
Luca Carlone

Lecture 15: RANSAC and 3D-3D correspondences
Today

• Recap on 2-view

• RANSAC

• 3D-3D correspondences


The essential matrix encodes relative pose (up to scale) between $C_1$ and $C_2$.
2-view Geometry

Last week’s assumptions:
- no wrong correspondences (outliers)
- 3D point is not moving
- camera calibration is known
Estimating Poses from Correspondences

Given $N$ calibrated pixel correspondences:

$$(\tilde{y}_{1,k}, \tilde{y}_{2,k}) \text{ for } k = 1, \ldots, N$$

1. leverage the epipolar constraints to estimate the essential matrix $E$

   \[
   \tilde{y}_{2,k}^T E \tilde{y}_{1,k} = 0
   \]

   For 8 points: $A e = 0$  
   N>8 points: $\arg \min_{\|e\|_1} \|A e\|^2$

2. Retrieve the rotation and translation (up to scale) from the $E$

   \[
   E = [t] \times R
   \]
2-view Geometry

In practice:
- Many wrong correspondences (outliers)
- Some 3D points might be moving
RANSAC

**Problem:** estimate model $P$ from $N$ data points, possibly corrupted with outliers.

**Assume:** we have an algorithm to estimate $P$ from $n$ data points ($n << N$)

**Basic idea:**
1. sample $n$ points
2. compute an estimate $P'$ of $P$
3. count how many other points agree with $P'$
4. repeat until you get a $P'$ that agrees with many points
RANSAC

Random Sample Consensus

**Problem:** estimate model $P$ from $N$ data points, possibly corrupted with outliers.

**Assume:** we have an algorithm to estimate $P$ from $n$ data points ($n << N$)

**Basic idea:**
1. sample $n$ points
2. compute an estimate $P'$ of $P$
RANSAC

**RANdom SAmple Consensus**

**Problem:** estimate model $P$ from $N$ data points, possibly corrupted with outliers.

**Assume:** we have an algorithm to estimate $P$ from $n$ data points ($n \ll N$)

**Basic idea:**
1. sample $n$ points
2. compute an estimate $P'$ of $P$
RANSAC

**RAN**dom **SA**mple **C**onsensus

**Problem:** estimate model $P$ from $N$ data points, possibly corrupted with outliers.

**Assume:** we have an algorithm to estimate $P$ from $n$ data points $(n << N)$

**Basic idea:**
1. sample $n$ points
2. compute an estimate $P'$ of $P$
RANSAC

**Problem:** estimate model $P$ from $N$ data points, possibly corrupted with outliers.

**Assume:** we have an algorithm to estimate $P$ from $n$ data points ($n << N$)

**Basic idea:**
1. sample $n$ points
2. compute an estimate $P'$ of $P$
3. count how many other points agree with $P'$
4. repeat until you get a $P'$ that agrees with many points
RANSAC

RANdom SAmple Consensus

Problem: estimate model $P$ from $N$ data points, possibly corrupted with outliers.

Assume: we have an algorithm to estimate $P$ from $n$ data points ($n << N$)

Basic idea:
1. sample $n$ points
2. compute an estimate $P'$ of $P$
RANSAC

**Random Sample Consensus**

**Problem:** estimate model $P$ from $N$ data points, possibly corrupted with outliers.

**Assume:** we have an algorithm to estimate $P$ from $n$ data points ($n << N$)

**Basic idea:**
1. sample $n$ points
2. compute an estimate $P'$ of $P$
RANSAC

Random Sample Consensus

**Problem:** estimate model $P$ from $N$ data points, possibly corrupted with outliers.

**Assume:** we have an algorithm to estimate $P$ from $n$ data points ($n << N$)

**Basic idea:**
1. sample $n$ points
2. compute an estimate $P'$ of $P$
Example: Linear Regression

Fit a line through N 2D points, possibly corrupted with outliers.

**Note:** we have an algorithm to estimate a line from n=2 points

**RANSAC:**
1. sample 2 points
2. compute a line estimate $P'$ of $P$
3. count how many points are within a **tolerance** from $P'$
4. repeat until you get a $P'$ that agrees with many points
Example: Linear Regression

Fit a line through N 2D points, possibly corrupted with outliers.

**Note:** we have an algorithm to estimate a line from n=2 points

**RANSAC:**
1. sample 2 points
2. compute a line estimate $P'$ of $P$
3. count how many points are within a **tolerance** from $P'$
4. repeat until you get a $P'$ that agrees with many points
Example: Linear Regression

Fit a line through N 2D points, possibly corrupted with outliers.

**Note:** we have an algorithm to estimate a line from n=2 points.

**RANSAC:**
1. sample 2 points
2. compute a line estimate $P'$ of $P$
3. count how many points are within a **tolerance** from $P'$
4. repeat until you get a $P'$ that agrees with many points.
Example: Linear Regression

Fit a line through N 2D points, possibly corrupted with outliers.

**Note:** we have an algorithm to estimate a line from n=2 points

**RANSAC:**
1. sample 2 points
2. compute a line estimate $P'$ of $P$
3. count how many points are within a **tolerance** from $P'$
4. repeat until you get a $P'$ that agrees with many points
Example: Linear Regression

Fit a line through N 2D points, possibly corrupted with outliers.

**Note:** we have an algorithm to estimate a line from n=2 points

**RANSAC:**
1. sample 2 points
2. compute a line estimate $P'$ of $P$
3. count how many points are within a **tolerance** from $P'$
4. repeat until you get a $P'$ that agrees with many points
Example: Linear Regression

Fit a line through N 2D points, possibly corrupted with outliers.

**Note:** we have an algorithm to estimate a line from n=2 points

**RANSAC:**

1. sample 2 points
2. compute a line estimate $P'$ of $P$
3. count how many points are within a tolerance from $P'$
4. repeat until you get a $P'$ that agrees with many points
Example: Linear Regression

Fit a line through N 2D points, possibly corrupted with outliers.

**Note:** we have an algorithm to estimate a line from n=2 points

**RANSAC:**
1. sample 2 points
2. compute a line estimate $P'$ of $P$
3. count how many points are within a **tolerance** from $P'$
4. repeat until you get a $P'$ that agrees with many points
Example: Linear Regression

Fit a line through N 2D points, possibly corrupted with outliers.

**Note:** we have an algorithm to estimate a line from n=2 points

**RANSAC:**
1. sample 2 points
2. compute a line estimate $P'$ of $P$
3. count how many points are within a **tolerance** from $P'$
4. repeat until you get a $P'$ that agrees with many points
RANSAC: Parameter Tuning

1. **Error Tolerance** $\epsilon$: depends on the noise

2. **Acceptable consensus set:**
   - from the paper: $n+5$
   - rule of thumb: >50% of points

3. **Maximum number of iterations**
Example: RANSAC for Essential Matrix estimation

**RANSAC:**

1. sample $n$ point correspondences
2. compute an estimate $E'$ of the essential matrix $E$
3. count how many points are within a **tolerance** from $E'$
4. repeat until you get a $E'$ that agrees with many points
Example: RANSAC for Essential Matrix estimation

RANSAC
- essentially selects the set of inliers
- provides **geometric verification** for the correspondences
Beyond Motion Estimation

The tools we discussed (feature matching, essential matrix estimation, RANSAC) can be used also for **object detection** and **localization**
3D-3D Point Correspondences

Structured Light Cameras

RGB-D cameras can measure depth (D) and image (RBG)

How can we use the depth information to estimate the relative pose between two RGB-D cameras observing the same scene?
3D-3D Point Correspondences

1. We can use camera images to establish 2D-2D correspondences:
\[(\tilde{y}_{1,k}, \tilde{y}_{2,k}) \text{ for } k = 1, \ldots, N\]

2. For each camera we can compute the set of 3D points corresponding to pixels
\[(p_{1,k}, p_{2,k}) \text{ for } k = 1, \ldots, N\]
How to estimate the relative pose between the cameras from 3D-3D correspondences $(p_{1,k}, p_{2,k})$ with $k = 1, \ldots, N$?
Few More Comments:

**3 points** are sufficient to compute the relative pose from 3D-3D correspondences.

We can use the solver seen today as a 3-point minimal solver within a **RANSAC** method.

Also useful for 3D objects localization:

Other names: vector registration, point cloud alignment, ..
Backup
Other Matrices in 2-view Geometry

Homography matrix $H$

$$\lambda_2 \mathbf{x}_2 = H \lambda_1 \mathbf{x}_1$$

Fundamental matrix $F$

$$F = K_2^{-\top} \begin{bmatrix} t \end{bmatrix} \times R \ K_1^{-1}$$

Section 5.3

Chapter 6
• A matrix is an essential matrix if and only if it has singular values \( \{\sigma, \sigma, 0\} \)

• The space of the essential matrices is called the **Essential space** \( S_E \) (i.e., the space of \( 3 \times 3 \) matrices that can be written as \([t]_\times R\) for some \( R \in SO(3) \) and \( t \in \mathbb{R}^3 \)). The projection of a matrix \( M \) onto the Essential space can be computed as prescribed in [1, Thm 5.9]:

\[
\arg\min_{E \in S_E} \|E - M\|^2_F = U \begin{bmatrix} \frac{\lambda_1 + \lambda_2}{2} & 0 & 0 \\ 0 & \frac{\lambda_1 + \lambda_2}{2} & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T
\]

where \( M = U \text{diag} (\lambda_1, \lambda_2, \lambda_3) V^T \) is a singular value decomposition of \( M \).