Practical Course: Vision-based Navigation SS 2018

Lecture 1. Basics

Dr. Jörg Stückler, Dr. Xiang Gao
Vladyslav Usenko, Prof. Dr. Daniel Cremers

Contents

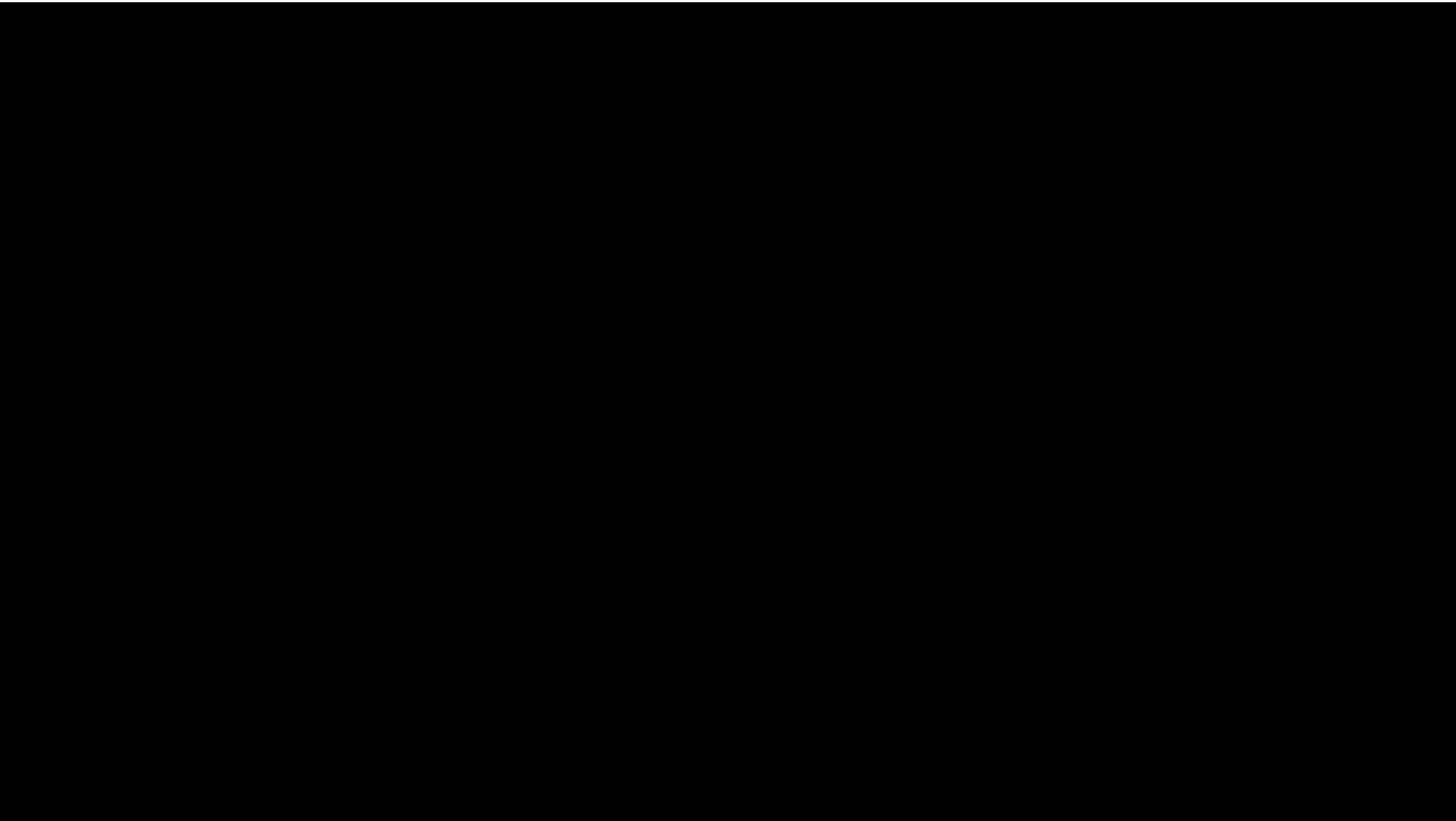
- Course contents and preliminary knowledges
- Framework and mathematic form of a SLAM problem
- 3D geometry

Contents

- Course contents and preliminary knowledges
- Framework and mathematic form of a SLAM problem
- 3D geometry

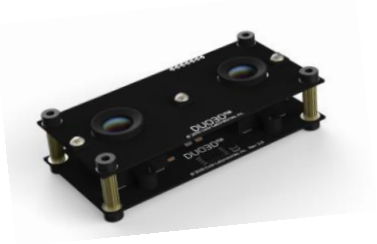
1. Course contents and preliminary knowledges

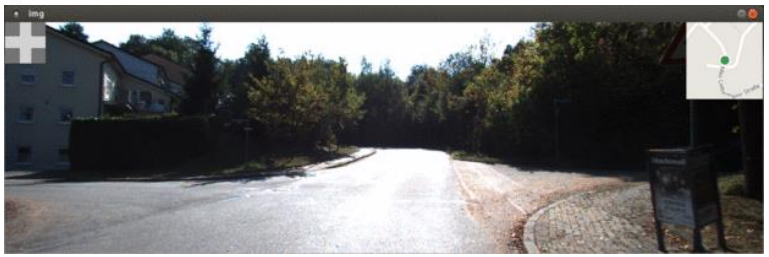
- General overview of computer vision tasks



1. Course contents and preliminary knowledges

- Computer vision





Object detection
Object recognition
Object tracking
Segmentation
…
SLAM

Real world cameras

Image and video sequences

CV tasks

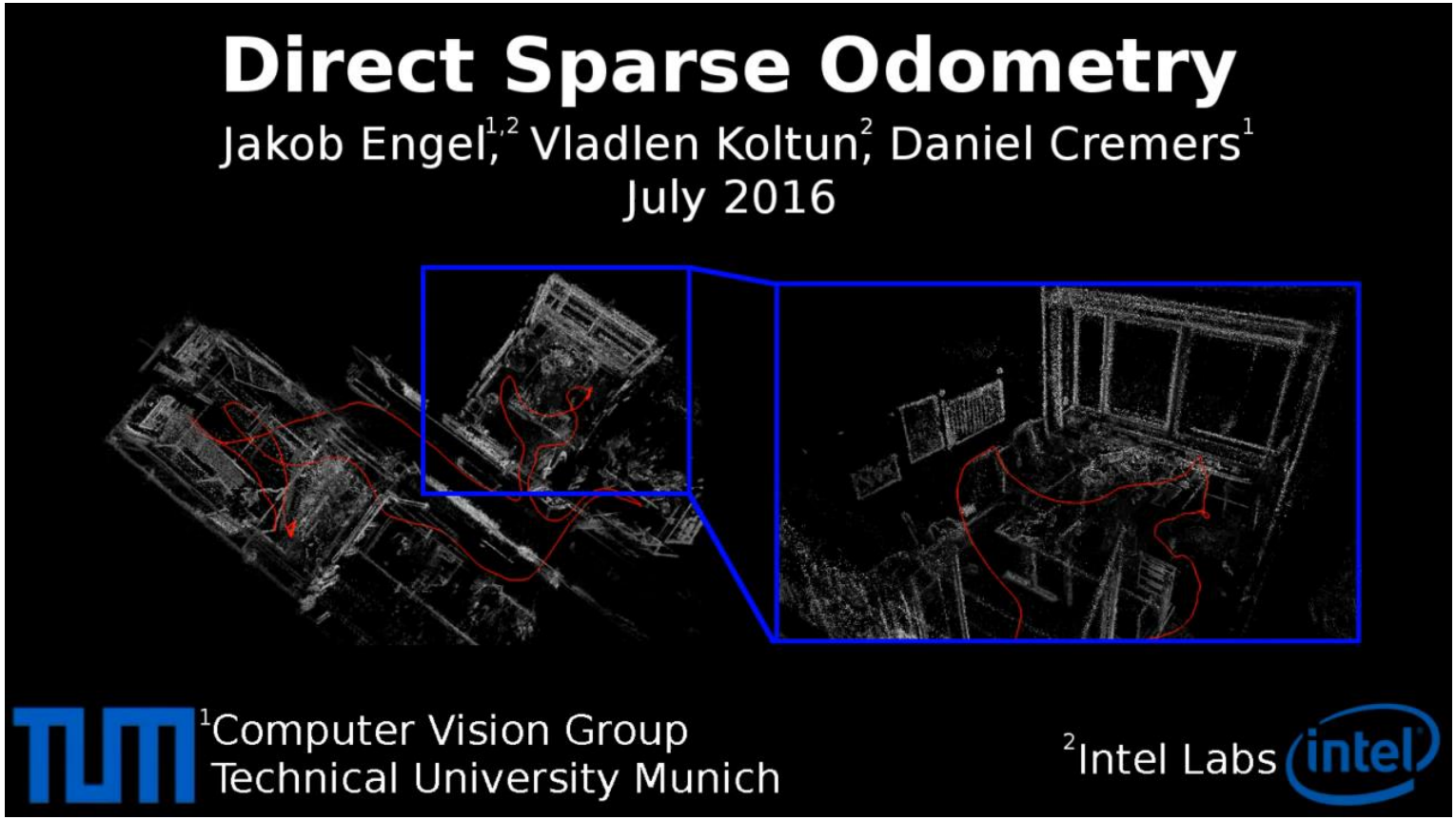
1. Course contents and preliminary knowledges

- What is SLAM?

Indoor/outdoor localization

1. Course contents and preliminary knowledges

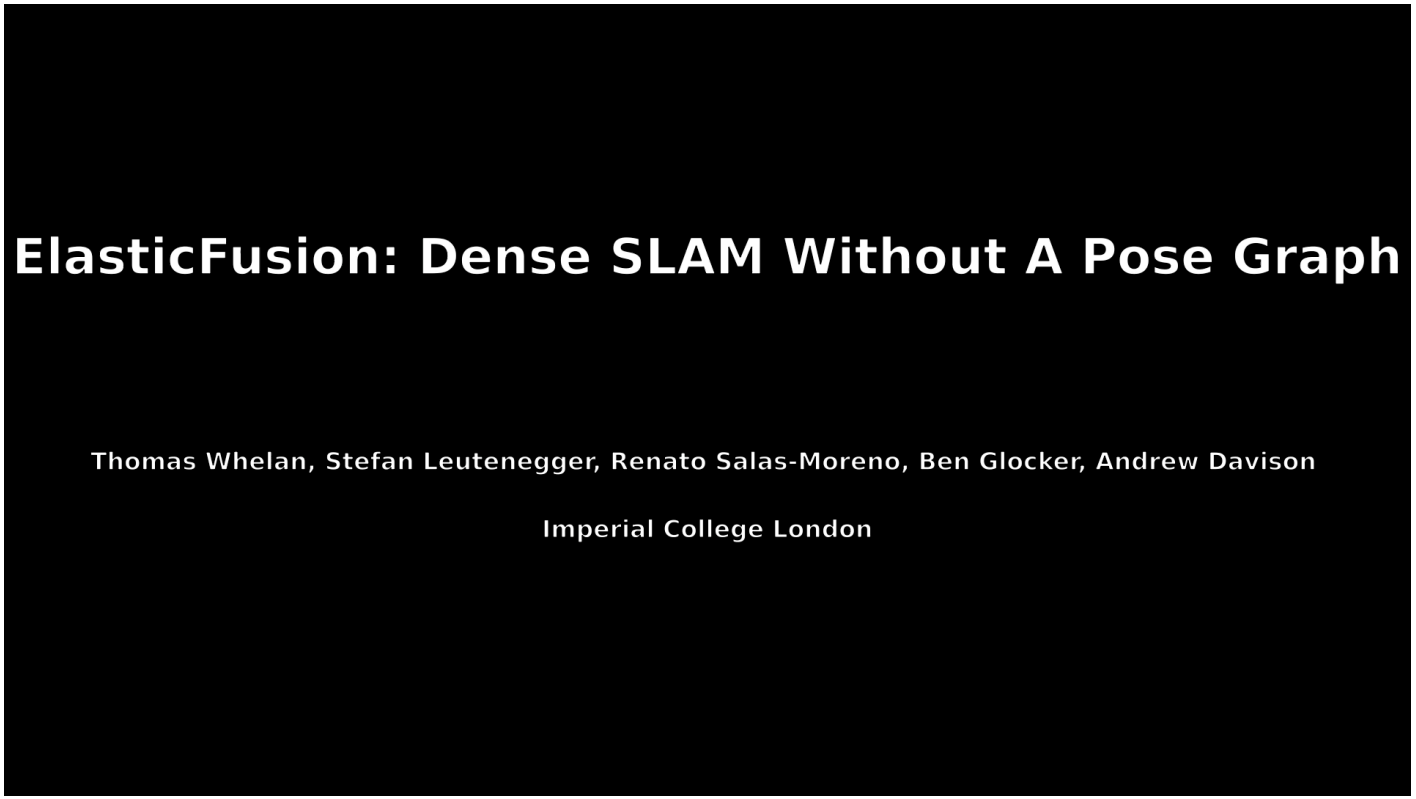
- Computer vision



Dense/semi-dense reconstruction

1. Course contents and preliminary knowledges

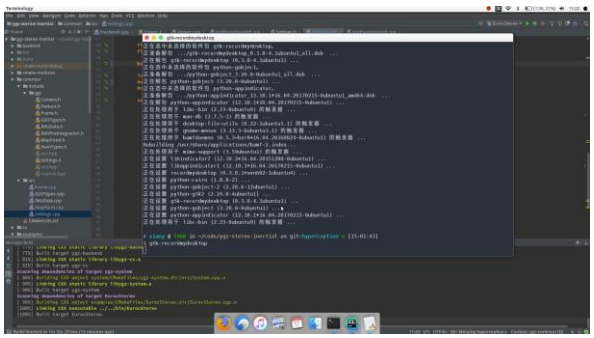
- What is SLAM?



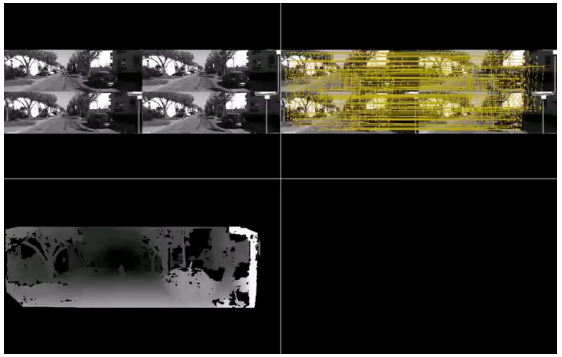
RGB-D dense reconstruction

1. Course contents and preliminary knowledges

- SLAM applications



Hand-held devices

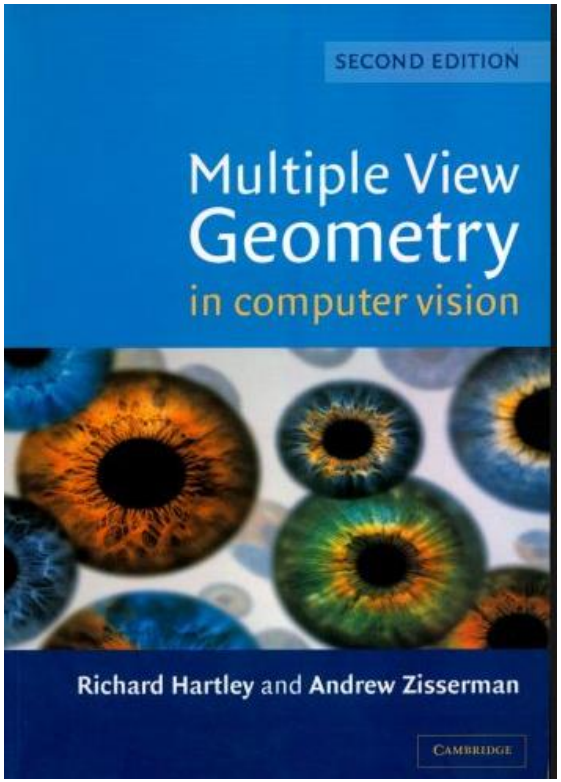


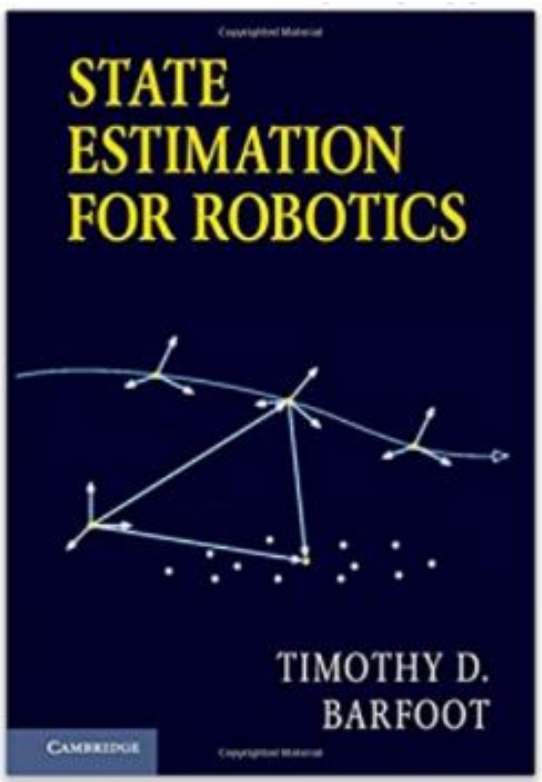
Autonomous driving

Augmented reality/VR

1. Course contents and preliminary knowledges

- Computer vision





Harley and Zisserman, Multiple view geometry in computer vision

Tim Barfoot, State estimation for robotics

1. Course contents and preliminary knowledges

- Course Contents
- Lecture 1. Basic knowledge, 3D motion
- Lecture 2. Lie group/Lie algebra, Camera models
- Lecture 3. State estimation, Nonlinear optimization
- Lecture 4. Visual Odometry
- Lecture 5. Backend Optimization

- Our course takes place at Monday a.m.
- Programming Assignments and Final Project are required

1. Course contents and preliminary knowledges

- Preliminary knowledge
- Math: Calculus, Linear algebra, Probability theory
- Programing: C++/Linux

- Our course takes place at Monday a.m.
- Programming Assignments and Final Project are required

Contents

- Course contents and preliminary knowledges
- Framework and mathematic form of a SLAM problem
- 3D geometry

2. Framework of SLAM

- SLAM problem
  - Fundamental problems in intelligent robots
  - Where am I?
    -Localization
  - What is around me?
    -Mapping

- Chicken and egg problem
  - Localization needs accurate map
  - Mapping needs accurate localization

2. Framework of SLAM

- How to do SLAM? -Sensors
- Sensor is the way to measure the outside environment
- Interoseptive sensors: accelerometer, gyroscope …
- Exteroceptive sensors: camera, laser rangefinder, GPS …

(a)

(b)

(c)

(d)

(e)

(f)

Some sensors must be placed in a cooperative environment, other can be directly equipped in the robot itself

2. Framework of SLAM

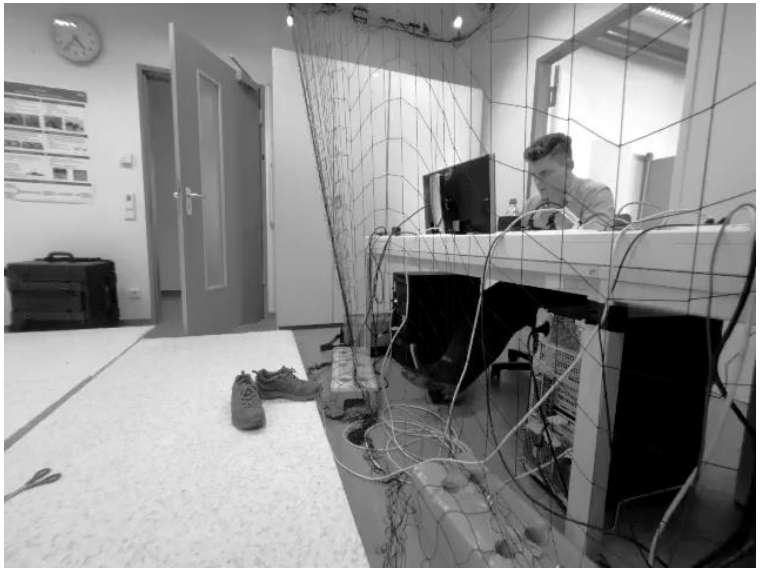
- Visual SLAM -SLAM (mainly) by cameras
- Cameras
  - Monocular
  - Stereo
  - RGB-D
  - Omnidirectional, Event camera, etc

Monocular camera

RGB-D (depth) camera

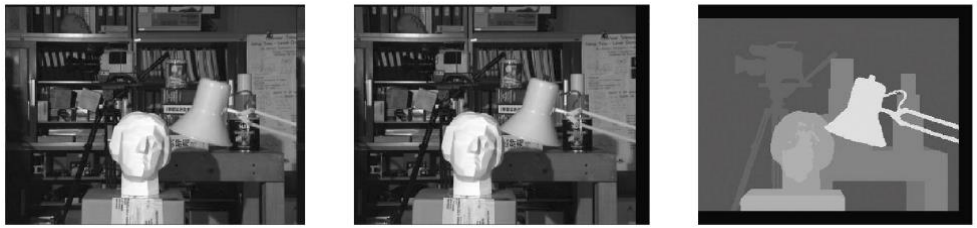
Stereo camera

- Cameras
  - Cheap, rich information
  - Record 2D projected image of the 3D world
  - The 3D-2D projection throws away one dimension: distance



2. Framework of SLAM

- Various kinds of cameras:
- Monocular: image only, need other methods to estimate the depth
- Stereo: disparity to depth
- RGB-D: physical depth measurements



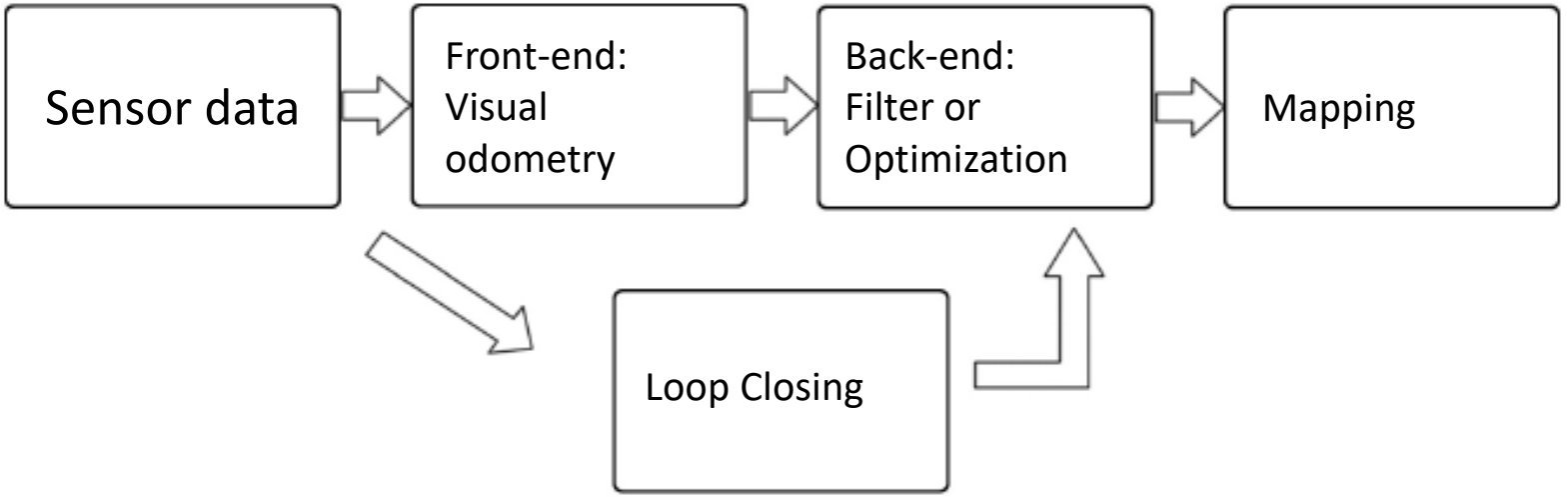
Stereo vision estimates the depth from disparity

Moving stereo: disparity can be estimated in the motion

Ambiguity in mono vision: small + close or large + far away?

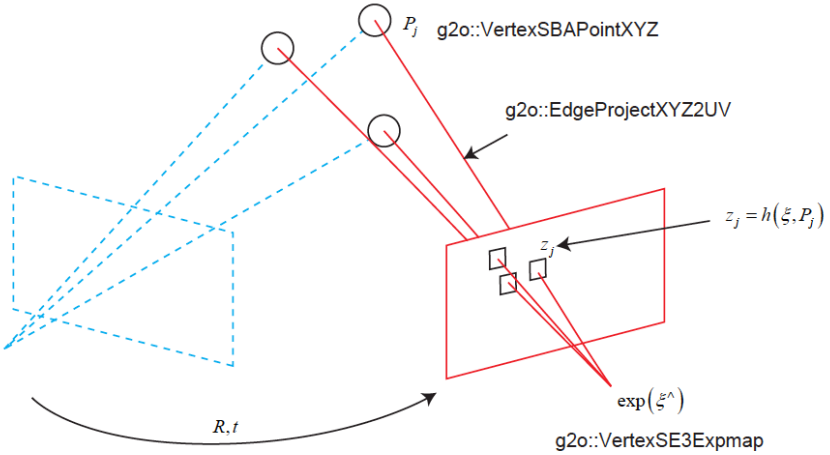
2. Framework of SLAM

- SLAM framework



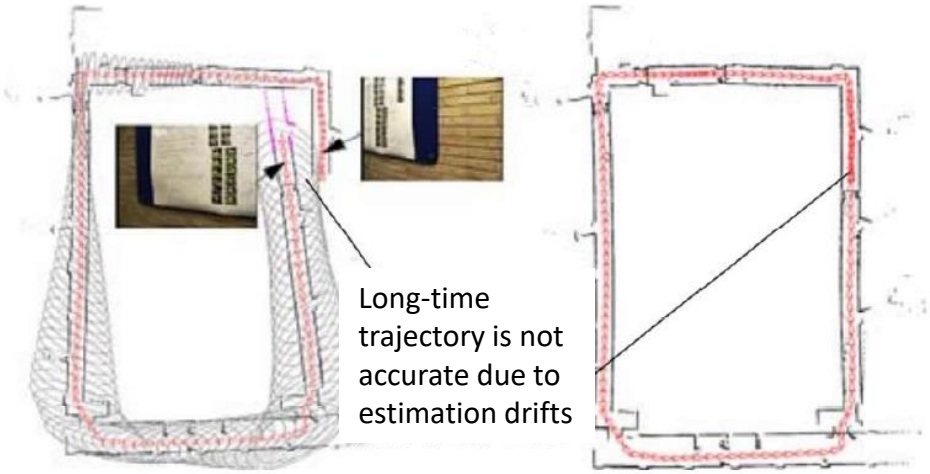
2. Framework of SLAM

- Visual odometry
  - Motion estimation betwee adjacent frames
  - Simplest: two-view geometry
- Method
  - Feature method
  - Direct method
- Backend
  - Long-term trajectory and map estimation
  - MAP: Maximum of a Posteri
  - Filters/Graph Optimization

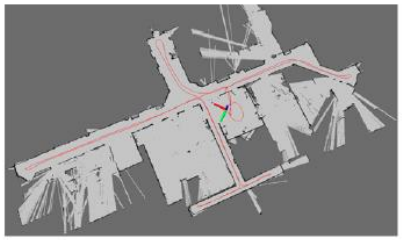


2. Framework of SLAM

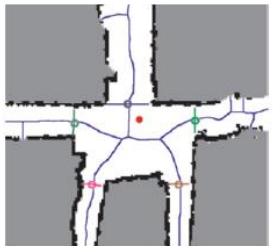
- Loop closing
  - Correct the drift in estimation
  - Loop detection and correction



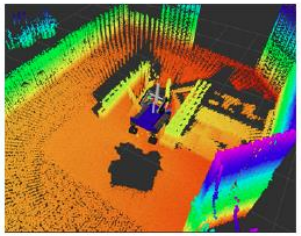
- Mapping
  - Generate globally consisten map for navigation/planning/comm nication/visualization etc
  - Grid/topological/hybrid maps
  - Pointcloud/Mesh/TSDF …



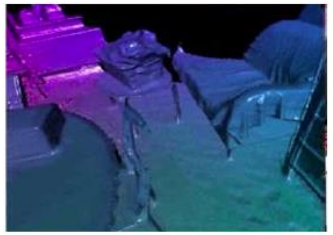
2D grid map



2D topological map



Point cloud maps



TSDF models

2. Framework of SLAM

- Mathematical representation of visual SLAM
- Assume a camera is moving in 3D space
  - But measurements are taken at discrete times:

$$\begin{cases} \boldsymbol{x}_k = f\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k, \boldsymbol{w}_k\right) \\ \boldsymbol{z}_{k,j} = h\left(\boldsymbol{y}_j, \boldsymbol{x}_k, \boldsymbol{v}_{k,j}\right) \end{cases}$$

Non-linear form

Motion model

Observation model

$$\begin{cases} x_k = A_k x_{k-1} + B_k u_k + w_k \\ z_{k,j} = C_j y_j + D_k x_k + v_{k,j} \end{cases}$$

linear form

2. Framework of SLAM

- Questions:

$$\begin{cases} \boldsymbol{x}_k = f\left(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k, \boldsymbol{w}_k\right) & \text{Motion model} \\ \boldsymbol{z}_{k,j} = h\left(\boldsymbol{y}_j, \boldsymbol{x}_k, \boldsymbol{v}_{k,j}\right) & \text{Observation model} \end{cases}.$$

- How to represent state variables?
  - 3D geometry, Lie group and Lie algebra
- Exact form of motion/observation model?
  - Camera intrinsic and extrinsics
- How to estimate the state given measurement data?
  - State estimation problem
  - Filters and optimization

Contents

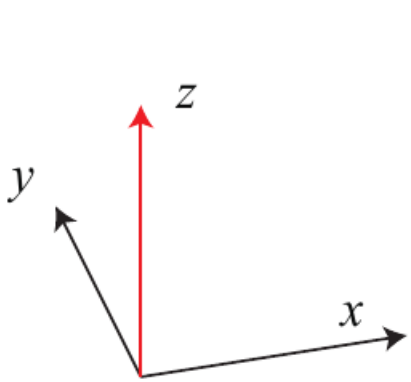
- Course contents and preliminary knowledges
- Framework and mathematic form of a SLAM problem
- 3D geometry

3. 3D geometry

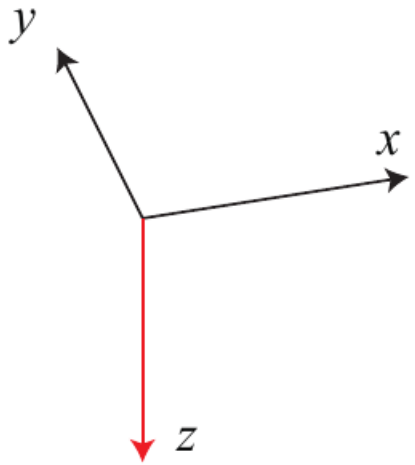
- Point and Coordinate system
- 2D: (x,y) and angle
- 3D?

3. 3D geometry

- 3D coordinate system
- Vectors and their coordinates



Right handed



Left handed

3. 3D geometry

- Vector operations
  - Addition/subtraction
  - Dot product

$$\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a}^T \boldsymbol{b} = \sum_{i=1}^{3} a_i b_i = |\boldsymbol{a}| \, |\boldsymbol{b}| \cos \langle \boldsymbol{a}, \boldsymbol{b} \rangle \, .$$

  - Cross product

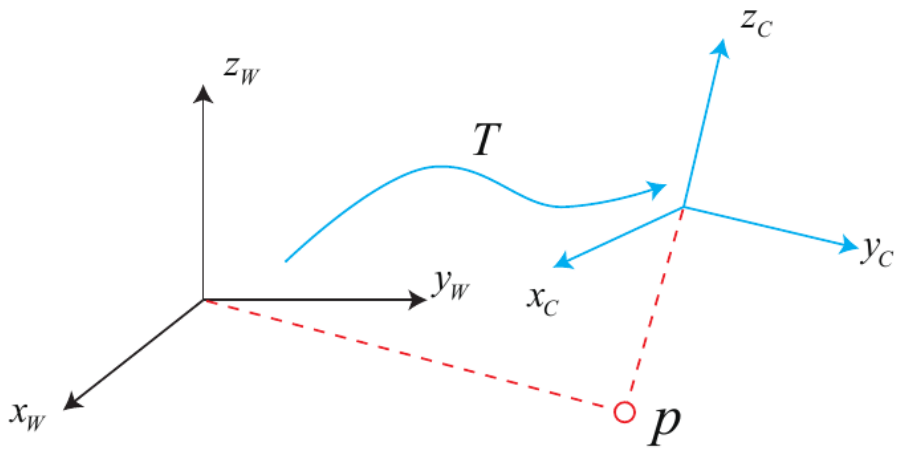
$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \boldsymbol{b} \triangleq \boldsymbol{a}^{\wedge} \boldsymbol{b}.$$

Skew-symmetric operator

3. 3D geometry

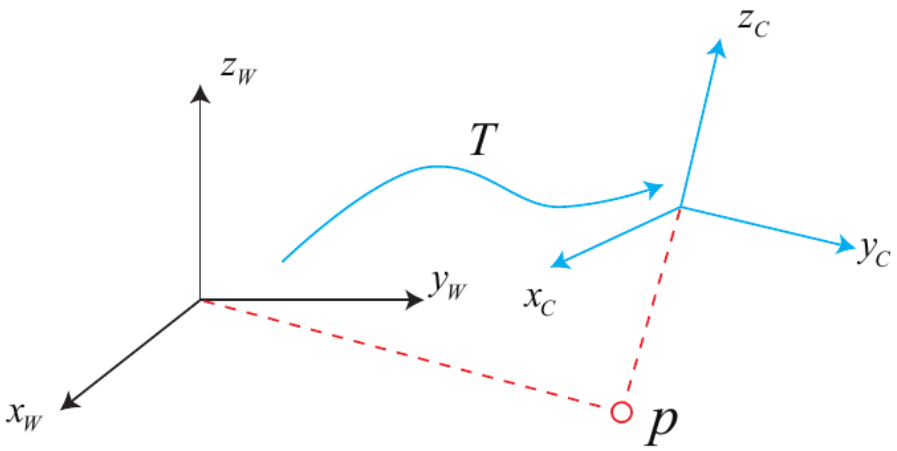
- Questions
  - Compute the coordinates in different systems?

- In SLAM:
  - Fixed world frame
  - Moving camera frame
  - Other sensor frames



3. 3D geometry

- 3D rigid body motion can be described with rotation and translation



- Translation is just a vector addition
- How to represent rotations?

3. 3D geometry

- Rotation
  - Consider coordinate system $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$ is rotated and become $(\boldsymbol{e}_1^{'}, \boldsymbol{e}_2^{'}, \boldsymbol{e}_3^{'})$
  - Vector $\boldsymbol{a}$ is fixed, then how are its coordinates changed?

$$[\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3] \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \left[\boldsymbol{e}_1^{'}, \boldsymbol{e}_2^{'}, \boldsymbol{e}_3^{'}\right] \begin{bmatrix} a_1^{'} \\ a_2^{'} \\ a_3^{'} \end{bmatrix}.$$

  - Left multiplied by $\left[e_1^T, e_2^T, e_3^T\right]^T$

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_1^T\boldsymbol{e}_1^{'} & \boldsymbol{e}_1^T\boldsymbol{e}_2^{'} & \boldsymbol{e}_1^T\boldsymbol{e}_3^{'} \\ \boldsymbol{e}_2^T\boldsymbol{e}_1^{'} & \boldsymbol{e}_2^T\boldsymbol{e}_2^{'} & \boldsymbol{e}_2^T\boldsymbol{e}_3^{'} \\ \boldsymbol{e}_3^T\boldsymbol{e}_1^{'} & \boldsymbol{e}_3^T\boldsymbol{e}_2^{'} & \boldsymbol{e}_3^T\boldsymbol{e}_3^{'} \end{bmatrix} \begin{bmatrix} a_1^{'} \\ a_2^{'} \\ a_3^{'} \end{bmatrix} \triangleq \boldsymbol{R}\boldsymbol{a}^{'}.$$

Rotation matrix

3. 3D geometry

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_1^T \boldsymbol{e}_1' & \boldsymbol{e}_1^T \boldsymbol{e}_2' & \boldsymbol{e}_1^T \boldsymbol{e}_3' \\ \boldsymbol{e}_2^T \boldsymbol{e}_1' & \boldsymbol{e}_2^T \boldsymbol{e}_2' & \boldsymbol{e}_2^T \boldsymbol{e}_3' \\ \boldsymbol{e}_3^T \boldsymbol{e}_1' & \boldsymbol{e}_3^T \boldsymbol{e}_2' & \boldsymbol{e}_3^T \boldsymbol{e}_3' \end{bmatrix} \begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix} \triangleq \boldsymbol{R}\boldsymbol{a}'.$$

- R is rotation matrix, which satisfies:
  - R is orthogonal
  - Det(R) = +1 (if Det(R)=-1 then it's improper rotation)

- Special orthogonal group:

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

- Rotation from frame 2 to 1 can be written as:

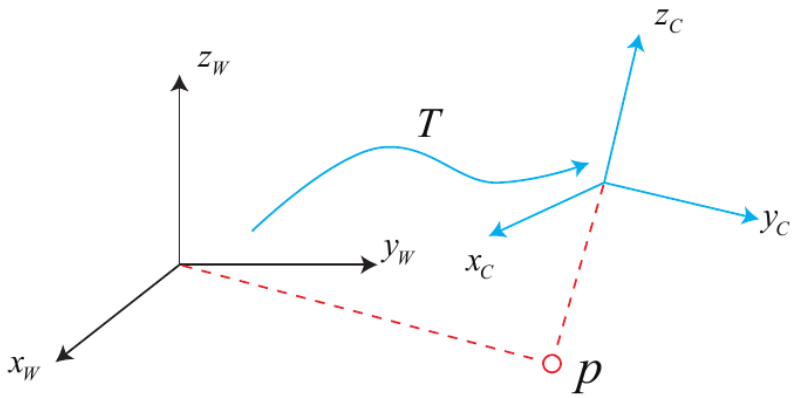
$a_1 = R_{12} a_2$ and vise vesa: $a_2 = R_{21} a_1$

$$R_{21} = R_{12}^{-1} = R_{12}^T$$

3. 3D geometry

- Rotation plus translation:

$$\boldsymbol{a}' = \boldsymbol{R}\boldsymbol{a} + \boldsymbol{t}.$$



- Compounding rotation and translation:
  - $\boldsymbol{b} = \boldsymbol{R}_1\boldsymbol{a} + \boldsymbol{t}_1, \quad \boldsymbol{c} = \boldsymbol{R}_2\boldsymbol{b} + \boldsymbol{t}_2.$ ⟹ $\boldsymbol{c} = \boldsymbol{R}_2\left(\boldsymbol{R}_1\boldsymbol{a} + \boldsymbol{t}_1\right) + \boldsymbol{t}_2.$
- Homogeneous form:

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

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

Inverse: $$\boldsymbol{T}^{-1} = \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T t \\ \boldsymbol{0}^T & 1 \end{bmatrix}.$$

3. 3D geometry

- Homogenous coordinates:

$$\tilde{a} = \begin{bmatrix} a \\ 1 \end{bmatrix} \qquad \tilde{a} = \begin{bmatrix} a \\ 1 \end{bmatrix} = k \begin{bmatrix} a \\ 1 \end{bmatrix}$$

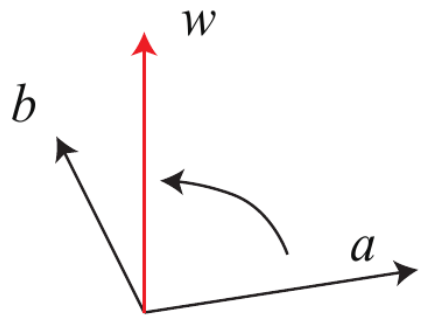
Still keeps equal when multiplying any non-zero factors

- Transform matrix forms Special Euclidean Group

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

3. 3D geometry

- Alternative rotation representations
  - Rotation vectors
  - Euler angles
  - Quaternions



Rotation vectors
Only three parameters

- Rotation vectors
  - Angle + axis: θn
  - Rotation angle θ
  - Rotation axis n
- Rotation vector to rotation matrix: Rodrigues' formula

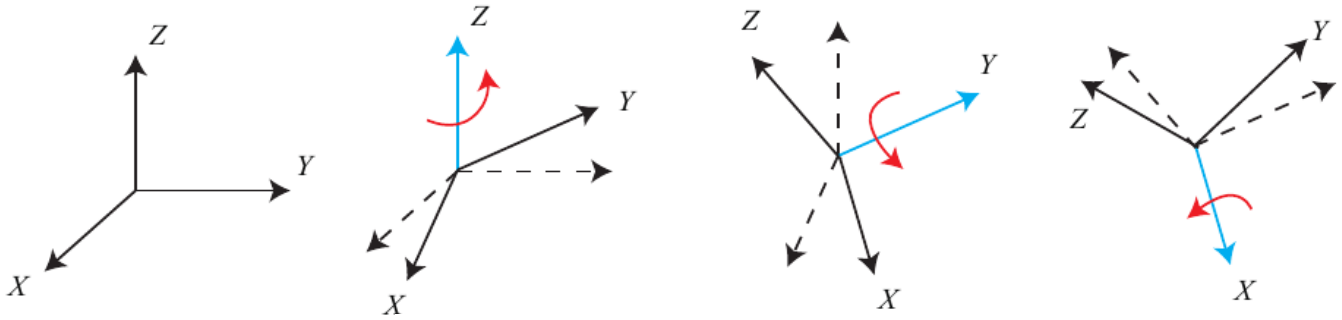
$$\boldsymbol{R} = \cos\theta \boldsymbol{I} + (1 - \cos\theta)\, \boldsymbol{n}\boldsymbol{n}^T + \sin\theta \boldsymbol{n}^\wedge.$$

- Inverse:

$$\theta = \arccos\left(\frac{\mathrm{tr}(\boldsymbol{R}) - 1}{2}\right). \qquad \boldsymbol{R}\boldsymbol{n} = \boldsymbol{n}.$$

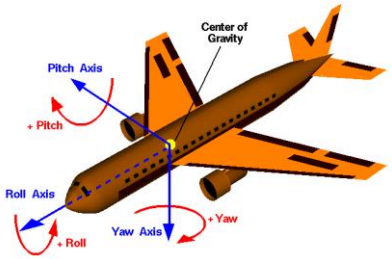
3. 3D geometry

- Euler angles
  - Any rotation can be decomposed into three principal rotations



Original First Second Third

  - However the order of axis can be defined very differently:
  - Roll-pitch-yaw (in navigation) Spin-nutation-precession in mechanics

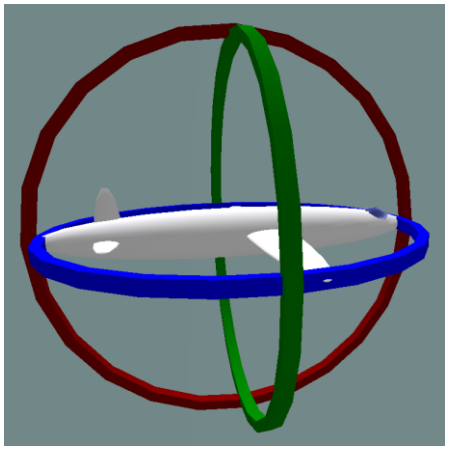


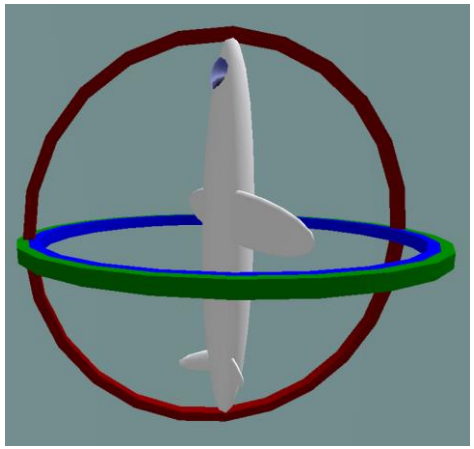
XYZ order 3-1-3 order

3. 3D geometry

- Gimbal lock
  - Singularity always exist if we want to use 3 parameters to describe rotation
  - Degree-of-Freedom is reduced in singular case
  - In yaw-pitch-roll order, when pitch=90 degrees

normal





singular

3. 3D geometry

- Quaternions
  - In 2D case, we can use (unit) complex numbers to denote rotations

$$z = x + iy = re^{iq}$$

Multiply i to rotate 90 degrees

  - How about 3D case?
- (Unit) Quaternions
  - Extended from complex numbers
  - Three imaginary and one real part:
  - The imaginary parts satisfy:

$$\boldsymbol{q} = q_0 + q_1 i + q_2 j + q_3 k,$$

$$\begin{cases} i^2 = j^2 = k^2 = -1 \\ ij = k, ji = -k \\ jk = i, kj = -i \\ ki = j, ik = -j \end{cases}.$$

i,j,k look like complex numbers when multiplying with themselves
And look like cross product when multiply with others

3. 3D geometry

- Quaternions

$$\boldsymbol{q} = q_0 + q_1 i + q_2 j + q_3 k, \qquad \boldsymbol{q} = [s, \boldsymbol{v}], \quad s = q_0 \in \mathbb{R}, \boldsymbol{v} = [q_1, q_2, q_3]^T \in \mathbb{R}^3,$$

- Operations

$$\boldsymbol{q}_a \pm \boldsymbol{q}_b = [s_a \pm s_b, \boldsymbol{v}_a \pm \boldsymbol{v}_b].$$

$$\begin{aligned} \boldsymbol{q}_a \boldsymbol{q}_b = \; & s_a s_b - x_a x_b - y_a y_b - z_a z_b \\ & + (s_a x_b + x_a s_b + y_a z_b - z_a y_b)\, i \\ & + (s_a y_b - x_a z_b + y_a s_b + z_a x_b)\, j \\ & + (s_a z_b + x_a y_b - y_b x_a + z_a s_b)\, k. \end{aligned}$$

$$\boldsymbol{q}_a \boldsymbol{q}_b = \left[s_a s_b - \boldsymbol{v}_a^T \boldsymbol{v}_b, s_a \boldsymbol{v}_b + s_b \boldsymbol{v}_a + \boldsymbol{v}_a \times \boldsymbol{v}_b \right].$$

$$\boldsymbol{q}_a^* = s_a - x_a i - y_a j - z_a k = [s_a, -\boldsymbol{v}_a].$$

$$\|\boldsymbol{q}_a\| = \sqrt{s_a^2 + x_a^2 + y_a^2 + z_a^2}.$$

$$\boldsymbol{q}^{-1} = \boldsymbol{q}^* / \|\boldsymbol{q}\|^2.$$

$$k\boldsymbol{q} = [ks, k\boldsymbol{v}].$$

$$\boldsymbol{q}_a \cdot \boldsymbol{q}_b = s_a s_b + x_a x_b i + y_a y_b j + z_a z_b k.$$

3. 3D geometry

- From quaternions to angle-axis:

$$\boldsymbol{q} = \left[\cos\frac{\theta}{2}, n_x \sin\frac{\theta}{2}, n_y \sin\frac{\theta}{2}, n_z \sin\frac{\theta}{2}\right]^T.$$

- Inverse:

$$\begin{cases} \theta = 2\arccos q_0 \\ [n_x, n_y, n_z]^T = [q_1, q_2, q_3]^T / \sin\frac{\theta}{2} \end{cases}.$$

- Rotate a vector by quaternions:
  - Vector p is rotated by q and become p', how to write their relationships?
  - Write p as quaternion (pure imaginary): $\boldsymbol{p} = [0, x, y, z] = [0, \boldsymbol{v}].$
  - Then: $\boldsymbol{p}' = \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^{-1}.$ Also pure imaginary

Notes on programing assignments

- Use cmake to manage your C++ project in Linux
- Use Eigen to handle matrix and geometry computations