Estimation of Camera Pose, Nonlinear

Computer Vision II

CSE 252B

Lecture 8

Announcements

- Assignment 2 is due Feb 4, 11:59 PM
- Assignment 3 will be released Feb 4
  - Due Feb 18, 11:59 PM

Calibrated camera

Camera calibration matrix

$$\mathtt{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where

$\alpha_x = f m_x$	Focal length in x direction in terms of pixel dimensions
$\alpha_y = f m_y$	Focal length in y direction in terms of pixel dimensions
s	Skew
$(x_0, y_0)^\top = (p_x m_x, p_y m_y)^\top$	Coordinates of principal point in terms of pixel dimensions

where m_x and m_y are number of pixels per unit disance in x and y directions, respectively

(and lens distortion parameters)

Single view geometry

Camera projection

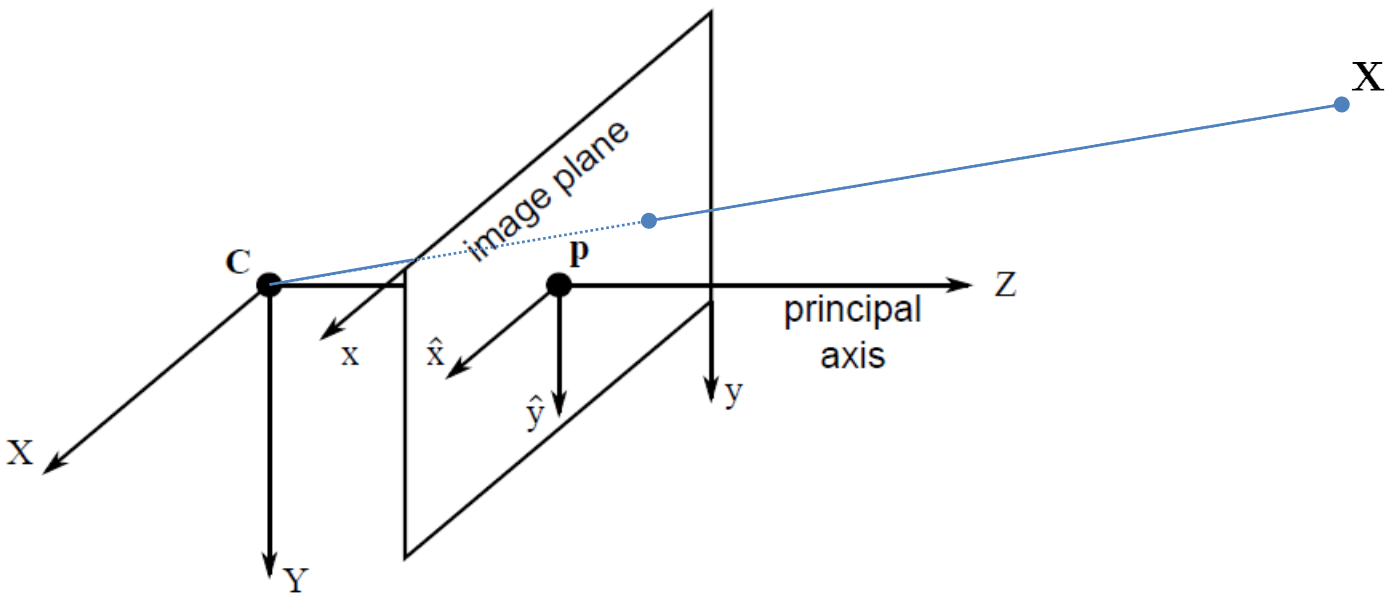
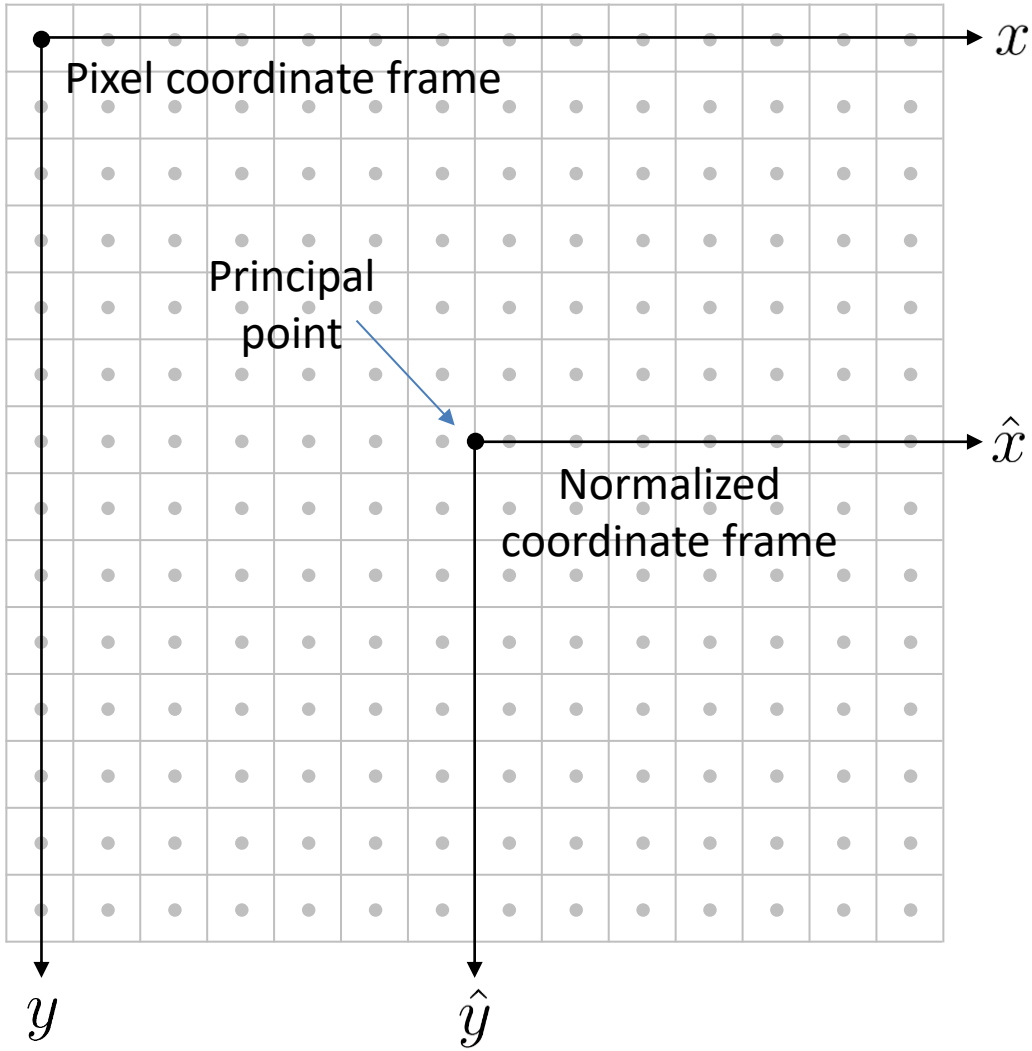


Image coordinate frames



Camera projection

Project 3D point in world coordinate frame $\mathbf{X}$ under camera projection matrix $\mathtt{P} = \mathtt{K}[\mathtt{R} \,|\, \mathbf{t}]$

$$\begin{aligned} \mathbf{x} &= \mathtt{P}\mathbf{X} \\ \mathbf{x} &= \mathtt{K}[\mathtt{R} \,|\, \mathbf{t}]\mathbf{X} \\ \mathtt{K}^{-1}\mathbf{x} &= [\mathtt{R} \,|\, \mathbf{t}]\mathbf{X} \\ \hat{\mathbf{x}} &= \hat{\mathtt{P}}\mathbf{X} \end{aligned}$$

Camera calibration matrix

where

Camera projection matrix	$\mathtt{P} = \mathtt{K}[\mathtt{R} \,|\, \mathbf{t}]$	$= \mathtt{K}\hat{\mathtt{P}}$
Normalized camera projection matrix	$\hat{\mathtt{P}} = [\mathtt{R} \,|\, \mathbf{t}]$	$= \mathtt{K}^{-1}\mathtt{P}$
Image point in pixel coordinates	$\mathbf{x} = \mathtt{K}[\mathtt{R} \,|\, \mathbf{t}]\mathbf{X}$	$= \mathtt{K}\hat{\mathbf{x}}$
Image point in normalized coordinates	$\hat{\mathbf{x}} = [\mathtt{R} \,|\, \mathbf{t}]\mathbf{X}$	$= \mathtt{K}^{-1}\mathbf{x}$

Together, the camera rotation and camera translation (or camera center $\tilde{\mathbf{C}} = -\mathtt{R}^{\top}\mathbf{t}$) are called the camera pose. The normalized camera projection matrix $\hat{\mathtt{P}} = [\mathtt{R} \,|\, \mathbf{t}]$ embodies the camera pose.

Calibrated camera

$$\mathtt{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where

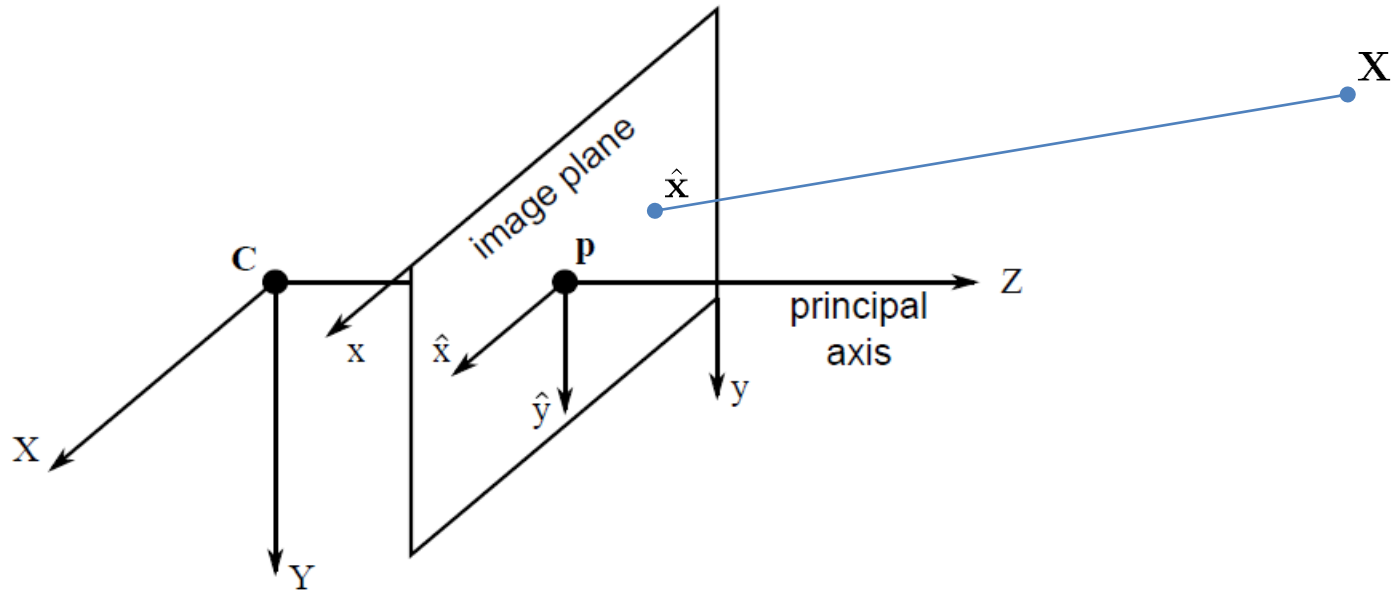
$\alpha_x = f m_x$	Focal length in x direction in terms of pixel dimensions
$\alpha_y = f m_y$	Focal length in y direction in terms of pixel dimensions
s	Skew
$(x_0, y_0)^\top = (p_x m_x, p_y m_y)^\top$	Coordinates of principal point in terms of pixel dimensions

where m_x and m_y are number of pixels per unit disance in x and y directions, respectively

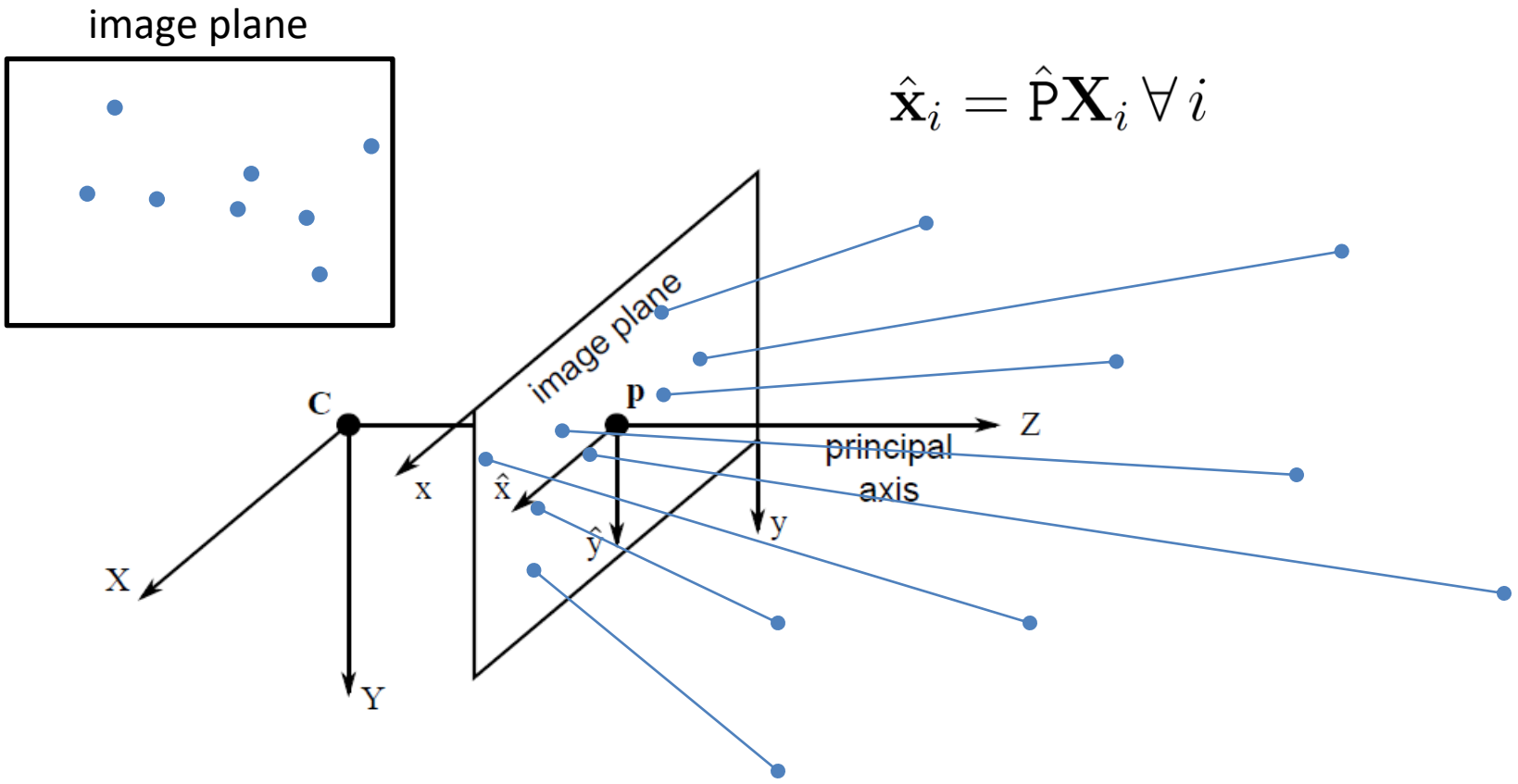
- If camera calibration parameters are known, then use normalized camera projection matrix and image points in normalized coordinates

Normalized camera projection matrix

$$\hat{\mathbf{x}} = \hat{\mathrm{P}}\mathbf{X}$$

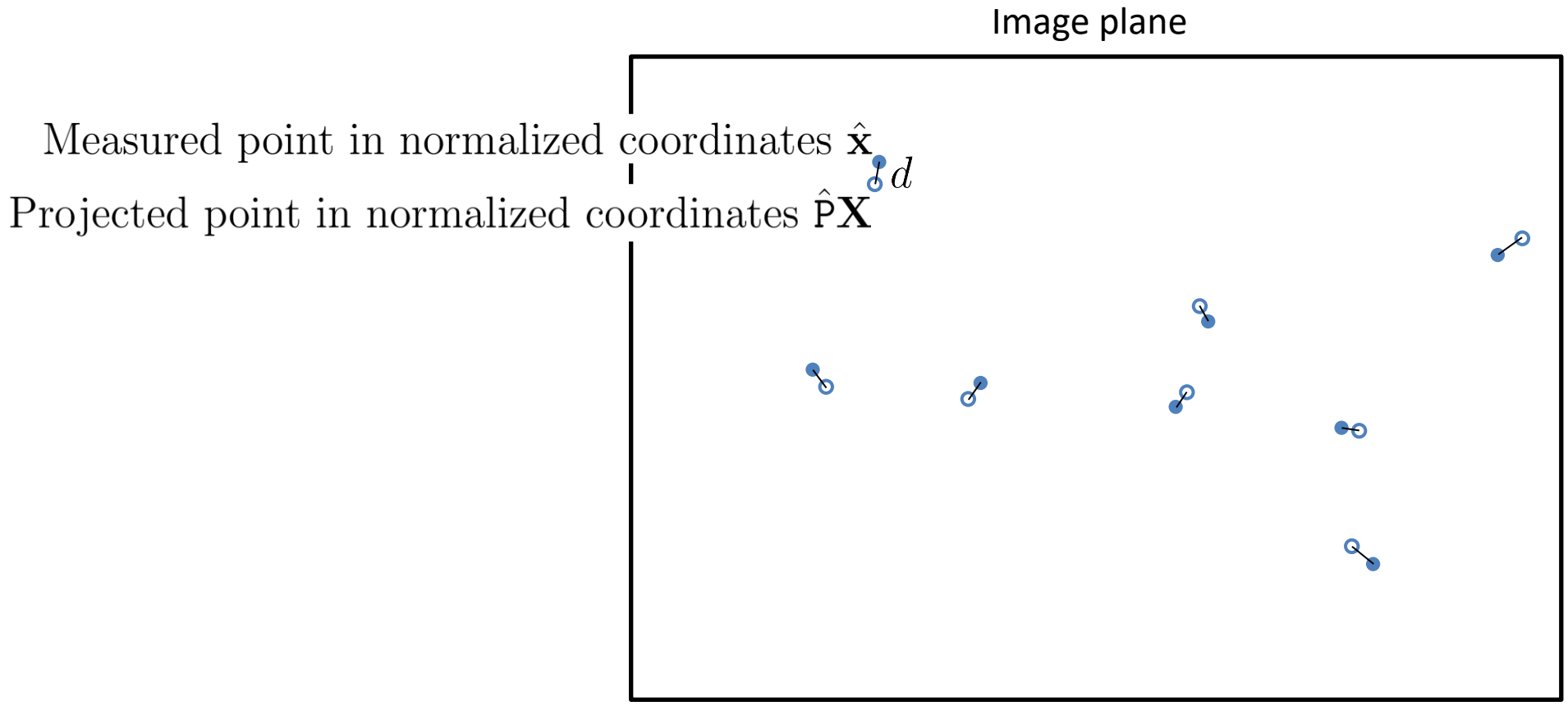


Normalized camera projection matrix



Camera projection matrix

- Error in image point measurements



Normalized camera projection matrix

Linear projection of points in homogeneous coordinates

$$\hat{\mathbf{x}} = \hat{\mathtt{P}}\mathbf{X}$$
$$\hat{\mathbf{x}} = [\mathtt{R} \,|\, \mathbf{t}]\mathbf{X}$$
$$\hat{\mathbf{x}} = [\mathtt{I} \,|\, \mathbf{0}]\mathbf{X}_{\text{cam}}$$
$$\hat{\mathbf{x}} = [\mathtt{I} \,|\, \mathbf{0}] \begin{bmatrix} \tilde{\mathbf{X}}_{\text{cam}} \\ 1 \end{bmatrix}$$
$$\hat{\mathbf{x}} = \tilde{\mathbf{X}}_{\text{cam}}, \text{ where } \tilde{\mathbf{X}}_{\text{cam}} = \mathtt{R}\tilde{\mathbf{X}} + \mathbf{t}$$
$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{w} \end{bmatrix} = \begin{bmatrix} \tilde{X}_{\text{cam}} \\ \tilde{Y}_{\text{cam}} \\ \tilde{Z}_{\text{cam}} \end{bmatrix}$$

Nonlinear projection of points in inhomogeneous coordinates

$$\hat{\tilde{x}} = \frac{\hat{x}}{\hat{w}} = \frac{\tilde{X}_{\text{cam}}}{\tilde{Z}_{\text{cam}}} \text{ and}$$
$$\hat{\tilde{y}} = \frac{\hat{y}}{\hat{w}} = \frac{\tilde{Y}_{\text{cam}}}{\tilde{Z}_{\text{cam}}}$$

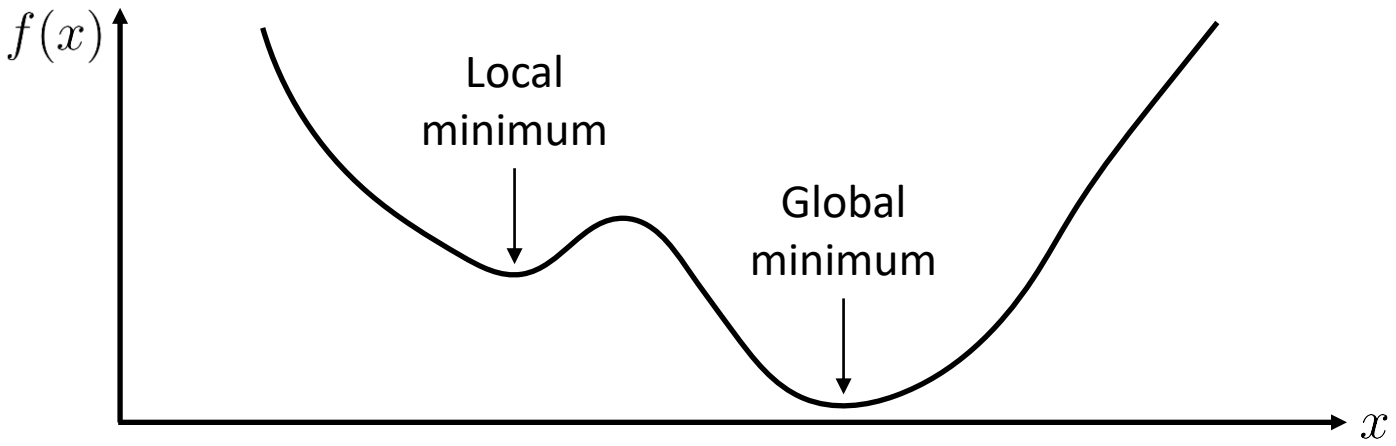
Do not forget this is a nonlinear projection

Estimation of normalized camera projection matrix

$$\hat{\mathbf{x}}_i = \hat{\mathtt{P}}\mathbf{X}_i \,\forall\, i$$

$$\hat{\mathbf{x}}_i = [\mathtt{R} \,|\, \mathbf{t}]\mathbf{X}_i \,\forall\, i, \text{ where } \hat{\mathbf{x}}_i = \mathtt{K}^{-1}\mathbf{x}_i \,\forall\, i \text{ and } \hat{\mathtt{P}} = [\mathtt{R} \,|\, \mathbf{t}], \text{ solve for } \mathtt{R} \text{ and } \mathbf{t}$$

- Given point correspondences $\hat{\mathbf{x}}_i \leftrightarrow \mathbf{X}_i$
- Minimize geometric error $\sum_i d(\hat{\mathbf{x}}_i, \hat{\mathtt{P}}\mathbf{X}_i)^2$
- A nonlinear mapping requires a nonlinear optimization problem solver
  - Use linear estimation for initial estimate (to mitigate converging to local optimum)
  - Iterative process to determine global optimum



Linear estimation of normalized camera projection matrix

Given point correspondences $\hat{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{X}}_i$

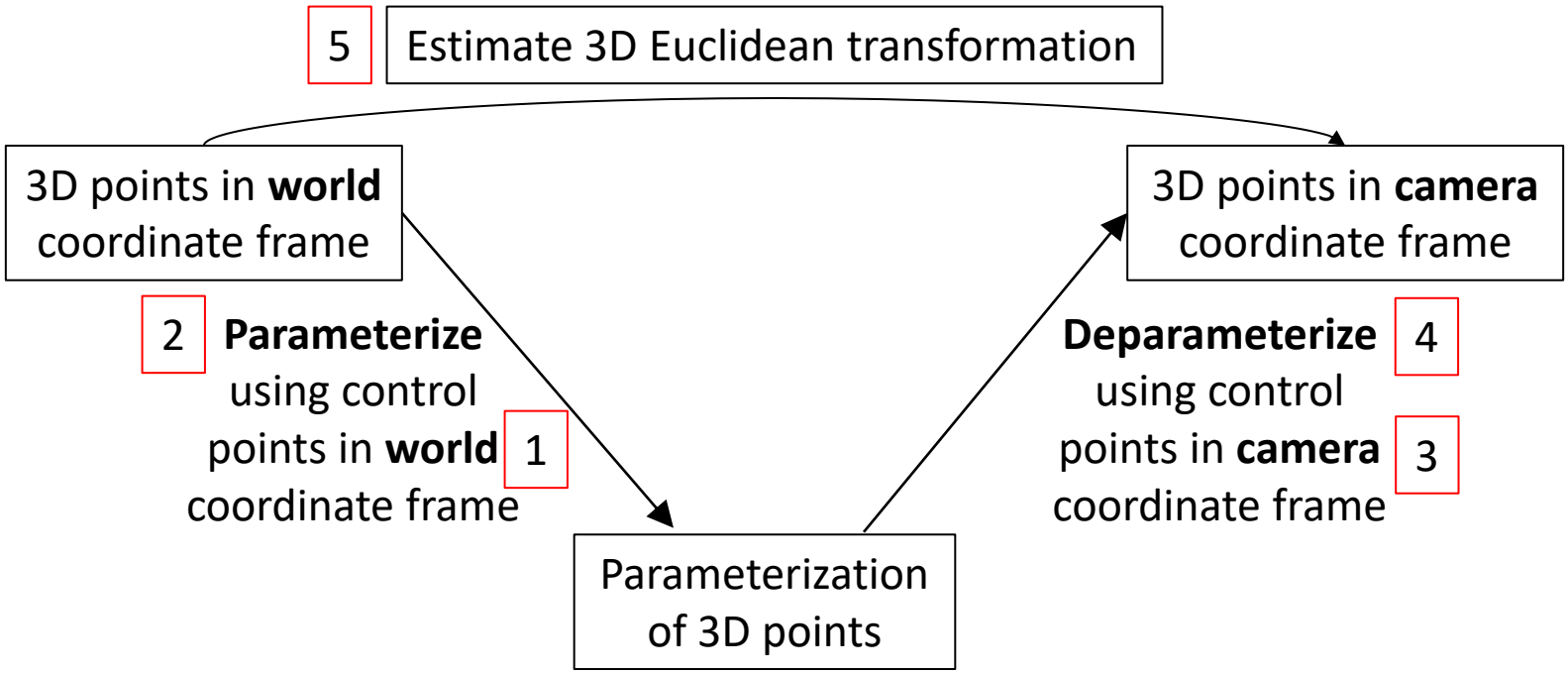
$$\hat{\mathbf{x}}_i = \hat{\mathtt{P}} \begin{bmatrix} \tilde{\mathbf{X}}_i \\ 1 \end{bmatrix} \forall i, \text{ solve for } \hat{\mathtt{P}}$$

$$\hat{\mathbf{x}}_i = [\mathtt{R} \,|\, \mathbf{t}] \begin{bmatrix} \tilde{\mathbf{X}}_i \\ 1 \end{bmatrix} \forall i, \text{ where } \hat{\mathtt{P}} = [\mathtt{R} \,|\, \mathbf{t}], \text{ solve for } \mathtt{R} \text{ and } \mathbf{t}$$

$$\hat{\mathbf{x}}_i = \tilde{\mathbf{X}}_{\text{cam},i} \forall i, \text{ where } \tilde{\mathbf{X}}_{\text{cam},i} = \mathtt{R}\tilde{\mathbf{X}}_i + \mathbf{t} \forall i, \text{ solve for } \mathtt{R} \text{ and } \mathbf{t}$$

Determine set of 3D points in the camera coordinate frame corresponding to set of 3D points in the world coordinate frame (i.e., $\hat{\mathbf{X}}_{\text{cam},i} \leftrightarrow \tilde{\mathbf{X}}_i$, where $\tilde{\mathbf{X}}_{\text{cam},i} = \mathtt{R}\tilde{\mathbf{X}}_i + \mathbf{t} \forall i$), then solve for $\mathtt{R}$ and $\mathbf{t}$ using $\hat{\mathbf{X}}_{\text{cam},i} \leftrightarrow \tilde{\mathbf{X}}_i$

Linear estimation of normalized camera projection matrix



Nonlinear estimation using iterative estimation methods

Measurement vector $\mathbf{X}$ ← Not 3D point

Parameter vector $\mathbf{P}$ ← Not camera projection matrix

Nonlinear function f, where $\mathbf{X} = f(\mathbf{P})$

Not normalized camera projection matrix

Given an estimate of the parameter vector $\hat{\mathbf{P}}$ and assuming f is approximately linear in the region about $\hat{\mathbf{P}}$

$$\mathbf{X} \approx f(\hat{\mathbf{P}}) + \frac{\partial \hat{\mathbf{X}}}{\partial \hat{\mathbf{P}}}(\mathbf{P} - \hat{\mathbf{P}})$$

$$\mathbf{X} \approx \hat{\mathbf{X}} + \mathtt{J}(\mathbf{P} - \hat{\mathbf{P}}), \text{ where } \hat{\mathbf{X}} = f(\hat{\mathbf{P}}) \text{ and } \mathtt{J} = \frac{\partial \hat{\mathbf{X}}}{\partial \hat{\mathbf{P}}}$$

$$\mathbf{X} - \hat{\mathbf{X}} \approx \mathtt{J}(\mathbf{P} - \hat{\mathbf{P}})$$

$$\boldsymbol{\epsilon} \approx \mathtt{J}\boldsymbol{\delta}, \text{ solve for } \boldsymbol{\delta} \text{ where } \boldsymbol{\epsilon} = \mathbf{X} - \hat{\mathbf{X}} \text{ and } \boldsymbol{\delta} = \mathbf{P} - \hat{\mathbf{P}}$$

Then

$$\boldsymbol{\delta} = \mathbf{P} - \hat{\mathbf{P}}$$

$$\hat{\mathbf{P}} + \boldsymbol{\delta} = \mathbf{P}$$

Adjustment

Perform again (i.e., iterate) using resulting $\mathbf{P}$ as $\hat{\mathbf{P}}$

Nonlinear estimation using the Levenberg-Marquardt algorithm

Given

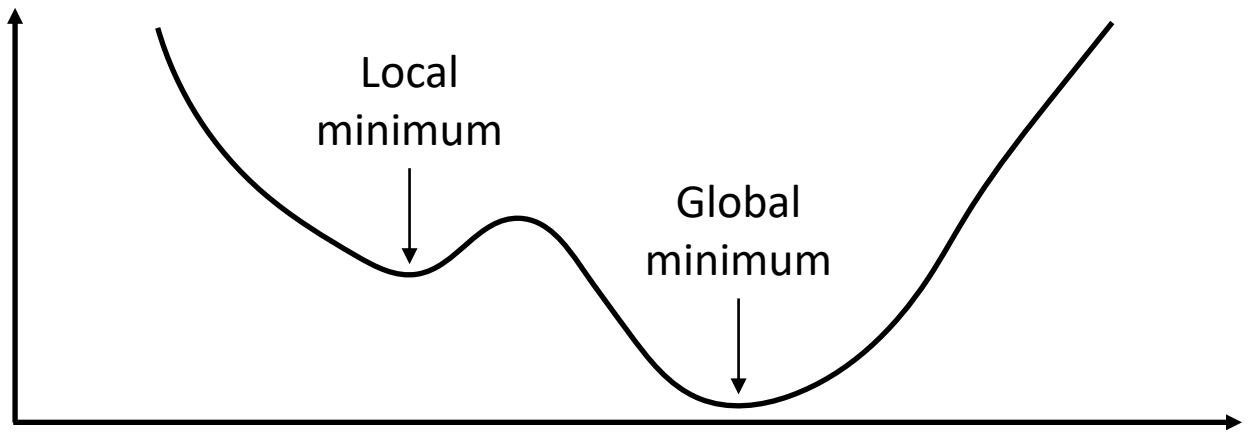
Measurement vector $\mathbf{X}$ with associated covariance matrix $\Sigma_{\mathbf{X}}$

Parameter vector $\hat{\mathbf{P}}$ (initial estimate)

Nonlinear function f, where $\hat{\mathbf{X}} = f(\hat{\mathbf{P}})$

Objective

Find $\hat{\mathbf{P}}$ that minimizes $\boldsymbol{\epsilon}^\top \Sigma_{\mathbf{X}}^{-1} \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = \mathbf{X} - \hat{\mathbf{X}}$



Nonlinear estimation using the Levenberg-Marquardt algorithm

Algorithm

1. $\lambda = 0.001$ and $\boldsymbol{\epsilon} = \mathbf{X} - \hat{\mathbf{X}}$
2. Jacobian $\mathtt{J} = \dfrac{\partial \hat{\mathbf{X}}}{\partial \hat{\mathbf{P}}}$
3. Weighted least-squares normal equations $\left(\mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \mathtt{J}\right) \boldsymbol{\delta} = \mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \boldsymbol{\epsilon}$
4. Augmented normal equations $\left(\mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \mathtt{J} + \lambda \mathtt{I}\right) \boldsymbol{\delta} = \mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \boldsymbol{\epsilon}$, solve for $\boldsymbol{\delta}$
5. Candidate parameter vector $\hat{\mathbf{P}}_0 = \hat{\mathbf{P}} + \boldsymbol{\delta}$
6. $\hat{\mathbf{X}}_0 = f(\hat{\mathbf{P}}_0)$, then $\boldsymbol{\epsilon}_0 = \mathbf{X} - \hat{\mathbf{X}}_0$
7. If candidate cost $\boldsymbol{\epsilon}_0^\top \Sigma_{\mathbf{X}}^{-1} \boldsymbol{\epsilon}_0 \geq \boldsymbol{\epsilon}^\top \Sigma_{\mathbf{X}}^{-1} \boldsymbol{\epsilon}$, then

   $\lambda = 10\lambda$

   Go to step 4 (this does not count as an iteration)

   else

   $\hat{\mathbf{P}} = \hat{\mathbf{P}}_0$ and $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}_0$

   $\lambda = \lambda/10$

   Go to step 2 or terminate (this counts as an iteration)

Outer loop

Inner loop

Adjustment

Nonlinear estimation using the Levenberg-Marquardt algorithm

Termination criteria

- Using ratio tolerance τ_1, when

$$1 - \frac{\text{cost}_{\text{current}}}{\text{cost}_{\text{previous}}} \leq \tau_1$$

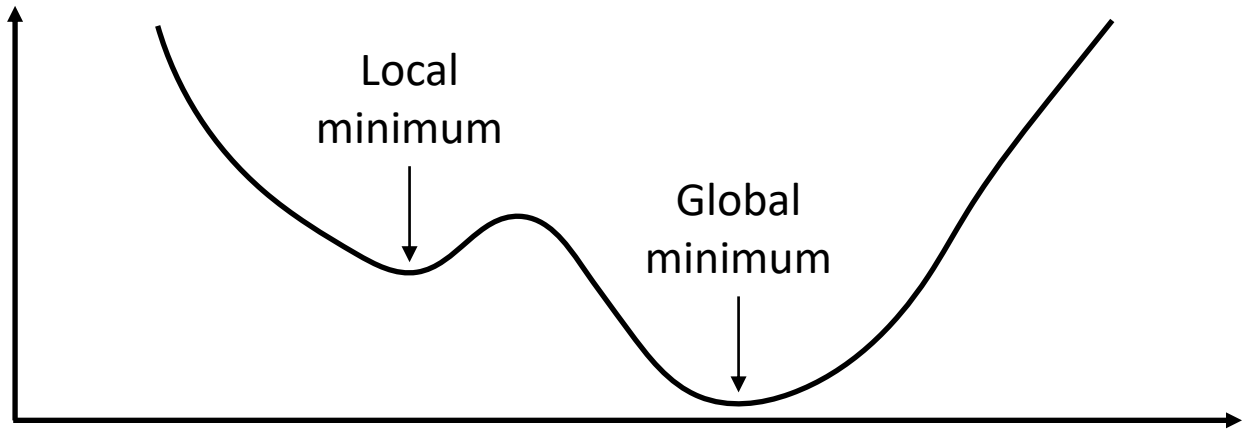
- Using difference tolerance τ_2, when

$$\text{cost}_{\text{previous}} - \text{cost}_{\text{current}} \leq \tau_2$$

- When the number of function evaluations (i.e., inner loop) has reached $100(\text{number of parameters} + 1)$

Parameter vector

- May get stuck in local minimum of high dimensional space
- **Use minimal parameterizations towards mitigating this**



Estimation of normalized camera projection matrix

Together, the camera rotation and camera translation (or camera center $\tilde{\mathbf{C}} = -\mathtt{R}^\top \mathbf{t}$) are called the camera pose. The normalized camera projection matrix $\hat{\mathtt{P}} = [\mathtt{R} \,|\, \mathbf{t}]$ embodies the camera pose.

- Linear estimate results in Euclidean transformation (6 degrees of freedom)
  - The camera translation vector is 3 elements with 3 degrees of freedom
    - Minimal parameterization
  - The camera rotation matrix is 9 elements with 3 degrees of freedom
    - **Not a minimal parameterization**

Rotations

- The rotation group

A group is a set of elements under a binary operation that together satisfy the four fundamental properties of closure, associativity, identity, and inverse.
The rotation group is a group in which the elements are orthogonal matrices $\mathtt{R} \in \mathbb{R}^{n \times n}$ with determinant 1 under the matrix product binary operation. This group is called the special orthogonal group $\mathrm{SO}(n)$ in n-dimensional space.

- Closure: if $\mathtt{R}_1$ and $\mathtt{R}_2$ are elements in $\mathrm{SO}(n)$, then the matrix product $\mathtt{R}_1\mathtt{R}_2$ is also an element in $\mathrm{SO}(n)$
- Associativity: the matrix product is associative, i.e., if $\mathtt{R}_3$ is also an element in $\mathrm{SO}(n)$, then $\mathtt{R}_1(\mathtt{R}_2\mathtt{R}_3) = (\mathtt{R}_1\mathtt{R}_2)\mathtt{R}_3$
- Identity: there is an identity element $\mathtt{I}$ in $\mathrm{SO}(n)$ such that $\mathtt{RI} = \mathtt{IR} = \mathtt{R}$ for every element $\mathtt{R} \in \mathrm{SO}(n)$
- Inverse: there must be an inverse of every element, i.e., the inverse $\mathtt{R}^{-1}$ of every element $\mathtt{R} \in \mathrm{SO}(n)$ is also an element in $\mathrm{SO}(n)$ such that $\mathtt{R}\mathtt{R}^{-1} = \mathtt{R}^{-1}\mathtt{R} = \mathtt{I}$

Rotations

- The rotation matrix

The rotation matrix $\mathtt{R} \in \mathrm{SO}(n)$ has orthonormal rows and orthonormal columns, $\mathtt{R}^{-1} = \mathtt{R}^{\top}$, and

$$\mathrm{SO}(n) \text{ has a manifold of dimension } \frac{n(n-1)}{2}$$

$$\dim(\mathrm{SO}(2)) = 1$$

$$\boxed{\dim(\mathrm{SO}(3)) = 3}$$

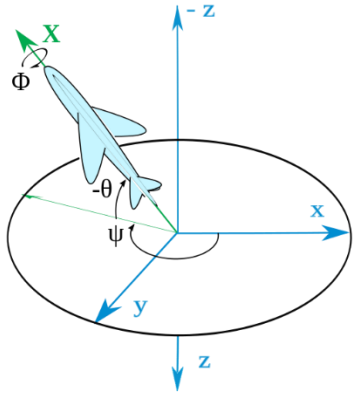
3D rotation

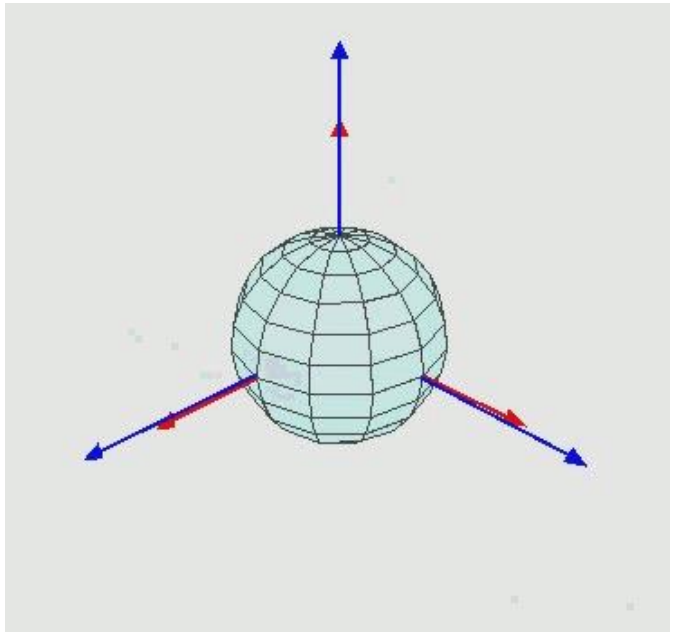
- Formalisms and example uses
  - Euler angles: platform or gimbal orientation (e.g., yaw-pitch-roll)
  - Angle-axis (Euler axis and angle): nonlinear optimization, robotics
  - Quaternion: many compositions of rotations (e.g., game engines)
  - Rotation matrix: everywhere else (and the above)

3D rotation, Euler angles

- 3 parameters (3 angles)
- A sequence of 3 elemental rotations
- 12 possible sequences

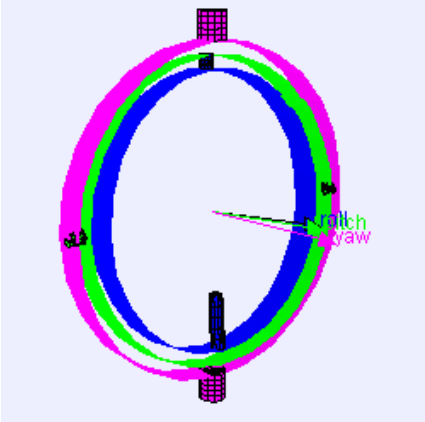
X-Y-X	Y-X-Y	Z-X-Y	
X-Y-Z	Y-X-Z	Z-X-Z	Euler Angles
X-Z-X	Y-Z-X	Z-Y-X	Tait-Bryan Angles
X-Z-Y	Y-Z-Y	Z-Y-Z	



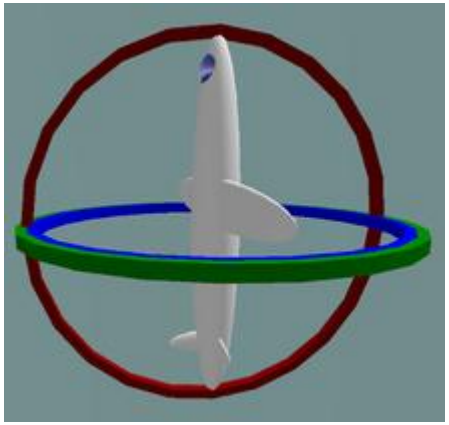


3D rotation, Euler angles

- Gimbal lock
  - Two of the three gimbals are in the same plane
  - One degree of freedom is lost
    - "Locked" into 2D rotation



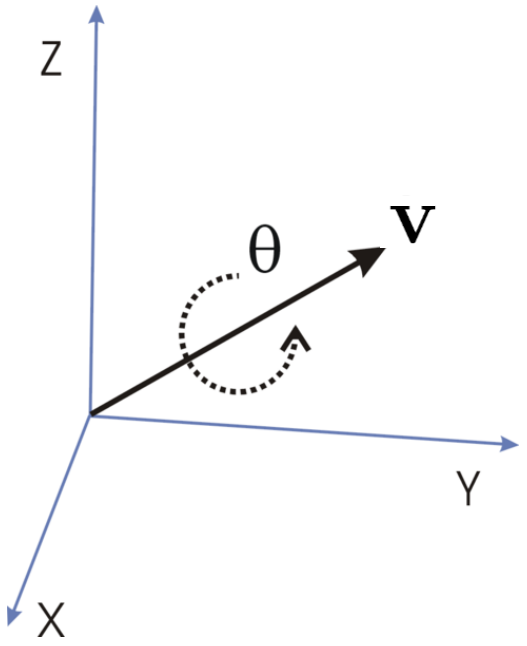
No gimbal lock



Gimbal lock

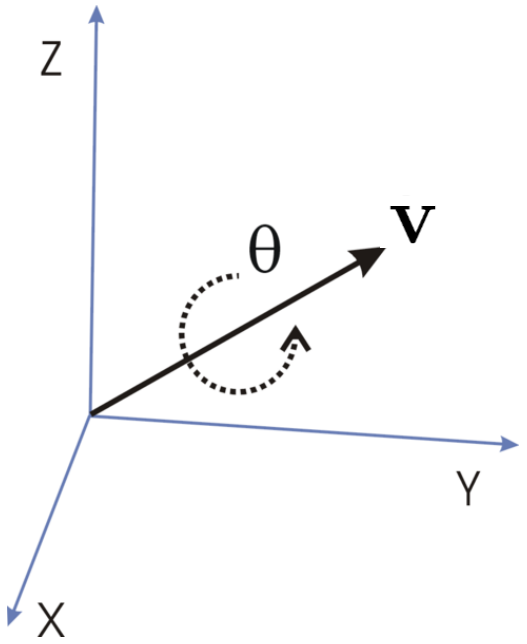
3D rotation, angle-axis representation

- Euler's rotation theorem
  - Any rotation of a rigid body in 3D is equivalent to a pure rotation about a single fixed axis
- 3 parameters, 3 degrees of freedom
  - **Minimal parameterization**
  - Axis of rotation defined by a unit 3-vector (2 degrees of freedom) and the angle of rotation about the axis (1 degree of freedom)



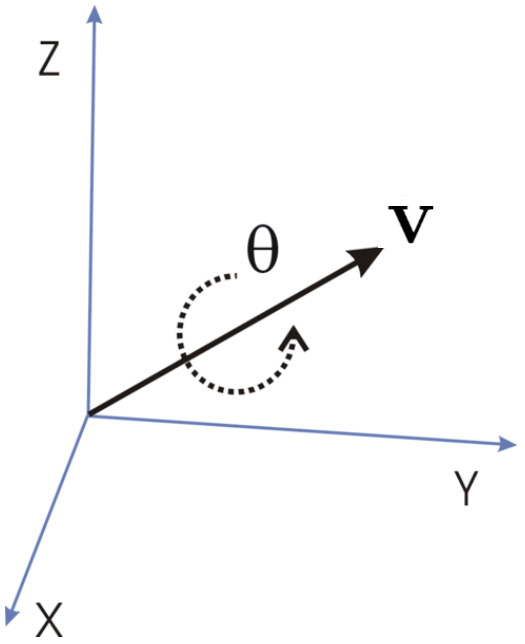
3D rotation, angle-axis representation

- Invert rotation by negating angle-axis coordinates
- Interpolation between 3D rotations
  - Spherical linear interpolation (Slerp)
    - Interpolate rotation through the angle about the axis



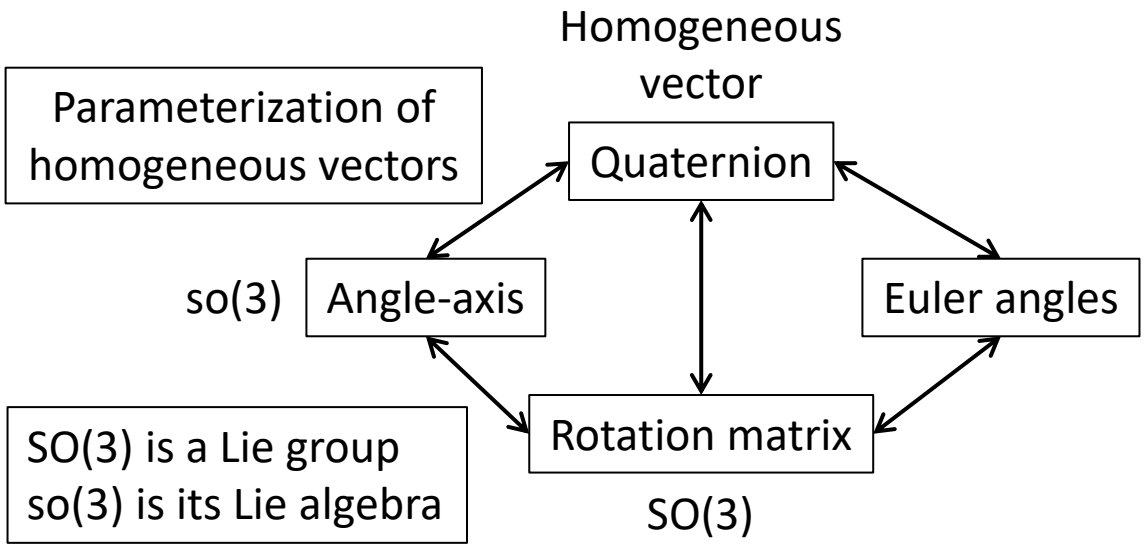
3D rotation, quaternion representation

- Euler's rotation theorem
  - Any rotation of a rigid body in 3D is equivalent to a pure rotation about a single fixed axis
- 4 parameters, 3 degrees of freedom
  - Homogeneous vector (defined up to nonzero scale)
    - Real part and imaginary part



3D rotation

- Conversions between formalisms



Best practice: Do not perform calculations using Euler angles. Only use them for storage, data transfer, or user interface.

Angle-axis representation

- Rotation matrix to angle-axis representation

The skew symmetric matrix form of the angle-axis representation $[\boldsymbol{\omega}]_\times$ corresponding to the rotation matrix $\mathtt{R}$ is the matrix logarithm of $\mathtt{R}$.

$$[\boldsymbol{\omega}]_\times = \ln(\mathtt{R})$$

$$\begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} = \ln\left(\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\right)$$

Angle-axis representation

The angle-axis representation is calculated from the unit vector axis $\mathbf{v}$ and rotation angle θ about $\mathbf{v}$ as

$$\boldsymbol{\omega} = \theta \mathbf{v}$$

where

$$(\mathtt{R} - \mathtt{I})\mathbf{v} = \mathbf{0}, \text{ solve for } \mathbf{v}$$

(i.e., $\mathbf{v}$ is the null space of $(\mathtt{R} - \mathtt{I})$) and

$$\theta = \tan^{-1}\left(\frac{\sin(\theta)}{\cos(\theta)}\right), \text{ where } \cos(\theta) = \frac{\mathrm{Tr}(\mathtt{R}) - 1}{2} \text{ and } \sin(\theta) = \frac{\mathbf{v}^\top \hat{\mathbf{v}}}{2}, \text{ where } \hat{\mathbf{v}} = \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

then, if $\|\boldsymbol{\omega}\| > \pi$, normalize by

$$\boldsymbol{\omega} = \left(1 - \frac{2\pi}{\|\boldsymbol{\omega}\|}\left\lceil \frac{\|\boldsymbol{\omega}\| - \pi}{2\pi} \right\rceil\right)\boldsymbol{\omega}$$

For no or very small rotations

$$\boldsymbol{\omega} = \frac{1}{2}\hat{\mathbf{v}}$$

No or very small rotation if null space is more than 1 column

Angle-axis representation

- Angle-axis representation to rotation matrix

The rotation matrix $\mathtt{R}$ corresponding to the angle-axis representation $\boldsymbol{\omega}$ is the matrix exponent of $[\boldsymbol{\omega}]_\times$.

$$\mathtt{R} = \exp([\boldsymbol{\omega}]_\times)$$

$$\mathtt{R} = \cos(\theta)\mathtt{I} + \operatorname{sinc}(\theta)[\boldsymbol{\omega}]_\times + \frac{1-\cos(\theta)}{\theta^2}\boldsymbol{\omega}\boldsymbol{\omega}^\top, \text{ where } \theta = \|\boldsymbol{\omega}\|$$

For no or very small rotations

$$\mathtt{R} = \mathtt{I} + [\boldsymbol{\omega}]_\times$$

No or very small rotation
if ratio is not a number

Nonlinear estimation of normalized camera projection matrix using the Levenberg-Marquardt algorithm

Parameter vector

$$\underset{6 \text{ x } 1}{(\hat{\boldsymbol{\omega}}^\top, \hat{\mathbf{t}}^\top)^\top}$$

Mapping

$$\hat{\tilde{\mathbf{x}}}_i = [\exp([\hat{\boldsymbol{\omega}}]_\times) \,|\, \hat{\mathbf{t}}] \begin{bmatrix} \tilde{\mathbf{X}}_i \\ 1 \end{bmatrix} = \exp([\hat{\boldsymbol{\omega}}]_\times)\tilde{\mathbf{X}}_i + \hat{\mathbf{t}} \,\forall\, i$$

Measurement vector

$$\underset{2n \text{ x } 1}{\left(\hat{\tilde{\mathbf{x}}}_1^\top, \hat{\tilde{\mathbf{x}}}_2^\top, \ldots, \hat{\tilde{\mathbf{x}}}_n^\top\right)^\top} \text{ with associated covariance matrices } \Sigma_{\hat{\tilde{\mathbf{x}}}_1}, \Sigma_{\hat{\tilde{\mathbf{x}}}_2}, \ldots, \Sigma_{\hat{\tilde{\mathbf{x}}}_n}$$

Cost

$$\sum_i \boldsymbol{\epsilon}_i^\top \Sigma_{\hat{\tilde{\mathbf{x}}}_i}^{-1} \boldsymbol{\epsilon}_i, \text{ where } \boldsymbol{\epsilon}_i = \hat{\tilde{\mathbf{x}}}_i - \hat{\hat{\tilde{\mathbf{x}}}}_i$$

Jacobian

$$\mathtt{J} = \underset{2n \text{ x } 6}{\begin{bmatrix} \mathtt{A}_1 \\ \mathtt{A}_2 \\ \vdots \\ \mathtt{A}_n \end{bmatrix}}, \text{ where } \mathtt{A}_i = \frac{\partial \hat{\hat{\tilde{\mathbf{x}}}}_i}{\partial(\hat{\boldsymbol{\omega}}^\top, \hat{\mathbf{t}}^\top)} = \begin{bmatrix} \underset{2 \text{ x } 3}{\frac{\partial \hat{\hat{\tilde{\mathbf{x}}}}_i}{\partial \hat{\boldsymbol{\omega}}}} & \underset{2 \text{ x } 3}{\frac{\partial \hat{\hat{\tilde{\mathbf{x}}}}_i}{\partial \hat{\mathbf{t}}}} \end{bmatrix}$$

But, use data normalization

Projection of a point under the normalized camera projection matrix

The inhomogeneous 3D point $\tilde{\mathbf{X}}$ is projected to the homogeneous 2D point in normalized coordinates $\hat{\mathbf{x}}$ under the normalized camera projection matrix $\hat{\mathtt{P}} = [\exp([\boldsymbol{\omega}]_\times) \,|\, \mathbf{t}] = [\mathtt{R} \,|\, \mathbf{t}]$, where $\mathtt{R} = \exp([\boldsymbol{\omega}]_\times)$, by

$$\hat{\mathbf{x}} = [\exp([\boldsymbol{\omega}]_\times) \,|\, \mathbf{t}] \begin{bmatrix} \tilde{\mathbf{X}} \\ 1 \end{bmatrix}$$

$$\hat{\mathbf{x}} = \exp([\boldsymbol{\omega}]_\times)\tilde{\mathbf{X}} + \mathbf{t}$$

$$\hat{\mathbf{x}} = \tilde{\mathbf{X}}_{\text{rotated}} + \mathbf{t}$$

where $\tilde{\mathbf{X}}_{\text{rotated}} = \exp([\boldsymbol{\omega}]_\times)\tilde{\mathbf{X}} = \mathtt{R}\tilde{\mathbf{X}}$. Dehomogenizing the 2D point results in the mapping $\tilde{\mathbf{X}} \mapsto \hat{\tilde{\mathbf{x}}}$. For this mapping

$$\frac{\partial \hat{\tilde{\mathbf{x}}}}{\partial \boldsymbol{\omega}} = \frac{\partial \hat{\tilde{\mathbf{x}}}}{\partial \tilde{\mathbf{X}}_{\text{rotated}}} \frac{\partial \tilde{\mathbf{X}}_{\text{rotated}}}{\partial \boldsymbol{\omega}}$$

$$\frac{\partial \hat{\tilde{\mathbf{x}}}}{\partial \mathbf{X}} = \frac{\partial \hat{\tilde{\mathbf{x}}}}{\partial \tilde{\mathbf{X}}_{\text{rotated}}} \frac{\partial \tilde{\mathbf{X}}_{\text{rotated}}}{\partial \tilde{\mathbf{X}}}$$

Projection of a point under the normalized camera projection matrix

$$\frac{\partial \hat{\tilde{\mathbf{x}}}}{\partial \boldsymbol{\omega}} = \frac{\partial \hat{\tilde{\mathbf{x}}}}{\partial \tilde{\mathbf{X}}_{\text{rotated}}} \frac{\partial \tilde{\mathbf{X}}_{\text{rotated}}}{\partial \boldsymbol{\omega}}$$

$$\frac{\partial \hat{\tilde{\mathbf{x}}}}{\partial \mathbf{X}} = \frac{\partial \hat{\tilde{\mathbf{x}}}}{\partial \tilde{\mathbf{X}}_{\text{rotated}}} \frac{\partial \tilde{\mathbf{X}}_{\text{rotated}}}{\partial \tilde{\mathbf{X}}}$$

where

$$\frac{\partial \hat{\tilde{\mathbf{x}}}}{\partial \tilde{\mathbf{X}}_{\text{rotated}}} = \begin{bmatrix} 1/\hat{w} & 0 & -\hat{\tilde{x}}/\hat{w} \\ 0 & 1/\hat{w} & -\hat{\tilde{y}}/\hat{w} \end{bmatrix}$$

where $\hat{w} = \tilde{Z}_{\text{rotated}} + t_3$ and $\frac{\partial \tilde{\mathbf{X}}_{\text{rotated}}}{\partial \tilde{\mathbf{X}}} = \exp([\boldsymbol{\omega}]_\times) = \mathtt{R}$, and

$$\frac{\partial \hat{\tilde{\mathbf{x}}}}{\partial \mathbf{t}} = \begin{bmatrix} 1/\hat{w} & 0 & -\hat{\tilde{x}}/\hat{w} \\ 0 & 1/\hat{w} & -\hat{\tilde{y}}/\hat{w} \end{bmatrix}$$

where $\hat{w} = \mathbf{r}^{3\top} \tilde{\mathbf{X}} + t_3$.

Angle-axis representation, rotate point

The inhomogeneous 3D point $\tilde{\mathbf{X}}$ is rotated to $\tilde{\mathbf{X}}_{\text{rotated}}$ under the angle-axis representation $\boldsymbol{\omega}$ by

$$\tilde{\mathbf{X}}_{\text{rotated}} = \exp([\boldsymbol{\omega}]_\times)\tilde{\mathbf{X}}$$

$$\tilde{\mathbf{X}}_{\text{rotated}} = \begin{cases} \tilde{\mathbf{X}} + \boldsymbol{\omega} \times \tilde{\mathbf{X}} & \text{if } \theta \text{ is 0 or nearly 0} \\ \tilde{\mathbf{X}} + \text{sinc}(\theta)\boldsymbol{\omega} \times \tilde{\mathbf{X}} + \frac{1-\cos(\theta)}{\theta^2}\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \tilde{\mathbf{X}}) & \text{otherwise} \end{cases}$$

Very small rotation if ratio is not a number

where $\theta = \|\boldsymbol{\omega}\|$. For this rotation

$$\frac{\partial \tilde{\mathbf{X}}_{\text{rotated}}}{\partial \boldsymbol{\omega}} = \begin{cases} [-\tilde{\mathbf{X}}]_\times & \text{if } \theta \text{ is 0 or nearly 0} \\ \text{sinc}(\theta)[-\tilde{\mathbf{X}}]_\times + (\boldsymbol{\omega} \times \tilde{\mathbf{X}})\frac{\text{d}\,\text{sinc}(\theta)}{\text{d}\theta}\frac{\text{d}\theta}{\partial \boldsymbol{\omega}} & \\ + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \tilde{\mathbf{X}})\frac{\text{d}s}{\text{d}\theta}\frac{\text{d}\theta}{\partial \boldsymbol{\omega}} + s([\boldsymbol{\omega}]_\times[-\tilde{\mathbf{X}}]_\times + [-(\boldsymbol{\omega} \times \tilde{\mathbf{X}})]_\times) & \text{otherwise} \end{cases}$$

where

$$s = \frac{1-\cos(\theta)}{\theta^2} \text{ and } \frac{\text{d}s}{\text{d}\theta} = \frac{\theta\sin(\theta) - 2(1-\cos(\theta))}{\theta^3}$$

and

$$\frac{\text{d}\theta}{\partial \boldsymbol{\omega}} = \frac{1}{\theta}\boldsymbol{\omega}^\top \text{ and } \frac{\partial \tilde{\mathbf{X}}_{\text{rotated}}}{\partial \tilde{\mathbf{X}}} = \exp([\boldsymbol{\omega}]_\times)$$

Very small rotation if ratio is not a number

Nonlinear estimation of normalized camera projection matrix using the Levenberg-Marquardt algorithm

Normal equations matrix

$$\underset{6 \text{ x } 6}{\mathtt{U}} = \mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \mathtt{J}, \text{ where } \Sigma_{\mathbf{X}}^{-1} = \underset{2n \text{ x } 2n}{\begin{bmatrix} \Sigma_{\hat{\hat{\mathbf{x}}}_1}^{-1} & & & \\ & \Sigma_{\hat{\hat{\mathbf{x}}}_2}^{-1} & & \\ & & \ddots & \\ & & & \Sigma_{\hat{\hat{\mathbf{x}}}_n}^{-1} \end{bmatrix}}$$

Normal equations vector

$$\underset{6 \text{ x } 1}{\boldsymbol{\epsilon}_\mathbf{a}} = \mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \underset{2n \text{ x } 1}{\boldsymbol{\epsilon}}, \text{ where } \boldsymbol{\epsilon} = \mathbf{X} - \hat{\mathbf{X}}$$

where $\mathbf{X} = \left(\hat{\hat{\mathbf{x}}}_1^\top, \hat{\hat{\mathbf{x}}}_2^\top, \ldots, \hat{\hat{\mathbf{x}}}_n^\top\right)^\top$ and $\hat{\mathbf{X}} = \left(\hat{\hat{\mathbf{x}}}_1^\top, \hat{\hat{\mathbf{x}}}_2^\top, \ldots, \hat{\hat{\mathbf{x}}}_n^\top\right)^\top$

Augmented normal equations

$$\mathtt{S} = \mathtt{U}^* = \mathtt{U} + \lambda \mathtt{I} \qquad 6 \text{ x } 6$$

$$\mathbf{e} = \boldsymbol{\epsilon}_\mathbf{a} \qquad 6 \text{ x } 1$$

$$\mathtt{S}\boldsymbol{\delta}_\mathbf{a} = \mathbf{e}, \text{ solve for } \boldsymbol{\delta}_\mathbf{a} \qquad 6 \text{ x } 1$$

Candidate parameter vector

$$(\hat{\boldsymbol{\omega}}_0^\top, \hat{\mathbf{t}}_0^\top)^\top = (\hat{\boldsymbol{\omega}}^\top, \hat{\mathbf{t}}^\top)^\top + \boldsymbol{\delta}_\mathbf{a}$$

Adjustment

Normalize angle-axis representation after adjustment

Map to normalized coordinates with covariance propagation

The image point in pixel coordinates $\mathbf{x}$ is mapped to the image point in normalized coordinates $\hat{\mathbf{x}}$ by

$$\hat{\mathbf{x}} = \mathtt{K}^{-1}\mathbf{x}$$

$$\begin{bmatrix}\hat{\tilde{\mathbf{x}}}\\ 1\end{bmatrix} = \begin{bmatrix}a_{11} & a_{12} & a_{13}\\ 0 & a_{22} & a_{23}\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}\\ 1\end{bmatrix}, \text{ where } \mathtt{K}^{-1} = \begin{bmatrix}a_{11} & a_{12} & a_{13}\\ 0 & a_{22} & a_{23}\\ 0 & 0 & 1\end{bmatrix}$$

$$\hat{\tilde{\mathbf{x}}} = \begin{bmatrix}a_{11} & a_{12}\\ 0 & a_{22}\end{bmatrix}\tilde{\mathbf{x}} + \begin{bmatrix}a_{13}\\ a_{23}\end{bmatrix}$$

Covariance propagation

$$\Sigma_{\hat{\tilde{\mathbf{x}}}} = \mathtt{J}\Sigma_{\tilde{\mathbf{x}}}\mathtt{J}^{\top}, \text{ where } \mathtt{J} = \frac{\partial\hat{\tilde{\mathbf{x}}}}{\partial\tilde{\mathbf{x}}} = \begin{bmatrix}a_{11} & a_{12}\\ 0 & a_{22}\end{bmatrix}$$

If unknown, then assume to be identity

Data normalizing transformation

- 3D

Determine the similarity transformation $\mathtt{U}$ such that the mean (i.e., centroid) of the transformed points is at the origin and their standard deviation from the origin is $\sqrt{3}$.

$$\mathtt{U} = \begin{bmatrix} s & 0 & 0 & -s\mu_{\tilde{X}} \\ 0 & s & 0 & -s\mu_{\tilde{Y}} \\ 0 & 0 & s & -s\mu_{\tilde{Z}} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ where } s = \sqrt{\frac{3}{\sigma^2_{\tilde{X}} + \sigma^2_{\tilde{Y}} + \sigma^2_{\tilde{Z}}}}$$

where $\mu_{\tilde{X}}$ and $\sigma^2_{\tilde{X}}$, $\mu_{\tilde{Y}}$ and $\sigma^2_{\tilde{Y}}$, and $\mu_{\tilde{Z}}$ and $\sigma^2_{\tilde{Z}}$ are the mean and variance of the $\tilde{X}$, $\tilde{Y}$, and $\tilde{Z}$ coordinates, respectively.

Nonlinear estimation of normalized camera projection matrix using the Levenberg-Marquardt algorithm

- Algorithm

1. Camera center $\tilde{\mathbf{C}} = -\mathtt{R}^\top \mathbf{t}$
2. Data normalize the 3D points and camera center

$$\mathbf{X}_{\mathrm{DN},i} = \mathtt{U}\mathbf{X}_i \,\forall\, i$$
$$\mathbf{C}_{\mathrm{DN}} = \mathtt{U}\mathbf{C}$$

3. Data normalize the camera translation $\mathbf{t}_{\mathrm{DN}} = -\mathtt{R}\tilde{\mathbf{C}}_{\mathrm{DN}}$
4. Estimate the camera rotation and data normalized camera translation from the data normalized points
5. Data denormalize the final estimate of the data normalized camera translation

$$\tilde{\mathbf{C}}_{\mathrm{DN}} = -\mathtt{R}^\top \mathbf{t}_{\mathrm{DN}}$$
$$\mathbf{C} = \mathtt{U}^{-1}\mathbf{C}_{\mathrm{DN}}$$
$$\mathbf{t} = -\mathtt{R}\tilde{\mathbf{C}}$$

Next lecture

- Camera pose (calibrated camera), minimum solution
- Outlier rejection
- Reading
  - Haralick et al., “Review and Analysis of Solutions of the Three Point Perspective”
  - Section 4.7