Essential Matrix Decomposition

\[ E = \begin{bmatrix} t \end{bmatrix} R = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \]

\[ U^T R = U \]

where \( R \in SO(3) \)

Define \( R = U W V^T \)

\[ E = \begin{bmatrix} t \end{bmatrix} R = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \]

\[ U^T U W V^T = U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} W V^T \]

\[ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} W \]

\[ W = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]
Camera Pose from Essential Matrix (Rotation)

\[ E = \begin{bmatrix} t \end{bmatrix} \]

\[ R = U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} U^T \]

where \( R \in SO(3) \)

\[ R = U \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T \]

or

\[ U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T \]
**Camera Pose Estimation**

\[ E = K^T F K \]

function \( E = \text{ComputeEssentialMatrix}(F, K) \)

\[
E = K' * F * K;
\]

\[
[u \ d \ v] = \text{svd}(E);
\]

\[
d(1,1) = 1;
d(2,2) = 1;
d(3,3) = 0;
\]

\[
E = u * d * v';
\]

\[
D = \begin{bmatrix}
1.0468 & 0 & 0 \\
0 & 0.9975 & 0 \\
0 & 0 & 0.0000
\end{bmatrix}, \quad D = \begin{bmatrix}
1.0000 & 0 & 0 \\
0 & 1.0000 & 0 \\
0 & 0 & 0.0000
\end{bmatrix}
\]

Before cleanup \hspace{1cm} After cleanup
function [R1 t1, R2, t2, R3, t3, R4, t4] = ...
CameraPoseFromEssentialMatrix(E)

[U D V] = svd(E);
W = [0 -1 0; 1 0 0; 0 0 1];
t1 = U(:,3);
R1 = U * W * V';
if det(R1) < 0
    t1 = -t1;
    R1 = -R1;
end

\[
\begin{bmatrix}
0 & -1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1 & 0 \\
-1 & 0 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

\[ t = \pm u_3 \]

Camera Pose Estimation
Camera Image Projection
Point Triangulation

× 2D correspondences
Point Triangulation

2D correspondences

3D point

\[ \mathbf{x} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \]

2D projection

\[ \mathbf{x}_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} \]

3D camera pose

\[ \mathbf{P}_1 \in \mathbb{R}^{3 \times 4} \]
Camera Projection Matrix

Camera projection of world point:

\[
\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & p_x \\ f & p_y \\ 1 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}
\]

\[
= \begin{bmatrix} f & p_x \\ f & p_y \\ 1 & 1 \end{bmatrix} \begin{bmatrix} r_{x1} & r_{x2} & r_{x3} & t_x \\ r_{y1} & r_{y2} & r_{y3} & t_y \\ r_{z1} & r_{z2} & r_{z3} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}
\]
Third person (world) perspective

$$C = -R^{-1}t$$

Camera center in world coordinate

$$P = K[R \mid t]$$

$$= K[R \mid -RC]$$

$$= KR[I_{3\times3} \mid -C]$$

Camera center seen from world coordinate system
Geometric Interpretation

Coordinate transformation from world to camera:

\[ X_c = ^c R_w X + ^c t \]

where \(^c t\) is translation from world to camera seen from camera.

Rotate and then, translate.

\[ X_c = ^c R_w (X - C) \]

where \(C\) is translation from world to camera seen from world.

\[ X_c = \begin{bmatrix} r_{x1} & r_{x2} & r_{x3} & t_x \\ r_{y1} & r_{y2} & r_{y3} & t_y \\ r_{z1} & r_{z2} & r_{z3} & t_z \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} \]

\[ X_c = \begin{bmatrix} r_{x1} & r_{x2} & r_{x3} & 0 \\ r_{y1} & r_{y2} & r_{y3} & 0 \\ r_{z1} & r_{z2} & r_{z3} & 0 \end{bmatrix} \begin{bmatrix} 1 & -C_x \\ 0 & 1 & -C_y \\ 0 & 0 & 1 & -C_z \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix} \]
Cross Product

Cross product with skew-symmetric matrix representation:

\[ \mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix} \]

function skew = Vec2Skew(v)
    skew = [0 -v(3) v(2);
            v(3) 0 -v(1);
            -v(2) v(1) 0];
Two 2D lines in an image intersect at a 2D point:

\[ a_1 u + b_1 v + c_1 = 0 \quad a_2 u + b_2 v + c_2 = 0 \]

\[ l_1^T x = 0 \quad l_2^T x = 0 \]

where

\[ x = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad l_1 = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} \quad l_2 = \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} \]

\[ \begin{bmatrix} l_1^T \\ l_2^T \end{bmatrix} x = 0 \]

\[ A \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0 \]

\[ = \text{null}(A) \]

or

\[ l = x_1 \times x_2 \]
function GetPointFromTwoLines
l1 = [-398;-752;1404124];
l2 = [310;-924;303790];
x = Vec2Skew(l1)*l2;
x = x/x(3)

x =

1779.0   925.6   1
similar to (1804,934)
Point Triangulation

\[
\lambda \begin{bmatrix} x_1 \\ 1 \end{bmatrix} = P_1 \begin{bmatrix} X \\ 1 \end{bmatrix}
\]

2D correspondences

3D point

\[
x = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}
\]

3D camera pose

\[P_1 \in \mathbb{R}^{3 \times 4}\]
Point Triangulation

2D correspondences

3D point

2D projection

3D camera pose

\[ \lambda \begin{bmatrix} x_1 \\ 1 \end{bmatrix} = P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ 1 \end{bmatrix} P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} = 0 \]

Cross product between two parallel vectors equals to zero.
Point Triangulation

\[ \lambda \begin{bmatrix} x_1 \\ 1 \end{bmatrix} = P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ 1 \end{bmatrix}_{P_1} \begin{bmatrix} X \\ 1 \end{bmatrix}_{P_2} = 0 \]

2D correspondences

3D point

2D projection

3D camera pose

\[ P_1 \in \mathbb{R}^{3 \times 4} \]

\[ P_2 \]
Point Triangulation

2D correspondences

3D point

2D projection

3D camera pose

$P_1 \in \mathbb{R}^{3 \times 4}$

$P_2$

$X = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$

$X = \begin{bmatrix} x_1 \\ 1 \end{bmatrix} = P_1 \begin{bmatrix} X \\ 1 \end{bmatrix}$

$\lambda \begin{bmatrix} x_1 \\ 1 \end{bmatrix} = P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ 1 \end{bmatrix} P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} = 0$

$\begin{bmatrix} x_2 \\ 1 \end{bmatrix} P_2 \begin{bmatrix} X \\ 1 \end{bmatrix} = 0$

$X = \begin{bmatrix} x_1 \\ 1 \\ x_2 \\ 1 \end{bmatrix}$

$\begin{bmatrix} x_1 \\ 1 \\ x_2 \\ 1 \end{bmatrix} P_1 \begin{bmatrix} X \\ 1 \\ X \\ 1 \end{bmatrix} = 0$
Point Triangulation

\[
\lambda \begin{bmatrix} x_1 \\ 1 \end{bmatrix} = P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} x_1 \\ 1 \end{bmatrix} P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} = 0
\]

2D correspondences

\[
\begin{align*}
x &= \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\
x_i &= \begin{bmatrix} u_i \\ v_i \end{bmatrix}
\end{align*}
\]

3D camera pose

\[
P_1, P_2, P_F \in \mathbb{R}^{3 \times 4}
\]

2D projection
Point Triangulation

\[
\lambda \begin{bmatrix} x_1 \\ 1 \end{bmatrix} = P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} x_1 \\ 1 \end{bmatrix} P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} = 0
\]

\[
\begin{bmatrix} x_1 \\ 1 \end{bmatrix} P_1 \begin{bmatrix} X \\ 1 \end{bmatrix} = 0
\]

\[
\begin{bmatrix} x_2 \\ 1 \end{bmatrix} P_2 \begin{bmatrix} X \\ 1 \end{bmatrix} = 0
\]

\[
\text{rank} \begin{bmatrix} x_1 & P_1 & X & 1 \end{bmatrix} = 2
\]

Least squares if \( F \geq 2 \)
Point Triangulation

\[ P_1 = K_1 \begin{bmatrix} I_{3 \times 3} & 0_3 \end{bmatrix} \]

\[ P_2 = K_2 \begin{bmatrix} I_{3 \times 3} & -C \end{bmatrix} \]

\[ C = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \]
% Intrinsic parameter
K1 = [2329.558 0 1141.452; 0 2329.558 927.052; 0 0 1];
K2 = [2329.558 0 1241.731; 0 2329.558 927.052; 0 0 1];

% Camera matrices
P1 = K1 * [eye(3) zeros(3,1)];
C = [1;0;0];
P2 = K2 * [eye(3) -C];

% Correspondences
x1 = [1382;986;1];
x2 = [1144;986;1];
skew1 = Vec2Skew(x1);
skew2 = Vec2Skew(x2);

% Solve
A = [skew1*P1; skew2*P2];
[u,d,v] = svd(A);
X = v(:,end)/v(end,end);

function skew = Vec2Skew(v)
skew = [0 -v(3) v(2); v(3) 0 -v(1); -v(2) v(1) 0];
Point Triangulation
3.1 Linear Triangulation

**Goal** Given two camera poses, \((C_1, R_1)\) and \((C_2, R_2)\), and correspondences \(x_1 \leftrightarrow x_2\), triangulate 3D points using linear least squares:

\[
X = \text{LinearTriangulation}(K, C_1, R_1, C_2, R_2, x_1, x_2)
\]

- **INPUT** \(C_1\) and \(R_1\): the first camera pose
- **INPUT** \(C_2\) and \(R_2\): the second camera pose
- **INPUT** \(x_1\) and \(x_2\): two \(N \times 2\) matrices whose row represents correspondence between the first and second images where \(N\) is the number of correspondences.
- **OUTPUT** \(X\): \(N \times 3\) matrix whose row represents 3D triangulated point.
Camera pose disambiguation via point triangulation

Four configurations:
3.2 Camera Pose Disambiguation

**Goal** Given four camera pose configuration and their triangulated points, find the unique camera pose by checking the *cheirality* condition—the reconstructed points must be in front of the cameras:

\[
[C \ R \ X_0] = \text{DisambiguateCameraPose}(\text{Cset}, \text{Rset}, \text{Xset})
\]

*(INPUT) Cset and Rset:* four configurations of camera centers and rotations  
*(INPUT) Xset:* four sets of triangulated points from four camera pose configurations  
*(OUTPUT) C and R:* the correct camera pose  
*(OUTPUT) X₀:* the 3D triangulated points from the correct camera pose

The sign of the \( Z \) element in the camera coordinate system indicates the location of the 3D point with respect to the camera, i.e., a 3D point \( X \) is in front of a camera if \((C, R)\) if \( r_3(X - C) > 0 \) where \( r_3 \) is the third row of \( R \). Not all triangulated points satisfy this condition due to the presence of correspondence noise. The best camera configuration, \((C, R, X)\) is the one that produces the maximum number of points satisfying the cheirality condition.
Alternative solution

"Just checking."
Ideally, $X$ is the point of intersection of two 3D rays:

- $P_l = \lambda_{Bob} K_{bob}^{-1} u$ measured in Bob’s frame
- $P_r^{alice} = \lambda_{alice} K^{-1} v$ measured in Alice’s frame
- $P_r = \lambda_{alice} R^T K^{-1} v$ measured in Bob’s frame

Alice’s camera location $T = -R^T t$
Alternative solution

Calibrated Triangulation

ideally, P is the point of intersection of two 3D rays:
  ray through $O_1$ with direction $P_1$
  ray through $O_r$ with direction $R^TP_r$
Triangulation with Noise

Unfortunately, these rays typically don’t intersect due to noise in point locations and calibration params
Triangulation with Noise

Unfortunately, these rays typically don’t intersect due to noise in point locations and calibration params

Solution: Choose P as the “pseudo-intersection point”. This is point that minimizes the sum of squared distance to both rays. (The SSD is 0 if the rays exactly intersect)
Solution from T&V Book

P is midpoint of the segment perpendicular to $P_1$ and $R^{TP_r}$

Let $w = P_1 \times R^{TP_r}$ (this is perpendicular to both)

Introducing three unknown scale factors $a, b, c$ we note we can write down the equation of a “circuit”
Solution from T&V Book

Writing vector “circuit diagram” with unknowns a,b,c

\[ a \mathbf{P}_1 + c (\mathbf{P}_1 \times \mathbf{R}_r^T \mathbf{P}_r) - b \mathbf{R}_r^T \mathbf{P}_r = \mathbf{T} \]

note: this is three linear equations in three unknowns a,b,c

=> can solve for a,b,c
Solution from T&V Book

After finding $a, b, c$, solve for midpoint of line segment between points $O_1 + a P_1$ and $O_1 + T + b R^T P_r$. 
How Many Correspondences?

What is minimum $m$?
Local Patch (Scale)
Local Patch (Scale)
Local Patch (Scale)
Local Visual Descriptor

Desired properties:
- Repeatability: the same point is repeatedly detected.
- Discriminativity: the point is unique.
Local Visual Descriptor

**Desired properties:**
- Repeatability: the same point is repeatedly detected.
- Discriminativeness: the point is unique.
- Orientation aware
Local Scale Invariant Feature Transform (SIFT)

**SIFT** automatically finds the optimal scale of feature point and its orientation.

**Desired properties:**
- Repeatability: the same point is repeatedly detected.
- Discriminativity: the point is unique.
- Orientation aware
Local Scale Invariant Feature Transform (SIFT)
Local Scale Invariant Feature Transform (SIFT)
Local Scale Invariant Feature Transform (SIFT)
Local Scale Invariant Feature Transform (SIFT)
Nearest Neighbor Search
Nearest Neighbor Search

**Discriminativity:** how is the feature point unique?

Feature match candidates
Nearest Neighbor Search w/ Ratio Test

**Discriminativity**: how is the feature point unique?

\[
\frac{d_1}{d_2} < 0.7
\]
Nearest Neighbor Search w/o Ratio Test

Left image ➔ right image
Nearest Neighbor Search w/ Ratio Test

Left image ➔ right image
Nearest Neighbor Search w/o Ratio Test

Left image ← right image
Nearest Neighbor Search w/ Ratio Test

Left image ↔ right image
Bi-directional Consistency Check

**Consistency**: would a feature match correspond to each other?
Bi-directional Consistency Check
RANSAC: Random Sample Consensus
Fundamental Matrix Computation: Linear Least Squares
Fundamental Matrix Computation: Linear Least Squares
Recall: Line Fitting \((Ax = b)\)
Outlier

Data
Ground truth
Least squares

\[
\begin{align*}
&u_1, 1 \\
&u_2, 1 \\
&\vdots \\
&u_n, 1
\end{align*}
\]

\[
\begin{align*}
v_1 \\
v_2 \\
\vdots \\
v_n
\end{align*}
\]

\[
A^T A X \approx A^T b
\]

\[
X = \left[ A^T A \right]^{-1} A^T b
\]
Line fitting error:

\[ E = (ex_1 - fy_1 - g)^2 + \cdots + (ex_N - fy_N - g)^2 \]

\[ = \sum_{i=1}^{N} (ex_i - fy_i - g)^2 \]

Perpendicular distance

\[ E_i = |ex_i + fy_i + g| \]

Quadratic magnification of error of outliers

Outlier
Outlier rejection strategy:
To find the best line that explains the maximum number of points.
Outlier rejection strategy:
To find the best line that explains the maximum number of points.

Assumptions:
1. Majority of good samples agree with the underlying model (good apples are same and simple.).
2. Bad samples does not consistently agree with a single model (all bad apples are different and complicated.).
RANSAC: Random Sample Consensus
RANSAC: Random Sample Consensus

1. Random sampling
RANSAC: Random Sample Consensus

1. Random sampling
2. Model building
1. Random sampling
2. Model building
3. Thresholding

RANSAC: Random Sample Consensus
RANSAC: Random Sample Consensus

1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

# of inliers: 7
RANSAC: Random Sample Consensus

1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting
RANSAC: Random Sample Consensus

1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting
RANSAC: Random Sample Consensus

1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting
1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

RANSAC: Random Sample Consensus

# of inliers: 10
RANSAC: Random Sample Consensus

1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting
1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

RANSAC: Random Sample Consensus
RANSAC: Random Sample Consensus

1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting
1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

# of inliers: 23
Maximum number of inliers

**RANSAC: Random Sample Consensus**
Required number of iterations with $p$ success rate:
Required number of iterations with $p$ success rate:

Probability of choosing an inlier: $w = \frac{\text{# of inliers}}{\text{# of samples}}$
Required number of iterations with $p$ success rate:

Probability of choosing an inlier: $w = \frac{\text{# of inliers}}{\text{# of samples}}$

Probability of building a correct model: $w^n$ where $n$ is the number of samples to build a model.
Required number of iterations with $p$ success rate:

Probability of choosing an inlier: $w = \frac{\# \text{ of inliers}}{\# \text{ of samples}}$

Probability of building a correct model: $w^n$ where $n$ is the number of samples to build a model.

Probability of not building a correct model during $k$ iterations: $(1 - w^n)^k$
Required number of iterations with $p$ success rate:

Probability of choosing an inlier: $w = \frac{\text{# of inliers}}{\text{# of samples}}$

Probability of building a correct model: $w^n$ where $n$ is the number of samples to build a model.

Probability of not building a correct model during $k$ iterations: $(1 - w^n)^k$

$$(1 - w^n)^k = 1 - p$$ where $p$ is desired RANSAC success rate.

$$k = \frac{\log(1 - p)}{\log(1 - w^n)}$$
Required number of iterations with $p$ success rate:

$$k = \frac{\log(1 - p)}{\log(1 - w^n)}$$

where $w = \frac{\# \text{ of inliers}}{\# \text{ of samples}}$

Probability of choosing an inlier:

$$w = \frac{\# \text{ of inliers}}{\# \text{ of samples}}$$

Probability of building a correct model: $w^n$ where $n$ is the number of samples to build a model.

Probability of not building a correct model during $k$ iterations:

$$(1 - w^n)^k = 1 - p$$

where $p$ is desired RANSAC success rate.

$$k = \frac{\log(1 - p)}{\log(1 - w^n)}$$
Fundamental Matrix Computation via RANSAC
Fundamental Matrix Computation via RANSAC
Fundamental Matrix

\[ v^T F u = 0 \]

Bob's image

Alice's image

\[ F = K^{-T} \begin{bmatrix} t \end{bmatrix} R K^{-1} \]
Fundamental Matrix Computation via RANSAC

Epipolar line: \[ l_u = Fu \]

Distance: \[ d = \frac{|au_x + bu_y + c|}{\sqrt{a^2 + b^2}} = \frac{|Fu|}{\sqrt{(F_{1,u})^2 + (F_{2,u})^2}} \]
Fundamental Matrix Computation via RANSAC

# of inliers: 65 out of 260
Fundamental Matrix Computation via RANSAC

# of inliers: 65 out of 260
Fundamental Matrix Computation via RANSAC

# of inliers: 186 out of 260
Four Camera Pose Config. From Essential Matrix