Contents

1 Rotation **2**
1.1 Special Euclidian Group . . . 3
1.2 Special Orthogonal Group $SO(3)$ . . . 5
1.3 2D Rotations . . . 6
1.3.1 Angles . . . 6
1.3.2 Unit-norm Complex Number . . . 6
1.4 3D Rotation . . . 7
1.4.1 Euler Angles . . . 7
1.4.2 Axis Angle Representation . . . 10
1.4.3 Quaternion . . . 14
1.5 Rigid Body Poses . . . 16

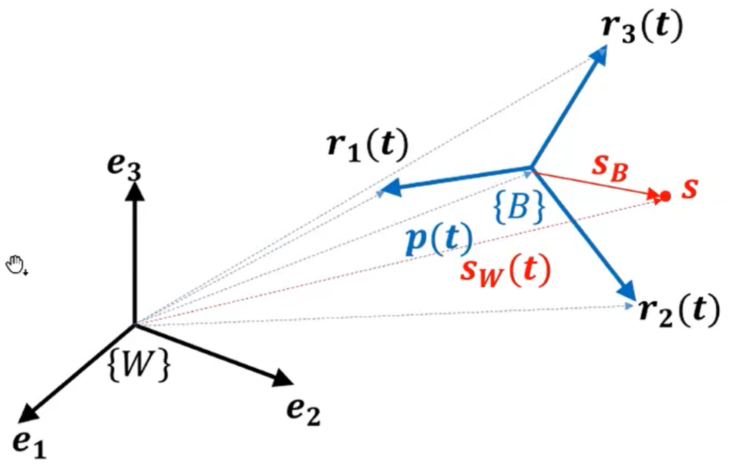
2 Robot Motion and Observation Models **18**
2.1 Motion Models . . . 20
2.2 Odometry . . . 21
2.3 Differential-Drive Kinemantic Model . . . 22

Chapter 1

Rotation

In this lecture, we will discuss how to handle orientation. Robots are typically rigid bodies, that have position and orientation. Positions are simply x, y, z coordinates but rotation is more complex. We will then discuss how we can model the motion and sensing of a robot and bring it back to the optimization theory we discussed. Hence, we have to be able to optimise over position and rotation.

We consider two frames - the world or the map $\{W\}$ and the body reference frame $\{B\}$, for the robot itself. Since the robot remains the same, we can specify the motion of one point $p(t) \in R^3$ and three coordinate axes $r_1(t), r_2(t), r_3(t)$, attached to that point.



In the above figure, we can see the black axes as the world frame, and the blue lines as the body frame. The point s will have constant coordinates with respect to the body frame (it is some point on the robot), but different coordinates in the world frame (since the body is constantly moving). Alternatively,

A point s on the rigid body has fixed coordinates $s_B \in R^3$ in the body frame $\{B\}$ but time-varying coordinates $s_w(t) \in R^3$ in $\{W\}$.

Pose

The pose $T(t) \in SE(3)$ of a rigid body reference frame $\{B\}$ at time t in a fixed world frame $\{W\}$ is determined by:

1. The position $p(t) \in R^3$ of $\{B\}$ relative to $\{W\}$.
2. The orientiation $R(t) \in SO(3)$ of $\{B\}$ relative to $\{W\}$, determined by the 3 coordinate axes $r_1(t), r_2(t), r_3(t)$.

Now, how do we describe the space of orientations of orientations $SO(3)$ and the space of poses $SE(3)$.

1.1 Special Euclidian Group

Rigid body motion is described by a sequence of functions that describe how the coordinates of 3-D points on the object change with time.

Rigid body motion preserves distances (vector norms) and does not allow reflection of the coordinate system (vector cross products).

For instance, the distance between our eyes should not change when we move around. Further, it shouldn't be possible to get a mirror image of a body when it moves.

The **Euclidian Group** $E(3)$ is a set of functions $g : R^3 \to R^3$ that preserves the norm of any two vectors.

The **Special Euclidian Group** $SE(3)$ is a set of functions $g : R^3 \to R^3$ that preserve the norm and the cross product of any two vectors.

1. Norm: $\|g_*(u) - g_*(v)\| = \|v - u\|, \forall u, v \in R^3$.
2. Cross product: $g_*(u) \times g_*(v) = g_*(u \times v), \forall u, v \in R^3$

where $g_*(x) := g(x) - g(0)$.

Corollary: $SE(3)$ elements g also preserve:

1. Angle: $u^T v = \frac{1}{4}(\|u+v\|^2 - \|u-v\|^2) \implies u^T v = g_*(u)^T g_*(v), \forall u, v \in R^3$.
2. Volume: $\forall u, v, w \in R^3, g_*(u)^T(g_*(v) \times g_*(w)) = u^T(v \times w)$

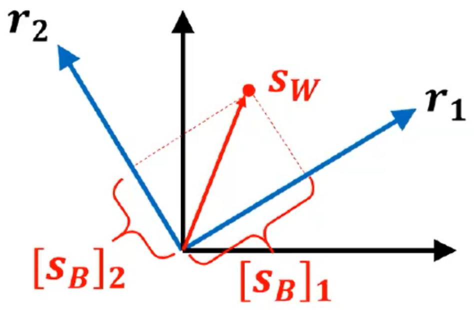
Pure rotation is a special case of rigid body motion. The orientation of the body frame $\{B\}$ in the world frame $\{W\}$ is determined by the coordinates of the three orthogonal vectors $r_1 = g(e_1), r_2 = g(e_2), r_3 = g(e_3)$, transformed from $\{B\}$ to $\{W\}$.

The vectors organized in a 3×3 matrix specify the orientation of $\{B\}$ in $\{W\}$:

$$_{\{W\}}R_{\{B\}} = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

Consider a point with coordinates $s_B \in R^3$ in $\{B\}$. Its coordiantes s_W in $\{W\}$ are:

$$\begin{aligned} s_W &= [s_B]_1 r_1 + [s_B]_2 r_2 + [s_B]_3 r_3 \\ &= R s_B \end{aligned}$$



Not sure if this is correct This is because the transformation from the body frame to world frame is the same as the position of the body frame in the world frame.

The rotation transformation g from $\{B\}$ to $\{W\}$ is:

$$g(s) = Rs$$

1.2 Special Orthogonal Group $SO(3)$

r_1, r_2, r_3 from an orthonormal basis: $r_i^T r_j = 1$ (when $i = j$), else 0.

Since r_1, r_2, r_3 form an orthonormal basis, the inverse of R is its transpose $R^T R = I$ and $R^{-1} = R^T$.

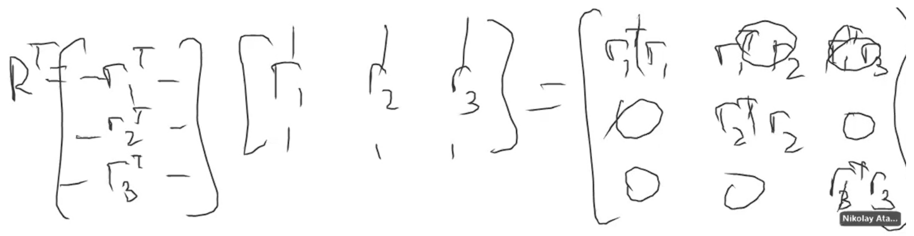


Figure 1.1: A quick illustration to see why $R^T R = I$

R belongs to the orthogonal group:

$$O(3) := \{R \in \mathbb{R}^{3\times 3} | R^T R = RR^T = I\}$$

Distance preserving: Let's look at -

$$\|Rx - Ry\|^2 = \|R(x-y)\|^2 = (x-y)^T R^T R(x-y) = (x-y)^T(x-y) = \|x-y\|^2$$

Reflections are not allowed since $\det(R) = r_1^T(r_2 \times r_3) = 1$:

$$R(x \times y) = R(x \times (R^T R y)) = (R\hat{x}R^T)Ry = \frac{1}{det(R)}(Rx) \times (Ry)$$

Hence, we can say that R belongs to the special orthogonal group:

$$SO(3) := \{R \in \mathbb{R}^{3\times 3} | R^T R = I, det(R) = 1\}$$

1.3 2D Rotations

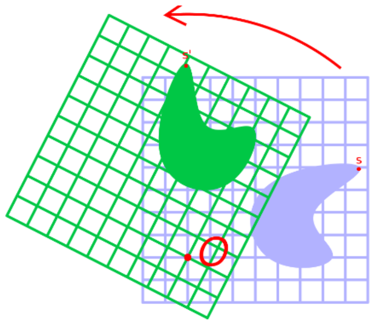
1.3.1 Angles

In two dimensions, we can just use an angle around Z axis to represent rotation.

- **Rotation angle**: a 2-D rotation of a point $\mathbf{s}_B \in \mathbb{R}^2$ can be parametrized by an angle θ around the z-axis:

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

- $\theta > 0$: counterclockwise rotation



The angle works out this way because of right hand thumb rule (fingers curl in direction of rotation - axis coincides with thumb)

The matrix values can be calculated by simply making the columns the positions of the X and Y axis after rotation.

1.3.2 Unit-norm Complex Number

We can represent a rotation by a complex number, something like $e^{i\theta}$, which can be expanded by euler's formula:

$$e^{i\theta} = \cos\theta + i\sin\theta$$

Or, if we have a 2D rotation of $[s_B]_1 + i[s_B]_2 \in \mathbb{C}$

$$e^{i\theta}([s_B]_1 + i[s_B]_2) = ([s_B]_1\cos\theta - [s_B]_2\sin\theta) + i([s_B]_1\sin\theta + [s_B]_2\cos\theta)$$

We'll revisit this in the 3D scenario.

1.4 3D Rotation

Euler Angles

This is our 3D extension of the 2D angle representation. When in 2D, we had an angle around one axis, in 3D, we have it about 3 different axes.

Or, Euler angles: an extension of the rotation angle parametrization of 2-D rotations that specifies rotation angles around the three principal axes.

Axis Angle

An extension of the rotation angle parametrization of 2-D rotations that allows the axis of rotation to be chosen freely instead of being a fixed principal axis.

Unit Quaternion

An extension of the unit-norm complex number parametrization of 2-D rotations.

1.4.1 Euler Angles

This uses three angles to specify rotation around three principal axes.

There are 24 different ways to apply these rotations - depending on the order, which axes we come back to, whether we rotate around original axes, or we keep using the updated one.

1. Extrinsic axes: The rotation axes remain static
2. Intrinsic Axes: THe rotation axes move with the rotations.
3. Further, there are two groups that they can be divided into:
    (a) Euler angles: Rotat ion about one axis, then a second, and then the first
    (b) Tait-bryan Angles: Rotation about all three axes.

The Euler and Tait-Bryan Angles each have 6 possible choices for each of the extrinsic/intrinsic groups leading to 2 * 2 * 6 = 24 possible conventions to specify a rotation sequence with three given angles.

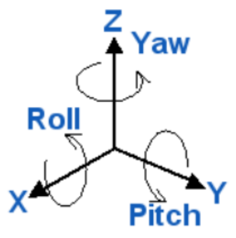
This is a mess - too many options!

Hence, when we deal with euler, we must specify rotations explicitly:

1. r (rotating = intrinsic) or s (static = extrinsic)
2. xyz or zyx or zxz, etc. (order of rotation axes)

The most widely used one is row-pitch-yaw convention

Row-Pitch-Yaw Convention



Roll (ϕ), pitch (θ), yaw (ψ) angles are used in aerospace engineering to specify rotation of an aircraft around the x, y, and z axes, respectively.

Now, elementary rotations can be represented as follows:

A rotation by an angle ϕ around the x-axis is represented by:

$$R_x(\phi) := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

A rotation by an angle θ around the y-axis is represented by:

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

A rotation by an angle ψ around the z-axis is represented by:

$$R_z(\psi) := \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If we are given some information about the roll, pitch, yaw, we can compute the R matrix as follows:

Intrinsic yaw (ψ), pitch (θ), roll (ϕ) rotation (*rzyx*):

- A rotation ψ about the original z-axis
- A rotation θ about the intermediate y-axis
- A rotation ϕ about the transformed x-axis

Extrinsic roll (ϕ), pitch (θ), yaw (ψ) rotation (*sxyz*):

- A rotation ϕ about the global x-axis
- A rotation θ about the global y-axis
- A rotation ψ about the global z-axis

Both conventions define the following body-to-world rotation:

$$\begin{aligned} R &= R_z(\psi)R_y(\theta)R_x(\phi) \\ &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \end{aligned}$$

Gimbal Lock

Ideally, with three variables, and three degrees of freedom, we should be able to represent all rotations but this is not true. This is gimbal lock. Alternatively, angle paramterizations are not one-to-one.

Example: if the pitch becomes $\theta = 90\,\text{deg}$, the roll and yaw become associated with the same degree of freedom and cannot be uniquely determined. Basically, $R = R_z(\phi)R_y(\pi/2)R_x(\psi + \delta) = R_z(\phi)R_y(\pi/2)R_x(\psi)$

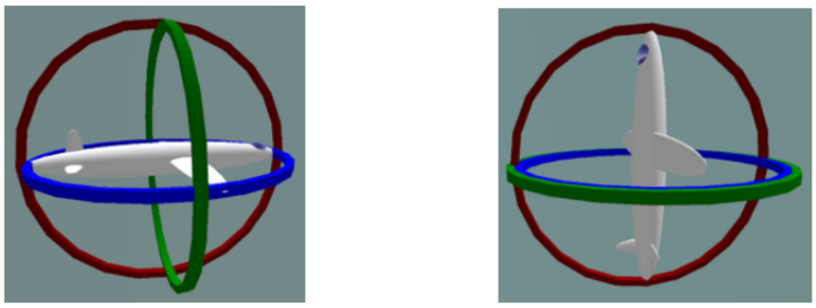


Figure 1.2: Look at what happens after the pitch - two axes align and we can't distinguish between roll and yaw.

Hence, two big problems with euler - it's confusing and gimbal lock.

1.4.2 Axis Angle Representation

Cross Product Review

The **cross product** of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ is also a vector in $\mathbb{R}^3$:

$$\mathbf{x} \times \mathbf{y} := \begin{bmatrix} x_2 y_3 - x_3 y_2 \\ x_3 y_1 - x_1 y_3 \\ x_1 y_2 - x_2 y_1 \end{bmatrix} = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \hat{\mathbf{x}}\mathbf{y}$$

$\hat{x}$ is the skew symmetric matrix used to represent the x that is used for cross product. It is also called the **hat map**.

The **hat map** $\hat{\cdot} : \mathbb{R}^3 \to \mathfrak{so}(3)$ transforms a vector $\mathbf{x} \in \mathbb{R}^3$ to a skew-symmetric matrix:

$$\hat{\mathbf{x}} := \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \qquad \hat{\mathbf{x}}^\top = -\hat{\mathbf{x}}$$

The vector space $\mathbb{R}^3$ and the space of skew-symmetric 3×3 matrices $so(3)$ are isomorphic, i.e., there exists a one-to-one map (the hat map) that preserves their structure.

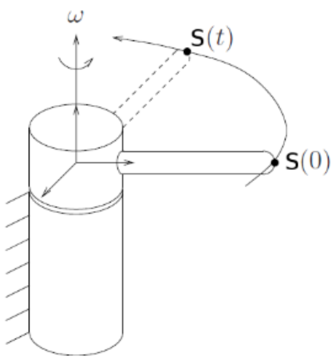
The **vee map** (v because its the upside down version of the hat) simply obtains x back from the skew-symmetric matrix $\hat{x}$.

Hat map properties: for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$, $A \in \mathbb{R}^{3\times 3}$:

- $\hat{\mathbf{x}}\mathbf{y} = \mathbf{x} \times \mathbf{y} = -\mathbf{y} \times \mathbf{x} = -\hat{\mathbf{y}}\mathbf{x}$
- $\hat{\mathbf{x}}^2 = \mathbf{x}\mathbf{x}^\top - \mathbf{x}^\top \mathbf{x}\, I$
- $\hat{\mathbf{x}}^{2k+1} = (-\mathbf{x}^\top \mathbf{x})^k \hat{\mathbf{x}}$
- $-\frac{1}{2}\operatorname{tr}(\hat{\mathbf{x}}\hat{\mathbf{y}}) = \mathbf{x}^\top \mathbf{y}$
- $\hat{\mathbf{x}}A + A^\top \hat{\mathbf{x}} = ((\operatorname{tr}(A)I - A)\mathbf{x})^\wedge$
- $\operatorname{tr}(\hat{\mathbf{x}}A) = \frac{1}{2}\operatorname{tr}(\hat{\mathbf{x}}(A - A^\top)) = -\mathbf{x}^\top (A - A^\top)^\vee$
- $(A\mathbf{x})^\wedge = \det(A) A^{-\top} \hat{\mathbf{x}} A^{-1}$

Coming back to axis-angle.

Let's assume that we have a stick on a cylinder that is rotating about the cylinder. Let us consider a point s at the end of this stick.



Let the point have a constant unit velocity, such that:

$$\dot{s}(t) = \eta \times s(t) = \hat{\eta} s(t)$$

Now, $s(t) = R_z(t)s(0)$.

This is simply a linear time-invariant system of ordinary differential equations determined by the skew symmetric matrix $\hat{\eta}$. The solution for this can be written as (this is a standard solution):

$$s(t) = \exp(\hat{\eta t} s(0))$$

$$s(t) = R_n(t)s(0)$$

We can just replace the t with θ,

$$s(t) = \exp(\hat{\eta\theta} s(0))$$

$$s(t) = R_n(\theta)s(0)$$

This is the exponential map:

Look at Hao Su's description of axis angle and the conversions there.

Exponential map exp : $\mathfrak{so}(3) \mapsto SO(3)$ maps a skew-symmetric matrix $\hat{\boldsymbol{\theta}}$ obtained from an axis-angle vector $\boldsymbol{\theta}$ to a rotation matrix R:

$$R = \exp(\hat{\boldsymbol{\theta}}) := \sum_{n=0}^{\infty} \frac{1}{n!}\hat{\boldsymbol{\theta}}^n = I + \hat{\boldsymbol{\theta}} + \frac{1}{2!}\hat{\boldsymbol{\theta}}^2 + \frac{1}{3!}\hat{\boldsymbol{\theta}}^3 + \dots$$

The skwigly so is simply the set of skew symmetric matrices. The exponential map maps every skew symmetric matix to a special orthogonal group rotation.

exp is not the same as taking exponential of every element, but rather:

$$\exp(A) = I + A + \frac{1}{2}AA + \frac{1}{3!}AAA$$

The exponential map is **surjective** but **not injective**: every element of $SO(3)$ can be generated from multiple elements of $\mathfrak{so}(3)$, e.g., any vector $(\|\boldsymbol{\theta}\| + 2\pi k)\frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}$ for integer k leads to the same R

The exponential map is **not commutative**: $e^{\hat{\boldsymbol{\theta}}_1}e^{\hat{\boldsymbol{\theta}}_2} \neq e^{\hat{\boldsymbol{\theta}}_2}e^{\hat{\boldsymbol{\theta}}_1} \neq e^{\hat{\boldsymbol{\theta}}_1+\hat{\boldsymbol{\theta}}_2}$, unless $\hat{\boldsymbol{\theta}}_1\hat{\boldsymbol{\theta}}_2 - \hat{\boldsymbol{\theta}}_2\hat{\boldsymbol{\theta}}_1 = 0$

Rodriguez Formula

It is the closed form expression for the exponential map from the skew symmetric space to special orthogonal group:

$$R = \exp(\hat{\boldsymbol{\theta}}) = I + \left(\frac{\sin\|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|}\right)\hat{\boldsymbol{\theta}} + \left(\frac{1-\cos\|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|^2}\right)\hat{\boldsymbol{\theta}}^2$$

An important property to derive this is that $\hat{x}\hat{x} = xx^T - x^TxI$. Further, any odd power of $\hat{x}$ can be written as some product of x.

The formula is derived using that $\hat{\boldsymbol{\theta}}^{2n+1} = (-\boldsymbol{\theta}^\top \boldsymbol{\theta})^n \hat{\boldsymbol{\theta}}$:

$$
\begin{aligned}
\exp(\hat{\boldsymbol{\theta}}) &= I + \sum_{n=1}^{\infty} \frac{1}{n!} \hat{\boldsymbol{\theta}}^n \\
&= I + \sum_{n=0}^{\infty} \frac{1}{(2n+1)!} \hat{\boldsymbol{\theta}}^{2n+1} + \sum_{n=0}^{\infty} \frac{1}{(2n+2)!} \hat{\boldsymbol{\theta}}^{2n+2} \\
&= I + \left(\sum_{n=0}^{\infty} \frac{(-1)^n \|\boldsymbol{\theta}\|^{2n}}{(2n+1)!} \right) \hat{\boldsymbol{\theta}} + \left(\sum_{n=0}^{\infty} \frac{(-1)^n \|\boldsymbol{\theta}\|^{2n}}{(2n+2)!} \right) \hat{\boldsymbol{\theta}}^2 \\
&= I + \left(\frac{\sin \|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|} \right) \hat{\boldsymbol{\theta}} + \left(\frac{1 - \cos \|\boldsymbol{\theta}\|}{\|\boldsymbol{\theta}\|^2} \right) \hat{\boldsymbol{\theta}}^2
\end{aligned}
$$

In the above equation, we split the infinite sum into odd powers and even powers respectively (look at the superscript).

Then, we subtitute the odd power of $\hat{\theta}$ with the equation that expresses it as a product of $\hat{\theta}$.

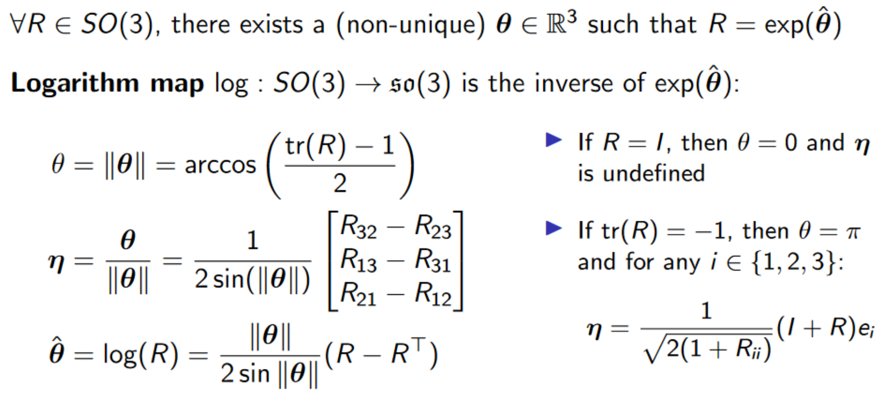
The final line is simply the expansion of sin and cos.

Why do we need it?

This is faster, closed form. Practically, doesn't matter but this is better obviously.

Log Map

There is a log map that simply inverts the exponential - it goes from SO group to the skew symmetric group.



$\forall R \in SO(3)$, there exists a (non-unique) $\boldsymbol{\theta} \in \mathbb{R}^3$ such that $R = \exp(\hat{\boldsymbol{\theta}})$

Logarithm map $\log : SO(3) \to \mathfrak{so}(3)$ is the inverse of $\exp(\hat{\boldsymbol{\theta}})$:

$$\theta = \|\boldsymbol{\theta}\| = \arccos\left(\frac{\operatorname{tr}(R) - 1}{2} \right)$$

$$\boldsymbol{\eta} = \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|} = \frac{1}{2\sin(\|\boldsymbol{\theta}\|)} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}$$

$$\hat{\boldsymbol{\theta}} = \log(R) = \frac{\|\boldsymbol{\theta}\|}{2\sin\|\boldsymbol{\theta}\|}(R - R^\top)$$

- If $R = I$, then $\theta = 0$ and $\boldsymbol{\eta}$ is undefined
- If $\operatorname{tr}(R) = -1$, then $\theta = \pi$ and for any $i \in \{1, 2, 3\}$:

$$\boldsymbol{\eta} = \frac{1}{\sqrt{2(1 + R_{ii})}}(I + R)e_i$$

Figure 1.3: All this is derived from the rodriguez formula

The log map is not unique - so it's not a true inverse. R is not truly invertible.

The matrix exponential "integrates" $\hat{\theta} \in se(3)$ for one second; the matrix logarithm "differentiates" $R \in SO(3)$ to obtain $\hat{\theta} \in se(3)$.

Note that gimbal lock is possible in axis-angle representation as well - Think of an example where this can happen it's not immediately clear to me.

1.4.3 Quaternion

We now look for a representation that doesn't have singularity. Quaternions are a solution for this, and are an extension of the complex number idea that we had disucssed earlier.

Quaternions: $\mathbb{H} = \mathbb{C} + \mathbb{C}j$ generalize complex numbers $\mathbb{C} = \mathbb{R} + \mathbb{R}i$

$$\mathbf{q} = q_s + q_1 i + q_2 j + q_3 k = [q_s,\ \mathbf{q}_v] \qquad ij = -ji = k,\ i^2 = j^2 = k^2 = -1$$

Here, $\mathbb{H}$ is the quaternion space. $\mathbb{H}_*$ is the space of unit-norm quaternions:

$$\mathbb{H}_* := q \in \mathbb{H} | q_s^2 + q_v^T q_v = 1$$

A vector with a unit-norm is essentially a sphere, so quaternions are 4-D spheres. The surface of the sphere is 3-D dimensional. Hence, every surface of the sphere is a rotation matrix. More accurately, only half of the space is a rotation matrix.

A rotation matrix $R \in SO(3)$ can be obtained from a unit quaternion $\mathbf{q}$:

$$R(\mathbf{q}) = E(\mathbf{q})G(\mathbf{q})^\top \qquad \begin{aligned} E(\mathbf{q}) &= [-\mathbf{q}_v,\ q_s I + \hat{\mathbf{q}}_v] \\ G(\mathbf{q}) &= [-\mathbf{q}_v,\ q_s I - \hat{\mathbf{q}}_v] \end{aligned}$$

The fact that only half the sphere is relevant is called double covering of quaternions - $R(q) = R(-q)$.

We can also convert quaternion from axis-angle representation.

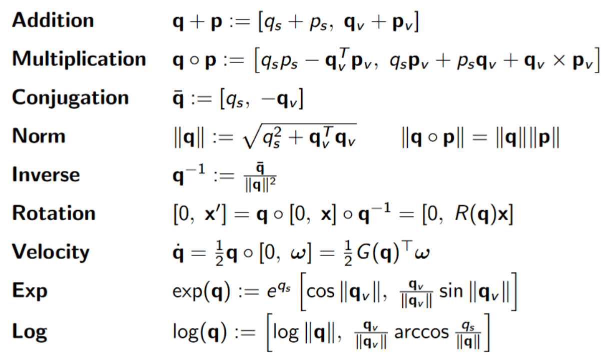
A rotation around a unit axis $\boldsymbol{\eta} := \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|} \in \mathbb{R}^3$ by angle $\theta := \|\boldsymbol{\theta}\|$ can be represented by a unit quaternion:

$$\mathbf{q} = \left[\cos\left(\frac{\theta}{2}\right),\ \sin\left(\frac{\theta}{2}\right)\boldsymbol{\eta}\right] \in \mathbb{H}_*$$

A rotation around a unit axis $\boldsymbol{\eta} \in \mathbb{R}^3$ by angle θ can be recovered from a unit quaternion $\mathbf{q} = [q_s,\ \mathbf{q}_v] \in \mathbb{H}_*$:

$$\theta = 2\arccos(q_s) \qquad \boldsymbol{\eta} = \begin{cases} \frac{1}{\sin(\theta/2)}\mathbf{q}_v, & \text{if } \theta \neq 0 \\ 0, & \text{if } \theta = 0 \end{cases}$$

Quaternion Operations

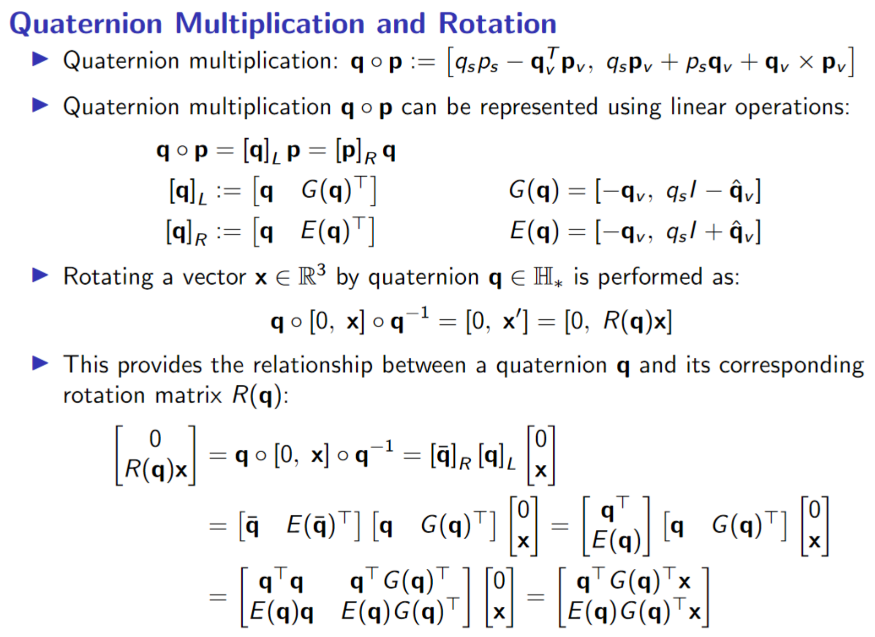


Addition	$\mathbf{q}+\mathbf{p} := [q_s + p_s,\ \mathbf{q}_v + \mathbf{p}_v]$
Multiplication	$\mathbf{q}\circ\mathbf{p} := \left[q_s p_s - \mathbf{q}_v^T \mathbf{p}_v,\ q_s \mathbf{p}_v + p_s \mathbf{q}_v + \mathbf{q}_v \times \mathbf{p}_v\right]$
Conjugation	$\bar{\mathbf{q}} := [q_s,\ -\mathbf{q}_v]$
Norm	$\lVert\mathbf{q}\rVert := \sqrt{q_s^2 + \mathbf{q}_v^T \mathbf{q}_v} \qquad \lVert\mathbf{q}\circ\mathbf{p}\rVert = \lVert\mathbf{q}\rVert\lVert\mathbf{p}\rVert$
Inverse	$\mathbf{q}^{-1} := \frac{\bar{\mathbf{q}}}{\lVert\mathbf{q}\rVert^2}$
Rotation	$[0,\ \mathbf{x}'] = \mathbf{q}\circ[0,\ \mathbf{x}]\circ\mathbf{q}^{-1} = [0,\ R(\mathbf{q})\mathbf{x}]$
Velocity	$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q}\circ[0,\ \boldsymbol{\omega}] = \frac{1}{2}G(\mathbf{q})^\top\boldsymbol{\omega}$
Exp	$\exp(\mathbf{q}) := e^{q_s}\left[\cos\lVert\mathbf{q}_v\rVert,\ \frac{\mathbf{q}_v}{\lVert\mathbf{q}_v\rVert}\sin\lVert\mathbf{q}_v\rVert\right]$
Log	$\log(\mathbf{q}) := \left[\log\lVert\mathbf{q}\rVert,\ \frac{\mathbf{q}_v}{\lVert\mathbf{q}_v\rVert}\arccos\frac{q_s}{\lVert\mathbf{q}\rVert}\right]$

For unit quaternions, the inverse is the same as the conjugate.

The rotation formula above, shows how we can rotate a vector given a quaternion. Typically we do $x' = Rx$. If x is the body position, R is the body frame, then x' is the position in the world frame. Let us convert our 3D position to a 4D vector: $\mathbf{x} \rightarrow [0, \mathbf{x}]$. Then, we can rotate it using $q \circ [0, \mathbf{x}] \circ \bar{q}$.

The derivation for quaternion multiplication and rotation can be seen as:



Quaternion Multiplication and Rotation

- Quaternion multiplication: $\mathbf{q}\circ\mathbf{p} := \left[q_s p_s - \mathbf{q}_v^T \mathbf{p}_v,\ q_s \mathbf{p}_v + p_s \mathbf{q}_v + \mathbf{q}_v \times \mathbf{p}_v\right]$
- Quaternion multiplication $\mathbf{q}\circ\mathbf{p}$ can be represented using linear operations:

$$\mathbf{q}\circ\mathbf{p} = [\mathbf{q}]_L\,\mathbf{p} = [\mathbf{p}]_R\,\mathbf{q}$$

$$[\mathbf{q}]_L := \begin{bmatrix}\mathbf{q} & G(\mathbf{q})^\top\end{bmatrix} \qquad G(\mathbf{q}) = [-\mathbf{q}_v,\ q_s I - \hat{\mathbf{q}}_v]$$

$$[\mathbf{q}]_R := \begin{bmatrix}\mathbf{q} & E(\mathbf{q})^\top\end{bmatrix} \qquad E(\mathbf{q}) = [-\mathbf{q}_v,\ q_s I + \hat{\mathbf{q}}_v]$$

- Rotating a vector $\mathbf{x}\in\mathbb{R}^3$ by quaternion $\mathbf{q}\in\mathbb{H}_*$ is performed as:

$$\mathbf{q}\circ[0,\ \mathbf{x}]\circ\mathbf{q}^{-1} = [0,\ \mathbf{x}'] = [0,\ R(\mathbf{q})\mathbf{x}]$$

- This provides the relationship between a quaternion $\mathbf{q}$ and its corresponding rotation matrix $R(\mathbf{q})$:

$$\begin{aligned}\begin{bmatrix}0\\ R(\mathbf{q})\mathbf{x}\end{bmatrix} &= \mathbf{q}\circ[0,\ \mathbf{x}]\circ\mathbf{q}^{-1} = [\bar{\mathbf{q}}]_R\,[\mathbf{q}]_L\begin{bmatrix}0\\ \mathbf{x}\end{bmatrix}\\ &= \begin{bmatrix}\bar{\mathbf{q}} & E(\bar{\mathbf{q}})^\top\end{bmatrix}\begin{bmatrix}\mathbf{q} & G(\mathbf{q})^\top\end{bmatrix}\begin{bmatrix}0\\ \mathbf{x}\end{bmatrix} = \begin{bmatrix}\mathbf{q}^\top\\ E(\mathbf{q})\end{bmatrix}\begin{bmatrix}\mathbf{q} & G(\mathbf{q})^\top\end{bmatrix}\begin{bmatrix}0\\ \mathbf{x}\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{q}^\top\mathbf{q} & \mathbf{q}^\top G(\mathbf{q})^\top\\ E(\mathbf{q})\mathbf{q} & E(\mathbf{q})G(\mathbf{q})^\top\end{bmatrix}\begin{bmatrix}0\\ \mathbf{x}\end{bmatrix} = \begin{bmatrix}\mathbf{q}^\top G(\mathbf{q})^\top\mathbf{x}\\ E(\mathbf{q})G(\mathbf{q})^\top\mathbf{x}\end{bmatrix}\end{aligned}$$

In ‘python’, we can use ‘transform3D’ to convert between representations and perform operations.

1.5 Rigid Body Poses

Let $\{B\}$ be the bodu frame whose position an dorientation with respect to the world frame $\{W\}$ are $\mathbf{p} \in \mathbb{R}^3$ and $R \in SO(3)$, respectively.

The coordinates of a point $\mathbf{s}_B \in \mathbb{R}^3$ can be converted to the world frame by first rotating the point and then translating it to the world frame:

$$\mathbf{s}_W = R\mathbf{s}_B + \mathbf{p}$$

This is also known as an affine transform, and we can combine the R and $\mathbf{p}$ into one matrix to make this transformation easier using homogeneous coordinates. For instance,

$$\begin{bmatrix} R & p \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}_B \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{s}_W \\ 1 \end{bmatrix}$$

We can see that our matrix here, $T \in \mathbb{R}^{4\times4}$. For representation purpose, we will now use underline to symbolise homogeneous coordinates $\underline{\mathbf{s}_B}$.

The pose of a rigid body can be described by a matrix T in the special Euclidian group.

$$SE(3) := \left\{ T = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix} \middle| \; R \in SO(3), \mathbf{p} \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{4\times4}$$

The pose of a rigid body T specifies a transformation from the body frame $\{B\}$ to the world frame $\{W\}$:

$$_{\{W\}}T_{\{B\}} := \begin{bmatrix} _{\{W\}}R_{\{B\}} & _{\{W\}}\mathbf{p}_{\{B\}} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

A point with body-frame coordinates $\mathbf{s}_B$, has world-frame coordinates:

$$\mathbf{s}_W = R\mathbf{s}_B + \mathbf{p} \quad \text{equivalent to} \quad \begin{bmatrix} \mathbf{s}_W \\ 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}_B \\ 1 \end{bmatrix}$$

A point with world-frame coordinates $\mathbf{s}_W$, has body-frame coordinates:

$$\begin{bmatrix} \mathbf{s}_B \\ 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{s}_W \\ 1 \end{bmatrix} = \begin{bmatrix} R^\top & -R^\top\mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}_W \\ 1 \end{bmatrix}$$

The inverse transform can be described as

$$\begin{bmatrix} R_T & -R^T p \\ 0 & 1 \end{bmatrix}$$

Composing Transformations

We also have the liberty to compose transformations, and this can be done in multiple ways via intermediate frames.

For example: Let T_k be the pose of a robot at time k, and let T_{k+1} be the pose at time $k+1$. The transformation from k to $k+1$ can be found as ${}_kT_{k+1} = {}_kT_{ww}T_{k+1} = T_k^{-1}T_{k+1}$.

	Rotation $SO(3)$	**Pose** $SE(3)$
Representation	$R : \begin{cases} R^T R = I \\ \det(R) = 1 \end{cases}$	$T = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
Transformation	$\mathbf{s}_W = R\mathbf{s}_B$	$\mathbf{s}_W = R\mathbf{s}_B + \mathbf{p}$
Inverse	$R^{-1} = R^\top$	$T^{-1} = \begin{bmatrix} R^\top & -R^\top\mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
Composition	${}_WR_B = {}_WR_A\,{}_AR_B$	${}_WT_B = {}_WT_A\,{}_AT_B$

Chapter 2

Robot Motion and Observation Models

Let us define R as a function of time, which gives us the orientation at time t. The trajectory $R(t)$ of the continuous rotation motion satisfies $R^T R(t) = I$ Let us define:

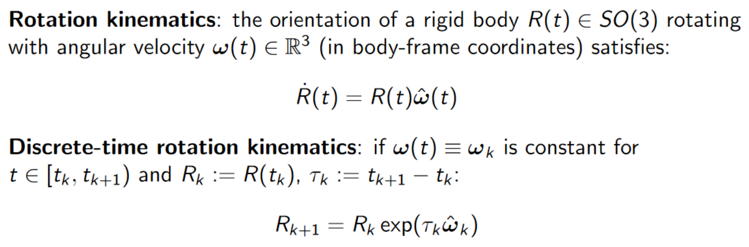
$$\frac{\partial R(t)}{\partial t} = \dot{R}(t)$$

We also say that:

$$\dot{R}^T(t)R(t) + R^T(t)\dot{R}(t) = 0$$

Since $R^T(t)\dot{R}(t)$ is skew symmetric, there exists $\omega(t) \in \mathbb{R}^3$, such that $R^T(t)\dot{R}(t) = \hat{\omega}(t)$.

$$R(t) = R(0)\exp(t\hat{\omega})$$



Rotation kinematics: the orientation of a rigid body $R(t) \in SO(3)$ rotating with angular velocity $\boldsymbol{\omega}(t) \in \mathbb{R}^3$ (in body-frame coordinates) satisfies:

$$\dot{R}(t) = R(t)\hat{\boldsymbol{\omega}}(t)$$

Discrete-time rotation kinematics: if $\boldsymbol{\omega}(t) \equiv \boldsymbol{\omega}_k$ is constant for $t \in [t_k, t_{k+1})$ and $R_k := R(t_k)$, $\tau_k := t_{k+1} - t_k$:

$$R_{k+1} = R_k \exp(\tau_k \hat{\boldsymbol{\omega}}_k)$$

We can do the same thing with quaternions:

Quaternion kinematics: the orientation of a rigid body $\mathbf{q}(t) \in \mathbb{H}_*$ rotating with angular velocity $\boldsymbol{\omega}(t) \in \mathbb{R}^3$ (in body-frame coordinates) satisfies:

$$\dot{\mathbf{q}}(t) = \mathbf{q}(t) \circ [0, \boldsymbol{\omega}(t)/2]$$

Discrete-time quaternion kinematics: if $\boldsymbol{\omega}(t) \equiv \boldsymbol{\omega}_k$ is constant for $t \in [t_k, t_{k+1})$ and $\mathbf{q}_k := \mathbf{q}(t_k)$, $\tau_k := t_{k+1} - t_k$:

$$\mathbf{q}_{k+1} = \mathbf{q}_k \circ \exp([0, \tau_k \boldsymbol{\omega}_k/2])$$

To represent this for a general pose, we must consider the linear velocity and angular velocity, which is also called twist.

Pose kinematics: the pose of a rigid body $T(t) \in SE(3)$ moving with **twist** (generalized velocity) $\boldsymbol{\zeta}(t) = \begin{bmatrix} \mathbf{v}(t) \\ \boldsymbol{\omega}(t) \end{bmatrix} \in \mathbb{R}^6$ (in body-frame coordinates) satisfies:

$$\dot{T}(t) = T(t)\hat{\boldsymbol{\zeta}}(t) \qquad \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix}^{\wedge} := \begin{bmatrix} \hat{\boldsymbol{\omega}} & \mathbf{v} \\ \mathbf{0} & 0 \end{bmatrix}$$

Discrete-time pose kinematics: if $\boldsymbol{\zeta}(t) \equiv \boldsymbol{\zeta}_k$ is constant for $t \in [t_k, t_{k+1})$ and $T_k := T(t_k)$, $\tau_k := t_{k+1} - t_k$:

$$T_{k+1} = T_k \exp(\tau_k \hat{\boldsymbol{\zeta}}_k)$$

If there are forces acting on the pose (we may not use this in the course), then it's called pose dynamics.

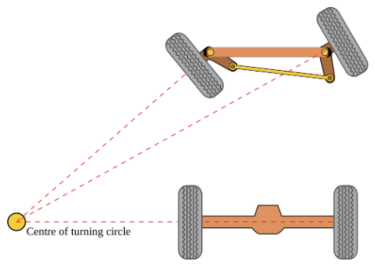
Pose dynamics: the pose $T(t) \in SE(3)$ and twist $\boldsymbol{\zeta}(t) \in \mathbb{R}^6$ of a rigid body with mass $m \in \mathbb{R}_{>0}$ and moment of inertia $J \in \mathbb{R}^{3\times 3}$, moving with **wrench** (generalized force) $\mathbf{w}(t) = \begin{bmatrix} \mathbf{f}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix} \in \mathbb{R}^6$ (in body-frame coordinates) satisfies:

$$\dot{T}(t) = T(t)\hat{\boldsymbol{\zeta}}(t) \qquad M := \begin{bmatrix} mI & 0 \\ 0 & J \end{bmatrix}$$

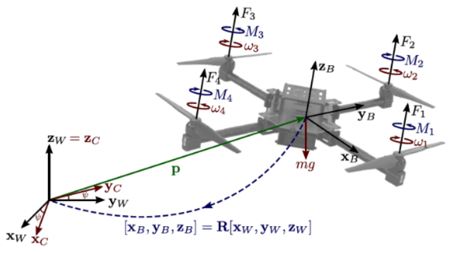
$$M\dot{\boldsymbol{\zeta}}(t) = \overset{\curlywedge}{\boldsymbol{\zeta}}(t)^\top M\boldsymbol{\zeta}(t) + \mathbf{w}(t) \qquad \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix}^{\curlywedge} := \begin{bmatrix} \hat{\boldsymbol{\omega}} & \hat{\mathbf{v}} \\ \mathbf{0} & \hat{\boldsymbol{\omega}} \end{bmatrix}$$

Here $\mathbf{f}$ is the force, and τ is the torque.

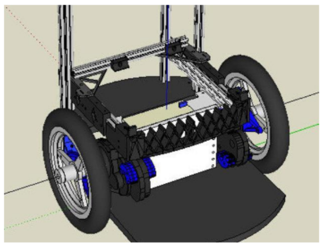
2.1 Motion Models



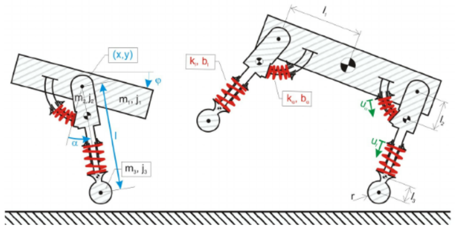
Ackermann Drive



Quadrotor



Differential Drive



Spring-loaded Gait

We will consider the ackerman and differentiable drive robots primarily.

There are a few variables describing the robot system:

1. time t
2. state x (position, orientation)
3. control input u (velocity, force)
4. disturbace w (tire slip, wind)

A motion model is a function f relating the current state $\mathbf{x}$ and input $\mathbf{u}$ of a robot with its state change:

- Continuous: $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$
- Discrete: $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$

If the robot motion is affected disturbance $\mathbf{w}$ modeled as a random variable, then the state $\mathbf{x}$ is also a random variable described either:

- in function form: $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t)$ or
- with the probability density function $p_f(\cdot \mid \mathbf{x}_t, \mathbf{u}_t)$ of $\mathbf{x}_{t+1}$

2.2 Odometry

Consider a rigid-body robot with state $\mathbf{x}_t = T_t \in SE(3)$ capturing the robot pose in the world frame $\{W\}$ at time t.

Determining the change of pose using sensors on the robot is known as odometry, which is done using onboard sensors.

Odometry: onboard sensors (camera, lidar, encoders, imu, etc.) may be used to estimate the relative pose of the robot body frame at time $t+1$ with respect to the body frame at time t:

$$\mathbf{u}_t = {}_tT_{t+1} = \begin{bmatrix} {}_tR_{t+1} & {}_t\mathbf{p}_{t+1} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in SE(3)$$

Odometry-based motion model: given the robot state $\mathbf{x}_t$ and input $\mathbf{u}_t$ at time t, the state at time $t+1$ satisfies:

$$T_{t+1} = \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) := \mathbf{x}_t\mathbf{u}_t = T_t\, {}_tT_{t+1}$$

Given an initial pose $\mathbf{x}_0$ and odometry measurements $\mathbf{u}_0, \ldots, \mathbf{u}_t$, the robot pose at time $t+1$ can be estimated as:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})\mathbf{u}_t = \ldots = \mathbf{x}_0\mathbf{u}_0\mathbf{u}_1 \cdots \mathbf{u}_t$$

The odometry estimate will drift however over time, because small measurements errors in each $\mathbf{u}$ will accumulate.

Now, if our state $X_k = T_k$, is given an input u_k, which is the twist $\left(\begin{bmatrix} v_k \\ \omega_k \end{bmatrix}\right)$.

Now, we can compute the new pose using:

$$T_{k+1} = T_k \exp(\tau_k \hat{\zeta}_k)$$

Where teh fancy symbol is the twist, τ is the $t_{k+1} - t_k$.

2.3 Differential-Drive Kinemantic Model

This robot cannot move sideways - or its degrees of freedom are limited - so it is a non-holonomic robot.

State: $\mathbf{x} = (\mathbf{p}, \theta)$, where $\mathbf{p} = (x, y) \in \mathbb{R}^2$ is the position and $\theta \in (-\pi, \pi]$ is the orientation (yaw angle) in the world frame

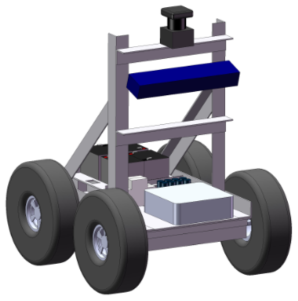
Control: $\mathbf{u} = (v, \omega)$, where $v \in \mathbb{R}$ is the linear velocity and $\omega \in \mathbb{R}$ is the angular velocity (yaw rate) in the body frame

Continuous-time model:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(\mathbf{x}, \mathbf{u}) := \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix}$$

Obtained from 2D pose kinematics with body twist $\boldsymbol{\zeta} = (v, 0, \omega)^\top$:

$$\begin{bmatrix} \dot{R}(\theta) & \dot{\mathbf{p}} \\ \mathbf{0} & 0 \end{bmatrix} = \begin{bmatrix} R(\theta) & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} 0 & -\omega & v \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



We can make a discrete version of this using euler discretization. This means that we approximate our derivative $\dot{s} = \frac{s_{k+1} - sk}{\tau_k}$. There is also an exact integration that is possible.

The *sinc* function is $\frac{\sin(x)}{x}$.

Euler discretization over time interval of length τ_t:

$$\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f_d(\mathbf{x}_t, \mathbf{u}_t) := \mathbf{x}_t + \tau_t \begin{bmatrix} v_t \cos(\theta_t) \\ v_t \sin(\theta_t) \\ \omega_t \end{bmatrix}$$

Exact integration over time interval of length τ_t:

$$\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f_d(\mathbf{x}_t, \mathbf{u}_t) := \mathbf{x}_t + \tau_t \begin{bmatrix} v_t \operatorname{sinc}\left(\frac{\omega_t \tau_t}{2}\right) \cos\left(\theta_t + \frac{\omega_t \tau_t}{2}\right) \\ v_t \operatorname{sinc}\left(\frac{\omega_t \tau_t}{2}\right) \sin\left(\theta_t + \frac{\omega_t \tau_t}{2}\right) \\ \omega_t \end{bmatrix}$$

The exact integration is equivalent to the discrete-time pose kinematics:

$$\begin{bmatrix} R(\theta_{t+1}) & \mathbf{p}_{t+1} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} R(\theta_t) & \mathbf{p}_t \\ \mathbf{0} & 1 \end{bmatrix} \exp\left(\tau_t \begin{bmatrix} 0 & -\omega_t & v_t \\ \omega_t & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right)$$