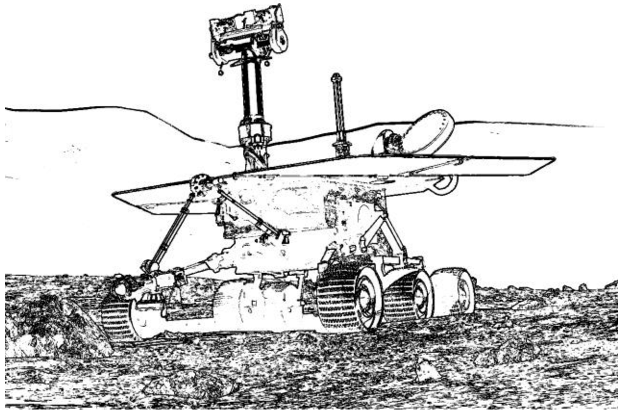


CSC477
Introduction to Mobile Robotics

Florian Shkurti

Week #2: Kinematics and Dynamics

Today’s slides borrow parts of Paul Furgale’s “Representing robot pose” presentation:

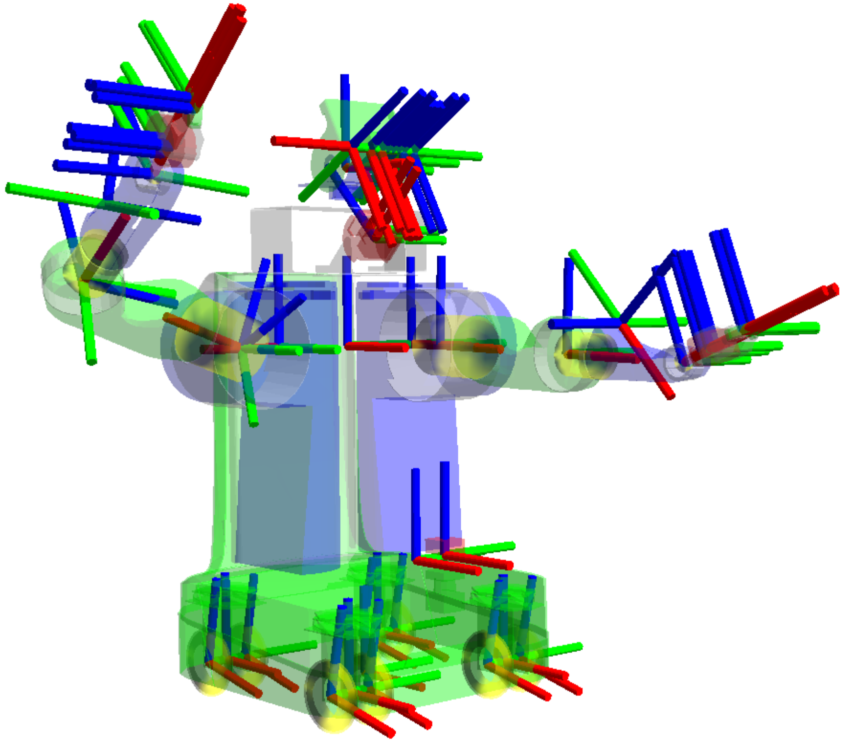
http://paulfurgale.info/news/2014/6/9/representing-robot-pose-the-good-the-bad-and-the-ugly

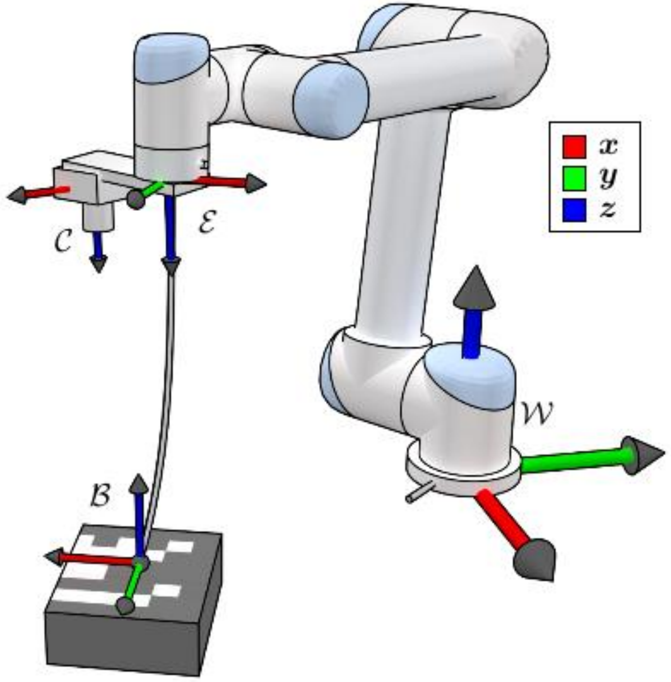
You should absolutely read it.

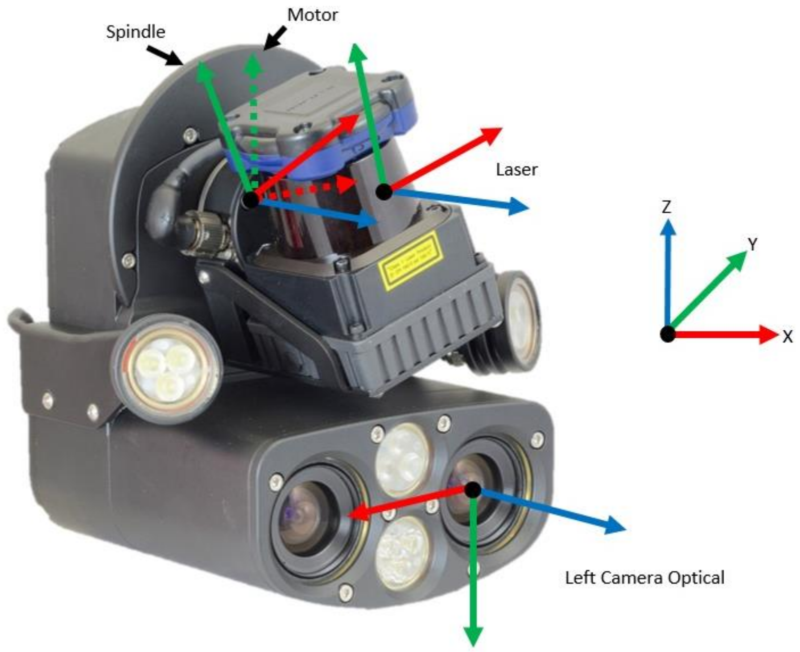
Today’s Agenda

- Frames of reference
- Ways to represent rotations
- Simplified models of vehicles
- Forward and inverse kinematics

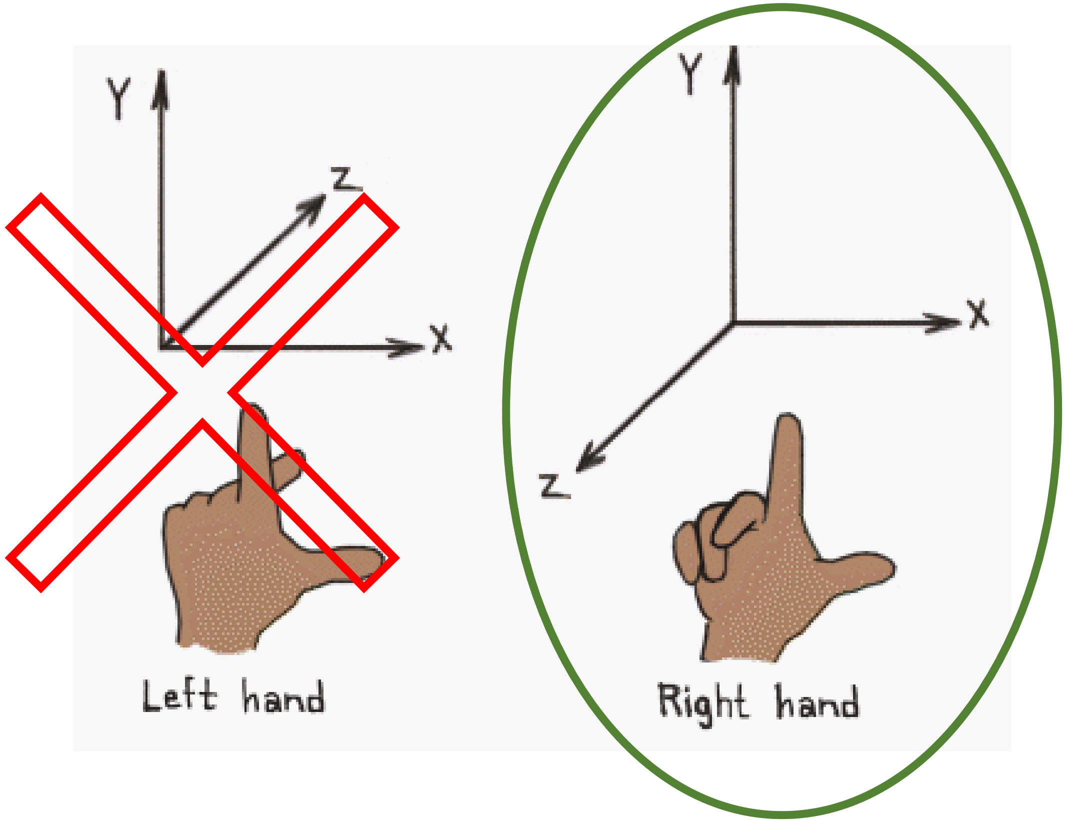
3D frames of reference are everywhere in robotics







Right-handed vs left-handed frames



Unless otherwise specified, we use right-handed frames in robotics

Why do we need to use so many frames?

- Because we want to reason and express quantities relative to their local configuration.

- For example: “grab the bottle behind the cereal bowl”

- This lecture is about defining and representing frames of reference and reasoning about how to express quantities in one frame to quantities in the other.

Rigid-body motion

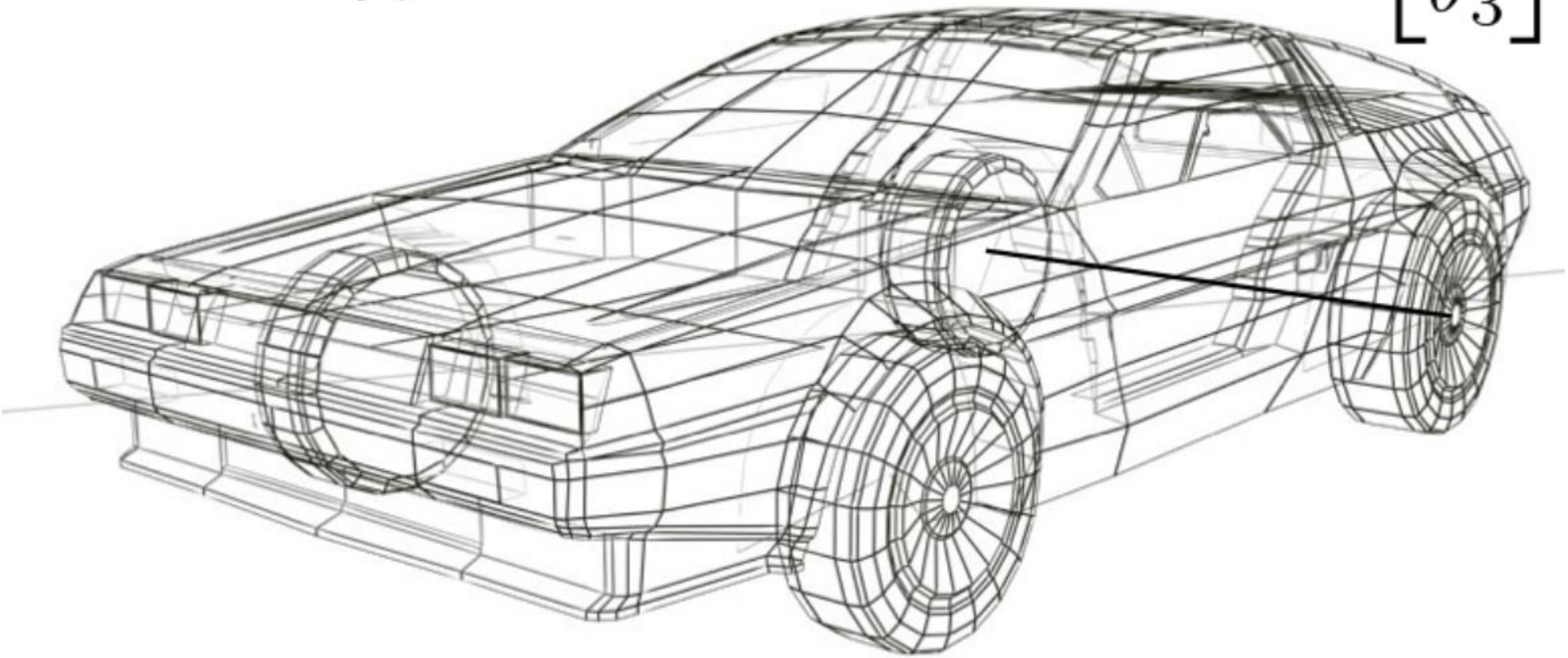
- Motion that can be described by a rotation and translation.

- All the parts making up the body move in unison, and there are no deformations.

- Representing rotations, translations, and vectors in a given frame of reference is often a source of frustration and bugs in robot software because there are so many options.

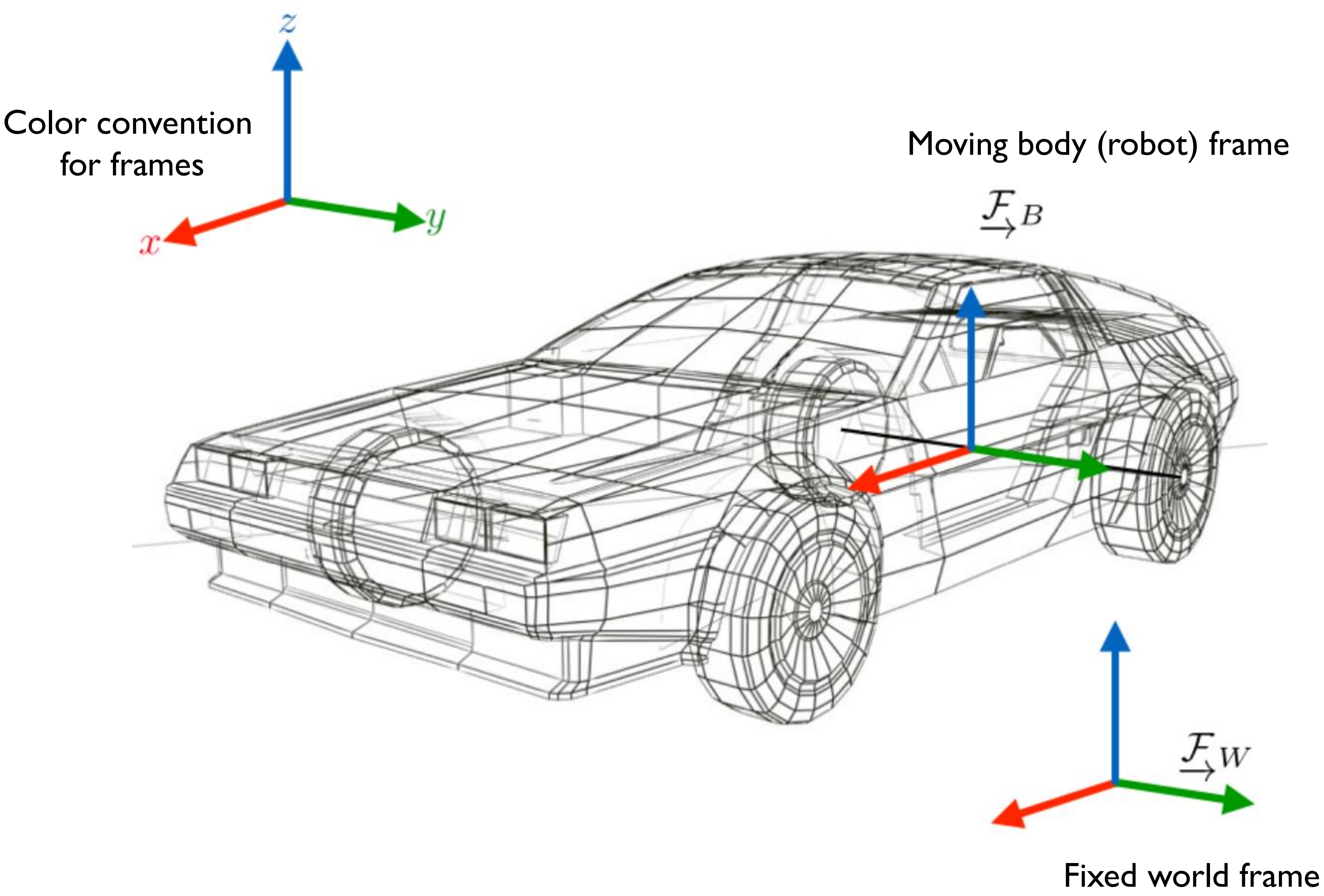
- The “three number” problem
- You are given three numbers representing *the orientation of the robot*
- How many possibilities are there?

$$\mathbf{C} \leftarrow \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$



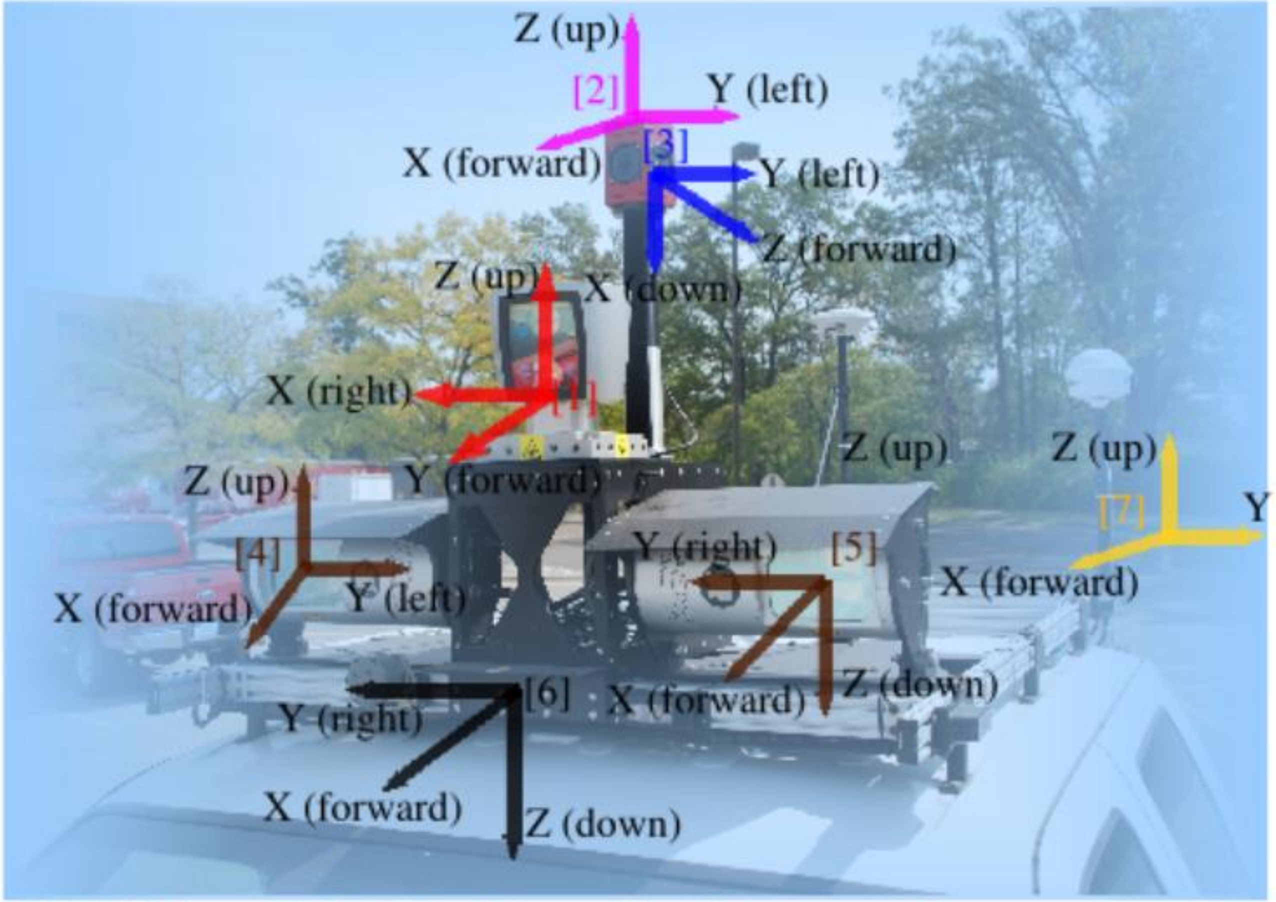
Super sweet wireframe Delorian from http://nanahra.blogspot.com

The answer is meaningless unless I provide a definition of the coordinate frames



z
Color convention for frames
x
y
Moving body (robot) frame
$\underrightarrow{\mathcal{F}}_B$
$\underrightarrow{\mathcal{F}}_W$
Fixed world frame

Always provide a frame diagram



[1] Velodyne, [2] Ladybug3 (actual location: center of camera system),
[3] Ladybug3 Camera 5, [4] Right Riegl, [5] Left Riegl,
[6] Body Frame (actual location: center of rear axle)
[7] Local Frame (Angle between the X-axis and East is known)

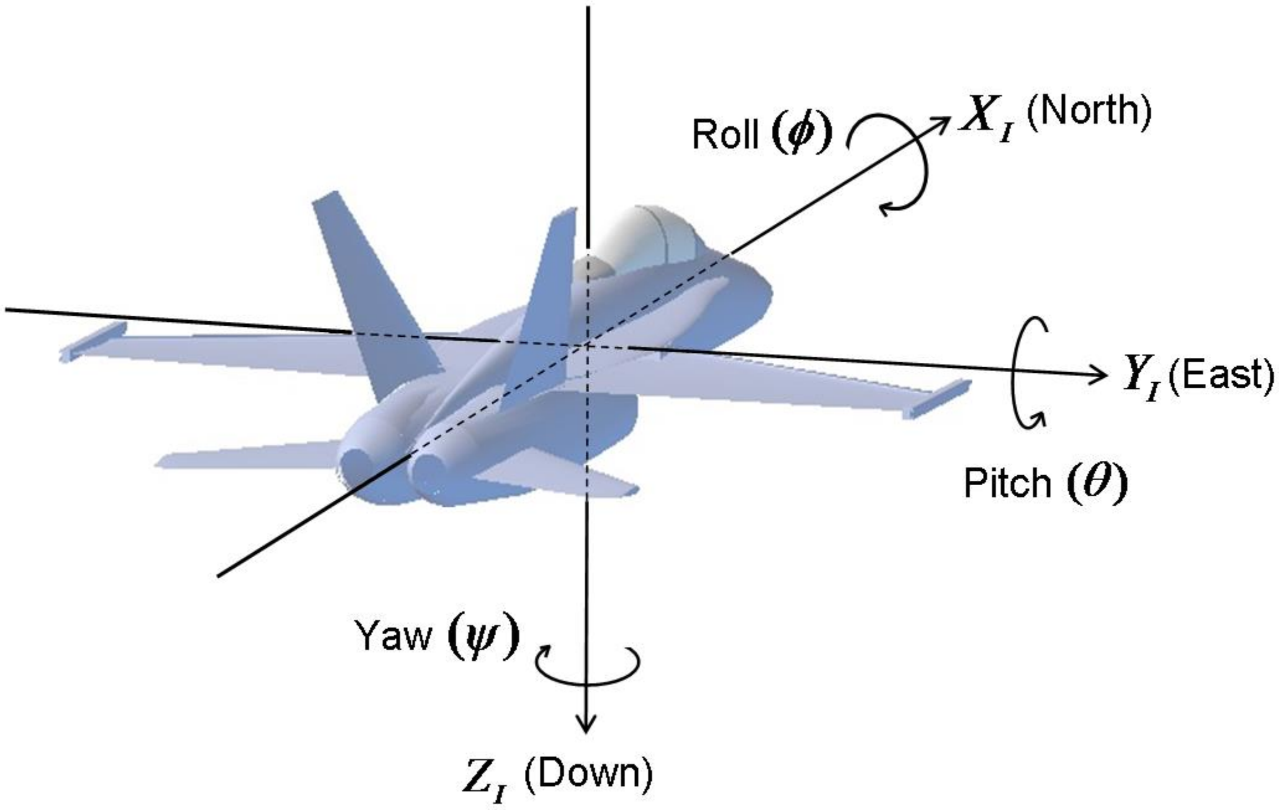
Inertial frames of reference

- G, the global frame of reference is fixed, i.e. with zero velocity in our previous example.
- But, in general it can move as long as it has zero acceleration. Such a frame is called an "inertial" frame of reference.
- Newton's laws hold for inertial reference frames only. For reference frames with non-constant velocity we need the theory of General Relativity.
- So, make sure that your global frame of reference is inertial, preferably fixed.

Today's Agenda

- Frames of reference
- Ways to represent rotations
- Simplified models of vehicles
- Forward and inverse kinematics

Representing Rotations in 3D: Euler Angles



Specification ambiguities in Euler Angles

- Need to specify the axes which each angle refers to.

- There are **12 different valid combinations** of fundamental rotations. Here are the possible axes:

- z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y
- x-y-z, y-z-x, z-y-x, x-z-y, z-y-x, y-x-z

Specification ambiguities in Euler Angles

- Need to specify the axes which each angle refers to.

- There are **12 different valid combinations** of fundamental rotations. Here are the possible axes:

- z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y
- x-y-z, y-z-x, z-y-x, x-z-y, z-y-x, y-x-z

- E.g.: x-y-z rotation with Euler angles (θ, ϕ, ψ) means the rotation can be expressed as a sequence of simple rotations $R_x(\theta)R_y(\phi)R_z(\psi)$

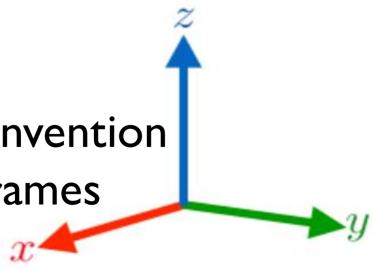
Specification ambiguities in Euler Angles

Simple rotations can be counter-clockwise or clockwise.
This gives **another 2 possibilities.**

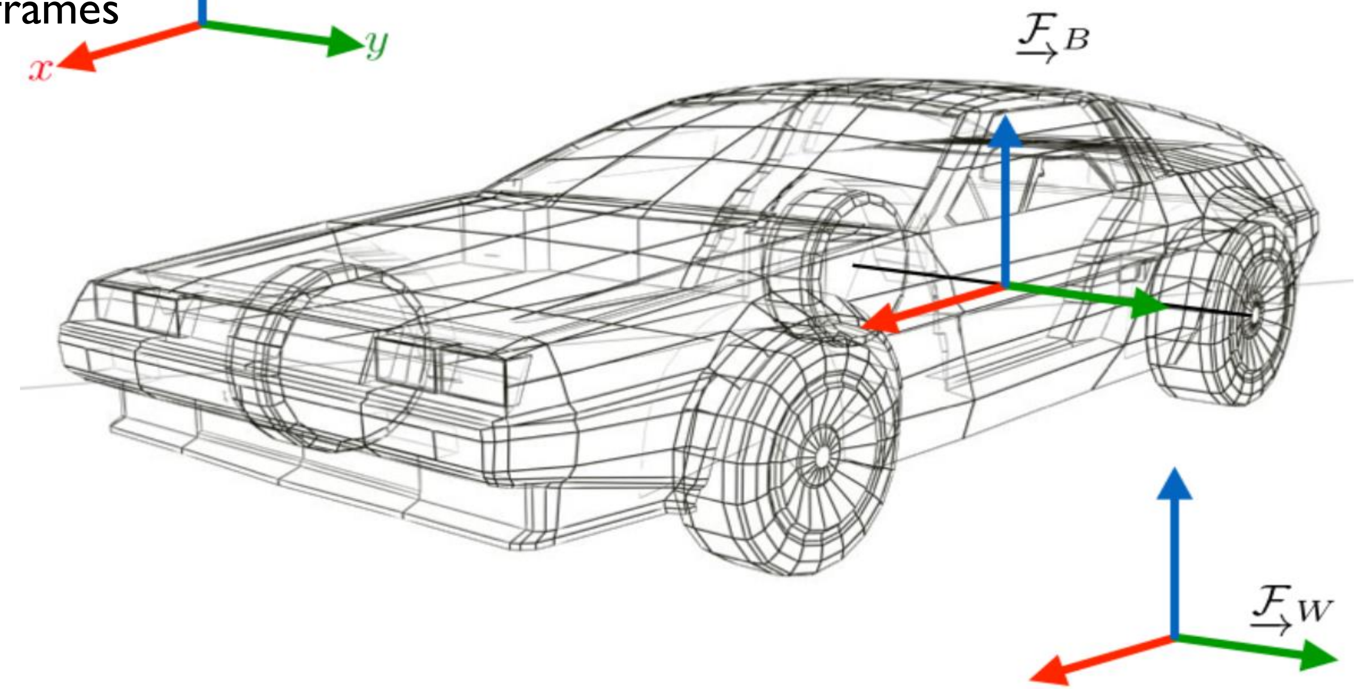
$$\mathbf{R}_z(\alpha) := \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{C}_z(\alpha) := \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Specification ambiguities in Euler Angles

Color convention for frames



Moving body (robot) frame



Fixed world frame

You need to specify whether the rotation rotates from the world frame to the body frame, or the other way around.

Another 2 possibilities. More possibilities if you have more frames.

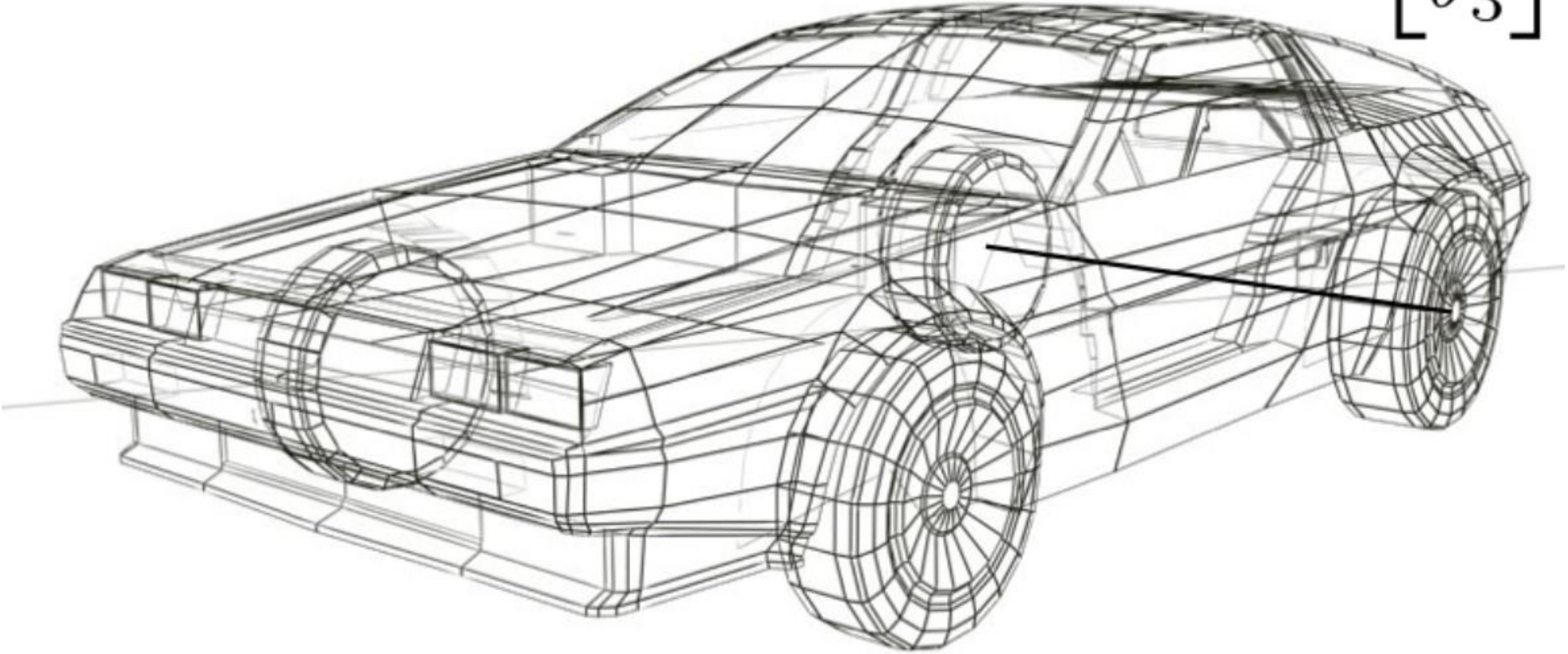
Degrees or radians? Another 2 possibilities

Specification ambiguities in Euler Angles

- Need to specify the ordering of the three parameters.
- 1-2-3, 1-3-2, 2-1-3, 2-3-1, 3-1-2, 3-2-1
- **Another 6 different valid combinations**

- The “three number” problem
- You are given three numbers representing *the orientation of the robot*
- How many possibilities are there?

$$\mathbf{C} \leftarrow \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

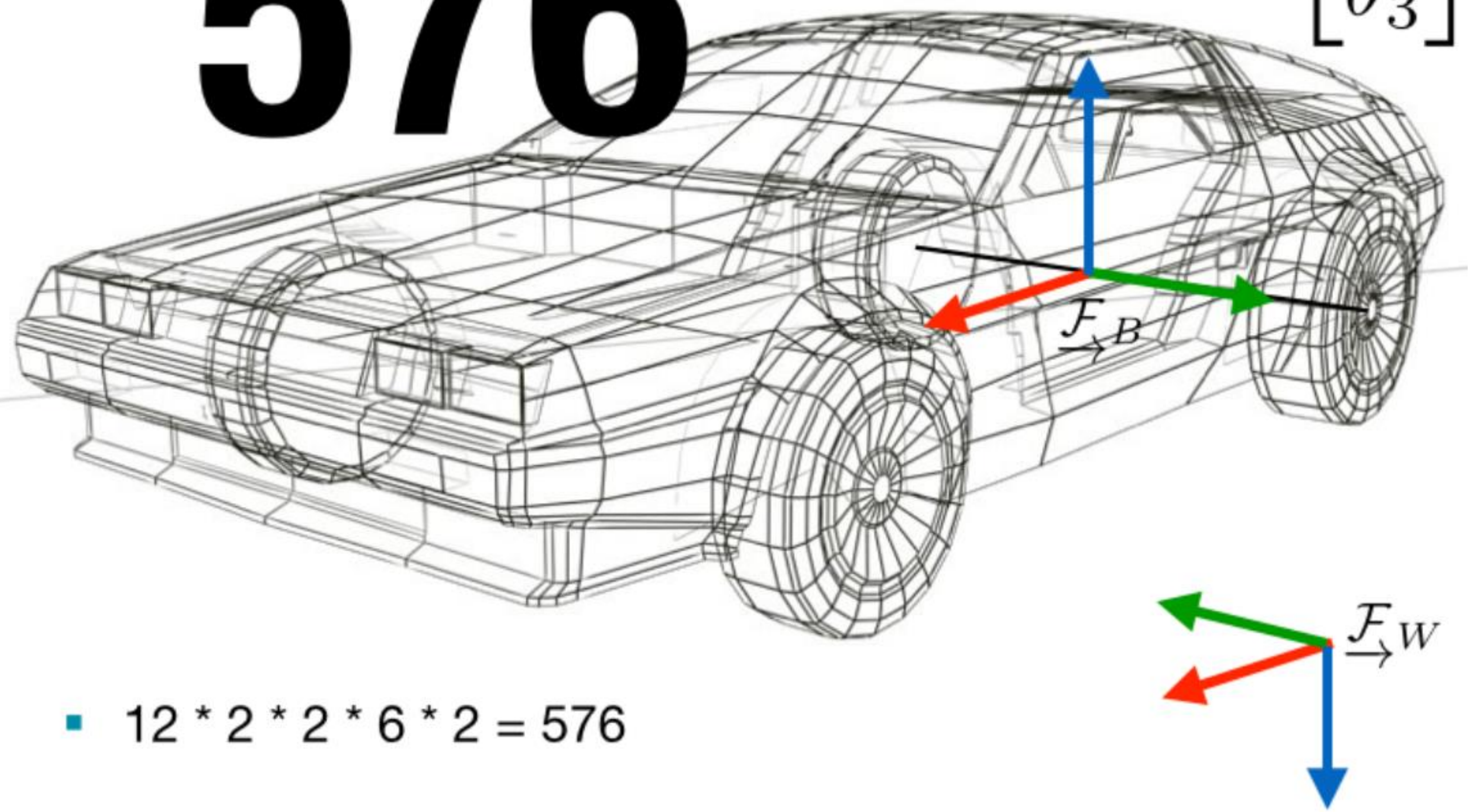


Super sweet wireframe Delorian from http://nanahra.blogspot.com

- The “three number” problem
- How many possibilities are there?

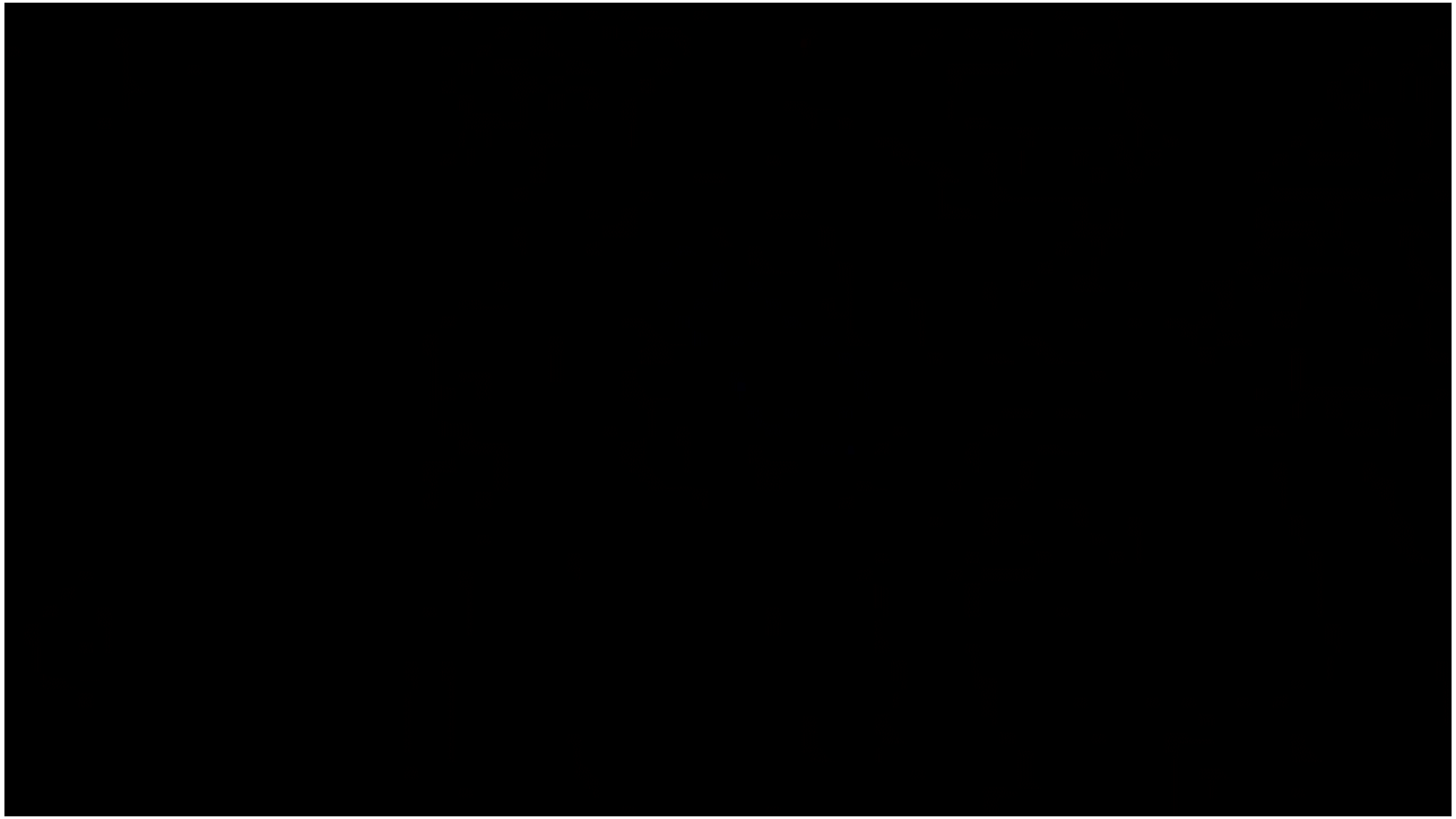
576

$$\mathbf{C} \leftarrow \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$



- 12 * 2 * 2 * 6 * 2 = 576

Another problem with Euler angles: Gimbal Lock



Another problem with Euler angles: Gimbal Lock

- Why should roboticists care about this?

- Because when it happens Euler angle representations lose one degree of freedom.

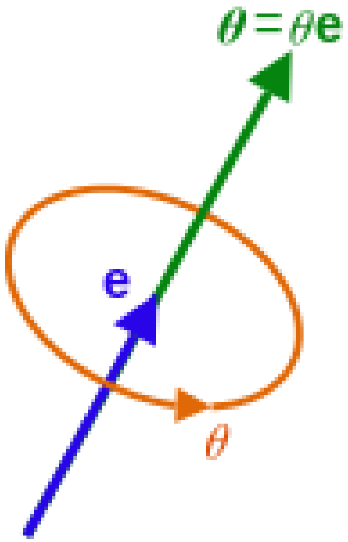
- They cannot represent the entire range of rotations any more.

- They get “locked” into a subset of the space of possible rotations.

So, we need other representations aside from Euler angles.

Even though they are a minimal representation.

Representing Rotations in 3D: Axis-Angle



- 4-number representation (angle, 3D axis)
- 2 ambiguities: (-angle, -axis) is the same as (angle, axis)

Representing Rotations in 3D: Rotation Matrix

- The royalty of rotation representations
- 3x3-number representation, very redundant
- No ambiguities, as long as source frame and target frame are specified correctly. For example, define your notation this way:
- Rotation from Body frame to World frame: $\mathbf{R}_{BW}$
- Or you can define it this way: ${}^{W}_{B}\mathbf{R}$

Inverse Rotation Matrix

$$ {}^{W}_{B}\mathbf{R}^{-1} = {}^{W}_{B}\mathbf{R}^{t} = {}^{B}_{W}\mathbf{R} $$

Rotation matrices are orthogonal matrices: their transpose is their inverse and they do not change the length of a vector, they just rotate it in space.

$$ {}^{W}_{B}\mathbf{R}^{t}\,{}^{W}_{B}\mathbf{R} = \mathbf{I} $$

Converting axis-angle to rotation matrix

- Given angle theta and axis v the equivalent rotation matrix is

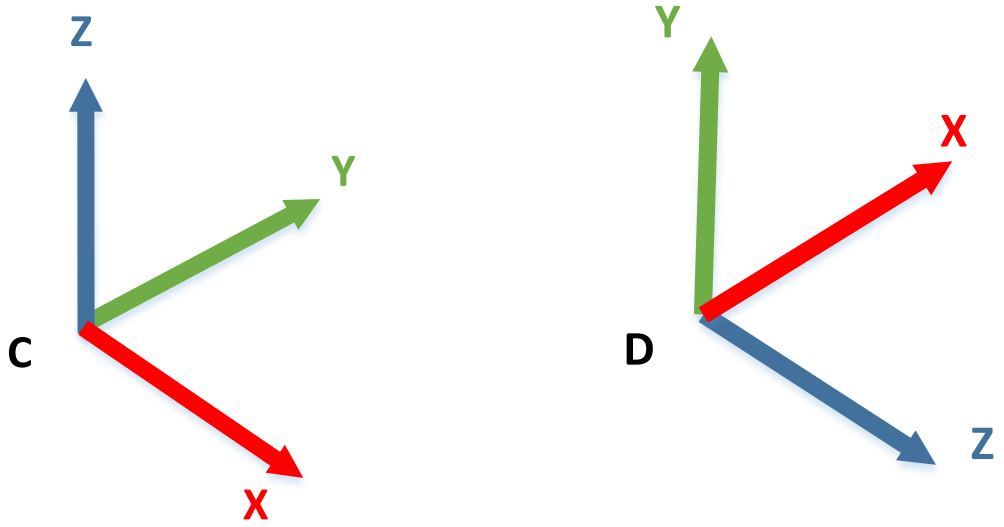
$$\mathbf{R} = \mathbf{I}\cos\theta + (1 - \cos\theta)\mathbf{v}\mathbf{v}^t + [\mathbf{v}]_\times$$

- Where I is the 3x3 identity and

$$[\mathbf{a}]_\times \stackrel{\text{def}}{=} \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}.$$

- This is called the “Rodrigues formula”

Example: finding a rotation matrix that rotates one vector to another



$$ {}^{D}_{C}\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} $$

This matrix transforms the x-axis of frame C to the z-axis of frame D. Same for y and z axes.

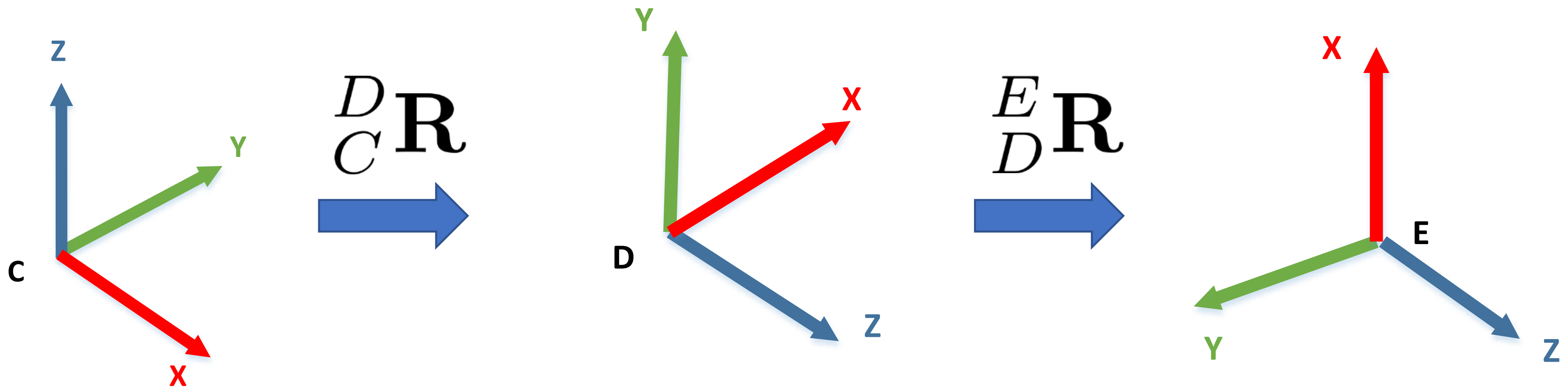
Rotation multiplication vs addition: 3D vs 2D

- In 2D adding angles with wraparound at 360 degrees is a valid operation.

- Rotation matrices can be added, but the result is not necessarily a valid rotation. Rotations are not closed under the operation of addition.

- Rotations are closed under the operation of multiplication. To compose a sequence of simple rotations we need to multiply them.

Compound rotations



$$^{E}_{C}\mathbf{R} = {}^{E}_{D}\mathbf{R}\,{}^{D}_{C}\mathbf{R}$$

Representing Rotations in 3D: Quaternions

- Based on axis-angle representation, but more computationally efficient.

- The main workhorse of rotation representations.

- Used almost everywhere in robotics, aerospace, aviation.

- Very important to master in this course. You will need it for the first assignment and for working with ROS in general.

Converting axis-angle to quaternion

- Given angle theta and axis v the equivalent quaternion representation is

$$\mathbf{q} = [\sin(\theta/2)v_1,\ \sin(\theta/2)v_2,\ \sin(\theta/2)v_3,\ \cos(\theta/2)]$$
$$\mathbf{q} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} + w$$

- Just like in the case of rotation matrices we denote the source and target frames of the rotation quaternion: ${}^{W}_{B}\mathbf{q}$

Converting axis-angle to quaternion

- Given angle theta and a unit axis v, the equivalent quaternion representation is:

$$\mathbf{q} = [\sin(\theta/2)v_1,\ \sin(\theta/2)v_2,\ \sin(\theta/2)v_3,\ \cos(\theta/2)]$$
$$\mathbf{q} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} + w$$

- Just like in the case of rotation matrices we denote the source and target frames of the rotation quaternion: ${}^{W}_{B}\mathbf{q}$

- We always work with unit length (normalized) quaternions.

Examples of quaternions

- 90 degree rotation about the z-axis

$$\mathbf{q} = [0,\ 0,\ \sin(\pi/4)v_3,\ \cos(\pi/4)]$$

Quaternion multiplication

- Defined algebraically by

$$Q = q_0 + q_1 i + q_2 j + q_3 k$$

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

$$ij = k,\ jk = i,\ ki = j$$

and usually denoted by the circular cross symbol. For example:

$${}^{W}_{F}\mathbf{q} = {}^{W}_{C}\mathbf{q} \otimes {}^{C}_{F}\mathbf{q}$$

Quaternion multiplication

$$ {}^{W}_{F}\mathbf{q} = {}^{W}_{C}\mathbf{q} \otimes {}^{C}_{F}\mathbf{q} $$

Direct correspondence with matrix multiplication:

$$ {}^{W}_{F}\mathbf{R}(\mathbf{q}) = {}^{W}_{C}\mathbf{R}(\mathbf{q})\,{}^{C}_{F}\mathbf{R}(\mathbf{q}) $$

NOTE: the quaternion to matrix conversion will not be given here.
It is usually present in all numerical algebra libraries. At the moment we'll take it for granted.

Quaternion inversion

$$\mathbf{q}^{-1} = -x\mathbf{i} - y\mathbf{j} - z\mathbf{k} + w$$

$$[0, 0, 0, 1] = \mathbf{q}^{-1} \otimes \mathbf{q}$$

Direct correspondence with matrix inversion:

$$\mathbf{I} = \mathbf{R}(\mathbf{q}^{-1})\mathbf{R}(\mathbf{q})$$

$$\mathbf{I} = \mathbf{R}(\mathbf{q})^{-1}\mathbf{R}(\mathbf{q})$$

Example: updating orientation based on angular velocity

- If the angular velocity of the Body frame is ${}^{B}\omega$ and the body-to-world rotation at time t is ${}^{W}_{B}\mathbf{q}(t)$

- Then, at time t+dt the new body-to-world rotation will be

$$ {}^{W}_{B(t+dt)}\mathbf{q} = {}^{W}_{B(t)}\mathbf{q} \otimes {}^{B(t)}_{B(t+dt)}\mathbf{q} $$

where ${}^{B(t)}_{B(t+dt)}\mathbf{q}$ has unit axis $\frac{{}^{B}\omega}{||{}^{B}\omega||}$ and angle $||{}^{B}\omega||dt$

Main ambiguities of quaternion representation

- The ones inherited from the axis-angle representation, but also:

  - Even with unit-length quaternions, there are choices
  - Parameter ordering
    - We won't consider arbitrary ordering
    - We do have to decide on scalar first or scalar last

$$Q = w + xi + yj + zk$$

$$\mathbf{q} := \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$ Scalar Last

$$\mathbf{q} := \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix}$$ Scalar First

Be clear about your orientation representation.

Suggested minimum documentation

- Frame diagram.
- Full description of how to build a transformation matrix from the provided scalars and down to the scalar level.
- A clear statement of which transformation matrix it is.

The resulting matrix, $\boldsymbol{T}_{\mathtt{WB}}$, represents the pose of the robot body frame, $\underrightarrow{\mathcal{F}}_{\mathtt{B}}$, with respect to the world frame, $\underrightarrow{\mathcal{F}}_{\mathtt{W}}$, such that a point in the body frame, ${}_{\mathtt{B}}\mathbf{p}$, can be transformed into the world frame by

$$ {}_{\mathtt{W}}\mathbf{p} = \boldsymbol{T}_{\mathtt{WB}}\,{}_{\mathtt{B}}\mathbf{p}. \tag{1} $$

Let's talk about code.

- Code has the same requirements as notation
- Rotation matrices have two frame decorations:
  - to
  - from
- Coordinates of vectors have three decorations:
  - to
  - from
  - expressed in

Let's talk about code.

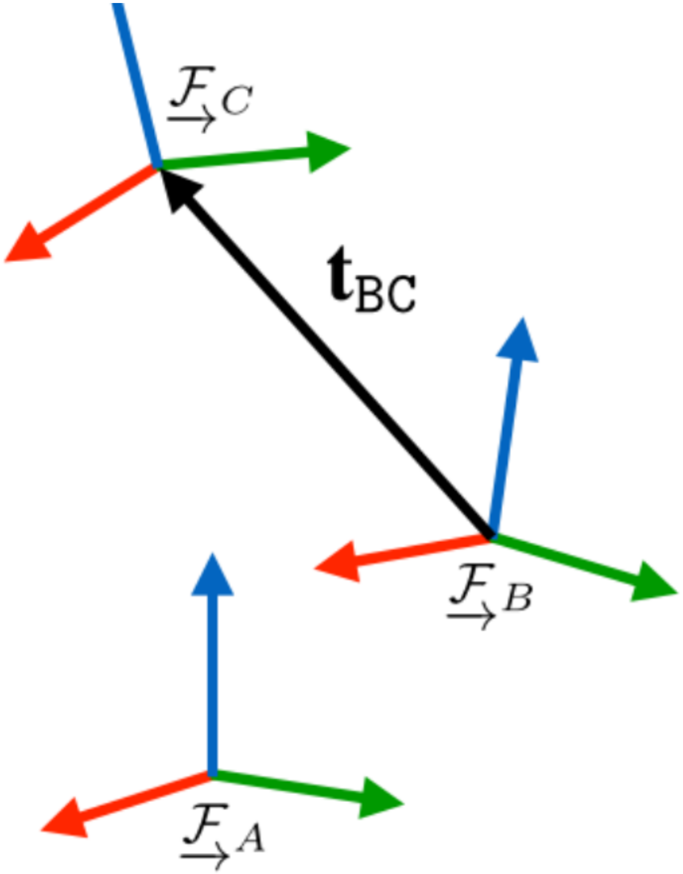
Physical quantity

$${}_{A}\mathbf{t}_{BC}$$

Expressed in

With respect to

of



Let’s talk about code.

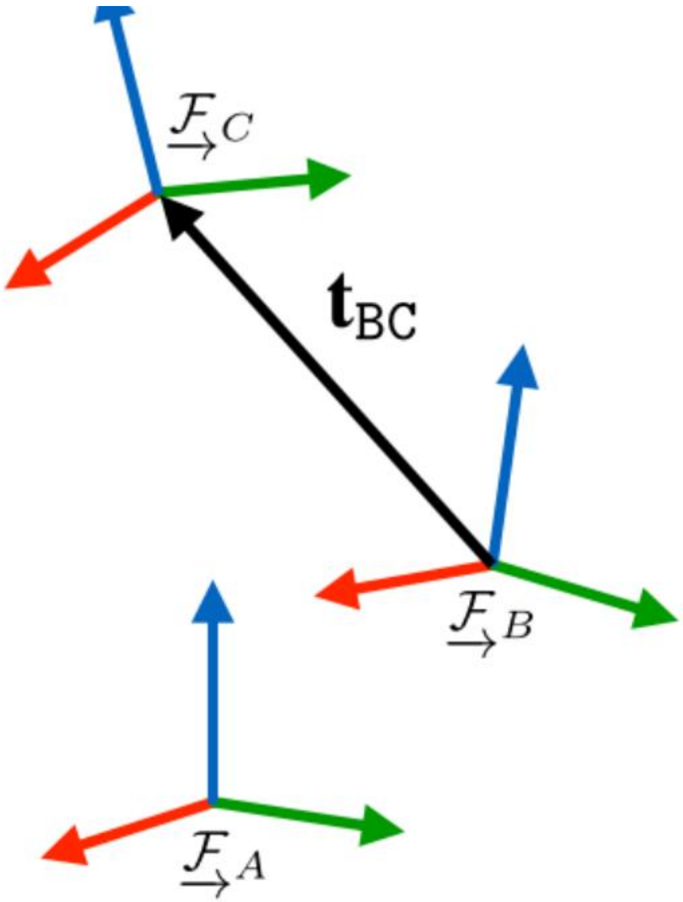
Physical quantity

```
A_t_B_C
```

Expressed in

With respect to

of



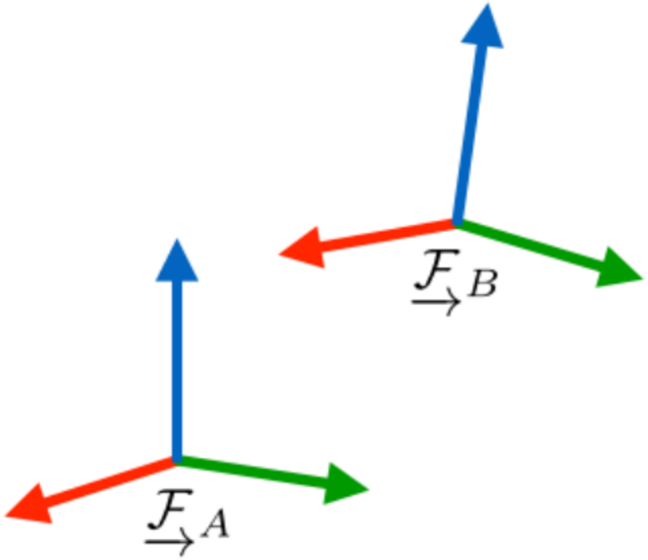
Let’s talk about code.

$$\mathbf{C}_{\mathtt{AB}}$$

into

rotates vectors from

$$_{\mathtt{A}}\mathbf{v}_{\mathtt{BC}} = \mathbf{C}_{\mathtt{AB}}\,{}_{\mathtt{B}}\mathbf{v}_{\mathtt{BC}}$$



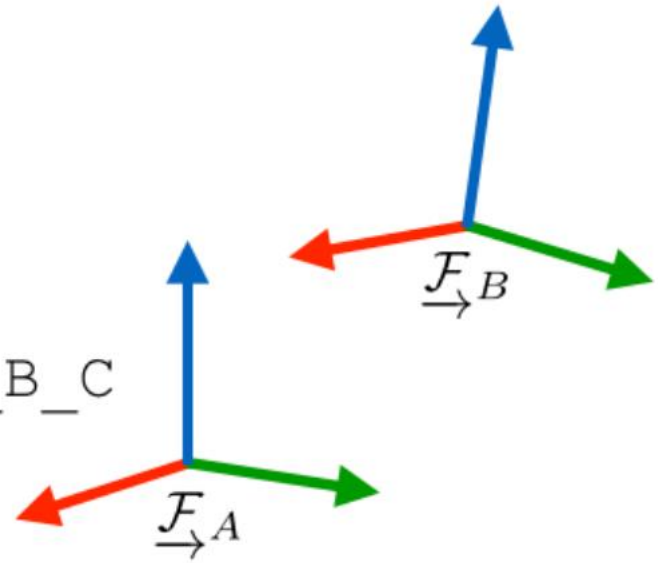
Let’s talk about code.

```
C_A_B
```

into

rotates vectors from

```
A_v_B_C = C_A_B * B_v_B_C
```



Let's talk about code.

- Comments

```
/// Coordinate frames in this function:
///   - C : The camera frame, indexed by time, k.
///   - W : The world frame.
Point pointToCamera( const Transformation& T_W_Ckm1,
                     const Transformation& T_Ckm1_Ck,
                     const Transformation& T_Ck_Ckp1,
                     const Point& W_p ) {

  Transformation T_Ckp1_W = (T_W_Ckm1 * T_Ckm1_Ck * T_Ck_Ckp1).inverse();
  return T_Ckp1_W * W_p

}
```

$\boldsymbol{T}_{\mathtt{W}\mathtt{C}_{\mathtt{k}-1}}$

$\boldsymbol{T}_{\mathtt{C}_{\mathtt{k}-1}\mathtt{C}_{\mathtt{k}}}$

$\boldsymbol{T}_{\mathtt{C}_{\mathtt{k}}\mathtt{C}_{\mathtt{k}+1}}$

${}_{\mathtt{W}}\mathbf{p}$

Let’s talk about code.

Choose an expressive coding style.
Explain it clearly.
Stick with it.

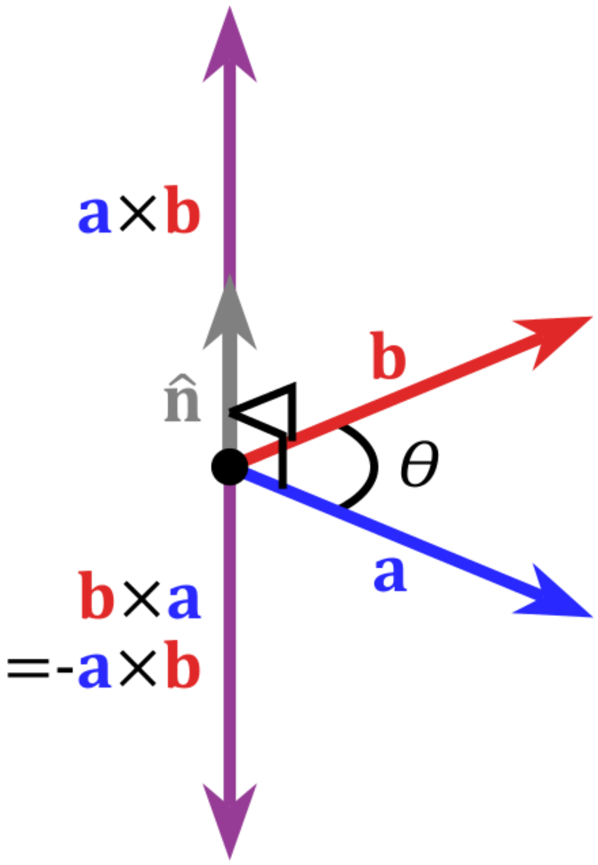
Example: finding quaternion that rotates one vector into another

- Suppose you have a vector in frame A, and a vector in frame B
- You want to find a quaternion that transforms ${}^{A}\mathbf{v}$ to ${}^{B}\mathbf{v}$

- Idea: use axis-angle and convert it to quaternion

- Can rotate from ${}^{A}\mathbf{v}$ to ${}^{B}\mathbf{v}$ along an axis that is perpendicular to both of them. How do we find that?

Cross Product



$$\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \sin(\theta) \; \mathbf{n}$$

Example: finding quaternion that rotates one vector into another

$$\mathbf{v}_{\text{rot axis}} = {}^{A}\mathbf{v} \times {}^{B}\mathbf{v}$$ is perpendicular to both of them

$$\theta_{\text{rot angle}} = \operatorname{acos}({}^{A}\mathbf{v} \cdot {}^{B}\mathbf{v})$$

Assuming the two vectors are unit length

Rotating a vector via a quaternion

- Let ${}^{A}\mathbf{v}$ be given and a quaternion ${}_{A}^{B}\mathbf{q}$
- To obtain ${}^{B}\mathbf{v}$ you have two choices:
- Either use the rotation matrix ${}^{B}\mathbf{v} = {}_{A}^{B}\mathbf{R}(\mathbf{q})\,{}^{A}\mathbf{v}$
- Or use quaternion multiplication directly

$$[{}^{B}\mathbf{v}, 0] = {}_{A}^{B}\mathbf{q} \otimes [{}^{A}\mathbf{v}, 0] \otimes {}_{B}^{A}\mathbf{q}$$

Transforming points from one frame to another

- **VERY IMPORTANT AND USEFUL**

- Suppose you have a point in the Body frame, $^{B}\mathbf{p}$ which you want to transform/express in the World frame. Then you can do any of the two following options:

$$^{W}\mathbf{p} = {}^{W}_{B}\mathbf{R}\,{}^{B}\mathbf{p} + {}^{W}\mathbf{t}_{WB} \qquad\qquad {}^{W}\mathbf{p} = {}^{W}_{B}\mathbf{R}({}^{B}\mathbf{p} - {}^{B}\mathbf{t}_{BW})$$

- Think of it as first rotating the point to be in the World frame and then adding to it the translation from Body to World.

Transforming vectors from one frame to another

- **VERY IMPORTANT AND USEFUL**

- Suppose you have a vector in the Body frame, ${}^{B}\mathbf{v}$ which you want to transform/express in the World frame. Then

$$ {}^{W}\mathbf{v} = {}^{W}_{B}\mathbf{R}\,{}^{B}\mathbf{v} $$

Combining rotations and translation into one transformation

- **VERY IMPORTANT AND USEFUL**

- Many times we combine the rotation and translation of a rigid motion into a 4x4 homogeneous matrix

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

Main advantage of homogeneous transformations: easy composition

$$ {}^{W}_{B}\mathbf{T} = {}^{W}_{A}\mathbf{T}\,{}^{A}_{B}\mathbf{T} $$

Composing rigid motions now becomes a series of matrix multiplications

Inverting a homogeneous transformation

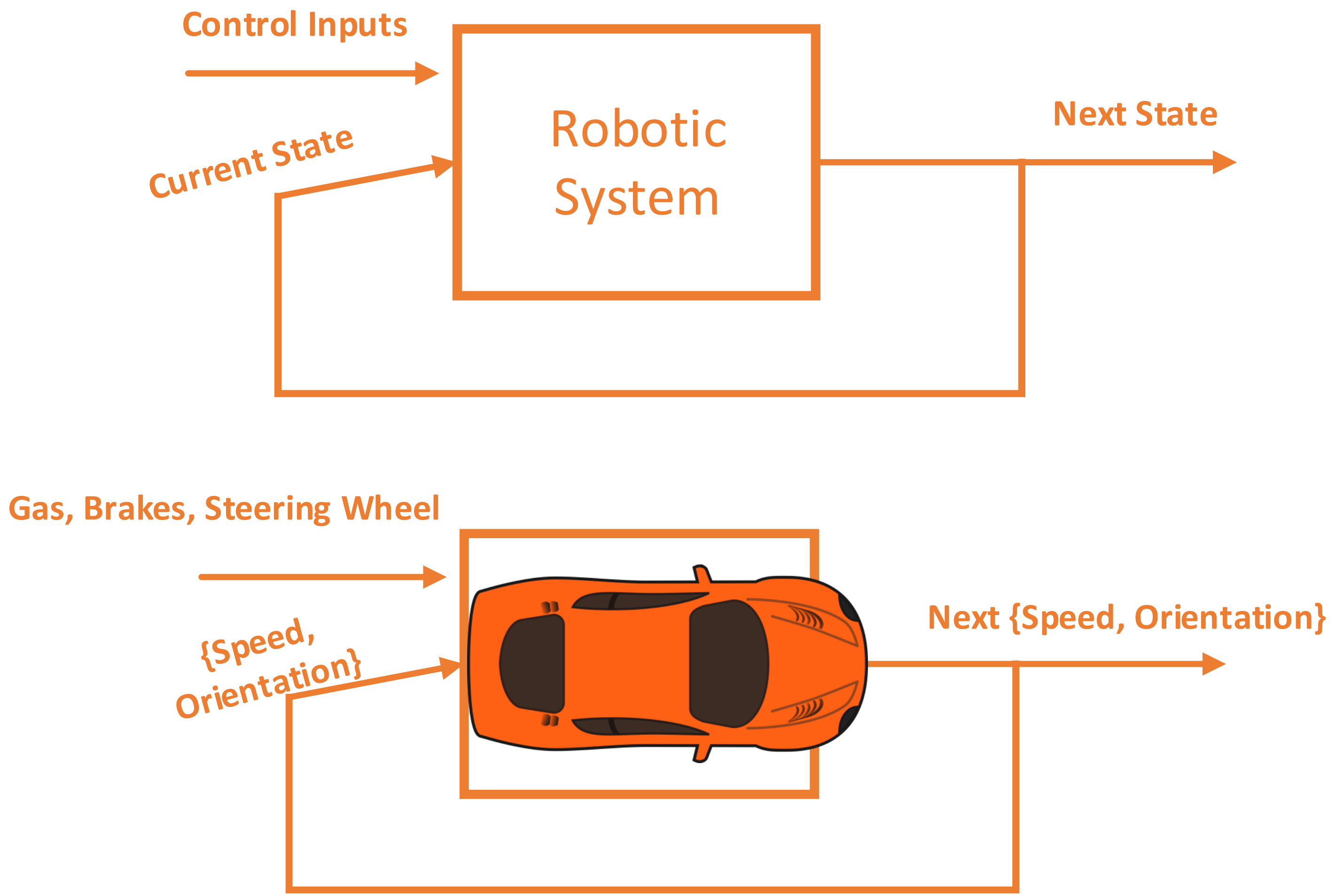
- Be careful:

$$ {}^{W}_{B}\mathbf{T}^{-1} \neq {}^{W}_{A}\mathbf{T}^{t} $$

as was the case with rotation matrices.

Physical models of how systems move

Kinematics & Dynamics:
physical models of
robotic systems and sensors



Main question: what is the next state given the current state and controls?

Today's Agenda

- Frames of reference
- Ways to represent rotations
- Simplified models of vehicles
- Forward and inverse kinematics

Why simplified?

- “All models are wrong, but some are useful” – George Box (statistician)
- Model: a function that describes a physical phenomenon or a system, i.e. how a set of input variables cause a set of output variables.
- Models are useful if they can predict reality <u>up to some degree</u>.
- Mismatch between model prediction and reality = **error / noise**

Noise

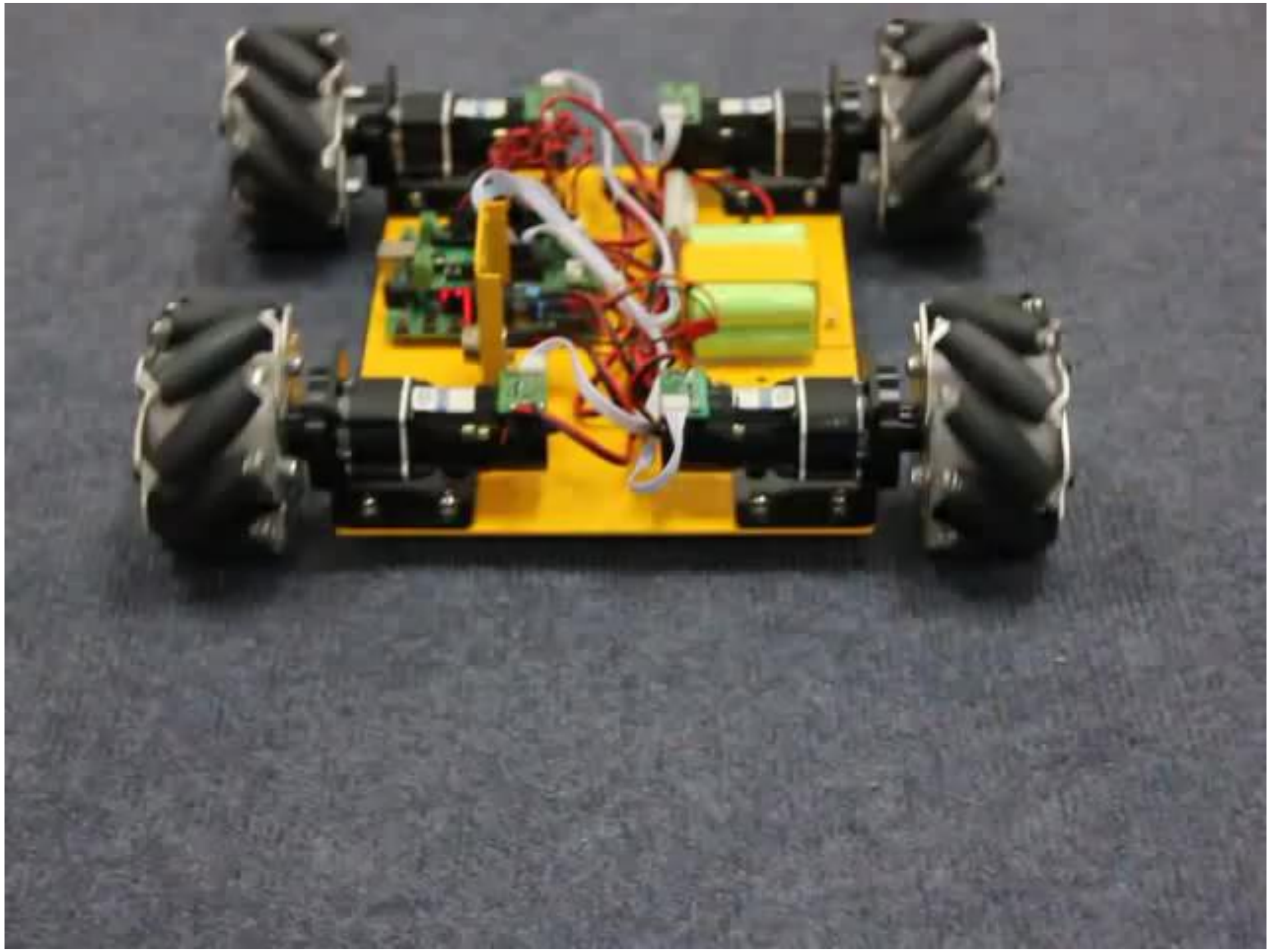
- Anything that we do not bother modelling with our model
- Example 1: “assume frictionless surface”
- Example 2: Taylor series expansion (only first few terms are dominant)
- With models, can be thought of as approximation error.

Simplified physical models of robotic vehicles

- Omnidirectional motion
- Dubins car
- Differential drive steering
- Ackerman steering
- Unicycle
- Cartpole
- Quadcopter

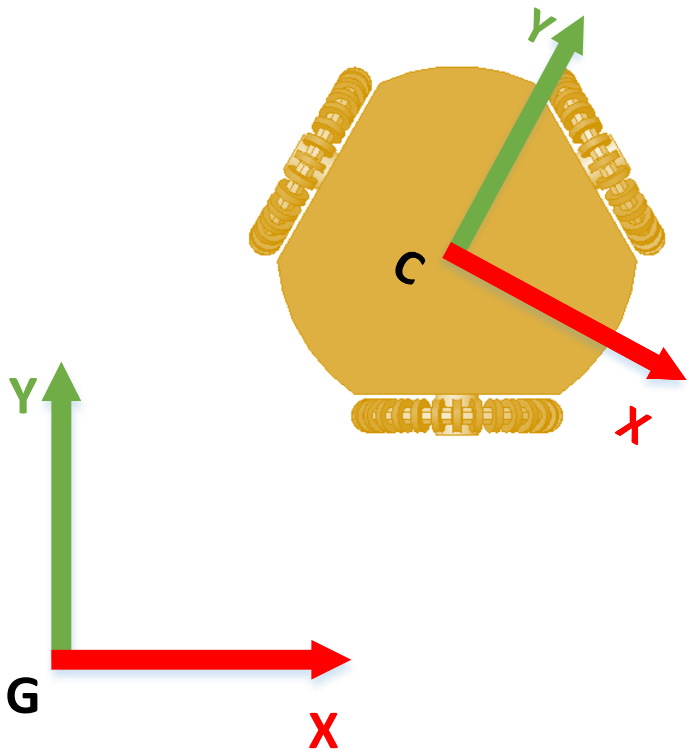
Omnidirectional Robots

Omnidirectional Robots



The state of an omnidirectional robot

State := Configuration := $\mathbf{x}$: = vector of physical quantities of interest about the system

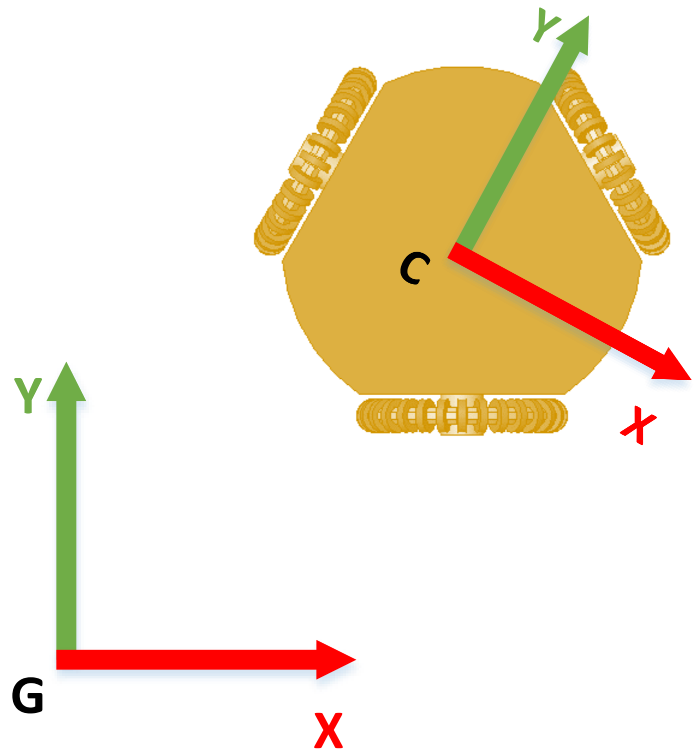


$$\mathbf{x} = [{}^{G}p_x, {}^{G}p_y, {}^{G}\theta]$$

State = [Position, Orientation]
Position of the robot's frame of reference C with respect to a fixed frame of reference G, expressed in coordinates of frame G. Angle is the orientation of frame C with respect to frame G.

Control of an omnidirectional robot

Control := $\mathbf{u}$:= a vector of input commands that can modify the state of the system

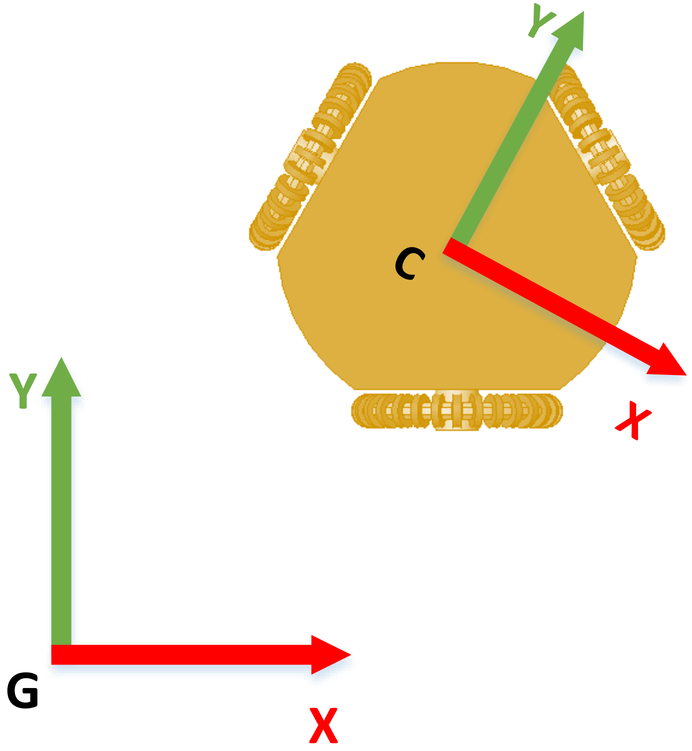


$$\mathbf{u} = [{}^{C}v_x, {}^{C}v_y, {}^{C}\omega_z]$$

Control = [Linear velocity, Angular velocity]
Linear and angular velocity of the robot's frame of reference C with respect to a fixed frame of reference G, expressed in coordinates of frame C.

Dynamics of an omnidirectional robot

Dynamical System : = Dynamics := a function that describes the time evolution of the state in response to a control signal

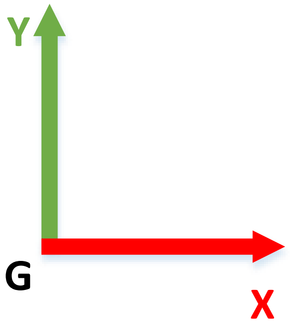


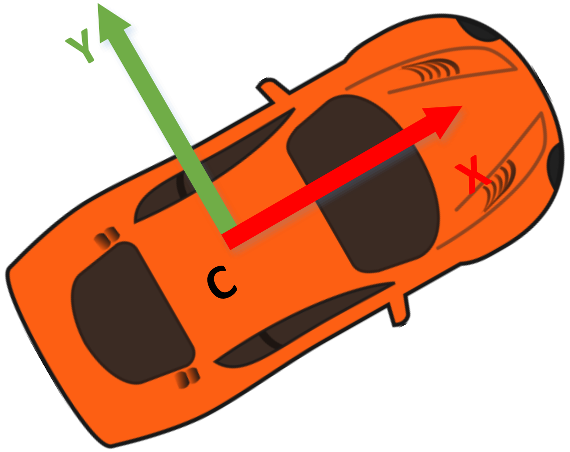
Continuous case: $$\frac{\mathbf{dx}}{\mathbf{dt}} = \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

$$\begin{aligned} \dot{p}_x &= v_x \\ \dot{p}_y &= v_y \\ \dot{\theta} &= \omega_z \end{aligned}$$

Note: reference frames have been removed for readability.

The state of a simple car

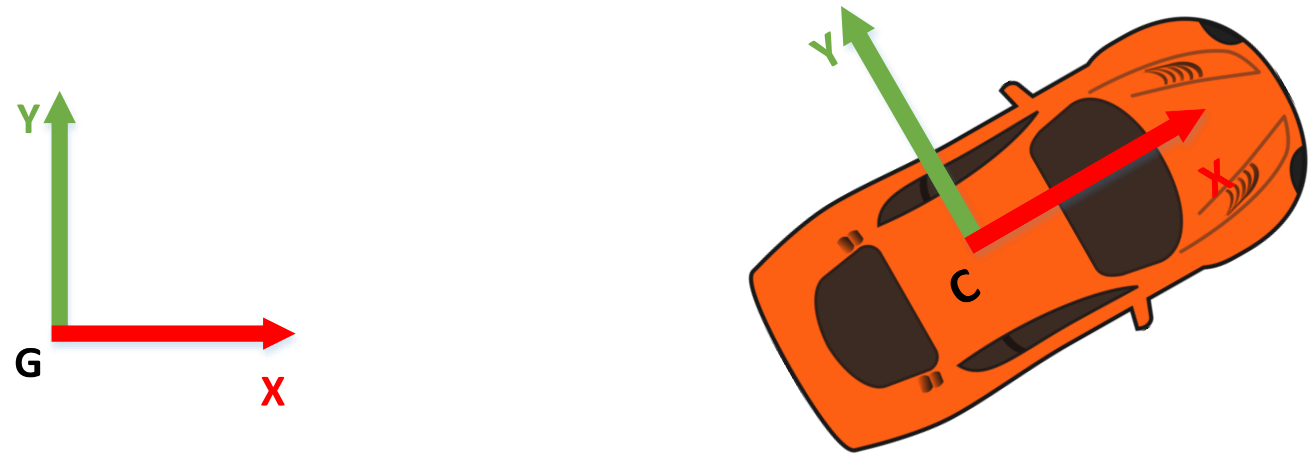




State = [Position and orientation]
Position of the car's frame of reference C with respect to a fixed frame of reference G, expressed in frame G.
The angle is the orientation of frame C with respect to G.

$$\mathbf{x} = [{}^{G}p_x, {}^{G}p_y, {}^{G}\theta]$$

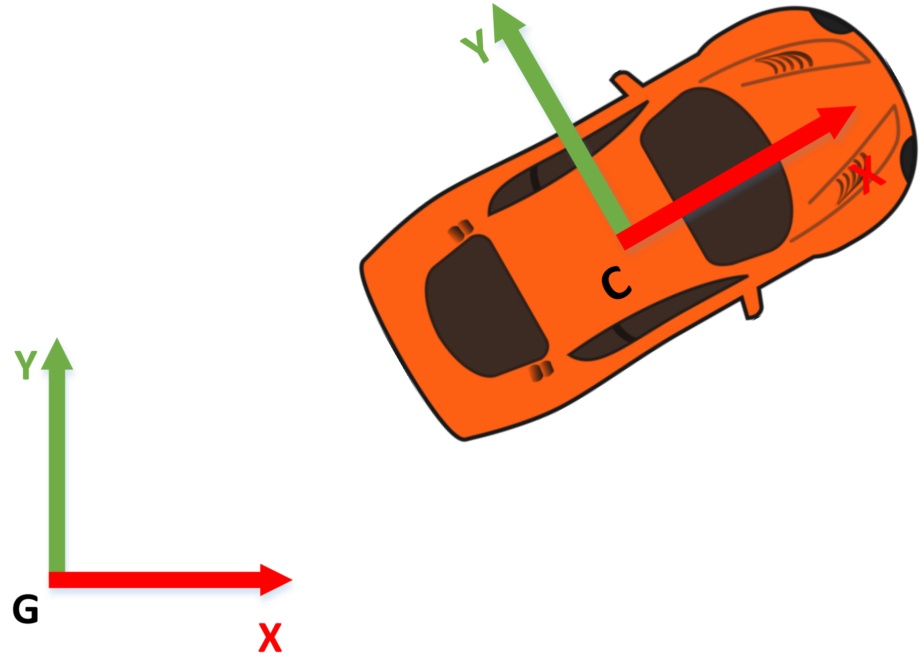
The controls of a simple car



Controls = [Forward speed and angular velocity]
Linear velocity and angular velocity of the car's frame of reference C with respect to a fixed frame of reference G, expressed in coordinates of C.

$$\mathbf{u} = [{}^{C}v_x, {}^{C}\omega_z]$$

The dynamical system of a simple car



$$\begin{aligned} \dot{p}_x &= v_x \cos(\theta) \\ \dot{p}_y &= v_x \sin(\theta) \\ \dot{\theta} &= \omega_z \end{aligned}$$

Note: reference frames have been removed for readability.

Kinematics vs Dynamics

- Kinematics considers models of locomotion independently of external forces and control.

- For example, it describes how the speed of a car affects the state without considering what the required control commands required to generate those speeds are.

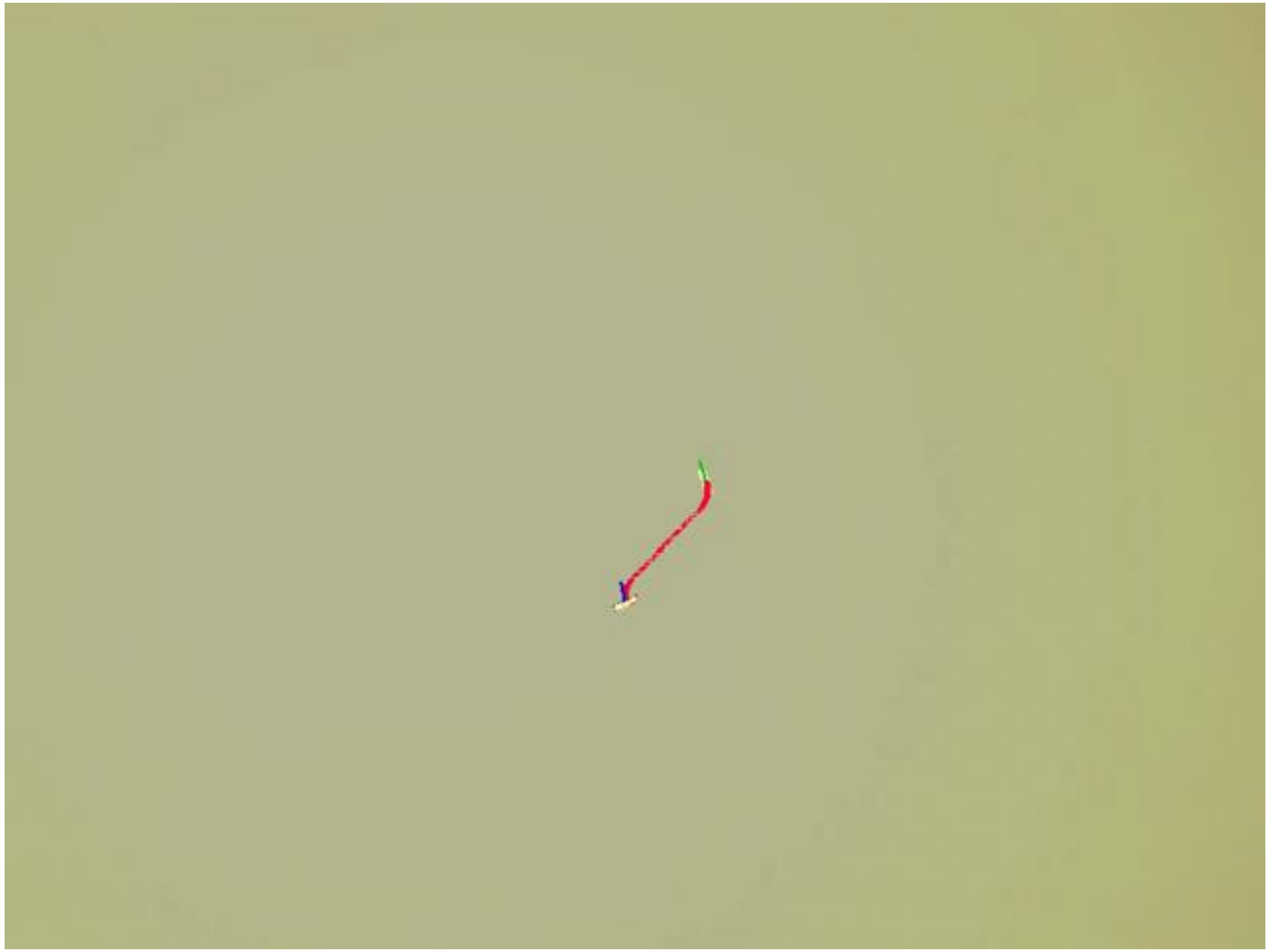
- Dynamics considers models of locomotion as functions of their control inputs and state.

Special case of simple car: Dubins car

- Can only go forward
- Constant speed

$$^{C}v_x = \text{const} > 0$$

- You only control the angular velocity

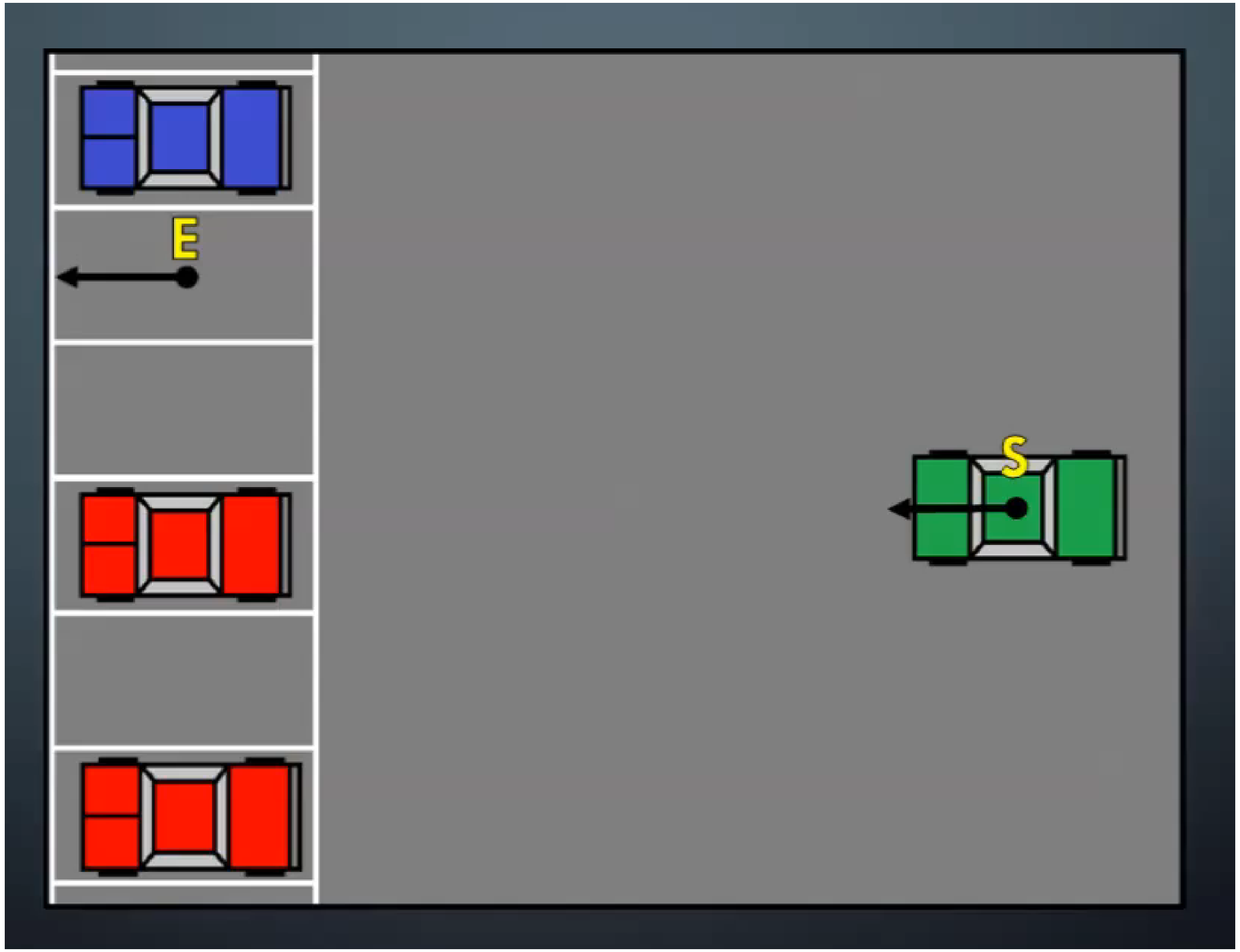


Special case of simple car: Dubins car

- Can only go forward
- Constant speed

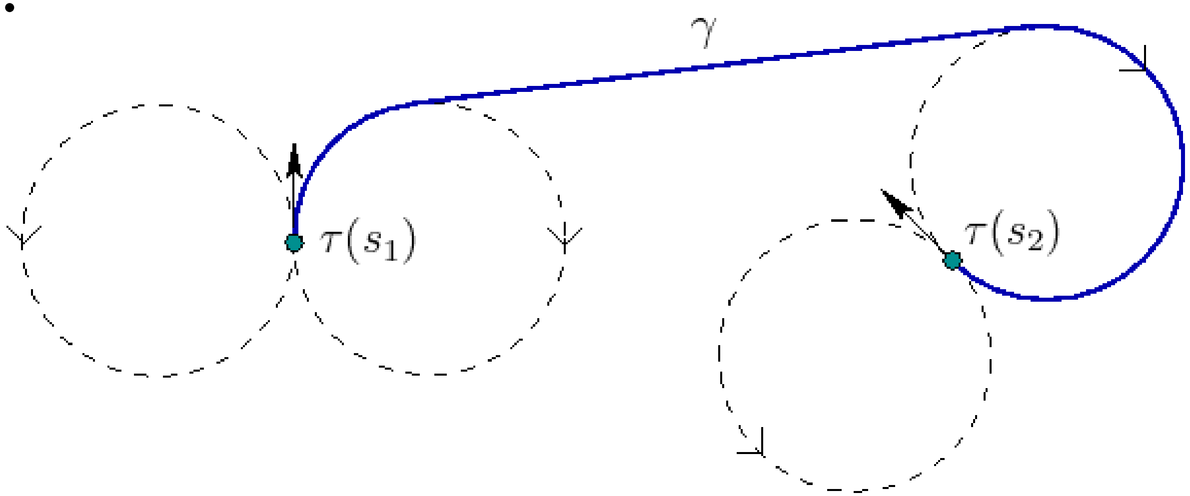
$$^{C}v_x = \text{const} > 0$$

- You only control the angular velocity



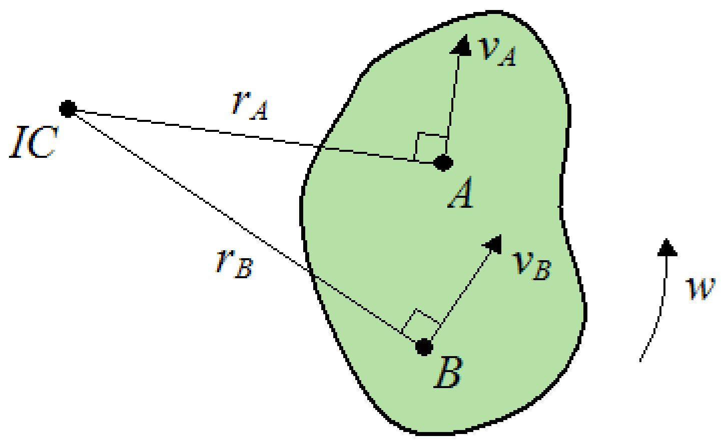
Dubins car: motion primitives

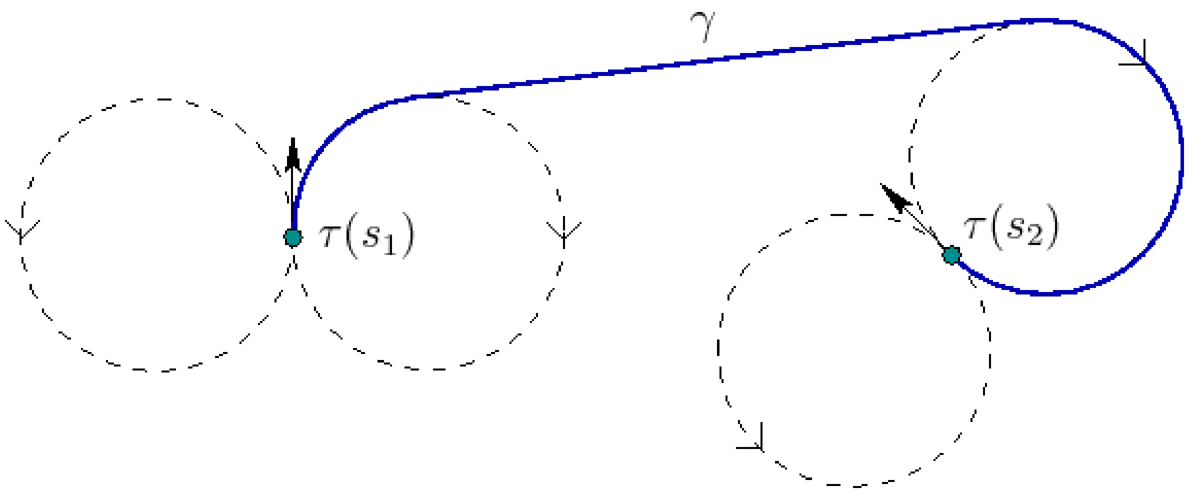
- The path of the car can be decomposed to L(eft), R(ight), S(traight) segments.



RSR path

Instantaneous Center of Rotation

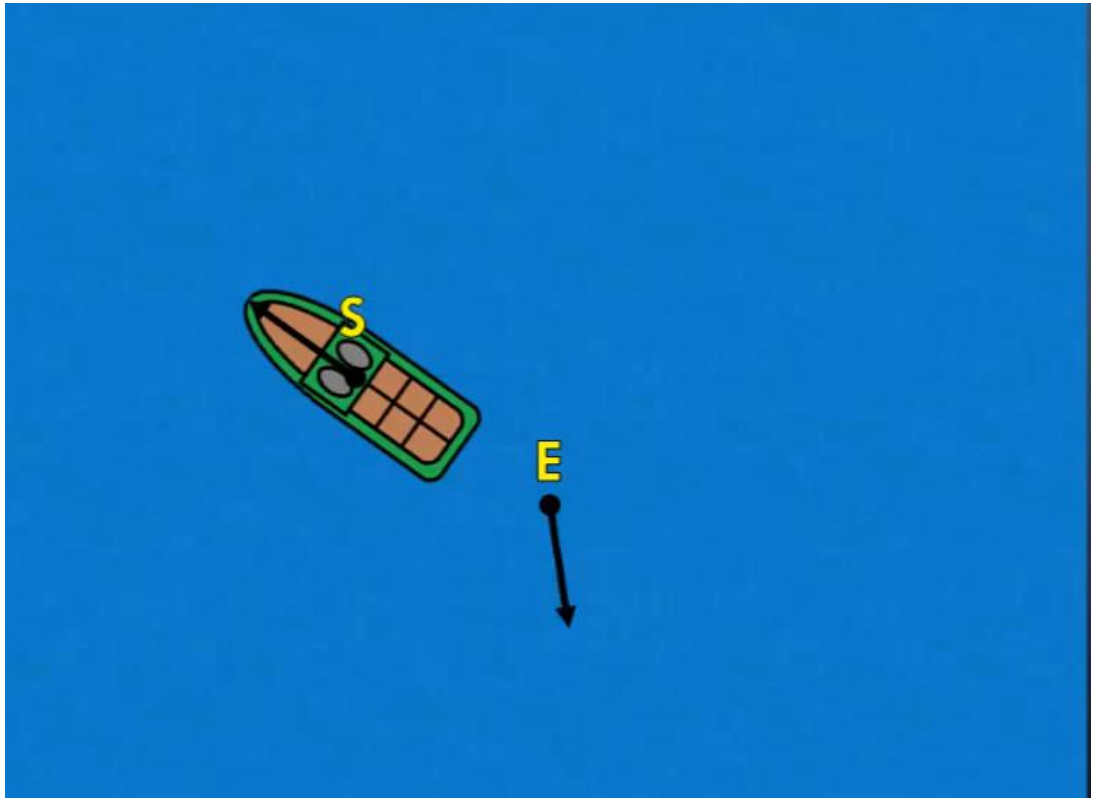




IC = Instantaneous Center of Rotation
The center of the circle circumscribed by the turning path.
Undefined for straight path segments.

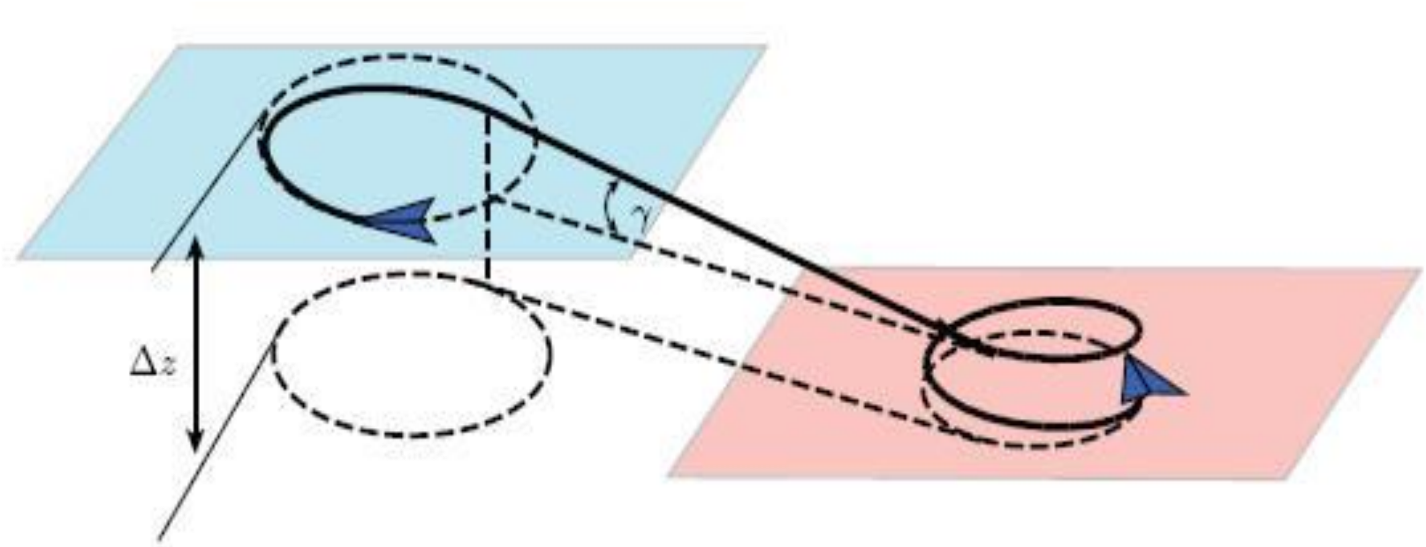
Dubins car → Dubins boat

- Why do we care about a car that can only go forward?
- Because we can also model idealized airplanes and boats
- Dubins boat = Dubins car



Dubins car → Dubins airplane in 3D

- Pitch angle ϕ and forward velocity determine descent rate
- Yaw angle θ and forward velocity determine turning rate



(a) The 3D view of the path.

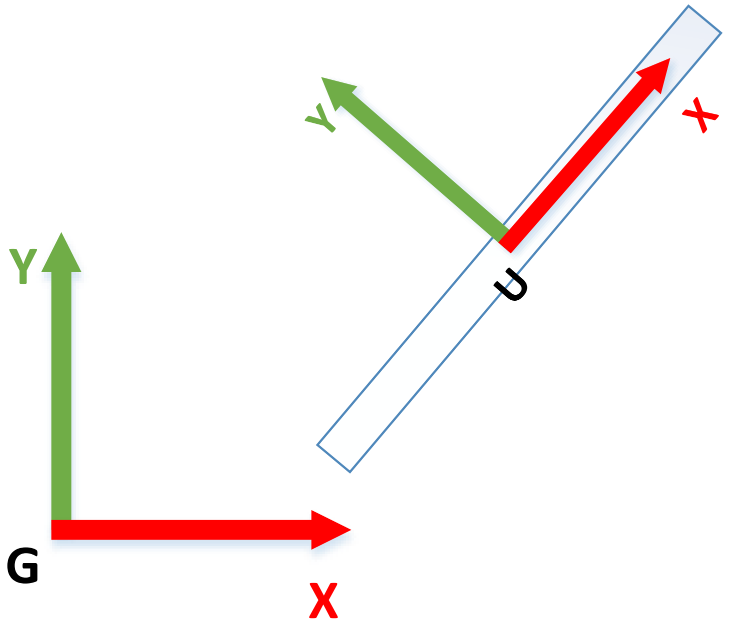
$$
\begin{aligned}
\dot{p}_x &= v_x \cos(\theta)\sin(\phi) \\
\dot{p}_y &= v_x \sin(\theta)\sin(\phi) \\
\dot{p}_z &= v_x \cos(\phi) \\
\dot{\theta} &= \omega_z \\
\dot{\phi} &= \omega_y
\end{aligned}
$$

θ is yaw

ϕ is pitch

The state of a unicycle

$$\mathbf{x} = [{}^{G}p_x, {}^{G}p_y, {}^{G}\theta]$$



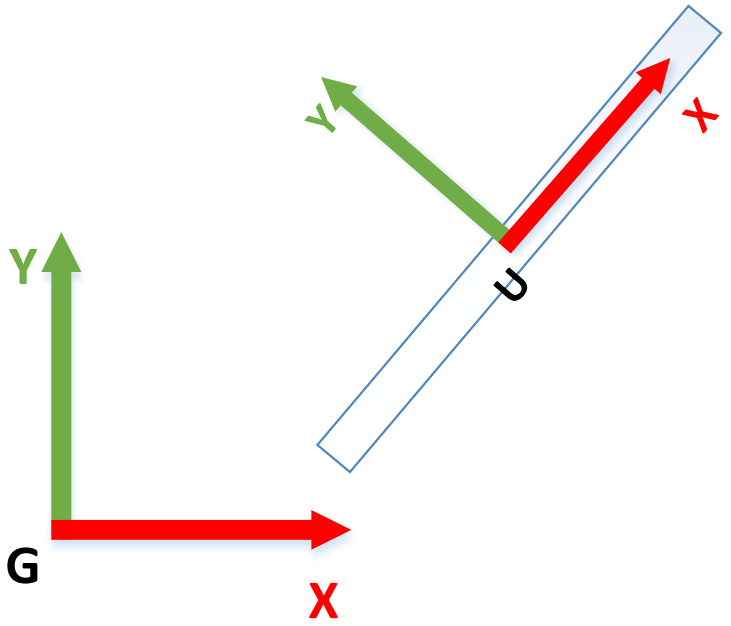
Top view of a unicycle

State = [Position, Orientation]
Position of the unicycle's frame of reference U with respect to a fixed frame of reference G, expressed in coordinates of frame G. Angle is the orientation of frame U with respect to frame G.

Q: Would you put the radius of the unicycle to be part of the state?

The state of a unicycle

$$\mathbf{x} = [{}^{G}p_x, {}^{G}p_y, {}^{G}\theta]$$



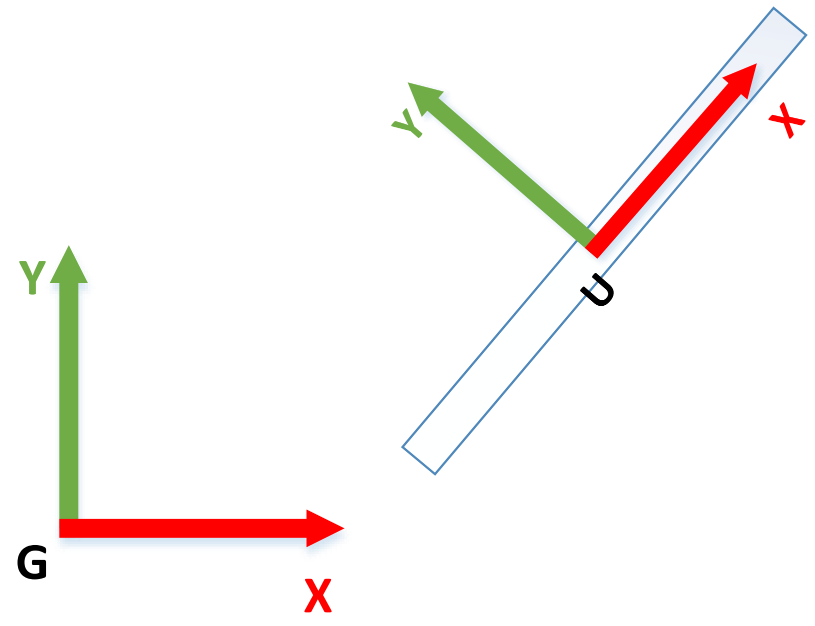
Top view of a unicycle

State = [Position, Orientation]
Position of the unicycle's frame of reference U with respect to a fixed frame of reference G, expressed in coordinates of frame G. Angle is the orientation of frame U with respect to frame G.

Q: Would you put the radius of the unicycle to be part of the state?
A: Most likely not, because it is a constant quantity that we can measure beforehand. But, if we couldn't measure it, we need to make it part of the state in order to estimate it.

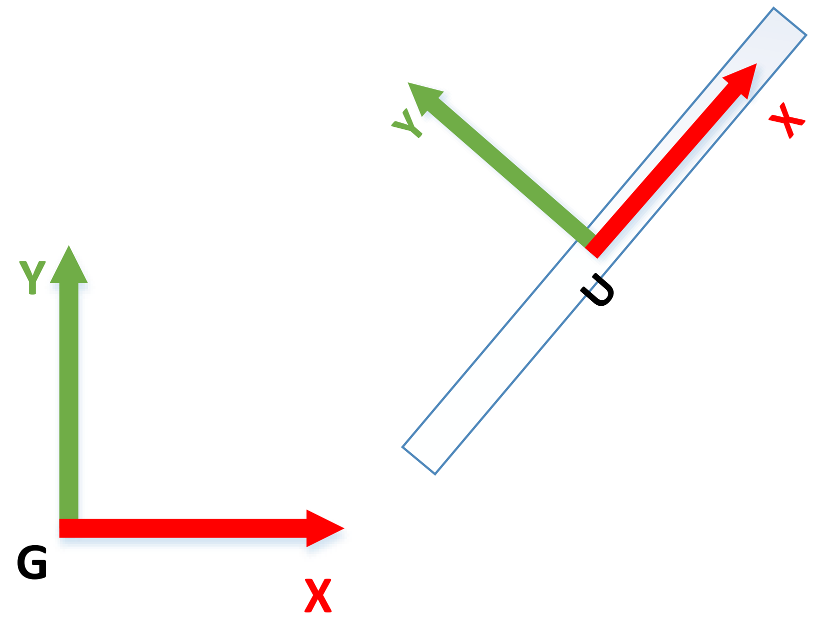
Controls of a unicycle

$$\mathbf{u} = [{}^{U}\omega_z, {}^{U}\omega_y]$$



Controls = [Yaw rate, and pedaling rate]
Yaw and pedaling rates describe the angular velocities of the respective axes of the unicycle's frame of reference U with respect to a fixed frame of reference G, expressed in coordinates of U.

Dynamics of a unicycle



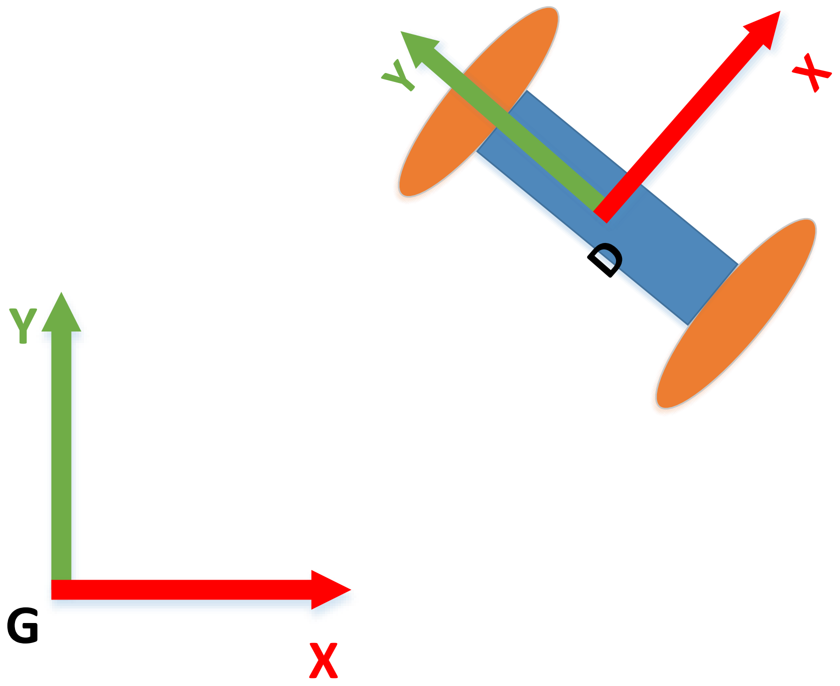
$$\begin{aligned} \dot{p}_x &= r\omega_y \cos(\theta) \\ \dot{p}_y &= r\omega_y \sin(\theta) \\ \dot{\theta} &= \omega_z \end{aligned}$$

r = the radius of the wheel

$r\omega_y$ is the forward velocity of the unicycle

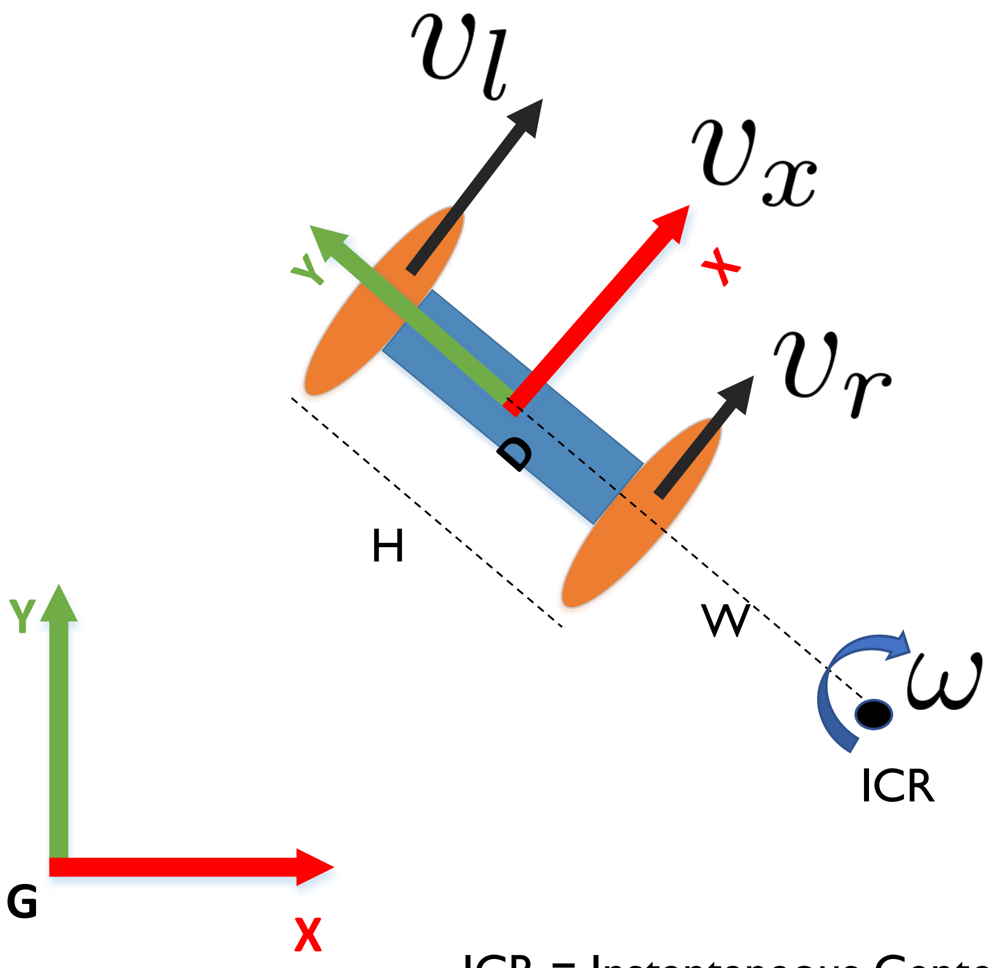
The state of a differential drive vehicle

$$\mathbf{x} = [{}^{G}p_x, {}^{G}p_y, {}^{G}\theta]$$



State = [Position, Orientation]
Position of the vehicle's frame of reference D with respect to a fixed frame of reference G, expressed in coordinates of frame G. Angle is the orientation of frame D with respect to frame G.

Controls of a differential drive vehicle



$$\mathbf{u} = [u_l, u_r]$$

Controls = [Left wheel and right wheel turning rates]
Wheel turning rates determine the linear velocities of the respective wheels of the vehicle's frame of reference D with respect to a fixed frame of reference G, expressed in coordinates of D.

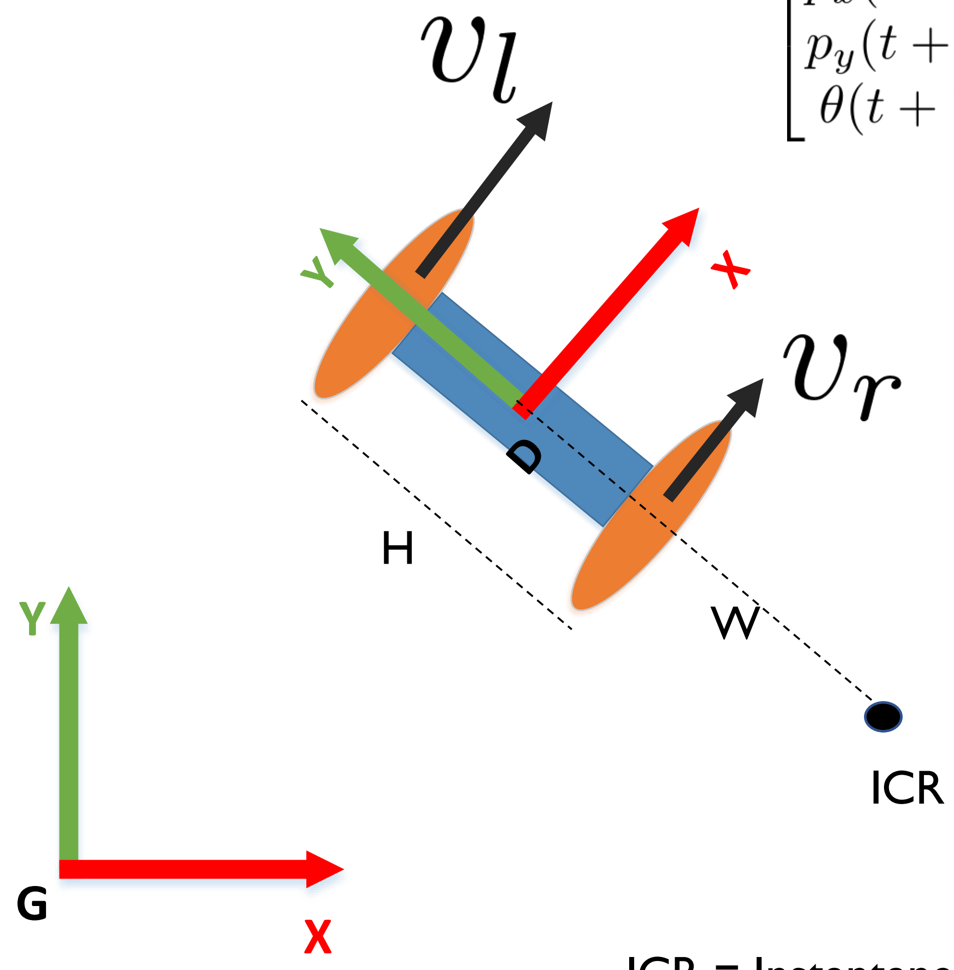
$$v_l = (W - H/2)\omega$$
$$v_r = (W + H/2)\omega$$
$$v_x = (v_l + v_r)/2$$

ICR = Instantaneous Center of Rotation

$$v_l = Ru_l$$
$$v_r = Ru_r$$

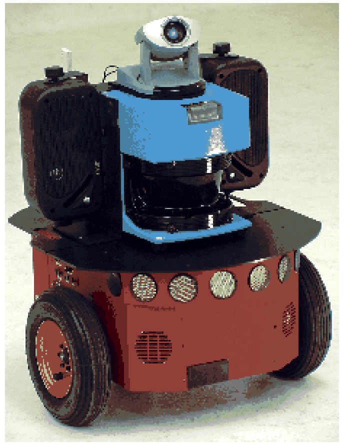
R is the wheel radius

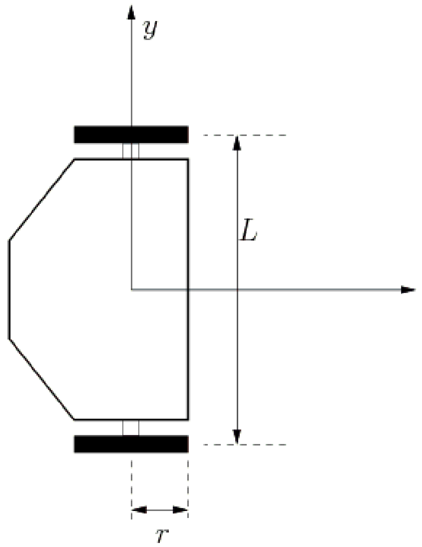
Dynamics of a differential drive vehicle



$$\begin{bmatrix} p_x(t+1) \\ p_y(t+1) \\ \theta(t+1) \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x(t) - \mathrm{ICR}_x \\ p_y(t) - \mathrm{ICR}_y \\ \theta(t) \end{bmatrix} + \begin{bmatrix} \mathrm{ICR}_x \\ \mathrm{ICR}_y \\ \omega\delta t \end{bmatrix}$$

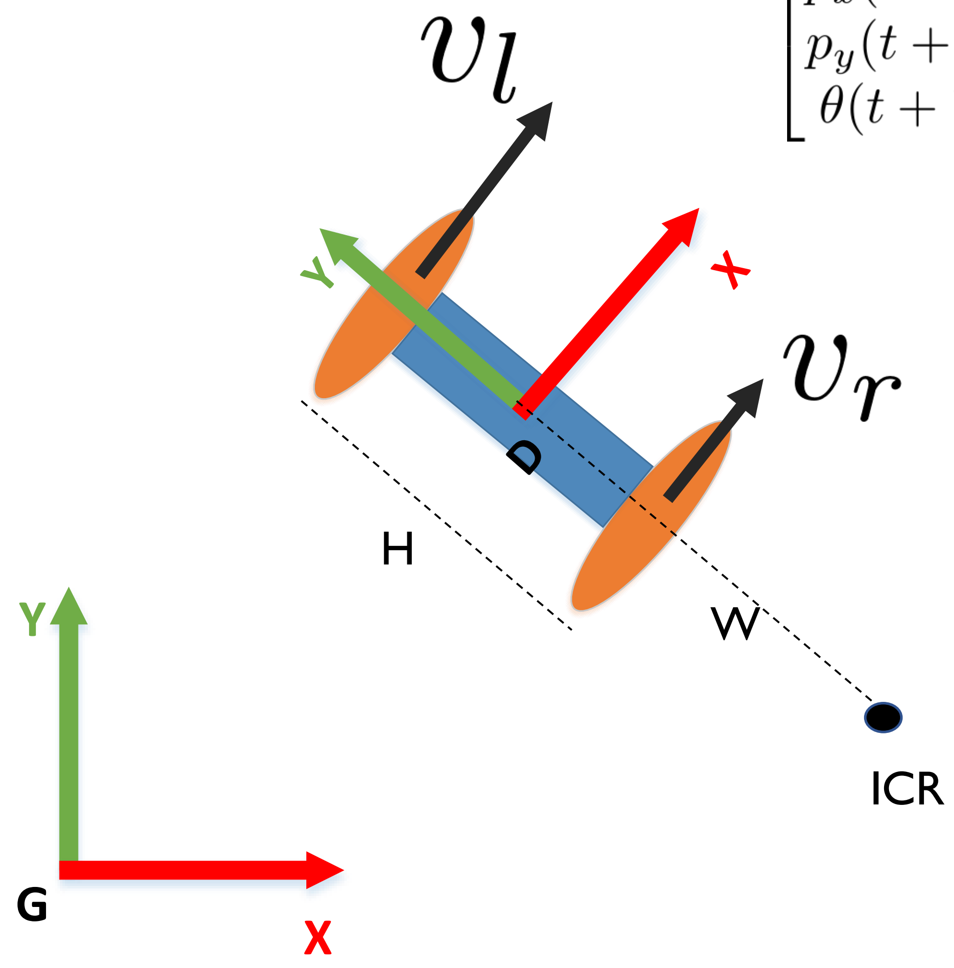
$$\mathrm{ICR} = [p_x - W\sin\theta, p_y + W\cos\theta]$$





ICR = Instantaneous Center of Rotation

Dynamics of a differential drive vehicle



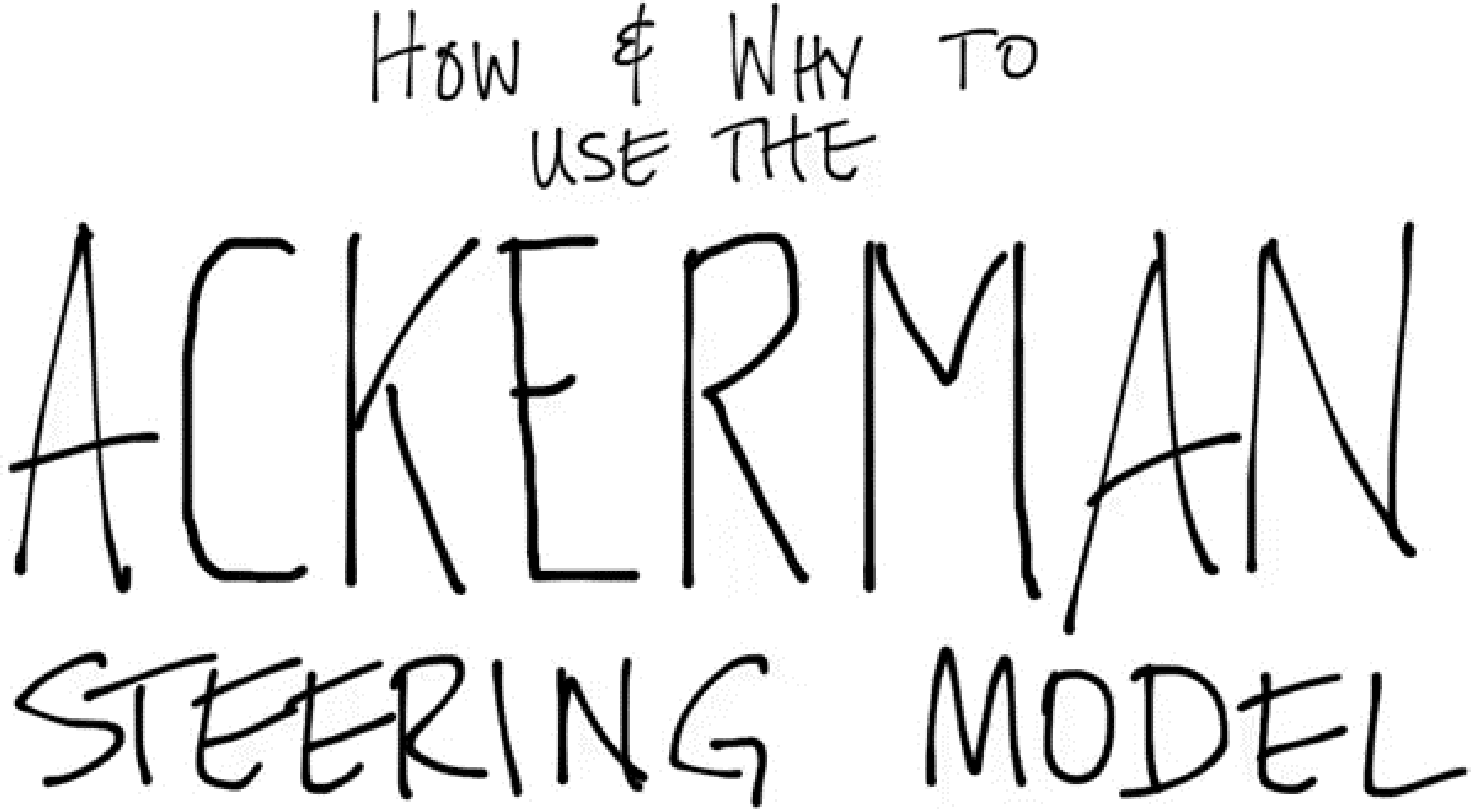
$$\begin{bmatrix} p_x(t+1) \\ p_y(t+1) \\ \theta(t+1) \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x(t) - \mathrm{ICR}_x \\ p_y(t) - \mathrm{ICR}_y \\ \theta(t) \end{bmatrix} + \begin{bmatrix} \mathrm{ICR}_x \\ \mathrm{ICR}_y \\ \omega\delta t \end{bmatrix}$$

$$\mathrm{ICR} = [p_x - W\sin\theta, p_y + W\cos\theta]$$

Special cases:

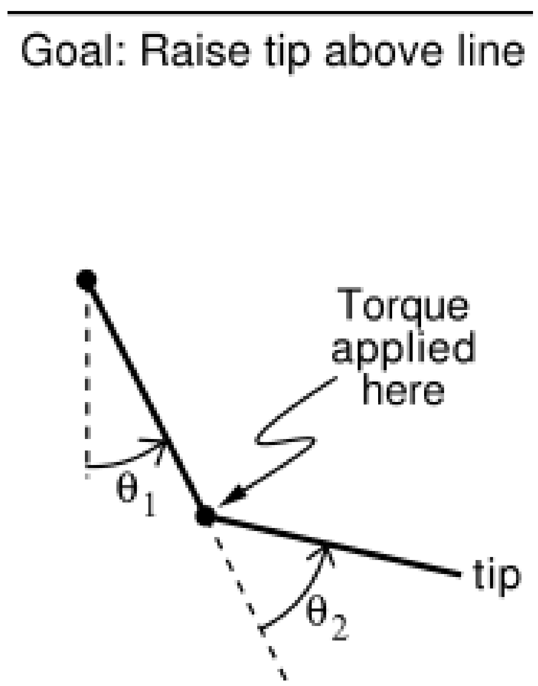
- moving straight $v_l = v_r$
- in-place rotation $v_l = -v_r$
- rotation about the left wheel $v_l = 0$

Ackerman steering



https://www.youtube.com/watch?v=i6uBwudwA5o

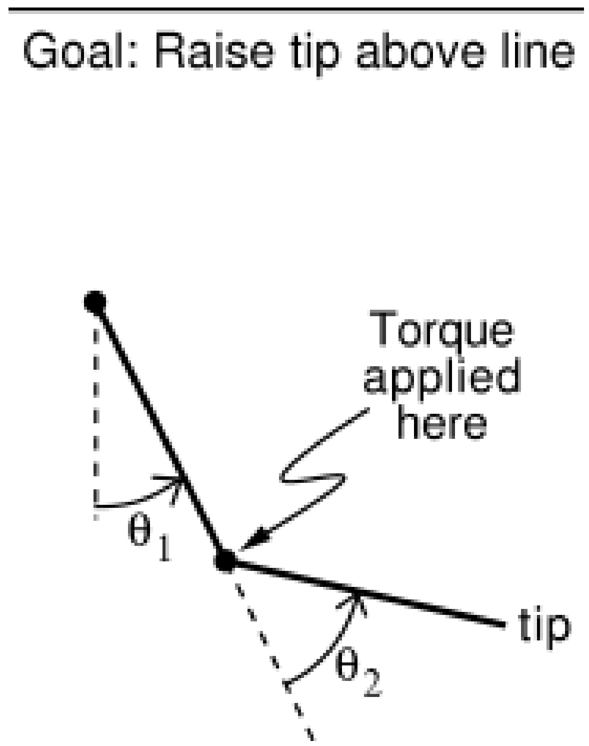
The state of a double-link inverted pendulum (a.k.a. Acrobot)



$$\mathbf{x} = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$$

State = [angle of joint 1, joint 2, joint velocities]
Angle of joint 2 is expressed with respect to joint 1. Angle of joint 1 is expressed compared to down vector.

Controls of a double-link inverted pendulum (a.k.a. Acrobot)



$$\mathbf{u} = [\tau_1]$$

Controls = [torque applied to joint 1]

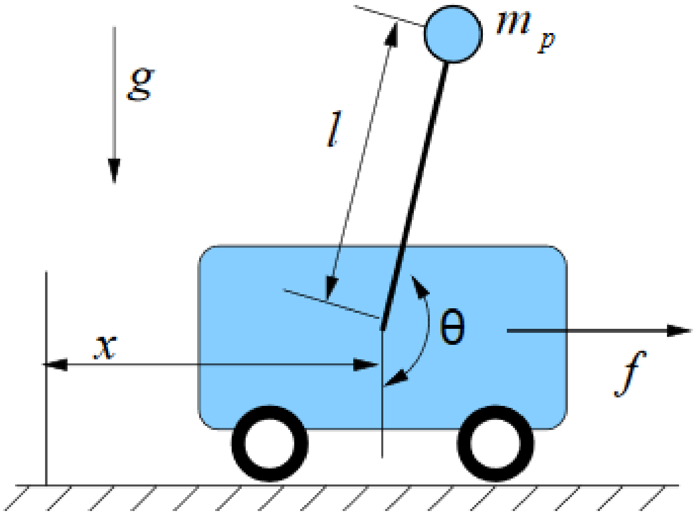
Dynamics of a double-link inverted pendulum (a.k.a Acrobot)

$$
\begin{aligned}
\ddot{\theta}_1 &= -d_1^{-1}(d_2\ddot{\theta}_2 + \phi_1) \\
\ddot{\theta}_2 &= \left(m_2 l_{c2}^2 + I_2 - \frac{d_2^2}{d_1}\right)^{-1}\left(\tau + \frac{d_2}{d_1}\phi_1 - m_2 l_1 l_{c2}\dot{\theta}_1^2 \sin\theta_2 - \phi_2\right) \\
d_1 &= m_1 l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + 2 l_1 l_{c2}\cos\theta_2) + I_1 + I_2 \\
d_2 &= m_2(l_{c2}^2 + l_1 l_{c2}\cos\theta_2) + I_2 \\
\phi_1 &= -m_2 l_1 l_{c2}\dot{\theta}_2^2 \sin\theta_2 - 2 m_2 l_1 l_{c2}\dot{\theta}_2\dot{\theta}_1 \sin\theta_2 \\
&\quad + \ (m_1 l_{c1} + m_2 l_1) g \cos(\theta_1 - \pi/2) + \phi_2 \\
\phi_2 &= m_2 l_{c2} g \cos(\theta_1 + \theta_2 - \pi/2)
\end{aligned}
$$

Provided here just for reference and completeness. You are not expected to know this.

Dynamics of a double-link inverted pendulum (a.k.a Acrobot)

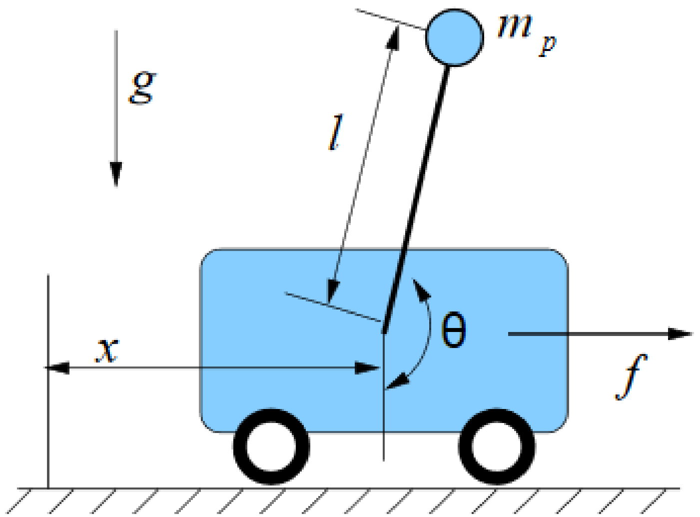
The state of a single-link cartpole



$$\mathbf{x} = [{}^{G}p_x, {}^{G}\dot{p}_x, {}^{G}\theta, {}^{G}\dot{\theta}]$$

State = [Position and velocity of cart, orientation and angular velocity of pole]

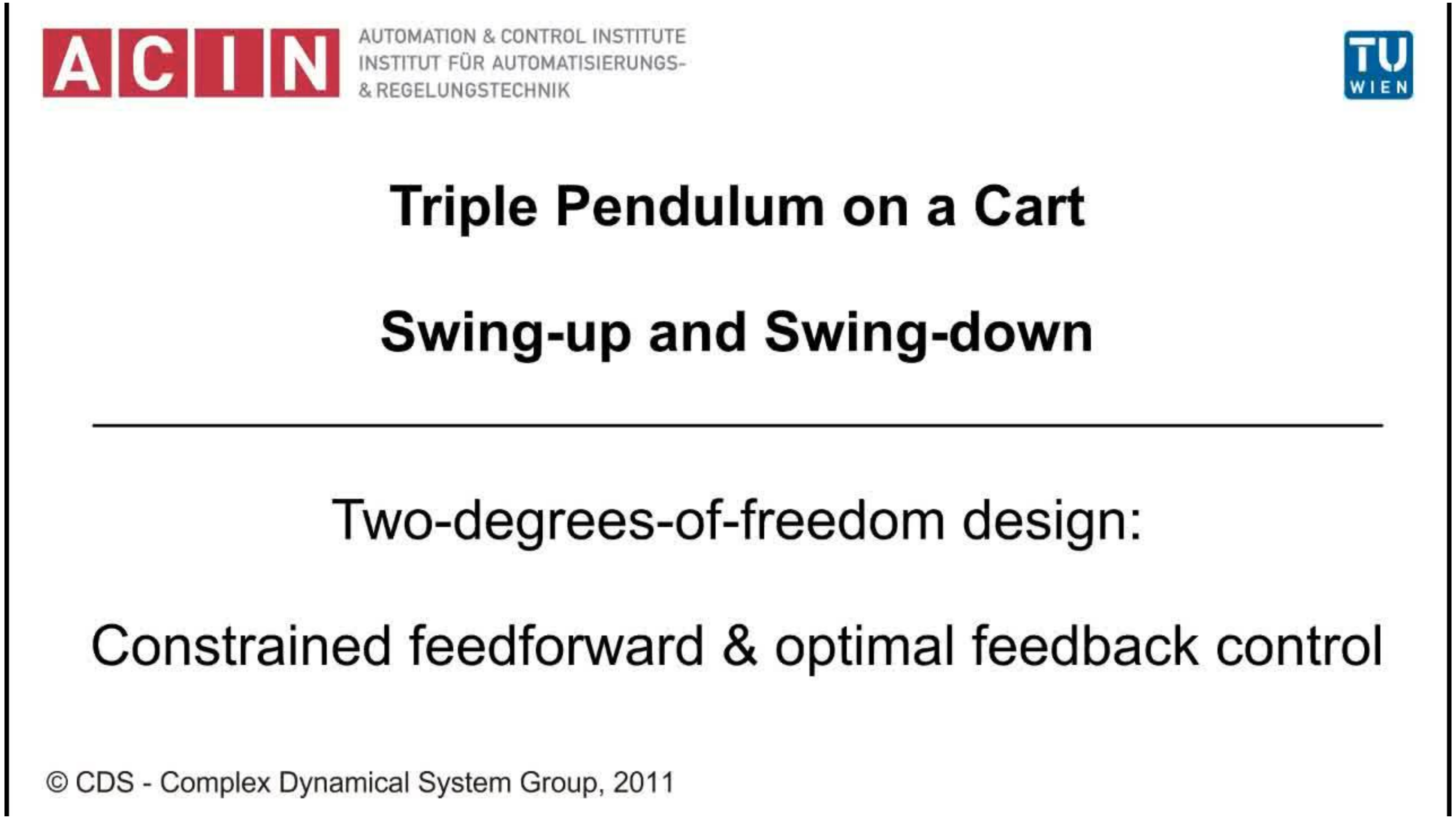
Controls of a single-link cartpole



$$\mathbf{u} = [f]$$

Controls = [Horizontal force applied to cart]

Balancing a triple-link pendulum on a cart



Extreme Balancing

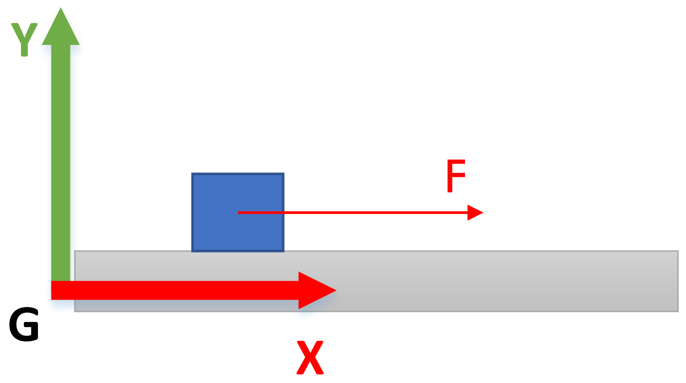
The Cubli

Building a cube that can jump up and balance

IDSC

ETH zürich

The state of a double integrator



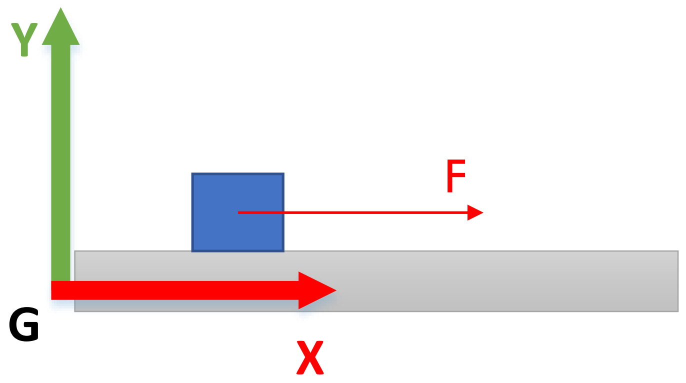
$$\mathbf{x} = \left[{}^{G}p_x \right]$$

State = [Position along x-axis]

Controls of a double integrator

$$\mathbf{u} = [{}^G u_x]$$

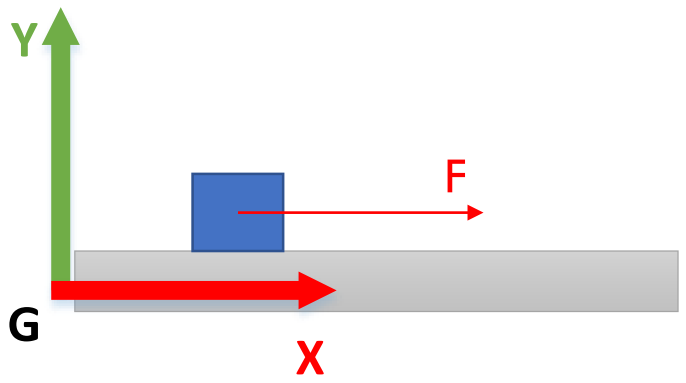
Controls = [Force along x-axis]



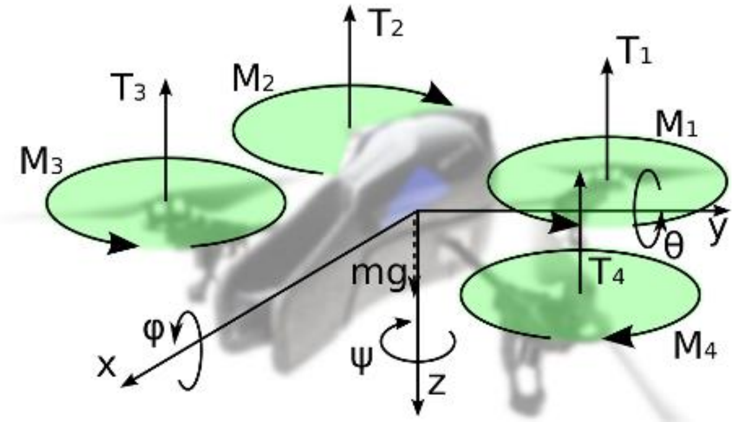
Dynamics of a double integrator

$$\ddot{\mathbf{x}} = \mathbf{F}$$

This corresponds to applying force to a brick of mass 1 to move on frictionless ice. Where is the brick going to end up? Similar to curling.

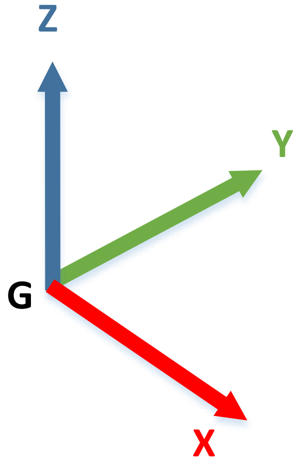


The state of a quadrotor

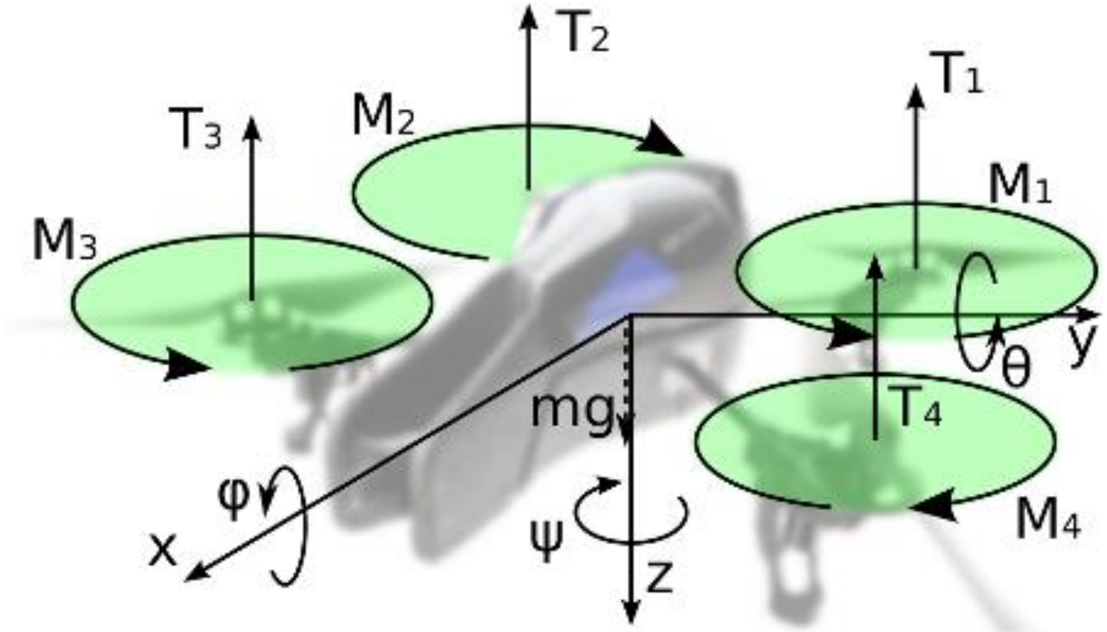


$$\mathbf{x} = [{}^G\phi, {}^G\theta, {}^G\psi, {}^G\dot{\phi}, {}^G\dot{\theta}, {}^G\dot{\psi}]$$

State = [Roll, pitch, yaw, and roll rate, pitch rate, roll rate]
Angles are with respect to the global frame.



Controls of a quadrotor



$$\mathbf{u} = [T_1, T_2, T_3, T_4]$$

Controls = [Thrusts of four motors]

OR

$$\mathbf{u} = [M_1, M_2, M_3, M_4]$$

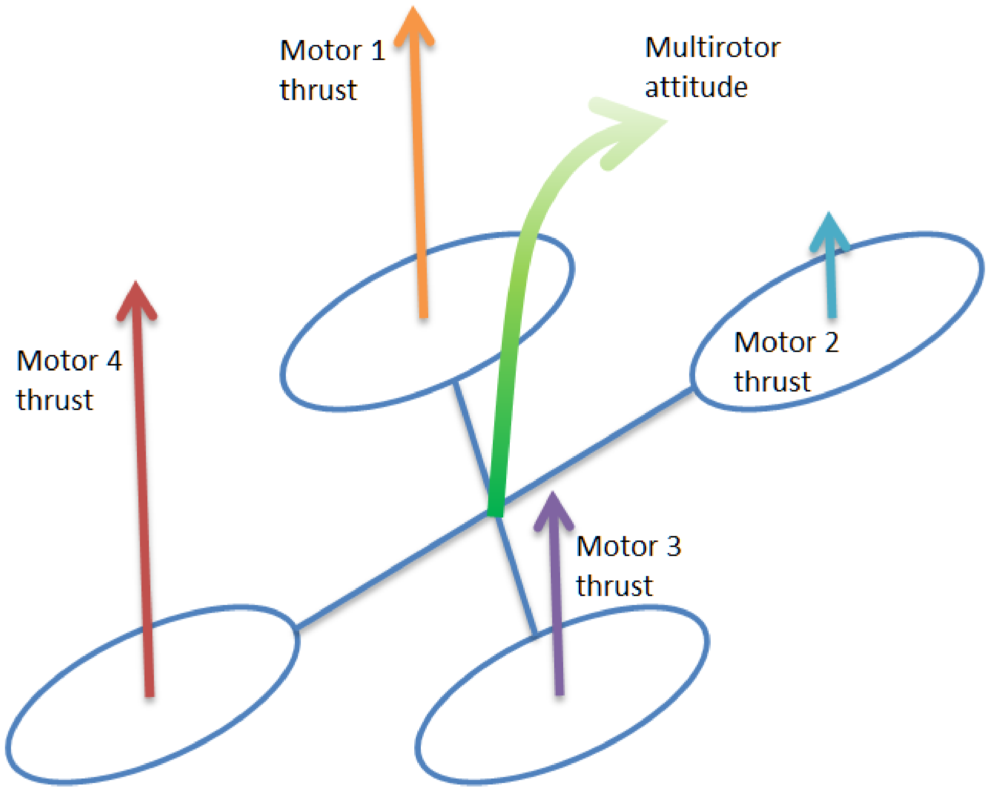
Controls = [Torques of four motors]

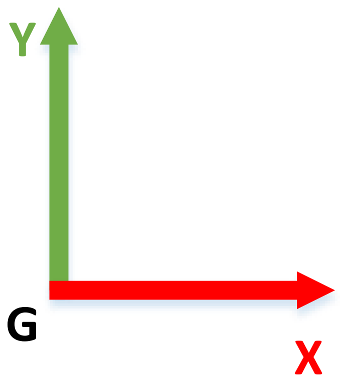
Notice how adjacent motors spin in opposite ways.
Why?

What if all four motors spin the same direction?

Dynamics of a quadrotor

Multirotor(quadcopter)-thrust scheme



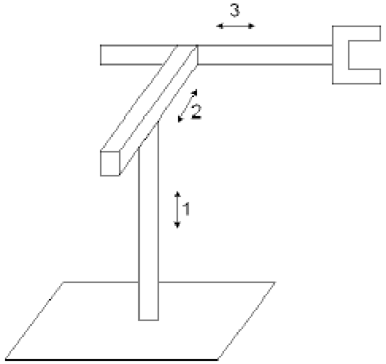


Manipulators

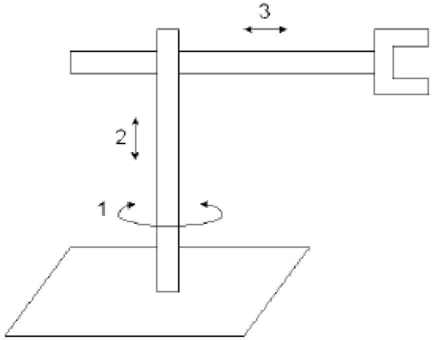
- Robot arms, industrial robot
  - Rigid bodies (links) connected by joints
  - Joints: revolute or prismatic
  - Drive: electric or hydraulic
  - End-effector (tool) mounted on a flange or plate secured to the wrist joint of robot

Manipulators

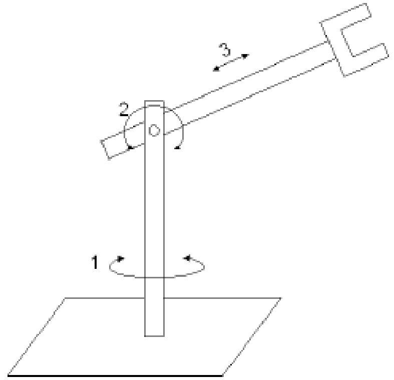
- Robot Configuration:



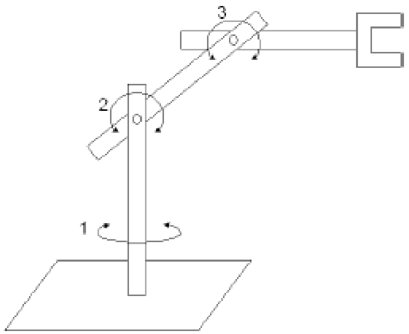
Cartesian: PPP



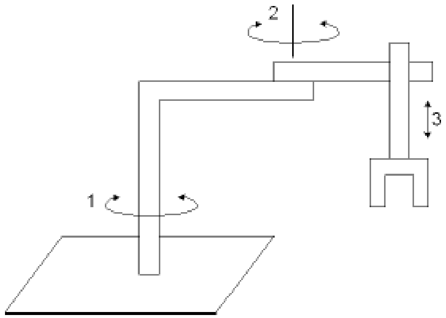
Cylindrical: RPP



Spherical: RRP

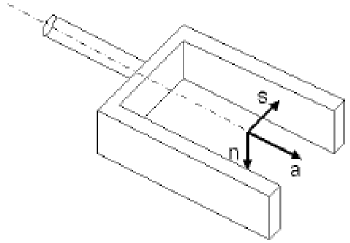


Articulated: RRR



SCARA: RRP

(Selective Compliance Assembly Robot Arm)



Hand coordinate:

n: normal vector; **s**: sliding vector;

a: approach vector, normal to the tool mounting plate

Manipulators

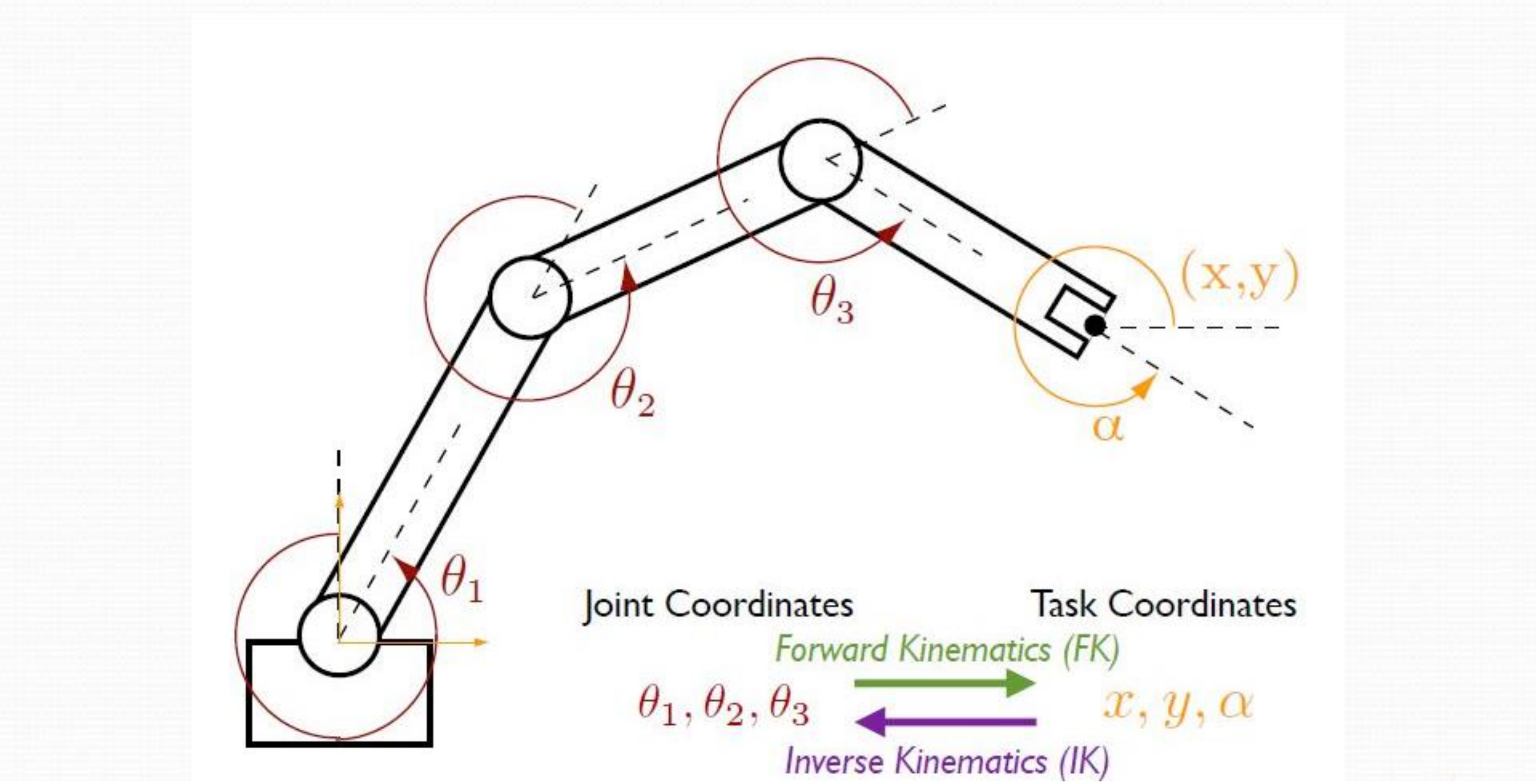
- Motion Control Methods
  - Point to point control
    - a sequence of discrete points
    - spot welding, pick-and-place, loading & unloading
  - Continuous path control
    - follow a prescribed path, controlled-path motion
    - Spray painting, Arc welding, Gluing

Manipulators

- Robot Specifications
  - Number of Axes
    - Major axes, (1-3) => Position the wrist
    - Minor axes, (4-6) => Orient the tool
    - Redundant, (7-n) => reaching around obstacles, avoiding undesirable configuration
  - Degree of Freedom (DOF)
  - Workspace
  - Payload (load capacity)
  - Precision v.s. Repeatability

Which one is more important?

Forward and Inverse Kinematics

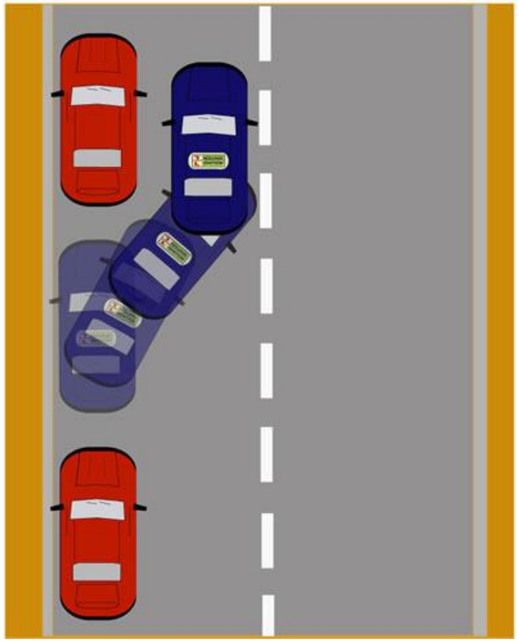


https://slideplayer.com/slide/4239432/

Controllability

- A system is controllable if there exist control sequences that can bring the system from any state to any other state, in finite time.

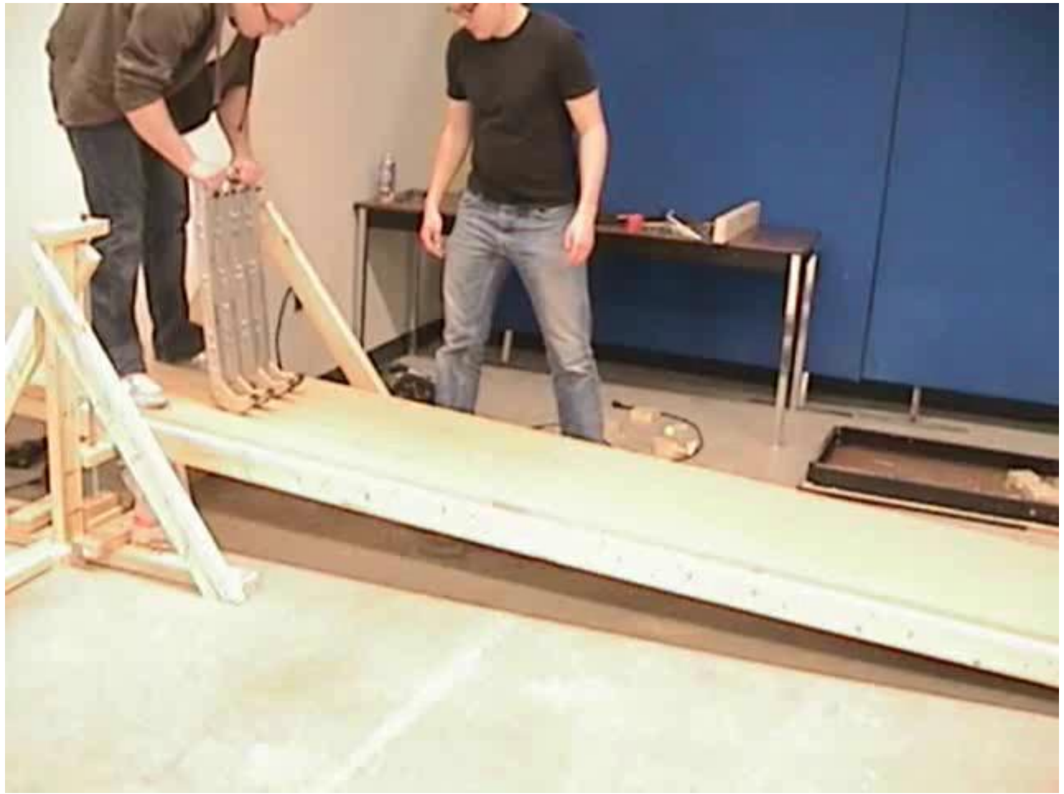
- For example, even though cars are subject to non-holonomic constraints (can't move sideways directly), they are controllable, They can reach sideways states by parallel parking.



Passive Dynamics

- Dynamics of systems that operate without drawing (a lot of) energy from a power supply.

- Interesting because biological locomotion systems are more efficient than current robotic systems.



Passive Dynamics

- Dynamics of systems that operate without drawing (a lot of) energy from a power supply.
- Usually propelled by their own weight.
- Interesting because biological locomotion systems are more efficient than current robotic systems.

Steve Collins & Andy Ruina, Cornell, 2001