Lab 6 Report

Robotics Integration Group Project I

Yuwei ZHAO (23020036096)
Group #31 2025-12-24

Abstract

…

See Resources on github.com/RamessesN/Robotics_MIT.

1 Introduction

2 Procedure

2.1 Individual Work

2.1.1 Nister's 5-point Algorithm

Read the following paper.
[1] Nistér, David. "An efficient solution to the five-point relative pose problem." 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Vol. 2. 2003. link here.

Questions:

1. **Outline the main computational steps required to get the relative pose estimate (up to scale) in Nister's 5-point algorithm.**

   Step I — Extract Nullspace Extraction:

   Build a 5×9 matrix, which consists of an epipolar constraint of 5 pairs of points, and calculate the zero-space vector $\tilde{X}, \tilde{Y}, \tilde{Z}, \widetilde{W}$ of the matrix. This step usually utilizes QR decomposition or SVD to achieve.

   Step II — Expand cubic constraints:

   Using the properties of the essential matrix, namely the trace constraint $2EE^TE - \text{trace}(EE^T)E = 0$. Express the eseential matrix as a linear combination of null space vectors $E = xX + yY + zZ + \omega W$, and then expand these cubic constraints.

   Step III — Gauss-Jordan Elimination:

   Build a 9×20 matrix A, and implement "Gauss-Jordan Elimination" to it.

Step IV — Build & Solve Polynomial Formulation:

Expand two 4×4 determinant polynomial of matrixes B and C. Then get the 10^{th} degree polynomial about the variable z through elimination.

Step V — Root Extraction:

Solve for the real roots of this 10^{th}-degree polynomial. This can be done by the Sturm sequence or by the eigenvalue decomposition of the adjoint matrix.

Step VI — Recover Pose & Disambiguation:

For each real root, the corresponding essential matrix E is recovered, and then the rotation matrix R and translation vector t are decomposed. The points are triangulated using Cheirality constraint (i.e. points must be in front of the camera), and the only correct physical and geometric solution is selected from a maximum of 10 possible mathematical solutions.

2. **Does the 5-point algorithm exhibit any degeneracy? (degeneracy = special arrangements of the 3D points or the camera poses under which the algorithm fails)**

   Conclusion — In the main context discussed in the paper (i.e. compared to uncalibrated methods), the 5-point algorithm overcomes Planar Structure Degeneracy, which is one of the big advantages over 7 or 8-point algorithms.

   Explanation:

   1. **Non-degradation of planar scenes:** Section 4 of the paper makes it clear that for uncalibrated methods, the algorithm fails when the scene points are coplanar (there is a continuous solution). However, under the calibrated 5-point method setting, coplanar points lead to at most 2-fold ambiguity, which can usually be resolved by a third view, or excluded by chiral constraints. Therefore, the paper concludes that the algorithm still works well in planar scenes.

   2. **General degradation:** While the paper focuses on its robustness in planar scenarios, according to the basics of epipolar geometry (and the implicit assumption in the paper), the essential matrix is not definable (baseline $t = 0$) without translational motion (pure rotation), at which point the algorithm degrades. In addition, multiple or infinite solutions also exist if 5 points and two optical centers lie on a specific critical surface such as twisted cubic, but it is considered robust for practical applications and planar scene comparisons highlighted in the paper.

3. **When used within RANSAC, what is the expected number of iterations the 5-point algorithm requires to find an outlier-free set?**

From the standard theory of RANSAC, we have the expected number of iterations N is determined by

$$N = \frac{\log(1-p)}{\log(1-\omega^s)}$$

which $\begin{cases} s\text{=5: The minimum number of points required by the 5-point algorithm, i.e. sample size} \\ \omega\text{(Inlier Ratio): Proportion of inliers (i.e. the probability of selecting an inlier point in a sample)} \\ p\text{(Confidence): The confidence that we want to find at least one full set of interior points} \end{cases}$

Since the term ω^s decreases exponentially with s, a smaller sample size s significantly reduces the required number of iterations. For the 5-point algorithm ($s = 5$), the probability of selecting an outlier-free set is ω^5, which is much higher than for the 7-point ($s = 7$) or 8-point ($s = 8$) algorithms. This reduction is critical for real-time structure and motion estimation.

E.g. Assuming a typical inlier ratio $\omega = 0.5$ and a desired confidence $p = 0.99$, the required iterations N are:

Method	**Sample Size (s)**	**Required Iterations (N)**
5-point algorithm	5	$\left\lceil \frac{\log(0.01)}{\log(1-0.5^5)} \right\rceil = 145$
7-point algorithm	7	$\left\lceil \frac{\log(0.01)}{\log(1-0.5^7)} \right\rceil = 588$
8-point algorithm	8	$\left\lceil \frac{\log(0.01)}{\log(1-0.5^8)} \right\rceil = 1177$

Additionally, if we consider the strictly theoretical mean number of trials $E[k]$ required to encounter the first outlier-free set (geometric distribution expectation), it is given by:

$$E[k] = \frac{1}{\omega^s}$$

For $\omega = 0.5$, the 5-point algorithm requires on average 32 trials, whereas the 8-point algorithm requires 256 trials. This efficiency allows the algorithm to be executed for hundreds of samples within a small time budget.

2.1.2 Designing a Minimal Solver

Can you do better than Nister? Nister's method is a minimal solver since it uses 5 point correspondences to compute the 5 degrees of freedom that define the relative pose (up to scale) between the two cameras (recall: each point induces a scalar equation). In the

presence of external information (e.g., data from other sensors), we may be able use less point correspondences to compute the relative pose.

Consider a drone flying in an unknown environment, and equipped with a camera and an Inertial Measurement Unit (IMU). We want to use the feature correspondences extracted in the images captured at two consecutive time instants t_1 and t_2 to estimate the relative pose (up to scale) between the pose at time t_1 and the pose at time t_2. Besides the camera, we can use the IMU (and in particular the gyroscopes in the IMU) to estimate the relative rotation between the pose of the camera at time t_1 and t_2.

You are required to solve the following problems:

1. **Assume the relative camera rotation between time and is known from the IMU. Design a minimal solver that computes the remaining degrees of freedom of the relative pose.**

   For the matching points p_1, p_2 on the normalized img plane, the epipolar constraint is

   $$p_2^T E p_1 = 0$$

   which essential matrix is $E = [t]_{\times} R$.

   Put E into the known rotation, we have

   $$p_2^T ([t]_{\times} R) p_1 = 0$$

   Let $p_1' = R p_1$, then we have $p_2^T [t]_{\times} p_1' = 0$ ①

   ∵ ① is equivalent to scalar triple product

   ∴ $t \cdot (p_1' \times p_2) = 0$, which means the translation vector t must be perpendicular to the vector $v = p_1' \times p_2$

   ∵ 1 point can only ensure t is on some plane

   ∴ we need 2 points to ensure the direction of t

   $$\begin{cases} \text{For } 1^{\text{st}} \text{ pair points: normal vector } v_1 = (R p_{1,a}) \times p_{2,a} \\ \text{For } 2^{\text{nd}} \text{ pair points: normal vector } v_2 = (R p_{1,b}) \times p_{2,b} \end{cases}$$

   ∵ translation vector t is perpendicular to v_1, v_2, thus

   $$t \sim v_1 \times v_2$$

   Then after normalizing, we have

   $$t \leftarrow \frac{t}{\|t\|}$$

2. **OPTIONAL (5 bonus pts): Describe the pseudo-code of a RANSAC algorithm using the minimal solver developed in point a) to compute the relative pose in presence of outliers (wrong correspondences).**

```
Algorithm: RANSAC for 2-point Relative Pose with Known Rotation

Input:
  Matches: A set of N feature correspondences S = {(p1_i, p2_i)}
  Rotation: Relative rotation matrix R (from IMU)
  Threshold: Inlier threshold epsilon (e.g., epipolar distance)
  Confidence: Desired confidence probability p (e.g., 0.99)
  InlierRatio: Estimated ratio of inliers w (e.g., 0.5)

Output:
  Best_t: The estimated translation direction
  Best_Inlier_Set: The set of consistent matches

1. Max_Iterations = log(1 - p) / log(1 - w^2)
2. Best_Score = 0
3. Best_t = [0, 0, 0]

4. For k = 1 to Max_Iterations:

    a. // Sampling
      Select 2 random correspondences {(p1_a, p2_a), (p1_b, p2_b)} from S.

    b. // Model Generation
      p1_prime_a = R * p1_a
      p1_prime_b = R * p1_b
      v1 = cross_product(p1_prime_a, p2_a)
      v2 = cross_product(p1_prime_b, p2_b)
      t_candidate = cross_product(v1, v2)
      Normalize t_candidate

    c. // Scoring (Count Inliers)
      Current_Score = 0
      Current_Inliers = {}
      E_candidate = [t_candidate]_cross * R  // Essential Matrix

      For each match (p1_i, p2_i) in S:
          // Calculate Sampson error or simple algebraic error
          error = abs( p2_i^T * E_candidate * p1_i )

          If error < epsilon:
              Current_Score = Current_Score + 1
              Add (p1_i, p2_i) to Current_Inliers

    d. // Update Best Model
      If Current_Score > Best_Score:
          Best_Score = Current_Score
```

```
            Best_t = t_candidate
            Best_Inlier_Set = Current_Inliers

5. // Refinement
   Re-estimate t using all points in Best_Inlier_Set via Linear Least
   Squares or SVD.

6. Return Best_t
```

which

- Sample Size — It merely needs to sample 2 points, as opposed to 5 for the Nister algorithm, which means that the number of cycles of RANSAC will be greatly reduced ($N \propto \omega^{-2}$ vs $N \propto \omega^{-5}$), which greatly improves the computation speed.
- Model Validation — epipolar geometric error $p_2^T E p_1$ is used for validation for R is known and t is assumed so that E is ensured.

2.2 Team Work

3 Reflection and Analysis

4 Conclusion

5 Source Code

•
•