Locomotion: Mobile Robots

EEL 4930/5934: Autonomous Robots

Spring 2023

Md Jahidul Islam

Lecture 5

ECE | Florida
Electrical & Computer Engineering
UF | UNIVERSITY of FLORIDA

Robot Locomotion

⇒ ***Kinematics:*** *Mathematical model of motion*

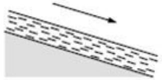
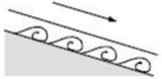
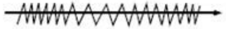
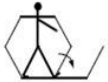
⇒ ***Locomotion:*** *Actual motion*

- *Motion type and constraints*
- *Motion geometry*

⇒ ***Things to know***

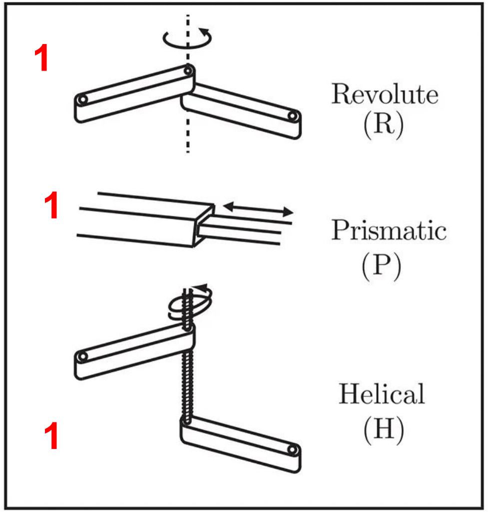
- *DOF: Degrees Of Freedom*
- *Motion gaits*
- *Robot workspace*

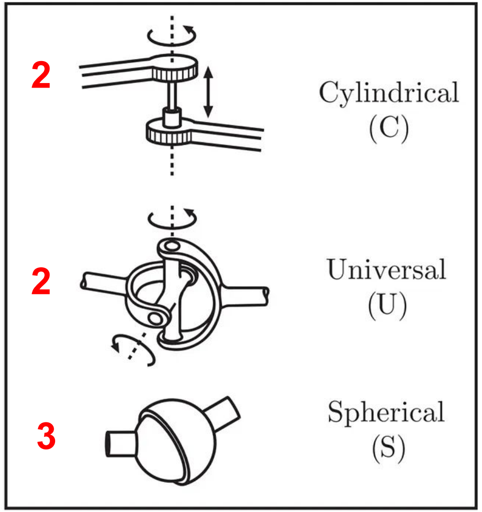
Locomotion Concepts: Principles

Type of motion		Resistance to motion	Basic kinematics of motion	
Flow in a Channel		Hydrodynamic forces	Eddies	
Crawl		Friction forces	Longitudinal vibration	
Sliding		Friction forces	Transverse vibration	
Running		Loss of kinetic energy	Oscillatory movement of a multi-link pendulum	
Jumping		Loss of kinetic energy	Oscillatory movement of a multi-link pendulum	
Walking		Gravitational forces	Rolling of a polygon (see figure 2.2)	

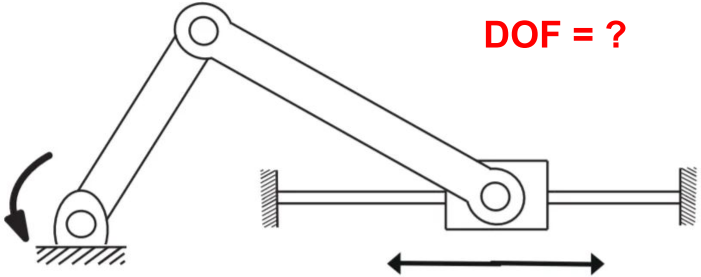
R. Siegwart

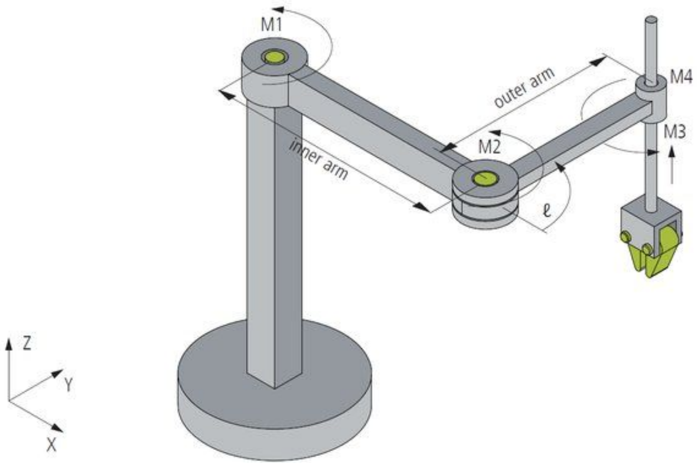
DOF: Degrees Of Freedom





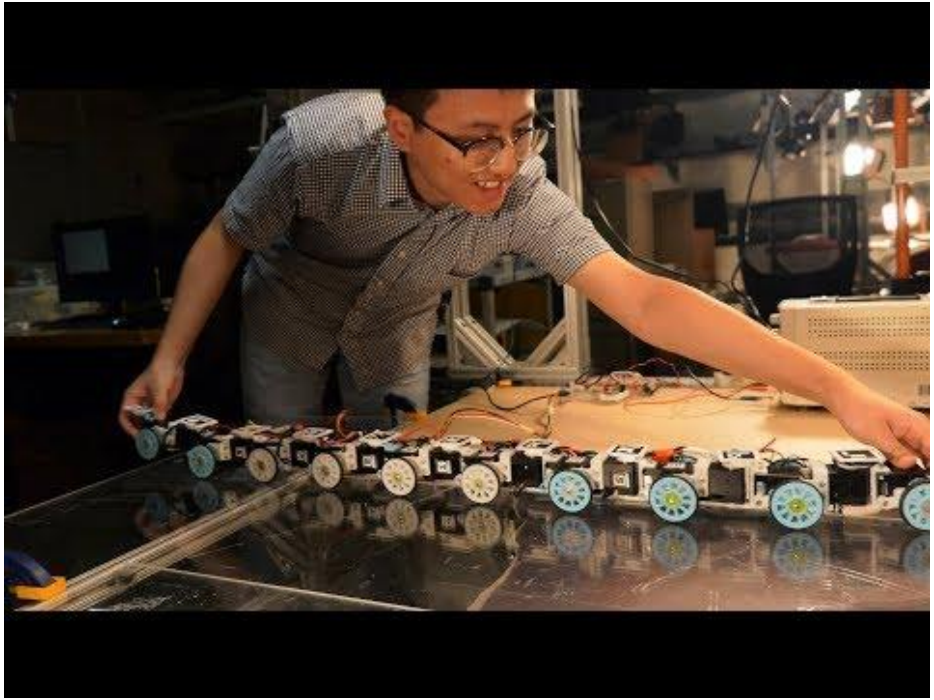
DOF = ?



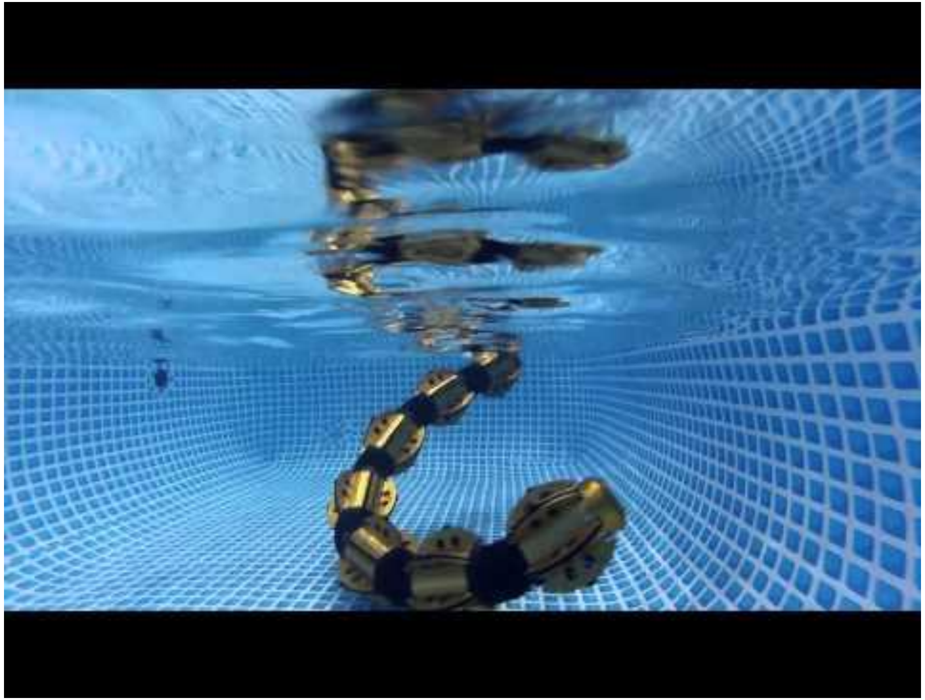


$$\text{dof} = \underbrace{m(N-1)}_{\text{rigid body freedoms}} - \underbrace{\sum_{i=1}^{J} c_i}_{\text{joint constraints}}$$

Sliding and Crawling Gaits



https://youtu.be/pv_MknD6jks



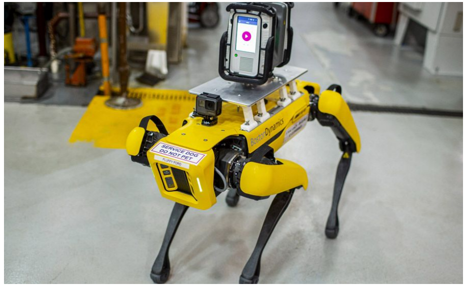
https://youtu.be/vCrN47cOmHQ

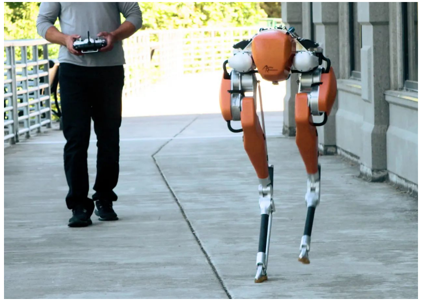
Swimming Gaits

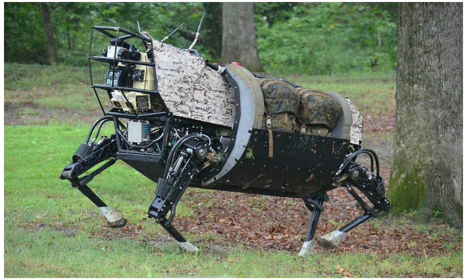
https://youtu.be/2vy861m2MAE

https://youtu.be/3H3GMxVATz8

Legged (Walking) Gaits







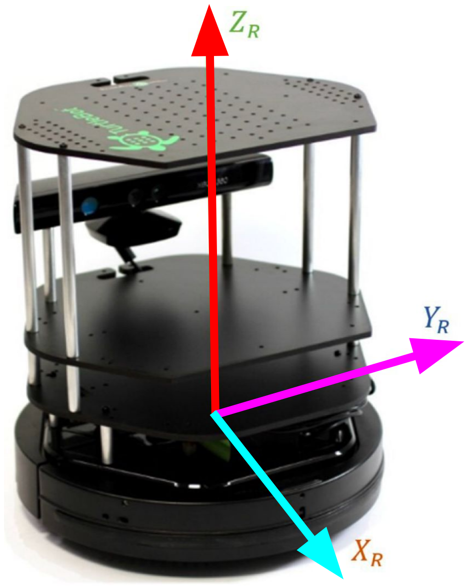
https://youtu.be/zrzCYgNnHfI

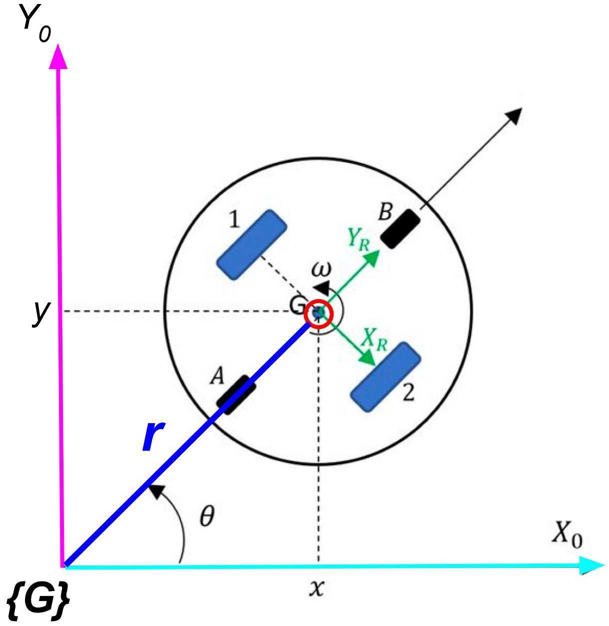
Wheel (Rolling) Gaits

https://youtu.be/w-0GMURCDBM

https://youtu.be/_rPvKlvyw2w

TurtleBot Motion Model



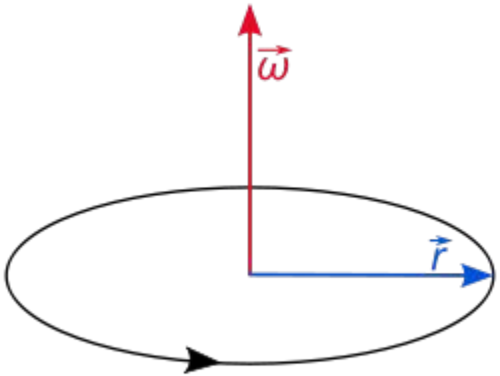


$$x = \mathbf{r} \cdot cos\theta$$

$$y = \mathbf{r} \cdot sin\theta$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix}$$

Basics: Linear And Angular Velocity

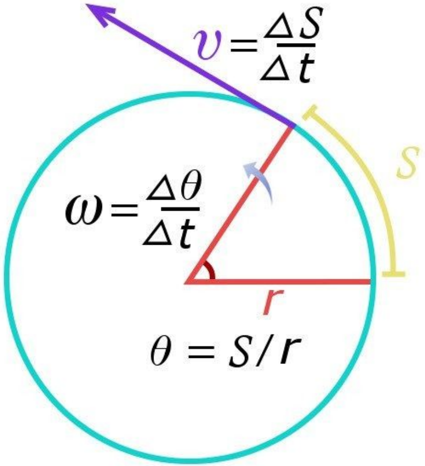


Linear velocity, $v = \frac{\Delta S}{\Delta t}$

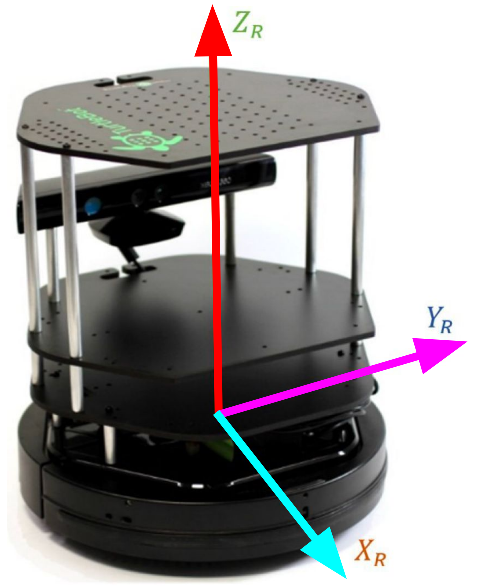
But $S = r \cdot \theta$,

Making $v = \frac{r \cdot \Delta\theta}{\Delta t}$

Or $\boxed{v = r \cdot \omega}$



Move TurtleBot In ROS



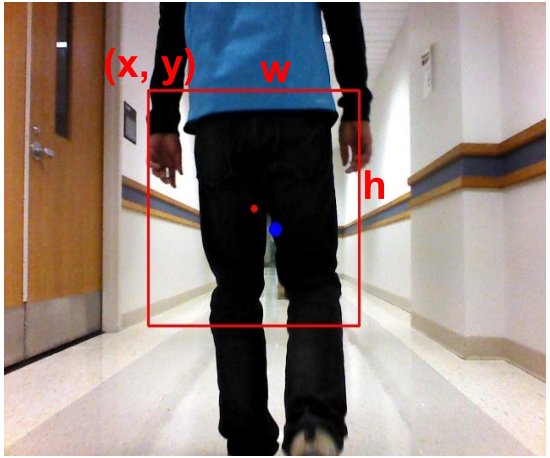

```
cmd_pub = rospy.Publisher('/cmd_vel_mux/input/teleop',
                                  Twist,
                                  queue_size=1)
```

```
base_cmd = Twist()

base_cmd.linear.x = 0.2

base_cmd.angular.z = -0.3

cmd_pub.publish(base_cmd)
```

⇒ *Notes*

- *Tune linear and angular displacements*
- *See [Twist](#) message documentation in ROS*
- *Move a Turtlebot from command line!*

Person Following: 3D Bounding Box (BBox)

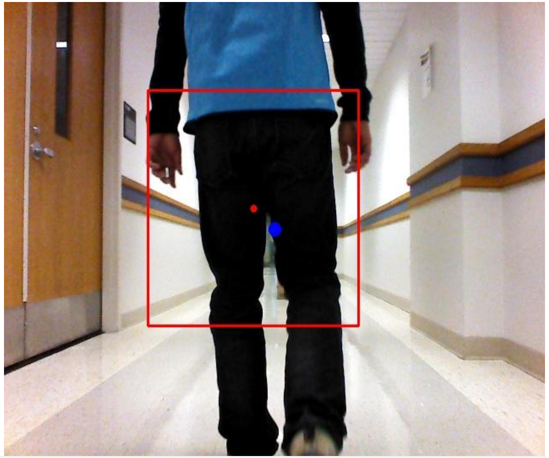


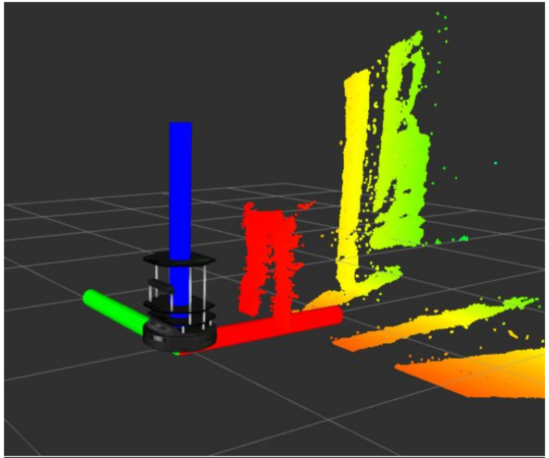

```
x_c, y_c = (x + w/2), (y + h/2)
offset_x = x_c - im_width/2
theta = offset_x / im_width
base_cmd.angular.z = - 2 * theta
base_cmd.linear.x = d[x_c, y_c] - d0
```

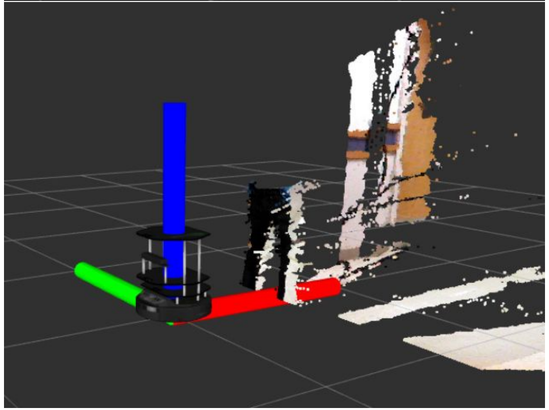
⇒ ***Given a bounding box (x, y, w, h)***

- *Get the center (`x_c, y_c`) of BBox*
- *Calculate **`offset_x`** from center with respect to the image width*
- *Calculate angular offset **`theta`***
- *Rotate with angular.z*
- *Get the depth value **`d[x_c, y_c]`***
- *Move with **`linear.x`** (maintain a minimum safe distance **`d0`**)*

Person Following: Point Cloud





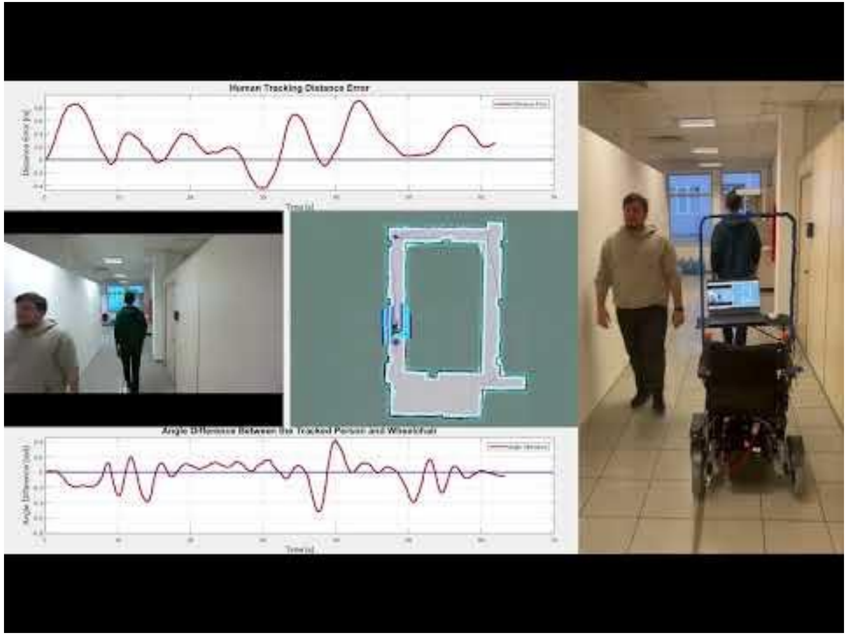


⇒ How to get the bounding box?

- *Get the center of the point cloud*
  - *Closest point cloud usually works!*
- *Human leg detectors (by laser scanners)*
- *Person detectors (by ML/DL methods)*

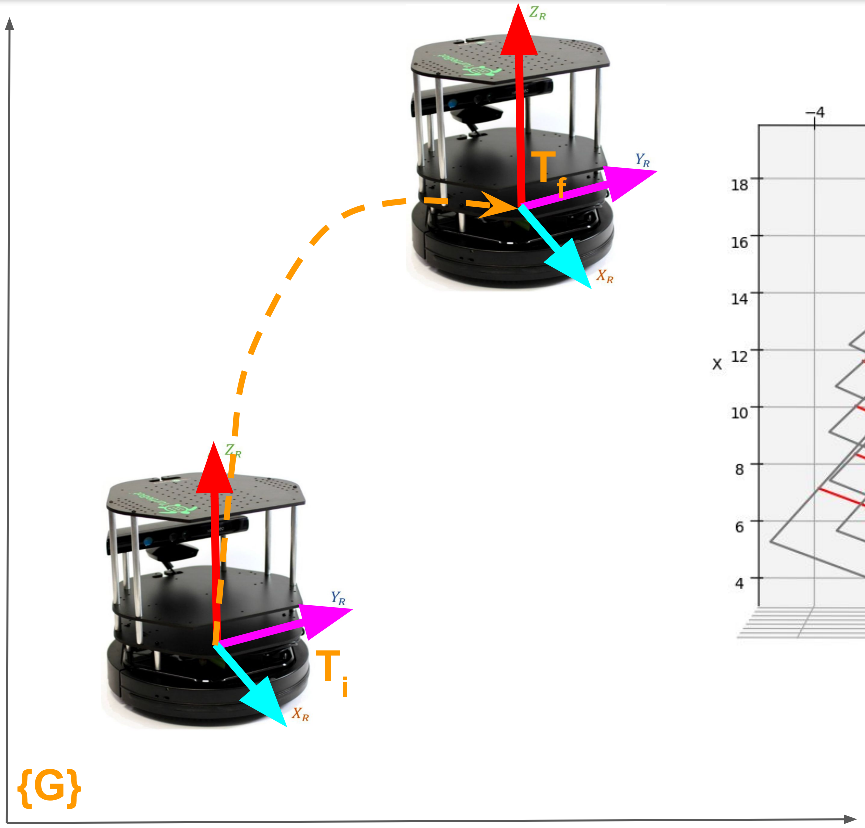
Sample Projects!

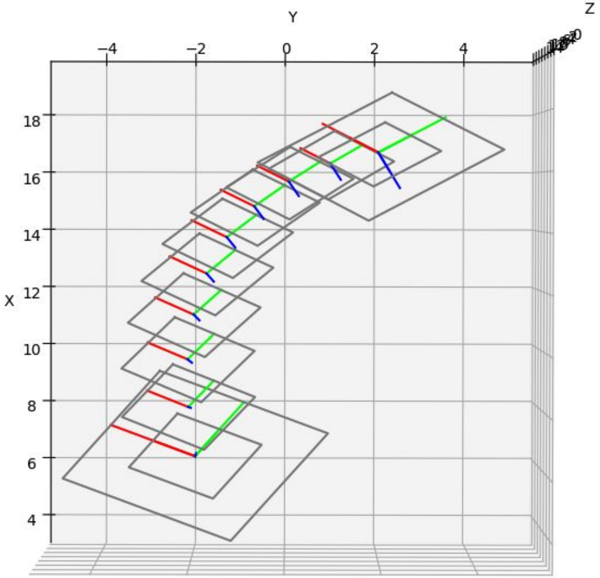
https://youtu.be/c53ISOKHKLc

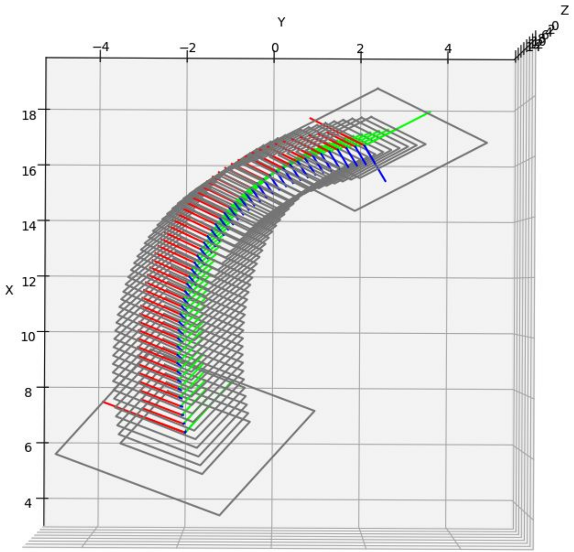


https://youtu.be/kaZIwYa-J60

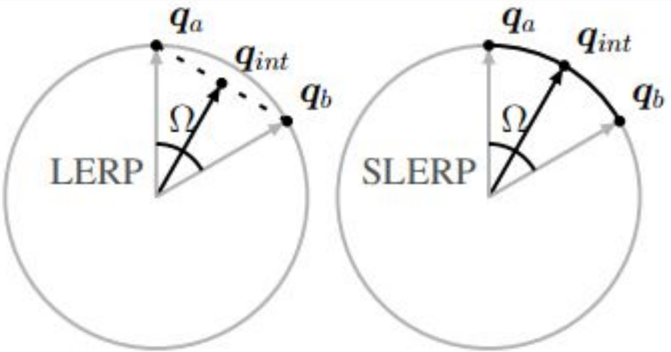
Pose Interpolation

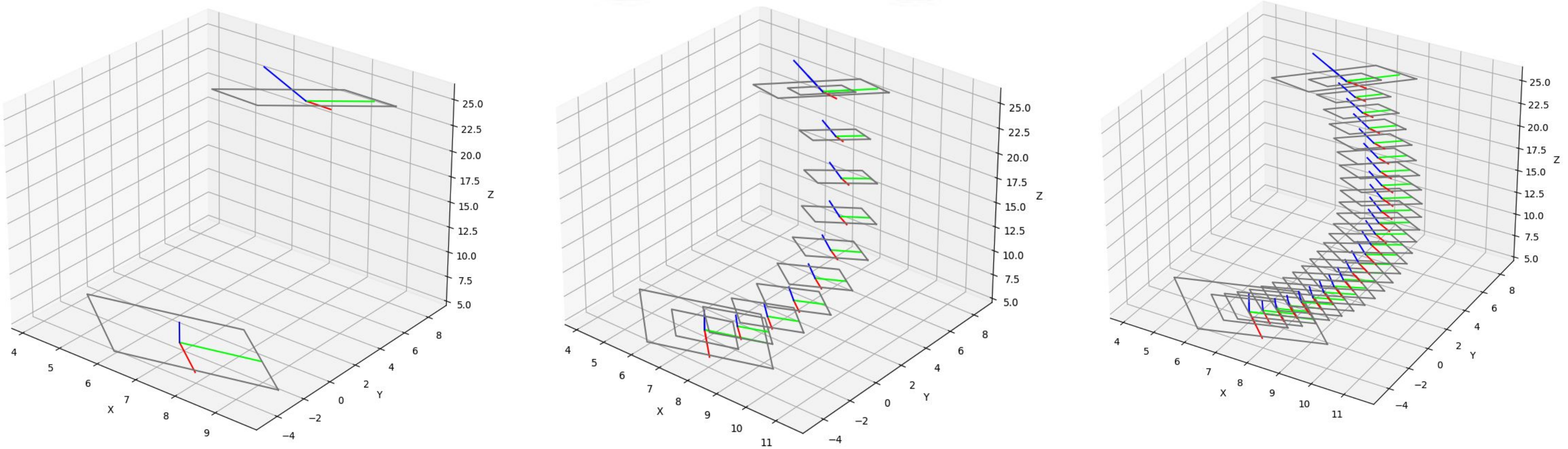






SLERP: Spherical Linear Interpolation





ECE Department of Electrical & Computer Engineering

UF UNIVERSITY of FLORIDA

SLERP: Quaternion Space

Rotation $R_{u,\theta}$:

axis u, angle θ

$$\begin{pmatrix} q_r^2+q_i^2-q_j^2-q_k^2 & 2q_iq_j-2q_rq_k & 2q_iq_k+2q_rq_j \\ 2q_iq_j+2q_rq_k & q_r^2-q_i^2+q_j^2-q_k^2 & 2q_jq_k-2q_rq_i \\ 2q_iq_k-2q_rq_j & 2q_jq_k+2q_rq_i & q_r^2-q_i^2-q_j^2+q_k^2 \end{pmatrix}$$

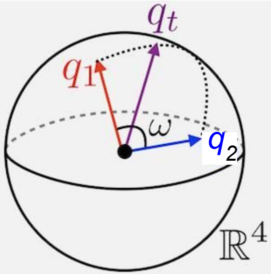
Unit quaternion:

$$q = \cos(\theta/2) + (u_x \mathrm{i} + u_y \mathrm{j} + u_z \mathrm{k}) \sin(\theta/2).$$

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

Spherical Linear Interpolation (SLERP):

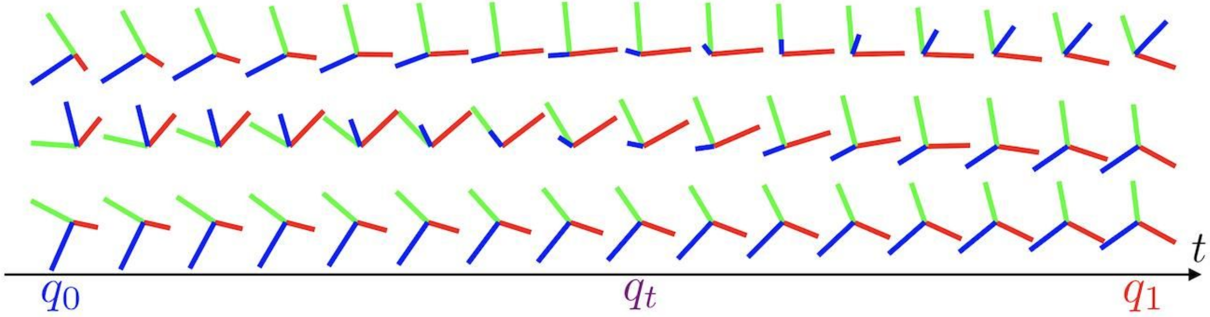
$$Slerp(q_1, q_2; \mu) = \frac{\sin\,(1-\mu)\theta}{\sin\theta} q_1 + \frac{\sin\mu\theta}{\sin\theta} q_2$$



Can you derive this?

Refs:

- *Wiki*
- *GeoTools*



@gabrielpeyre

Review: Quaternion

Unit quaternion: $\mathbf{q} = \begin{bmatrix} \bar{q} \\ q_4 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}; \quad |q| = 1$

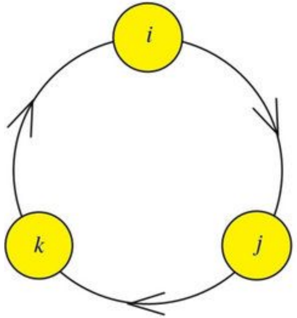
$$\mathbf{q} = q_1 i + q_2 j + q_3 k + q_4$$

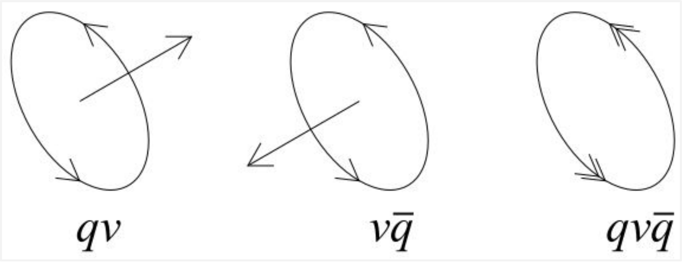
$$i^2 = j^2 = k^2 = ijk = -1$$

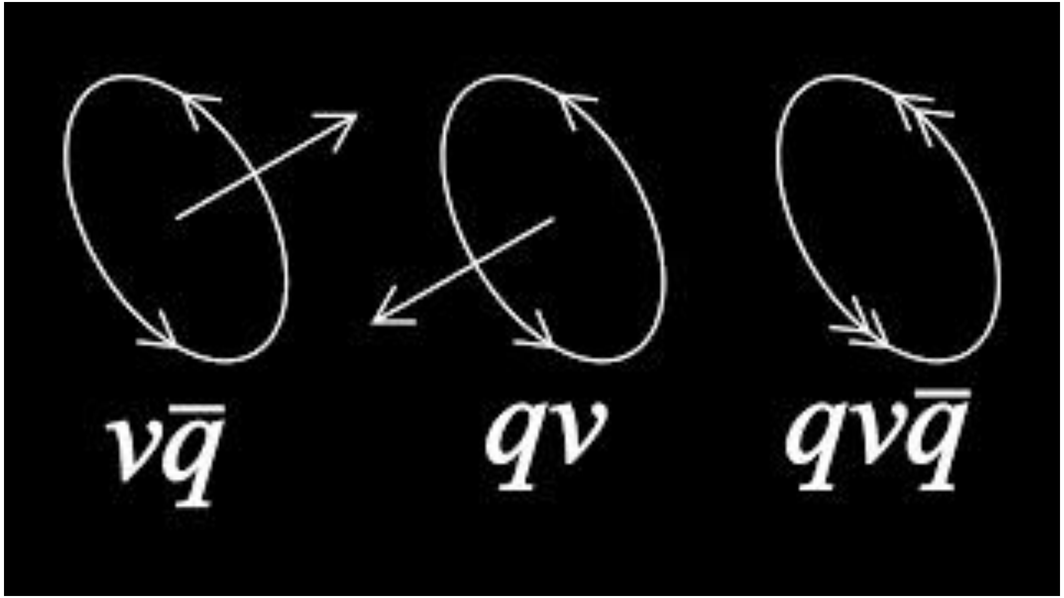
$$ij = -ji = k$$

$$jk = -kj = i$$

$$ki = -ik = j$$







https://youtu.be/jTgdKoQv738

** See good visualizations: https://eater.net/quaternions

Review: Quaternion Rotation

Unit quaternion: $\mathbf{q} = \begin{bmatrix} \bar{q} \\ q_4 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}; \quad |q| = 1$

$$\mathbf{q} = q_1 i + q_2 j + q_3 k + q_4$$

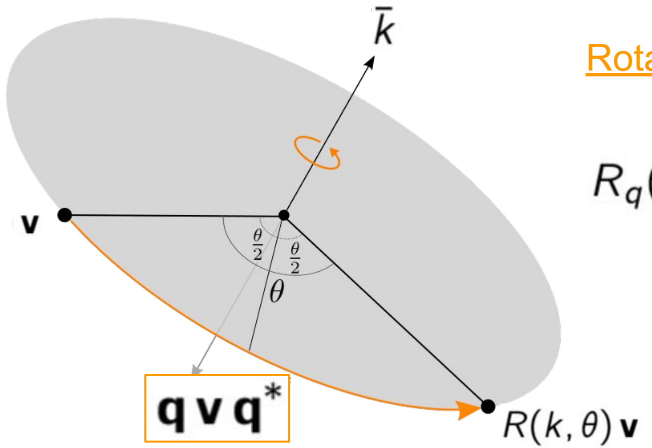
$$i^2 = j^2 = k^2 = ijk = -1$$

Rotation around a unit quaternion

$$\mathbf{q} = \begin{bmatrix} \bar{k}\ s(\theta/2) \\ c(\theta/2) \end{bmatrix} = \begin{bmatrix} k_x\ s(\theta/2) \\ k_y\ s(\theta/2) \\ k_z\ s(\theta/2) \\ c(\theta/2) \end{bmatrix}; \quad |q| = |k| = 1$$

See more at:

- OpenGL Blog Stanford Graphics Notes



Rotation matrix from its quaternion:

$$R_q(k,\theta) = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2(q_1 q_2 - q_3 q_4) & 2(q_1 q_3 + q_2 q_4) \\ 2(q_1 q_2 + q_3 q_4) & 1 - 2q_1^2 - 2q_3^2 & 2(q_2 q_3 - q_1 q_4) \\ 2(q_1 q_3 - q_2 q_4) & 2(q_2 q_3 + q_1 q_4) & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix}$$

How find quaternion from its rotation matrix?

Review: Quaternion Rotation

Unit quaternion: $\mathbf{q} = \begin{bmatrix} \bar{q} \\ q_4 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}; \quad |q| = 1$

$$\mathbf{q} = q_1 i + q_2 j + q_3 k + q_4$$

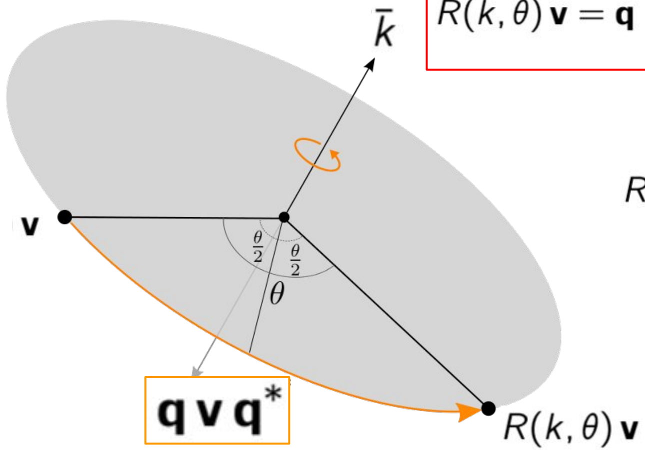
$$i^2 = j^2 = k^2 = ijk = -1$$

$$R(k,\theta)\,\mathbf{v} = \mathbf{q} \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} \mathbf{q}^* = \mathbf{q} \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} \mathbf{q}^{-1}$$

Rotation around a unit quaternion

$$\mathbf{q} = \begin{bmatrix} \bar{k}\, s(\theta/2) \\ c(\theta/2) \end{bmatrix} = \begin{bmatrix} k_x\, s(\theta/2) \\ k_y\, s(\theta/2) \\ k_z\, s(\theta/2) \\ c(\theta/2) \end{bmatrix}; \quad |q| = |k| = 1$$

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{|q|} = \mathbf{q}^* = -q_1 i - q_2 j - q_3 k + q_4 = \begin{bmatrix} -q_1 \\ -q_2 \\ -q_3 \\ q_4 \end{bmatrix}$$



$$R(k,\theta) = \begin{bmatrix} k_x^2(1-c\theta)+c\theta & k_x\, k_y(1-c\theta)-k_z s\theta & k_x\, k_z(1-c\theta)+k_y s\theta \\ k_x\, k_y(1-c\theta)+k_z s\theta & k_y^2(1-c\theta)+c\theta & k_y\, k_z(1-c\theta)-k_x s\theta \\ k_x\, k_z(1-c\theta)+k_y s\theta & k_y\, k_z(1-c\theta)+k_x s\theta & k_z^2(1-c\theta)+c\theta \end{bmatrix}$$

$$= \begin{bmatrix} 1-2q_2^2-2q_3^2 & 2(q_1q_2-q_3q_4) & 2(q_1q_3+q_2q_4) \\ 2(q_1q_2+q_3q_4) & 1-2q_1^2-2q_3^2 & 2(q_2q_3-q_1q_4) \\ 2(q_1q_3-q_2q_4) & 2(q_2q_3+q_1q_4) & 1-2q_1^2-2q_2^2 \end{bmatrix}$$

Review: Properties of Quaternion

Unit quaternion: $\mathbf{q} = \begin{bmatrix} \bar{q} \\ q_4 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} ; \quad |q| = 1$

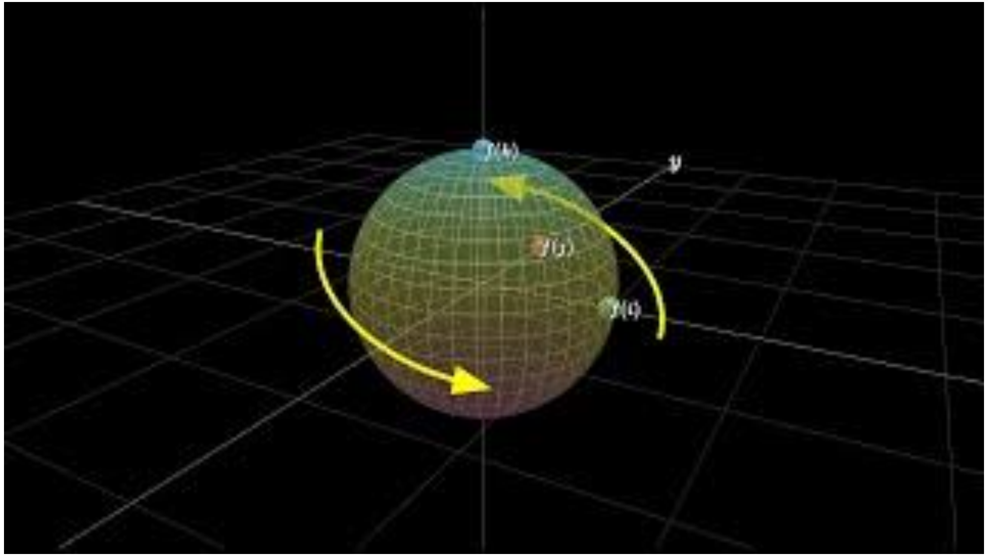
$$\mathbf{q} = q_1 i + q_2 j + q_3 k + q_4$$

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{|q|} = \mathbf{q}^* = -q_1 i - q_2 j - q_3 k + q_4 = \begin{bmatrix} -q_1 \\ -q_2 \\ -q_3 \\ q_4 \end{bmatrix}$$

$$qq^* = q^*q \qquad |pq| = \sqrt{pq(pq)^*}$$

$$R(k,\theta)\,\mathbf{v} = \mathbf{q} \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} \mathbf{q}^* = \mathbf{q} \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} \mathbf{q}^{-1} = \mathbf{q} \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix} \mathbf{q}^*$$

$pq \neq qp$	(NOT commutative)
$(op)q = o(pq)$	(associative)
$o(p+q) = op + oq$	(distributive across addition)



https://youtu.be/zjMuIxRvygQ

Some basic properties of quaternion:

- https://users.ncsa.illinois.edu/kindr/emtc/quaternions/
- http://www.songho.ca/math/quaternion/quaternion.html
- https://en.wikipedia.org/wiki/Quaternion

Using SLERP

Given poses

T1 = {R1, t1}

T2 = {R2, t2}

```
t_lerp = μ * t1 + (1-μ) * t2
```

$$Slerp(q_1, q_2; \boldsymbol{\mu}) = \frac{\sin\,(1-\mu)\theta}{\sin\theta} q_1 + \frac{\sin\mu\theta}{\sin\theta} q_2$$

```
q1 = quaternion_from_rotation(R1)
q2 = quaternion_from_rotation(R2)
μ = 0.3 # choose [0, 1]
q_slerp = quaternion_slerp(q1, q2, μ)
R_slerp = rotation_from_quaternion(q)
```

Interpolated pose

Tμ = {R_slerp, t_lerp}

Using SLERP: Example

Given poses

$$T1 = \begin{bmatrix} 0.866 & -0.5 & 0 & 5 \\ 0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T2 = \begin{bmatrix} 0.866 & 0 & 0.5 & 5 \\ 0. & 1 & 0 & 0 \\ -0.5 & 0 & 0.866 & 15 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$q1 = quaternion\_from\_rotation(R1) = \begin{bmatrix} 0.96592583 & 0. & 0. & 0.25881905 \end{bmatrix}^T$$

$$q1 = quaternion\_from\_rotation(R2) = \begin{bmatrix} 0.96592583 & 0. & 0.25881905 & 0. \end{bmatrix}^T$$

$$\theta = acos(q1^T \cdot q2))$$

$$Slerp(q_1, q_2; \boldsymbol{\mu}) = \frac{\sin\ (1 - \ \mu)\theta}{\sin\theta} q_1 + \frac{\sin\mu\theta}{\sin\theta} q_2$$

$$q_{slerp} = Slerp(q1, q2; \mu = 0.3) = \begin{bmatrix} 0.97985867 & 0. & 0.07926597 & 0.18328637 \end{bmatrix}^T$$

$$R_{slerp} = R\_from\_quaternion(q_{slerp}) = \begin{bmatrix} 0.92024602 & -0.35918948 & 0.15533889 \\ 0.35918948 & 0.93281221 & 0.02905674 \\ -0.15533889 & 0.02905674 & 0.98743381 \end{bmatrix}$$

$$t_{lerp} = \mu * t1 + (1 - \mu) * t2 = \begin{bmatrix} 5. & 0. & 10.5 \end{bmatrix}^T$$

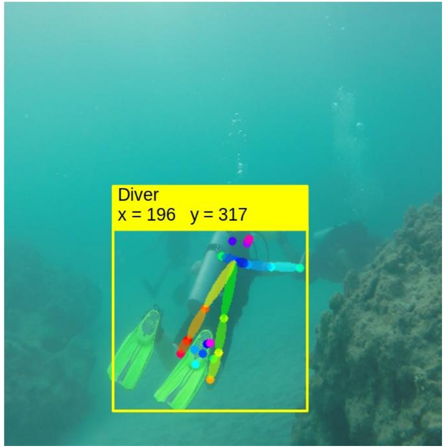
Interpolated pose (μ = 0.3)

$$T_\mu = \begin{bmatrix} 0.92024602 & -0.35918948 & 0.15533889 & 5. \\ 0.35918948 & 0.93281221 & 0.02905674 & 0. \\ -0.15533889 & 0.02905674 & 0.98743381 & 10.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Aerial Domain: 6DOF Motion

https://youtu.be/imt2qZ7uw1s

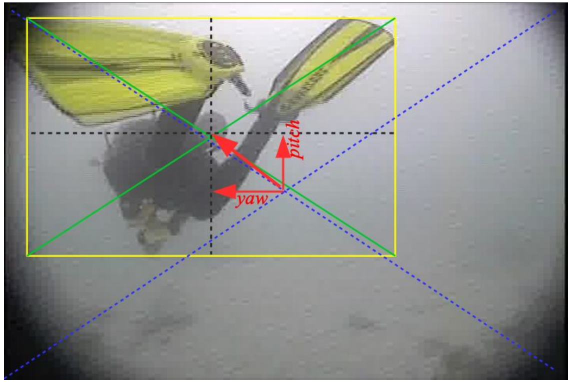
Diver Following: 2D BBox



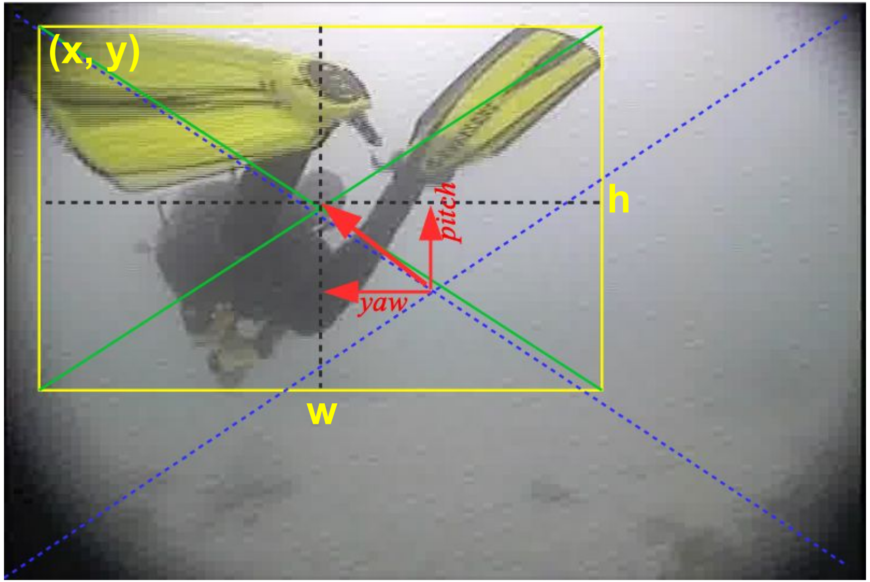
https://youtu.be/sKJs3TqdZJU

⇒ Simplest approach

- *Detect BBox: DL-based detectors or pose estimators*
- *Find the center and offset of the BBox*
- *Feed the filtered data to a BBox-reactive controller*
- *Robot motion:*
    - *Yaw-Pitch modulator*
    - *Separate PID controllers*



Yaw-Pitch Controller: BBox




```
x_c, y_c = (x + w/2), (y + h/2)
x0, y0 = im_width/2, im_height/2
offset_yaw = (x_c - x0)/im_width
offset_pitch = (y_c - y0)/im_height
```

```
yaw_angle ∝ offset_yaw
pitch_angle ∝ offset_pitch
velocity_forward ∝ distance (known?)
```

- How many PID controllers?
- How to set the gains?
- How to tune the hyper-parameters?