EECE 5550
Mobile Robotics

Lecture 2: Coordinate Systems and Geometry

Derya Aksaray

Assistant Professor

Department of Electrical and Computer Engineering

Northeastern University

Today's Agenda

- Coordinate systems
- Coordinate transformations and transformation groups
- Rotation representations in 2 and 3 dimensions

Sources & references for this lecture:

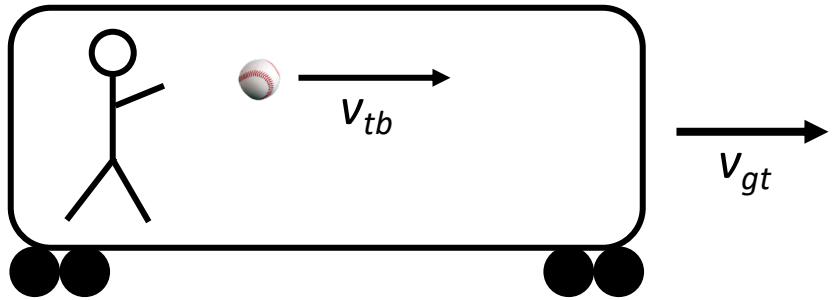
- Handbook of Robotics, Sec. 1.1-1.2
- "Math Fundamentals" – A. Kelly
- "Representing Robot Pose – The Good, the Bad, and the Ugly" – P. Furgale
  http://paulfurgale.info/news/2014/6/9/representing-robot-pose-the-good-the-bad-and-the-ugly

Modeling geometric / physical relations

Almost all physically-meaningful quantities are *relative!*

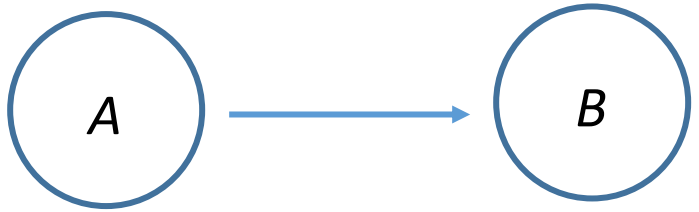
Examples

- **Position**: I am 2 m *in front of* the whiteboard
- **Velocity**: The velocity of the ball *with respect to* the train
- **Potential energy**: Gravitational potential *above ground level*



Implication: We are typically interested in modeling *geometric relations* of the form:

"Property P of object B with respect to object A"

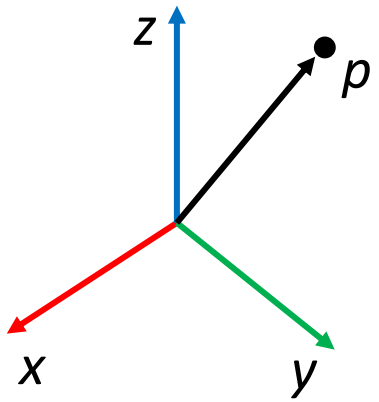


Coordinate Systems

A coordinate system is a system that uses one or more numbers, or **coordinates**, to uniquely determine the position of the points or other geometric elements on a manifold such as Euclidian space.

The use of a coordinate system allows problems in geometry to be translated into problems about numbers, so we can use computers!

$$\vec{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$$



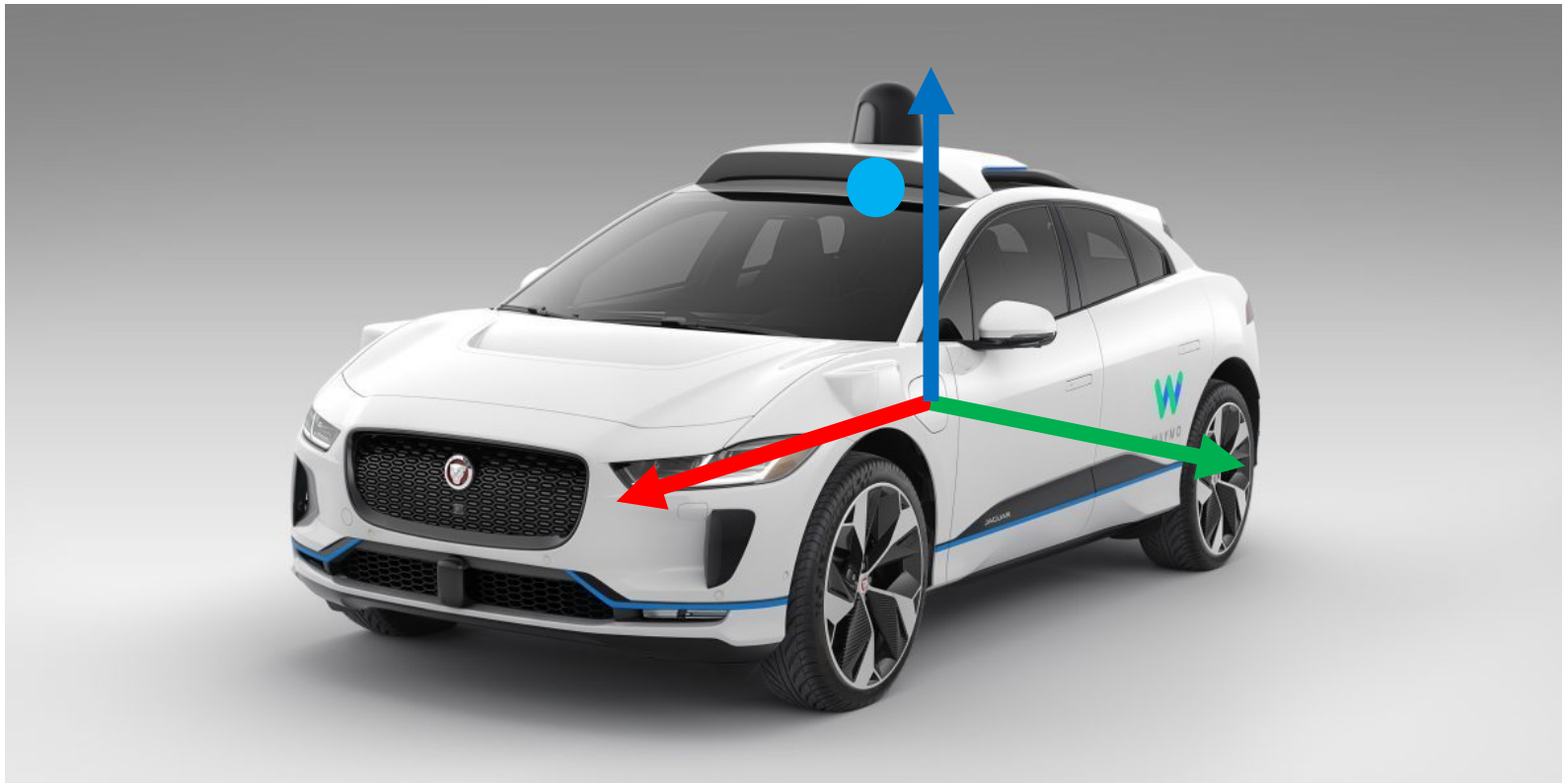
- Represent geometric objects
  Ex: 3d points ⇔ 3-dimensional vectors
- Model geometric operations
  Ex: translation ⇔ vector subtraction,
  rotation ⇔ rotation matrix multiplication

Coordinate systems

Question: Where is the point x = [1, 2.5, 2.5] with respect to the car?

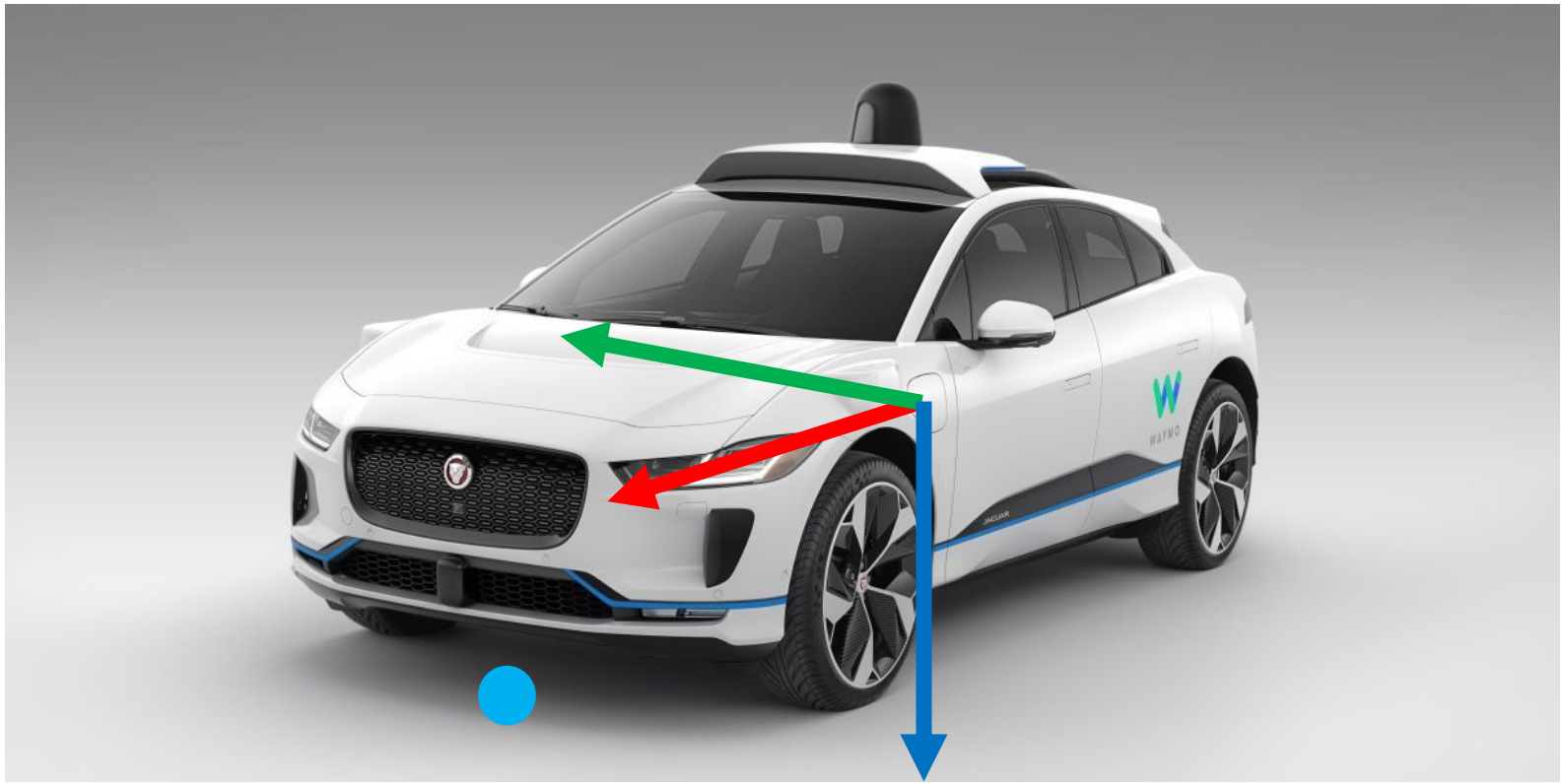
Coordinate systems

Question: Where is the point x = [1, 2.5, 2.5] with respect to the car?



Coordinate systems

Question: Where is the point x = [1, 2.5, 2.5] with respect to the car?



Coordinate systems

Question: Where is the point x = [1, 2.5, 2.5] with respect to the car?

Coordinate systems

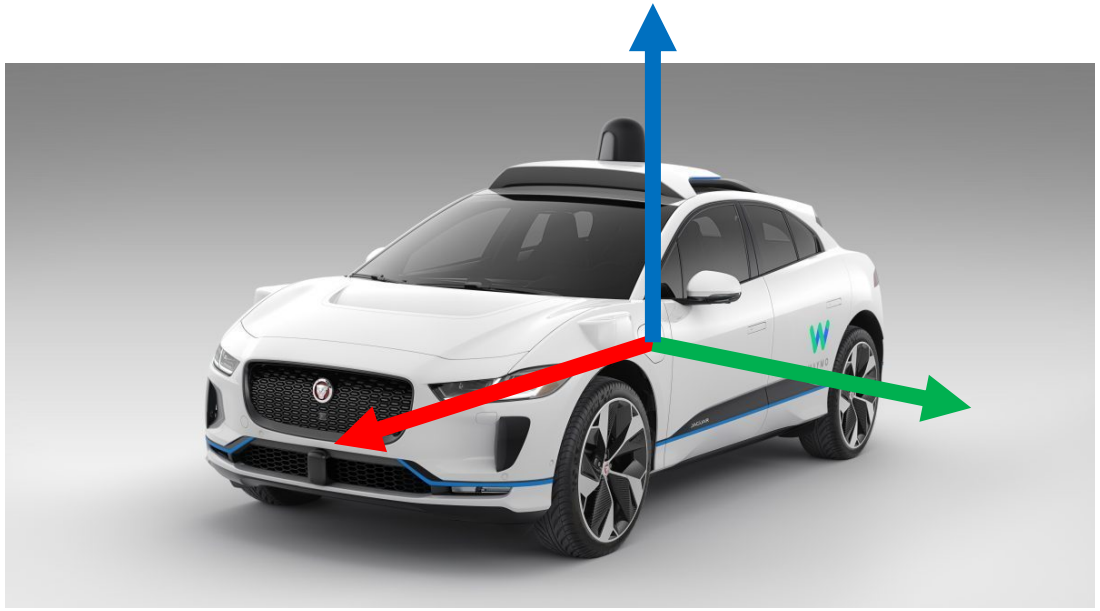
Question: Where is the point x = [1, 2.5, 2.5] with respect to the car?

Key Point: Coordinate expressions are only meaningful *if you know the reference frame*.

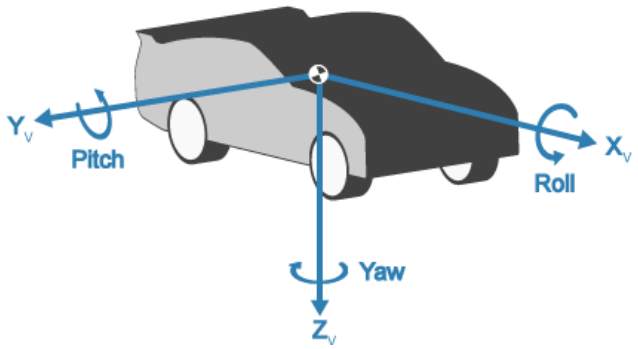
Pitfalls

- **LOTS** of different frame conventions
- **LOTS** of different frames to keep track of

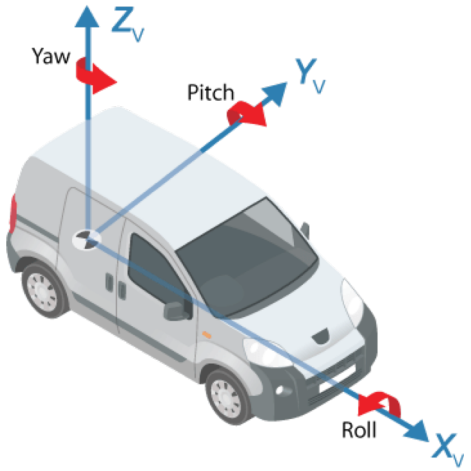
⇒ **LOTS** of opportunities for mistakes!



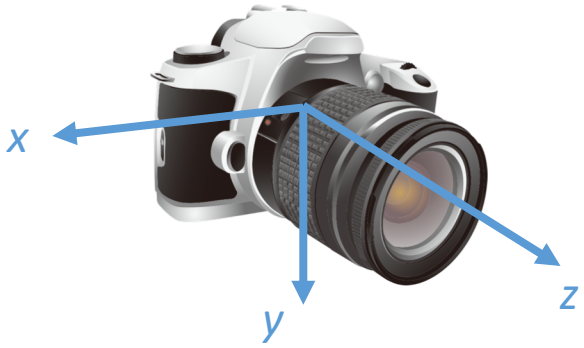
Examples of common frames



Vehicle *z*-down (robotics)



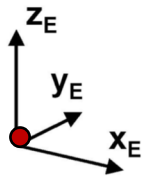
Vehicle *z*-up (robotics)



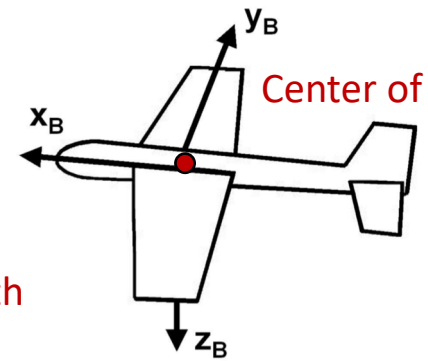
Camera body-fixed (computer vision)

Earth or Ground Fixed frame

x_E axis - positive in the direction of north
y_E axis - positive in the direction of east
z_E axis - positive towards the center of the Earth



Arbitrary point on Earth



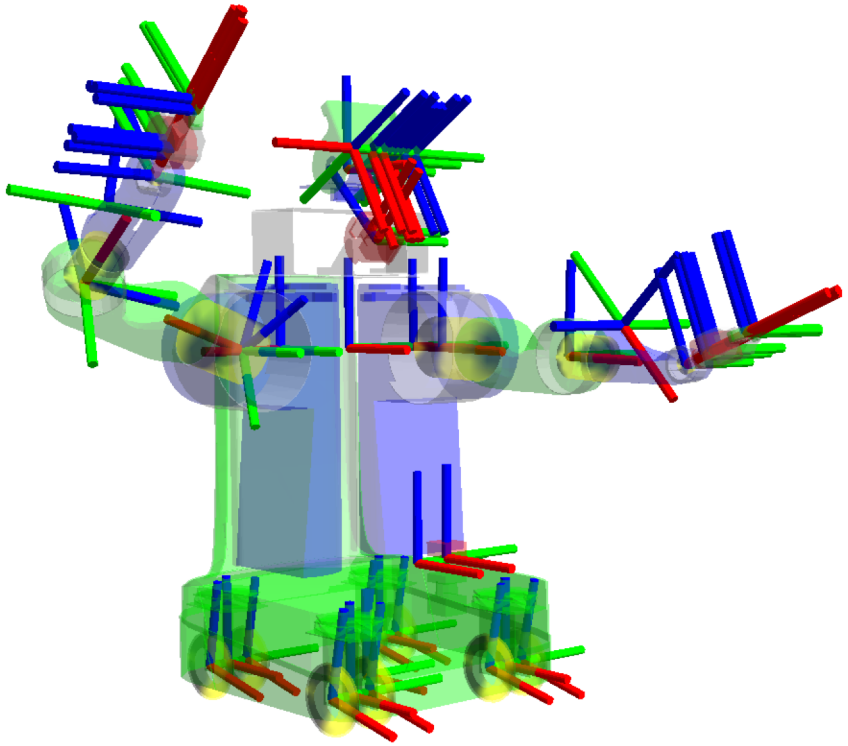
Center of gravity

Aircraft or Body frame

x_B axis - positive out the nose of the aircraft
z_B axis - perpendicular to the x_b axis, positive below the aircraft
y_B axis - perpendicular to the x_b,z_b-plane, positive determined by RHR

Examples of coordinate frames on robots

Willow Garage PR2 robot



PR2 coordinate frames

Examples of coordinate frames on robots

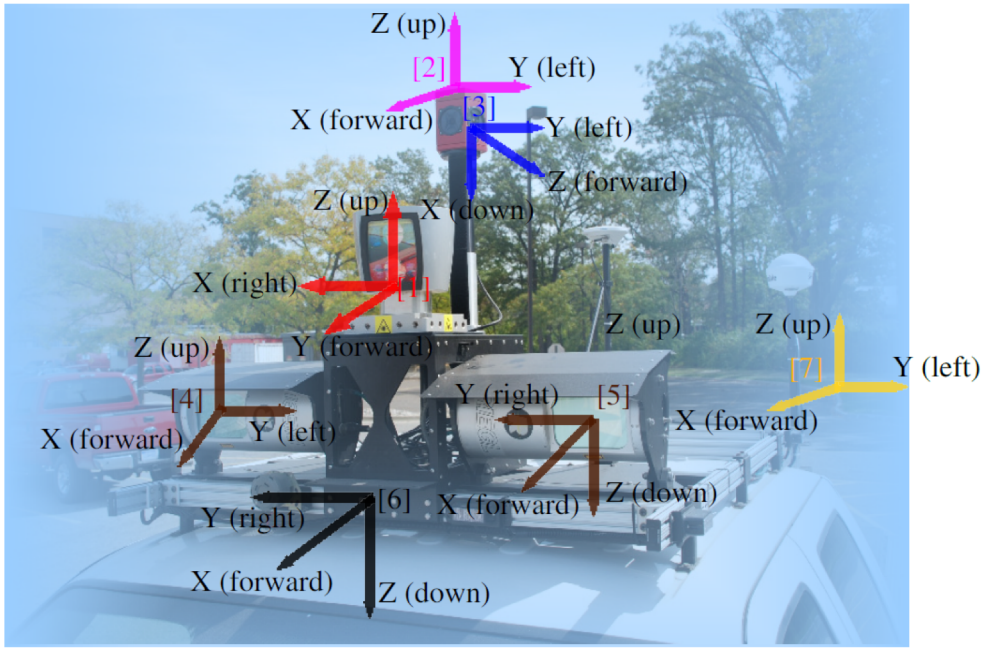
Note that

- every sensor
- every rigid link / moving part

has its own associated coordinate frame.

That is **A LOT** to keep track of …

=> **LOTS** of opportunities for mistakes!



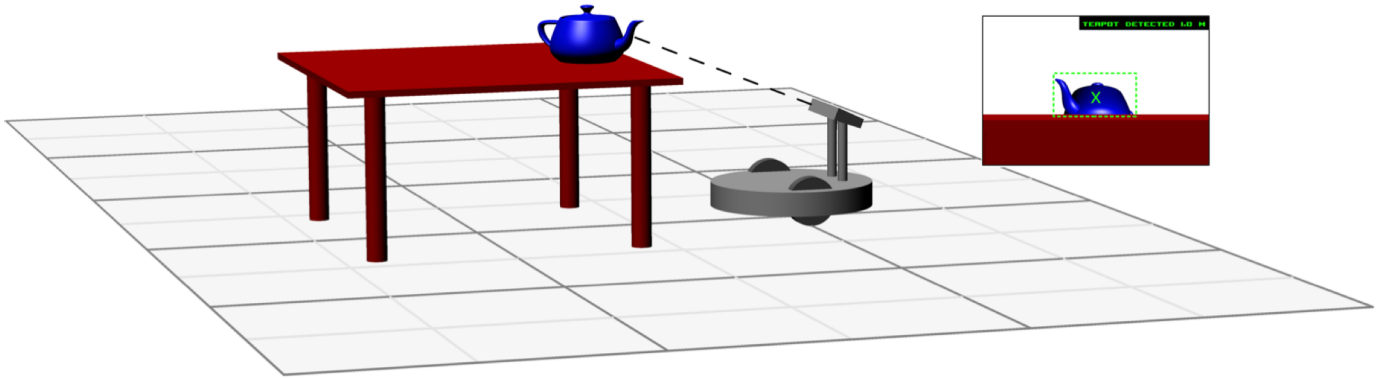
[1] Velodyne, [2] Ladybug3 (actual location: center of camera system),
[3] Ladybug3 Camera 5, [4] Right Riegl, [5] Left Riegl,
[6] Body Frame (actual location: center of rear axle)
[7] Local Frame (Angle between the X-axis and East is known)

Pandey et al.: "Ford Campus Vision and Lidar Data Set"

Why do we need to use so many frames?

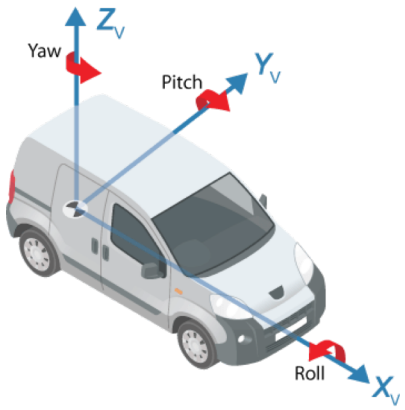
- The position of the teapot relative to the camera is 1m.
- The position and orientation of the camera relative to the base of the robot.
- The position of the robot in the room.

Actuators/sensors mostly process/produce information in their local coordinate frames.



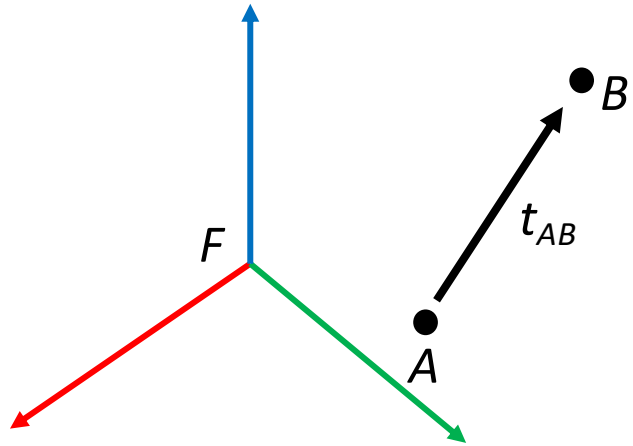
3 Simple Rules for Minimizing Coordinate-Based Suffering

1. **Have Fear*:** if it is not *absolutely clear* what the reference frame is, you can (and should!) always check this using a 3D plotting tool



2. Always draw a **frame diagram**!

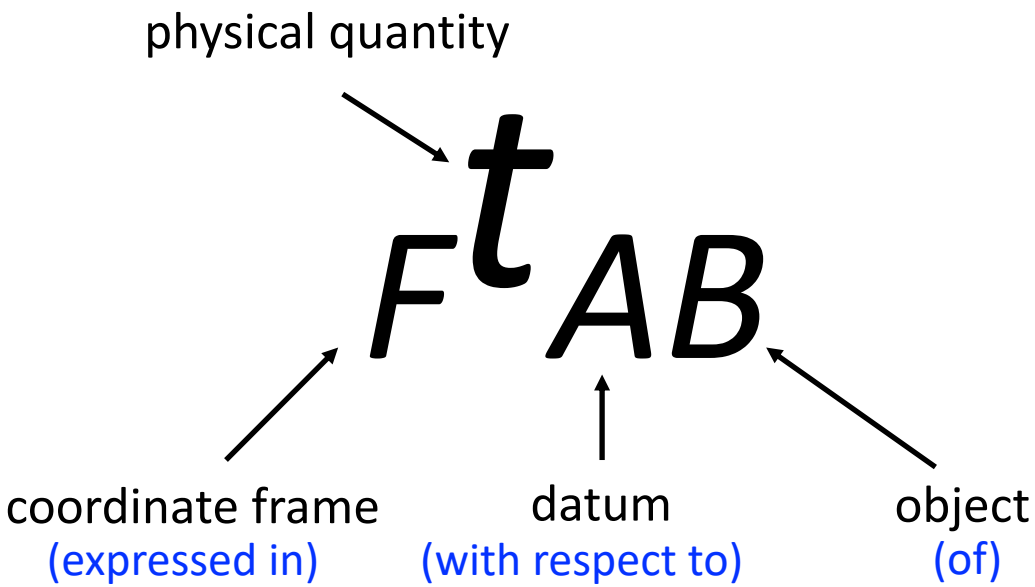
3. Use **explicit notation** for coordinate expressions that captures:
   - The object of interest (to which the property attaches)
   - The datum (reference) for the property
   - The coordinate frame

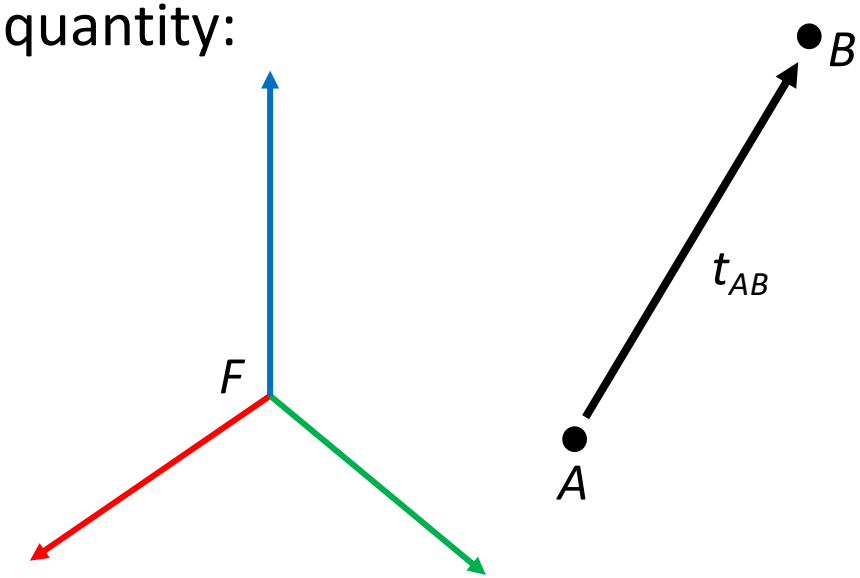


*h/t Paul Furgale

Notational conventions for coordinate expressions

Common **explicit** notational convention for a physical quantity:





Example: $_F t_{AB}$ means "the translation of *B* relative to *A*, expressed in coordinate frame *F*"

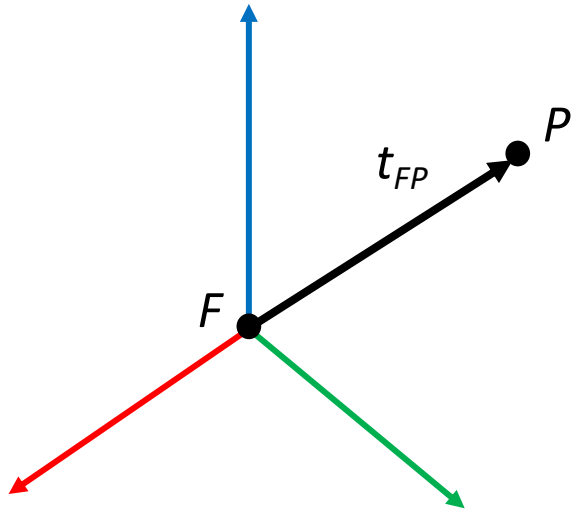
"the vector from A to B expressed in F"

Notational conventions for coordinate expressions

Special case: We may drop explicit reference to the datum if we are expressing a quantity with respect to the origin of the coordinate frame.

$$ {}_{F}p := {}_{F}t_{FP} $$

Example: ${}_{F}p$ means "the position of point *P* expressed in frame *F*"



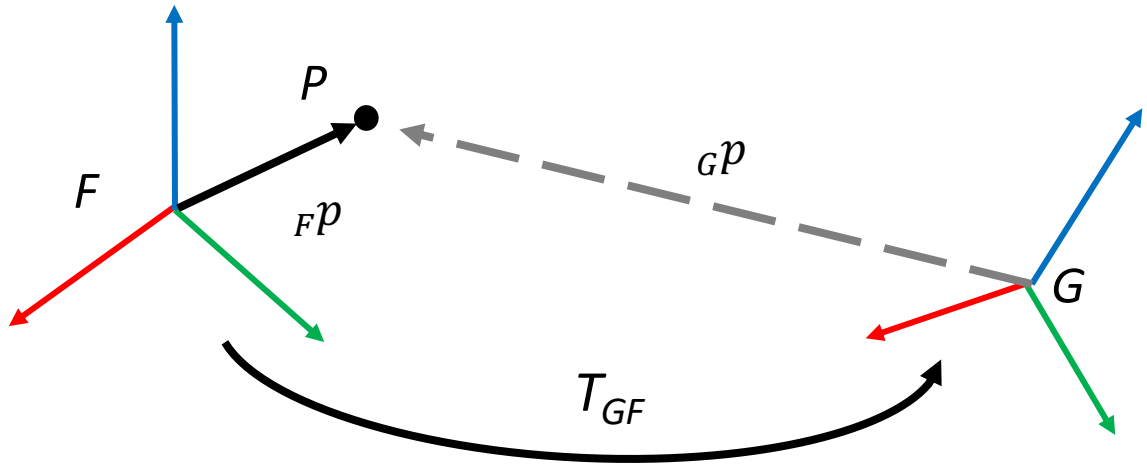
Transformations between coordinate frames

Two coordinate frames *F* and *G* are related by an *affine transformation.*

Notation convention:

$$T_{GF}$$

Transform to Transform from



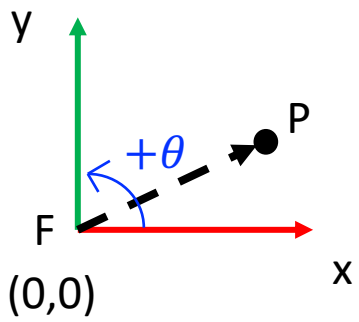
Example: T_{GF} means "the affine transformation mapping coordinates *from frame F to frame G*."
" the orientation of frame F relative to frame G."

Payoff: Using this convention, coordinate transformations satisfy simple "cancellation laws":

$${}_Gp = T_{GF}({}_Fp) \qquad T_{HF} = T_{HG}\,T_{GF}$$

Coordinate Frame Transformations

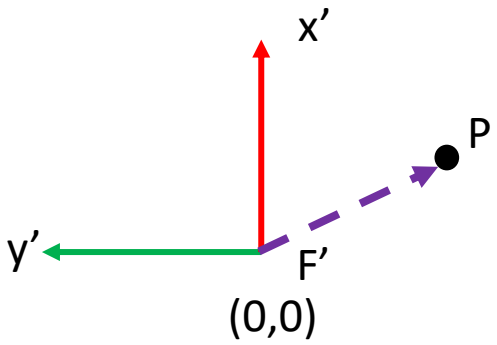
Example: 2D frames



$$_{F}p = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

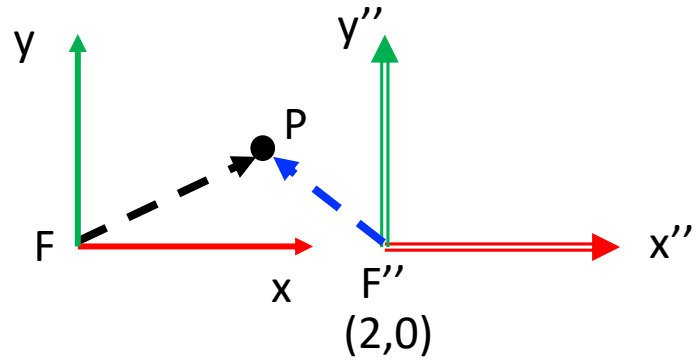
Rotation

$T_{F'F}$: Frame F is oriented at -90 deg wrt frame F'.



$$_{F'}p = T_{F'F}(\,_{F}p) = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -1 \end{bmatrix}$$

Translation



$$_{F''}p = \,_{F}p - \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}$$

Homogeneous coordinates

Cartesian coordinates (x, y) vs. Homogenous coordinates (x, y, w)

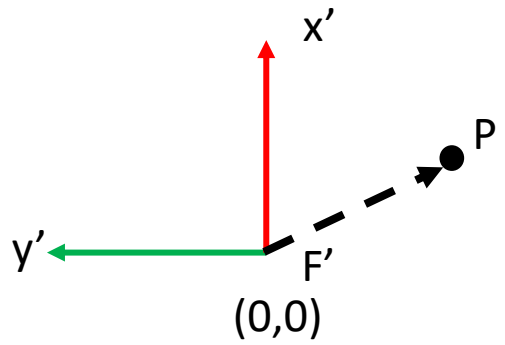
- $w > 0$
- Homogenous coordinates increase the dimension by one

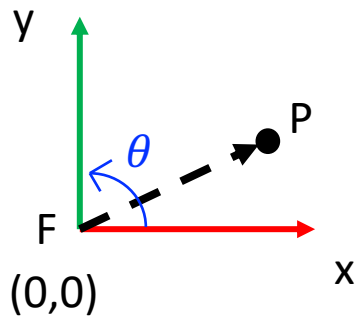
Example:

- The point $(1.5,2)$ in Cartesian coordinates is the same as
- $(1.5,2,1)$, or $(3,4,2)$, or $(150,200,100)$, … in homogenous coordinates.

Coordinate Frame Transformations

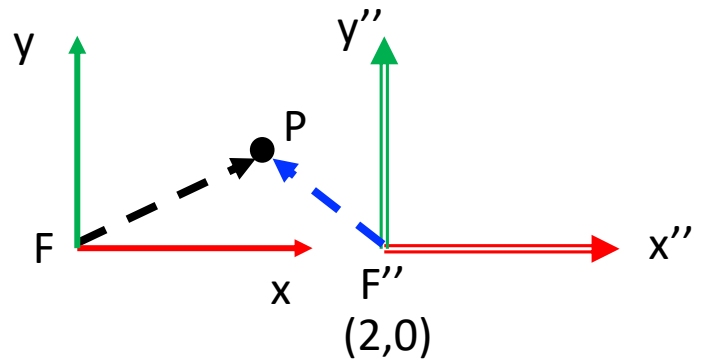
Rotation





$$_{F}p = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

Translation



$$_{F'}p = \begin{bmatrix} 0.5 \\ -1 \end{bmatrix} = T_{F'F}(\,_{F}p) = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

$$_{F''}p = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix} = \,_{F}p - \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$_{F'}p = \begin{bmatrix} 0.5 \\ -1 \\ 1 \end{bmatrix} = \tilde{T}_{F'F}(\,_{F}p) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0.5 \\ 1 \end{bmatrix}$$

We can now represent everything in matrix multiplication form.

$$_{F''}p = \begin{bmatrix} -1 \\ 0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0.5 \\ 1 \end{bmatrix}$$

Homogenous representations of coordinate transformations

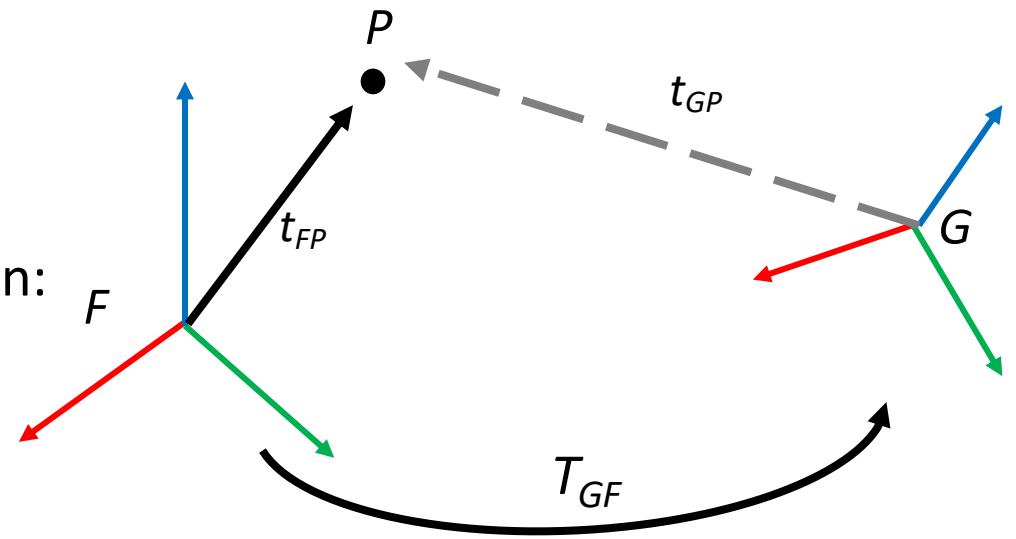
Two coordinate frames *F* and *G* are related by an *affine transformation*.

$$T_{GF}(x) = A_{GF}x + b_{GF}$$

$$\begin{pmatrix} A_{GF} & b_{GF} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ 1 \end{pmatrix} = \begin{pmatrix} A_{GF}x + b_{GF} \\ 1 \end{pmatrix} = \begin{pmatrix} T_{GF}(x) \\ 1 \end{pmatrix}$$

If we write a vector in homogeneous coordinates, then:

$$\widehat{T_{GF}(x)} = \underbrace{\begin{pmatrix} A_{GF} & b_{GF} \\ 0 & 1 \end{pmatrix}}_{\text{Affine linear transformation}} \hat{x}$$



Exercise

Given frame transformations T_{GF} and T_{HG}, with:

$$T_{GF}(x) = A_{GF}x + b_{GF} \qquad T_{HG}(x) = A_{HG}x + b_{HG}$$

Find:

1. The frame transformation T_{HF}.

2. The frame transformation T_{FG}.

Solution

Given frame transformations T_{GF} and T_{HG}, with:

$$T_{GF}(x) = A_{GF}x + b_{GF} \qquad T_{HG}(x) = A_{HG}x + b_{HG}$$

Find:

1. The frame transformation T_{HF}.

$$T_{HF} = T_{HG}\,T_{GF} = \begin{pmatrix} A_{HG} & b_{HG} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} A_{GF} & b_{GF} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} A_{HG}A_{GF} & A_{HG}b_{GF} + b_{HG} \\ 0 & 1 \end{pmatrix}$$

$$\Rightarrow T_{HF}(x) = A_{HG}A_{GF}x + (A_{HG}b_{GF} + b_{HG})$$

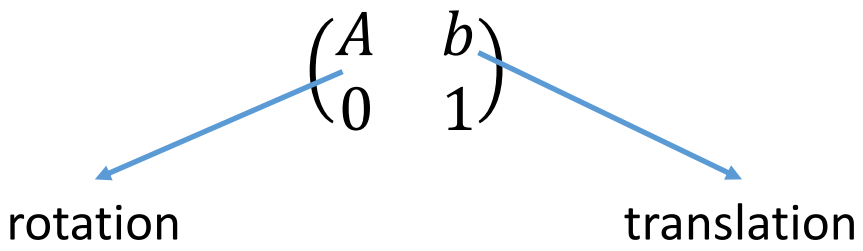
2. The frame transformation T_{FG}.

$$T_{FG} = T_{GF}^{-1} = \begin{pmatrix} A_{GF} & b_{GF} \\ 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} A_{GF}^{-1} & -A_{GF}^{-1}b_{GF} \\ 0 & 1 \end{pmatrix}$$

$$\Rightarrow T_{FG}(x) = A_{GF}^{-1}x - A_{GF}^{-1}b_{GF}$$

Generalizing the transformation matrices

- We have seen that every coordinate transformation between frames in $\mathbb{R}^d$ can be represented by a (d+1) x (d+1) matrix in the form of:



- We will use the concept of “groups” in mathematics.

Group definition

A group G is a set of elements $\{X, Y, Z, \ldots\}$ with an operation " $*$ " such that:

- Composition stays in the group: $X * Y$ is in G
- Identity element is in the group: $X * E = E * X = X$
- Inverse element is in the group: $X^{-1} * X = X * X^{-1} = E$
- Operation is associative: $X * (Y * Z) = (X * Y) * Z$

- In many groups of interest, the operation " $*$ " is non-commutative: $X * Y \neq Y * X$

Group examples

Let's check if $(\mathbb{R},+)$ and $(\mathbb{R}\backslash\{0\},.)$ are groups:

- Composition stays in the group:
  - $a, b \in \mathbb{R}, \quad a + b \in \mathbb{R}, \quad a.b \in \mathbb{R}$
- Identity element is in the group:
  - $a + 0 = a, \quad 1.a = a$
- Inverse element is in the group:
  - $a + (-a) = 0, \quad a.a^{-1} = 1$
- Operation is associative:
  - $a + (b + c) = (a + b) + c, \quad a.(b.c) = (a.b).c$

Some important groups

- A matrix group is a group of invertible matrices.

Def: The general linear group over $\mathbb{R}$ is:

$$GL(d) = \{A \in \mathbb{R}^{d\times d} | \det(A) \neq 0\}$$

Def: The d-dimensional affine group over $\mathbb{R}$ is:

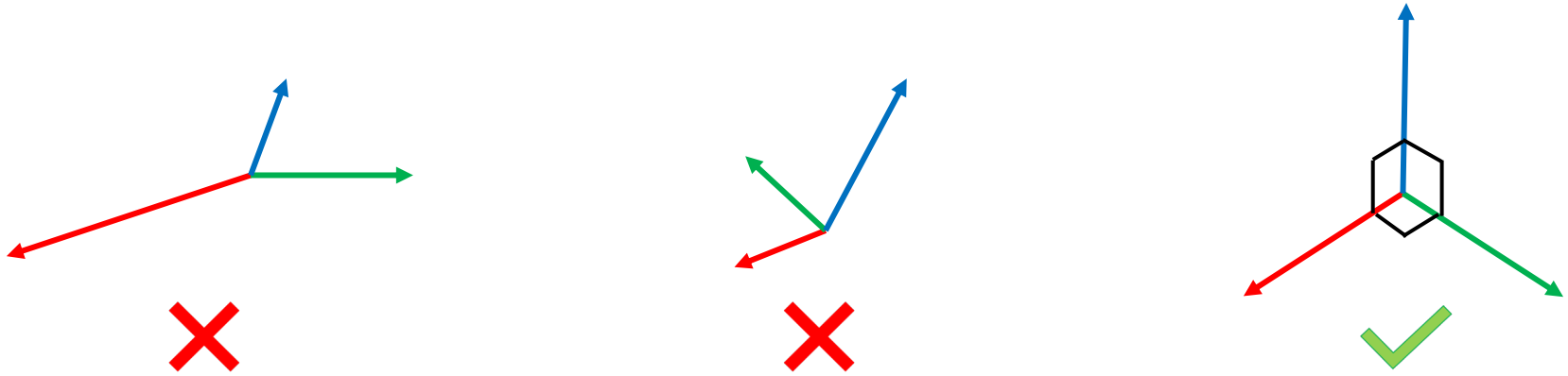
All coordinate transformations

$$Aff(d) \coloneqq \left\{ \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} | \, A \in GL(d), b \in \mathbb{R}^d \right\}$$

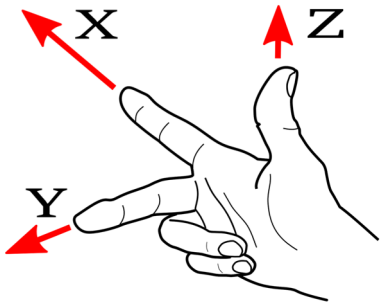
- Accordingly, $Aff(d)$ is a subgroup of the general linear group $GL(d+1)$.

Structure-preserving subgroups of the affine group

So far we have considered *completely generic* coordinate systems and transformations.



But: often we would like our coordinate frames & transformations to have some “nice” additional structure.



In particular, we often take our coordinate systems to be:

- Orthonormal
- Right-handed

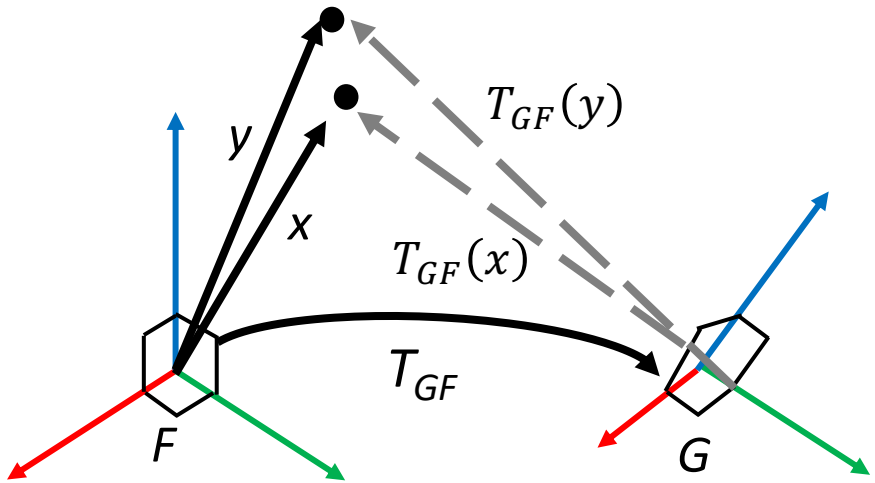
Structure-preserving subgroups of the affine group

This restriction imposes extra constraints on the transformations *T* that relate these frames.

- If *F* and *G* are both **orthonormal**, then T_{GF} *preserves distances* between mapped points:

$$d\big(T_{GF}(x), T_{GF}(y)\big) = d(x, y)$$

$$\|(A_{GF}x + b) - (A_{GF}y + b)\| = \|x - y\|$$



$$\sqrt{(A_{GF}x + b - A_{GF}y - b)^T(A_{GF}x + b - A_{GF}y - b)} = \sqrt{(x - y)^T(x - y)}$$

$$x^T A_{GF}^T A_{GF} x - x^T A_{GF}^T A_{GF} y - y^T A_{GF}^T A_{GF} x + y^T A_{GF}^T A_{GF} y = x^T x - x^T y - y^T x + y^T y$$

$$T_{GF} = \begin{pmatrix} A_{GF} & b_{GF} \\ 0 & 1 \end{pmatrix}$$

This should be true for any x and y. Suppose that y is all zeros. Then, $x^T A_{GF}^T A_{GF} x = x^T x \quad \rightarrow \quad A_{GF}^T A_{GF} = I$

Some important groups

Def: The d-dimensional orthogonal group over $\mathbb{R}$ is:

$$O(d) \coloneqq \{R \in \mathbb{R}^{d\times d} | R^T R = I\}$$

Note: The determinant of an orthogonal matrix is either +1 or -1.

Def: The d-dimensional Euclidean group over $\mathbb{R}$ is:

$$E(d) \coloneqq \left\{ \begin{pmatrix} R & b \\ 0 & 1 \end{pmatrix} | \, R \in O(d), b \in \mathbb{R}^d \right\}$$

*Distance preserving rotations and reflections are the orthogonal group.

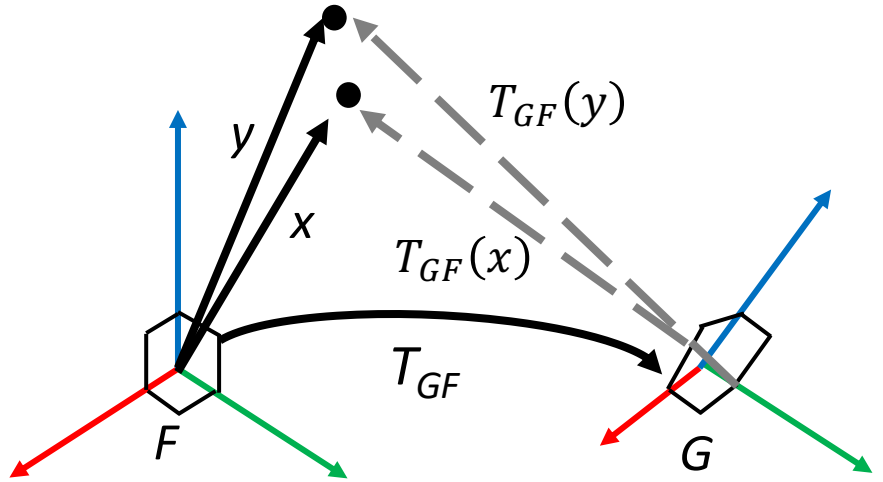
**Distance preserving transformations (i.e., translation, rotation, reflection) are the Euclidian group.

Structure-preserving subgroups of the affine group

- If *F* and *G* are both **orthonormal *and* right-handed**, then T_{GF} preserves *distance and orientation*.
- In this case, A_{GF} belongs to the **special orthogonal group**:

$$SO(d) \coloneqq \{R \in \mathbb{R}^{d\times d} | R^T R = I, \det(R) = 1\}$$

*The group of rotations



$$T_{GF} = \begin{pmatrix} A_{GF} & b_{GF} \\ 0 & 1 \end{pmatrix}$$

The group of all *distance- and orientation-preserving transformations* of $\mathbb{R}^d$ is called the **special Euclidean group**, or the group of ***rigid motions:***

$$SE(d) \coloneqq \left\{ \begin{pmatrix} R & b \\ 0 & 1 \end{pmatrix} \mid R \in SO(d), b \in \mathbb{R}^d \right\}$$

- translation
- rotation
- ~~reflection~~

A taxonomy of spatial transformation groups

- **Affine group:** group of transformations between *general coordinate frames* in $\mathbb{R}^d$:

$$Aff(d) \coloneqq \left\{ \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} \mid A \in GL(d), b \in \mathbb{R}^d \right\}$$

- **Euclidean group:** *distance-preserving* transformations of $\mathbb{R}^d$:

$$E(d) \coloneqq \left\{ \begin{pmatrix} R & b \\ 0 & 1 \end{pmatrix} \mid R \in O(d), b \in \mathbb{R}^d \right\}$$

- **Special Euclidean group (rigid motions):** *distance- and orientation-preserving* transformations of $\mathbb{R}^d$

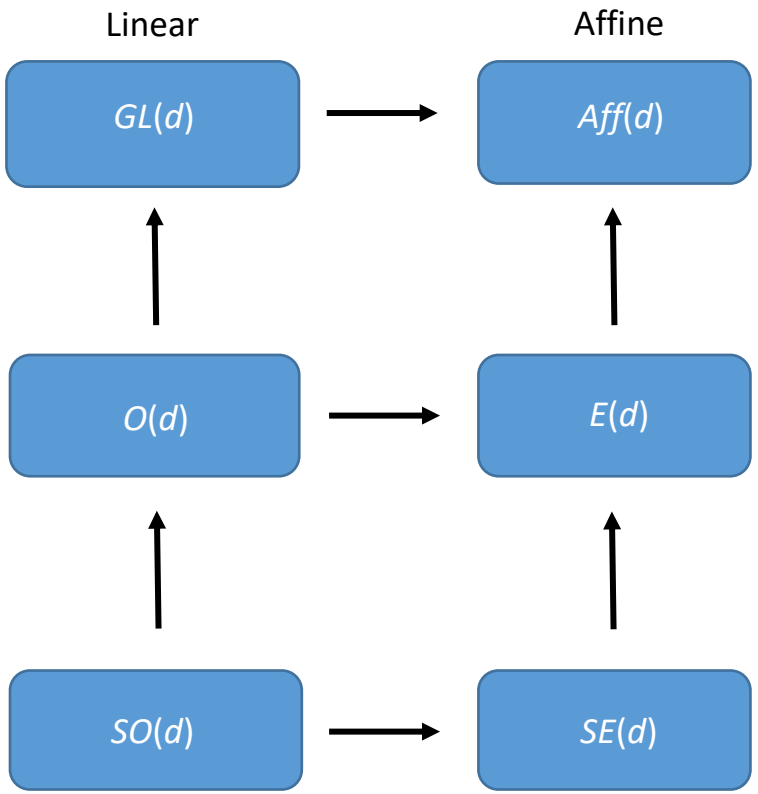
$$SE(d) \coloneqq \left\{ \begin{pmatrix} R & b \\ 0 & 1 \end{pmatrix} \mid R \in SO(d), b \in \mathbb{R}^d \right\}$$

- **Orthogonal group:** *distance-preserving linear maps* of $\mathbb{R}^d$:

$$O(d) \coloneqq \{R \in \mathbb{R}^{d \times d} | R^T R = I\}$$

- **Special orthogonal group (rotations):** *distance- and orientation-preserving linear maps* of $\mathbb{R}^d$:

$$SO(d) \coloneqq \{R \in \mathbb{R}^{d \times d} | R^T R = I, \det(R) = +1\}$$



Transformation group containment

Rigid body motion

We can model the *pose* (*position* and *orientation*) of a rigid body (i.e. robot) by attaching a fixed coordinate frame *B* to its body.

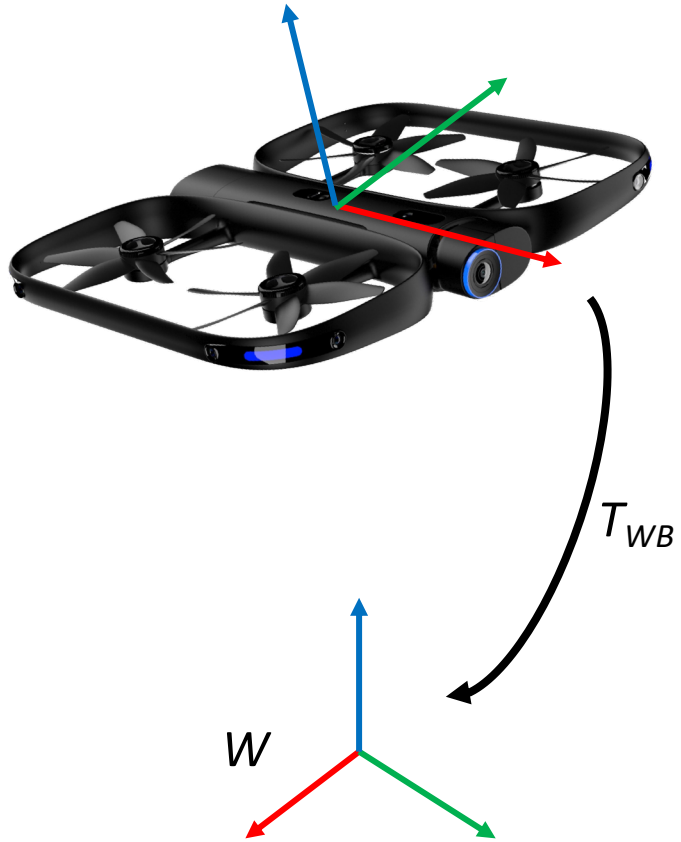
- We represent the pose of *B* with respect to a reference frame *W* using T_{WB}, the transformation that sends *B* to *W*:

$$T_{WB} = \begin{pmatrix} R_{WB} & t_{WB} \\ 0 & 1 \end{pmatrix}$$

  We call T_{WB} the "pose of B with respect to W".

- We typically take *W* and *B* to be orthonormal and right-handed, so that $T_{WB} \in \mathrm{SE(d)}$ and $R_{WB} \in \mathrm{SO(d)}$.

- t_{WB} gives the *position* of *B* with respect to *W.*
- R_{WB} gives the *orientation* (or *attitude*) of *B* with respect to *W.*



Rotation representations in 2 & 3 dimensions

Recall that the group of rotations of $\mathbb{R}^d$ is:

$$SO(d) \coloneqq \{R \in \mathbb{R}^{d\times d} | R^T R = I, \det(R) = 1\}$$

A matrix $R \in SO(d)$ has d^2 elements, and satisfies $d(d+1)/2$ constraints (from upper triangle of symmetric matrix equation $R^T R = I$). Therefore:

$$\dim\big(SO(d)\big) = \frac{d^2 - d}{2}$$

In particular:

- $\dim(SO(2)) = 1 < 2^2 = 4$
- $\dim(SO(3)) = 3 < 3^2 = 9$

Punchline: Maybe we can save some storage & compute by using an alternative representation …

Rotations in 2D

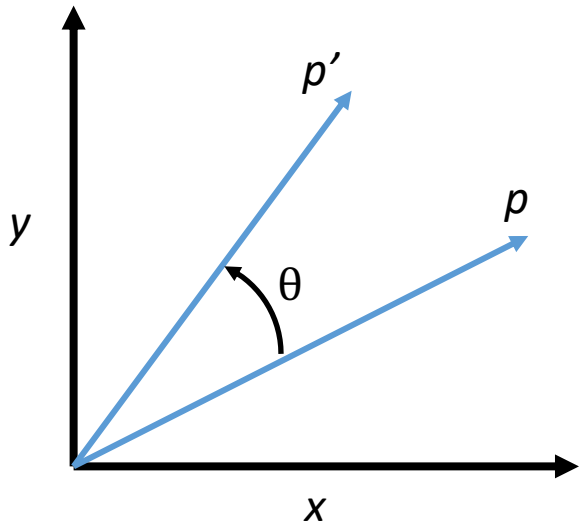
- Recall that we can describe a counterclockwise rotation in the plane using a single parameter: the *rotation angle* θ

$$\dim\big(SO(d)\big) = \frac{d^2 - d}{2} \quad \Rightarrow \quad \dim\big(SO(2)\big) = 1$$

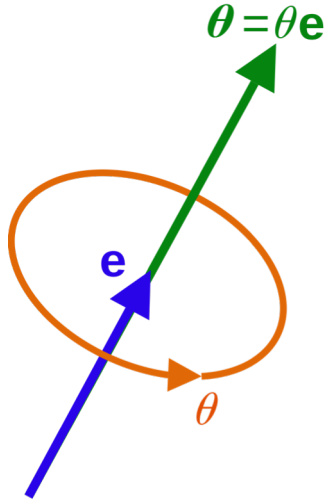
The corresponding rotation matrix is then:

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

*Knowing θ will help us to construct this matrix



Rotations in 3D



Euler's rotation theorem: Every 3D rotation is equivalent to an elementary right-handed rotation through an angle θ about an axis $\hat{e}$.

Axis-angle representation: We can describe a 3D rotation by specifying:

- The rotation axis $\hat{e}$ (3 params)
- The rotation angle θ (1 param)

Letting axis $\hat{e}$ be a unit vector, we can represent a 3D rotation using a single 3D vector as: $\boldsymbol{\theta} = \theta\hat{e}$

Rodrigues' Theorem:

Given a point $v \in \mathbb{R}^3$, axis-angle pair $(\theta, \hat{e})$, the rotated point v_r can be calculated as follows:

$$v_r = \cos(\theta)\, v + \sin(\theta)\, (\hat{e} \times v) + (\hat{e} \cdot v)(1 - \cos(\theta))\hat{e}$$

Also, an efficient algorithm to calculate the corresponding rotation matrix:

$$R(\boldsymbol{\theta}) = \cos(\theta)\, I + \sin(\theta)\, [\hat{e}]_\times + (1 - \cos(\theta))\hat{e}\hat{e}^T$$

$$[\hat{e}]_\times := \begin{bmatrix} 0 & -e_z & e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{bmatrix}$$

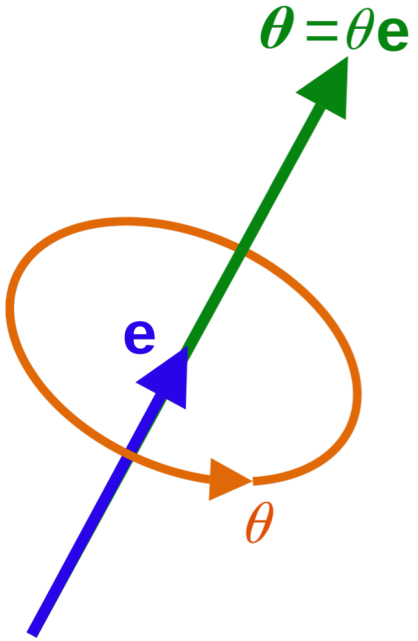
Axis-angle practicalities

Pro:

- Provides a concise [in fact *minimal*] description of a 3D rotation
- Admits an intuitive geometric description

Con:

- Does not admit computing *compositions* (*products*) of rotations easily
- Acting on (i.e., *rotating*) *points* requires use of trigonometric functions (using Rodrigues' formula)



Quaternions

Quaternions provide a representation of 3D rotations that is closely related to axis-angle, but overcomes some of its weaknesses.

Formally, a **quaternion** is a linear combination of the form:

$$a + bi + cj + dk$$

where *i,j,* and *k* are *elementary quaternions* that satisfy the following multiplication rules:

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

Key facts:

- Quaternions form a 4-dimensional vector space over $\mathbb{R}$
- They are a generalization of the complex numbers
- They are an algebra: a vector space with an *additional* bilinear product (quaternion multiplication)
- They are *non-commutative*: in general $q_1 q_2 \neq q_2 q_1$ for quaternions q_1 and q_2.

Unit quaternions and 3D rotation

Consider a rotation by an angle θ around a unit axis $v = (v_x, v_y, v_z)$.

Remember a unit quaternion:

$$q = \underbrace{a}_{\text{real part}} + \underbrace{bi + cj + dk}_{\text{vector part}}$$

We can write

$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)(v_x i + v_y j + v_z k) = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) v$$

Applying a quaternion to rotate a point in 3D space:

Let p be a point represented as a quaternion: $p = 0 + xi + yj + zk$

Let q be the quaternion representing the rotation.

$$p_r = q \otimes p \otimes q^{-1}$$

Example: Rotate the point $\begin{bmatrix}5\\9\\6\end{bmatrix}$ by the rotation quaternion $q = \begin{bmatrix}0.16\\-0.07\\0.11\\-0.98\end{bmatrix}$

$$\mathrm{q} = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) v$$

$$q \otimes p \otimes \bar{q} = \begin{bmatrix}0.16\\-0.07\\0.11\\-0.98\end{bmatrix} \otimes \begin{bmatrix}0\\5\\9\\6\end{bmatrix} \otimes \begin{bmatrix}0.16\\0.07\\-0.11\\0.98\end{bmatrix}$$

$$= \begin{bmatrix}0.16 & 0.07 & -0.11 & 0.98\\-0.07 & 0.16 & -0.98 & -0.11\\0.11 & 0.98 & 0.16 & -0.07\\-0.98 & 0.11 & 0.07 & 0.16\end{bmatrix}\begin{bmatrix}0\\5\\9\\6\end{bmatrix} \otimes \begin{bmatrix}0.16\\0.07\\-0.11\\0.98\end{bmatrix}$$

$$= \begin{bmatrix}5.24\\10.28\\-3.04\\-0.22\end{bmatrix} \otimes \begin{bmatrix}0.16\\0.07\\-0.11\\0.98\end{bmatrix}$$

$$= \begin{bmatrix}5.24 & -10.28 & 3.04 & 0.22\\10.28 & 5.24 & -0.22 & 3.04\\-3.04 & 0.22 & 5.24 & 10.28\\-0.22 & -3.04 & -10.28 & 5.24\end{bmatrix}\begin{bmatrix}0.16\\0.07\\-0.11\\0.98\end{bmatrix} = \begin{bmatrix}0\\-0.99\\-11.15\\4.18\end{bmatrix}$$

Rotated point coordinates

Credit: Kaveh Fathian

Quaternion to rotation matrix

$$q = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} \quad \Rightarrow \quad R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}$$

$$R = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & w^2 - x^2 + y^2 - z^2 & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & w^2 - x^2 - y^2 + z^2 \end{bmatrix}$$

Credit: Kaveh Fathian

Summary of 3D rotation parameterizations

Representation	# Parameters	# Constraints	Pro	Con
Rotation matrix	9	6 (orthonormality)	Fast action on points (mat-vec multiplication)	• Overparameterization • More complex constraints • Composition is (slightly) more expensive (mat-mat multiplication)
Axis-angle	3	0	• Minimal params • Geometrically intuitive	• Action on points is expensive to compute • Can't easily compute compositions
Unit quaternion	4	1 (normalization)	• Slight overparameterization • Simple constraints • Can easily compute compositions	• Action on points is expensive to compute

Key point: Different reps are good for different things! Many software libraries will use *both* for different tasks. (Ex: use quaternions for calculating compositions, and then convert to a rotation matrix to act on points.)

Summary

- Coordinate systems and notational conventions
- Coordinate transformations and transformation groups
- Representations of rotation in 2D and 3D space