CS 3630, Fall 2026

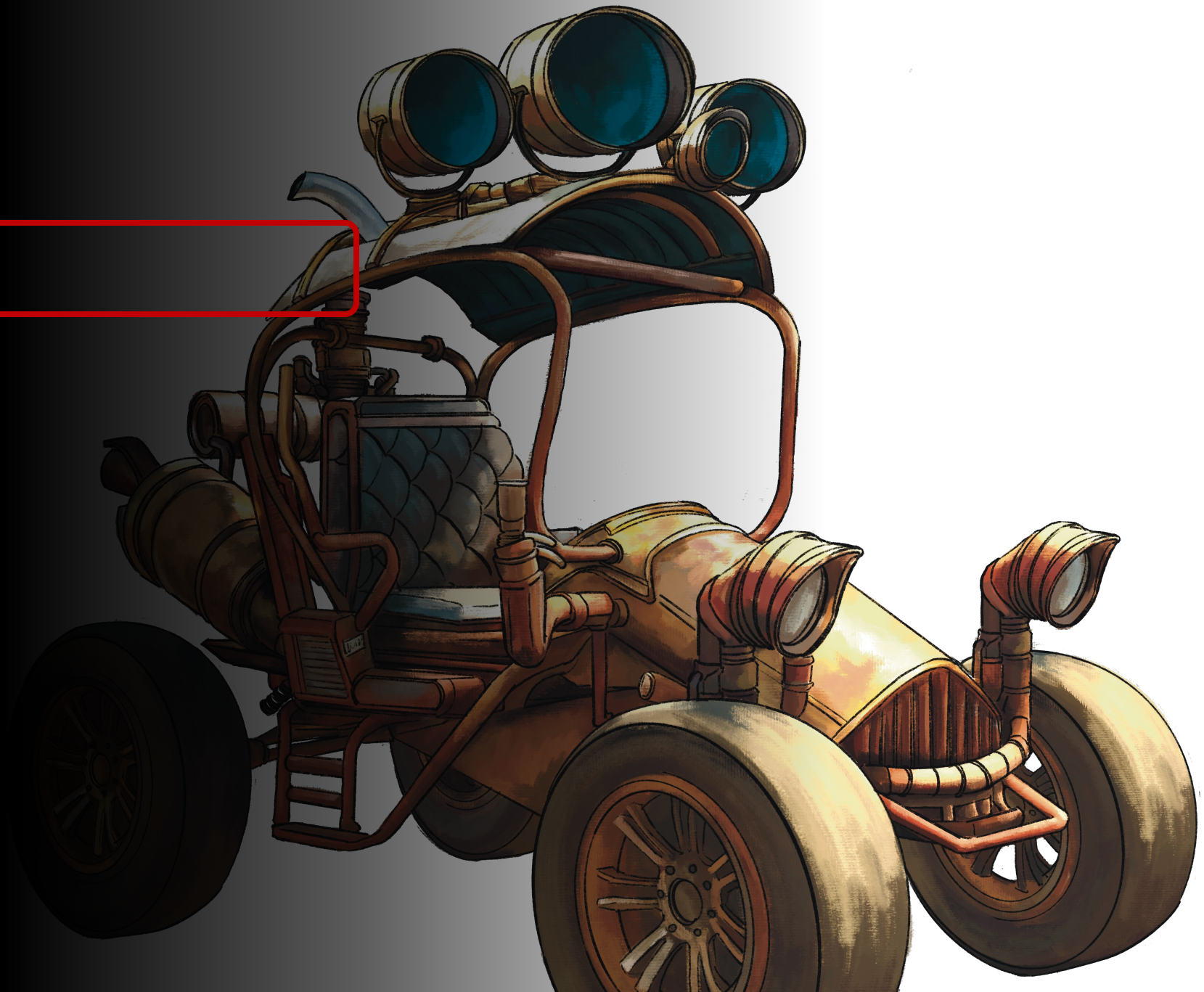
Lecture 23:

SE(2) and Ackermann Steering



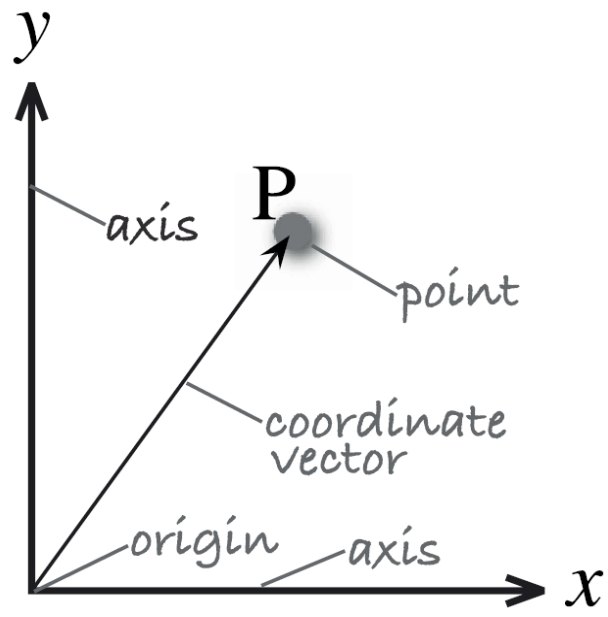
Outline

1. Reference Frames
2. Rotations and SO(2)
3. Poses and SE(2)
4. Car Kinematics
5. Ackermann Steering
6. Differential Kinematics



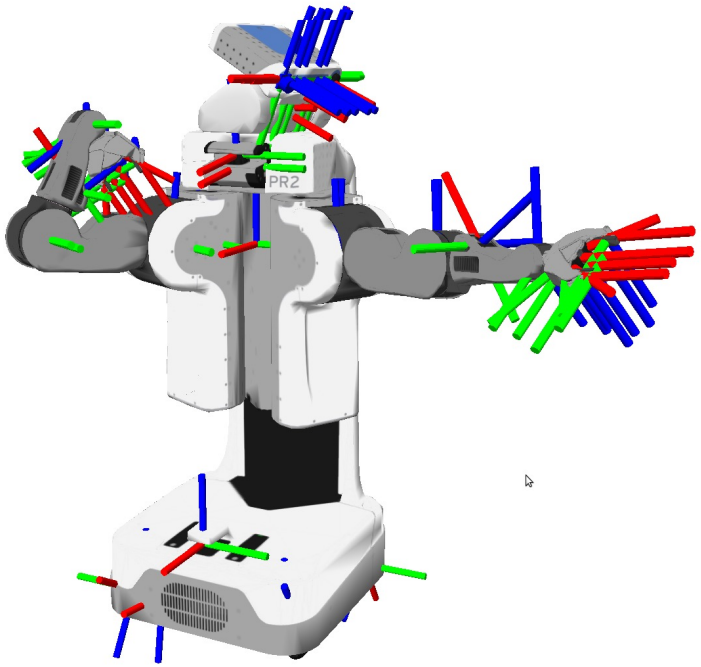
Reference Frames

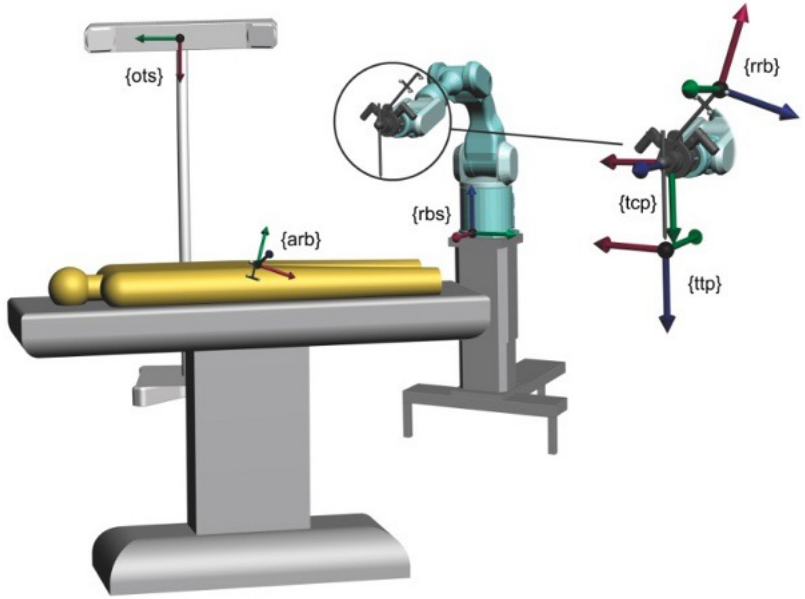
- Robotics is all about management of reference frames
  - **Perception** is about estimation of reference frames
  - **Planning** is how to move reference frames
  - **Control** is the implementation of trajectories for reference frames
- The relation between references frames is essential to a successful system



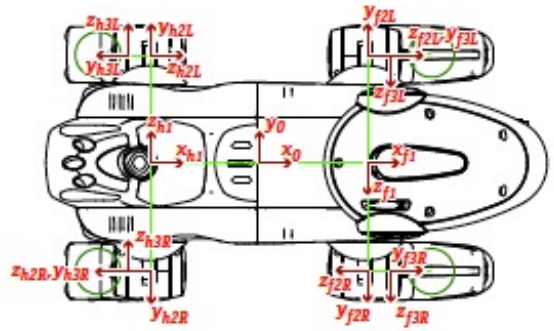
Examples of the types of reference frames we're talking about

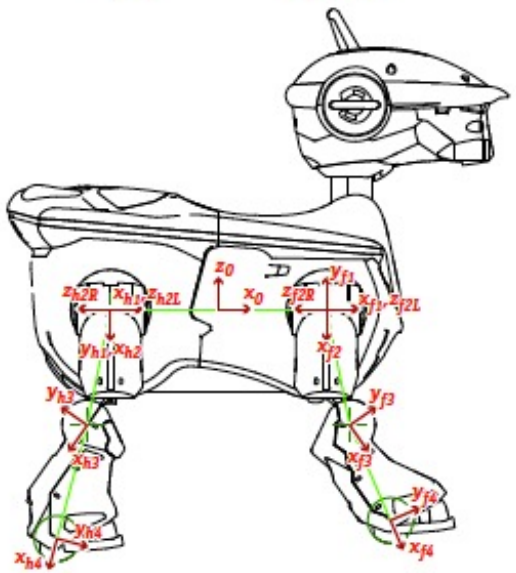
We rigidly attach coordinate frames to objects of interest. To specify the position and orientation of the object, we merely specify the position and orientation of the attached coordinate frame.

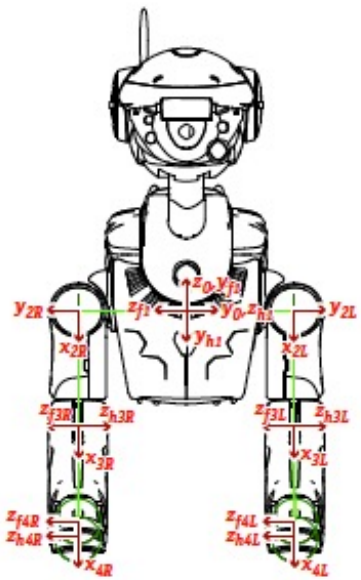


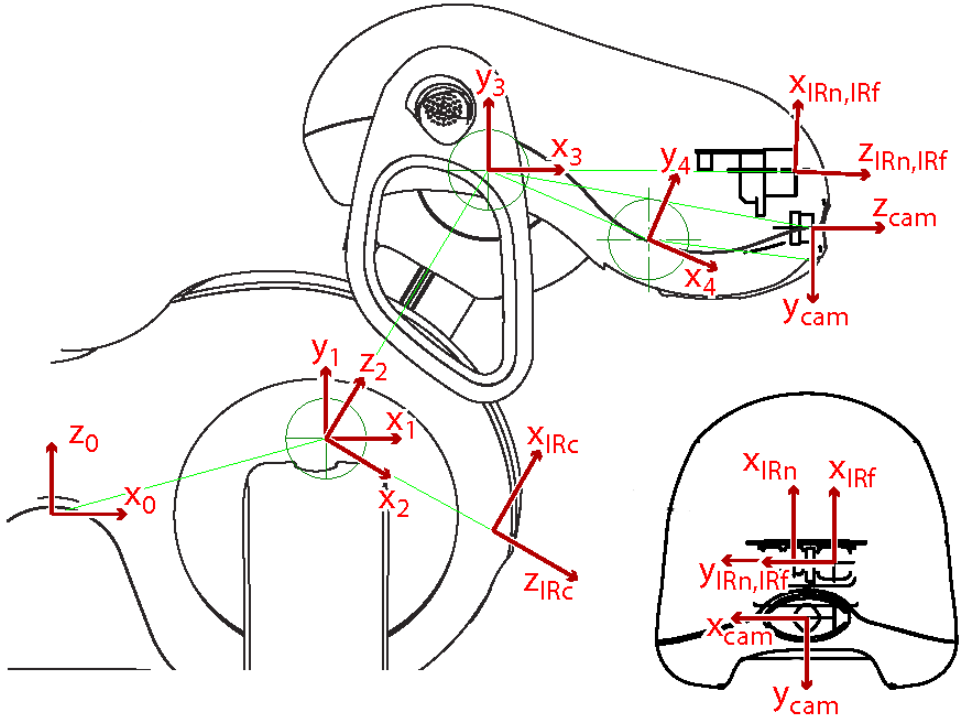










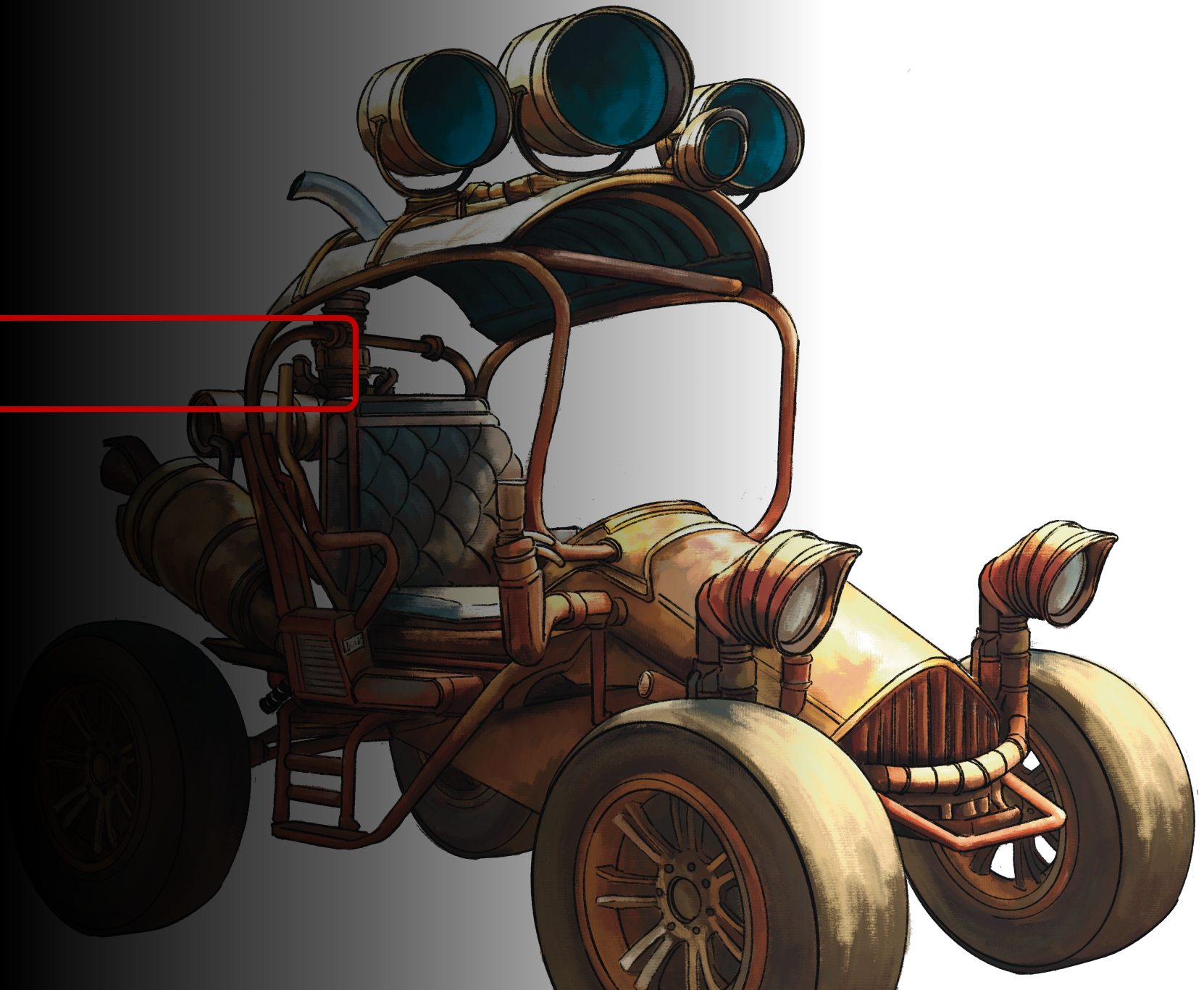


- The relationship between frames is often very simple to define, as in the case when two frames are related by the motion of a single joint/motor.
- For example, the upper and lower leg of the dog robot are related by a single motor at the knee.

Today – we consider only the case of 2D reference frames, corresponding to mobile robots moving in the plane.

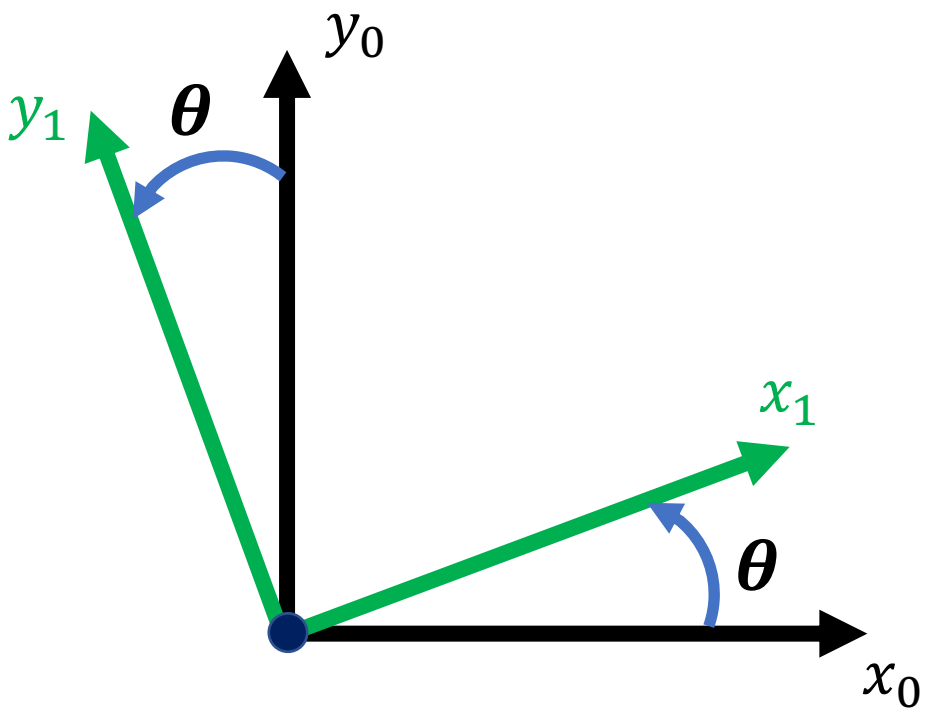
Outline

1. Reference Frames
2. Rotations and SO(2)
3. Poses and SE(2)
4. Car Kinematics
5. Ackermann Steering
6. Differential Kinematics



Specifying Orientation in the Plane

Given two coordinate frames with a common origin, how should we describe the orientation of Frame 1 w.r.t. Frame 0?

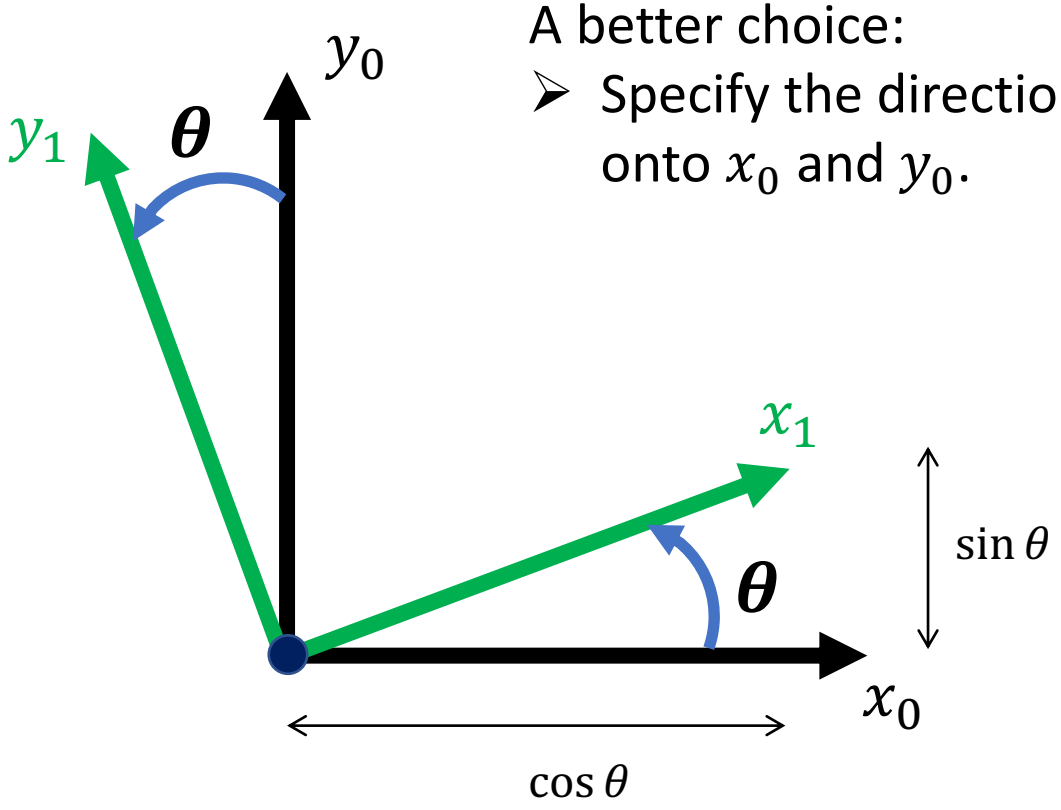


The obvious choice is to merely use the angle θ.
This isn't a great idea for two reasons:

- We have problems at $\theta = 2\,\pi - \epsilon$. For ϵ near 0, we approach a discontinuity: for small change in ϵ, we can have a large change in θ.
- This approach does not generalize to rotations in three dimensions (and not all robots live in the plane).

Specifying Orientation in the Plane

Given two coordinate frames with a common origin, how should we describe the orientation of Frame 1 w.r.t. Frame 0?



A better choice:

- Specify the directions of x_1 and y_1 with respect to Frame 0 by projecting onto x_0 and y_0.

$$x_1^0 = \begin{bmatrix} x_1 \cdot x_0 \\ x_1 \cdot y_0 \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

Notation: x_1^0 denotes the x-axis of Frame 1, specified w.r.t Frame 0.

$$y_1^0 = \begin{bmatrix} y_1 \cdot x_0 \\ y_1 \cdot y_0 \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

We obtain y_1^0 in the same way.

Rotation Matrices (rotation in the plane)

We combine these two vectors to obtain a ***rotation matrix***: $R_1^0 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

All rotation matrices have certain properties:

1. The two columns are each unit vectors.
2. The two columns are orthogonal, i.e., $c_1 \cdot c_2 = 0$.
3. $\det R = +1$

For such matrices $R^{-1} = R^T$

- The first two properties imply that the matrix R is ***orthogonal***.
- The third property implies that the matrix is ***special***! (After all, there are plenty of orthogonal matrices whose determinant is -1, not at all special.)

The collection of 2×2 rotation matrices is called the ***Special Orthogonal Group of order 2***, or, more commonly $\boldsymbol{SO(2)}$.

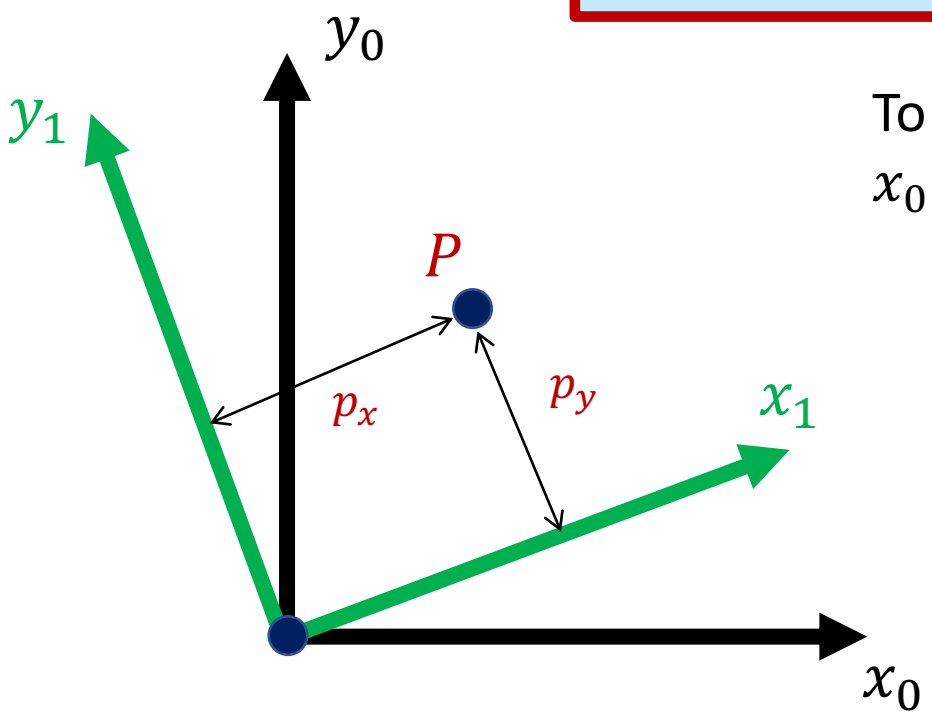
This concept generalizes to $\boldsymbol{SO(n)}$ for $n\times n$ rotation matrices.

Coordinate Transformations (rotation only)

Suppose a point P is rigidly attached to coordinate Frame 1, with coordinates given by $P^1 = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$.

We can express the location of the point P in terms of its coordinates

$$P = p_x x_1 + p_y y_1$$



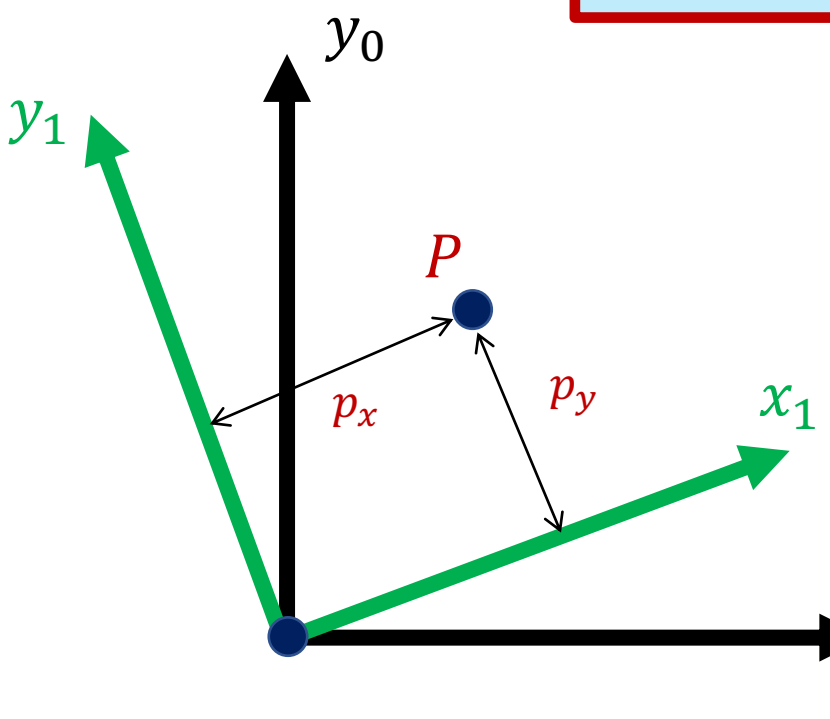
To obtain the coordinates of P w.r.t. Frame 0, we project P onto the x_0 and y_0 axes:

Coordinate Transformations (rotation only)

Suppose a point P is rigidly attached to coordinate Frame 1, with coordinates given by $P^1 = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$.

We can express the location of the point P in terms of its coordinates

$$P = p_x x_1 + p_y y_1$$



To obtain the coordinates of P w.r.t. Frame 0, we project P onto the x_0 and y_0 axes:

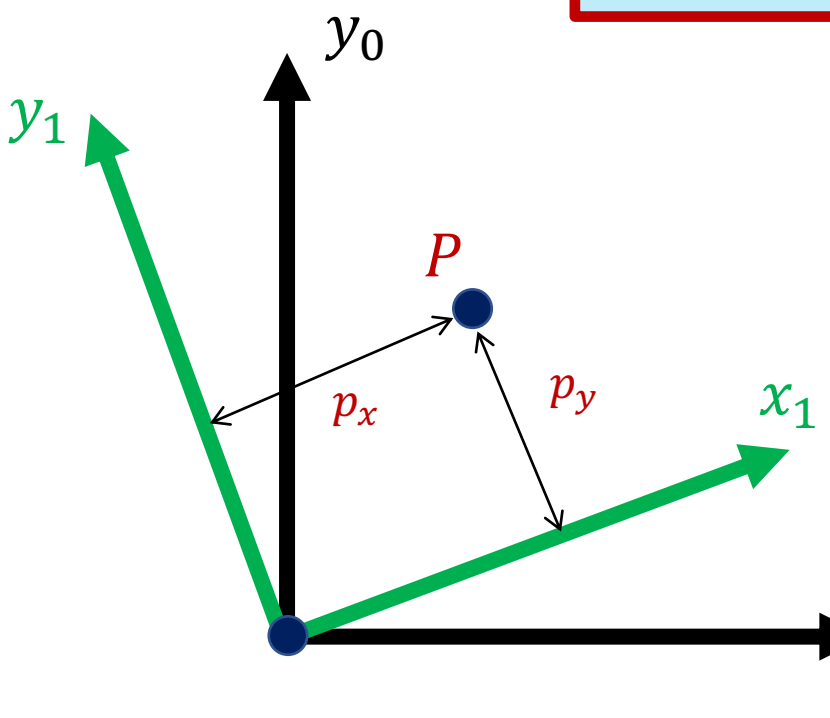
$$P^0 = \begin{bmatrix} P \cdot x_0 \\ P \cdot y_0 \end{bmatrix} =$$

Coordinate Transformations (rotation only)

Suppose a point P is rigidly attached to coordinate Frame 1, with coordinates given by ${}^1P = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$.

We can express the location of the point P in terms of its coordinates

$$P = p_x x_1 + p_y y_1$$



To obtain the coordinates of P w.r.t. Frame 0, we project P onto the x_0 and y_0 axes:

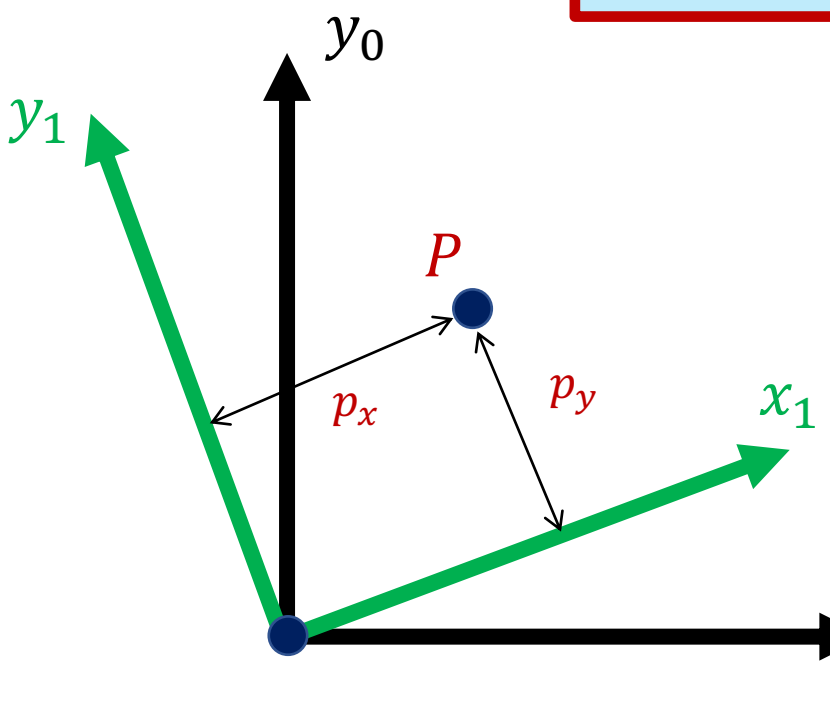
$$P^0 = \begin{bmatrix} P \cdot x_0 \\ P \cdot y_0 \end{bmatrix} = \begin{bmatrix} (p_x x_1 + p_y y_1) \cdot x_0 \\ (p_x x_1 + p_y y_1) \cdot y_0 \end{bmatrix} =$$

Coordinate Transformations (rotation only)

Suppose a point P is rigidly attached to coordinate Frame 1, with coordinates given by ${}^1P = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$.

We can express the location of the point P in terms of its coordinates

$$P = p_x x_1 + p_y y_1$$



To obtain the coordinates of P w.r.t. Frame 0, we project P onto the x_0 and y_0 axes:

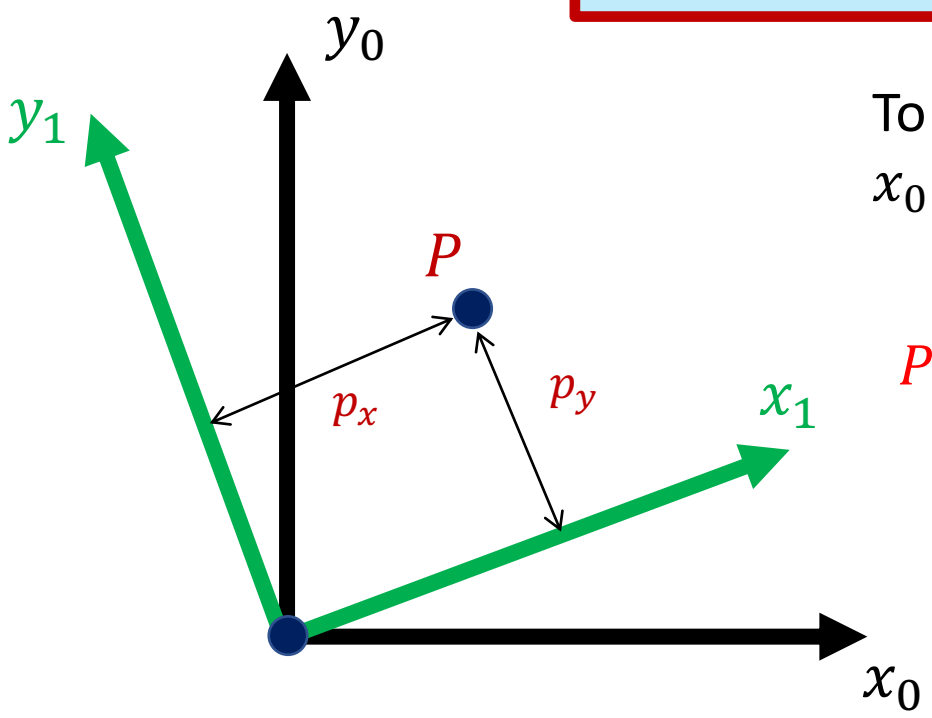
$$P^0 = \begin{bmatrix} P \cdot x_0 \\ P \cdot y_0 \end{bmatrix} = \begin{bmatrix} (p_x x_1 + p_y y_1) \cdot x_0 \\ (p_x x_1 + p_y y_1) \cdot y_0 \end{bmatrix} = \begin{bmatrix} p_x (x_1 \cdot x_0) + p_y (y_1 \cdot x_0) \\ p_x (x_1 \cdot y_0) + p_y (y_1 \cdot y_0) \end{bmatrix}$$

Coordinate Transformations (rotation only)

Suppose a point P is rigidly attached to coordinate Frame 1, with coordinates given by ${}^1P = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$.

We can express the location of the point P in terms of its coordinates

$$P = p_x x_1 + p_y y_1$$



To obtain the coordinates of P w.r.t. Frame 0, we project P onto the x_0 and y_0 axes:

$$P^0 = \begin{bmatrix} P \cdot x_0 \\ P \cdot y_0 \end{bmatrix} = \begin{bmatrix} (p_x x_1 + p_y y_1) \cdot x_0 \\ (p_x x_1 + p_y y_1) \cdot y_0 \end{bmatrix} = \begin{bmatrix} p_x (x_1 \cdot x_0) + p_y (y_1 \cdot x_0) \\ p_x (x_1 \cdot y_0) + p_y (y_1 \cdot y_0) \end{bmatrix}$$

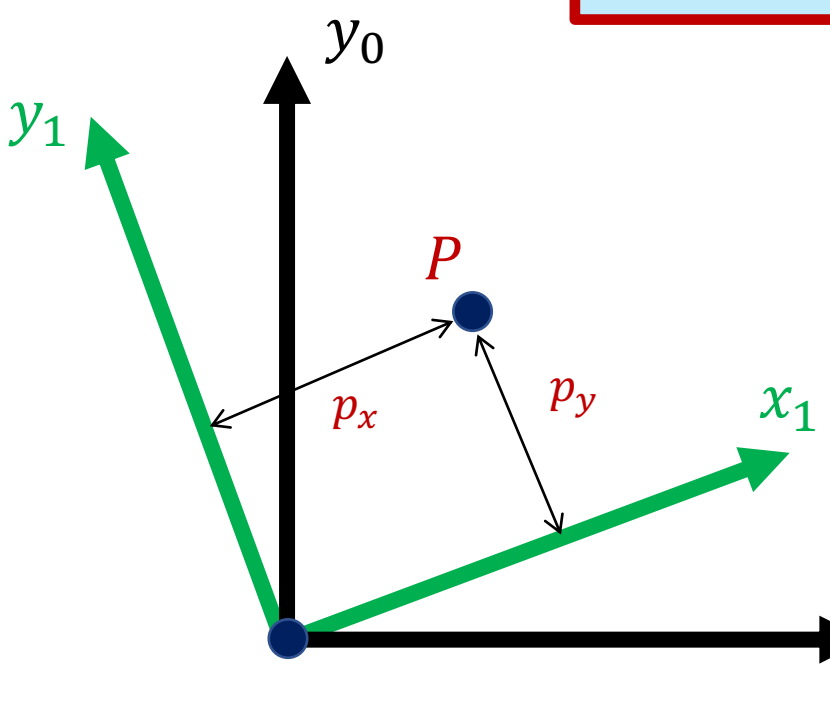
$$= \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

Coordinate Transformations (rotation only)

Suppose a point P is rigidly attached to coordinate Frame 1, with coordinates given by $P^1 = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$.

We can express the location of the point P in terms of its coordinates

$$P = p_x x_1 + p_y y_1$$



To obtain the coordinates of P w.r.t. Frame 0, we project P onto the x_0 and y_0 axes:

$$p^0 = \begin{bmatrix} P \cdot x_0 \\ P \cdot y_0 \end{bmatrix} = \begin{bmatrix} (p_x x_1 + p_y y_1) \cdot x_0 \\ (p_x x_1 + p_y y_1) \cdot y_0 \end{bmatrix} = \begin{bmatrix} p_x (x_1 \cdot x_0) + p_y (y_1 \cdot x_0) \\ p_x (x_1 \cdot y_0) + p_y (y_1 \cdot y_0) \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \boldsymbol{R_1^0 \, P^1}$$

$$\boldsymbol{P^0 = R_1^0 \, P^1}$$

SO(2) is a commutative group

- Group properties:

1. **Closure**: For all rotations $R, R' \in SO(2)$, their product is also in $SO(2)$, i.e., $RR' \in SO(2)$.
2. **Identity Element**: The 2×2 identity matrix I is included in the group, and for all $R \in SO(2)$ we have $RI = IR = R$.
3. **Inverse**: For every $R \in SO(2)$ there exists $R^{-1} \in SO(2)$ such that $R^{-1}R = RR^{-1} = I$.
4. **Associativity**: For all $R_1, R_2, R_3 \in SO(2)$, $(R_1R_2)R_3 = R_1(R_2R_3)$.

In addition (only in 2D!) rotations in $SO(2)$ *commute*:

1. **Commutativity**: For all $R_1, R_2 \in SO(2)$, $R_1R_2 = R_2R_1$.

SO(2) in GTSAM

- Rot2 is a class that internally actually stores sin/cos, but you can always get the rotation matrix from it:

```
theta = math.radians(30)
R = gtsam.Rot2(theta)
print(R.matrix())
```

```
[[ 0.8660254 -0.5       ]
 [ 0.5        0.8660254]]
```

Rot2 obeys the group properties

- Rotations in 2D form a *commutative* group, as demonstrated here:

```
print(isinstance(R * R, gtsam.Rot2)) # closure
I2 = gtsam.Rot2.identity()
print(R.equals(R * I2, 1e-9)) # identity
print((R * R.inverse()).equals(I2, 1e-9)) # inverse
R1, R2, R3 = gtsam.Rot2(1), gtsam.Rot2(2), gtsam.Rot2(3)
print(((R1 * R2)* R3).equals(R1 * (R2 * R3), 1e-9)) # associative
print((R1 * R2).equals(R2 * R1, 1e-9)) # commutative
```

```
True
True
True
True
True
```

Rotations act on points:

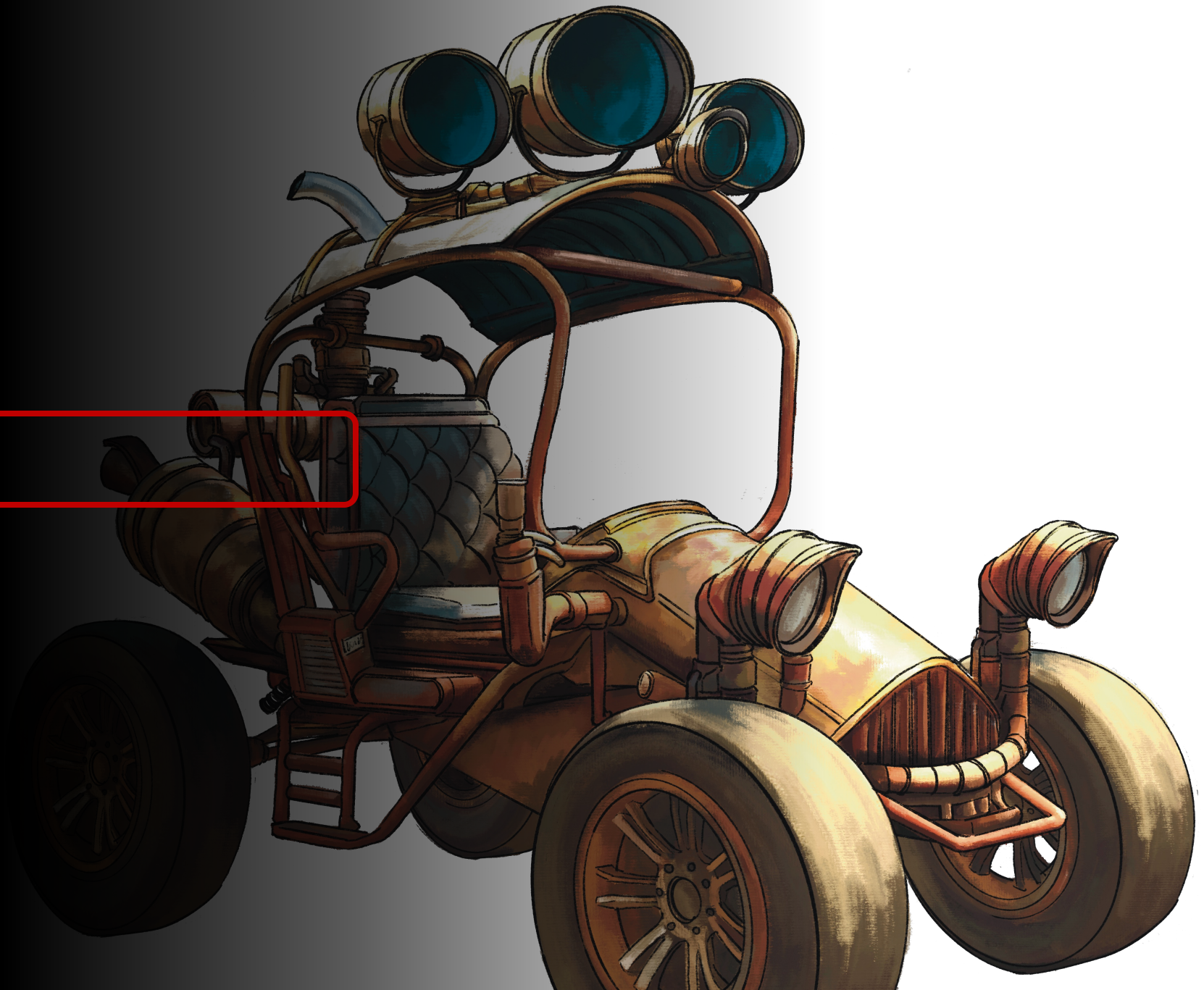
- Rotations can act on points, which we can do using matrix multiplication, or using the Rot2.rotate method:

```
R01 =  gtsam.Rot2(math.radians(20))
P1 = gtsam.Point2(4,3)
print(f"P0 = {R01.matrix() @ P1}")
print(f"P0 = {R01.rotate(P1)}")
```

```
P0 = [2.73271005 4.18715844]
P0 = [2.73271005 4.18715844]
```

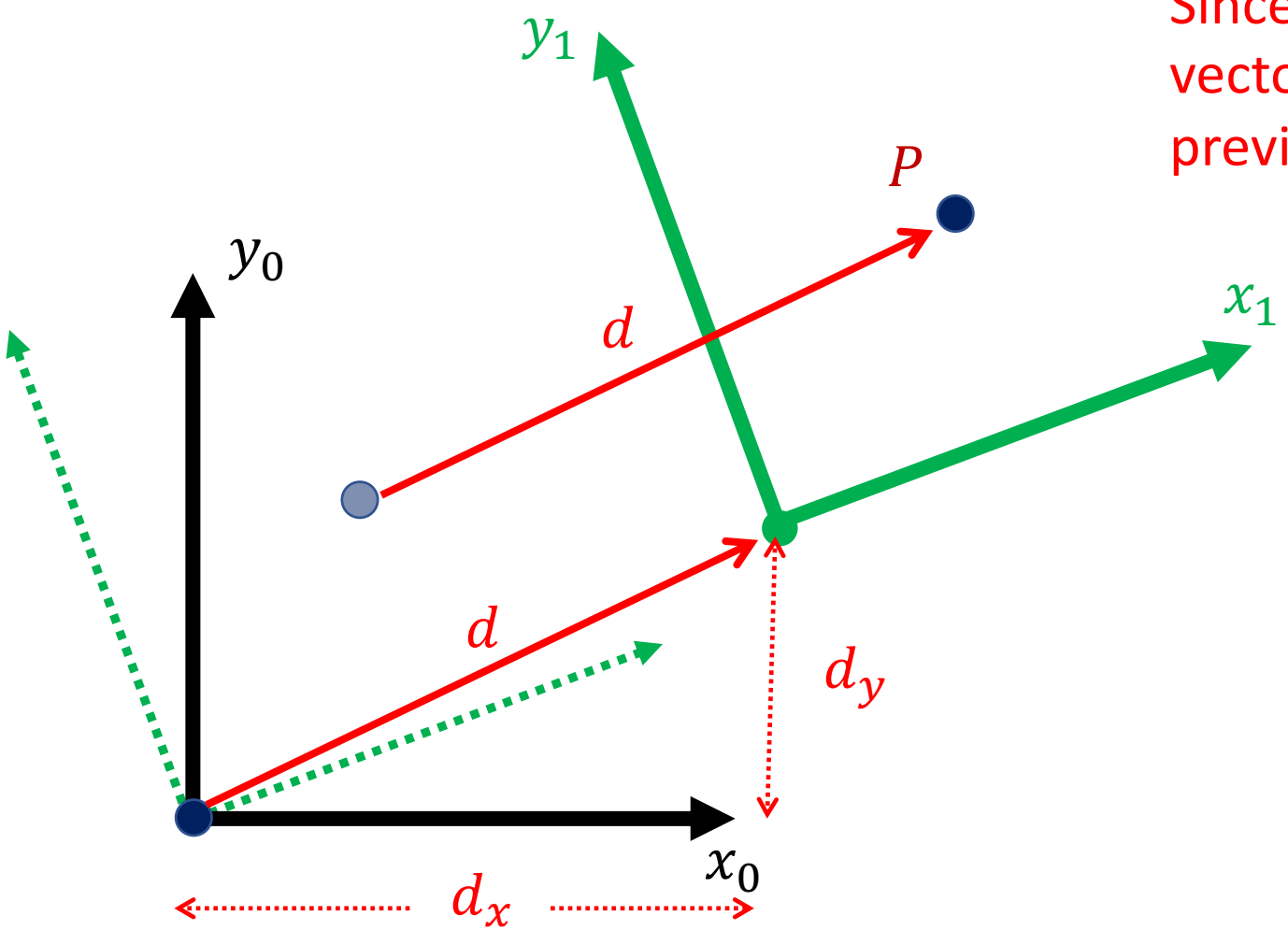
Outline

1. Reference Frames
2. Rotations and SO(2)
3. Poses and SE(2)
4. Car Kinematics
5. Ackermann Steering
6. Differential Kinematics



Specifying Pose in the Plane

Suppose we now translate Frame 1 (*no new rotation*).
What are the coordinates of P w.r.t. Frame 0?



Since we merely translated P by a fixed vector d, simply add the offset to our previous result!

$$\boldsymbol{P^0 = R_1^0 P^1 + d^0}$$

$$\boldsymbol{d^0 = \begin{bmatrix} d_x \\ d_y \end{bmatrix}}$$

Homogeneous Transformations

We can simplify the equation for coordinate transformations by augmenting the vectors and matrices with an extra row:

This is just our eqn from the previous page

$$\begin{bmatrix} \boldsymbol{P^0} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R_1^0 P^1 + d^0} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R_1^0} & \boldsymbol{d^0} \\ 0_2 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{P^1} \\ 1 \end{bmatrix}$$

in which $0_2 = [0 \quad 0]$

The set of matrices of the form $\begin{bmatrix} R & d \\ 0_n & 1 \end{bmatrix}$, where $R \in SO(n)$ and $d \in \mathbb{R}^n$ is called

the ***Special Euclidean Group of order n***, or $SE(n)$.

Inverse of a Homogeneous Transformation

What is the relationship between T_1^0 and T_0^1?

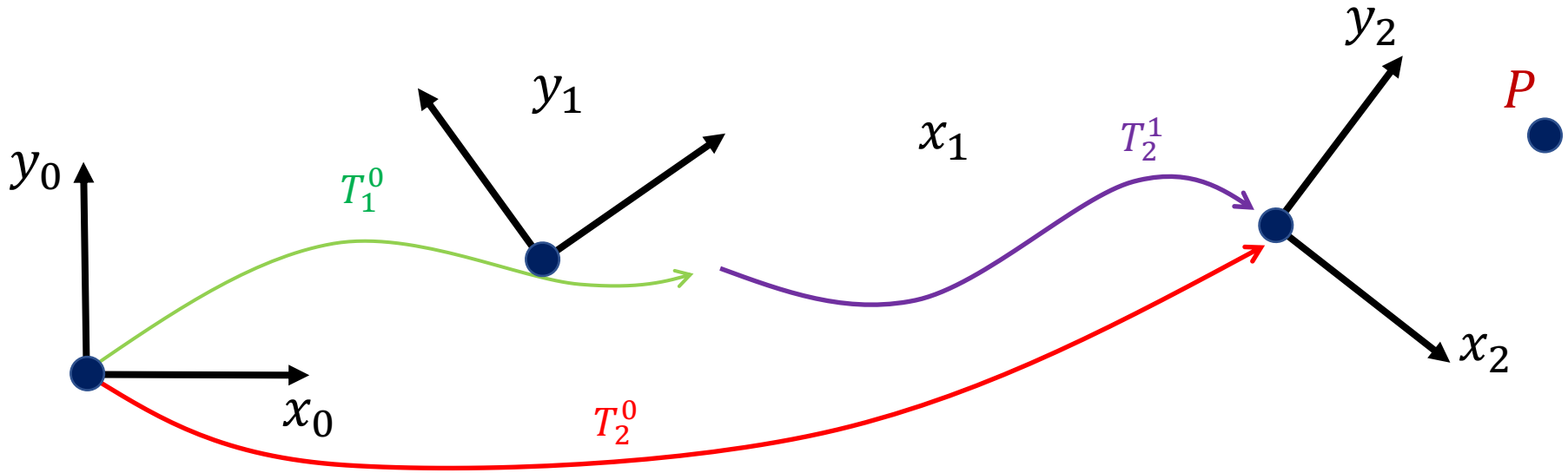
$$T_1^0 T_0^1 = \begin{bmatrix} 0.5\sqrt{2} & -0.5\sqrt{2} & 4 \\ 0.5\sqrt{2} & 0.5\sqrt{2} & 8 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5\sqrt{2} & 0.5\sqrt{2} & -6\sqrt{2} \\ -0.5\sqrt{2} & 0.5\sqrt{2} & -2\sqrt{2} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In general, $T_k^j = \left(T_j^k\right)^{-1}$ and $\begin{bmatrix} \boldsymbol{R} & \boldsymbol{d} \\ \boldsymbol{0}_n & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{R^T} & \boldsymbol{-R^T d} \\ \boldsymbol{0}_n & 1 \end{bmatrix}$

This is easy to verify:

$$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{d} \\ \boldsymbol{0}_n & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R^T} & \boldsymbol{-R^T d} \\ \boldsymbol{0}_n & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{RR^T} & \boldsymbol{-RR^T d + d} \\ \boldsymbol{0}_n & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{I_{n\times n}} & \boldsymbol{0}_n \\ \boldsymbol{0}_n & 1 \end{bmatrix} = I_{(n+1)\times(n+1)}$$

Composition of Transformations



From our previous results, we know:

$$P^0 = T_1^0 P^1$$
$$P^1 = T_2^1 P^2$$

$$\Rightarrow \quad P^0 = T_1^0 T_2^1 P^2$$

But we also know: $P^0 = T_2^0 P^2$

$$\Rightarrow \quad T_2^0 = T_1^0 T_2^1$$

This is the composition law for homogeneous transformations.

SE(2) is a *non-commutative* group

- Group properties:

1. **Closure**: For all transforms $T, T' \in SE(2)$, their product is also in $SE(2)$, i.e., $TT' \in SE(2)$.
2. **Identity Element**: The 3×3 identity matrix I is included in the group, and for all $T \in SE(2)$ we have $TI = IT = T$.
3. **Inverse**: For every $T \in SE(2)$ there exists $T^{-1} \in SE(2)$ such that $T^{-1}T = TT^{-1} = I$.
4. **Associativity**: For all $T_1, T_2, T_3 \in SE(2)$, $(T_1T_2)T_3 = T_1(T_2T_3)$.

However, in contrast to 2D rotations, 2D rigid transforms do *not* commute. Also, The inverse T^{-1} is not just the transpose; instead, we have:

$$T^{-1} = \begin{bmatrix} R & d \\ 0_2 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0_2 & 1 \end{bmatrix}$$

GTSAM Example

- All of this is built into GTSAM, where both 2D poses and 2D rigid transforms are represented by the type Pose2:

```
theta = math.radians(30)
T = gtsam.Pose2(3, -2, theta)
print(f"2D Pose (x, y, theta) = {T}corresponding to transformation matrix:\n{T.matrix()}")
```

```
2D Pose (x, y, theta) = (3, -2, 0.523599)
corresponding to transformation matrix:
[[ 0.8660254 -0.5        3.       ]
 [ 0.5        0.8660254 -2.       ]
 [ 0.         0.        1.       ]]
```

Pose2 obeys the group properties

- All group properties *except* not commutative:

```
print(isinstance(T * T, gtsam.Pose2)) # closure
I3 = gtsam.Pose2.identity()
print(T.equals(T * I3, 1e-9)) # identity
print((T * T.inverse()).equals(I3, 1e-9)) # inverse
T1, T2, T3 = gtsam.Pose2(1,2,3), gtsam.Pose2(4,5,6), gtsam.Pose2(7,8,9)
print(((T1 * T2)* T3).equals(T1 * (T2 * T3), 1e-9)) # associative
print((T1 * T2).equals(T2 * T1, 1e-9)) # NOT commutative
```

```
True
True
True
True
False
```

Pose2 acts on Point2

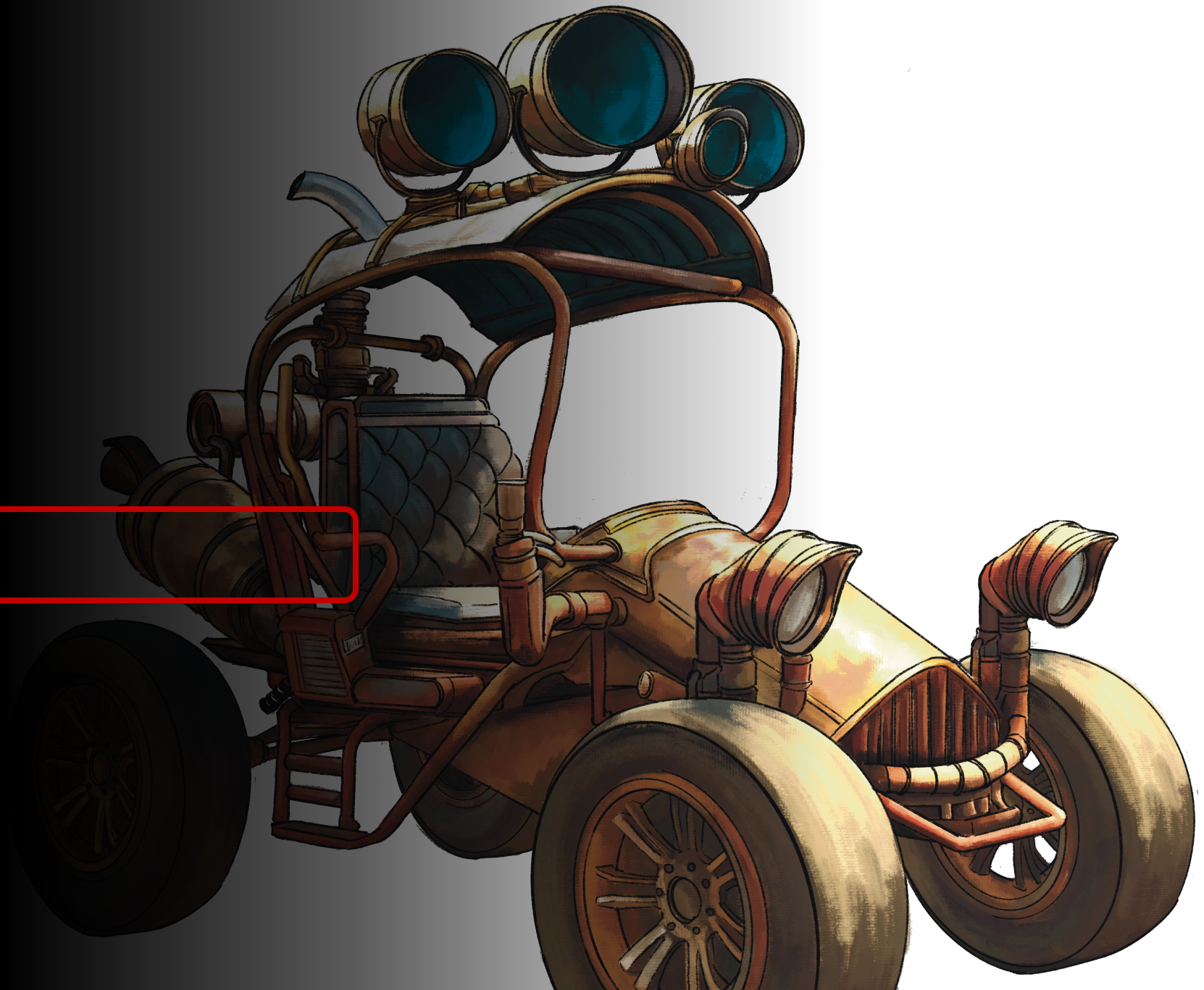
- 2D transforms can act on points, which we can do using matrix multiplication, or using the Pose2.transformFrom method:

```
T01 =  gtsam.Pose2(1, 2, math.radians(20))
P1 = gtsam.Point2(4,3)
print(f"P0 = {T01.matrix() @ [4, 3, 1]}")  # need to make P0 homogeneous
print(f"P0 = {T01.transformFrom(P1)}")
```

```
P0 = [3.73271005 6.18715844 1.        ]
P0 = [3.73271005 6.18715844]
```

Outline

1. Reference Frames
2. Rotations and SO(2)
3. Poses and SE(2)
4. Car Kinematics
5. Ackermann Steering
6. Differential Kinematics



The Kinematics of Cars

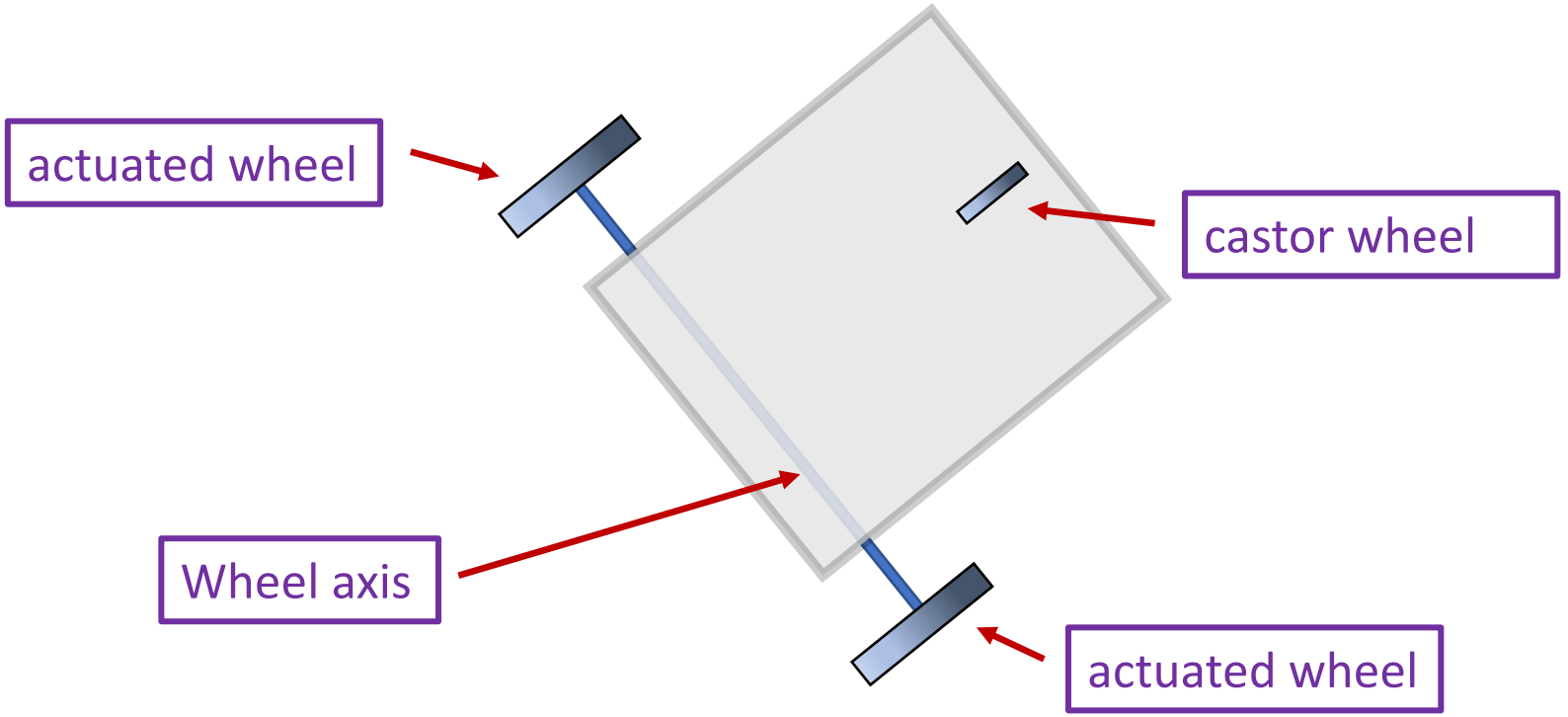
Cars are slightly more complicated than differential drive robots:

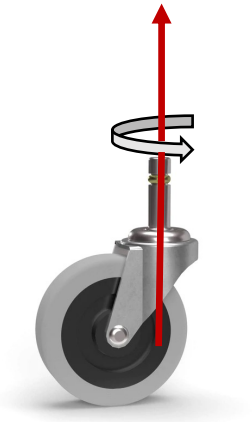
- Cars cannot “turn in place” – they have a finite turning radius.
- Most cars have front-wheel steering, and rear wheels with a fixed orientation – this will require a bit of (not too complicated) geometry.
- Inputs to a car are:
  - ***Steering direction***, which is determined by the angle of a steering wheel for the driver,
  - ***Forward speed***, which is determined by accelerator.
- For real cars, the input is more accurately modeled in terms of acceleration, rather than speed, but for most purposes, treating the car as a velocity-controlled vehicle (and thereby ignoring physics) is a useful abstraction.

Differential Drive Robots

Differential drive robots (aka DDRs):

- Two actuated wheels that share an axis
- A castor wheel that rotates freely, mainly to stabilize the robot (three points define a plane – castor wheel keeps the robot from tipping over).

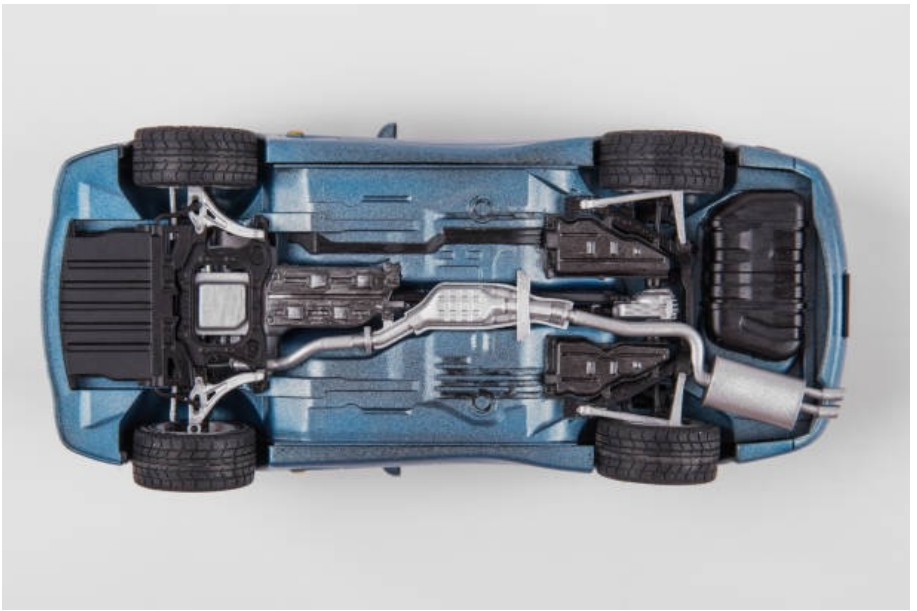




A castor wheel is able to spin freely about the vertical axis.

Cars





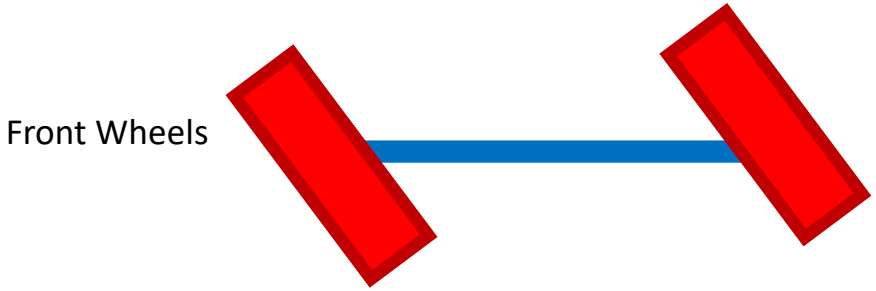
Cars have four wheels, and all four matter.

- All four wheels (should) roll without slipping.
- Only the front wheels can turn.
- No castor wheels: rear wheels have a fixed orientation.
- Finite turning radius: cannot spin in place.
- We can develop a simpler model for car kinematics that captures the essential geometry.

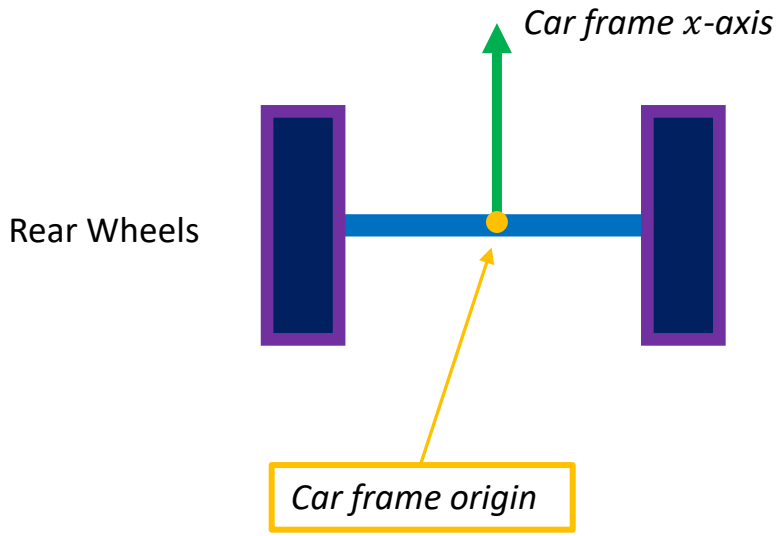


Modeling Front-Wheel Steering

- Typically, cars have front-wheel steering, meaning that turning the steering wheel changes the direction of the two front wheels.



- We assign a coordinate frame to the car:
  - Origin is at the midpoint along the rear axle.
  - Direction of x-axis is perpendicular to the rear axle.

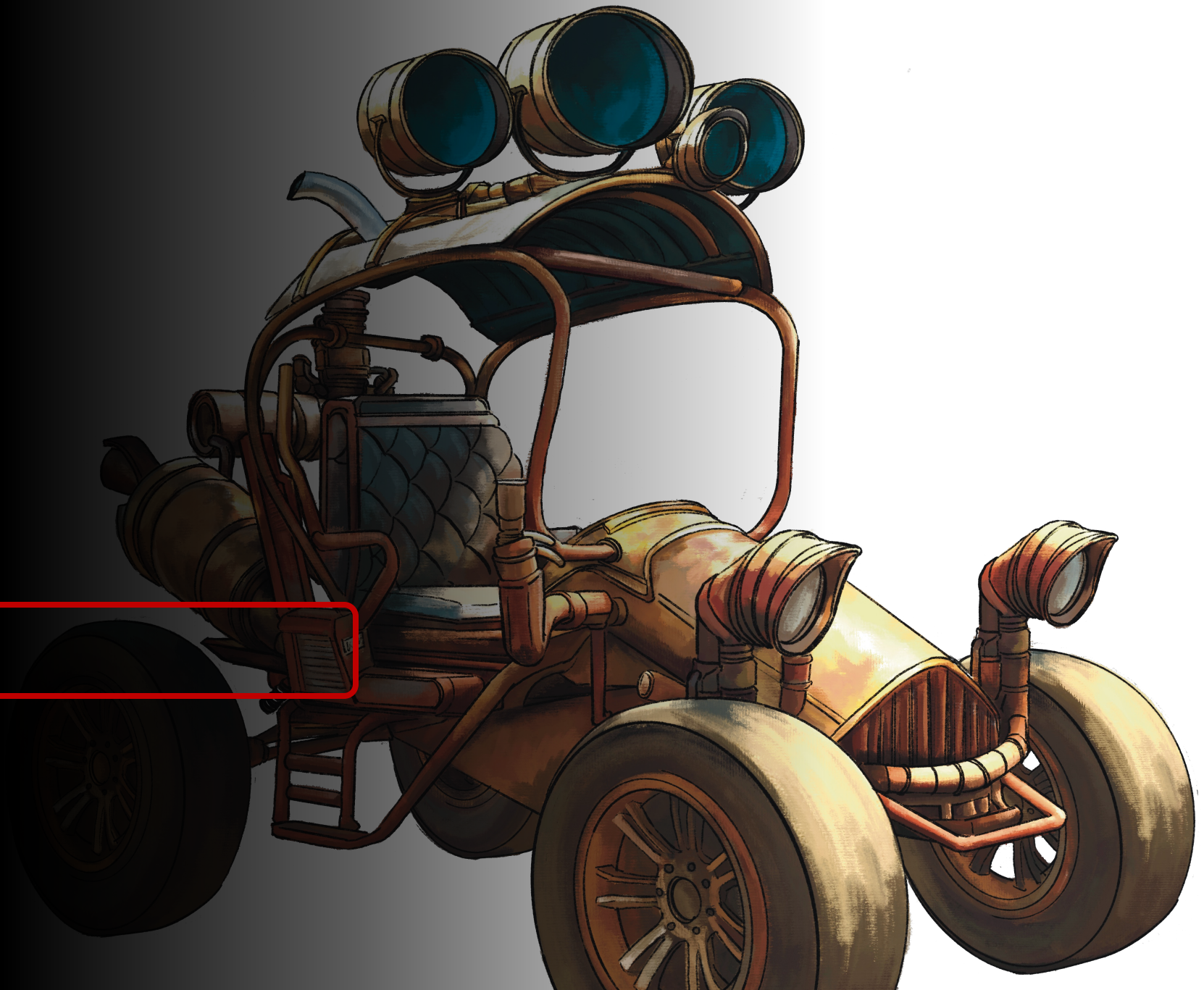


- The linear velocity of the car is defined to be the velocity of the origin of the car frame. This velocity is *always* in the direction of the car frame x-axis.

- This linear speed is denoted by v_{car}.

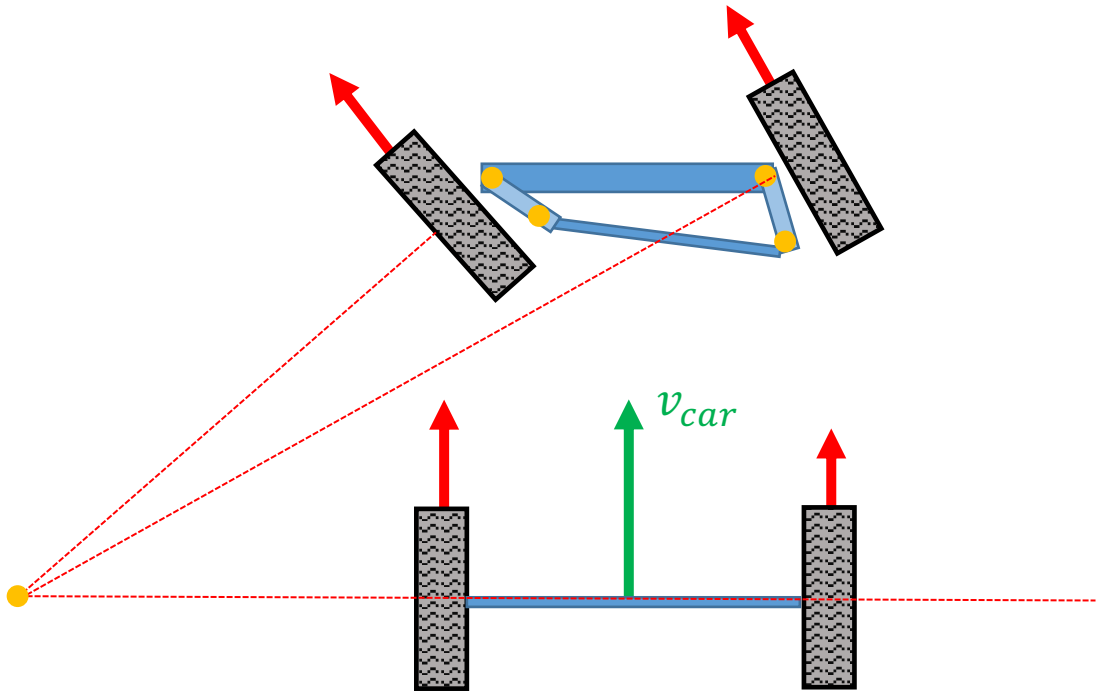
Outline

1. Reference Frames
2. Rotations and SO(2)
3. Poses and SE(2)
4. Car Kinematics
5. Ackermann Steering
6. Differential Kinematics



Ackermann Steering

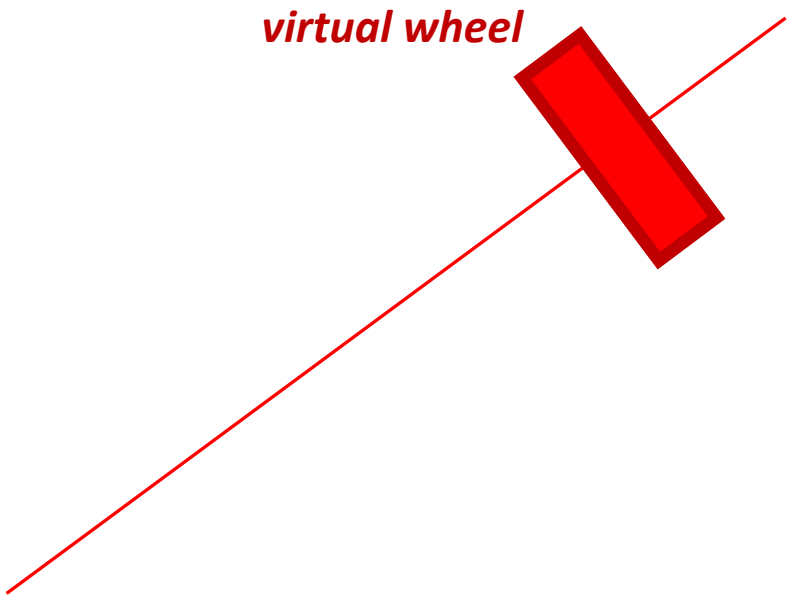
- When a car makes a turning maneuver, in order to avoid slipping (or skidding), the inner and outer wheels must (instantaneously) trace out circles of different radii, but with the same center.
- The instantaneous velocity of each wheel is tangent to the circle that it traces.

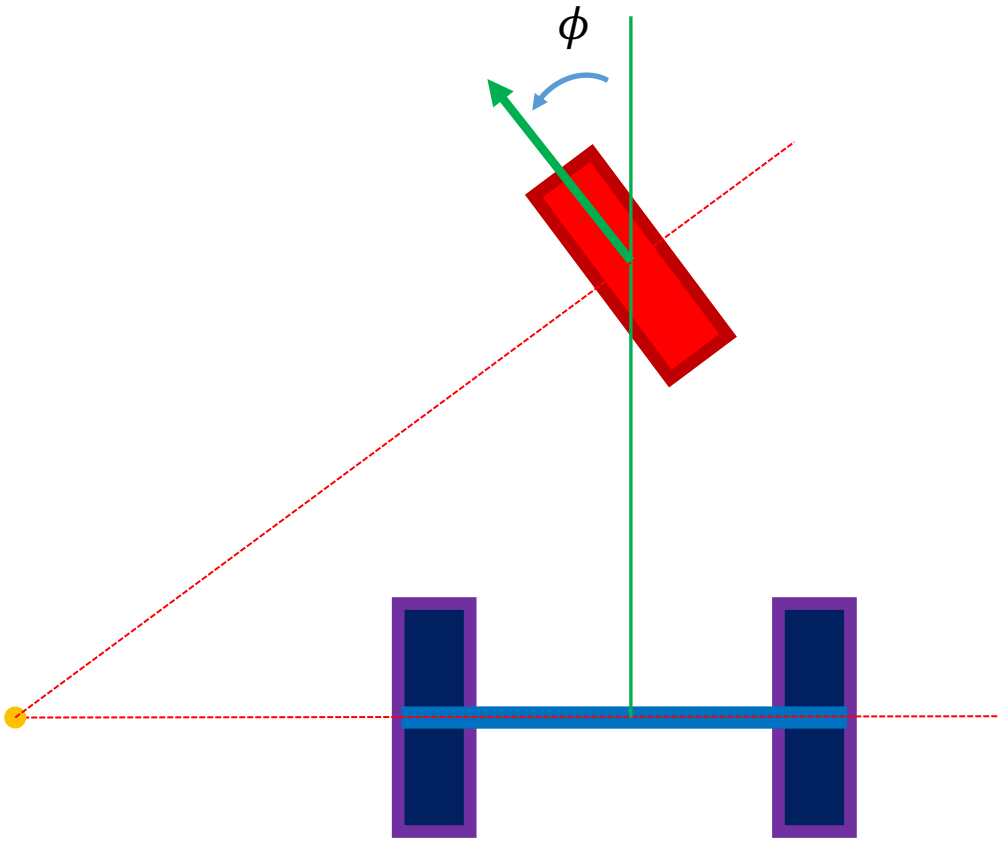


The center point of the rear axle moves with the same velocity as the two rear wheels.

Ackermann Steering (modeling)

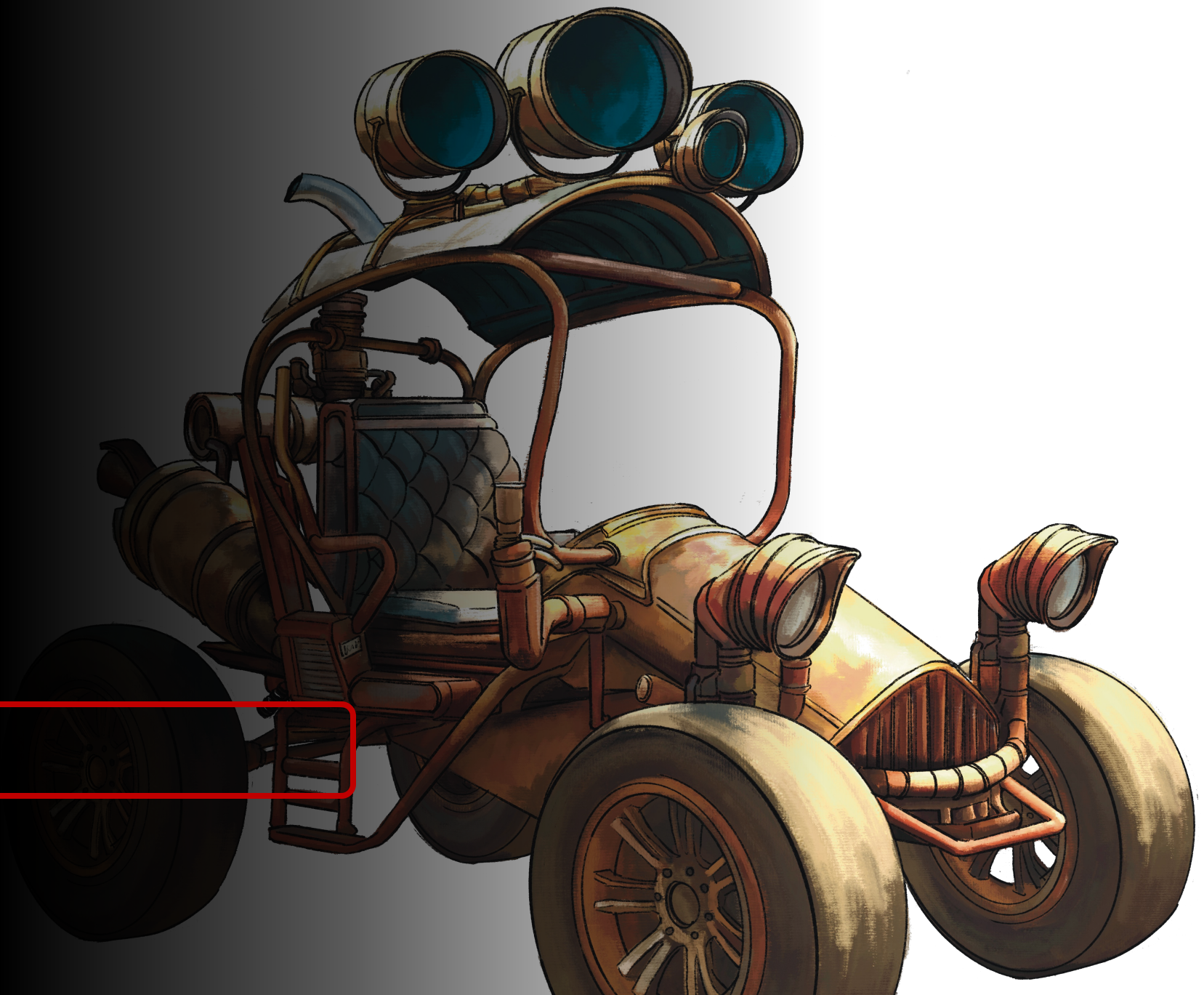
- To simplify the geometry, we typically consider a virtual wheel, placed at the midpoint between the two front wheels.
- We'll denote by ϕ the angle of the virtual wheel w.r.t. the car's x-axis.





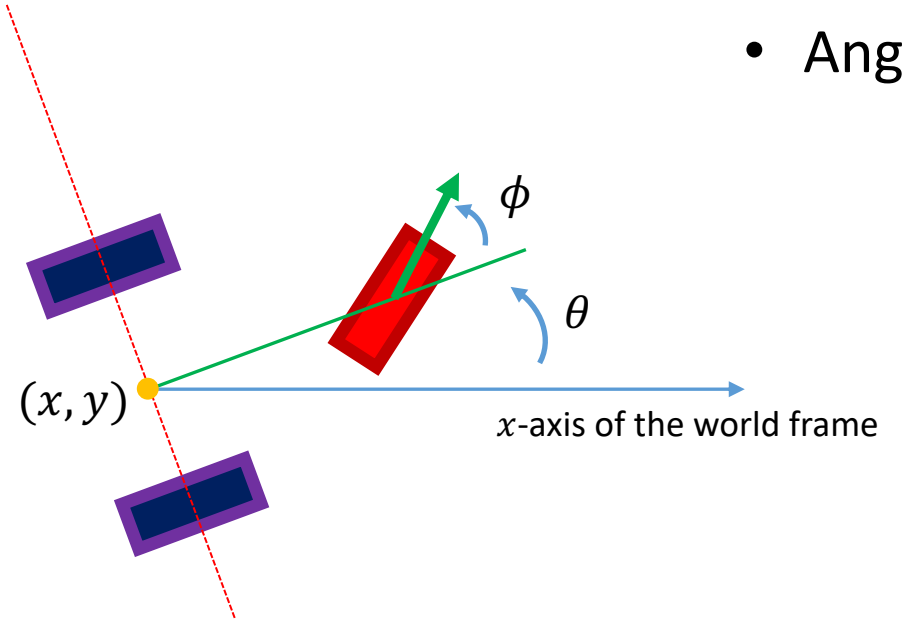
Outline

1. Reference Frames
2. Rotations and SO(2)
3. Poses and SE(2)
4. Car Kinematics
5. Ackermann Steering
6. Differential Kinematics



Differential Kinematics

- The configuration of the car can be represented by $q = (x, y, \theta, \phi)$, where
  - x, y defines the origin of the car frame (midpoint of rear axle).
  - θ is the angle from the x-axis of the world frame to the x-axis of the car frame.
  - ϕ is the steering angle.



We'll model the car as having two inputs:

- Linear speed in the direction of the car x-axis: $u_1 = v_{car}$
- Angular turning rate of the virtual wheel: $\mathrm{u}_2 = \dot{\phi}$

The car's differential kinematics are specified by the equation:

$$\dot{q} = f(q, u_1, u_2)$$

Differential Kinematics

Most of this is easy.

- For the linear velocity of the car, expressed w.r.t. the world coordinate frame:

$$\dot{x} = v_{car} \cos\theta = u_1 \cos\theta$$
$$\dot{y} = v_{car} \sin\theta = u_1 \sin\theta$$

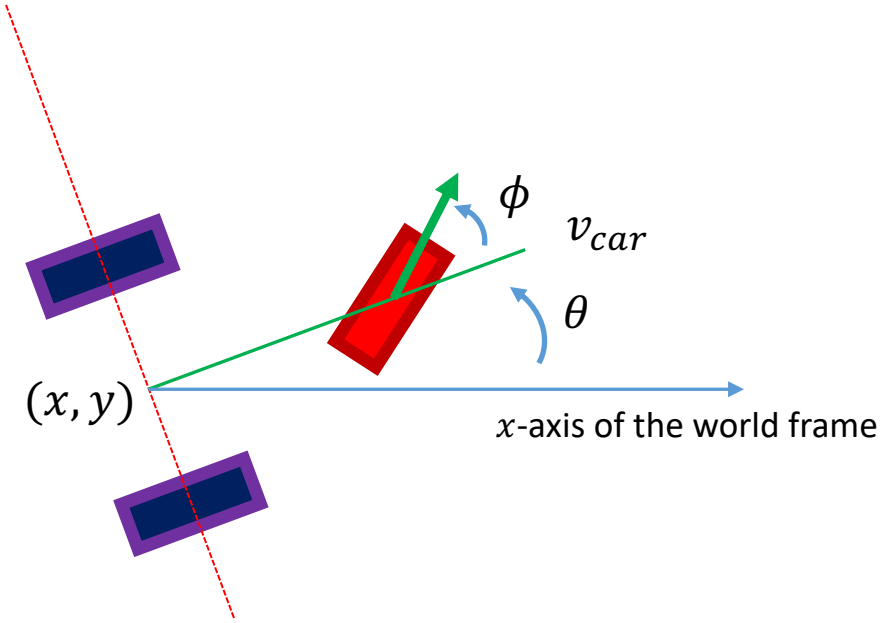
- For the steering angle:

$$u_2 = \dot{\phi}$$

Note that we have direct control over the rate of change of the steering angle, but only indirect control of the car's forward velocity (since it depends on θ).

NOTE:

- ***With this model, we cannot instantaneously choose the steering angle ϕ.***
- ***We can choose $\dot{\phi}$, and by integrating over time, we can achieve a desired steering angle ϕ.***



Differential Kinematics

For $\dot{\theta}$, first solve for the velocity $v_{tangent}$ perpendicular to the car direction of motion:

$$v_{wheel}\cos\phi = v_{car} \rightarrow v_{wheel} = \frac{v_{car}}{\cos\phi}$$

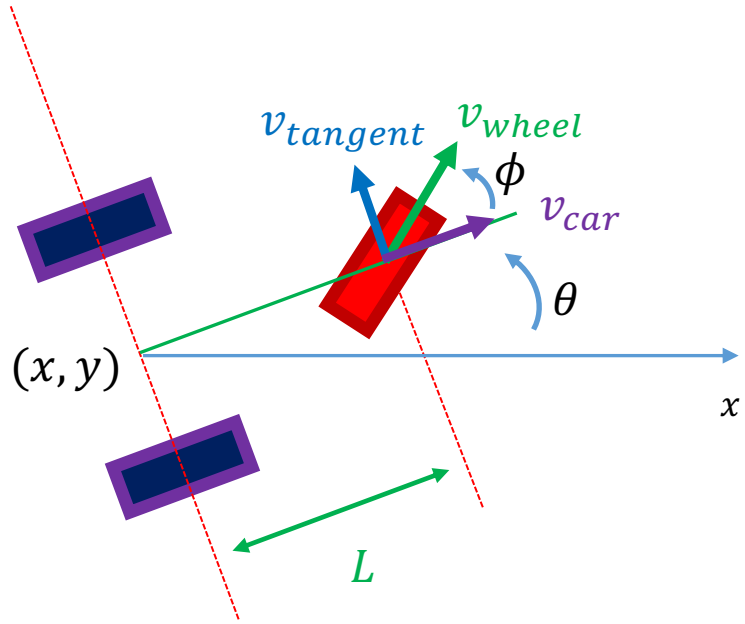
$$v_{wheel}\sin\phi = v_{tangent}$$

Combining these to eliminate v_{wheel} we obtain:

$$\frac{v_{car}}{\cos\phi}\sin\phi = v_{car}\tan\phi = v_{tangent}$$

Now use $L\dot{\theta} = v_{tangent}$ and we arrive to

$$\dot{\theta} = \mathrm{v_{car}}\frac{1}{L}\tan\phi = u_1\frac{1}{L}\tan\phi$$



Denote by L the baseline distance, from rear axle to center of steering wheel.

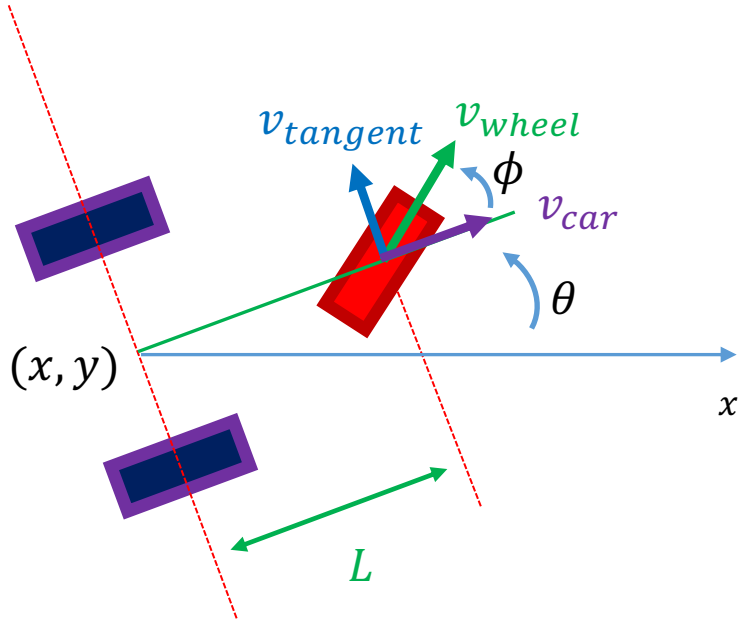
NOTE:

- ***We cannot instantaneously choose θ.***

Differential Kinematics

We can write the differential kinematics as:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{1}{L}\tan\phi \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2$$



In which

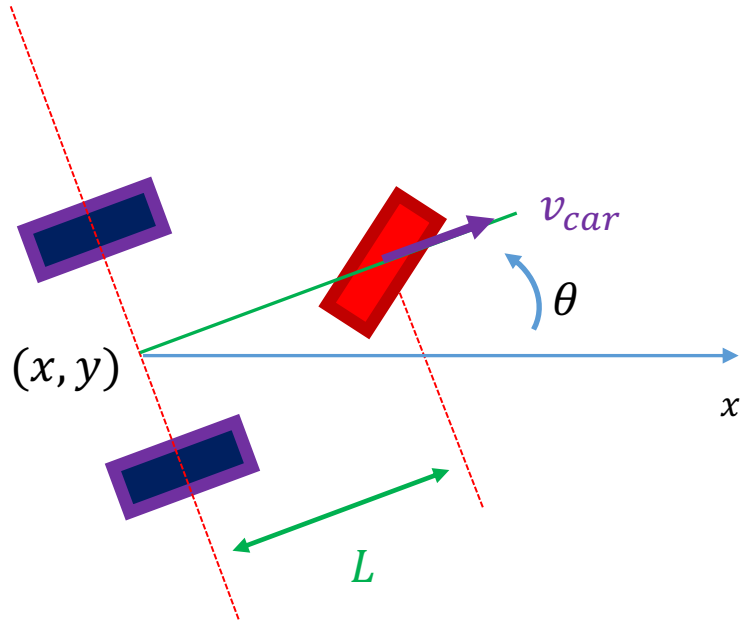
- $q = (x, y, \theta, \phi)$
- $u_1 = v_{car}$
- $u_2 = \dot{\phi}$

Differential Kinematics: simplified form

- It is convenient to write the differential kinematics considering the car angular velocity as an input.
- Instead of controlling θ indirectly via

$$\dot{\theta} = u_1 \frac{1}{L} \tan\phi\,, \qquad \dot{\phi} = u_2$$

we assume that we can directly control ϕ (instead of $\dot{\phi}$) and define $u_2 = \dot{\theta}$ (it's easy to solve for the ϕ that achieves this).



- We now write the differential kinematics as:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_2$$

In which

- $q = (x, y, \theta)$
- $u_1 = v_{car}$
- $u_2 = \dot{\theta}$, the angular velocity of car frame

This is reasonable if we can turn the steering wheel quickly, relative to the forward speed of the car.

Summary

1. Reference Frames
2. Rotations and SO(2)
3. Poses and SE(2)
4. Car Kinematics
5. Ackermann Steering
6. Differential Kinematics

