 1. Remember to construct a rank-ordered list.

2. **Estimate an Essential Matrix** E Create the following function that returns an essential matrix `E` from RANSAC and a list of inlier indices:

   ```
   function [E,inlier_Idx] = Ransac4Essential(PARAMS,gamma1,gamma2,K)
   ```

   The input `PARAMS` is a structure of all parameters passed to RANSAC, `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 from the top n rank-ordered list and feed them into the provided function `fivePointAlgorithmSelf.m`. Follow the instruction in that file to properly structure your input. It is suggested that n be the top 80% of the entire rnak-ordered list you constructed in Problem 2.

   (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 the previous problem.

(c) Using the epipolar line, count the number of inliers among all feature correspondences. You have done this in the previous problem. The distance threshold from points to the corresponding epipolar lines is defined by `PARAMS.INLIER_THRESH`.

(d) Repeat steps (a) to (c) under an iterative loop with a large number of iterations defined by `PARAMS.RANSAC_ITERATIONS`.

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

3. **Recover R and T from E:** Decompose E returned by 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. **Find Veridical R and T:** One out of the four R and T found by the previous step is veridical. To find the true set, loop over all inlier matches (matches indexed by the indices returned by your `Ransac4Essential` function), and compute their respective depths ρ_1 and ρ_2 in image 1 and 2. Return R and T for which the maximal number of inliers have positive depths ρ_1 and ρ_2. Please refer to the Appendix for more information on how to find the depths.

   Report what your veridical estimated $\mathcal{R}$ and $\mathcal{T}$ are.

Problem 5. Evaluate Relative Pose Estimation How good is your estimated relative camera pose from the previous problem? In this problem, we aim to evaluate its accuracy using the ground truth given by Problem 2. Let $\mathcal{R}_g$ and $\mathcal{T}_g$ be the ground truth rotation and translation, while $\mathcal{R}$ and $\mathcal{T}$ be the estimated pose.

1. A rotation matrix $\mathcal{R}$ has a special property that $\mathcal{R}^T\mathcal{R} = I$. Thus, the error of your estimated R with respect to the ground truth can be measured by

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

2. The estimated translation $\mathcal{T}$ is up to a scale due to metric ambiguity. Normalize both the ground truth and your estimated $\mathcal{T}$ so that their lengths are 1. Then, the error of your estimated $\mathcal{T}$ with respect to the ground truth can be

$$\Delta\mathcal{T}\left(\mathcal{T}_g, \mathcal{T}\right) = \left|\langle\mathcal{T}_g, \mathcal{T}\rangle - 1\right|, \tag{2}$$

   where $\langle\mathcal{T}_g, \mathcal{T}\rangle$ is the dot product of $\mathcal{T}_g$ and $\mathcal{T}$.

Code in the provided `P5_main.m` and report the errors of your estimated $\mathcal{R}$ and $\mathcal{T}$.

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

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

   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,

$$TT^T = \frac{1}{2} trace(EE^T)I - EE^T, \tag{4}$$

   the translation can be obtained by taking any row from the left hand side of the equation and normalize it, *i.e.*,

$$\hat{T} = \frac{T}{||T||}. \tag{5}$$

   The two translations are $\hat{T}$ and $-\hat{T}$. The rotations can be subsequently computed from

$$R = \frac{1}{||T||^2} cofactor(E)^T \pm \frac{1}{||T||^2} [T]_\times E, \tag{6}$$

   where the sign $\pm$ gives you two rotations. Two translations and two rotations compose four set of solutions.

Appendix: Depths Computation

Let γ_1 and γ_2 be a pair of correspondences in meters, and their corresponding 3D points which project to γ_1 and γ_2 be Γ_1 and Γ_2 under the first and second camera coordinate, respectively. The relative geometry between two cameras can be captured by a rotation matrix R and a translation vector T such that a 3D point whose representation in two cameras is related as

$$\Gamma_2 = R\Gamma_1 + T, \tag{7}$$

or

$$\rho_2\gamma_2 = R\rho_1\gamma_1 + T, \tag{8}$$

where ρ_1 and ρ_2 are the depths of γ_1 and γ_2. Given the correspondence γ_1 and γ_2, the goal is to find the depths ρ_1 and ρ_2.

1. **Method 1: Singular Value Decomposition (SVD)**
   Rewriting Equation 8 in a form of a linear system,

   $$\begin{bmatrix} -R\gamma_1 & \gamma_2 \end{bmatrix} \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} = T, \tag{9}$$

   we can solve ρ_1 and ρ_2 given R and T. Note that $\begin{bmatrix} -R\gamma_1 & \gamma_2 \end{bmatrix}$ is a 3×2 non-square matrix, to solve Equation 9 a pseudo-inverse method or a SVD method can be used.

2. **Method 2: Analytic Expressions for ρ_1 and ρ_2**
   From Equation (8), by inner product with $e_3^T = [0 \;\; 0 \;\; 1]^T$ gives

   $$\rho_2 = \rho_1 e_3^T \mathcal{R}\gamma_1 + e_3^T \mathcal{T}. \tag{10}$$

   Substituting this back into Equation (8) gives

   $$\left(\rho_1 e_3^T \mathcal{R}\gamma_1 + e_3^T \mathcal{T}\right)\gamma_2 = \rho_1 \mathcal{R}\gamma_1 + \mathcal{T}. \tag{11}$$

   This is a vector equation to solve for ρ_1 with redundancy due to the epipolar constraint. Inner product with $e_1 = [1 \;\; 0 \;\; 0]$ gives

   $$\left(\rho_1 e_3^T \mathcal{R}\gamma_1 + e_3^T \mathcal{T}\right) e_1^T\gamma_2 = \rho_1 e_1^T \mathcal{R}\gamma_1 + e_1^T \mathcal{T}, \tag{12}$$

   which gives

   $$\rho_1 = \frac{e_1^T \mathcal{T} - \left(e_3^T \mathcal{T}\right)\left(e_1^T\gamma_2\right)}{\left(e_3^T \mathcal{R}\gamma_1\right)\left(e_1^T\gamma_2\right) - e_1^T \mathcal{R}\gamma_1}, \tag{13}$$

   and ρ_2 can be expressed in a similar way as

   $$\rho_2 = \frac{\left(e_1^T \mathcal{T}\right)\left(e_3^T \mathcal{R}\gamma_1\right) - \left(e_3^T \mathcal{T}\right)\left(e_1^T \mathcal{R}\gamma_1\right)}{\left(e_3^T \mathcal{R}\gamma_1\right)\left(e_1^T\gamma_2\right) - e_1^T \mathcal{R}\gamma_1}. \tag{14}$$