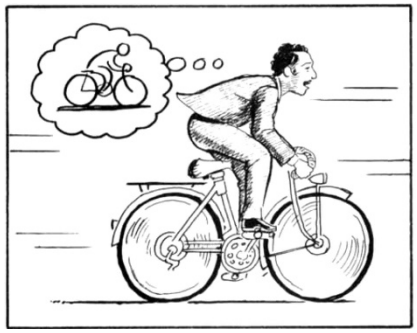
UC San Diego

Introduction to Rigid-Body Motion

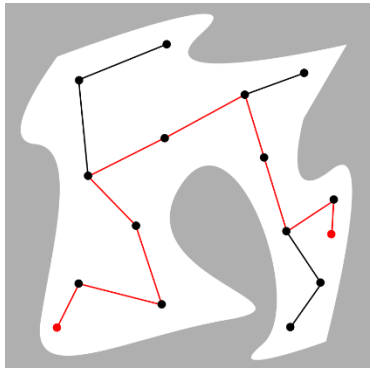
Yuzhe Qin

Physics-based Learning System



Modeling

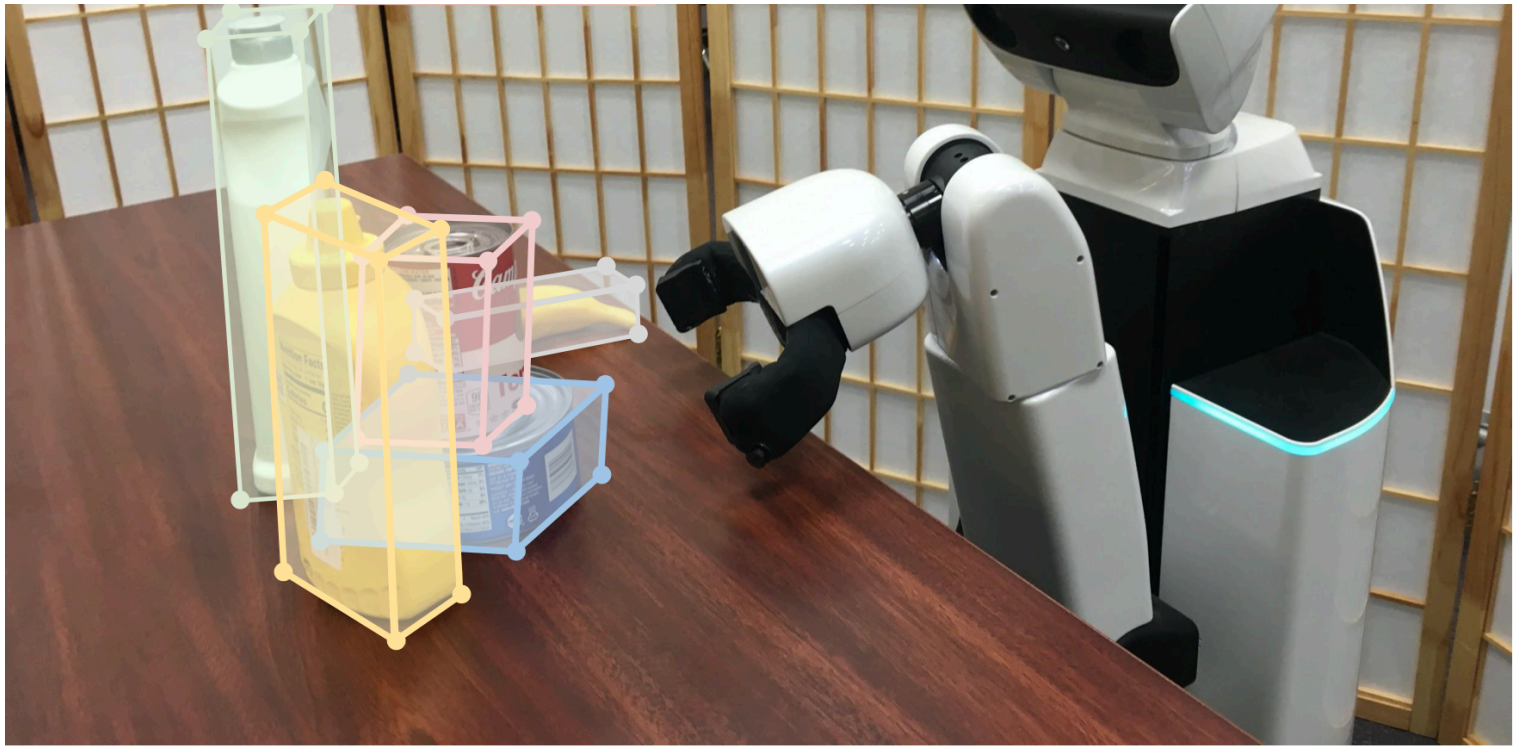
Perception



Planning

Toward Better Modeling

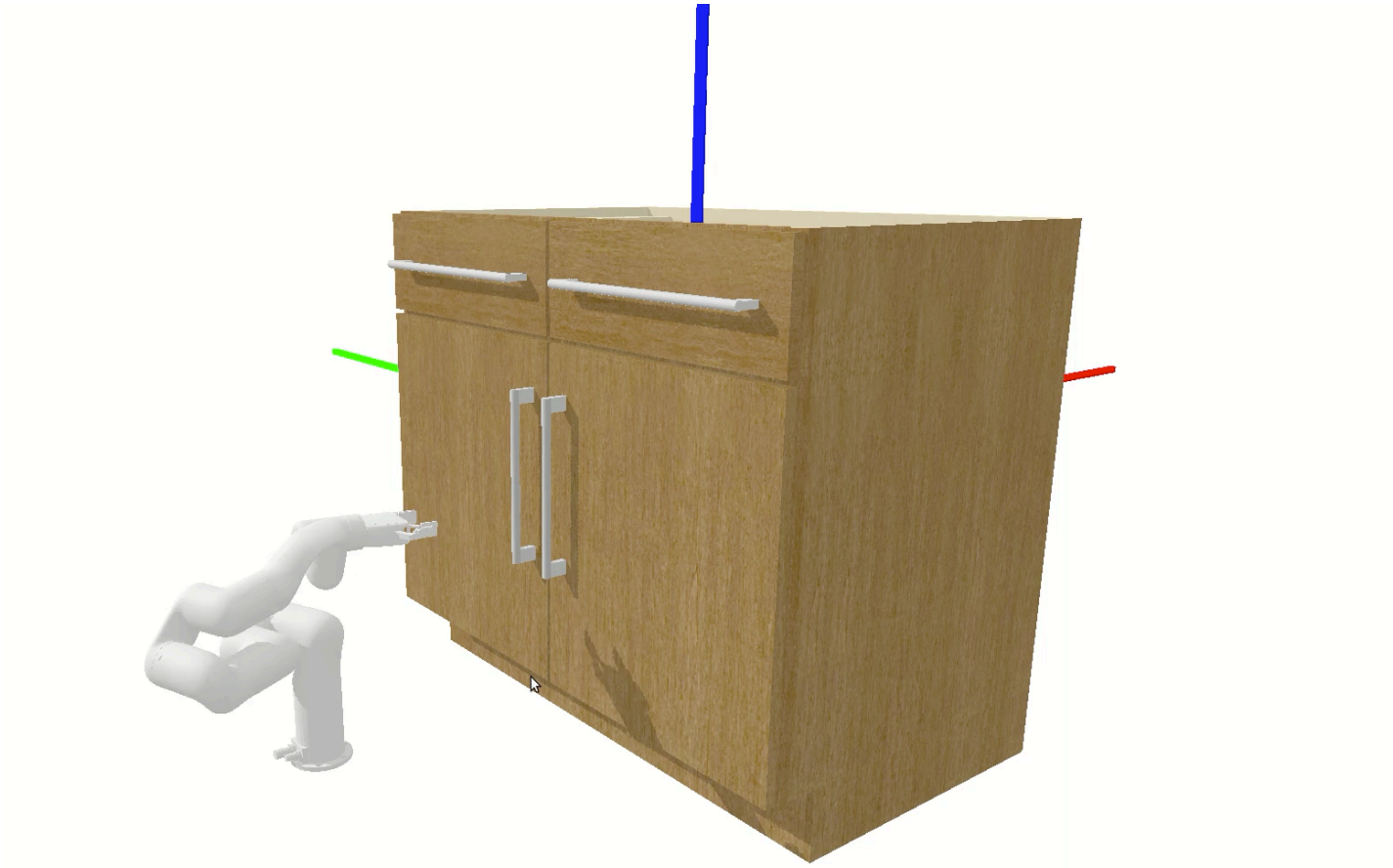
How to represent the spatial relationship of robot and objects?



Wang et al., "**DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion**", *CVPR 2019*

Toward Better Modeling

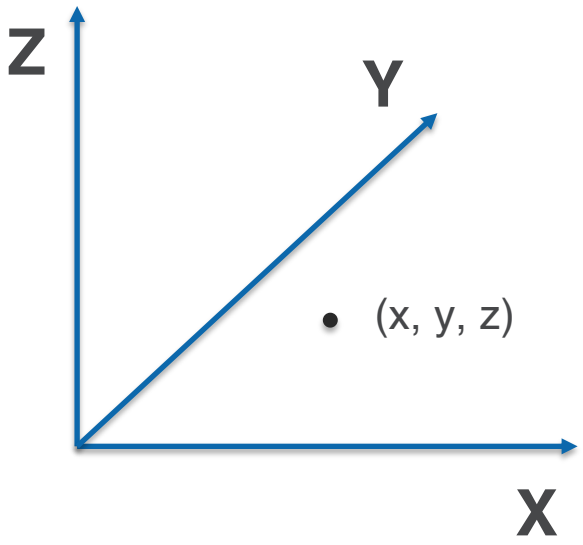
How to command the robot to execute the desired motion?



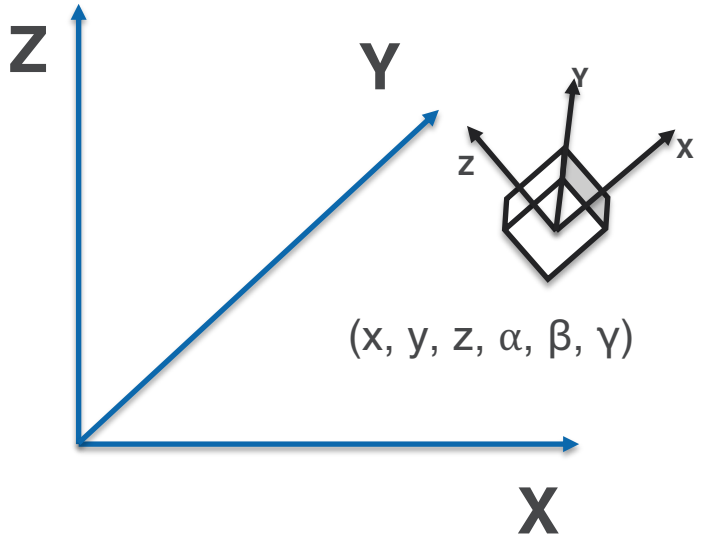
Topics

- **Rotation and SO(3)**
- Rotation Parameterizations
- Learning to Predict Rotation by NN
- Homogenous Transformation and SE(3)
- Case-Study: Hand-eye Calibration

Rigid-Body in 3D Space



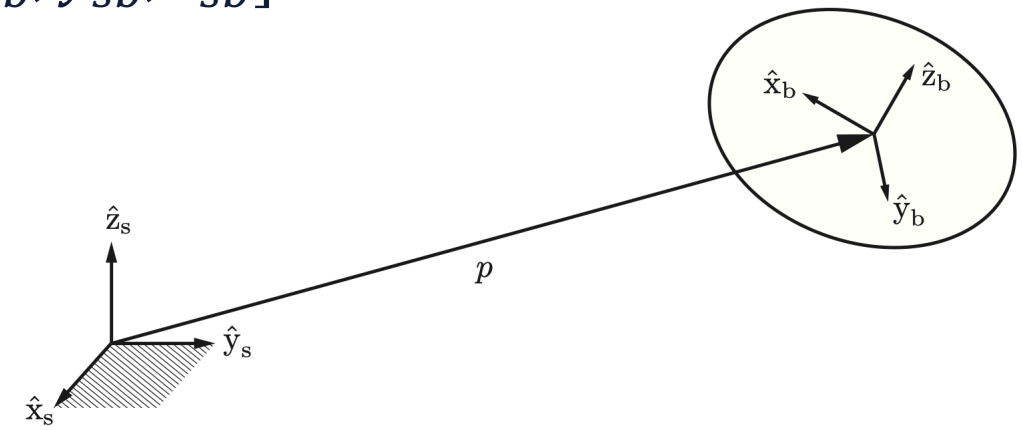
Point:
Position
3-Dimensional



Rigid-Body:
Position + Orientation
6-Dimensional

Represent Orientation

- The relationship of two frames define orientation
- Orientation is defined via two frames:
  - Space Frame: $\{s\} = \{\hat{x}_s, \hat{y}_s, \hat{z}_s\}$
  - Body Frame: $\{b\} = \{\hat{x}_b, \hat{y}_b, \hat{z}_b\}$
- $R_{sb} = [\vec{x}_{sb}, \vec{y}_{sb}, \vec{z}_{sb}]$ is called a rotation matrix



Property of Rotation Matrix

- Special Orthogonal Group $SO(n)$ for R^n space:
  - $SO(n) = \{ R \in R^{n \times n} : \det(R) = 1, RR^T = I \}$
- Degree of freedom of $SO(n)$ is $\frac{n(n-1)}{2}$
- Standard property of Group:
  - Associativity, closure, identity element, inverse element
- Change reference frame:
  - $p_s = R_{sb} p_b$, where $p \in R^3$

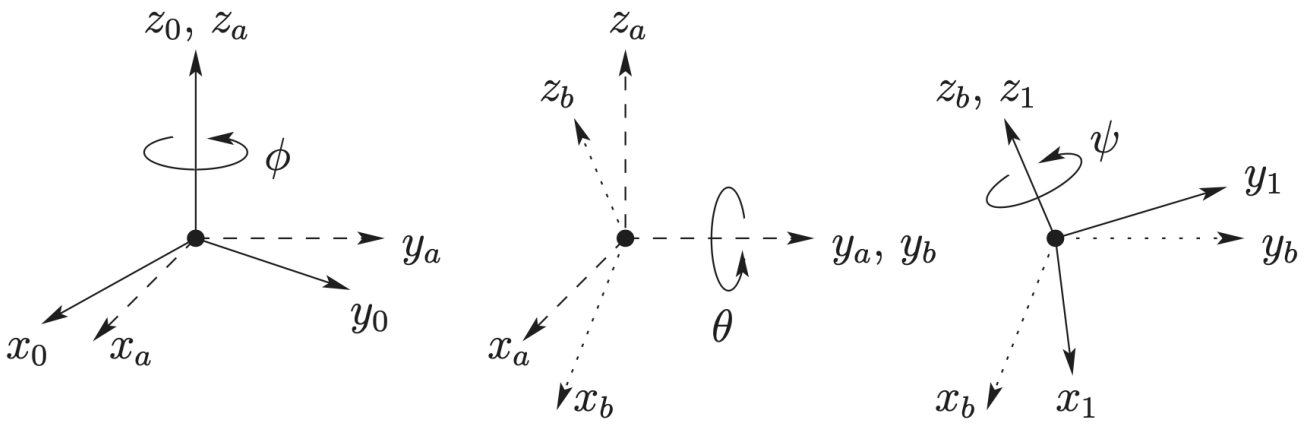
Topics

- Rotation and SO(3)
- **Rotation Parameterizations**
- Learning to Predict Rotation by NN
- Homogenous Transformation and SE(3)
- Case-Study: Hand-eye Calibration

Representation I: Euler Angle

- Euler Angle Definition:
  1. Body frame {b} coincident with space frame {s} on the beginning
  2. Rotate {b} about one axis
  3. Rotate {b} about another axis, different from first one
  4. Finally rotate {b} about another axis in step 2

- E.g. ZYZ Euler Angle



Euler Angle to Rotation Matrix

- Rotation about principal axis is represented as:

$$R_x(\theta) \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}$$

$$R_y(\theta) \triangleq \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

$$R_z(\theta) \triangleq \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- E.g. axes='rxyz':
  - $R_{sb} = R_x(\theta_1)R_y(\theta_2)R_z(\theta_3)$

Euler Angle Singularity

- Singularity: for a $R \in SO(3)$, there is more than one Euler angle to represent

- Euler Angle is not unique for some rotation matrix

$$\mathbf{R} = \mathbf{R}_z(90°)\,\mathbf{R}_y(90°)\,\mathbf{R}_x(90°) = \begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 0.000 & 1.000 & 0.000 \\ -1.000 & 0.000 & 0.000 \end{bmatrix}$$

$$\mathbf{R} = \mathbf{R}_z(45°)\,\mathbf{R}_y(90°)\,\mathbf{R}_x(45°) = \begin{bmatrix} 0.000 & 0.000 & 1.000 \\ 0.000 & 1.000 & 0.000 \\ -1.000 & 0.000 & 0.000 \end{bmatrix}$$

https://www.mecademic.com/resources/Euler-angles/Euler-angles

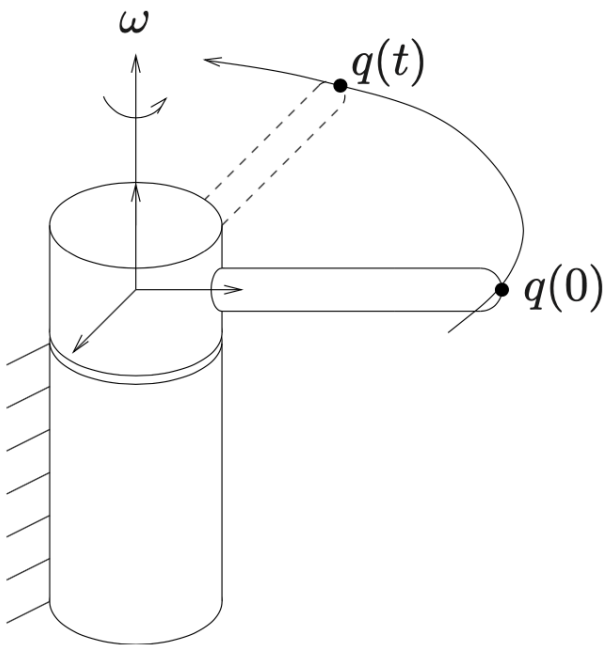
Representation II: Axis-Angle

- Euler Theorem:
  - Any rotation in R^3 is equivalent to rotation about a fixed axis $\hat{\omega} \in R^3$ through an positive angle θ
  - $\hat{\omega}$: unit vector of rotation axis
  - θ: angle of rotation
  - $R \in SO(3) \triangleq Rot(\hat{\omega}, \theta)$

Axis-Angle Conversion

- Consider a point q in body frame. At time $t = 0$, the position is q_0

- Rotate q with unit angular velocity around axis $\hat{\omega}$:
  - $v = \hat{\omega} \times r$
  - $\dot{q}(t) = \hat{\omega} \times q(t) = [\hat{\omega}]q(t)$



Skew-Symmetric Matrix

- A is skew-symmetric $A = -A^T$
- Skew-symmetric matrix operator:

$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, [\omega] \triangleq \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

- Cross product can be a linear transformation:
  - $a \times b = [a]b$
- Lie Algebra of 3d rotation: $so(3) \triangleq \{S \in R^{3\times 3} : S^T = -S\}$

Axis-Angle Conversion

- Rotate p with unit angular velocity around axis $\hat{\omega}$:
  - $\hat{v} = \hat{\omega} \times r$
  - $\dot{p}(t) = \hat{\omega} \times p(t) = [\hat{\omega}]p(t)$
  - Solution of this ODE: $\mathrm{p(t)} = e^{[\hat{\omega}]t}p_0$
  - For unit angular velocity, $\theta(t) = t$, θ is the total rotation angle
  - So $p(\theta) = e^{[\hat{\omega}]\theta}p_0$
  - It means that $Rot(\hat{\omega}, \theta) = e^{[\hat{\omega}]\theta}$ (exponential map)

- $\vec{\omega} = \hat{\omega}\theta$ also called **rotation vector** or **exponential coordinate**

Axis-Angle to Rotation

- Definition of Matrix Exponential:
  - $e^{[\hat{\omega}]\theta} = \mathrm{I} + \theta\hat{\omega} + \frac{\theta^2}{2!}[\hat{\omega}]^2 + \frac{\theta^3}{3!}[\hat{\omega}]^3 + \cdots + \frac{\theta^n}{n!}[\hat{\omega}]^n, n \to +\infty$
- Sum of infinite series? **Rodrigues Formula**
  - Can prove $[\hat{\omega}]^3 = -[\hat{\omega}]$
  - Then use Taylor expansion of **sin** and **cos**

$$e^{[\hat{\omega}]\theta} = \mathrm{I} + [\hat{\omega}]\sin(\theta) + [\hat{\omega}]^2(1 - \cos(\theta))$$

Exponential: $[\hat{\omega}]\theta \in so(3) \quad \rightarrow \quad R \in SO(3)$

Rotation to Axis-Angle

- Is Exponential Coordinate unique?
  - $\theta \in [0, \pi)$, otherwise simple reverse the axis and angle
  - $R \neq I$, otherwise $\theta = 0$ and $\hat{\omega}$ can be any direction
  - $tr(R) \neq -1$, otherwise $\theta = \pi$ and $\hat{\omega}$ have three possible solutions
- Then, a unique inverse mapping exist:

$$\theta = \cos^{-1}\left(\frac{1}{2}(tr(R) - 1)\right)$$

$$[\hat{\omega}] = \frac{1}{2\sin(\theta)}(R - R^T)$$

Logarithm: $R \in SO(3) \quad \rightarrow \quad [\hat{\omega}]\theta \in so(3)$

Representation III: Quaternion

- Recall the complex number $a + b\boldsymbol{i}$
- Quaternion is more generalized complex number:

$$\boldsymbol{q} = w + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$$

  - w is the real part and $\vec{q} = (x, y, z)$ is the imaginary part
  - Imaginary: $i^2 = j^2 = k^2 = ijk = -1$
  - anticommutative : $ij = k = -ji, jk = i = -kj, ki = j = -ik$

Quaternion Property

- In vector-form, product of two quaternion:

$$\boldsymbol{q_1} = (w_1, \vec{q}_1), \boldsymbol{q_2} = (w_2, \vec{q}_2)$$

$$\boldsymbol{q_1 q_2} = (w_1 w_2 - \vec{q}_1^{\mathrm{T}} \vec{q}_2, \omega_1 \vec{q}_2 + \omega_2 \vec{q}_1 + \vec{q}_1 \times \vec{q}_2)$$

- Norm:

$$\|\boldsymbol{q}\| = w^2 + \vec{q}^T \vec{q} = \boldsymbol{q q^*} = \boldsymbol{q^* q}$$

- Conjugate:

$$\boldsymbol{q^*} = (w, -\vec{q}\,)$$

- Unit quaternion:

$$\boldsymbol{q_0} = (1,0,0,0)$$

- Inverse:

$$\boldsymbol{q^{-1}} \triangleq \frac{\boldsymbol{q^*}}{\|\boldsymbol{q}\|^2}$$

Quaternion Conversions

- A **unit** quaternion $\|\boldsymbol{q}\| = 1$ can represent rotation

- Exponential coordinates $\hat{\omega}\theta$ to unit quaternion:

$$\boldsymbol{q} = \left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{\omega}\right]$$

- Unit quaternion to axis-angle representation:

$$\theta = 2\cos^{-1}(\omega), \hat{\omega} = \begin{cases} \frac{1}{\sin\left(\frac{\theta}{2}\right)}\vec{q}, & \theta \neq 0 \\ 0, & \theta = 0 \end{cases}$$

- Obtain rotation matrix from unit quaternion:

$$R(\boldsymbol{q}) = E(\boldsymbol{q})G(\boldsymbol{q})^T, \quad E(\boldsymbol{q}) = \left[-\vec{\boldsymbol{q}}, wI + [\vec{\boldsymbol{q}}]\right], \quad G(\boldsymbol{q}) = \left[-\vec{\boldsymbol{q}}, wI - [\vec{\boldsymbol{q}}]\right]$$

Quaternion as Rotation

- Represent orientation:
- Change reference frame for $p_b \in R^3$ in body frame:
  1. Construct a purely imaginary quaternion $\boldsymbol{p} = (0, \vec{p})$
  2. Quaternion form body to world: $\boldsymbol{q_{sb}} = (\omega, \vec{q})$
  3. $\overrightarrow{P_s} = Im(\boldsymbol{q_{sb}} \boldsymbol{p} \boldsymbol{q_{sb}^*}) = (\omega^2 - \|\vec{q}\|^2)\vec{p} + 2(\vec{q}^T\vec{p})\vec{q} + 2\omega(\vec{q} \times \vec{v})$
- Composing rotation:
  - Similar as rotation matrix, whichever quaternion is on the *right* is the rotation that is performed first:

$$\boldsymbol{q_{ab} q_{bc} p_c q_{ab}^* q_{bc}^* = p_a}$$

More About Quaternion

- **No singularity** with $SO(3)$
  - No singularity if we require the real part to be positive
  - Embeds a 3-D space into a 4-D space with unit norm constraint

- **Singularity** with axis-angle at $\theta = 0$
  - Many-to-one from axis-angle to quaternion

- Quaternion is computationally cheap:
  - Internal representation of Physical Engine and Robot

- Pay attention to convention (w, x, y, z) or (x, y, z, w)?
  - (w, x, y, z): SAPIEN, transforms3d, Eigen, blender, MuJoCo, V-Rep
  - (x, y, z, w): ROS, Physx, PyBullet

Rotation Representation Sum.

	Form	Change Frame?	Inverse?	Composing?	No Singularity?
Rotation Matrix	$SO(3)$				N/A
Euler Angle	R^3				
Rotation Vector	R^3				
Skew-symmetrical Matrix	$so(3)$				
Quaternion	R^4				

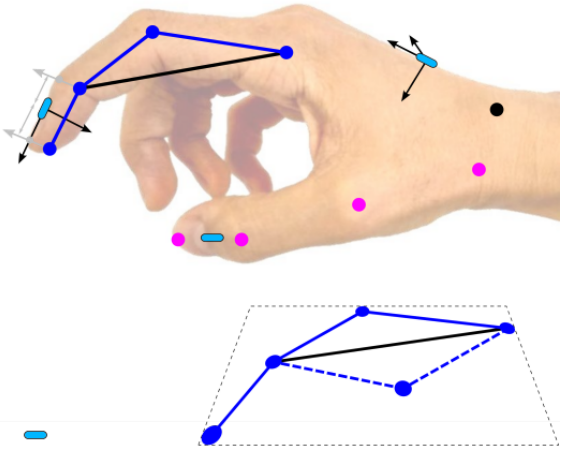
Topics

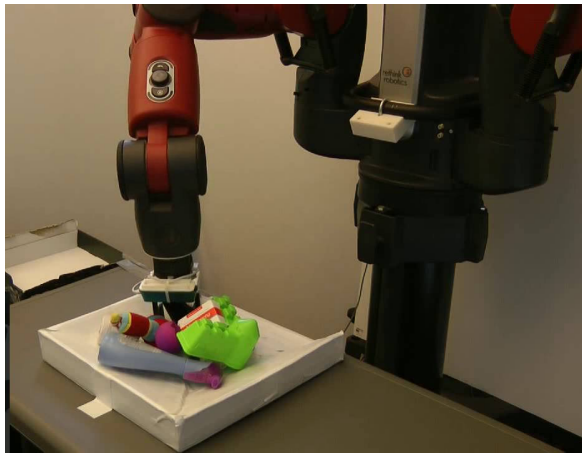
- Rotation and SO(3)
- Rotation Parameterizations
- **Learning to Predict Rotation by NN**
- Homogenous Transformation and SE(3)
- Case-Study: Hand-eye Calibration

Predict Rotation via Neural Network

Learn to predict rotation via supervised learning?

- 3D Shape Pose Estimation[1]
- Human Pose Estimation[2]
- Grasping Pose Prediction[3]





Tremblay et al., "**Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects**", *CoRL 2018*
Yuan et al., "**BigHand2.2M Benchmark: Hand Pose Dataset and State of the Art Analysis**", *CVPR 2017*
Pas et al., "**DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion**", *IJRR 2017*

Rotation Regression: Loss and Representation

- We need to choose the **representation** and **loss**
- This is not a trivial task, simply combining loss and representation will fail
- E.g., Euler Angle + L2 Loss is not a good choice
  - Prediction: $(\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2})$, Ground-Truth: $\left(\frac{\pi}{4}, \frac{\pi}{2}, \frac{\pi}{4}\right)$, but they are the same
- Representation should be chosen based on loss function

Distance Metric on SO(3)

- How far it is between two rotations?
  - L2 distance of rotation representation?

- Remember metric in mathematics:
  - $d(x, y) > 0$
  - $d(x, y) = 0 \Leftrightarrow x = y$
  - $d(x, y) = d(y, x)$
  - $d(x, y) < d(x, z) + d(z, y)$

- Can we find a metric to define the distance in $SO(3)$

Loss: Distance Metric on SO(3)

- Key idea: use the relative transformation of two rotation
- How much angle from R_{pred} to R_{gt} ?
  - First compute the relative rotation, $R_{loss} = R_{gt}^{T} R_{pred}$
  - Then compute the axis-angle representation for rotation matrix, only the angle is used in loss term
  - $\theta_{loss} = \log(R_{gt}^{T} R_{pred}) = \cos^{-1}(\frac{tr(R_{loss})-1}{2})$,
- It's easy to prove that this quantity is a metric

An Empirically-good Representation/Loss

- Representations for the 3D rotations are **discontinuous** in **4** or fewer dimensions
- Should use redundant representation for rotation
- E.g. 5D/6D continuous representation for 3D rotations:
  - The first two column (for 6D case) of rotation matrix

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

Zhou et al., "**On the Continuity of Rotation Representation in Neural Networks**", *ICCV 2017*

Training Rotation Regression Network

- Regress 6D batched vector on the last layer
  - $r_{11}, r_{21}, r_{31}, r_{12}, r_{22}, r_{32}$

- Denote $a_1 = [r_{11}, r_{21}, r_{31}]^T, a_2 = [r_{12}, r_{22}, r_{32}]^T$, $B = [b_1, b_2, b_3] \in SO(3)$

$$\begin{pmatrix} \vdots & \vdots \\ a_1 & a_2 \\ \vdots & \vdots \end{pmatrix} \xrightarrow{\text{operation}} \begin{pmatrix} \vdots & \vdots & \vdots \\ b_1 & b_2 & b_3 \\ \vdots & \vdots & \vdots \end{pmatrix}$$

1. Normalize the first column, denote as $N(a_1) = b_1$
2. Gram-Schmidt Orthogonalization: $N(a_2 - (b_1 \cdot a_2)b_1) = b_2$
3. Cross for third column: $b_1 \times b_2 = b_3$
4. Calculate loss based on the L2 loss of 5/6D representation
5. Update network parameters

Zhou et al., "**On the Continuity of Rotation Representation in Neural Networks**", *ICCV 2017*

Classification in Rotation Prediction

- Classification can always be a simpler choice
- Fine-grained space with discretized angles[1]
  - E.g., divide rotation into 360 section
  - Using Bin classification for rotation prediction
- Classification method not suffer from discontinuity

Su et al., "**Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views**", *ICCV 2015*

Topic

- Rotation and SO(3)
- Rotation Parameterizations
- Learning to Predict Rotation by NN
- **Homogenous Transformation and SE(3)**
- Case-Study: Hand-eye Calibration

Rigid-Body Configuration and SE(3)

- General rigid-body configuration includes both position $p \in R^3$ and rotation $R \in SO(3)$.

- Special Euclidean Group for 3D Space:
  - $SE(3) \triangleq \{T = \begin{bmatrix} R & p \\ 0_{1\times 3} & 1 \end{bmatrix}, R \in SO(3), p \in R^3\}$

- $T \in SE(3)$ called homogenous transformation matrix. Similar to rotation, T can represent pose of a rigid-body and change reference frame of point $p \in R^3$

- Composing: $T_{ab}T_{bc} = T_{ac}$

Homogenous Coordinates

- Homogeneous Coordinate for 3D Space:
  - $\tilde{p} \triangleq \begin{bmatrix} p \\ 1 \end{bmatrix} \in R^4$

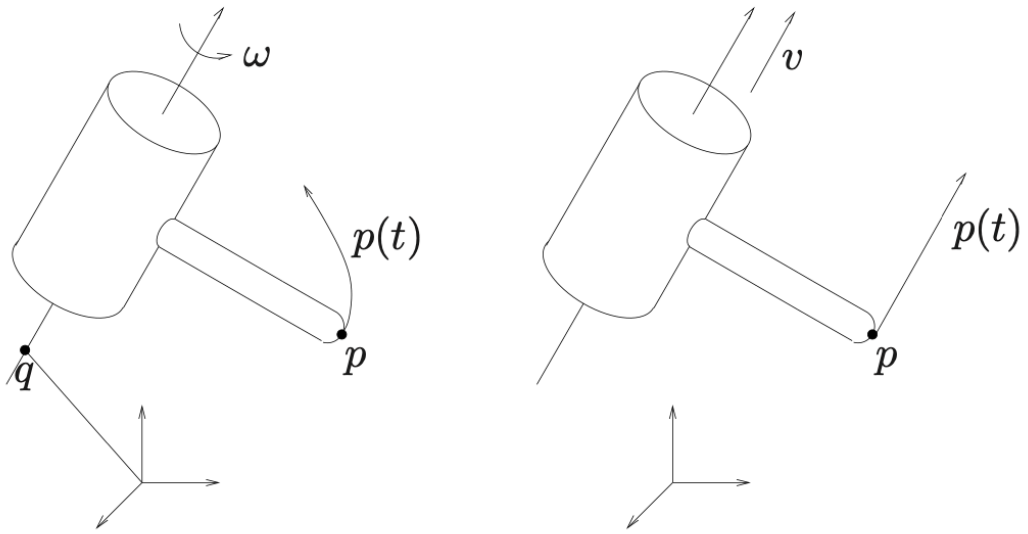
- Homogeneous coordinates make it easier when changing reference frame:

  - $p_s = R_{sb}p_b + p_{sb}$, where p_{sb} is the position of frame $\{b\}$ origin in frame $\{s\}$. The formula below is equivalent:

$$\tilde{p}_s = T_{sb}\tilde{p}_b$$

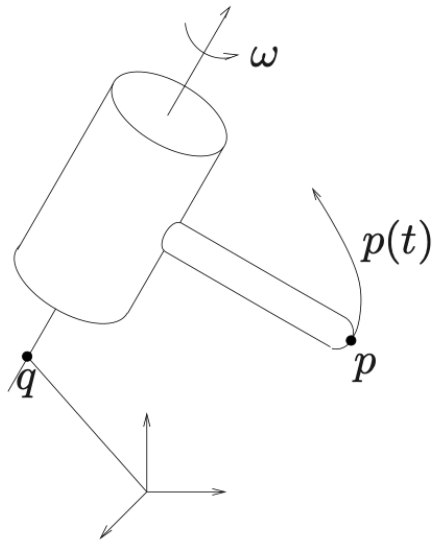
Geometrical Interpretation of Rigid-body Motion

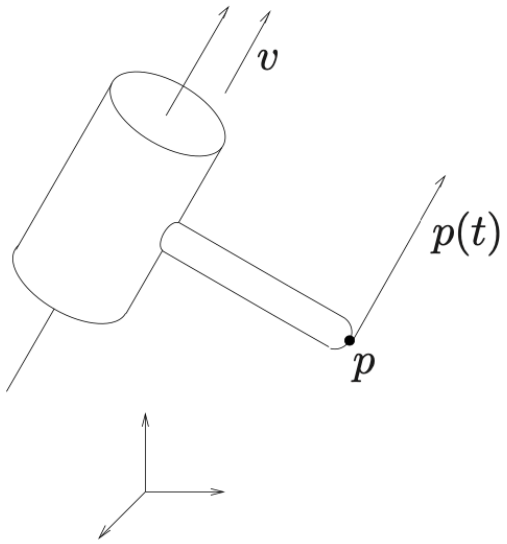
- Screw motion:
  - Any rigid body motion is equivalent to rotating about one axis while also translating along axis
  - The axis may not pass the origin



Exponential Map of Rigid-body Motion

- Differential equation of rigid-body motion:
  - $\dot{p}(t) = \omega \times (p(t) - q) + v = [\omega]p(t) - \omega \times q + v$
  - $A \triangleq \begin{bmatrix} [\omega] & -[\omega]q + v \\ 0 & 0 \end{bmatrix}, \tilde{p}(t) = \begin{bmatrix} p(t) \\ 1 \end{bmatrix}$
  - $\dot{\tilde{p}}(t) = \begin{bmatrix} \dot{p}(t) \\ 0 \end{bmatrix} = A \begin{bmatrix} p(t) \\ 1 \end{bmatrix} = A\tilde{p}(t)$
  - $\tilde{p}(t) = e^{At}\tilde{p}(0)$





Next

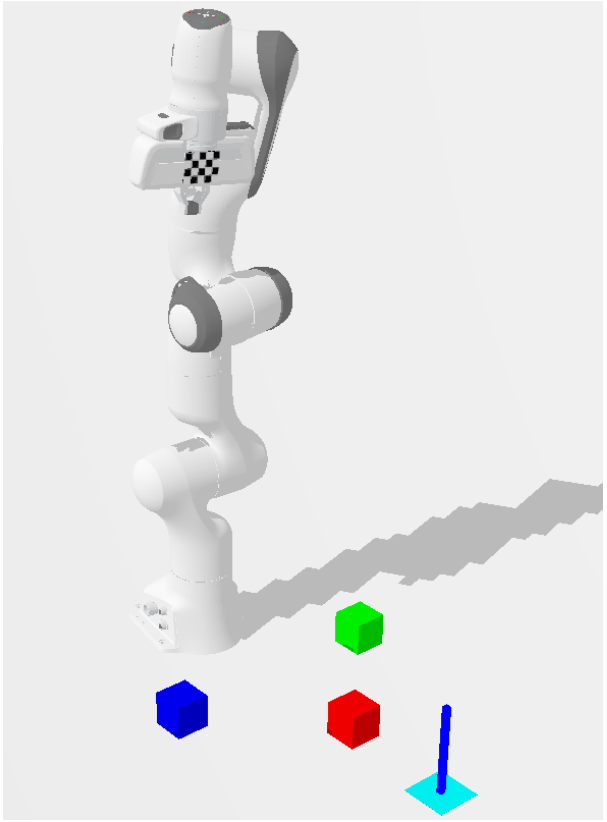
- General Rigid-body Velocity
- Forward Kinematics
- Velocity Kinematics
- Inverse Kinematics

Mid Term Project

- Task: Stack box in SAPIEN

- Objective: Pick up the three box and stack them on the cyan plane. The elevation of three box should be: blue > green > red

- You will need to implement each subtask to achieve the final objective. The main structure of the project code is **provided**, you only need to implement the functions marked as “unimplemented”.



Mid Term Project

- This project cover the content **from lecture 3 to lecture 7**. It does **not** include reinforcement learning and physical simulation.

- This project includes two parts:
  - In the first part, you are asked to move you robot hand to pick the box (**unphysically, do not consider robot dynamics**) based on camera observation. It only covers the content **from lecture 4 to lecture 5**.

  - In the second part, you are asked to move you hand to pick the box (**physically**). You need to implement a motion planning algorithm and execute it with a PID controller. Then you will need to stack the box to achieve the desired configuration. It covers the content **from lecture 3 to lecture 7.**

First Part of the Project

- You should write the code in “env/hw1_env.py” following the instructions on comments, including the function input and output type.

- Run “hw1.py” to evaluate your code. No need to change this file.

Topic

- Rotation and SO(3)
- Rotation Parameterizations
- Learning to Predict Rotation by NN
- Homogenous Transformation and SE(3)
- **Case-Study: Hand-eye Calibration**

Vision-based Robotic Planning

- Visual perception comes from camera/lidar, the position and orientation is captured in sensor frame
  - E.g., point cloud in camera frame

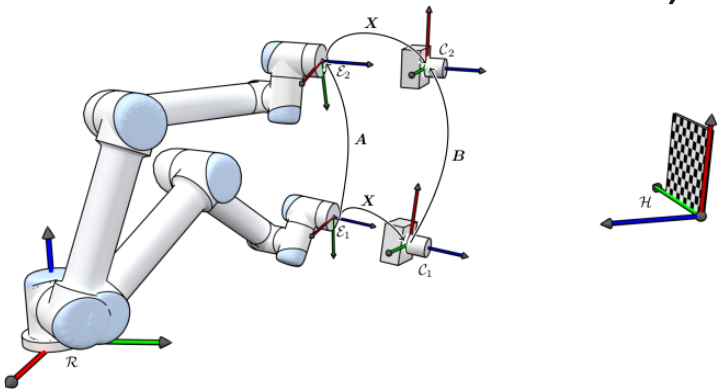
- Moving signal is command in robot frame
  - E.g., move the robot hand left in robot base frame

- Hand-eye Calibration computes the transformation from camera to robot

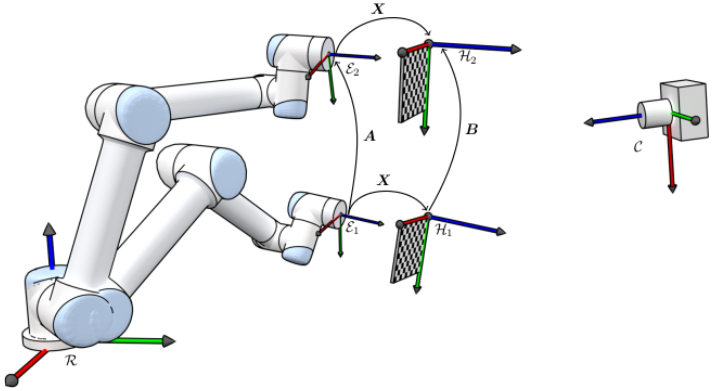
Settings of Hand-eye Calibration

- There are two kinds of problem for hand-eye calibration

- Eye-in-hand (camera mounted on hand):



- Eye-to-hand (camera not fixed with hand):



https://www.torsteinmyhre.name/snippets/robcam_calibration.html

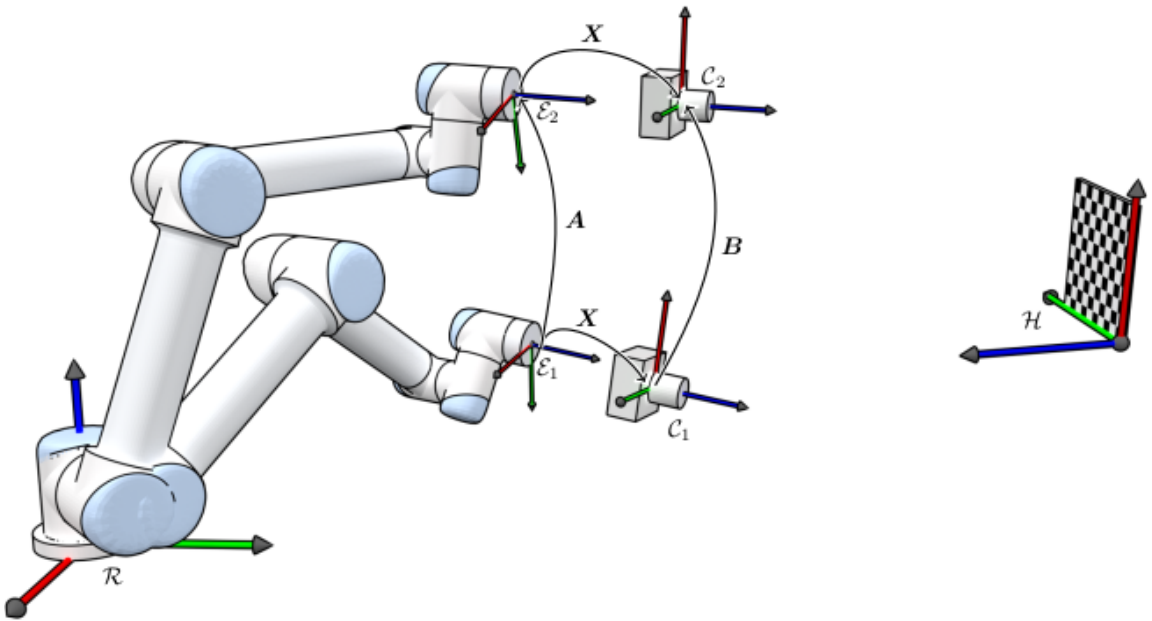
Hand-Eye Transformation Equation

Take *eye-in-hand* (e.g. camera fixed to hand) as example

- Goal: transformation from camera to hand T_{hc}

- Denote: spatial frame (robot base) $\{s\}$, camera $\{c\}$,hand $\{h\}$, and marker (auxiliary object) $\{m\}$

$$T_{sh}T_{hc}T_{cm} = \mathrm{T}_{sm}$$



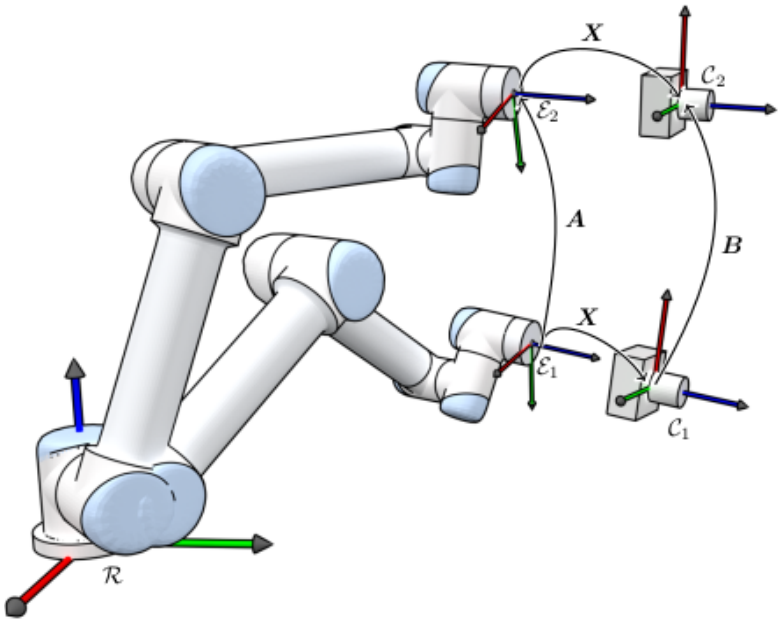
Hand-Eye Transformation Equation

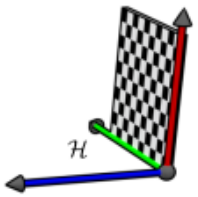
Take *eye-in-hand* (e.g. camera fixed to hand) as example

- Assume we can get the pose of marker in camera frame

$$T_{sh}T_{hc}T_{cm} = \mathrm{T}_{sm}$$

Known Variant	What we want Constant	Assume known Variant	Unknown Constant





Capture Calibration Data

To solve the hand-eye transformation equation, we need to prepare multiple pairs of T_{sh} and T_{cm}

- Repeat the following steps for n-times:
  1. Move the robot hand to a target pose, where camera can see the marker
  2. Capture the T_{sh}^{i} for i-th pose of hand to base, often calculated by forward kinematics
  3. Capture the T_{cm}^{i} for i-th pose of marker to camera, calculated by a marker-specific algorithm

AX=XB for Hand-Eye Calibration

- The pose from marker to spatial frame T_{sm} is fixed

$$T_{sh}^{i}\, T_{hc}\, T_{cm}^{i} = \mathrm{T}_{sm} = T_{sh}^{i+1}\; T_{hc}\, T_{cm}^{i+1}$$

$$\left(T_{sh}^{i+1}\right)^{-1} T_{sh}^{i} T_{hc} = T_{hc}\, T_{cm}^{i+1}\, \left(T_{cm}^{i}\right)^{-1}$$

- Now we get a $AX = XB$ type function with constraints
  - $\mathrm{A} = \left(T_{sh}^{i+1}\right)^{-1} T_{sh}^{i}$ and $B = T_{cm}^{i+1}\, \left(T_{cm}^{i}\right)^{-1}$ are all known

- It is common to use multiple pairs of data for the equation. Actually, **at least three pairs** are necessary in order for a unique solution.

Horaud et al., "**Hand-eye Calibration**", *IJRR 1995*

Solving AX=XB

Note that the $X \in SE(3)$, which is a constrain to this equation

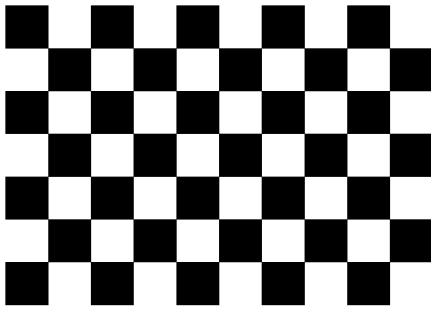
- Two mainstream to solve this equation
  1. Determine first rotation and then translation[1]
  2. Determine rotation and translation simultaneously[2]
- To solve the equation more precisely:
  - Poses of hand are chosen follow some solve-specific guidelines
  - More data

Tsai et al., "**A new technique for fully autonomous and efficient 3 D robotics hand/eye calibration**", *TRO 1989*

Daniilidis, "**Hand-Eye Calibration Using Dual Quaternions**", *IJRR 1999*

Markers for Hand-eye Calibration

- Checkerboard is a most common visual marker in robotics:
  - Checkerboard pose can be easily solved using standard method like PnP



- As long as we have method to estimate its pose, anything can be a marker