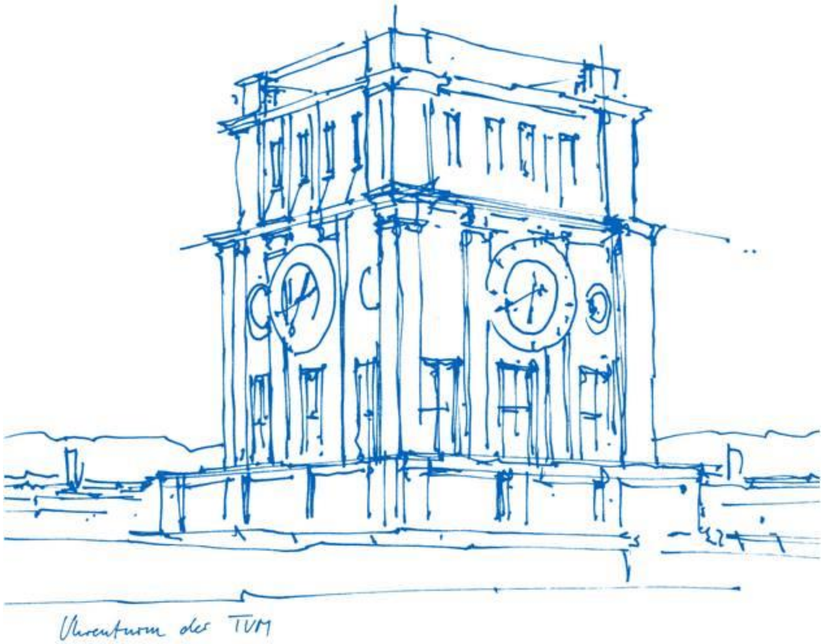
Modeling and Control of Legged Robots
SS 2024

L5: 3D Vision for Robotics



M. Sc. Hao Xing

Technical University of Munich

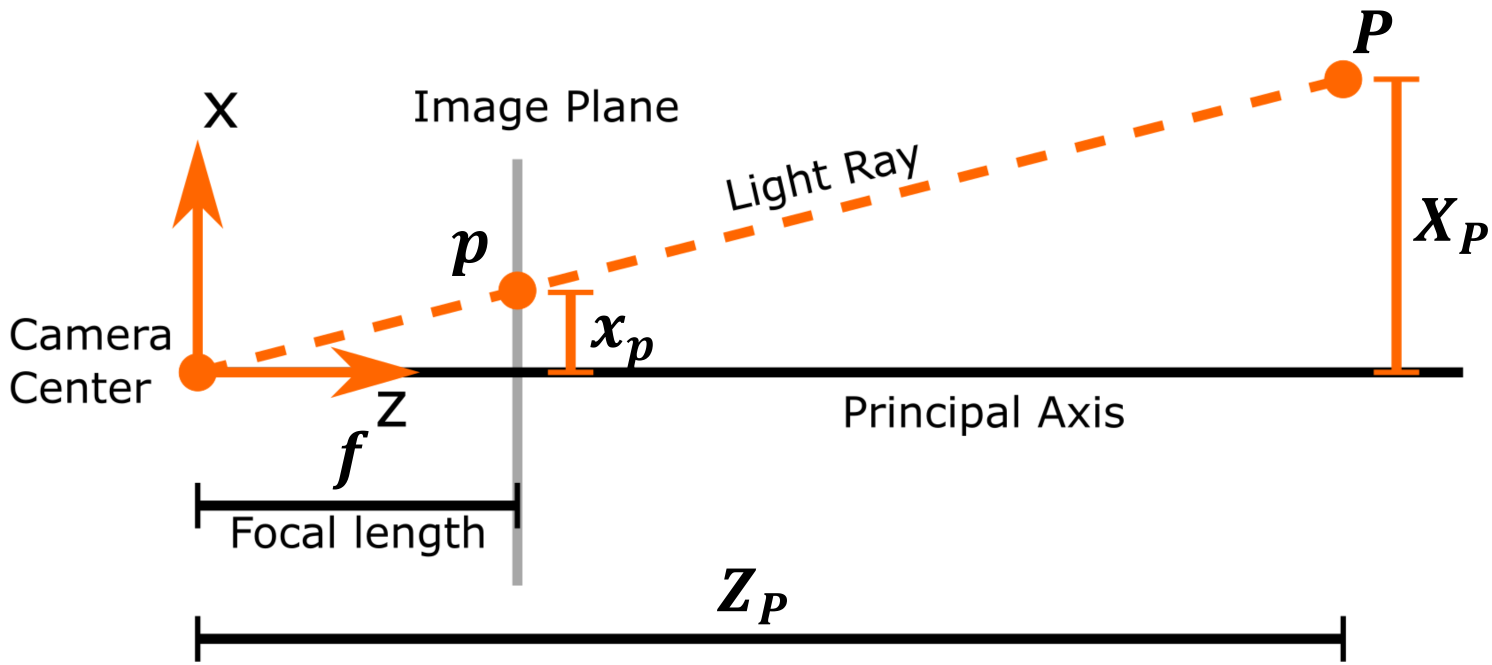
School of Computation, Information and Technology

Chair of Cognitive Systems

Munich 04. June 2024

Contents

Introduction to Camera Projection



In the camera coordinate (x, y, z) :

$$\frac{x_p}{f} = \frac{X_P}{Z_P} \qquad \frac{y_p}{f} = \frac{Y_P}{Z_P}$$

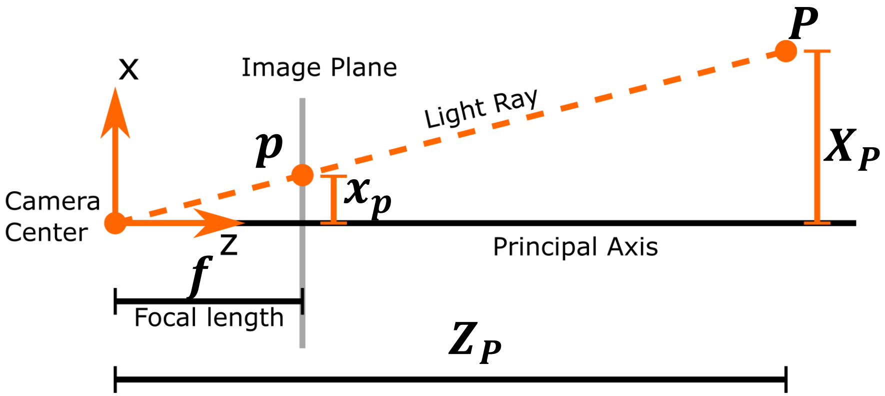
Introduction to Camera Projection

In the camera coordinate (x, y, z) :

$$\frac{x_p}{f} = \frac{X_P}{Z_P} \qquad \frac{y_p}{f} = \frac{Y_P}{Z_P}$$

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \begin{pmatrix} fX_P/Z_P \\ fY_P/Z_P \end{pmatrix}$$

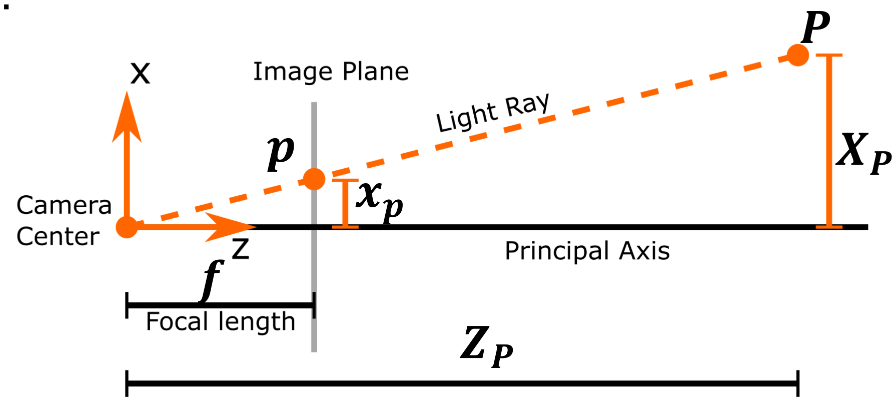
$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_P/Z_P \\ Y_P/Z_P \\ 1 \end{bmatrix}$$



Introduction to Camera Projection

In the camera coordinate (x, y, z) :

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_P/Z_P \\ Y_P/Z_P \\ 1 \end{bmatrix}$$



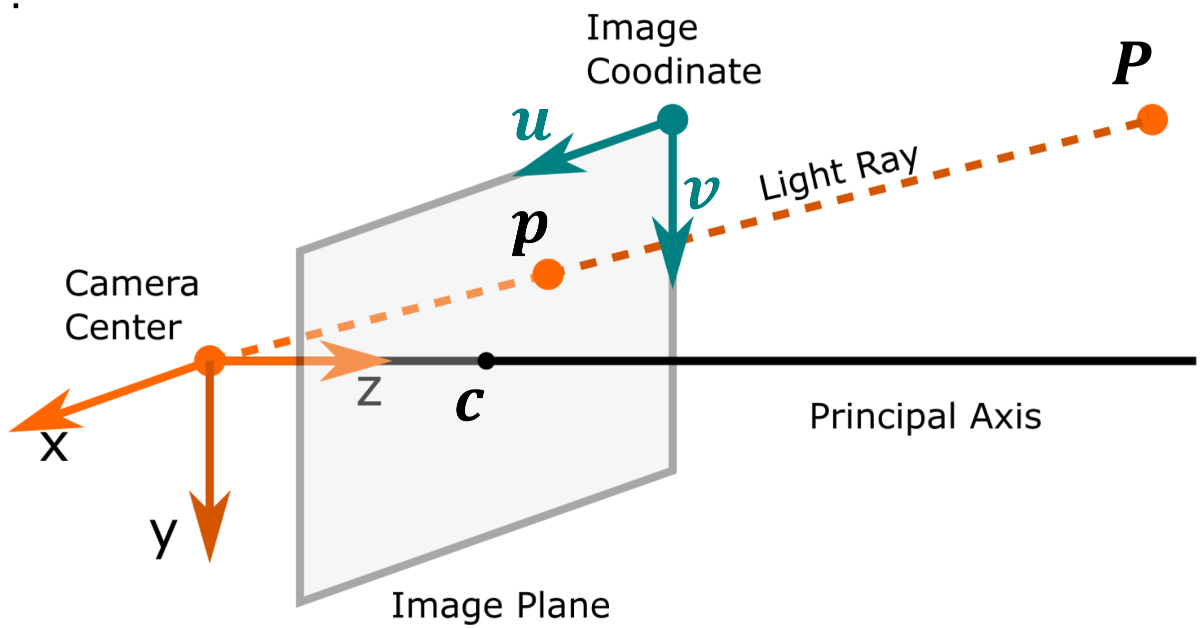
- In perspective projection, 3D points in **camera coordinates** are mapped to the image plane by dividing them by their **Z** component and multiplying with the focal length
- After the projection it is not possible to recover the distance of the 3D point from the image.

Introduction to Camera Projection

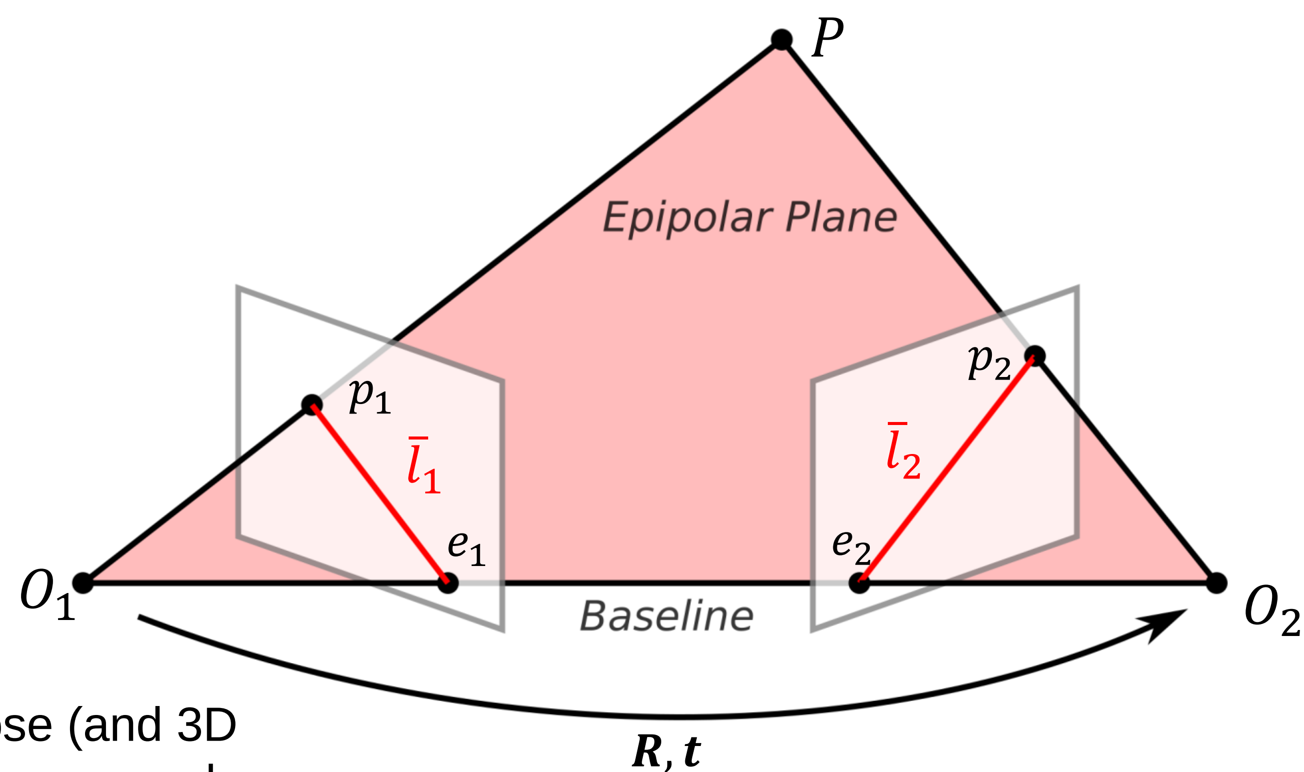
In the image coordinate (u, v) :

$$\begin{pmatrix} u_p \\ v_p \end{pmatrix} = \begin{pmatrix} \frac{fX_P}{Z_P} + c_x \\ \frac{fY_P}{Z_P} + c_y \end{pmatrix}$$

$$\begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_P/Z_P \\ Y_P/Z_P \\ 1 \end{bmatrix}$$



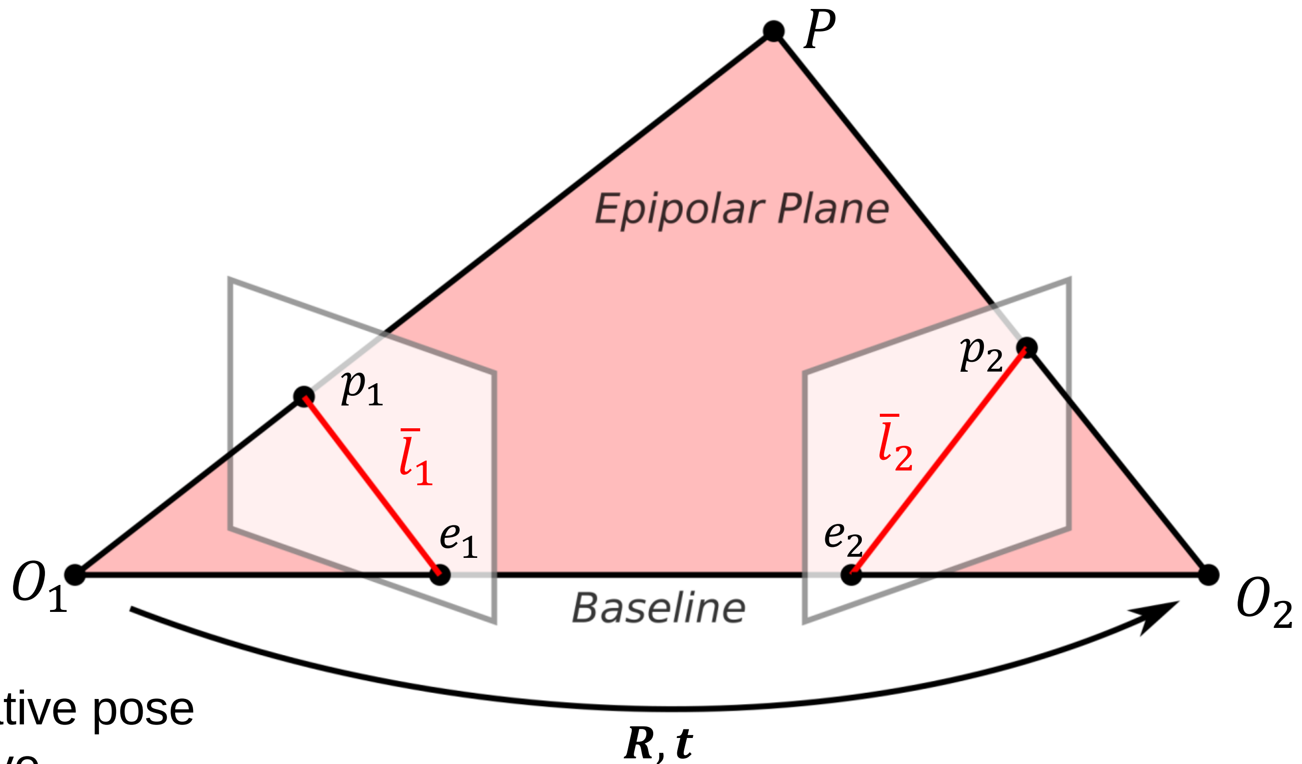
Epipolar Geometry



Recovery of camera pose (and 3D structure) from image correspondences. The required relationships are described by the two-view **epipolar geometry**.

Epipolar lines: $\bar{l}_1, \bar{l}_2$
Epipoles: e_1, e_2

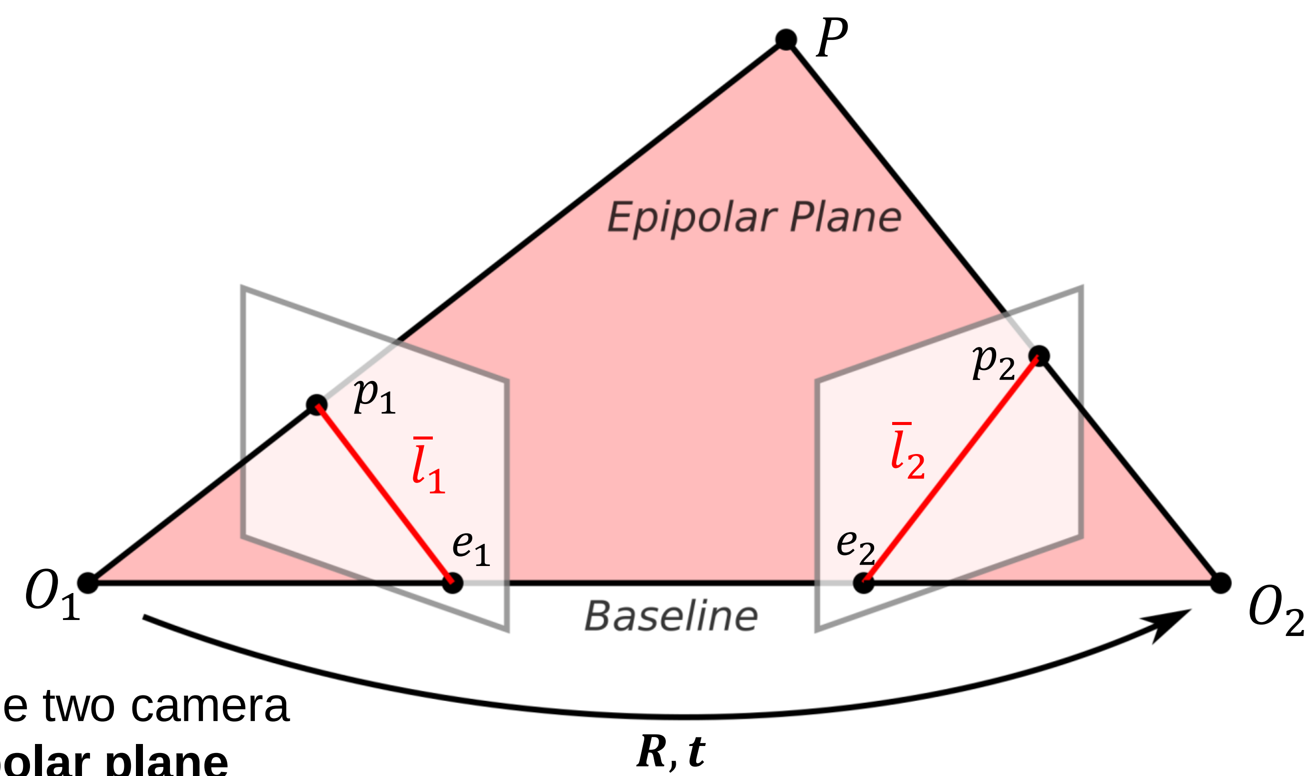
Epipolar Geometry



R and **t** denote the relative pose between two perspective cameras

Epipolar lines: $\bar{l}_1, \bar{l}_2$
Epipoles: e_1, e_2

Epipolar Geometry



The 3D point **P** and the two camera centers span the **epipolar plane**

The correspondence of pixel p_1 in image 2 must lie on the **epipolar line** $\bar{l}_2$ in image 2

Epipolar lines: $\bar{l}_1, \bar{l}_2$
Epipoles: e_1, e_2

Epipolar Geometry

$$\tilde{P}_2 = \boldsymbol{R}\tilde{P}_1 + \boldsymbol{t}$$

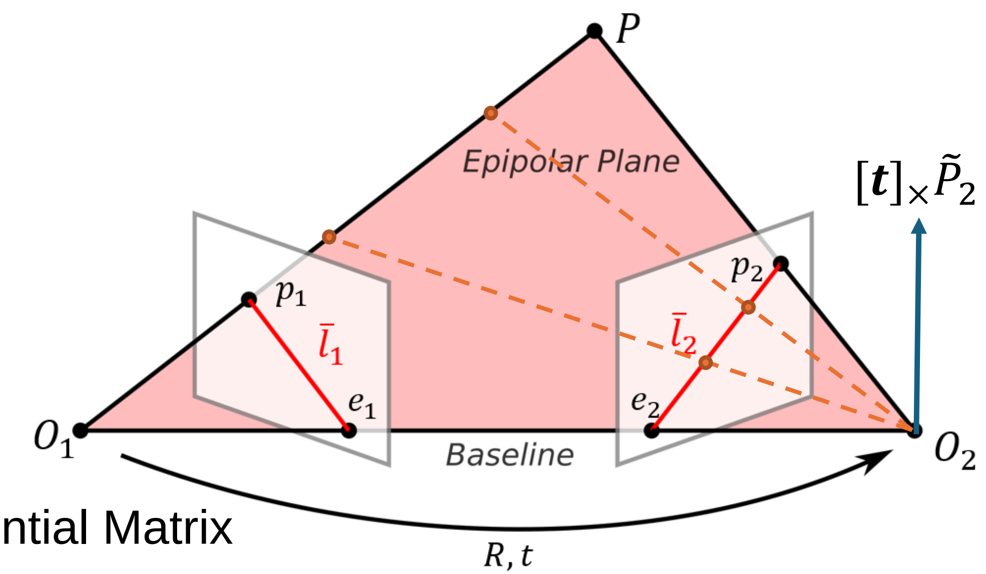
$$[\boldsymbol{t}]_\times \tilde{P}_2 = [\boldsymbol{t}]_\times \boldsymbol{R}\tilde{P}_1$$

$$\tilde{P}_2^T [\boldsymbol{t}]_\times \tilde{P}_2 = \tilde{P}_2^T [\boldsymbol{t}]_\times \boldsymbol{R}\tilde{P}_1 = 0$$

$$\boxed{\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}}$$

Essential Matrix

$$\tilde{P}_2^T \boldsymbol{E}\tilde{P}_1 = 0$$



Epipolar lines: $\bar{l}_1, \bar{l}_2$

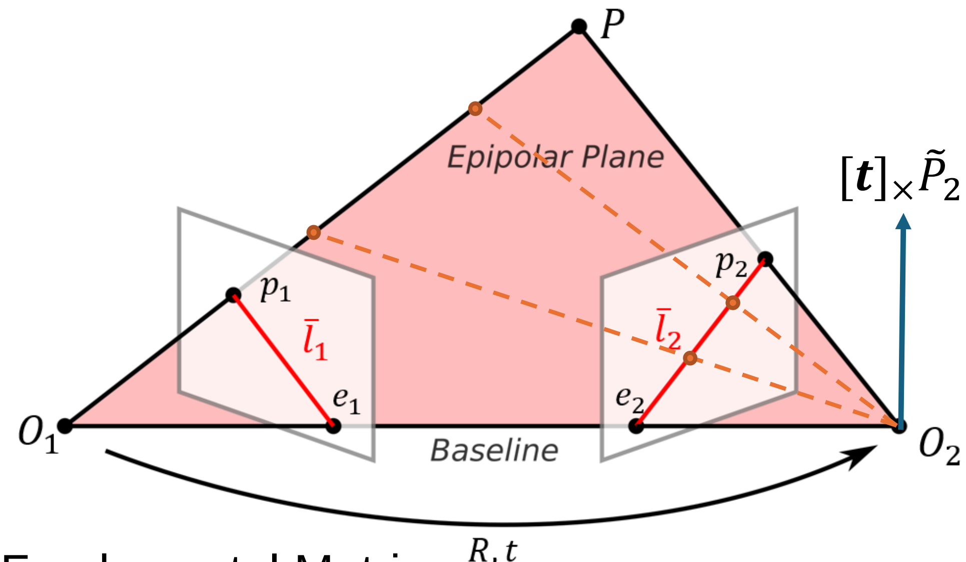
Epipoles: e_1, e_2

Epipolar Geometry

$$p_1 = K\tilde{P}_1,\, p_2 = K\tilde{P}_2$$

$$\tilde{P}_1 = K^{-1}p_1,\, \tilde{P}_2 = K^{-1}p_2$$

$$p_2^T {K^{-1}}^T E K^{-1} p_1$$



$$\boxed{\boldsymbol{F} = {K^{-1}}^T \boldsymbol{E} K^{-1}} \qquad p_2^T \boldsymbol{F} p_1 = 0 \qquad \text{Fundamental Matrix}$$

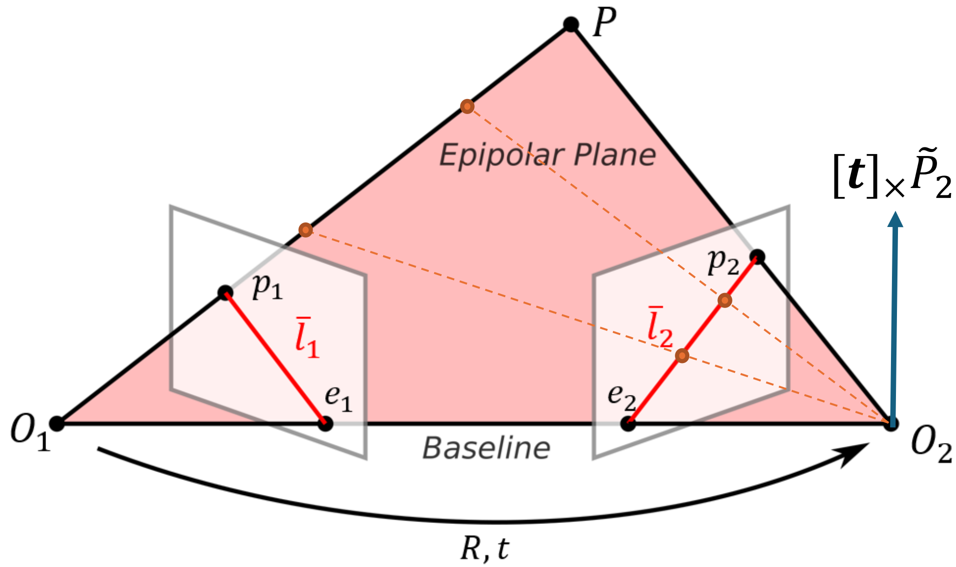
Epipolar lines: $\bar{l}_1, \bar{l}_2$
Epipoles: e_1, e_2

$$\bar{l}_2 = \boldsymbol{F} p_1 \text{ , } \bar{l}_1 = p_2^T \boldsymbol{F} \text{ , } p_2^T \bar{l}_2 = 0 \text{ , } \bar{l}_1 p_1 = 0$$

Estimating the Essential Matrix

$$\tilde{P}_2^T \boldsymbol{E} \tilde{P}_1 = 0$$

$$x_1 x_2 e_{11} + y_1 x_2 e_{12} + x_2 e_{13}$$
$$+ x_1 y_2 e_{21} + y_1 y_2 e_{22} + y_2 e_{23}$$
$$+ x_1 e_{31} + y_1 e_{32} + e_3 = 0$$



As $\boldsymbol{E}$ is homogeneous we use singular value decomposition to constrain the scale.
Note that some terms are products of two image measurements and hence amplify measurement noise asymmetrically. Thus, the normalized 8-point algorithm whitens the observations to have zero-mean and unit variance before the calculation and back-transforms the matrix recovered by SVD accordingly.

Estimating the Translation and Rotation

From $\boldsymbol{E}$, we can recover the direction of the translation vector $\boldsymbol{t}$. We have:

$$\hat{\boldsymbol{t}}^T \boldsymbol{E} = \hat{\boldsymbol{t}}^T [\boldsymbol{t}]_\times \boldsymbol{R} = 0$$

Thus, $\boldsymbol{E}$ is singular and we obtain $\hat{\boldsymbol{t}}$ as the left singular vector associated with singular value 0. In practice the singular value will not be exactly 0 due to measurement noise, and we choose the smallest one. The other two singular values are roughly equal

$$\boldsymbol{E} = \boldsymbol{U\Sigma V}^T,\ \boldsymbol{\Sigma} = diag(1, 1, 0) \text{ to scale}$$

Estimating the Translation and Rotation

Two possible choices of the rotation matrix $\boldsymbol{R}$:

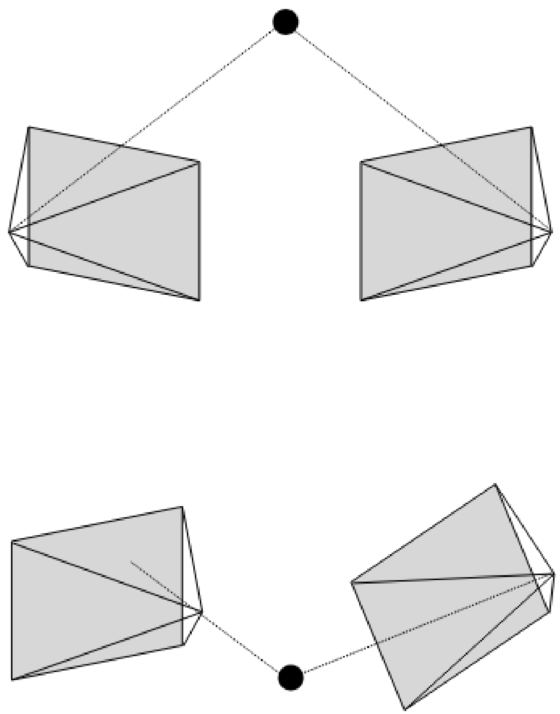
$$\boldsymbol{R_1} = \boldsymbol{UZV}^T \text{ or } \boldsymbol{R_1} = \boldsymbol{UZ^TV}^T, \text{ where } \boldsymbol{Z} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Two possible choices of translation direction $\hat{\boldsymbol{t}}$

$$\hat{\boldsymbol{t}} = \boldsymbol{u}_3 \text{ or } -\boldsymbol{u}_3, \text{ where } \boldsymbol{u}_3 \text{ is the last column of } \boldsymbol{U}$$

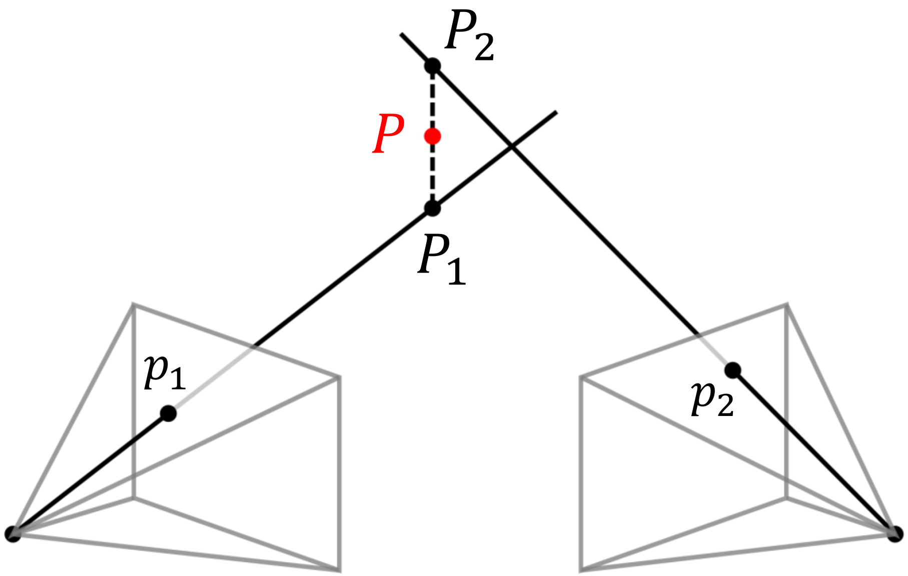
Estimating the Translation and Rotation

Four possible solutions: can be solved by point triangulation



Triangulation

How to recover 3D geometry with known intrinsic and extrinsic?



Given noisy 2D image correspondings p_1 and p_2, the two light rays might not intersect.
We want to recover the 3D point P that is closest to the two rays.

Triangulation

In the projection process $p_i = K\tilde{P}_i$, both sides are homogeneous, they have the same direction but different magnitude.
To account for this, we consider the cross product $p_i \times K\tilde{P}_i = 0$, in a rowwise form:

$$\underbrace{\begin{bmatrix} x_i k_{i3} - k_{i1} \\ y_i k_{i3} - k_{i2} \end{bmatrix}}_{\boldsymbol{A}_i} \tilde{P}_i = 0$$

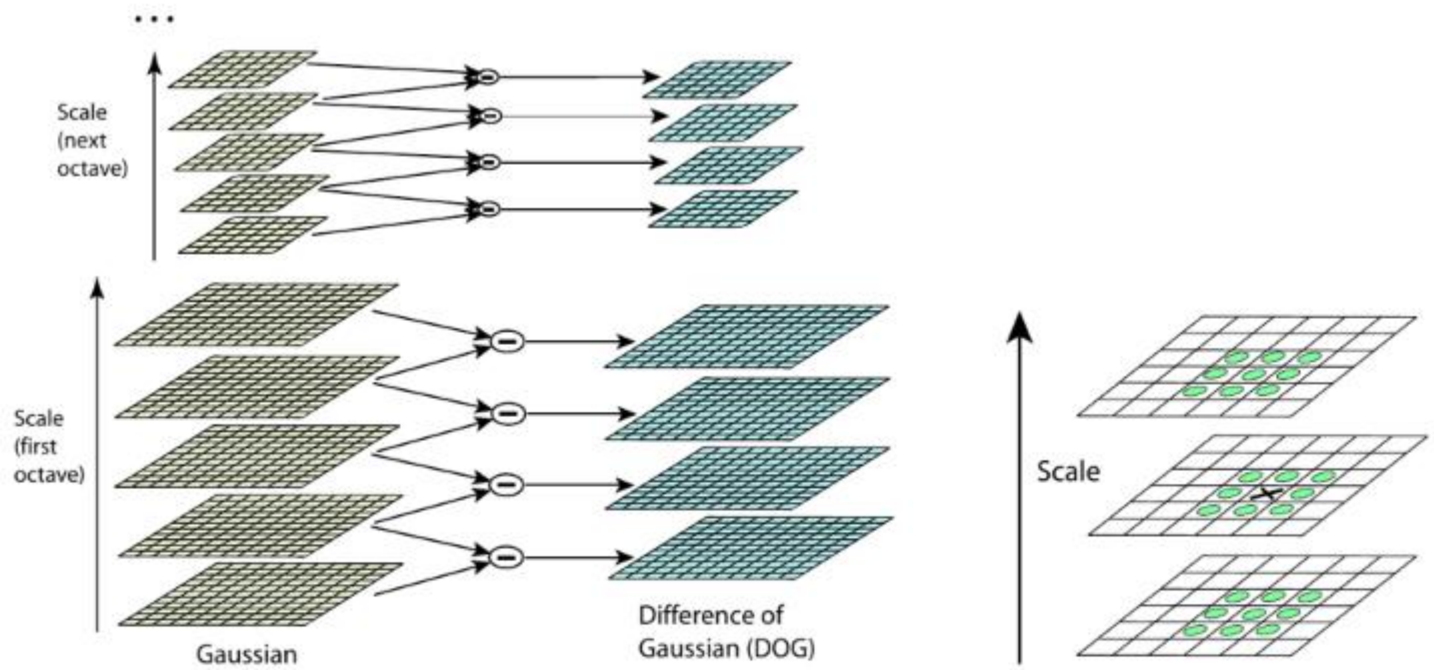
Stacking N ≥ 2 observations of a point, we obtain a linear system $\boldsymbol{A}\tilde{P}_i = 0$.
As $\tilde{P}_i$ is homogeneous this leads to a constrained least squares problem. The solution to this problem is the **right singular vector** corresponding to the smallest singular value of $\boldsymbol{A}$.

Feature Points

Detection and description

- Features should be invariant to perspective effects and illumination
- The same point should have similar vectors independent of pose/viewpoint
- Plain RGB/intensity patches will not have this property, we need something better

Feature Points:
Scale Invariant Feature Transform (SIFT)



- SIFT constructs a scale space by iteratively filtering the image with a Gaussian
- Adjacent scales are subtracted, yielding Difference of Gaussian (DoG) images
- Interest points (=blobs) are detected as extrema in the resulting scale space

Feature Points:
Scale Invariant Feature Transform (SIFT)

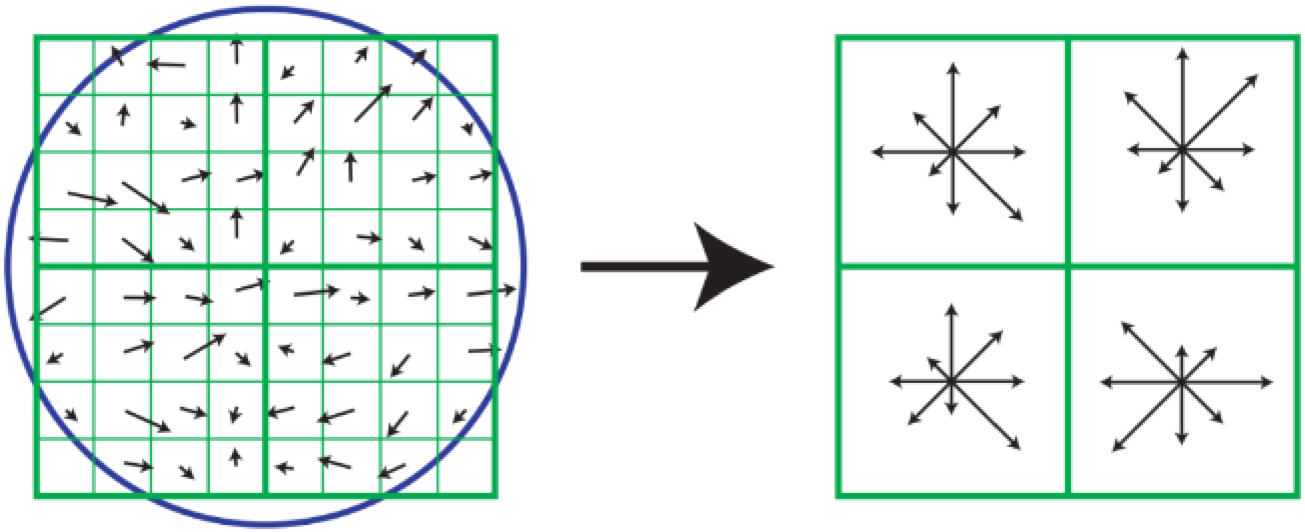
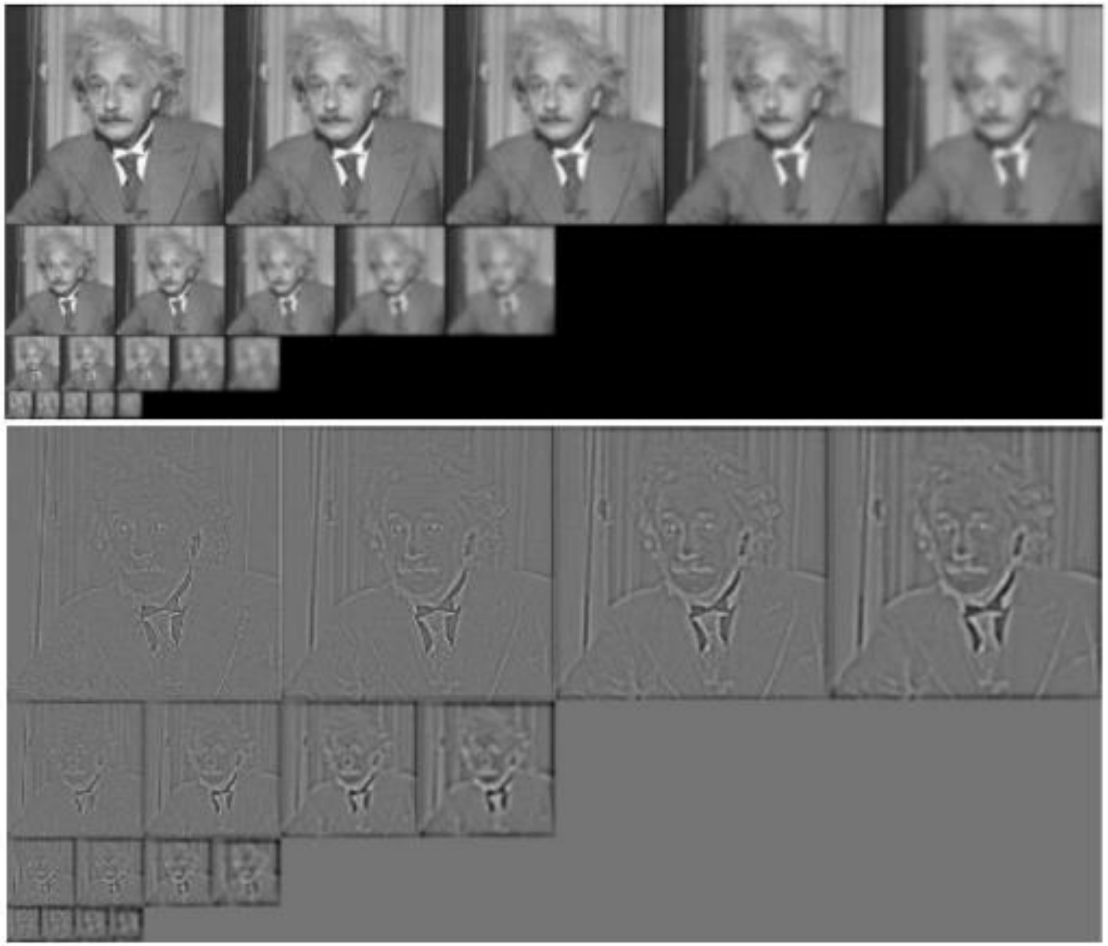


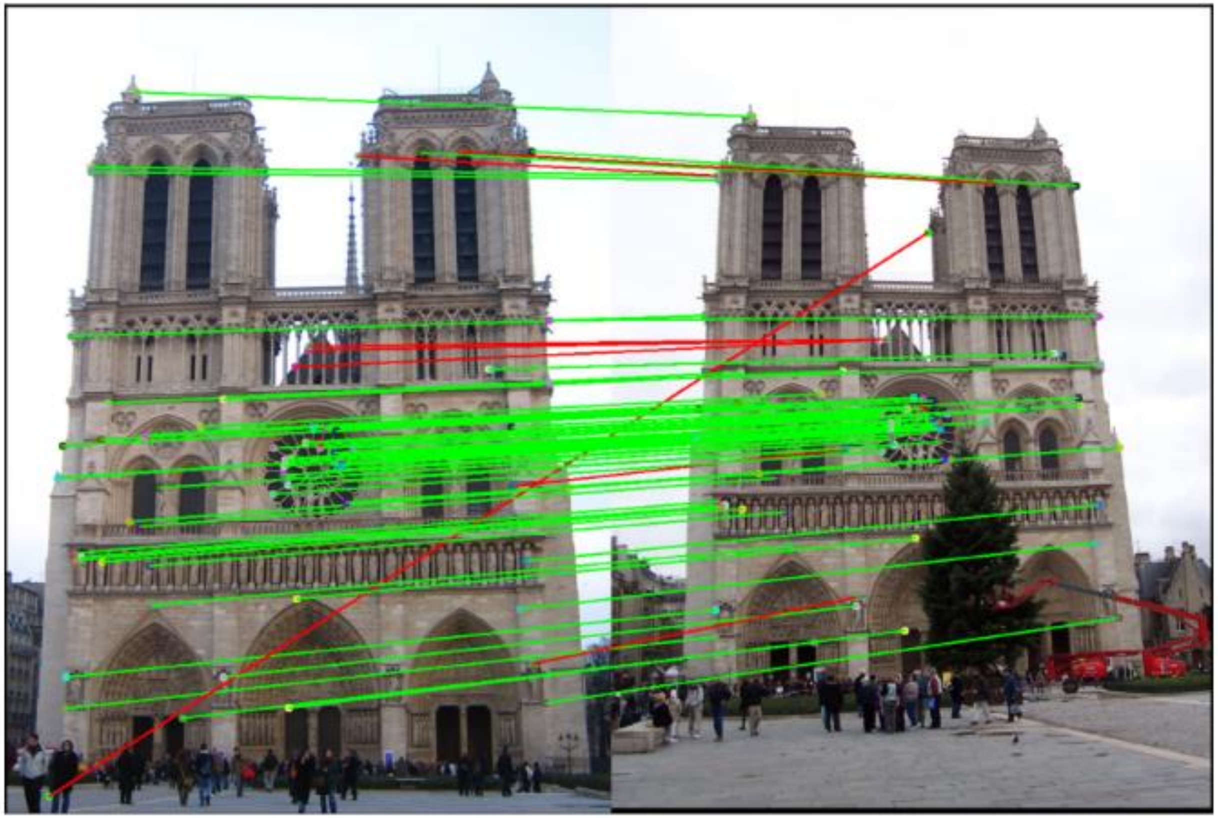
Image gradients Keypoint descriptor

- SIFT rotates the descriptor to align with the dominant gradient orientation
- Gradient histograms are computed for local sub-regions of the descriptor
- All histograms are concatenated and normalized to form a 128D feature vector

Feature Points:
Scale Invariant Feature Transform (SIFT)



Feature Points:
Example



Example of two images with SIFT correspondences (green: correct, red: wrong)

Feature Points: Detection and Description

- Many algorithms for feature detection and description have been developed, e.g., SIFT, SURF, U-SURF, BRISK, ORB, FAST, and recently deep learning based ones
- SIFT was a seminal work due to its invariance and robustness which revolutionized recognition and in particular matching and enabled the development of large-scale SfM techniques which we discuss in this lecture
- Despite >20 years old, SIFT is still used today (e.g., in the SfM pipeline COLMAP)
- Feature correspondences can be retrieved with efficient nearest neighbor search
- Ambiguous matches are typically filtered by computing the ratio of distance from the closest neighbor to the distance of the second closest
- A large ratio (>0.8) indicates that the found match might not be the correct one

Tutorial:
Requirements

Python 3, numpy, opencv >= 3.1x

Contents:

1. Feature points detection: SIFT and OBR
2. Feature matching: Brute Force, KNN.
3. Estimating the essential matrix
4. Decompose the estimated essential matrix: singular values, vectors
5. Estimating translation and rotation matrix
6. Get the angle errors for the translation direction and rotation matrix.