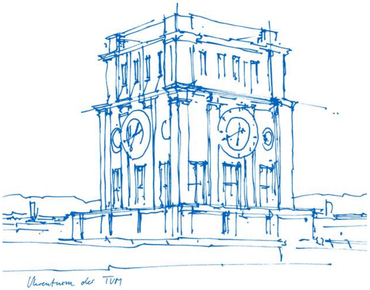
# Computer Vision II: Multiple View Geometry (IN2228)

## Chapter 02 Motion and Scene Representation
## (Part 1 Basic Expression)

Dr. Haoang Li

26 April 2023 12:00-13:30



# Announcements

All the following Exam information are from the Department of Studies.

## ➢ Summer Semester Exam

- Our exam will tentatively take place on **04 August** from **8:00 am** to **10:00 am**
- The registration for these exams is possible **between 22 May and 30 June**
- Deadline for grading of exams: **06 September 2023**

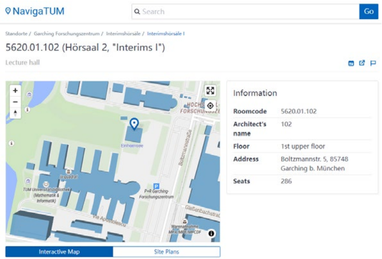
## ➢ Winter Semester Exam (Repeat Exam)

- Currently, the exact date has not been determined.
- Repeat exams will take place **between 02 October and 21 October**.
- The registration for these exams is possible **between 11 September and 25 September**.

# Announcements

Today, we will have the first exercise class.

- ✓ Time: from 16:00 to 18:00
- ✓ Room: 102, Hörsaal 2, "Interims I" (5620.01.102)
- ✓ Detailed content will be provided by teaching assistants.





Sergei Solonets


Daniil Sinitsyn


Viktoria Ehm

# Announcements

## Exam Content

- ✓ If a slide contains the sentence "this knowledge will not be asked in the exam", it means that our exam will not involve this slide.
- ✓ The reason why we prepare these slides are that they may be useful for your future research projects.
- ✓ If necessary, I will prepare a class for knowledge review in the early July (it is not finally determined).
- ✓ Other information about exam content will be released in the further.

# Outline

- Overview
- Coordinate System
- Camera Motion Expression
- 3D Scene Expression

# Overview

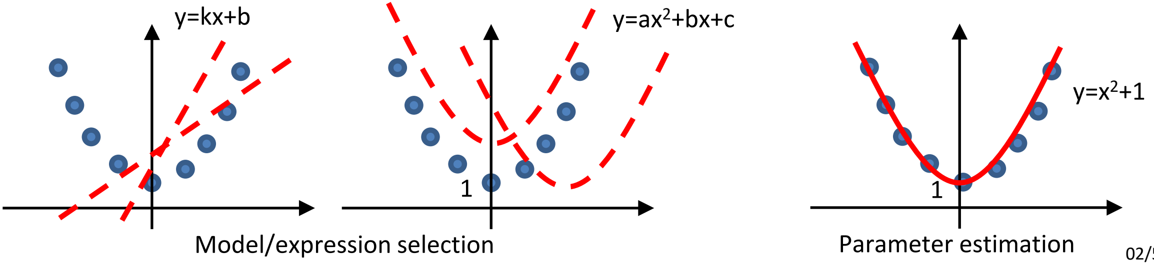
➢ General Pipeline of Solving Multi-view Geometry Problem

✓ Input: A set of observed data

✓ Procedure of problem solving

- Select a suitable model/expression with unknown parameters
- Estimate the parameters by data fitting (do not consider outliers)

✓ Output: Estimated parameters



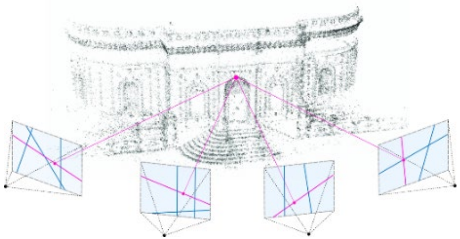
# Overview

➢ Recap on Tasks of Multi-view Geometry

✓ Establish point correspondences (observed data)

✓ Estimate camera motions
✓ Reconstruct 3D structure
} They require knowledge about basic expression of camera motion and 3D structure

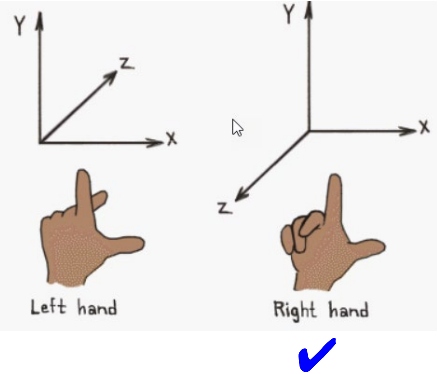
✓ Optimization — It requires knowledge about advanced expression of camera motion, i.e., Lie group and Lie algebra

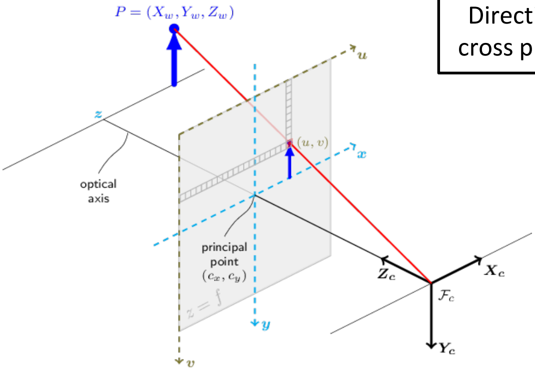


# Coordinate System

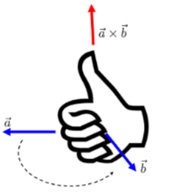
➢ Left-hand and Right-hand Frames

Right-hand XYZ coordinate system is more common in 3D computer vision.



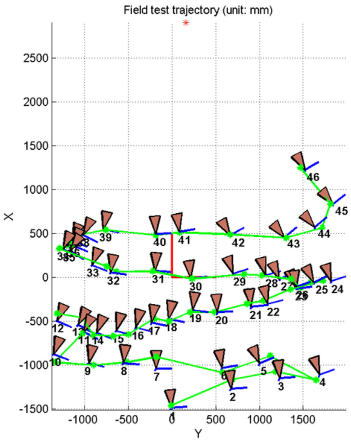

Camera frame

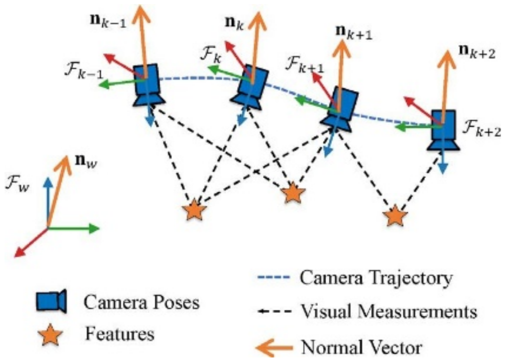



Direction of cross product

# Coordinate System

- Absolute Position

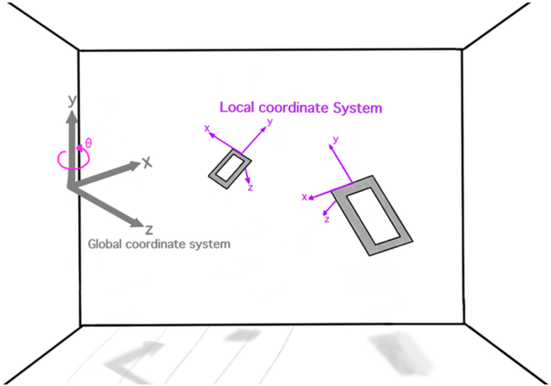


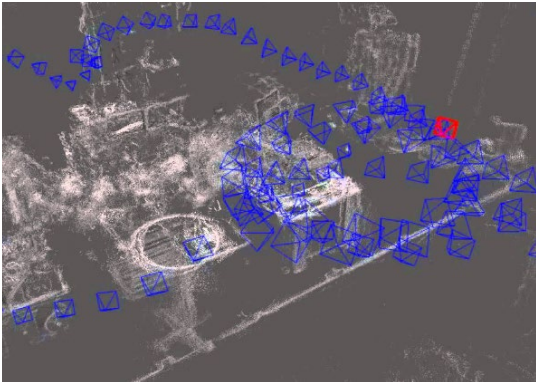




World frame and camera frames in VO/SLAM/SFM (2D case)

# Coordinate System

➢ Absolute Position

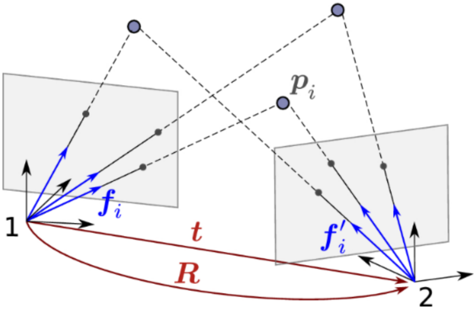


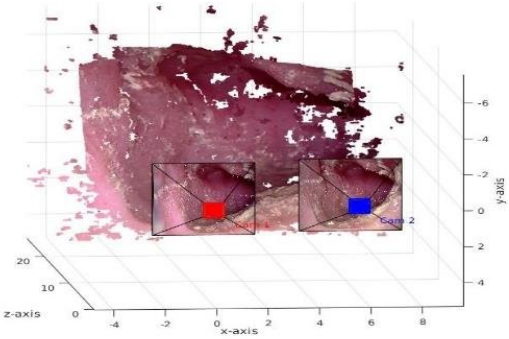



World frame and camera frames in VO/SLAM/SFM (3D case)

# Coordinate System

- Relative Position



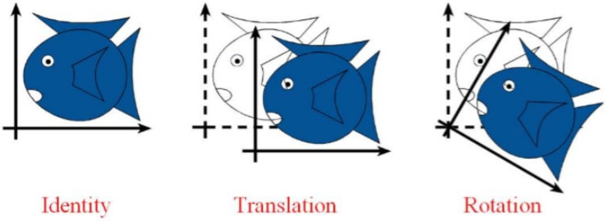




Left and right camera frames in VO/SLAM/SFM

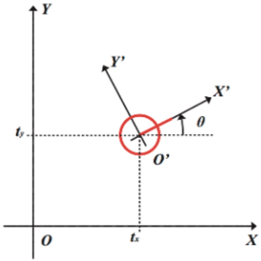
# Camera Motion Expression

➢ Recap on 2D Case

Euclidian/Rigid Transformation




$$\begin{bmatrix} 1 & 0 & X \\ 0 & 1 & Y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$




$$\begin{cases} x_w = x_r \cos\theta - y_r \sin\theta + t_x \\ y_w = x_r \sin\theta + y_r \cos\theta + t_y \end{cases}$$

⇩

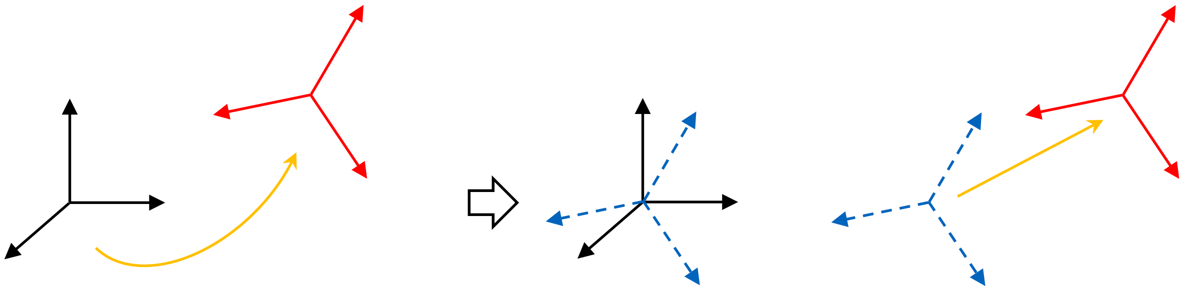
$$\mathbf{x}_w = \mathbf{R}\mathbf{x}_r + \mathbf{t}$$

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad \mathbf{t} = [t_x, t_y]^T$$

# Camera Motion Expression

➢ Rigid Transformation in 3D

Rigid transformation consists of rotation and translation



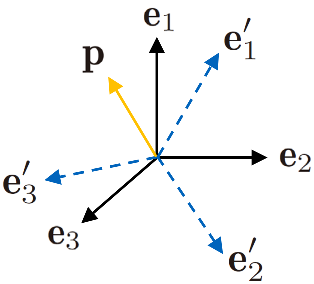
Step 1: Rotation

Step 2: Translation

# Camera Motion Expression

- Rigid Transformation in 3D

For the a 3D point $\boldsymbol{p}$, its coordinates in the world frame $\boldsymbol{p}_W$ and coordinates in the camera frame $\boldsymbol{p}_C$ are different.




Linear expression

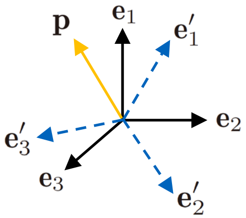
$$[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = [\mathbf{e}'_1, \mathbf{e}'_2, \mathbf{e}'_3] \begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \end{bmatrix}$$

Basis

Coefficients
(3D coordinates)

# Camera Motion Expression

- Rigid Transformation in 3D



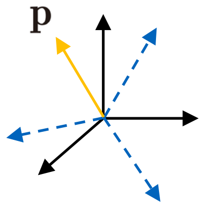

$$[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = [\mathbf{e}_1', \mathbf{e}_2', \mathbf{e}_3'] \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \\ \mathbf{e}_3^T \end{bmatrix} [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \\ \mathbf{e}_3^T \end{bmatrix} [\mathbf{e}_1', \mathbf{e}_2', \mathbf{e}_3'] \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{e}_1^T\mathbf{e}_1' & \mathbf{e}_1^T\mathbf{e}_2' & \mathbf{e}_1^T\mathbf{e}_3' \\ \mathbf{e}_2^T\mathbf{e}_1' & \mathbf{e}_2^T\mathbf{e}_2' & \mathbf{e}_2^T\mathbf{e}_3' \\ \mathbf{e}_3^T\mathbf{e}_1' & \mathbf{e}_3^T\mathbf{e}_2' & \mathbf{e}_3^T\mathbf{e}_3' \end{bmatrix}}_{\text{rotation matrix}} \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix} \triangleq \mathbf{R}\mathbf{a}'$$

# Camera Motion Expression

- Rigid Transformation in 3D




**Rotation (special orthogonal group)**

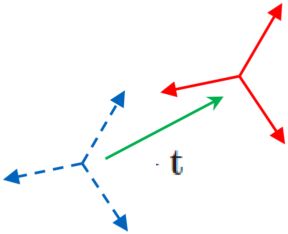
$$\mathbf{a} = \mathbf{R}\mathbf{a}'$$

$$\mathrm{SO}(n) = \{\mathbf{R} \in \mathbb{R}^{n\times n} | \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det(\mathbf{R}) = 1\}$$

$$\mathbf{a}' = \mathbf{R}^{-1}\mathbf{a} = \mathbf{R}^T\mathbf{a}$$

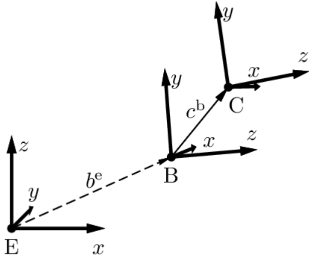
**Translation** $\in \mathbb{R}^3$

$$\mathbf{a}' = \mathbf{R}\mathbf{a} + \mathbf{t}$$



# Camera Motion Expression

➢ Rigid Transformation in 3D




**Multiple transformations**

- Imprecise way

$$\mathbf{b} = \mathbf{R}_1 \mathbf{a} + \mathbf{t}_1, \quad \mathbf{c} = \mathbf{R}_2 \mathbf{b} + \mathbf{t}_2$$

$$\mathbf{c} = \mathbf{R}_2 \left( \mathbf{R}_1 \mathbf{a} + \mathbf{t}_1 \right) + \mathbf{t}_2$$

- More compact way

$$\tilde{\mathbf{b}} = \mathbf{T}_1 \tilde{\mathbf{a}}, \ \tilde{\mathbf{c}} = \mathbf{T}_2 \tilde{\mathbf{b}} \quad \Rightarrow \tilde{\mathbf{c}} = \mathbf{T}_2 \mathbf{T}_1 \tilde{\mathbf{a}}$$

How to achieve this?

# Camera Motion Expression

➢ Rigid Transformation in 3D

**Multiple transformations**

- Homogeneous coordinates

$$\begin{bmatrix} \mathbf{a}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ 1 \end{bmatrix} \triangleq \mathbf{T} \begin{bmatrix} \mathbf{a} \\ 1 \end{bmatrix}$$

- Definition of special Euclidean group

$$\mathrm{SE}(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} | \mathbf{R} \in \mathrm{SO}(3), \mathbf{t} \in \mathbb{R}^3 \right\}$$

# Camera Motion Expression

➢ Rigid Transformation in 3D

## Inverse transformation

- Derivation

$$Y = RX + t \quad \Rightarrow \quad X = R^T(Y - t) = \boxed{R^T} Y \boxed{- R^T t}$$

- Conclusion

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

# Camera Motion Expression

➢ Rigid Transformation in 3D

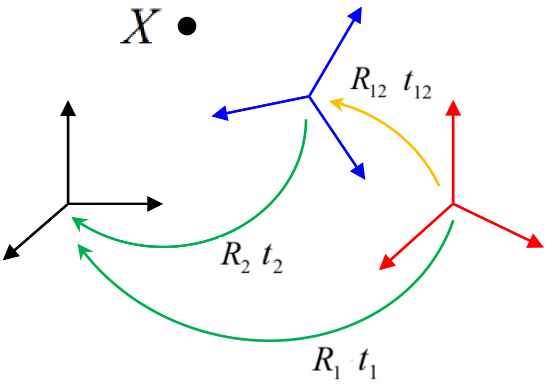
**From absolute poses to relative pose**

- Given absolute poses $(R_1, t_1)$ and $(R_2, t_2)$ , how to compute the relative pose $(R_{12}, t_{12})$ ?

$$X_W = R_1 X_1 + t_1 = R_2 X_2 + t_2$$

$$R_1 X_1 + (t_1 - t_2) = R_2 X_2$$

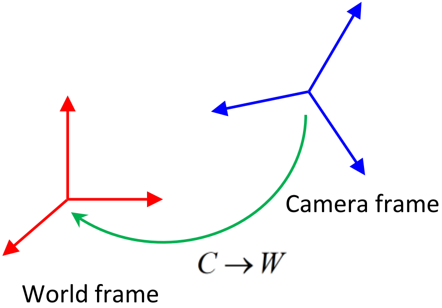
$$\underbrace{R_2^T R_1}_{R_{12}} X_1 + \underbrace{R_2^T (t_1 - t_2)}_{t_{12}} = X_2$$



# Camera Motion Expression

- Rigid Transformation in 3D

**Camera position and translation**




To express the position of a camera in the world frame, which one should we use?

$$(\mathbf{R}_{W \rightarrow C}, \mathbf{t}_{W \rightarrow C})$$

$$(\mathbf{R}_{C \rightarrow W}, \mathbf{t}_{C \rightarrow W}) \checkmark$$

Origin of the camera in the world frame

$$\begin{bmatrix} \mathbf{t} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Origin of the camera in the camera frame

# Camera Motion Expression

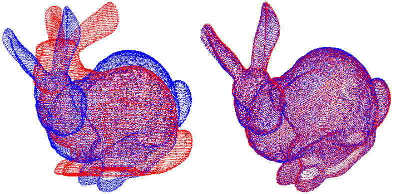
➢ Similarity Transformation in 3D

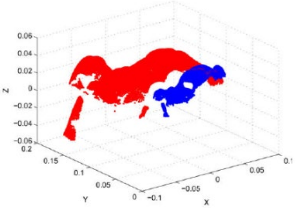
Definition

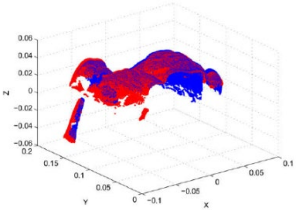
$$\mathrm{SE}(3) \quad \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad \Rightarrow \quad \mathrm{Sim}(3) \quad \mathbf{T}_S = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

6 degrees of freedom (DOF)

7 degrees of freedom (DOF)





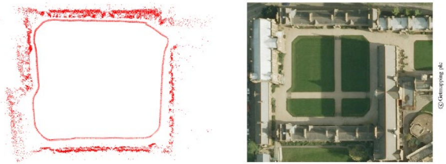
# Camera Motion Expression

➢ Similarity Transformation in 3D

Application of Sim(3)

(a) before optimisation (b) 6 DoF optimisation (c) 7 DoF optimisation

(a) before optimisation (b) 6 DoF optimisation



(c) 7 DoF optimisation (d) aerial photo

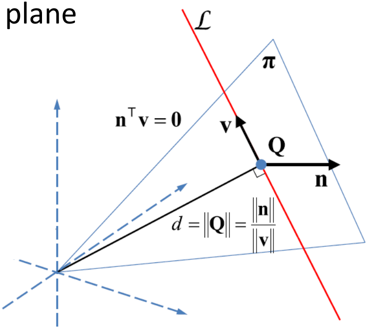
# Camera Motion Expression

- Motion of 3D Line

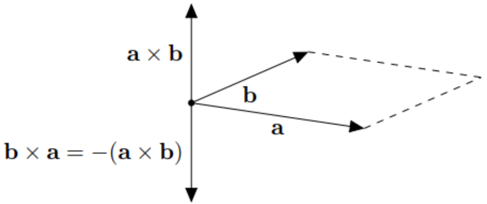
Plücker coordinates

**v**: direction of 3D line (typically a unit vector)
**n**: normal of projection plane

n = **Q** $\times$ **v**
||**n**|| = *d* * ||**v**||




$$\|\mathbf{a} \times \mathbf{b}\| = \|\mathbf{a}\| \, \|\mathbf{b}\| \sin \theta$$

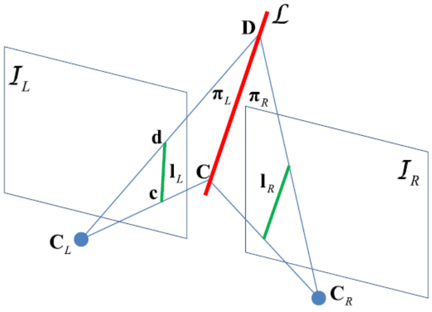


# Camera Motion Expression

- Motion of 3D Line

Plücker coordinates defined by endpoints

$$L = \begin{pmatrix} \bar{l} \\ m \end{pmatrix} = \begin{pmatrix} \frac{\overline{M}}{m} - \frac{\overline{N}}{n} \\ \overline{M} \times \overline{N} \end{pmatrix} = \begin{pmatrix} n\overline{M} - m\overline{N} \\ \overline{M} \times \overline{N} \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix}$$




✓ Homogeneous coordinates is up to scale

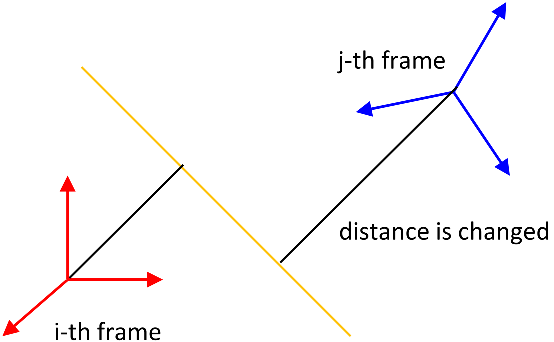
✓ Two directions are orthogonal $a^T b = 0$

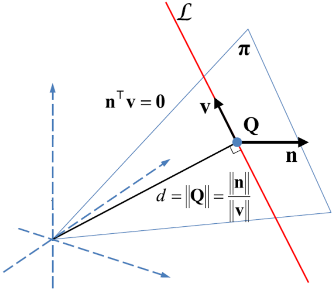
Degrees of freedom: 4

# Camera Motion Expression

➢ Motion of 3D Line

The transformation for the Plücker line coordinates [1]




Norm of **n** is changed

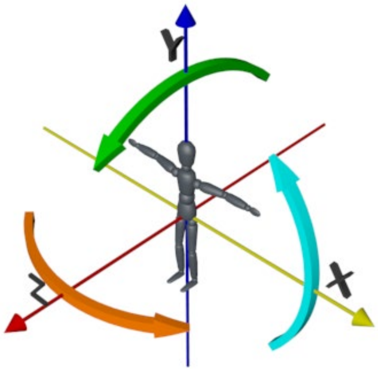
$$\begin{bmatrix} \mathbf{n}_j \\ \mathbf{v}_j \end{bmatrix} = \begin{bmatrix} \mathrm{R}_{ji} & [\mathbf{t}_{ji}]_{\times}\mathrm{R}_{ji} \\ \mathbf{0} & \mathrm{R}_{ji} \end{bmatrix} \begin{bmatrix} \mathbf{n}_i \\ \mathbf{v}_i \end{bmatrix}$$

Norm of **v** is unchanged

[1] A. Bartoli and P. Sturm, “The 3D line motion matrix and alignment of line reconstructions,” in Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recog., 2001, vol. 1, pp. 287–292.

# Camera Motion Expression

## ➢ Rotation Expression




### Common methods

- Rotation matrix
- Euler angles
- Angle-axis (rotation vector)
- Quaternion
- Cayley’s representation

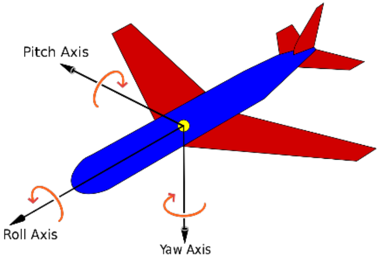
### Relationship

- There is no ideal rotation representation for all purposes
- in some sense, all are equivalent because each representation has an equivalent rotation matrix representation.
- A choice must indeed be made for calculations and coordinate conventions.

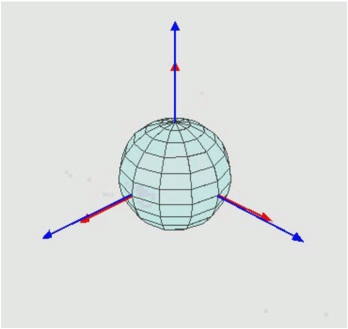
# Camera Motion Expression

➢ Euler Angles

Definition




An intuitive illustration

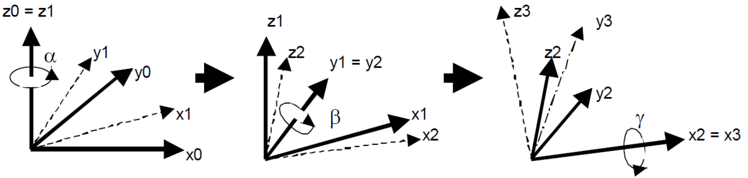


- Intrinsic rotations are w.r.t. axes of a coordinate system XYZ attached to a moving body (i.e. rotation about axis in the current coordinate, like object space).
- Extrinsic rotations are w.r.t. the axes of the fixed coordinate system xyz (i.e. rotation about axis in the the original coordinate, like world space).

# Camera Motion Expression

- Euler Angles

Convert Euler angles to rotation matrix




Euler angles in the ZYX order

$$T_{0,3} = T_{0,1}T_{1,2}T_{2,3} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{bmatrix} =$$

$$\begin{bmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) \\ \sin(\alpha)\cos(\beta) & \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) & \sin(\alpha)\sin(\beta)\cos(\gamma) - \cos(\alpha)\sin(\gamma) \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma) \end{bmatrix}$$

# Camera Motion Expression

- Euler Angles

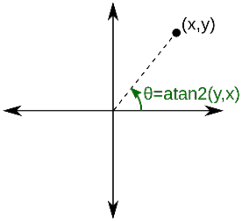
Convert Euler angles to rotation matrix

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\theta_x = atan2(r_{32}, r_{33})$$

$$\theta_y = atan2\left(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}\right)$$

$$\theta_z = atan2(r_{21}, r_{11})$$




atan2(y, x) returns the angle θ between the ray to the point (x, y) and the positive x-axis, confined to (−π, π].

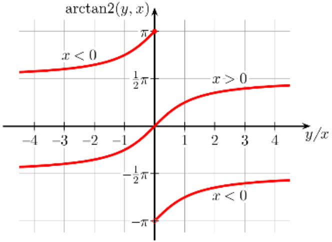
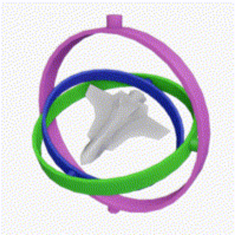



Illustration of atan2

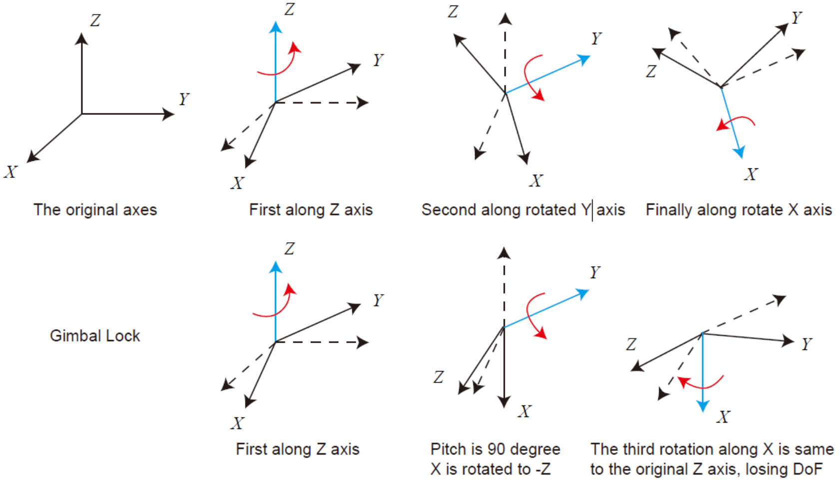
# Camera Motion Expression

➢ Euler Angles

A main limitation (gimbal lock)



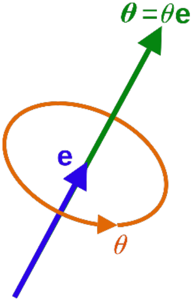
When the pitch (green) and yaw (magenta) gimbals become aligned, changes to roll (blue) and yaw apply the same rotation to the airplane.




The third rotation is using the same axis as the first one

# Camera Motion Expression

➢ Axis–angle Representation (Rotation Vector)

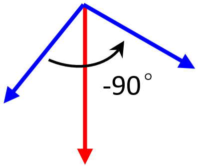



## Definition

The angle **θ** and axis unit vector **e** define a rotation, concisely represented by the rotation vector **θe**.

## Example

$$(\text{axis}, \text{angle}) = \left( \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix}, \theta \right) = \left( \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \frac{-\pi}{2} \right)$$



# Camera Motion Expression

➢ Axis–angle Representation (Rotation Vector)

Convert axis-angle representation to rotation matrix

Rodrigues' rotation formula

$$\mathbf{R} = \cos\theta \mathbf{I} + (1 - \cos\theta)\, \mathbf{n}\mathbf{n}^T + \sin\theta \mathbf{n}^\wedge$$

$$\mathbf{R}(\mathbf{n}, \theta) = \begin{bmatrix} n_x^2 (1 - c\theta) + c\theta & n_x n_y (1 - c\theta) + n_z s\theta & n_x n_z (1 - c\theta) - n_y s\theta \\ n_x n_y (1 - c\theta) - n_z s\theta & n_y^2 (1 - c\theta) + c\theta & n_y n_z (1 - c\theta) + n_x s\theta \\ n_x n_z (1 - c\theta) + n_y s\theta & n_y n_z (1 - c\theta) - n_x s\theta & n_z^2 (1 - c\theta) + c\theta \end{bmatrix}$$

# Camera Motion Expression

➢ Axis–angle Representation (Rotation Vector)

Convert rotation matrix to axis-angle representation

- Rotation angle

$$\begin{aligned}\mathrm{tr}\left(\mathbf{R}\right) &= \cos\theta\,\mathrm{tr}\left(\mathbf{I}\right) + (1-\cos\theta)\,\mathrm{tr}\left(\mathbf{n}\mathbf{n}^T\right) + \sin\theta\,\mathrm{tr}(\mathbf{n}^\wedge) \\ &= 3\cos\theta + (1-\cos\theta) \\ &= 1 + 2\cos\theta.\end{aligned}$$

$$\theta = \arccos\left(\frac{\mathrm{tr}(\mathbf{R})-1}{2}\right)$$

"tr" represents trace of matrix

- Rotation axis

$$\mathbf{n} = \frac{1}{2\sin\theta}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}$$

# Camera Motion Expression

- Quaternion

Definition

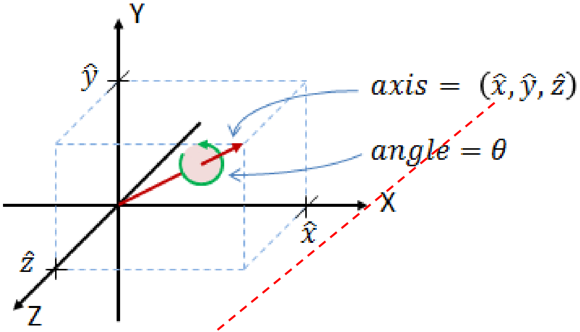
$$\boldsymbol{q} = (q_0, q_1, q_2, q_3)$$

$$q_0 = \cos\left(\frac{\theta}{2}\right)$$

$$q_1 = \hat{x}\sin\left(\frac{\theta}{2}\right)$$

$$q_2 = \hat{y}\sin\left(\frac{\theta}{2}\right)$$

$$q_3 = \hat{z}\sin\left(\frac{\theta}{2}\right)$$




Euler's formula $\mathbf{q} = e^{\frac{\theta}{2}(u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})} = \cos\frac{\theta}{2} + (u_x\mathbf{i}+u_y\mathbf{j}+u_z\mathbf{k})\sin\frac{\theta}{2}$

# Camera Motion Expression

- Quaternion

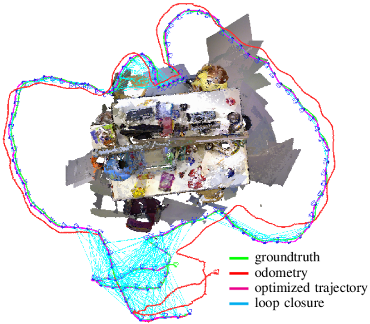
## Application to SLAM

### Ground-truth trajectories

We provide the groundtruth trajectory as a text file containing the translation and orientation of the camera in a fixed coordinate frame. Note that also our automatic evaluation tool expects both the groundtruth and estimated trajectory to be in this format.

- Each line in the text file contains a single pose.
- The format of each line is **'timestamp tx ty tz qx qy qz qw'**
- **timestamp** (float) gives the number of seconds since the Unix epoch.
- **tx ty tz** (3 floats) give the position of the optical center of the color camera with respect to the world origin as defined by the motion capture system.
- **qx qy qz qw** (4 floats) give the orientation of the optical center of the color camera in form of a unit quaternion with respect to the world origin as defined by the motion capture system.
- The file may contain comments that have to start with "#".

https://cvg.cit.tum.de/data/datasets/rgbd-dataset/file_formats




TUM RGB-D SLAM Dataset

# Camera Motion Expression

- Quaternion

Convert quaternion to a matrix rotation

Quaternion

$$\mathbf{q} = q_r + q_i\mathbf{i} + q_j\mathbf{j} + q_k\mathbf{k}$$

$$\mathbf{q} = [\cos\frac{\theta}{2}, \mathbf{n}\sin\frac{\theta}{2}]$$

A 3D point is treated as a quaternion with a real coordinate equal to zero

$$\mathbf{p} = (p_x, p_y, p_z) = p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}$$

$$\mathbf{p} = [0, x, y, z] = [0, \mathbf{v}]$$

# Camera Motion Expression

- Quaternion

Convert quaternion to a matrix rotation

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$$

(Hamilton product)

Real part is zero

$$\mathbf{n}^T(\mathbf{n} \times \mathbf{v}) = 0$$

$$\mathbf{p}' = \mathbf{R}\mathbf{p},$$

$$\mathbf{R} = \begin{bmatrix} 1 - 2s(q_j^2 + q_k^2) & 2s(q_i q_j - q_k q_r) & 2s(q_i q_k + q_j q_r) \\ 2s(q_i q_j + q_k q_r) & 1 - 2s(q_i^2 + q_k^2) & 2s(q_j q_k - q_i q_r) \\ 2s(q_i q_k - q_j q_r) & 2s(q_j q_k + q_i q_r) & 1 - 2s(q_i^2 + q_j^2) \end{bmatrix}$$

$s = 1^{-2} = 1.$ for unit quaternion

# Camera Motion Expression

- Summary

| Representation | Parameters |
|---|---|
| Matrix | $3 \times 3$ matrix $R$ with 9 parameters, with 6 d.o.f. removed via orthogonality constraints. |
| Euler angles: | 3 parameters $(\phi, \theta, \psi)$, in range $[0, 2\pi) \times [-\pi/2, \pi/2] \times [0, 2\pi)$ |
| Axis-angle | 3 + 1 parameters $(\mathbf{a}, \theta)$, in range $S_2 \times [0, \pi)$ with 1 d.o.f. removed via unit vector constraint |
| Rot. vector | 3 parameters $\mathbf{m}$, in range |
| Quaternion | 4 parameters $(q_0, q_1, q_2, q_3)$, with 1 d.o.f. removed via unit quaternion constraint. |

# Camera Motion Expression

➢ Cayley's Representation

✓ Definition

The Cayley transform, which maps any skew-symmetric matrix A to a rotation matrix

$$A \mapsto (I + A)(I - A)^{-1}$$

quaternion

$$q_0 = \cos\left(\frac{\theta}{2}\right)$$
$$q_1 = \hat{x}\sin\left(\frac{\theta}{2}\right)$$
$$q_2 = \hat{y}\sin\left(\frac{\theta}{2}\right)$$
$$q_3 = \hat{z}\sin\left(\frac{\theta}{2}\right)$$

$(\hat{x}, \hat{y}, \hat{z})$ is the rotation axis

$$\begin{bmatrix} 0 & \tan\frac{\theta}{2} \\ -\tan\frac{\theta}{2} & 0 \end{bmatrix} \leftrightarrow \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

2D case

The 180° rotation matrix is excluded, because tan θ⁄2 goes to infinity.

$$\begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \mapsto \frac{1}{1+x^2+y^2+z^2}\begin{bmatrix} 1+x^2-y^2-z^2 & 2xy-2z & 2y+2xz \\ 2xy+2z & 1-x^2+y^2-z^2 & 2yz-2x \\ 2xz-2y & 2x+2yz & 1-x^2-y^2+z^2 \end{bmatrix}.$$

$$x = \hat{x}\tan(\theta / 2)$$

3D case

The 180° rotation matrix is excluded, because tan θ⁄2 goes to infinity.

This knowledge will not be asked in the exam.

# Camera Motion Expression

## ➢ Cayley's Representation

✓ Discussion

- Although in practical applications we can hardly afford to ignore 180° rotations, the Cayley transform is still a potentially useful tool.
- For example, in SLAM, we have prior knowledge of a rough constant velocity motion model. We can leverage this information for disambiguation.
- This rotation parameterization is free of trigonometric functions.
- It has a smaller number of parameters than quaternion.

This knowledge will not be asked in the exam.

# Camera Motion Expression

## ➢ From Representation to Estimation: An Overview

✓ Solution 1: Disentangle translation from rotation

- Generate a linear system with respect to translation.

$$\mathbf{A}\mathbf{t} = \mathbf{b}$$

- Obtain the least-squares solution of translation with respect to rotation.

$$\mathbf{t} = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{b}$$

- Define an objective function with respect to translation.

$$\min_{\mathbf{R},\mathbf{t}} F(\mathbf{R},\mathbf{t}) \triangleq \sum_{i=0}^{m} f_i^2(\mathbf{R},\mathbf{t}) + \sum_{j=0}^{n} g_j^2(\mathbf{R},\mathbf{t}) \Rightarrow \min_{\mathbf{s}} F(\mathbf{s})$$

**s** represents the rotation parameters, e.g., Euler angles

# Camera Motion Expression

## ➢ From Representation to Estimation: An Overview

✓ Solution 1: Disentangle translation from rotation

- Generate a high-order univariate polynomial or multivariate polynomial system

$$\begin{aligned} f_1(x_1, \ldots, x_m) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_m) &= 0, \end{aligned}$$

e.g.,

$$\begin{aligned} x^2 + y^2 - 5 &= 0 \\ xy - 2 &= 0. \end{aligned}$$

$$f(x) = \sum_{k=0}^{8} \delta_k x^k = 0$$

Multivariate polynomial system (lower order in general)

Univariate polynomial system (higher order in general)

- Solvers [1]

Eigenvalue of coefficient matrix

Groebner basis

[1] Ji Zhao, Laurent Kneip, Yijia He, and Jiayi Ma. Minimal Case Relative Pose Computation using Ray-Point-Ray Features. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(5): 1176 - 1190, 2020.

# Camera Motion Expression

## ➢ From Representation to Estimation: An Overview

✓ Solution 2: Simultaneously computing translation and rotation

- Generate a matrix

$$\boldsymbol{x}'^{\top}(\mathbf{R}[\mathbf{t}_{\times}])\boldsymbol{x} = 0 \qquad \boldsymbol{x}'^{\top}\mathbf{E}\boldsymbol{x} = 0$$

Essential matrix

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$$

Projection matrix (simplified by co-planarity constraint)

- Rotation decomposition

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\hat{T} = U \begin{bmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Sigma V^T \qquad \hat{T} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & t_x \\ -t_y & t_x & 0 \end{bmatrix} \Rightarrow \hat{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$\hat{R} = U \begin{bmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T \qquad T = K_2 \hat{t}, \quad R = K_2 \hat{R} K_1^{-1}$$

Singular value decomposition (SVD)

$$\begin{bmatrix} h_{11}^j & h_{12}^j & h_{13}^j \\ h_{21}^j & h_{22}^j & h_{23}^j \\ h_{31}^j & h_{33}^j & h_{33}^j \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11}^j & r_{12}^j & t_1^j \\ r_{21}^j & r_{22}^j & t_2^j \\ r_{31}^j & r_{32}^j & t_3^j \end{bmatrix}$$

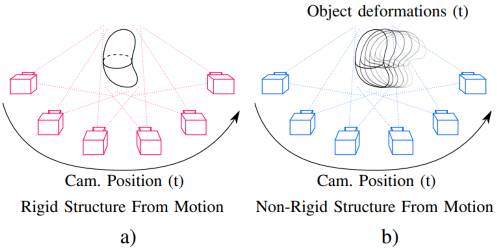
QR decomposition

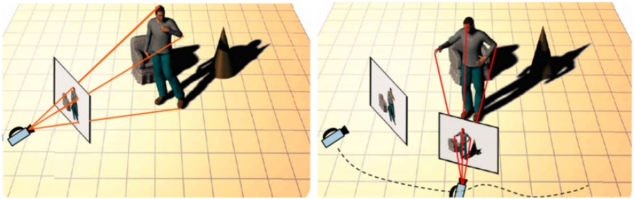
# Camera Motion Expression

- Non-rigid Motion

Comparison between rigid and non-rigid Structure from Motion (SFM)

- ✓ Rigid SFM allowing a reconstruction of the world from different views.
- ✓ Non-rigid SFM implies that both the camera and the scene are both dynamic (time-dependent).



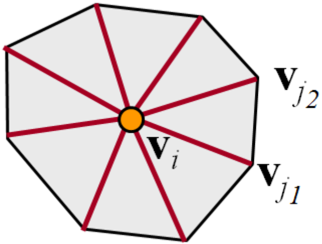



This knowledge will not be asked in the exam.

# Camera Motion Expression

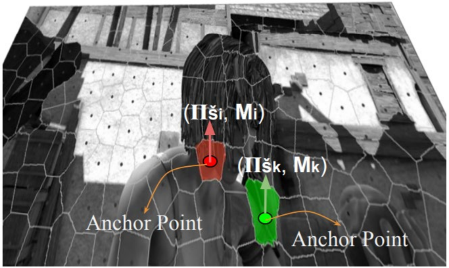
- Non-rigid Motion

One prior constraint: as rigid as possible




Each **edge** basically satisfies the rigid transformation

$$f(p', R) = \sum_i \sum_{j \in \mathcal{N}(i)} w_{ij} ||(p'_i - p'_j) - R_i(p_i - p_j)||^2$$



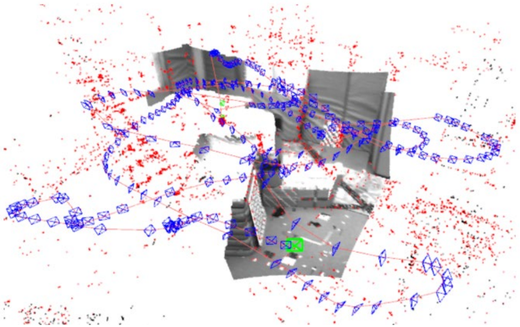

All the points from the same super pixel satisfy the same rigid transformation

This knowledge will not be asked in the exam.

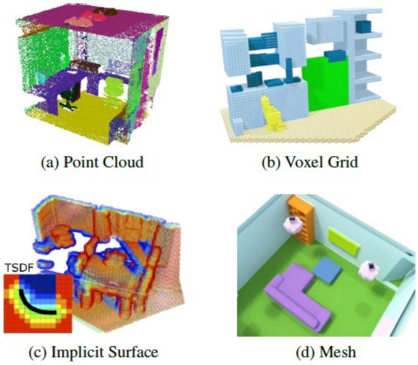
# 3D Scene Representation

➢ Overview

Common 3D representation methods



A 3D map reconstructed by a SLAM method



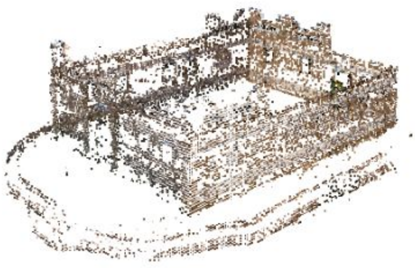

(a) Point Cloud (b) Voxel Grid (c) Implicit Surface (d) Mesh

How to choose appropriate representation?

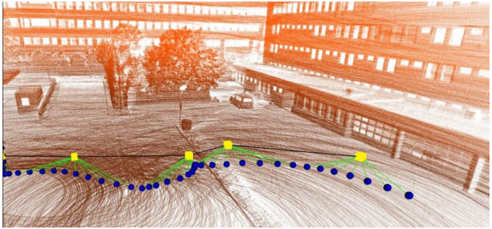
# 3D Scene Representation

➢ Point Cloud

A point cloud is a discrete set of data points in space. The points may represent a 3D shape or object. Each point position has its set of Cartesian coordinates (X, Y, Z).



Point cloud obtained by visual SLAM

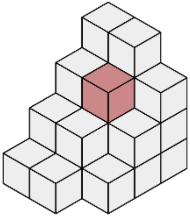


Point cloud obtained by Laser SLAM

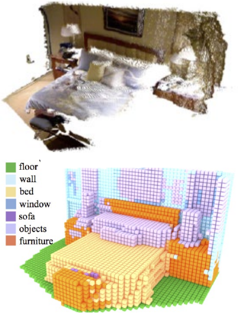
# 3D Scene Representation

➢ Voxel Grid

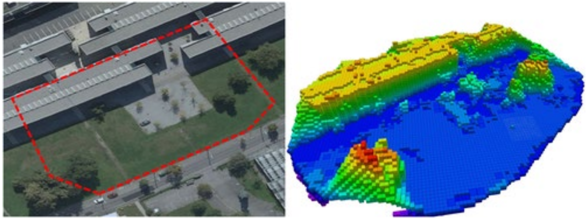
A voxel grid geometry is a 3D grid of values organized into layers of rows and columns. Each row, column, and layer intersection in the grid is called a voxel or small 3D cube.



Voxels




Semantic scene completion

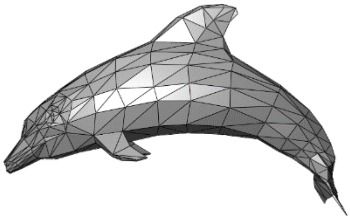


Obstacle map for drones

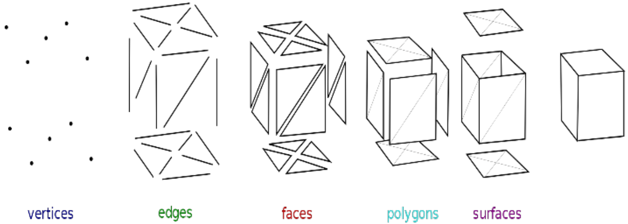
# 3D Scene Representation

➢ Mesh

A polygon mesh is a collection of vertices, edges and faces that defines the shape of a polyhedral object.



A low poly triangle mesh representing a dolphin

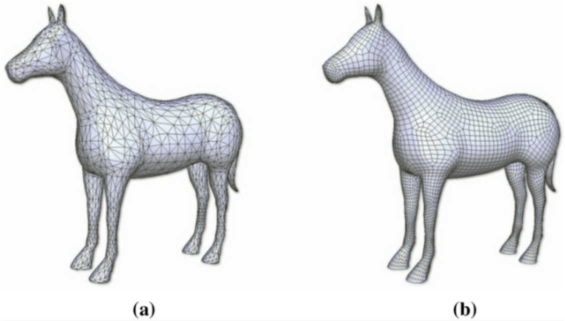



Primitives to define a mesh

# 3D Scene Representation

➢ Mesh

Triangle Mesh vs. Quad Mesh

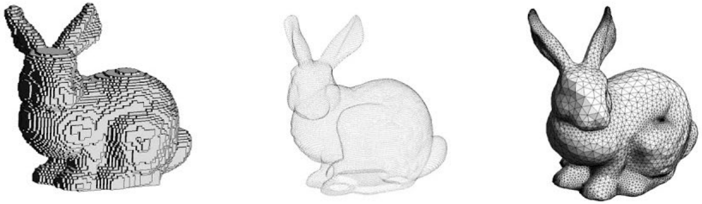


Triangle mesh    Quad mesh

- Triangle mesh type is preferred in the case where geometry function is quite easy and less complex, mostly for regular geometrical shapes.
- Quad mesh give us relatively accurate results, and are more used in complex systems in general.

# 3D Scene Representation

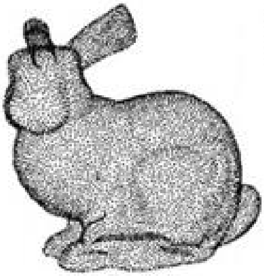
➢ Comparison



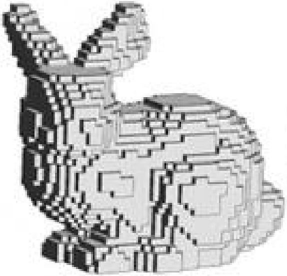
| | Voxel | Point cloud | Polygon mesh |
|---|---|---|---|
| Memory efficiency | Poor | Not good | Good |
| Textures | Not good | No | Yes |
| For neural networks | Easy | Not easy | Not easy |

# 3D Scene Representation

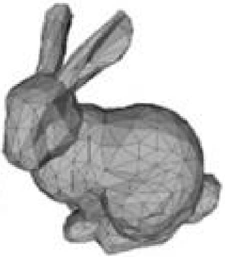
➢ Signed Distance Function (SDF)



Point cloud



Voxel grid



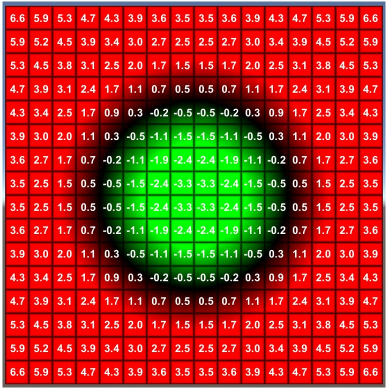
Mesh

SDF

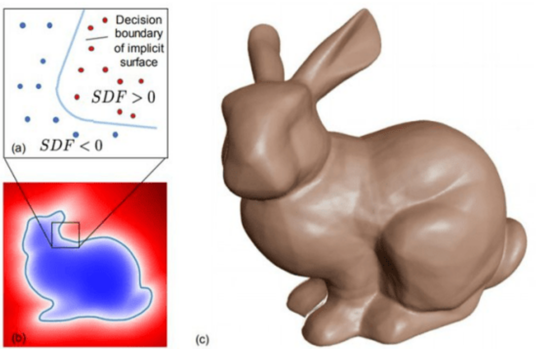
✔

# 3D Scene Representation

- Signed Distance Function (SDF)



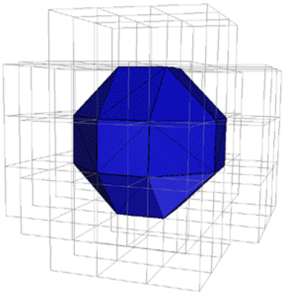

Zero-value SDF isosurface



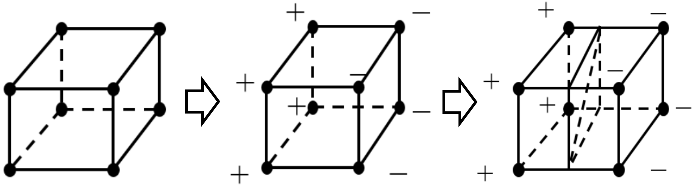

From isosurface to mesh:
Marching cubes algorithm

# 3D Scene Representation

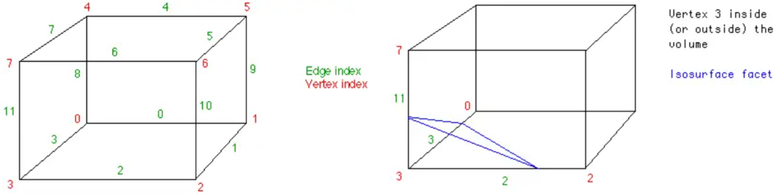
- Signed Distance Function (SDF)



3D space discretization



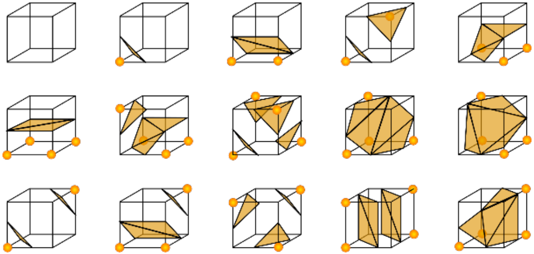

Isosurface detection based on SDF



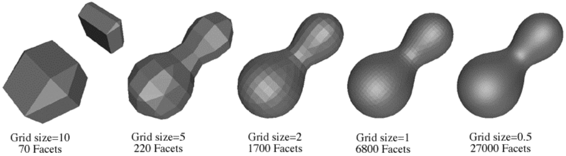

Interpolation of vertices of isosurface

# 3D Scene Representation

- Signed Distance Function (SDF)



15 cube configurations




Higher number of cubes leads to higher resolution of mesh

# 3D Scene Representation

- Signed Distance Function (SDF)

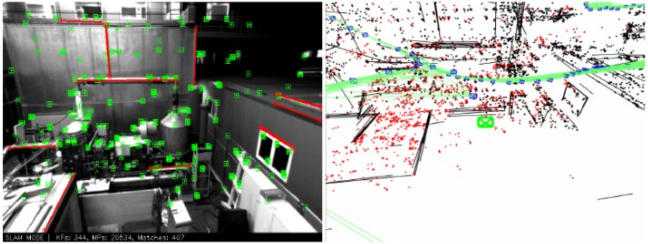
Demo video of DeepSDF [1]

[1] Jeong Joon Park et al., “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”, in CVPR, 2019

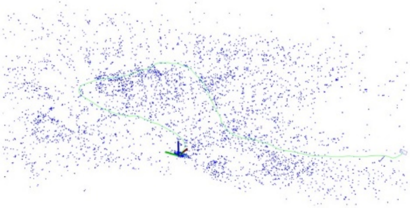
# 3D Scene Representation

- "Line" Cloud


Line cloud




Combination of points and lines



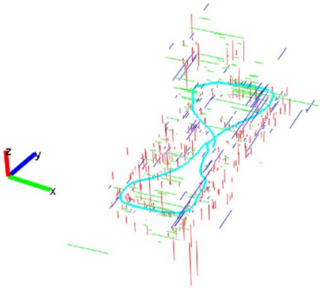
Point cloud

# 3D Scene Representation

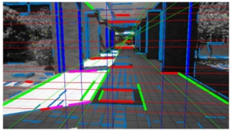
➢ Integrated Information

✓ Structured information

- Parallelism and orthogonality

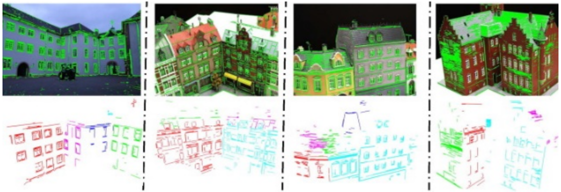



A map composed of structured 3D lines



2D lines clustered by vanishing points

- Co-planarity



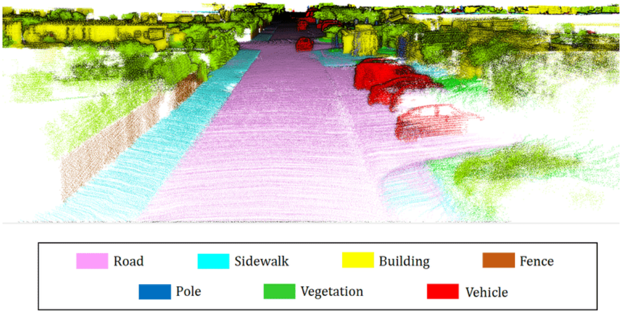
Reconstuctrued 3D maps with coplanar lines

# 3D Scene Representation

➢ Integrated Information

✓ Semantic information


Semantic 2D maps




Semantic 3D maps

# Summary

- Overview
- Coordinate System
- Camera Motion Expression
- 3D Scene Expression

Thank you for your listening!

If you have any questions, please come to me :-)