اصول علم ربات – جلسه پنجم

Fundamentals of Robotics – Lecture 05

Pose Representations and Transformations

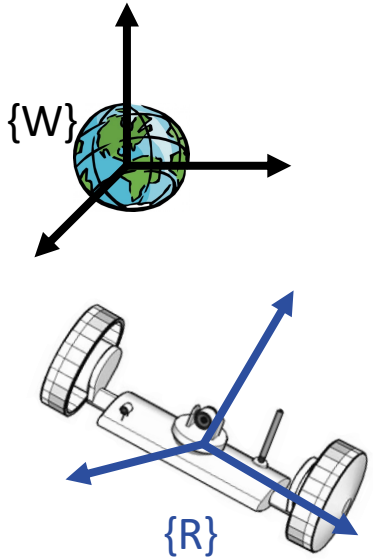
دکتر مهدی جوانمردی

زمستان ۱۴۰۰

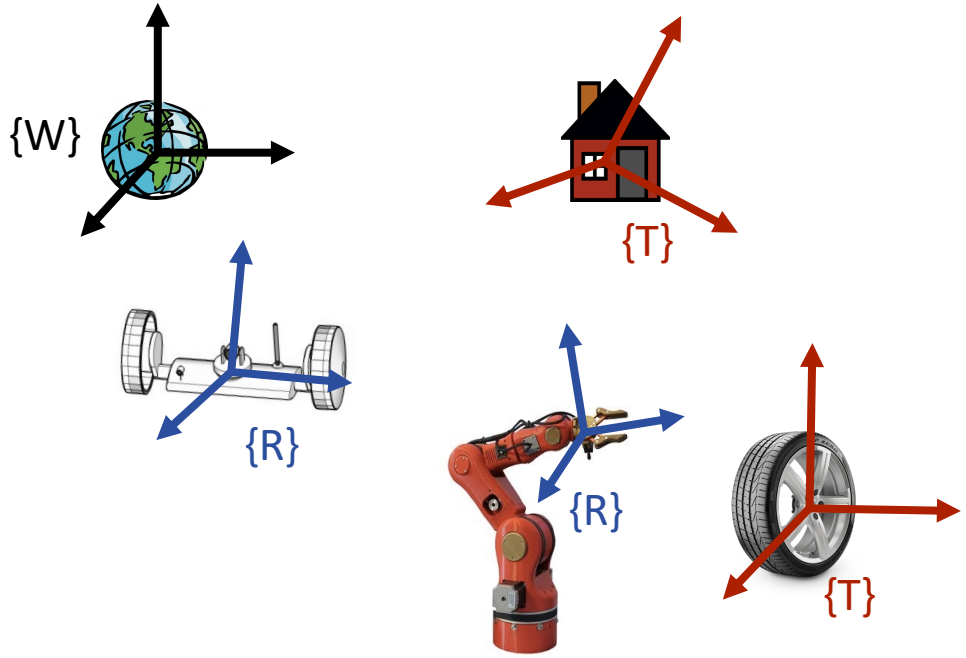
[slides adapted from Gianni Di Caro, @CMU with permission]

Representing & computing (relative) pose: core problem in robotics

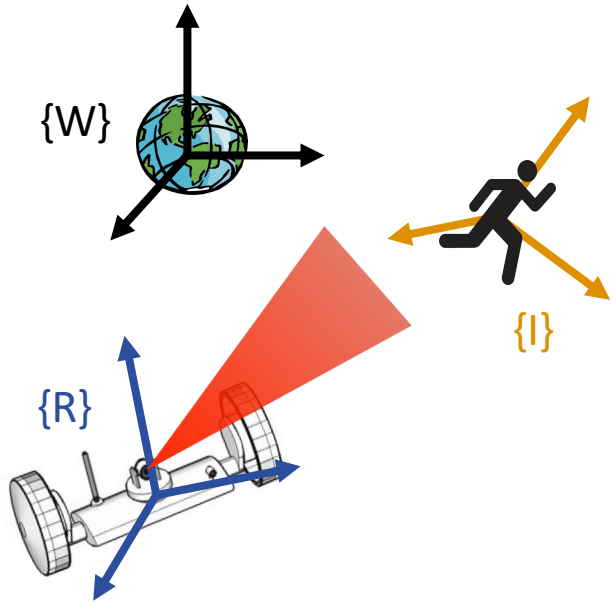
Where's the robot? What is robot's pose with respect to the world reference frame {W}?



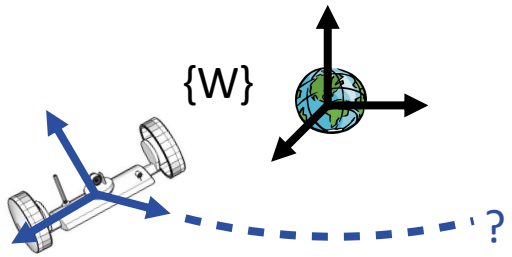
Robot's pose with respect to the external frame {T} (e.g., a target) ?



What is intruder's pose, observed using robot's lateral camera (local frame), in the world frame {W}?



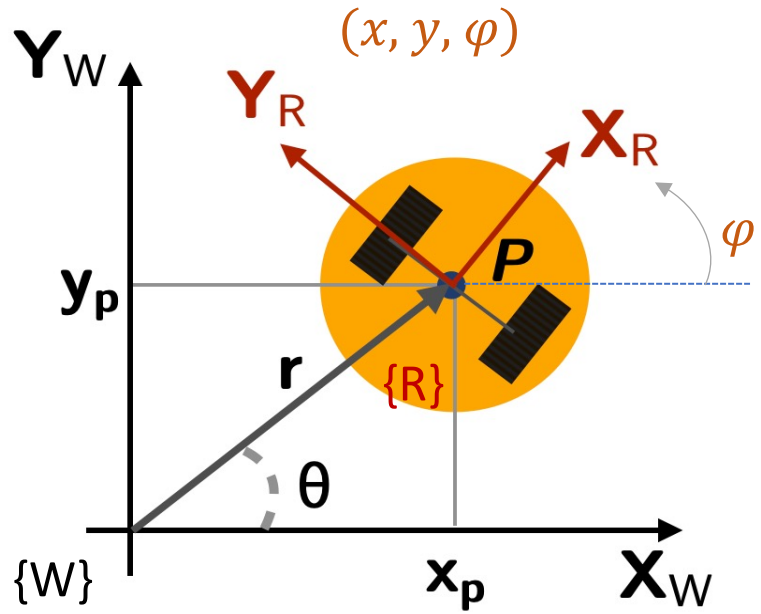
Predict: What is robot's pose in {W} after moving at a velocity v for 1 minute?



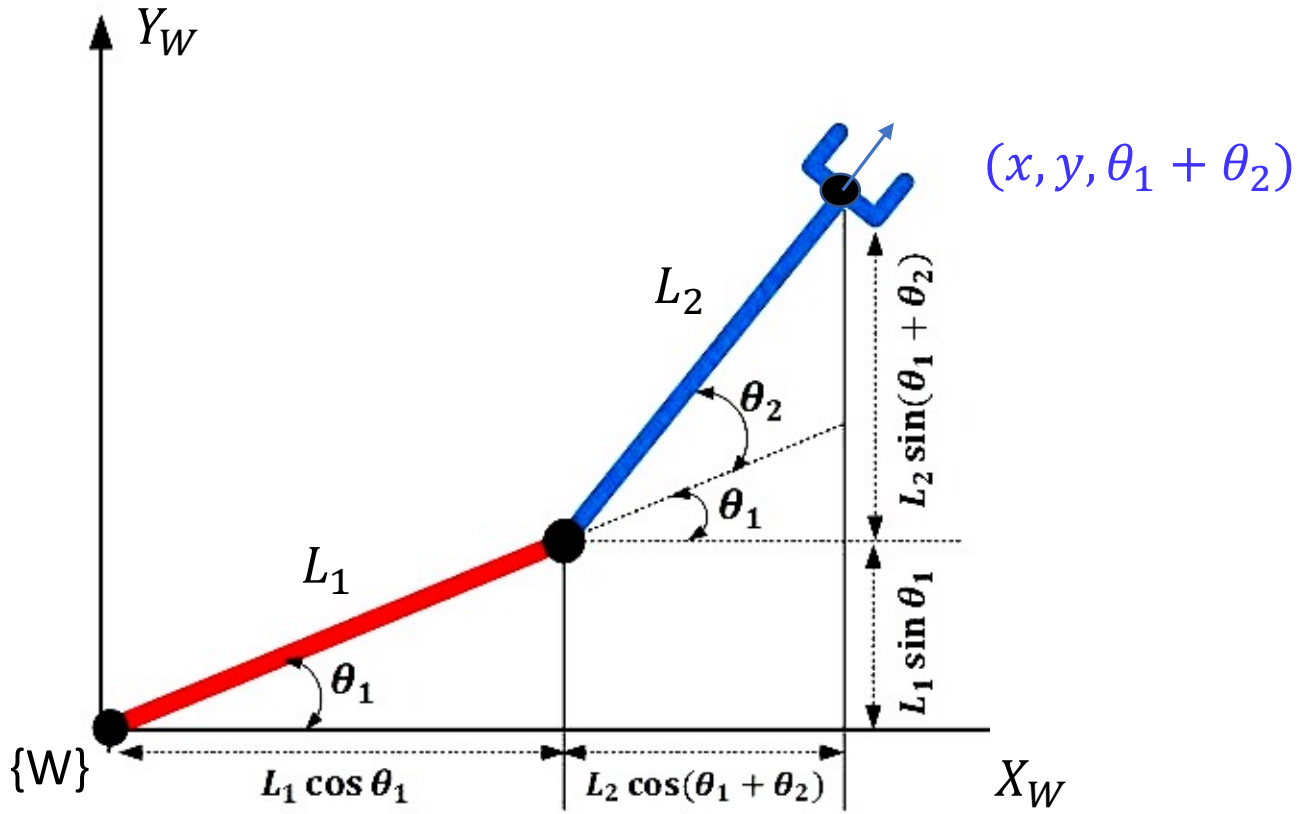
Plan: What is the velocity profile that allows to reach a pose ξ in {W}?

Robot Pose

Pose: *position + orientation* of the robot relative to some reference system



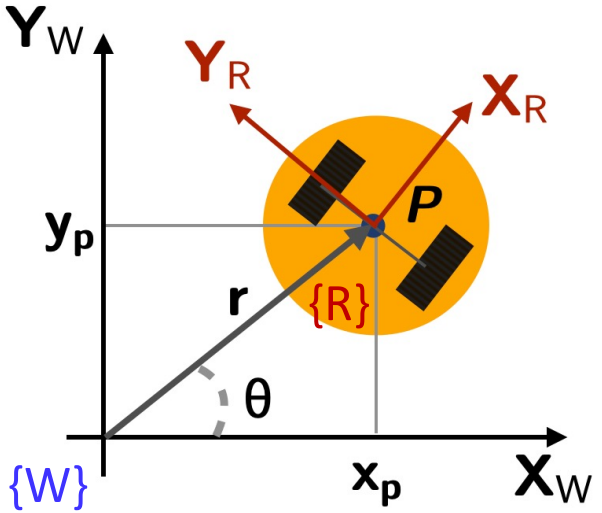
Single body



Multi-body

Robot Pose (for single body robots)

- Choose a Coordinate System, select a Fixed World Reference Coordinate Frame {W}
- Center a (local) coordinate frame {R} in a selected robot's **reference point** P, (possibly) oriented according to robot's natural orientation
- A (reference) point P is described by a coordinate vector $\boldsymbol{r}$ representing the displacement of the point with respect to the *reference coordinate frame* {W}
- Different coordinate systems can be used based on convenience:
  Cartesian, Polar, Spherical, Cylindric,…



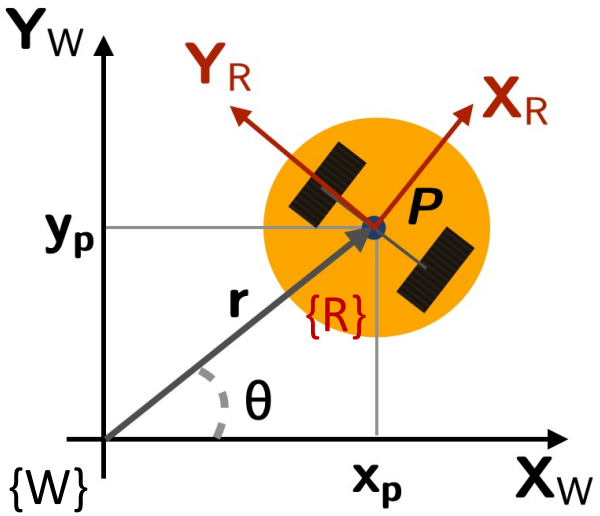
Position of P (origin of {R})

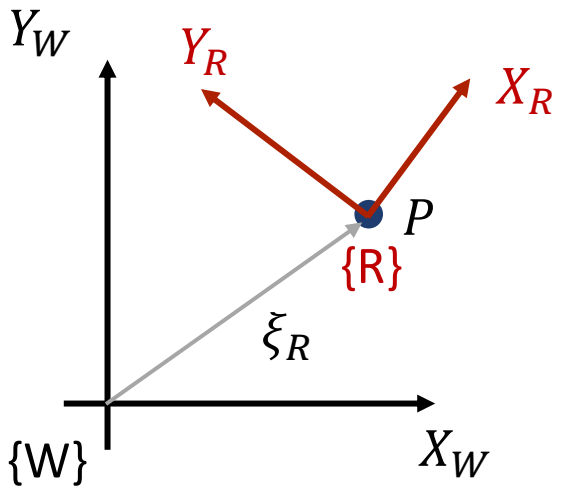
Cartesian:
$(x = 3, y = 4)$

Polar:
$(r = \sqrt{3^2 + 4^2} = 5,\ \theta = \arctan\frac{4}{3} \cong 53.13^{\circ})$

Relative Pose

The **(relative) pose/configuration** of the object/robot in {W} is described by the **position** and **orientation** of the (local) coordinate frame {R} with respect to {W}

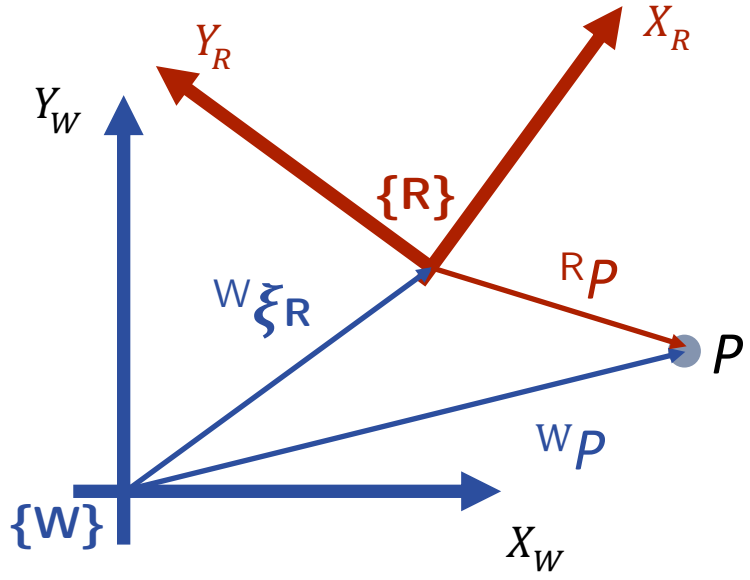




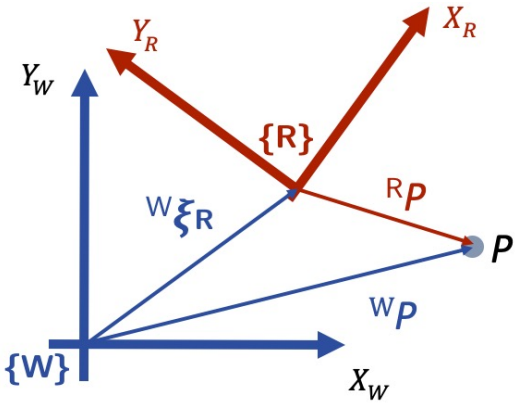
ξ_R is the relative pose of the frame/robot with respect to the reference coordinate frame

Use and properties of relative poses

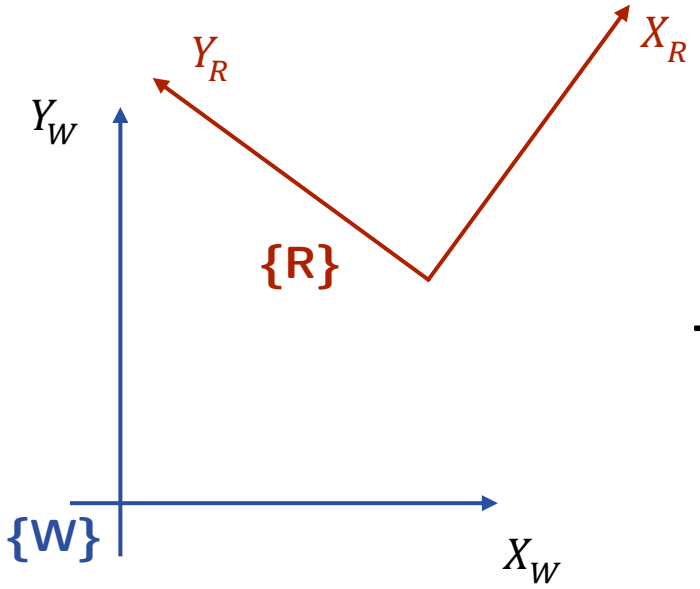
- **The relative pose ${}^{W}\xi_{R}$ describes the frame {R} with respect to the frame {W}**
- The leading superscript denotes the reference coordinate frame, the subscript denotes the frame being described.
  - E.g, ${}^{R}\xi_{W}$ describes the pose of the world frame {W} from the robot's frame {R} point of view
- If the initial superscript is missing the reference frame is a world coordinate frame.
- ξ is the object being described according to chosen coordinate system



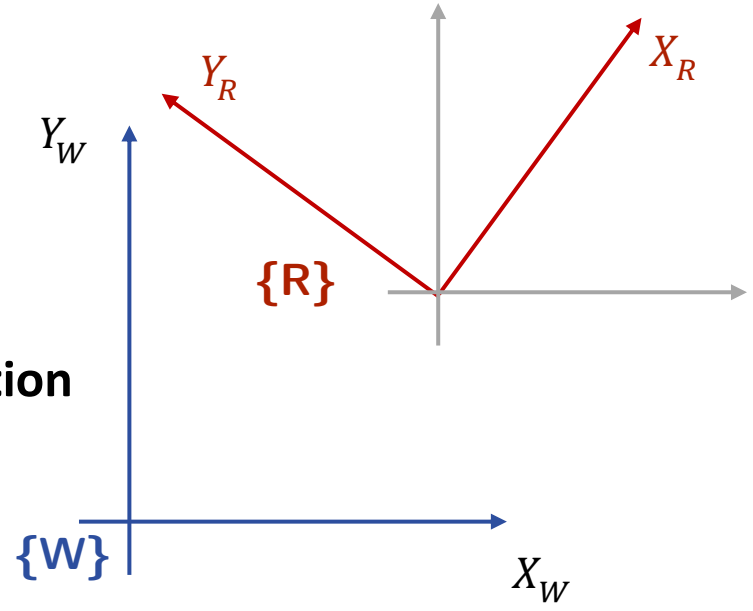
Use and properties of relative poses

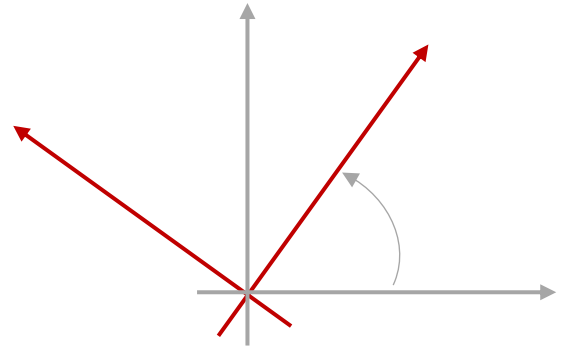


${}^W\xi_R$ can be seen as describing a **motion**: {R} results from first applying a **translation** and then a **rotation** to {W}



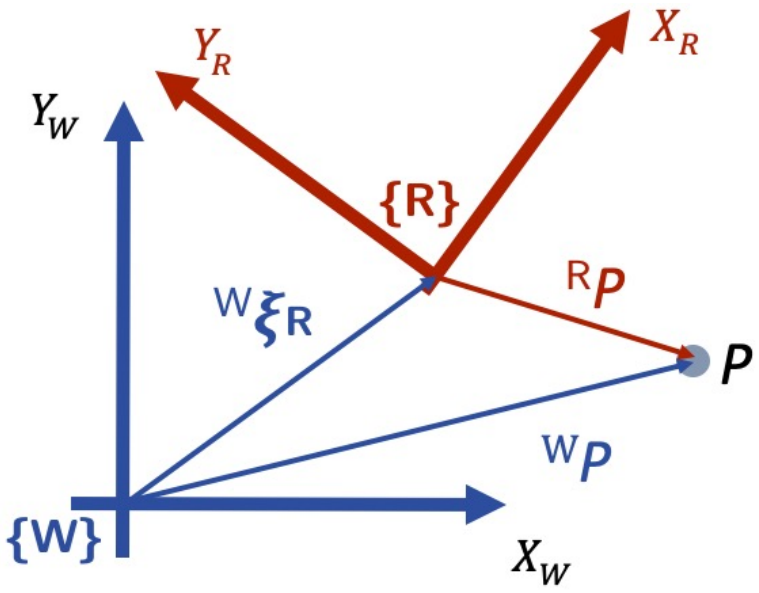
Translation





Rotation

Use and properties of relative poses



- The point *P* can be described with respect to either coordinate frame, ${}^{R}P$, ${}^{W}P$
- Transformation of coordinates can be done using the relative pose between the two frames:

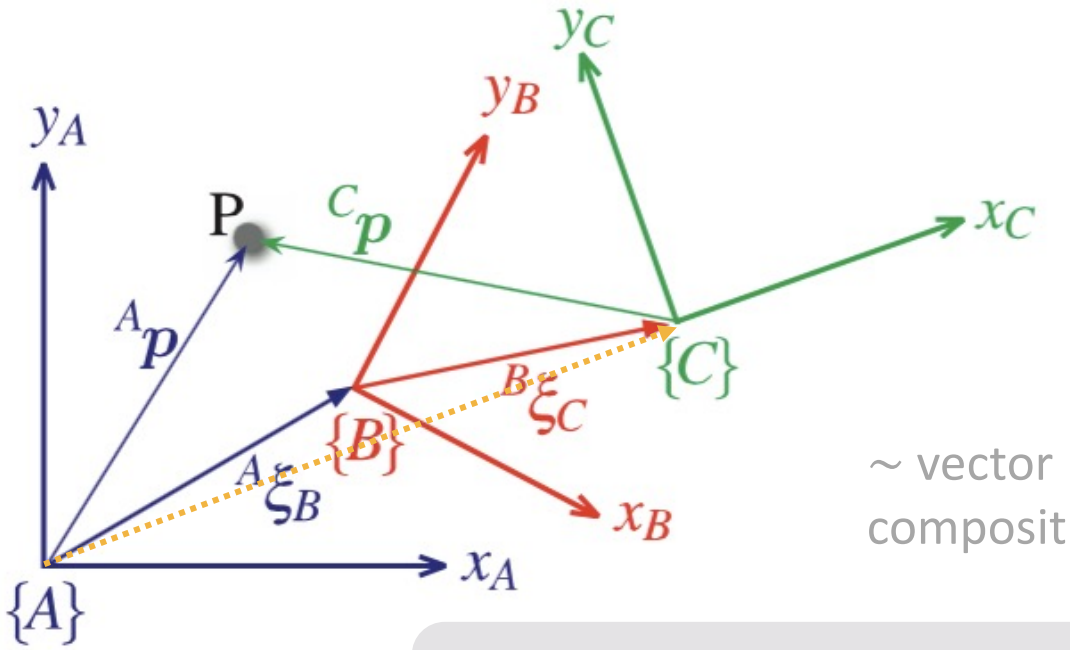
$${}^{W}P = {}^{W}\xi_R \cdot {}^{R}P$$

$${}^{R}P = {}^{R}\xi_W \cdot {}^{W}P$$

✓ The operator $\cdot$ transforms the position vector, resulting in a **new *position* vector** that describes the same point but with respect to a different coordinate frame.

${}^{W}P = {}^{W}\xi_R \cdot {}^{R}P$ The right-hand side expresses the **motion**: {W} → {R} → P

Composition of relative poses



~ vector composition!

- One frame can be described in terms of another by using their relative pose
- ➢ The procedure can be applied **sequentially** across multiple frames by a relative pose composition operator, $\oplus$

Pose({C} relative to {A}) = **Pose**({B} relative to {A}) $\oplus$ **Pose**({C} relative to {B})

$$ {}^A\xi_C = {}^A\xi_B \oplus {}^B\xi_C $$

- A relative pose can transform a point expressed as a vector relative to one frame to a vector relative to another using the $\oplus$ and $\cdot$ operators:

$$ {}^A P = {}^A\xi_C \cdot {}^C P = ({}^A\xi_B \oplus {}^B\xi_C) \cdot {}^C P $$

Poses form an additive group

The set of poses equipped with the combination operator $\oplus$ form an *additive group.*
In the case of a *mobile robot* this is the **SE(2) Special Euclidean group** (SE(3) in 3D)

~ Addition
Preserve Euclidean distances

Closure: $\xi_1 \oplus \xi_2 = \xi_3$

- ✓ Composition of two poses results in a new pose: ${}^{A}\xi_{B} \oplus {}^{B}\xi_{C} = {}^{A}\xi_{C}$ ${}^{A}\xi_{B} \oplus {}^{A}\xi_{C} = {}^{B}\xi_{C}$
- ➢ Composition is NOT commutative:
  (because of the angle part of poses) $\xi_1 \oplus \xi_2 \neq \xi_2 \oplus \xi_1$

Associativity: $({}^{A}\xi_{B} \oplus {}^{B}\xi_{C}) \oplus {}^{C}\xi_{D} = {}^{A}\xi_{B} \oplus ({}^{B}\xi_{C} \oplus {}^{C}\xi_{D})$

Poses form an additive group

Identity element: $\xi \oplus \mathbf{0} = \mathbf{0} \oplus \xi = \xi$

0 is the *null relative pose*

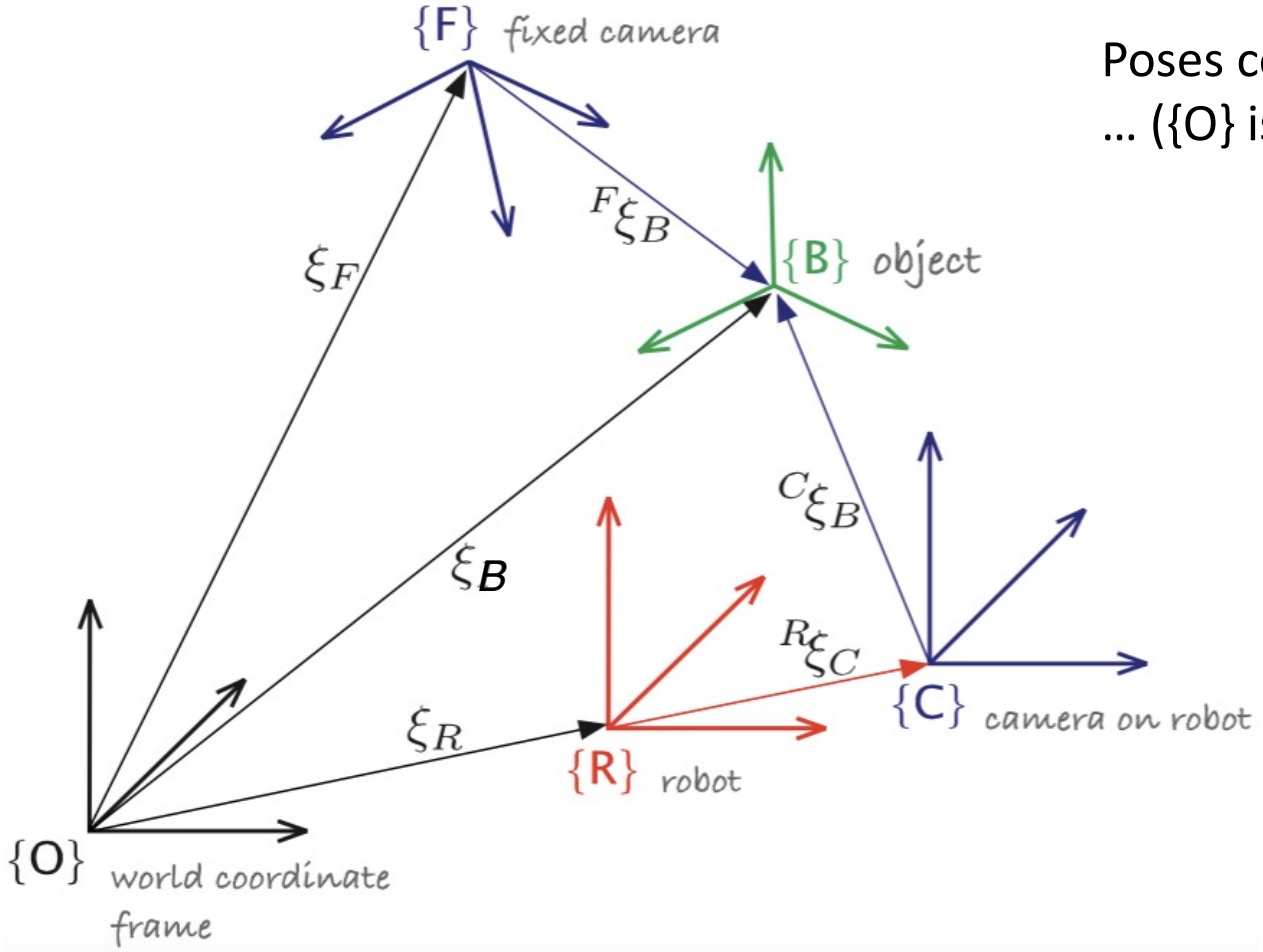
Inverse: $\ominus\, {}^{A}\xi_{B} = {}^{B}\xi_{A}$

$$(\ominus\xi) \oplus \xi = \xi \oplus (\ominus\xi) = \mathbf{0}$$

$$\xi \ominus \mathbf{0} = \xi$$

$$\xi \ominus \xi = \mathbf{0}$$

A more complex 3D example



Poses composition allows to represent useful **spatial relations** … ({O} is dropped for short)

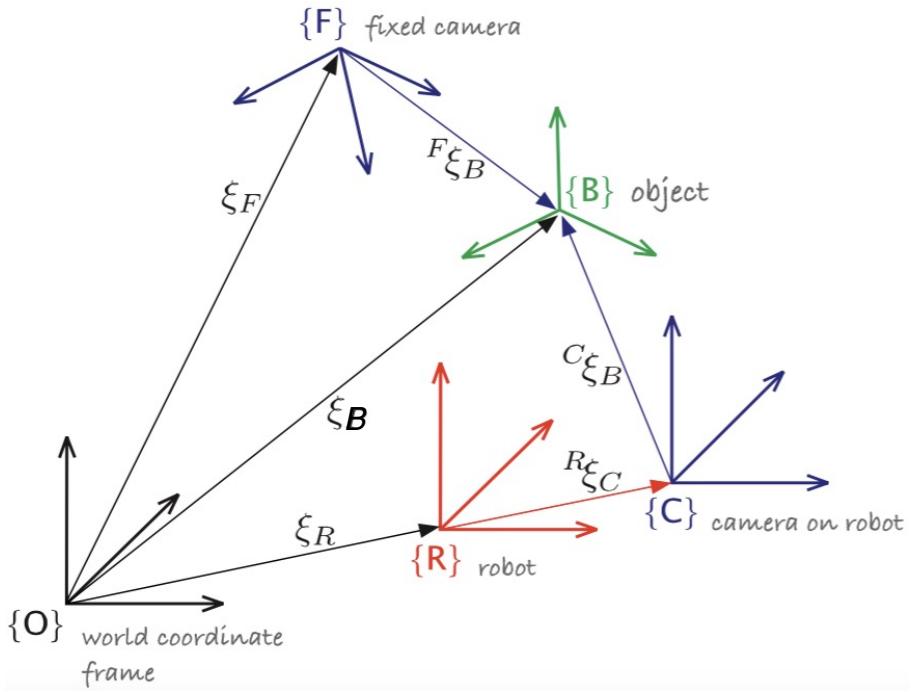
- $\xi_F \oplus {}^F\xi_B = \xi_R \oplus {}^R\xi_C \oplus {}^C\xi_B$
- $\xi_F \oplus {}^F\xi_R = \xi_R$

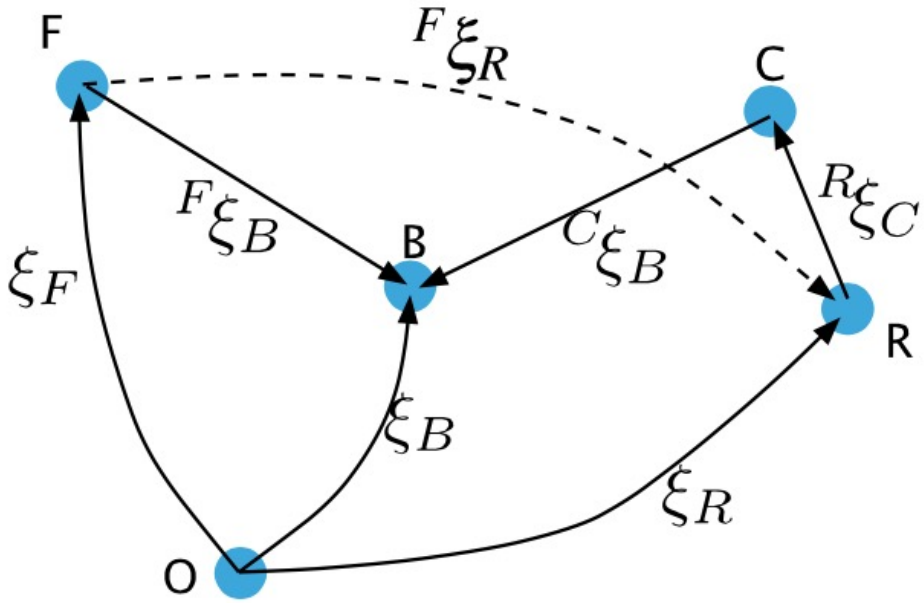
… a perform computations
(adding the inverse to both sides of 2nd eq.)

$$\ominus\xi_F \oplus \xi_F \oplus {}^F\xi_R = \ominus\xi_F \oplus \xi_R$$

Pose of the robot relative to the fixed camera $^F\xi_R = \ominus\xi_F \oplus \xi_R$ $^F\xi_R = {}^F\xi_O \oplus \xi_R$

A more complex 3D example: Graph description





In the graph, each node represents a pose, and each arc is a relative pose *which is known*

- A spatial relation is a *loop in the graph that can be exploited to derive relative poses*
- Both sides of an equation start and end at the same node.

$$\xi_F \oplus {}^F\xi_B = \xi_R \oplus {}^R\xi_C \oplus {}^C\xi_B$$

$$\xi_F \oplus {}^F\xi_R = \xi_R$$

RECAP

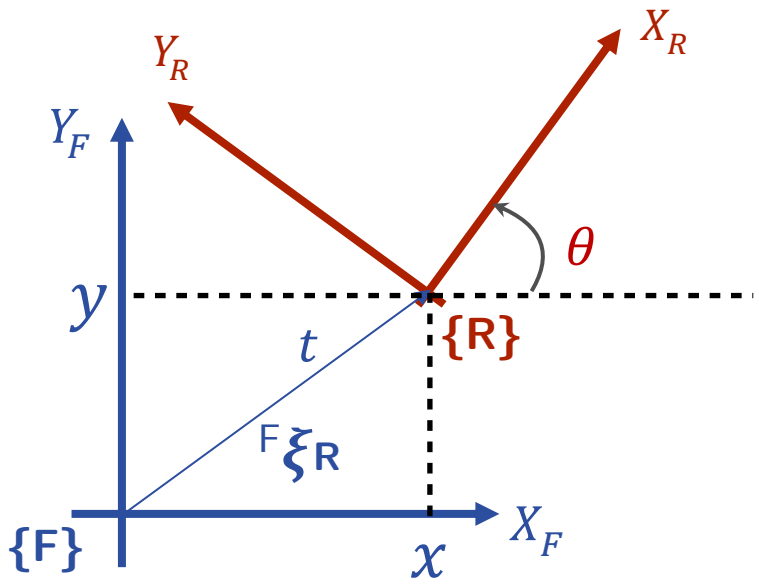
1. A point is described by a coordinate vector that represents its displacement from a reference coordinate system;
2. A set of points that represent a rigid object can be described by a single coordinate frame, and its constituent points are described by displacements from that coordinate frame;
3. The position and orientation of an object's coordinate frame is referred to as its pose;
4. A relative pose describes the pose of one coordinate frame with respect to another and is denoted by an algebraic variable ξ;
5. A coordinate vector describing a point can be represented with respect to a different coordinate frame by applying the relative pose to the vector using the $\cdot$ operator;
6. We can perform algebraic manipulation of expressions written in terms of relative poses.

Poses in 2D (for mobile robots)

What are in practice ξ and ⊕?

✓ Any mathematical objects/operators that support the described group properties (algebra) and are suited to the problem at hand.

➢ In 2D, for a wheeled robot, a concrete representation of a pose is in a Cartesian coordinate system through the (x, y) coordinates and the θ angle for the orientation

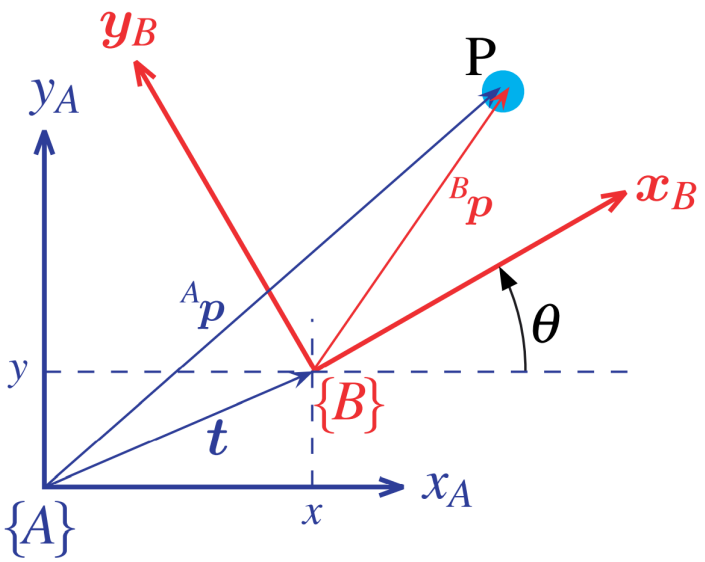


$${}^F\xi_R \sim (x, y, \theta)$$

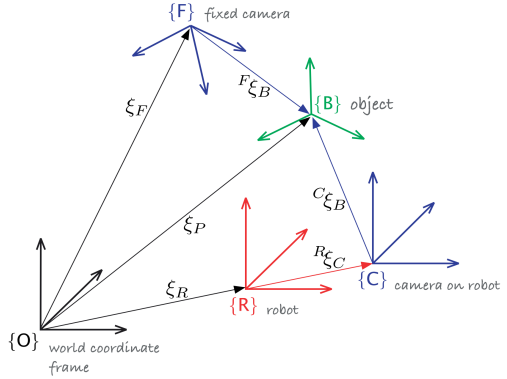
Remember, to describe a coordinate frame {R} wrt a reference frame {F}:

1. Origin of {R} has been displaced by the vector $t = (x, y)$
   → **Translation**
2. {R} has been rotated counter-clockwise by an angle θ
   → **Rotation**

Poses in 2D



$$\boldsymbol{\xi}_F \oplus {}^F\boldsymbol{\xi}_B = \boldsymbol{\xi}_R \oplus {}^R\boldsymbol{\xi}_C \oplus {}^C\boldsymbol{\xi}_B$$



Two representations are equivalent

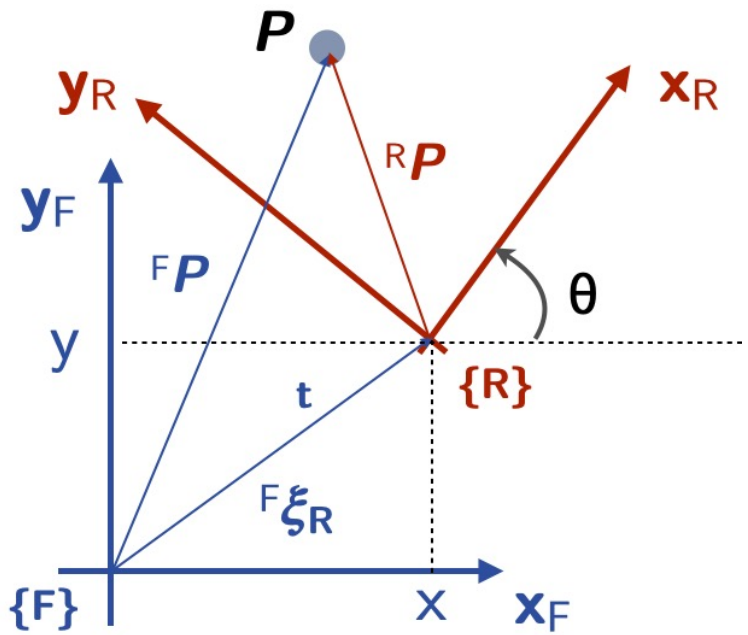
${}^F\boldsymbol{\xi}_R \sim (x, y, \theta)$

Unfortunately, this representation is not really convenient for compounding since

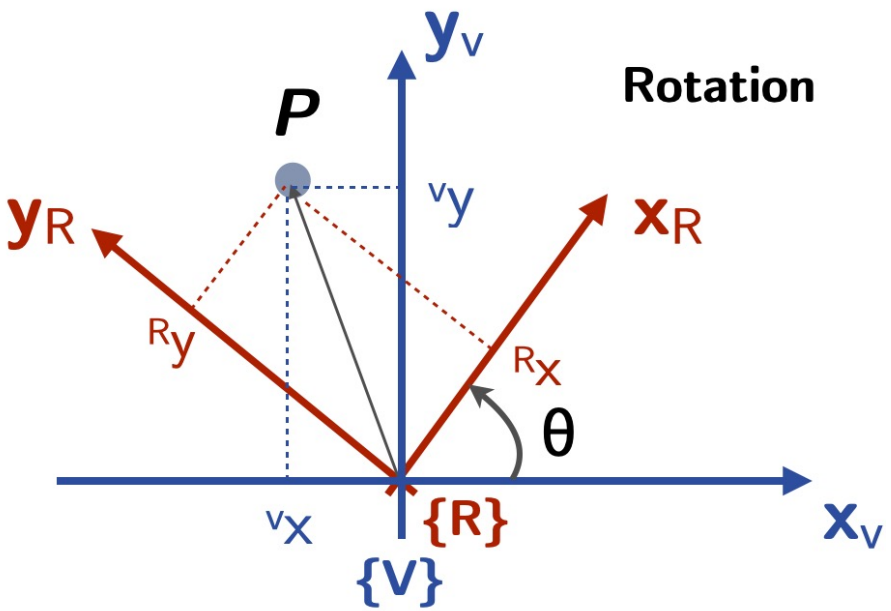
$$(x_1, y_1, \theta_1) \oplus (x_2, y_2, \theta_2)$$

would require some *complex trigonometric functions to compose angles* (composing cartesian coordinates is instead easy)

Use of matrix operators for representing poses

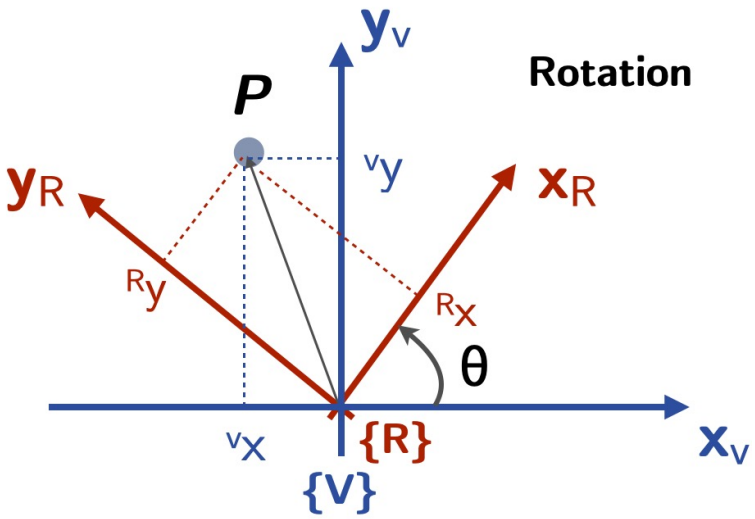


- Let's consider an arbitrary point ***P*** and its position with respect to each of the coordinate frames {F} and {R}
- The relationship between ${}^{F}\boldsymbol{P}$ and ${}^{R}\boldsymbol{P}$ can be determined as *rotation* ⊕ *translation*



- Let {V} be a new coordinated frame, centered in {R} but rotated as {F}
  - The new frame {V} has axes parallel to {F} but origin is the same as {R}

Use of matrix operators for representing poses



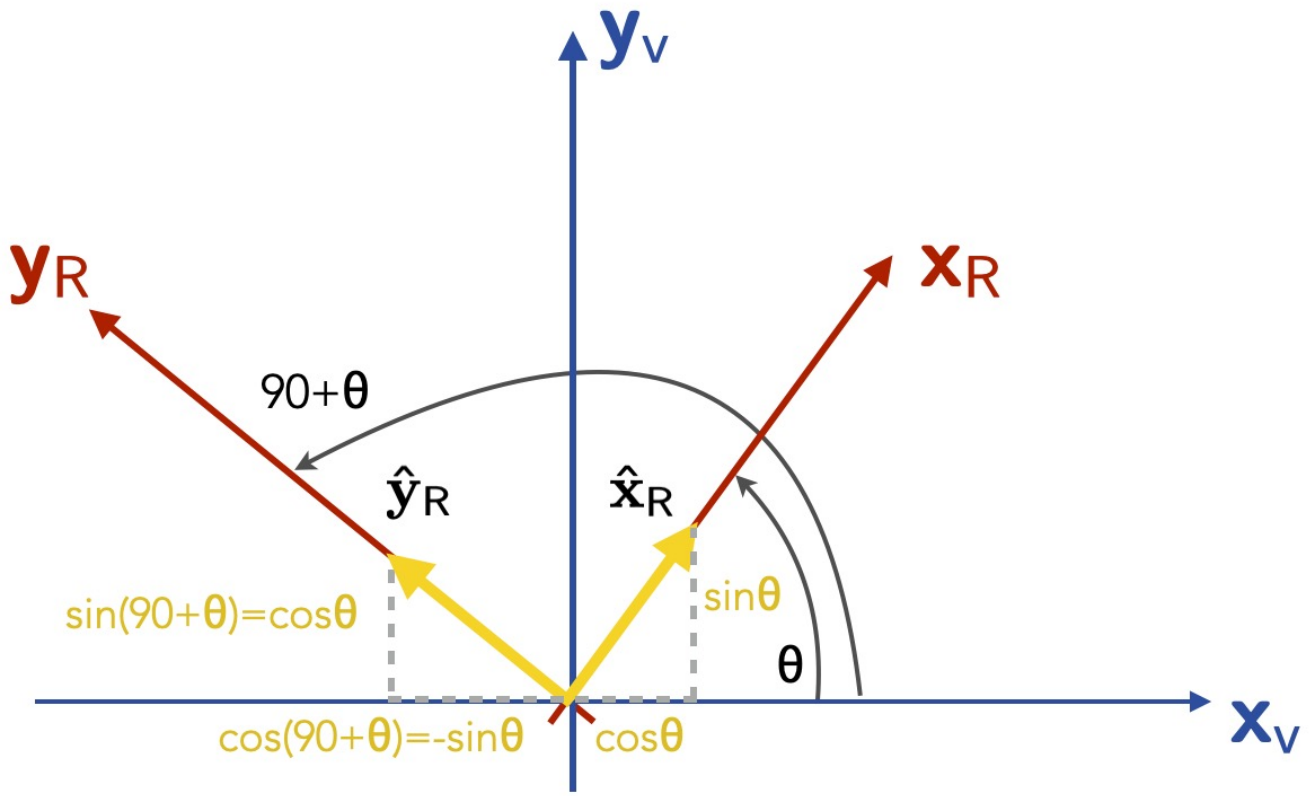
Coordinates of point **P** in {V}: ${}^{v}\mathbf{P} = {}^{v}x\,\hat{\mathbf{x}}_v + {}^{v}y\,\hat{\mathbf{y}}_v = \begin{bmatrix}\hat{\mathbf{x}}_v & \hat{\mathbf{y}}_v\end{bmatrix}\begin{bmatrix}{}^{v}x \\ {}^{v}y\end{bmatrix}$

In {V}, the coordinate unit vectors: $\hat{\mathbf{x}}_v = (1,0)$, $\hat{\mathbf{y}}_v = (0,1)$

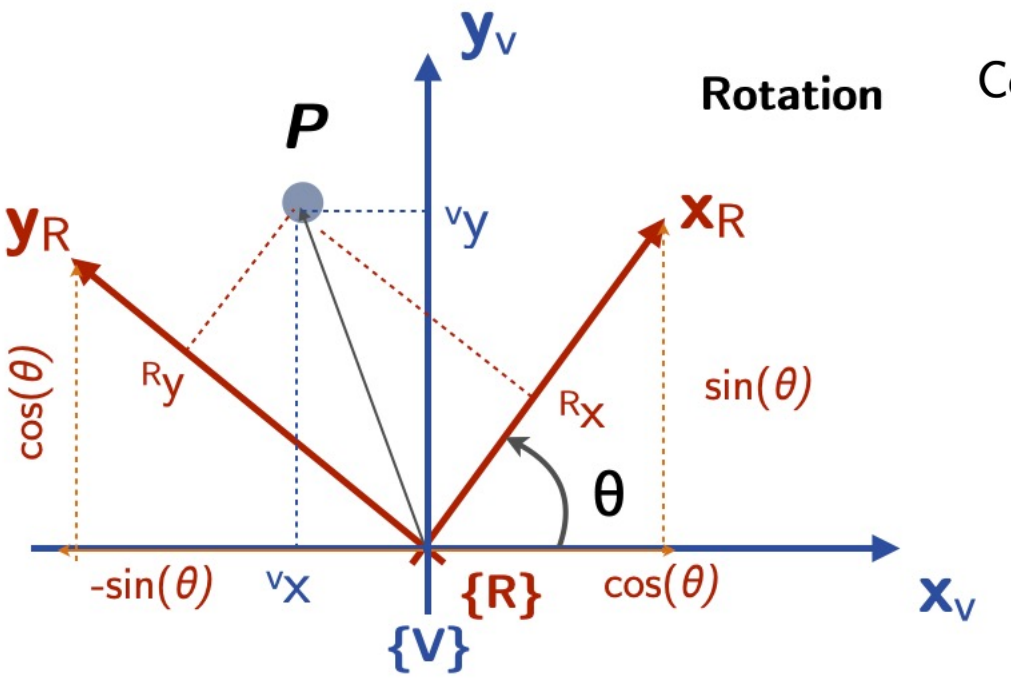
Coordinate frame {R} is fully described by its orthogonal axes, whose *unit vectors* can be expressed in terms of {V}'s *unit vectors*:

$$\hat{\mathbf{x}}_R = \cos(\theta)\hat{\mathbf{x}}_v + \sin(\theta)\hat{\mathbf{y}}_v$$
$$\hat{\mathbf{y}}_R = -\sin(\theta)\hat{\mathbf{x}}_v + \cos(\theta)\hat{\mathbf{y}}_v$$

$$\begin{bmatrix}\hat{\mathbf{x}}_R & \hat{\mathbf{y}}_R\end{bmatrix} = \begin{bmatrix}\hat{\mathbf{x}}_v & \hat{\mathbf{y}}_v\end{bmatrix}\begin{bmatrix}\cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta)\end{bmatrix}$$



Use of matrix operators for representing poses



Coordinates of point **P** in {V}: $^V\mathbf{P} = {}^Vx\,\hat{\mathbf{x}}_V + {}^Vy\hat{\mathbf{y}}_V = \begin{bmatrix}\hat{\mathbf{x}}_V & \hat{\mathbf{y}}_V\end{bmatrix}\begin{bmatrix}{}^Vx\\ {}^Vy\end{bmatrix}$

$$^R\mathbf{P} = {}^Rx\,\hat{\mathbf{x}}_R + {}^Ry\hat{\mathbf{y}}_R = \begin{bmatrix}\hat{\mathbf{x}}_R & \hat{\mathbf{y}}_R\end{bmatrix}\begin{bmatrix}{}^Rx\\ {}^Ry\end{bmatrix}$$

$$\begin{bmatrix}\hat{\mathbf{x}}_R & \hat{\mathbf{y}}_R\end{bmatrix} = \begin{bmatrix}\hat{\mathbf{x}}_V & \hat{\mathbf{y}}_V\end{bmatrix}\begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}$$

Substituting ...

$$^R\mathbf{P} = \begin{bmatrix}\hat{\mathbf{x}}_V & \hat{\mathbf{y}}_V\end{bmatrix}\begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}\begin{bmatrix}{}^Rx\\ {}^Ry\end{bmatrix}$$

Transformation of point coordinates (from {R} to {V}) when the frame is rotated by an angle θ:

$$\begin{bmatrix}{}^Vx\\ {}^Vy\end{bmatrix} = \begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}\begin{bmatrix}{}^Rx\\ {}^Ry\end{bmatrix}$$

$$\begin{bmatrix}{}^Vx\\ {}^Vy\end{bmatrix} = {}^V\mathbf{R}_R(\theta)\begin{bmatrix}{}^Rx\\ {}^Ry\end{bmatrix} \qquad \begin{bmatrix}{}^Rx\\ {}^Ry\end{bmatrix} = {}^V\mathbf{R}_R^{-1}(\theta)\begin{bmatrix}{}^Vx\\ {}^Vy\end{bmatrix}$$

$\boldsymbol{R(\theta), R^{-1}(\theta)}$ **- Rotation matrices**

Rotation matrix

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \qquad \mathbf{R}^{-1}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

- Instead of using one scalar, the angle θ, we are using a 2x2 matrix to represent orientation!
- The rotation matrix is **orthonormal:** each of its columns is a unit vector and the columns are orthogonal, that is, represent an **orthonormal basis** (the columns are the unit vectors that define {R} with respect to {V})
  - 4 parameters, 3 functional relations/constraints → one independent value (the angle!)
- The determinant is +1: R belongs to the special orthogonal group of dimension 2, SO(2), acting as an isometry → length of a vector is unchanged after a rotation

Rotation matrix

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \qquad \mathbf{R}^{-1}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

- The inverse is the same as the transpose: $R^{-1} = R^T$
- Inverting the matrix is the same as swapping superscript and subscript, which leads to the identity:

$$\mathbf{R}(-\theta) = \mathbf{R}^T(\theta) = \mathbf{R}^{-1}(\theta)$$

$$\begin{bmatrix} {}^{R}x \\ {}^{R}y \end{bmatrix} = {}^{v}\mathbf{R}_{R}^{-1}(\theta) \begin{bmatrix} {}^{v}x \\ {}^{v}y \end{bmatrix} = {}^{v}\mathbf{R}_{R}^{T}(\theta) \begin{bmatrix} {}^{v}x \\ {}^{v}y \end{bmatrix} = {}^{R}\mathbf{R}_{v}(\theta) \begin{bmatrix} {}^{v}x \\ {}^{v}y \end{bmatrix}$$

Common 2D rotations

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Common rotation matrices:

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

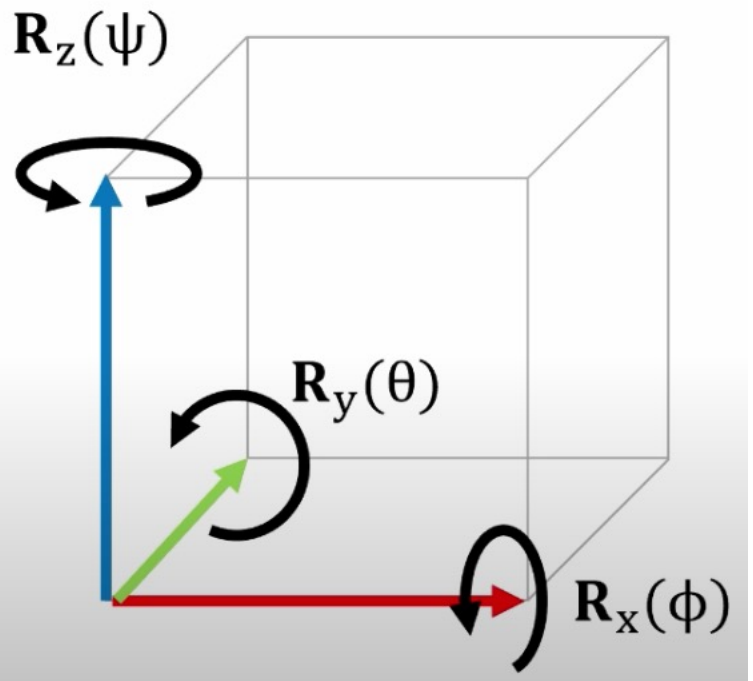
90° 180° 270°

3D: rotations about the axes (elementary rotations)

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

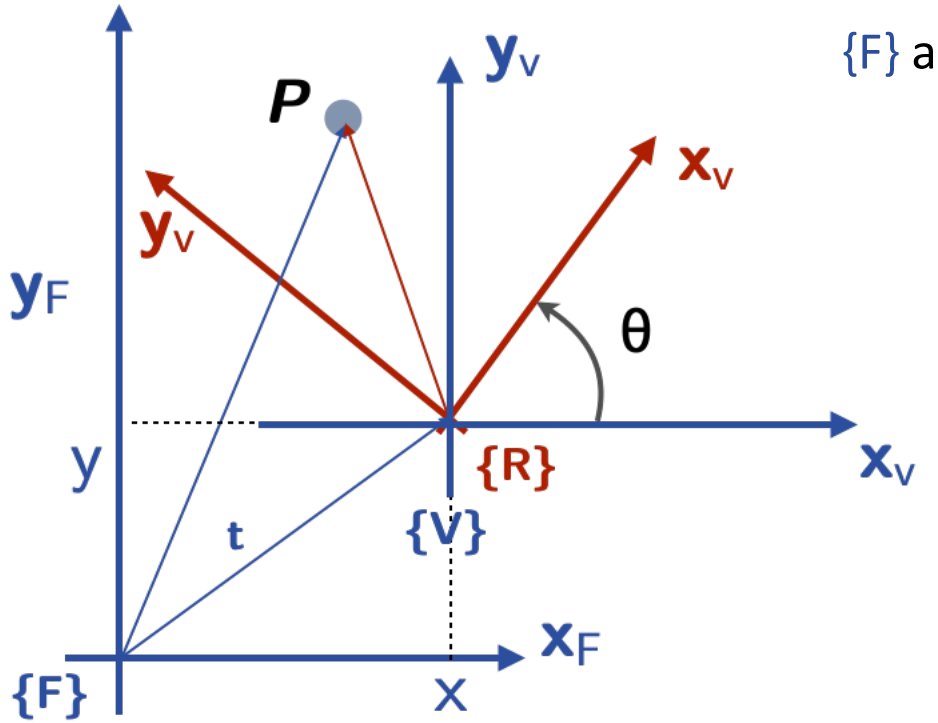
$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



We'll see this in a moment ...

Add the translation



{F} and {V} have parallel coordinate axes, we can *add* coordinates:

$$ {}^{F}\mathbf{P} = \begin{bmatrix} {}^{F}\mathrm{x} \\ {}^{F}\mathrm{y} \end{bmatrix} = \begin{bmatrix} {}^{V}\mathrm{x} \\ {}^{V}\mathrm{y} \end{bmatrix} + \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \end{bmatrix} $$

${}^{V}\mathbf{P}$ was obtained from the rotation:

$$ \begin{bmatrix} {}^{\mathrm{v}}\mathrm{x} \\ {}^{\mathrm{v}}\mathrm{y} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} {}^{\mathrm{R}}\mathrm{x} \\ {}^{\mathrm{R}}\mathrm{y} \end{bmatrix} $$

✓ In a more compact form

$$ \begin{bmatrix} {}^{F}\mathrm{x} \\ {}^{F}\mathrm{y} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & \mathrm{x} \\ \sin(\theta) & \cos(\theta) & \mathrm{y} \end{bmatrix} \begin{bmatrix} {}^{R}\mathrm{x} \\ {}^{R}\mathrm{y} \\ 1 \end{bmatrix} $$

$$ {}^{F}\mathbf{P} = \begin{bmatrix} {}^{F}\mathrm{x} \\ {}^{F}\mathrm{y} \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} {}^{R}\mathrm{x} \\ {}^{R}\mathrm{y} \end{bmatrix}}_{Rotation} + \underbrace{\begin{bmatrix} \mathrm{x} \\ \mathrm{y} \end{bmatrix}}_{Translation} $$

Homogeneous transformation matrix

$$\begin{bmatrix} {}^{F}\mathrm{x} \\ {}^{F}\mathrm{y} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & \mathrm{x} \\ \sin(\theta) & \cos(\theta) & \mathrm{y} \end{bmatrix} \begin{bmatrix} {}^{R}\mathrm{x} \\ {}^{R}\mathrm{y} \\ 1 \end{bmatrix}$$

Homogenous 2D vector!
→ allow to represent a **translation** by a **matrix product**

$$\begin{bmatrix} {}^{F}\mathrm{x} \\ {}^{F}\mathrm{y} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{F}\mathbf{R}_{\mathrm{R}} & \mathbf{t} \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} \begin{bmatrix} {}^{R}\mathrm{x} \\ {}^{R}\mathrm{y} \\ 1 \end{bmatrix}$$

Coordinate vectors for $\boldsymbol{P}$ are now 3-vectors in 2D homogenous form:
homogenous coordinate transformation

$${}^{F}\tilde{\mathbf{P}} = \begin{bmatrix} {}^{F}\mathbf{R}_{\mathrm{R}} & \mathbf{t} \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} {}^{R}\tilde{\mathbf{P}}$$

$${}^{F}\tilde{\mathbf{P}} = {}^{F}\mathbf{T}_{R}\,{}^{R}\tilde{\mathbf{P}}$$

$${}^{F}\mathbf{T}_{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & \mathrm{x} \\ \sin(\theta) & \cos(\theta) & \mathrm{y} \\ 0 & 0 & 1 \end{bmatrix}$$

The roto-translation matrix $\mathbf{T}$ is a ***homogenous transformation*** and belongs to SE(2)

The displacement of a robot / rigid body is fully described by a *homogeneous transformation matrix*

Homogeneous transformations in 2D and 3D

- Representation of Relative pose ξ of Frame {R} with respect to a Frame {F}

$$^{F}\xi_{R} \sim {}^{F}\mathbf{T}_{R}$$

✓ This representation for a relative pose is much richer than (x, y, θ) and allows to deal with angles in a natural way

- From relative pose to transformation of a coordinate point from Frame {R} to Frame {P}:

$$^{F}P = {}^{F}\xi_{R} \cdot {}^{R}P \equiv {}^{F}\mathbf{T}_{R} \cdot {}^{R}P$$

Matrix-Vector product

2D

$$^{F}\tilde{\mathbf{P}} = \begin{bmatrix} {}^{F}\mathbf{R}_{R}^{2\times 2} & \mathbf{t}_{2\times 1} \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} {}^{R}\tilde{\mathbf{P}}$$

$$^{W}\xi = [x\ y\ \theta]$$

3D

$$^{F}\tilde{\mathbf{P}} = \begin{bmatrix} {}^{F}\mathbf{R}_{R}^{3\times 3} & \mathbf{t}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} {}^{R}\tilde{\mathbf{P}}$$

$$^{W}\xi = [x\ y\ z\ \theta\ \varphi\ \psi]$$

SE( ) Group properties

$^{F}\boldsymbol{\xi}_{R} \sim {}^{F}\mathbf{T}_{R} \in \mathrm{SE}(2) \subseteq \mathbb{R}^{3\times3}$

❖ SE( ) Group properties

- Pose composition: $^{A}\boldsymbol{\xi}_{R} \oplus {}^{R}\boldsymbol{\xi}_{B} \rightarrow$ Matrix multiplication: $^{A}T_{R} \cdot {}^{R}T_{B}$

$$T_1 \oplus T_2 = T_1 T_2$$

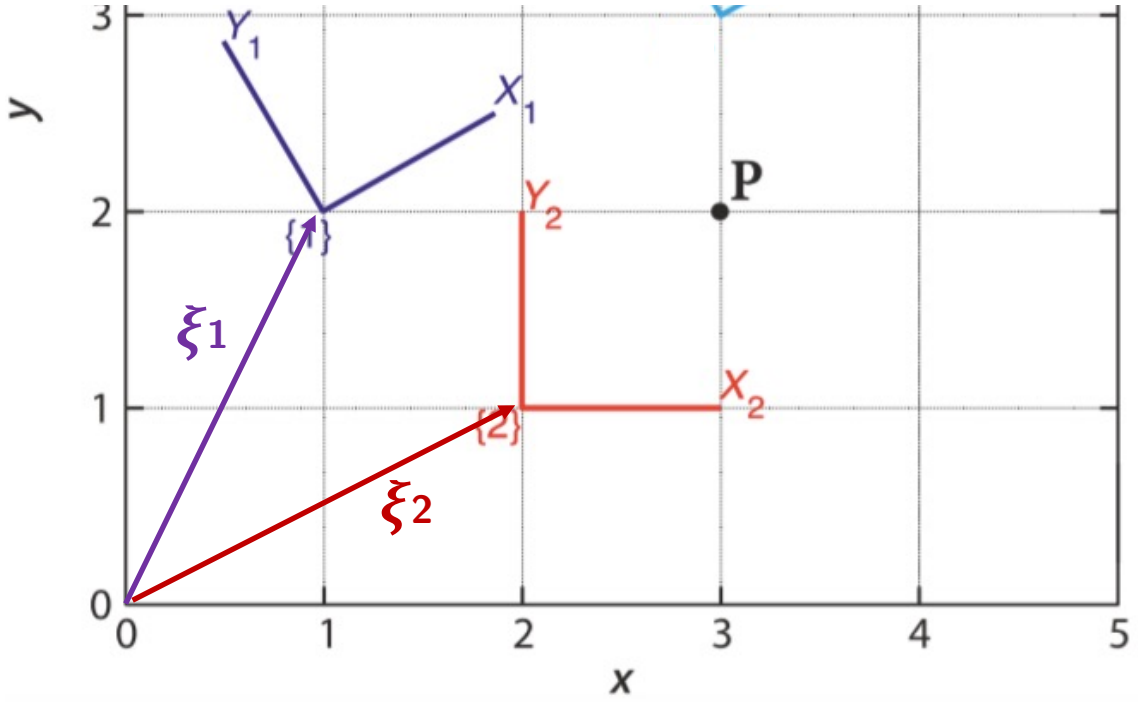
$$T = \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix}$$

$$T_1 T_2 = \begin{pmatrix} \boldsymbol{R}_1 & \boldsymbol{t}_1 \\ \mathbf{0}_{1\times2} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{R}_2 & \boldsymbol{t}_2 \\ \mathbf{0}_{1\times2} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_1\boldsymbol{R}_2 & \boldsymbol{t}_1 + \boldsymbol{R}_1\boldsymbol{t}_2 \\ \mathbf{0}_{1\times2} & 1 \end{pmatrix}$$

- Point coordinate transformation: $^{A}\widetilde{\boldsymbol{P}} = {}^{A}\boldsymbol{\xi}_{B} \cdot {}^{B}\widetilde{\boldsymbol{P}} \rightarrow$ Matrix-vector multiplication: $^{A}T_{B} \cdot {}^{B}\widetilde{\boldsymbol{P}}$
- Identity element: $\boldsymbol{\xi} \oplus \mathbf{0} = \mathbf{0} \rightarrow$ Identity matrix: $T \cdot I = T$
- Inverse: $\boldsymbol{\xi} \ominus \boldsymbol{\xi} = \mathbf{0} \rightarrow$ Inverse matrix: $T \cdot T^{-1} = I$

$$T^{-1} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0}_{1\times2} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{t} \\ \mathbf{0}_{1\times2} & 1 \end{pmatrix}$$

Pose transformation example, no commutativity



$\xi_0 \sim (0,0,0)$

$\xi_1 = \xi_0 \oplus (1,\ 2,\ 30^o)$

$\xi_2 = \xi_0 \oplus (2,1,0^o)$

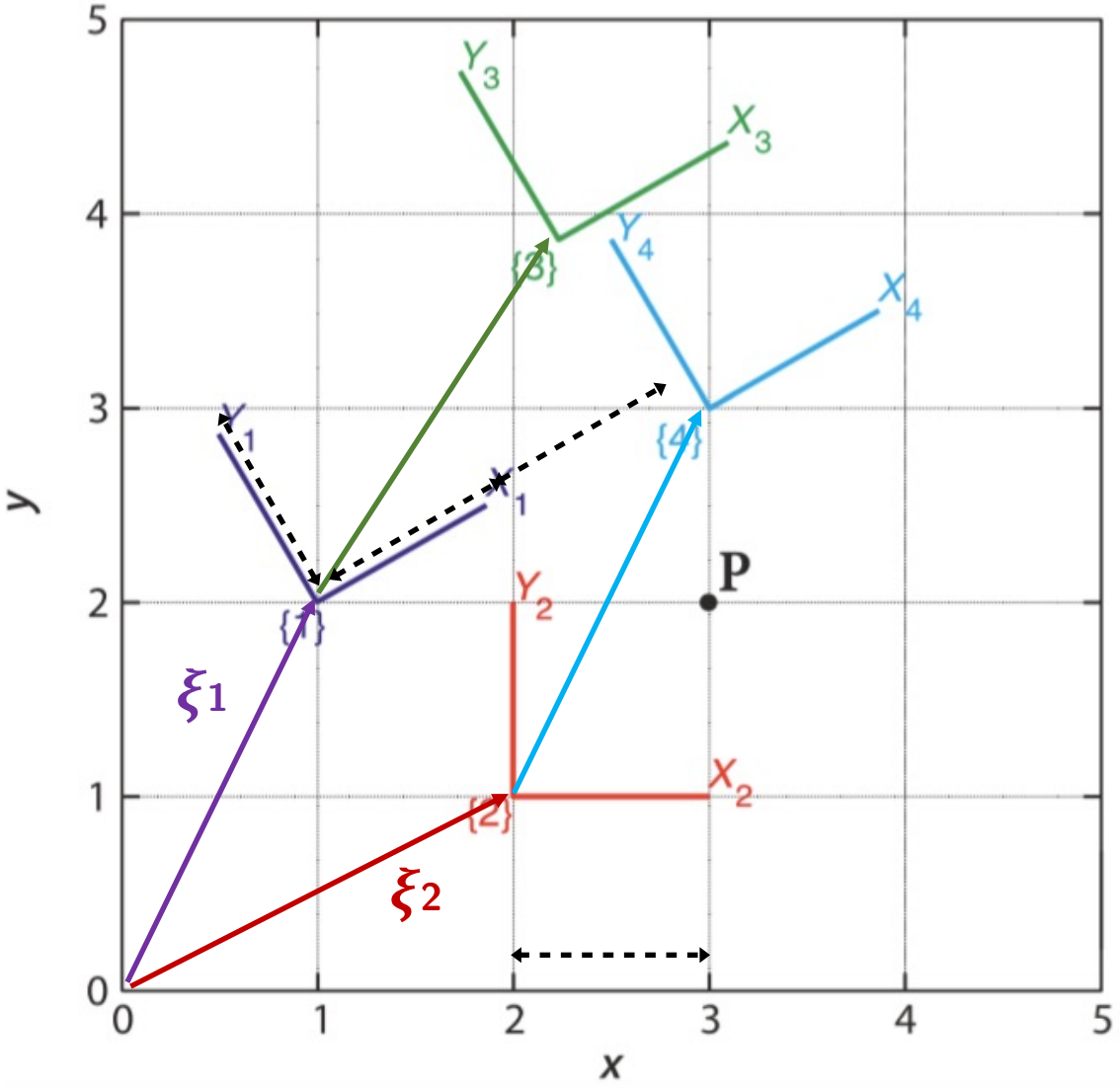
$$T_1 = \begin{bmatrix} 0.866 & -0.5 & 1 \\ 0.5 & 0.866 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

? $\xi_3 = \xi_1 \oplus \xi_2$

? $\xi_4 = \xi_2 \oplus \xi_1$

Pose transformation example, no commutativity



$$\xi_3 = \xi_1 \oplus \xi_2$$

- ξ_2 represents a transformation T_2 consisting of a translation of 2 long *x*, 1 long *y*, plus a counterclockwise rotation of 0 degrees
- $\xi_3 = T_1 T_2$ meaning that T_2 is applied wrt T_1 : the origin of {3} has coordinates (2,1) in {1} and no further rotation is applied to the coordinated axes

$$\xi_4 = \xi_2 \oplus \xi_1$$

- $\xi_4 = T_2 T_1$ meaning that T_1 is applied wrt T_2 : the origin of {4} has coordinates (1,2) in {2} and is rotated of 30 degrees wrt {2}'s coordinate axes

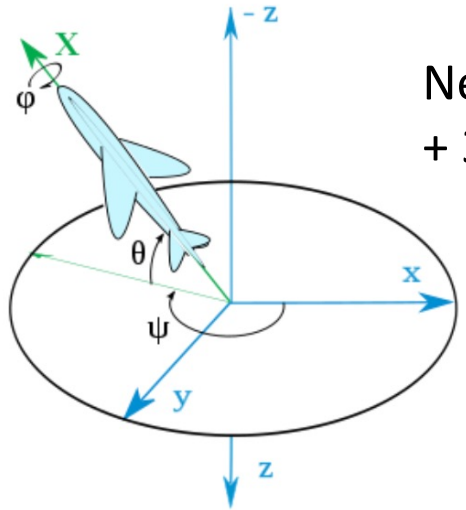
Relative Poses in 3D? → 4×4 Homogeneous transformation matrices

$$ {}^{F}\tilde{\mathbf{P}} = \begin{bmatrix} {}^{F}\mathbf{R}_{R}^{2\times 2} & \mathbf{t}_{2\times 1} \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} {}^{R}\tilde{\mathbf{P}} $$

$$ {}^{F}\tilde{\mathbf{P}} = \begin{bmatrix} {}^{F}\mathbf{R}_{R}^{3\times 3} & \mathbf{t}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} {}^{R}\tilde{\mathbf{P}} $$

$$ {}^{W}\boldsymbol{\xi} = [x\ y\ \theta] $$

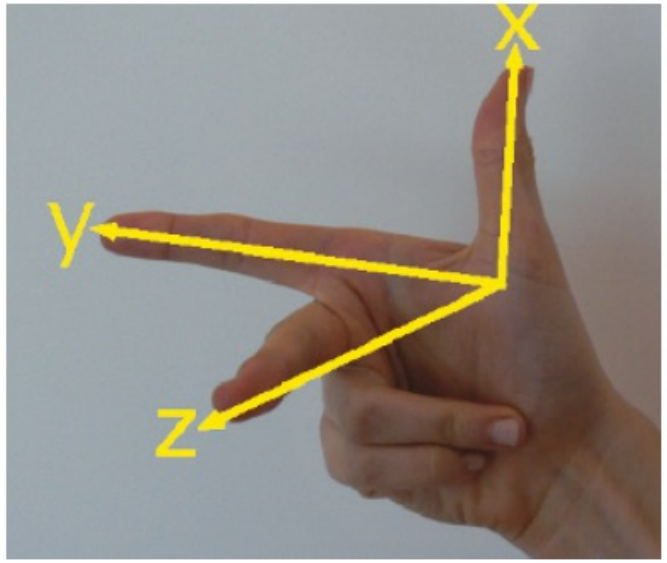
$$ {}^{W}\boldsymbol{\xi} = [x\ y\ z\ \theta\ \varphi\ \psi] $$



Need 3 angles for orientation
+ 3 coordinates for positions

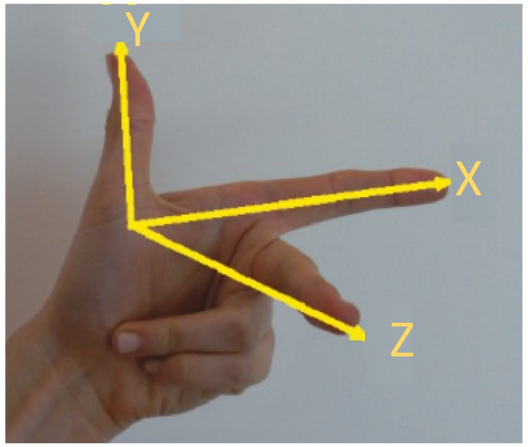
We'll go through the following slides in the next lecture ....

Relative orientation of coordinate axes in 3D



Right-hand rule for the relative orientation of the coordinate axes

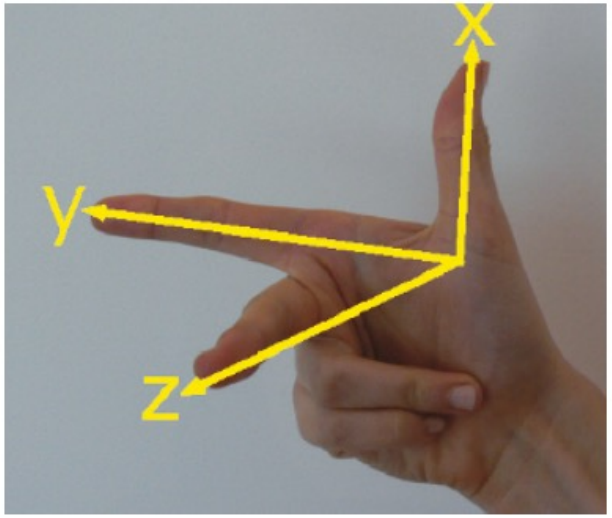
Positive rotation is **counterclockwise** about the axis of rotation.



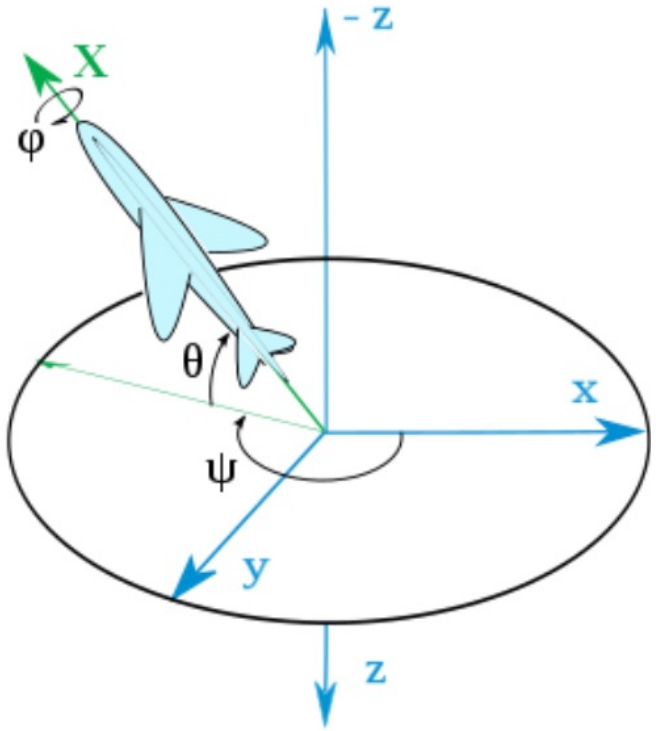
Left-hand rule for the relative orientation of the coordinate axes

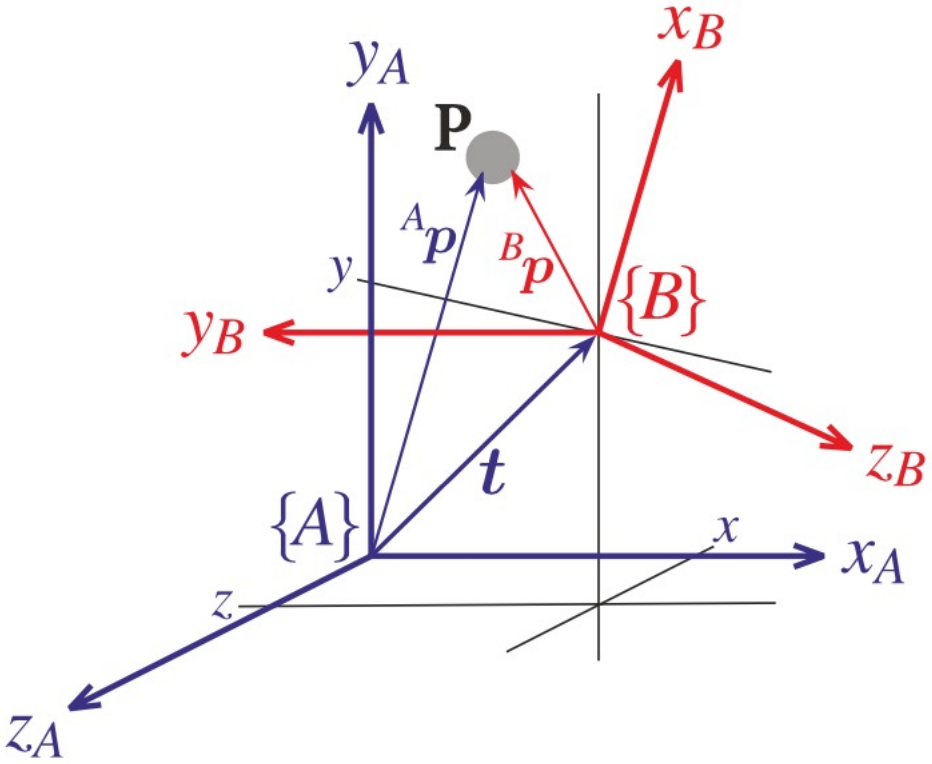
Positive rotation is **clockwise** about the axis of rotation.

Orientation of a rigid body in 3D



Right-hand rule for the relative orientation of the coordinate axes

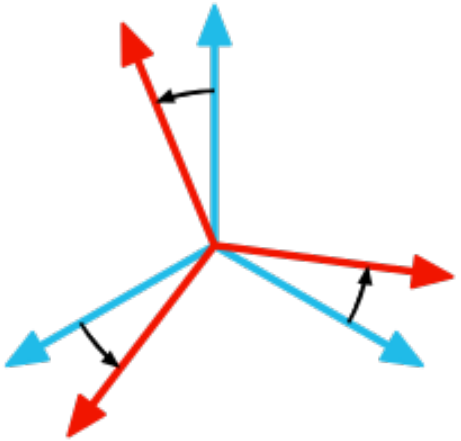


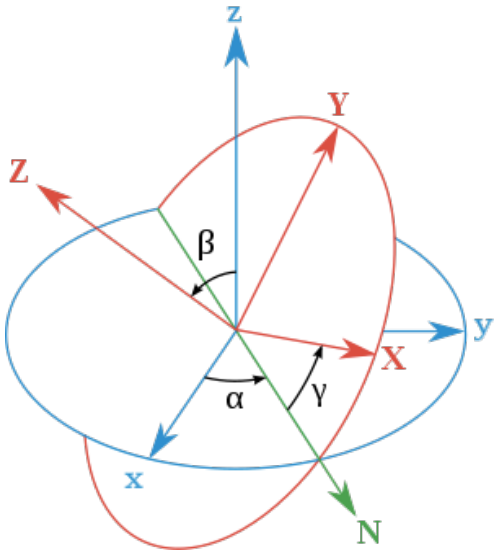


Transformation between coordinate frames: Euler's Theorem

- **Euler's Theorem**: Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than **three**) about coordinate axes, where no two successive rotations may be about the same axis.

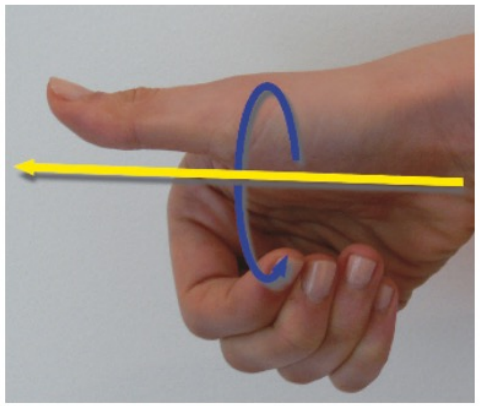
  - Any target orientation can be reached by composing three ***elemental*** rotations around the three coordinate axes



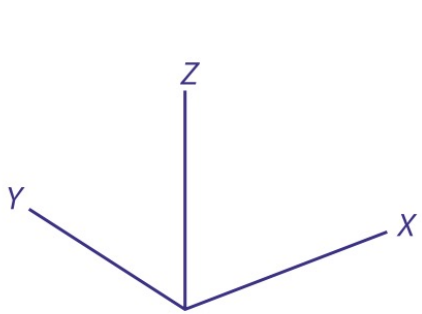


In 3-dimensions rotation is not commutative – the order in which rotations are applied makes a difference to the result.

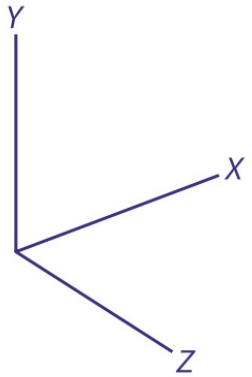
Transformation between coordinate frames: Euler's Theorem



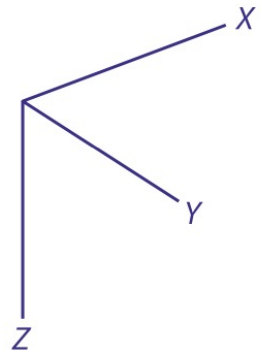
Rotation about a vector. Wrap your right hand around the vector with your thumb (your x-finger) in the direction of the arrow. The curl of your fingers indicates the direction of increasing angle.



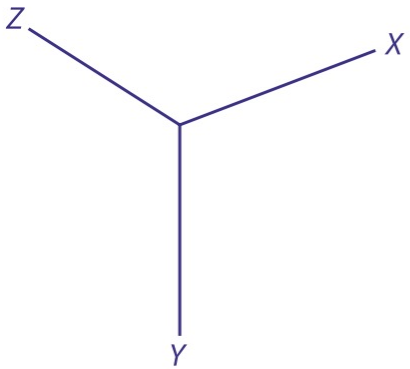
a Original



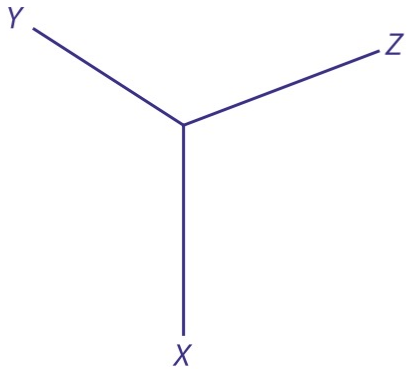
b $\frac{\pi}{2}$ about x-axis



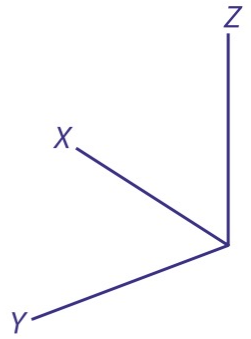
c π about x-axis



d $-\frac{\pi}{2}$ about x-axis



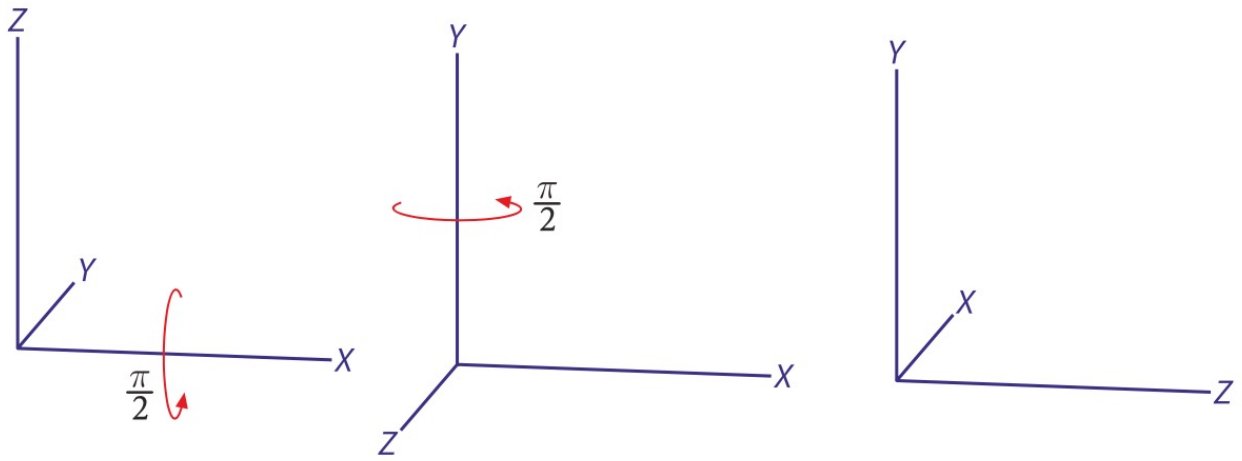
e $\frac{\pi}{2}$ about y-axis



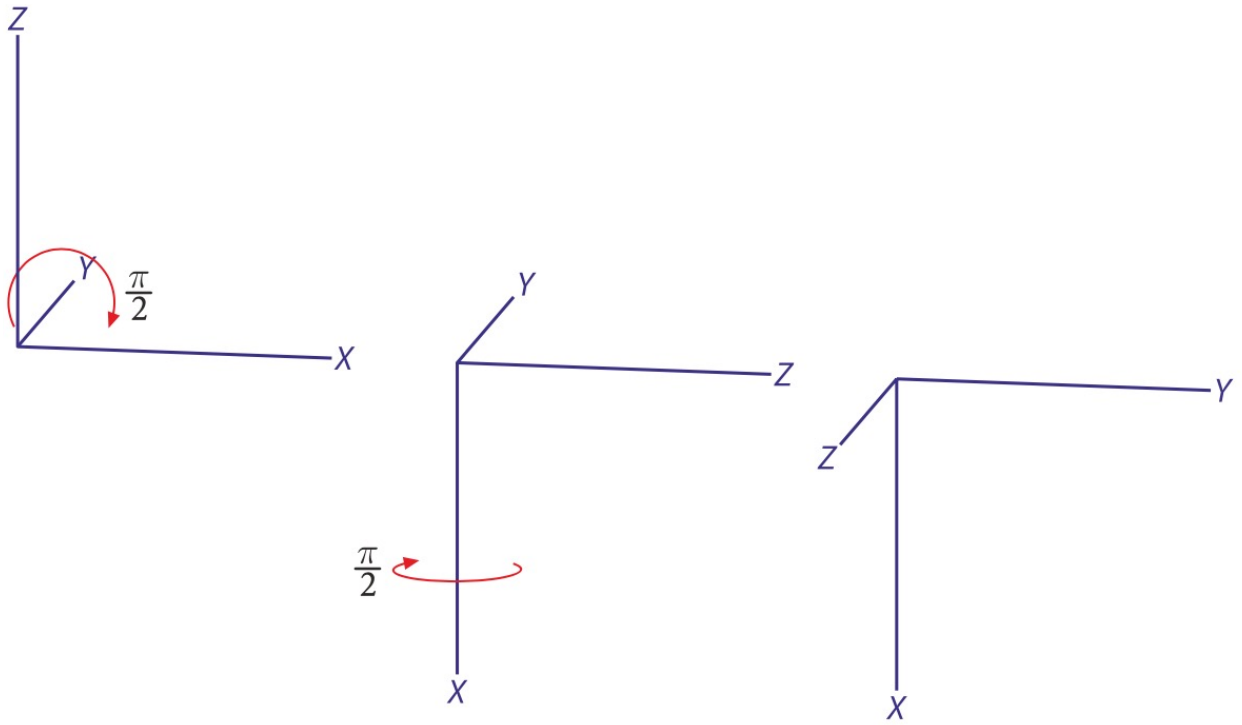
f $\frac{\pi}{2}$ about z-axis

Non-commutativity

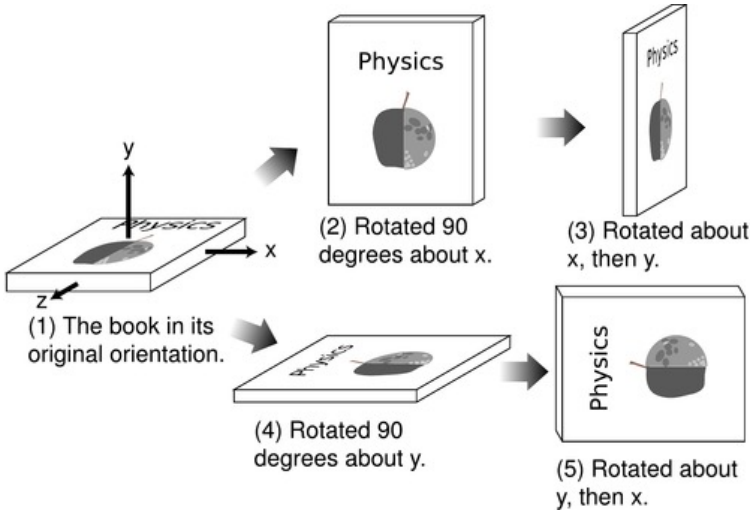
$X_{90} \oplus Y_{90}$

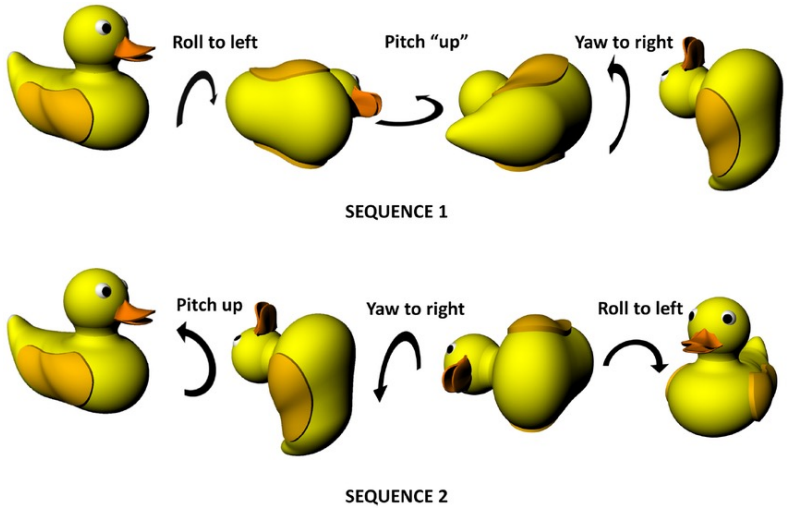


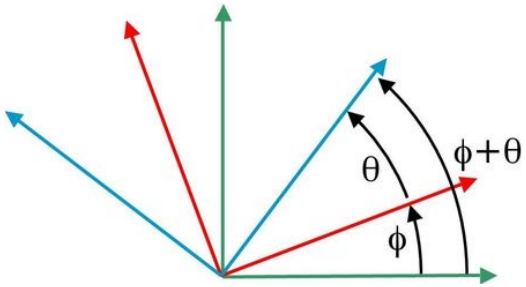
$Y_{90} \oplus X_{90}$



Rotations are not commutative in more than 2 dimensions



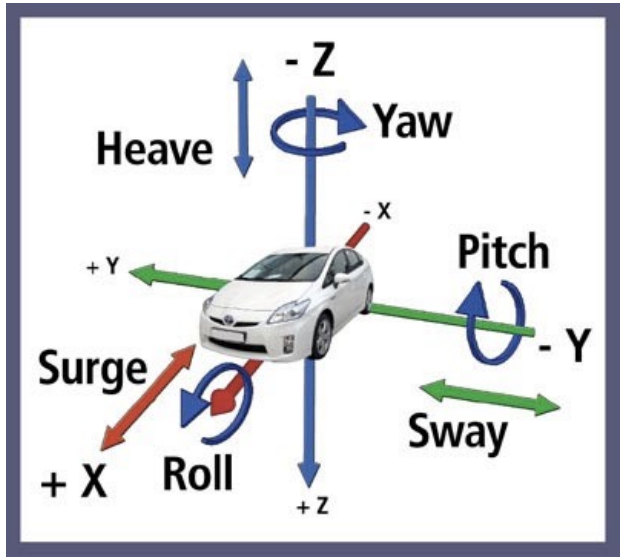


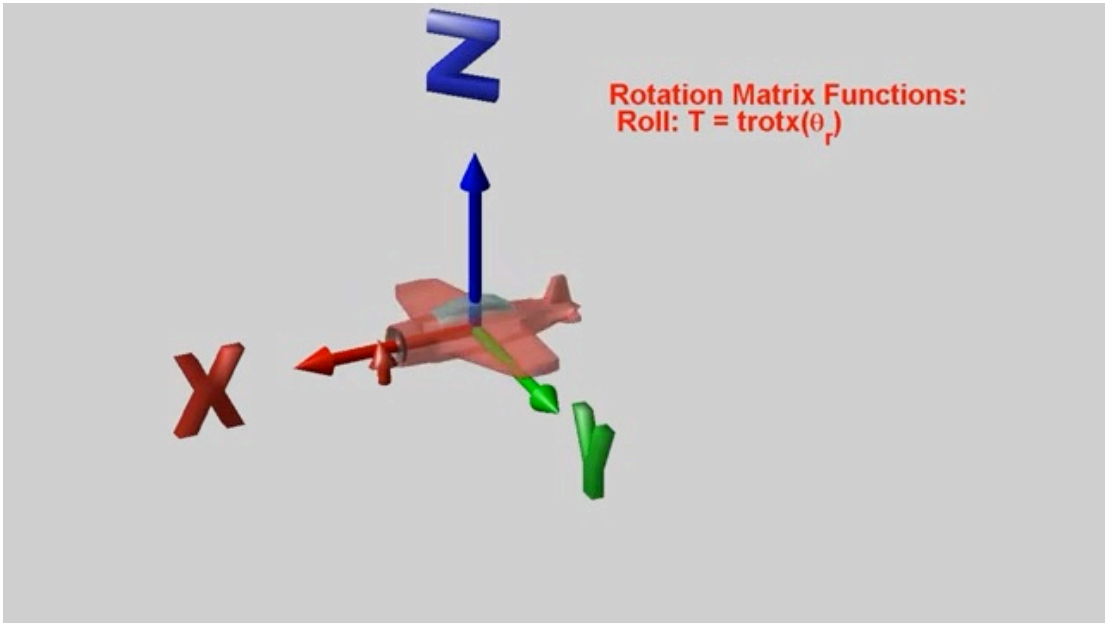


$$R_x(\frac{\pi}{4}) \cdot R_y(\frac{\pi}{4}) = \begin{bmatrix} 0.707 & 0 & -0.707 \\ -0.5 & 0.707 & -0.5 \\ 0.5 & 0.707 & 0.5 \end{bmatrix}, \; R_x(\frac{\pi}{4}) \cdot R_y(\frac{\pi}{4}) \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -1.414 \\ 0.586 \\ 3.414 \end{bmatrix}$$

$$R_y(\frac{\pi}{4}) \cdot R_x(\frac{\pi}{4}) = \begin{bmatrix} 0.707 & -0.5 & -0.5 \\ 0 & 0.707 & -0.707 \\ 0.707 & 0.5 & 0.5 \end{bmatrix}, \; R_y(\frac{\pi}{4}) \cdot R_x(\frac{\pi}{4}) \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -1.793 \\ 0.707 \\ 3.207 \end{bmatrix}$$

3D angles


- Z
Heave
Yaw
- X
+ Y
Pitch
- Y
Surge
Sway
+ X
Roll
+ Z

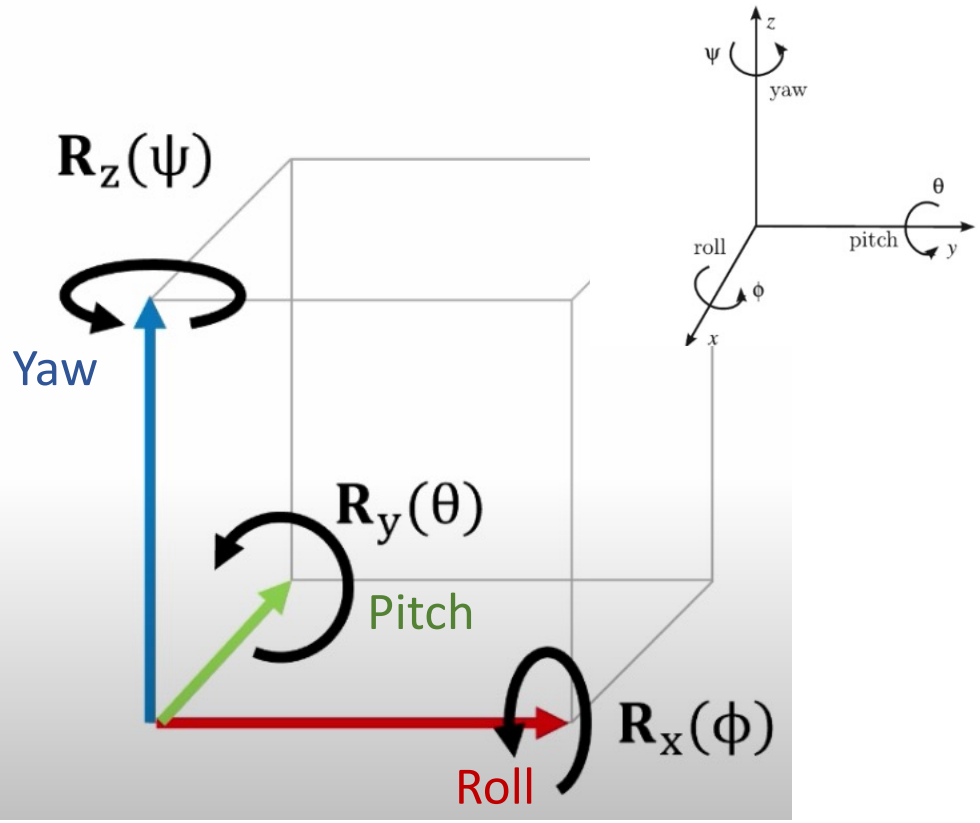

Z
Rotation Matrix Functions:
Roll: T = trotx(θr)
X
Y

3D rotations about the axes (elementary rotations)

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

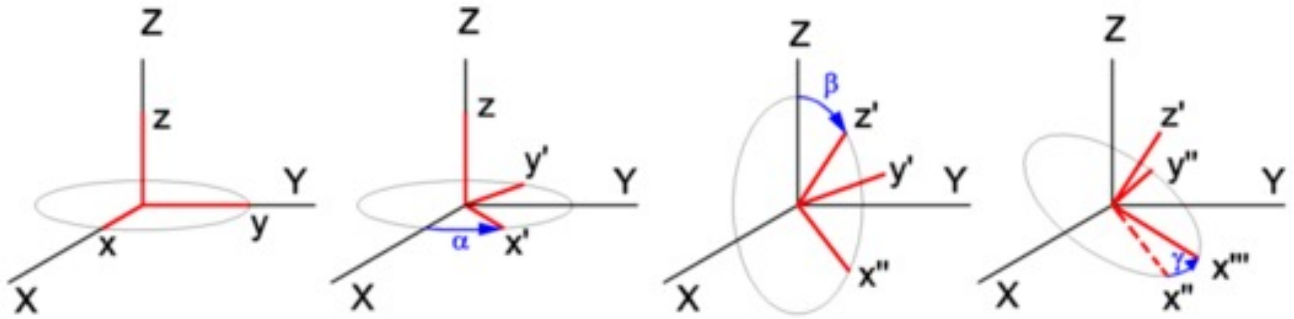
$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Rotations in 3D - SO(3)

Sequences of elemental rotations: Eulerian and Cardanian

- Euler's rotation theorem requires successive rotation about three axes such that no two successive rotations are about the same axis.

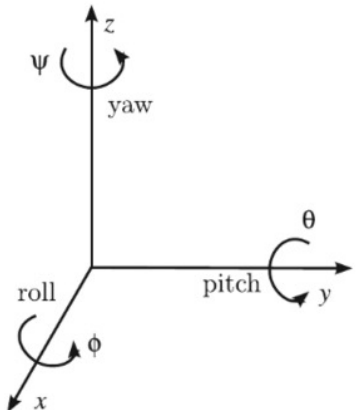


It is common practice to refer to all 3-angle representations as Euler angles but this is underspecified since there are twelve different types to choose from. The particular angle sequence is often a convention within a particular technological field.

- There are two classes of rotation sequence: **Eulerian** and **Cardanian**
  - The **Eulerian** type involves repetition, but not successive, of rotations about one particular axis: XYX, XZX, YXY, YZY, ZXZ, ZYZ.
  - The **Cardanian** type is characterized by rotations about all three axes: XYZ, XZY, YZX, YXZ, ZXY, ZYX.

Homogeneous transformations in 3D

Sequence matter! R-P-Y (Roll-Pitch-Yaw) Cardanian sequence



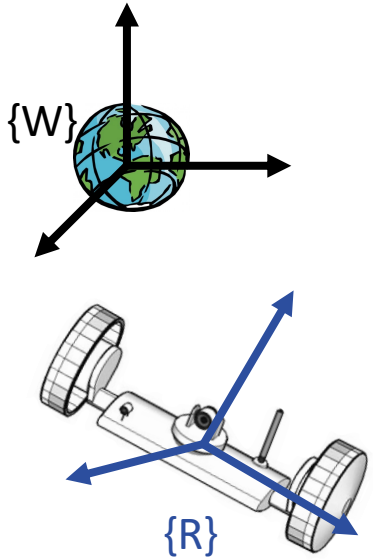
$$\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) = \begin{pmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta s_\phi + c_\psi c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix}$$

c_x := cos(x), s_x := sin(x)

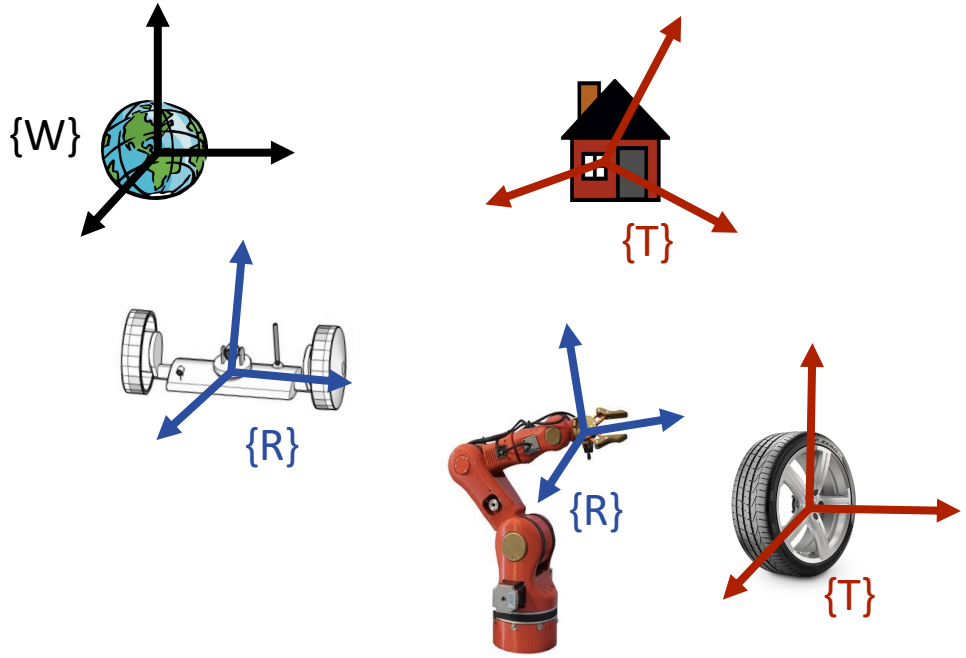
$$T = \begin{pmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi & t_x \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi & t_y \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Now we can deal with these problems….

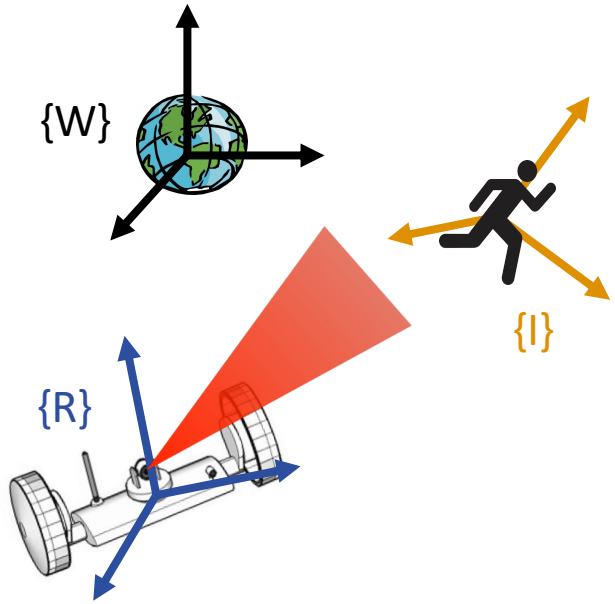
Where's the robot? What is robot's pose with respect to the world reference frame {W}?



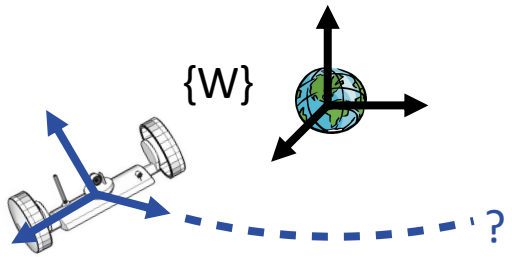
Robot's pose with respect to the external frame {T} (e.g., a target) ?



What is intruder's pose, observed using robot's lateral camera (local frame), in the world frame {W}?



Predict: What is robot's pose in {W} after moving at a velocity v for 1 minute?



Plan: What is the velocity profile that allows to reach a pose ξ in {W}?

Calculations….

- A point *p* expressed in the local (robot) coordinate frame is transformed in {W} coordinates

$$f_{\mathbf{xw}}(\mathbf{p}) = \begin{pmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix} \mathbf{p} + \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{R}\mathbf{p} + \mathbf{t}$$

- A rigid body 3D pose in {W} after applying a homogeneous transformation **T**

$\boldsymbol{\xi}_A = \boldsymbol{\xi}_W \oplus (x,y,z,\phi,\theta,\psi)$, where **T** is the transformation representing (x,y,z,ϕ,θ,ψ)

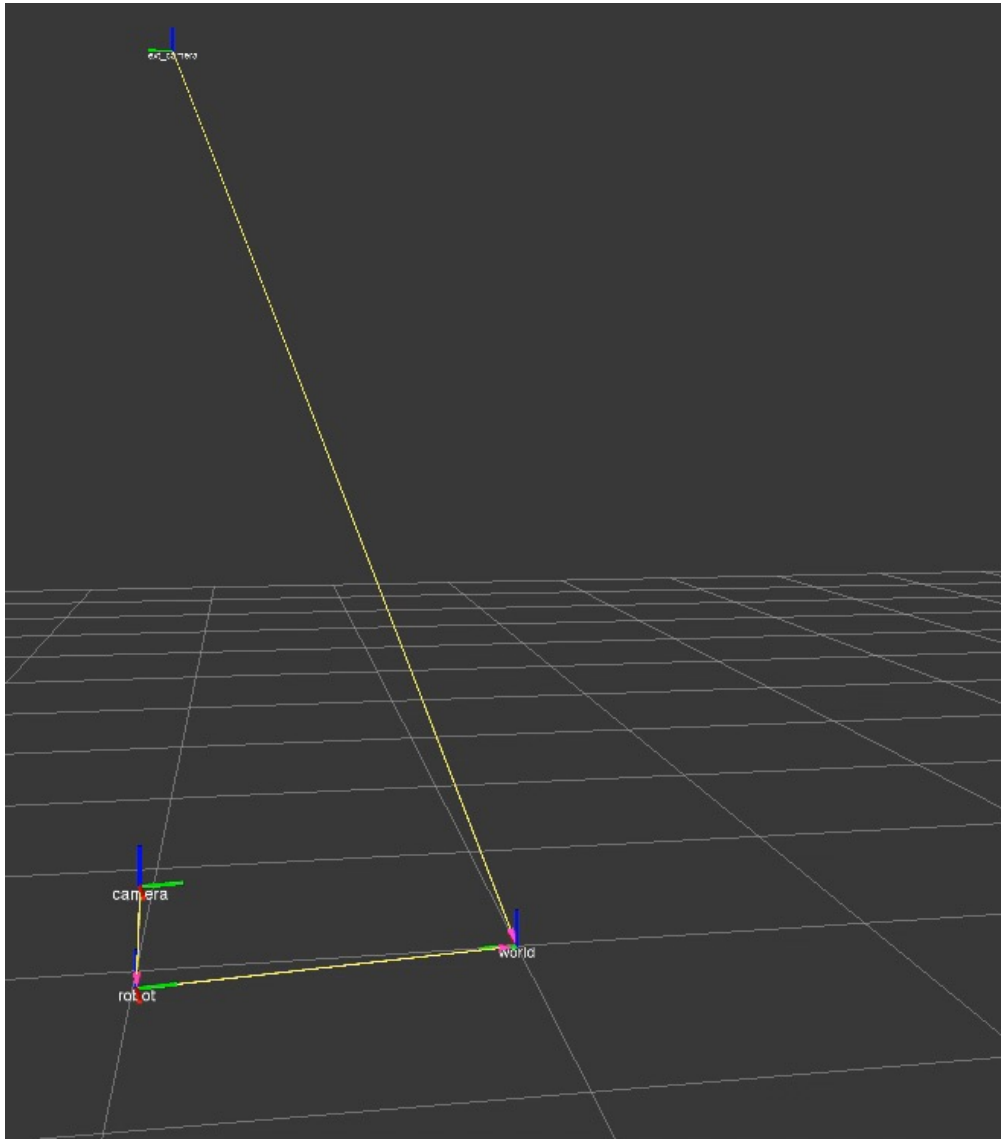
$$\mathbf{T} := \begin{pmatrix} t_{1,1} & t_{1,2} & t_{1,3} & t_{1,4} \\ t_{2,1} & t_{2,2} & t_{2,3} & t_{2,4} \\ t_{3,1} & t_{3,2} & t_{3,3} & t_{3,4} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi & t_x \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi & t_y \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

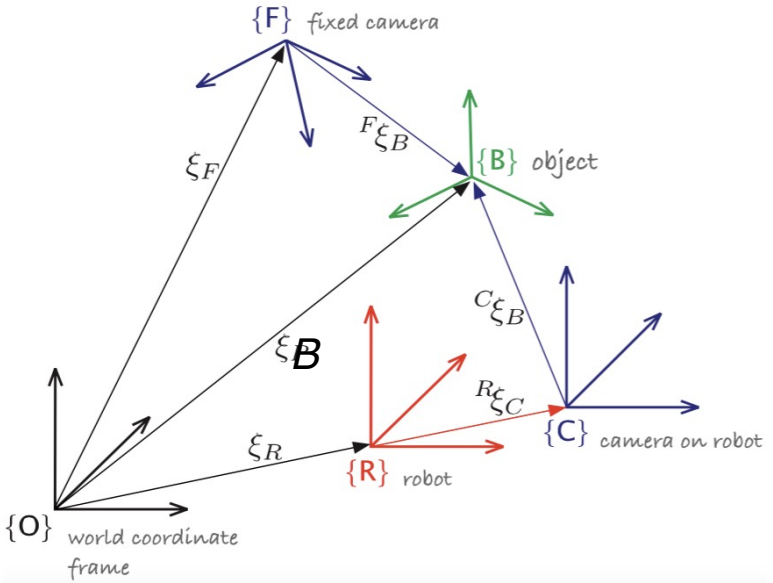
$$\boldsymbol{\xi}_A(\mathbf{T}) \sim \begin{pmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{pmatrix}$$

$$\begin{aligned} x &= t_{1,4} \\ y &= t_{2,4} \\ z &= t_{3,4} \\ \phi &= \operatorname{atan2}(t_{2,1}, t_{1,1}) \\ \theta &= \operatorname{atan2}(-t_{3,1}, \cos(\phi)\, t_{1,1} + \sin(\phi)\, t_{2,1}) \\ \psi &= \operatorname{atan2}(\sin(\phi)\, t_{1,3} - \cos(\phi)\, t_{2,3}, -\sin(\phi)\, t_{1,2} + \cos(\phi)\, t_{2,2}) \end{aligned}$$

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \pi + \arctan(\frac{y}{x}) & y \geq 0, x < 0 \\ -\pi + \arctan(\frac{y}{x}) & y < 0, x < 0 \\ \frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}.$$

ROS helps!


camera
world
robot


{F} fixed camera
ξ_F
$^F\xi_B$
{B} object
$^C\xi_B$
ξ_B
$^R\xi_C$
{C} camera on robot
ξ_R
{R} robot
{O} world coordinate frame

ROS helps!

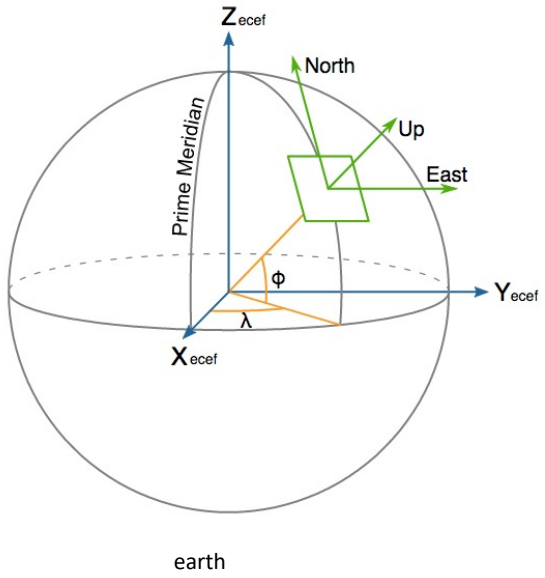
```cpp
#include <stdio.h>
#include <math.h>
#include <ros/ros.h>
#include <tf/transform_broadcaster.h>
#include <tf/transform_listener.h>

int main (int argc, char** argv)
{
  ros::init(argc, argv, "tf_example");

  ros::NodeHandle node;

  // to publish and read the topics regarding frame transformation
  tf::TransformListener tf_ls;
  tf::TransformBroadcaster tf_br;

  // transformation objects for the robot, the camera on board of the robot,
  // the external camera
  tf::Transform robot_tf;
  tf::Transform camera_tf;
  tf::Transform ext_camera_tf;


  // set a displacement (translation + rotation) for external camera
  ext_camera_tf.setOrigin(tf::Vector3(3.0, 1.0, 3.0));
  ext_camera_tf.setRotation(tf::Quaternion(0.0, 0.0, 0.0, 1.0));

  // set a displacement (translation + rotation) for external camera
  camera_tf.setOrigin(tf::Vector3(0.15, 0.0, 0.30));
  camera_tf.setRotation(tf::Quaternion(0.0, 0.0, 0.0, 1.0));

  // robot start position, it will then follow a circular trajectory
  float R = 1.0;
  float x = 0.0;
  float y = 0.0;
  float phi = 0.0;

  // orientation for robot trajectory
  tf::Quaternion robot_ori;
  tf::Vector3 pt_on_robot(0.0, 0.0, 0.0);


  // where to store/publish time-stamped information about frame transformation
  tf::StampedTransform relative_pose_tf;
```

```cpp
// main control cycle, with a defined frequency
ros::Rate loop_rate(100);
while(ros::ok())
{
  // set robot's pose to follow a circular trajectory
  x = R * cos(phi);
  y = R * sin(phi);
  phi += 0.01;
  ROS_INFO("Ground truth: %3.2f, %3.2f, %3.2f", x, y, 0.0);

  // to make the robot curving on the trajectory
  robot_ori.setRPY(0.0, 0.0, phi + M_PI / 2.0);

  // robot's relative pose
  robot_tf.setOrigin(tf::Vector3(x, y, 0.0));
  robot_tf.setRotation(robot_ori);

  ros::Time stamp = ros::Time::now();

  // parent-child relations between the frames attached to the considered entities
  tf_br.sendTransform(tf::StampedTransform(robot_tf, stamp, "/world", "/robot"));
  tf_br.sendTransform(tf::StampedTransform(camera_tf, stamp, "/robot", "/camera"));
  tf_br.sendTransform(tf::StampedTransform(ext_camera_tf, stamp, "/world", "/ext_ca

  try
  {
    // computing the homogeneous transformation from robot frame to the external ca
    tf_ls.lookupTransform("/ext_camera", "/robot", ros::Time(0), relative_pose_tf);

    // a point in the robot frame is transformed in a point in the external camera
    tf::Vector3 pt_in_ext_camera = relative_pose_tf * pt_on_robot;
    // the point the external camera frame is transformed in a point in the world f
    tf::Vector3 pt_in_world = ext_camera_tf * pt_in_ext_camera;

    ROS_INFO("Point in /ext_camera: %3.2f, %3.2f, %3.2f",
             pt_in_ext_camera.getX(), pt_in_ext_camera.getY(), pt_in_ext_camera.get
    ROS_INFO("Point in /world: %3.2f, %3.2f, %3.2f",
             pt_in_world.getX(), pt_in_world.getY(), pt_in_world.getZ());
  }
  catch(tf::TransformException &e)
  {
    ROS_ERROR("%s", e.what());
    ros::Duration(1.0).sleep();
    continue;
  }
  loop_rate.sleep();
  ros::spinOnce();
}
return 0;
```

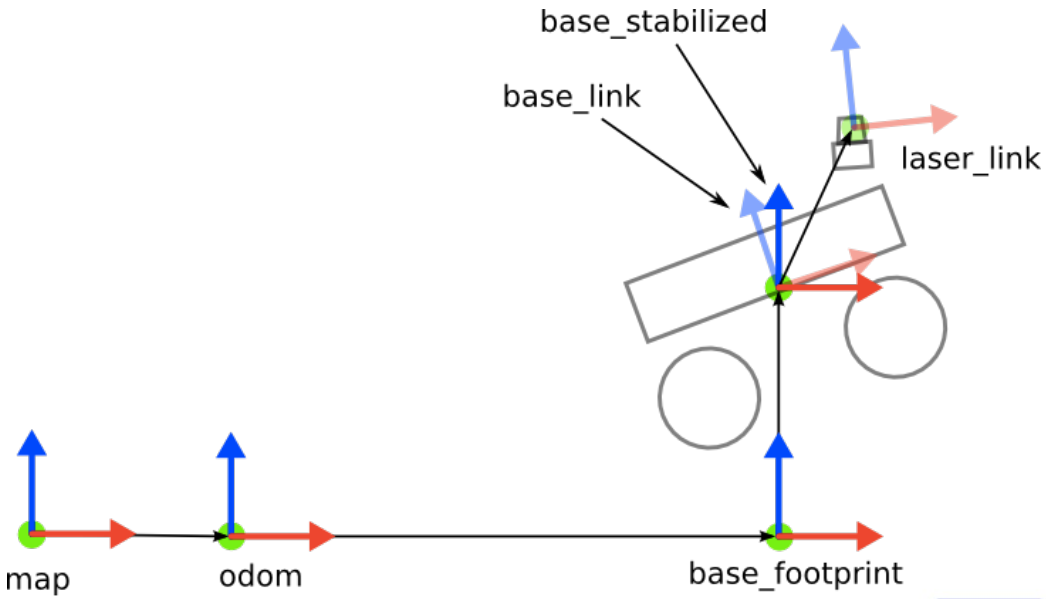
ROS helps!

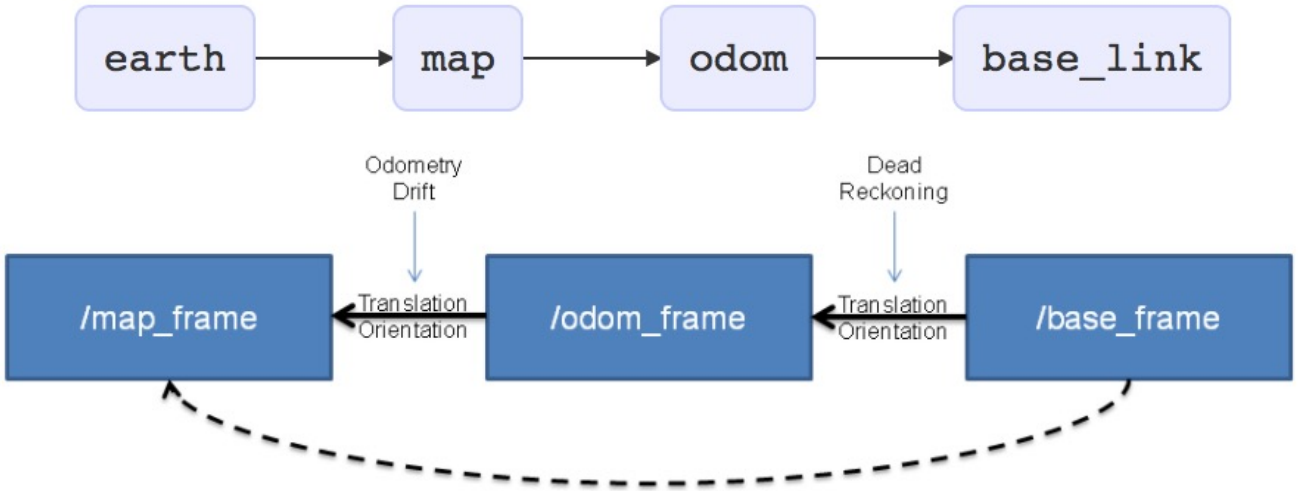
```
  // main control cycle, with a defined frequency
  ros::Rate loop_rate(100);
  while(ros::ok())
  {
    // set robot's pose to follow a circular trajectory
    x = R * cos(phi);
    y = R * sin(phi);
    phi += 0.01;
    ROS_INFO("Ground truth: %3.2f, %3.2f, %3.2f", x, y, 0.0);

    // to make the robot curving on the trajectory
    robot_ori.setRPY(0.0, 0.0, phi + M_PI / 2.0);

    // robot's relative pose
    robot_tf.setOrigin(tf::Vector3(x, y, 0.0));
    robot_tf.setRotation(robot_ori);

    ros::Time stamp = ros::Time::now();

    // parent-child relations between the frames attached to the considered entities are defined
    tf_br.sendTransform(tf::StampedTransform(robot_tf, stamp, "/world", "/robot"));
    tf_br.sendTransform(tf::StampedTransform(camera_tf, stamp, "/robot", "/camera"));
    tf_br.sendTransform(tf::StampedTransform(ext_camera_tf, stamp, "/world", "/ext_camera"));

    try
    {
      // computing the homogeneous transformation from robot frame to the external camera frame
      tf_ls.lookupTransform("/ext_camera", "/robot", ros::Time(0), relative_pose_tf);

      // a point in the robot frame is transformed in a point in the external camera frame
      tf::Vector3 pt_in_ext_camera = relative_pose_tf * pt_on_robot;
      // the point the external camera frame is transformed in a point in the world frame
      tf::Vector3 pt_in_world = ext_camera_tf * pt_in_ext_camera;

      ROS_INFO("Point in /ext_camera: %3.2f, %3.2f, %3.2f",
               pt_in_ext_camera.getX(), pt_in_ext_camera.getY(), pt_in_ext_camera.getZ());
      ROS_INFO("Point in /world: %3.2f, %3.2f, %3.2f",
               pt_in_world.getX(), pt_in_world.getY(), pt_in_world.getZ());
    }
    catch(tf::TransformException &e)
    {
      ROS_ERROR("%s", e.what());
      ros::Duration(1.0).sleep();
      continue;
    }
    loop_rate.sleep();
    ros::spinOnce();
  }
  return 0;
}
```

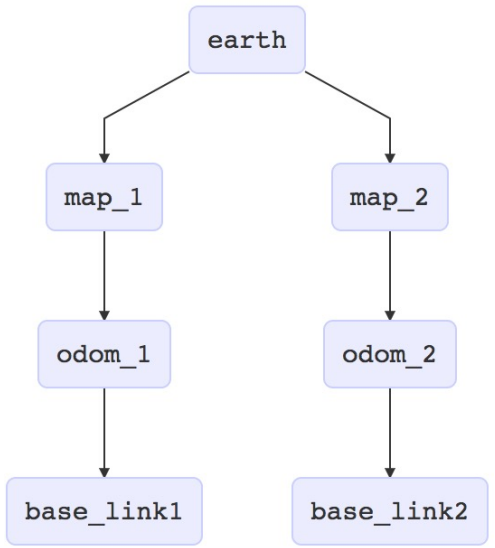
In which frame poses / odometry measures are represented



earth

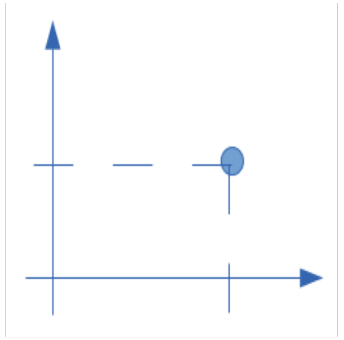




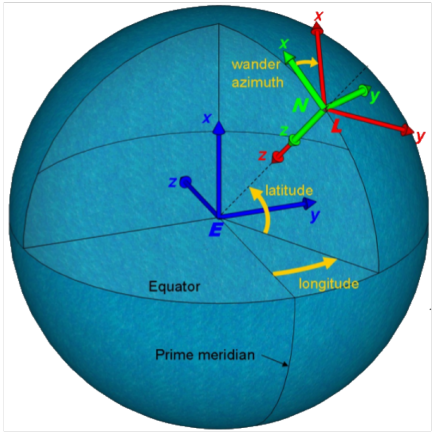


Maps….

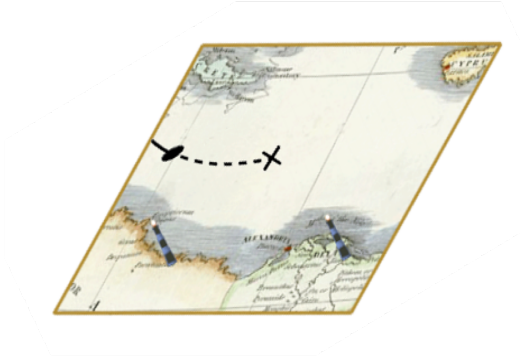
Cartesian, 2D

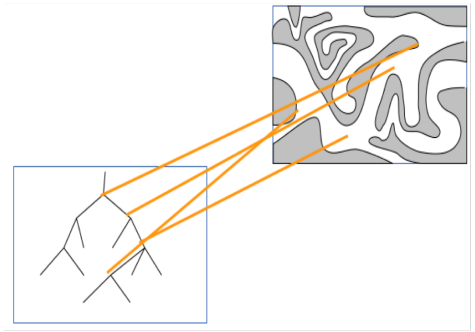


3D earth coordinates

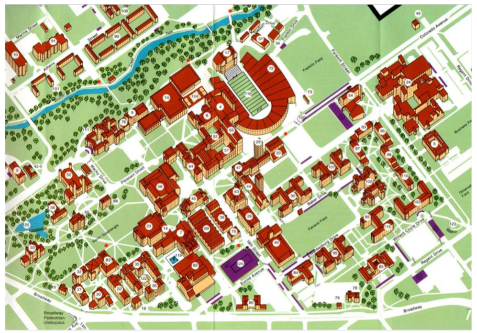


Geographical map, landmarks

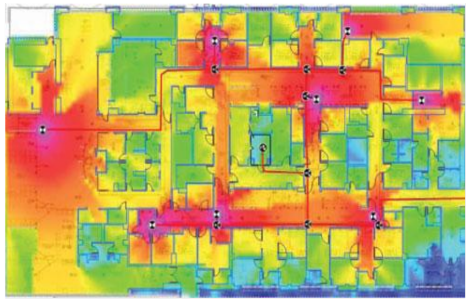




Toplogical map



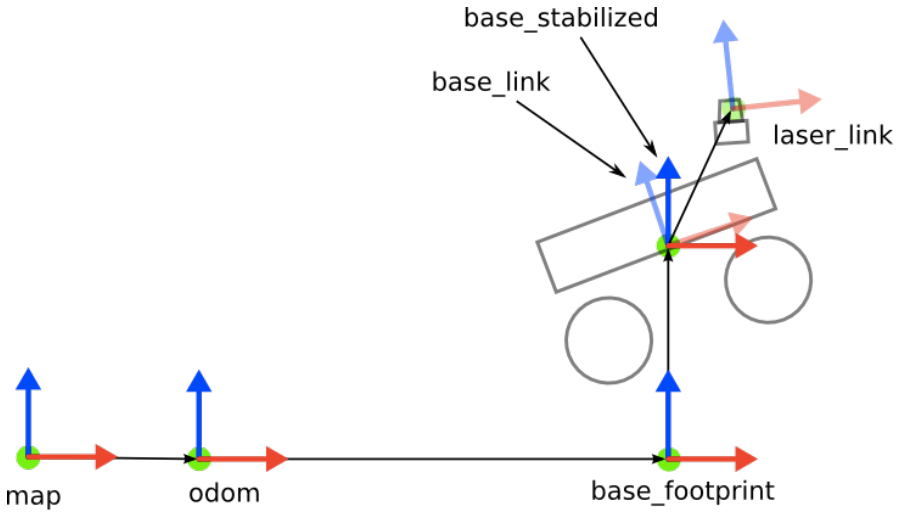
Metric map



Sensor map

Basic Ros frames (http://www.ros.org/reps/rep-0105.html)

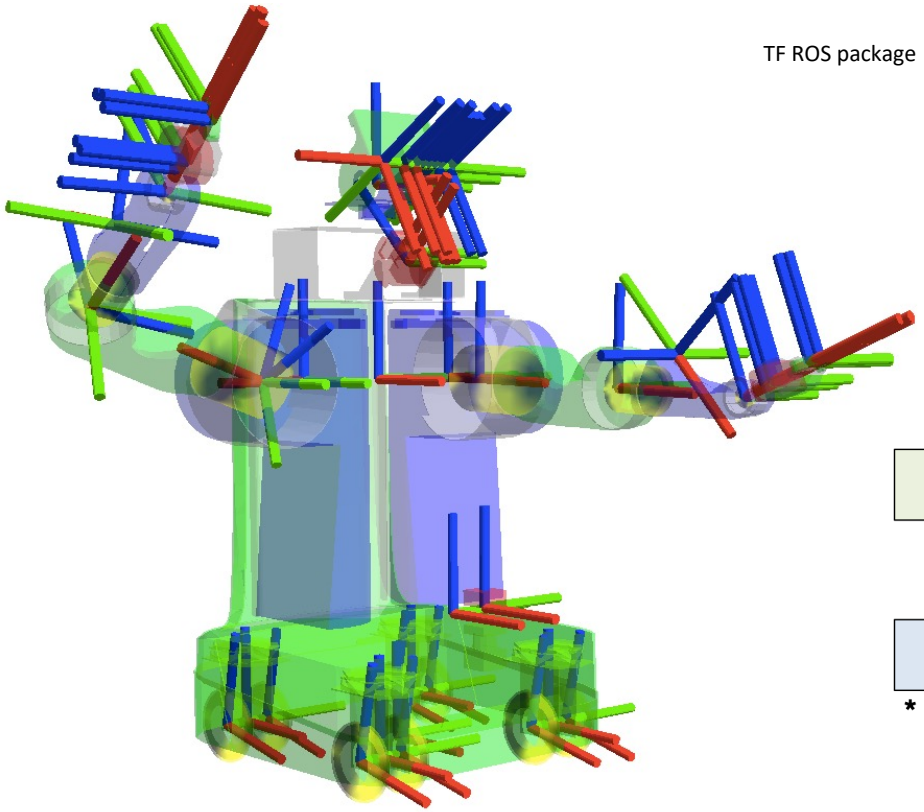
- **base_link** is rigidly attached to the mobile robot base. It can be attached to the base in any arbitrary position or orientation
- **odom** is a world-fixed frame. The pose of a mobile platform in the odom frame can drift over time, without any bounds. This drift makes the odom frame useless as a long-term global reference.
- The pose of a robot in the odom frame is guaranteed to be continuous, meaning that the pose of a mobile platform in the odom frame always evolves in a smooth way, without discrete jumps.
- In a typical setup the odom frame is computed based on an odometry source, such as wheel odometry, visual odometry or an IMU



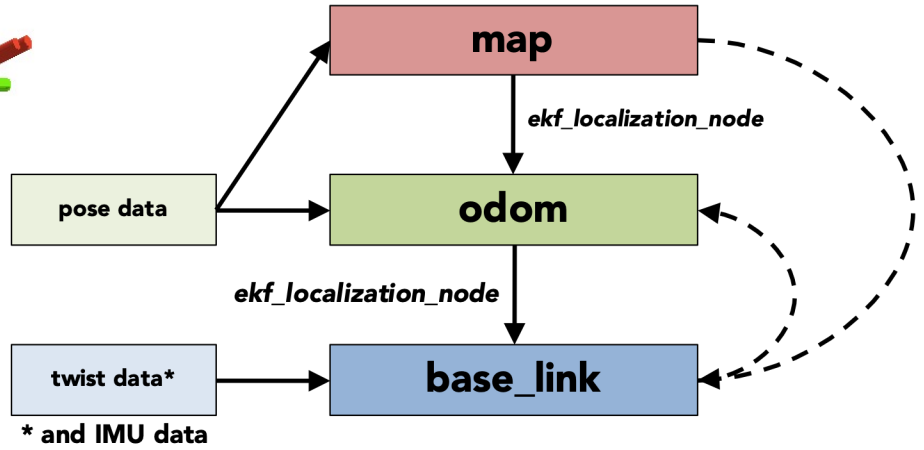
- **map** is a world fixed frame, with its Z-axis pointing upwards. The pose of a mobile platform, relative to the map frame, should not significantly drift over time. The map frame is not continuous, meaning the pose of a mobile platform in the map frame can change in discrete jumps at any time.
- In a typical setup, a localization component constantly re-computes the robot pose in the map frame based on sensor observations, therefore eliminating drift, but causing discrete jumps when new sensor information arrives. The map frame is useful as a long-term global reference, but discrete jumps in position estimators make it a poor reference frame for local sensing and acting.

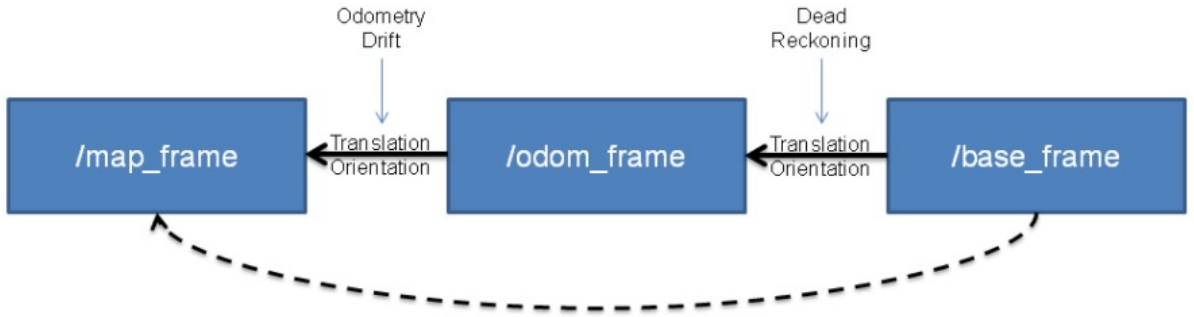
- **earth** is the origin of ECEF. This frame is designed to allow the interaction of multiple robots in different map frames. If the application only needs one map the earth coordinate frame is not expected to be present.

Transformation between frames



TF ROS package





Important concepts to take home

- Robot pose (positions + orientations) and its importance / use
- Representation and composition of (relative poses)
- Poses as an additive group
- Representation of relative poses as a graph
- Implicit representation of orientations using rotation matrices
- 2D and 3D representations
- Properties of rotation matrices
- Homogeneous transformations (translations + rotations)
- Euler angles for explicit representation of rotations (singularity issues)
- Quaternions as implicit representations of orientations
- Basic ROS frames
- Use of relative pose transformations in ROS, TF package