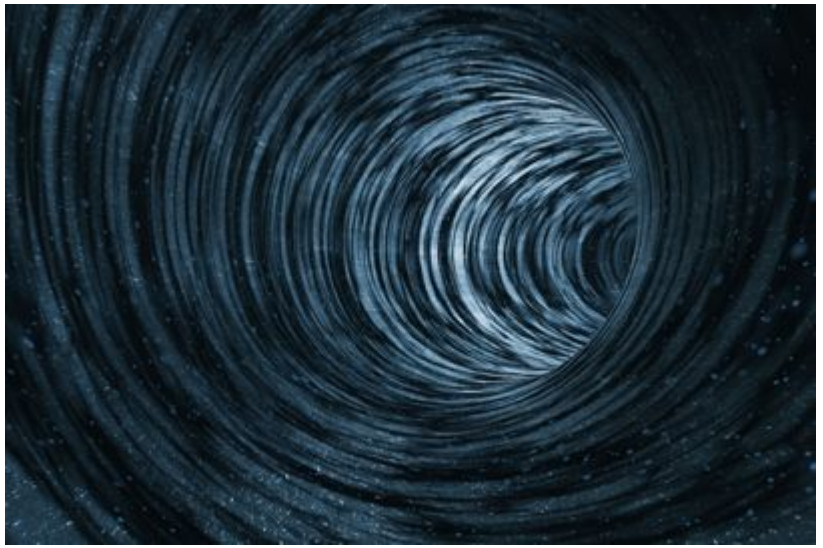
Representing and Modeling Space



* The topology of quantum worm-holes will be left for self-study.

McGill COMP 765

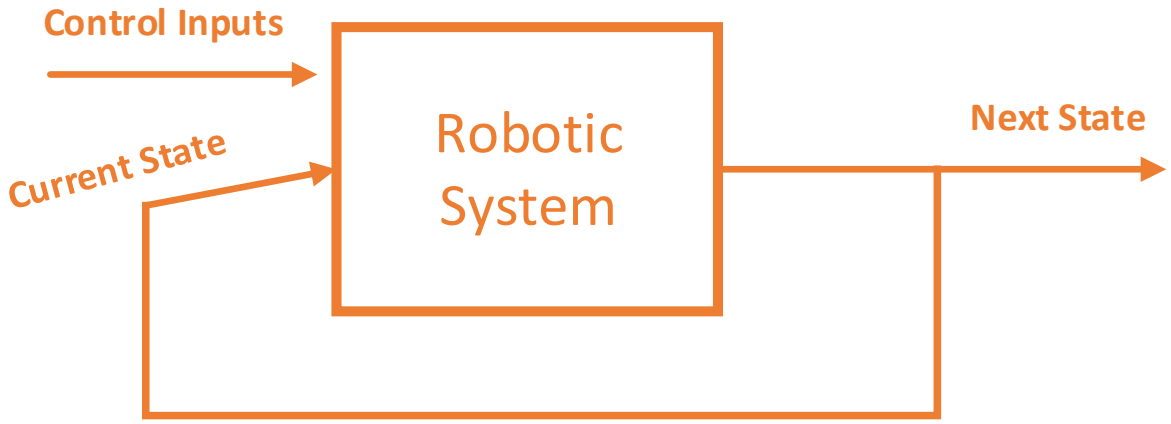
Jan 9th, 2019

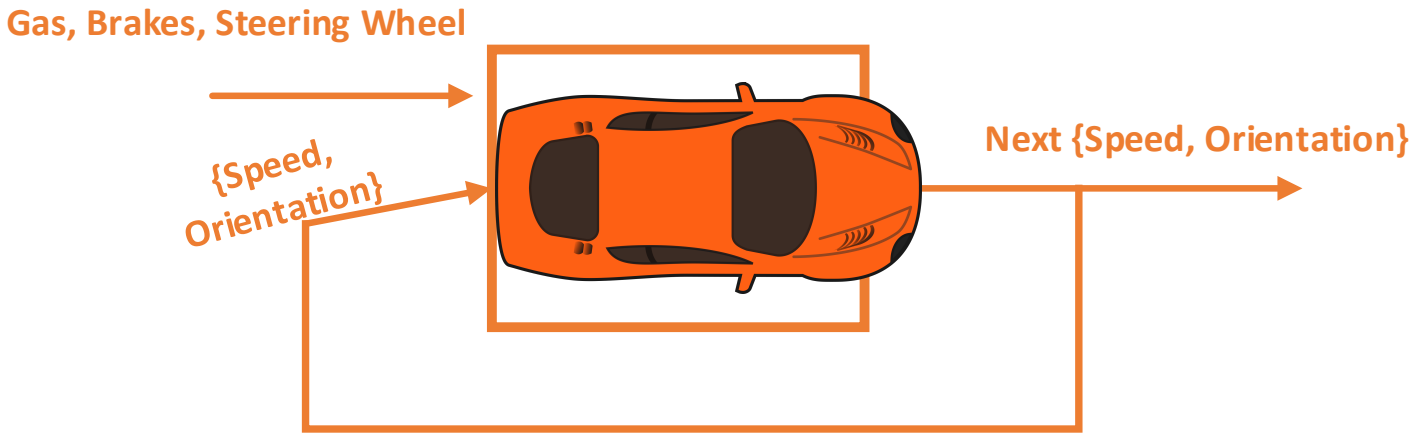
Goals today

- Brief beginning on models of a robot's movements
- Some first models to represent the space around the robot (environment)
- Introduction of key robotics problems: planning and mapping

Physical models of how systems move

Kinematics & Dynamics:
Idealized models of
robotic motion



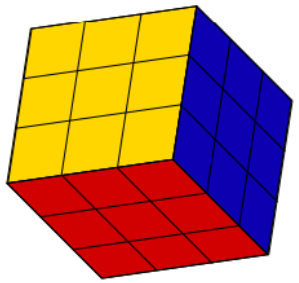


States and Actions

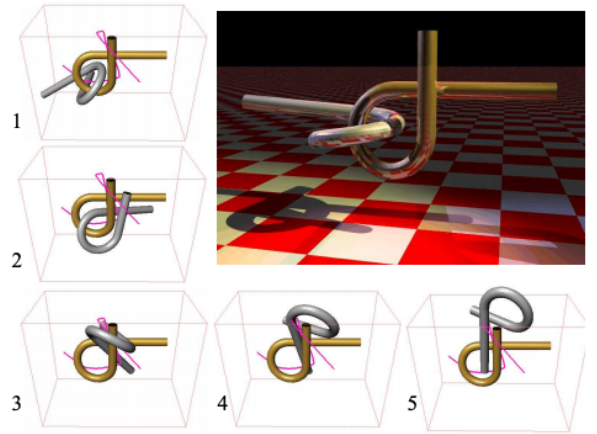
- The ***state space*** of a robot is the set of all physical situations that could arise. It captures the meaningful properties of the robot.
  - A physical robot doesn't come with its state space on the label. Determining the correct representation is task #1 of a roboticist.

- The ***action space*** of a robot are the commands, forces, or other effects it can take to change its own state, or that of the environment.
  - The action space can be state dependent. For example, I can execute "open" when I'm in front of a door, but not in the center of the room.

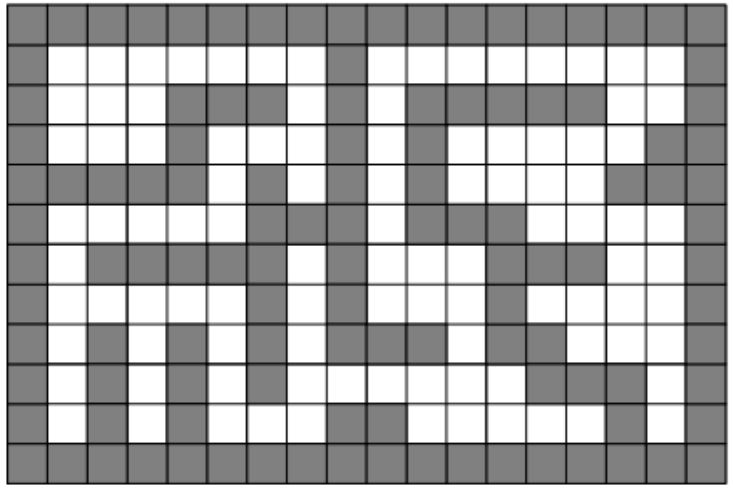
Sample State and Action Spaces



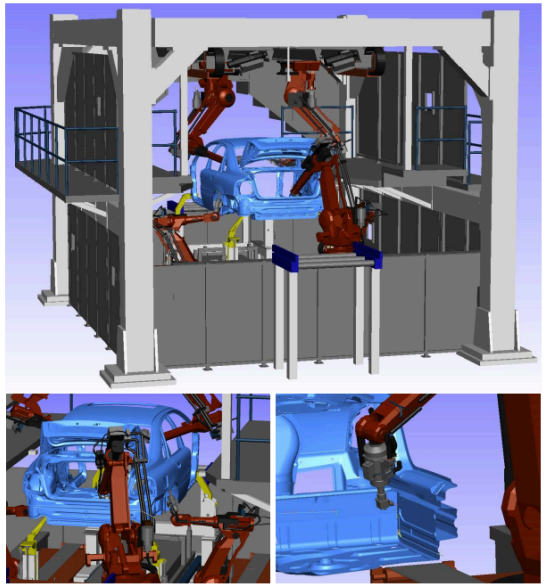
Ex1



Ex3



Ex2



Ex4

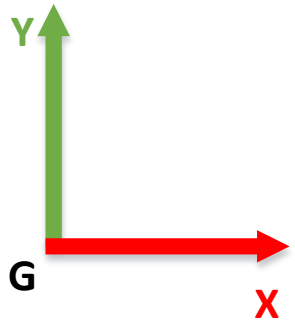
Kinematics and Dynamics

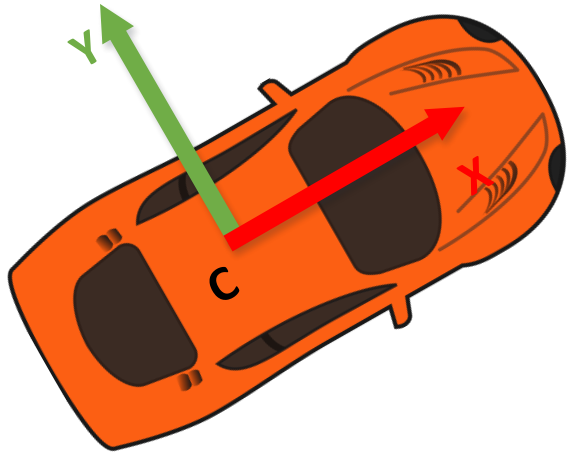
- Kinematics considers constraints on state space and how we can transition between states:
  - Can I reach the top shelf?
  - Does the car fit through that hole?
  - Is there any way to reach my office?
- Dynamics considers idealized physics of motion:
  - How will a quadrotor will move when the rotors spin at 50%?
  - Can I lift that couch?
  - Will a free swinging pendulum eventually stop, and where?

Why idealized?

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

Example: Kinematics of a simple car - state





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

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

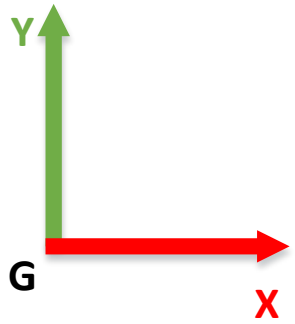
Note on Inertial frames of reference

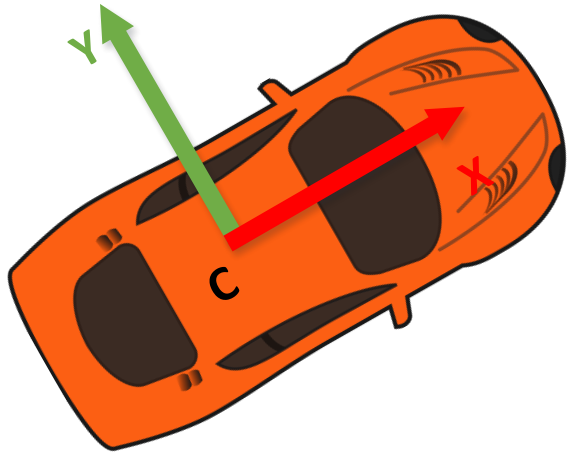
- G, the global frame of reference is fixed, i.e. with zero velocity in our previous example.

- If a robot's pose is known in the global frame, life is good:
  - It can correctly report its position
  - We can compute the direction and distance to a goal point
  - We can avoid collisions with obstacles also known in global frame

- We may also know the pose in another frame: e.g. of a map, sensor, or relative to it's starting point. These are often sub-steps for us.

Kinematics of a simple car - controls

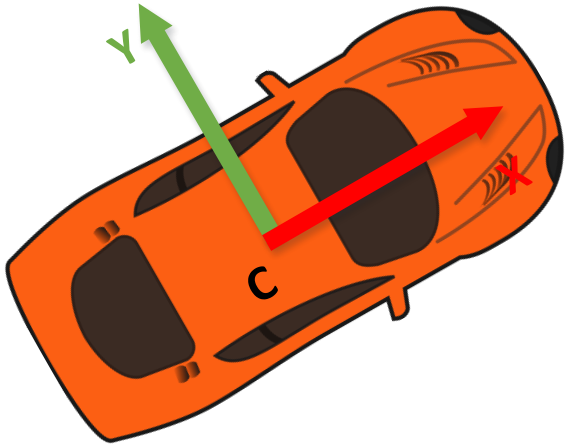


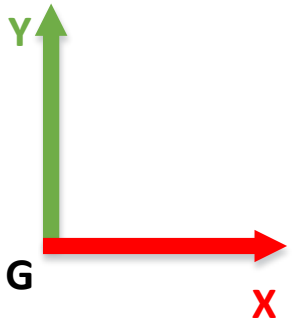


$$\mathbf{u} = [{}^{C}v_x, {}^{C}\omega_z]$$

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

Kinematics of a simple car - velocities

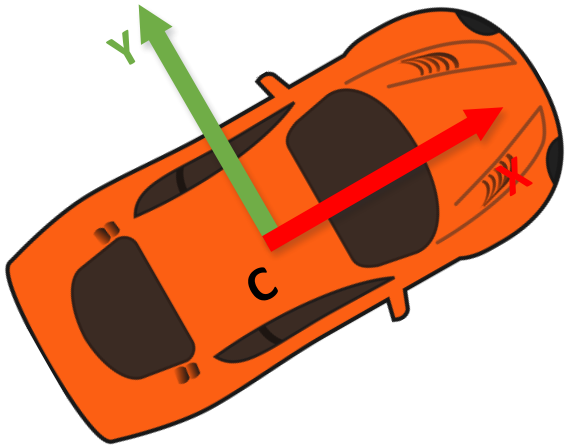


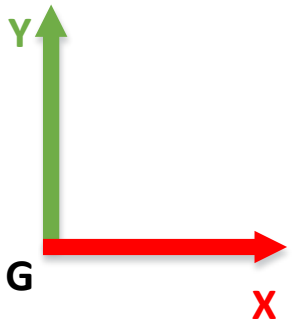


$$\begin{aligned} \dot{p}_x &= v_x \cos(\theta) \\ \dot{p}_y &= v_x \sin(\theta) \\ \dot{\theta} &= \omega_z \end{aligned}$$

Note: reference frames have been removed for readability.

Dynamics of a simple car – engine force and steering torque





$$v_x(t+\Delta) = v_x(t) + \frac{gas_x}{mass}\Delta$$

$$\dot{\omega}_z(t) = \dot{\omega}_z(0) + \frac{steer_x}{\frac{1}{2}MR^2}\Delta$$

Note: an example of an approximate model, this treats the car as a circular disk!

Kinematics as Constraints

- Our idea of using vector spaces or sets to represent everything that can happen is too broad to capture robot details:
  - Joints
  - Contact
  - One—directional motors
- A common form of kinematics is to state a set of constraints also. These can be on the allowable states or how to move between states.
  - Example: the state x is all 2D vectors on a unit circle
  - How would we also express only clock-wise motion?

$$x \in \mathbb{R}^2$$
$$s.t.\ x1^2 + x2^2 = r$$

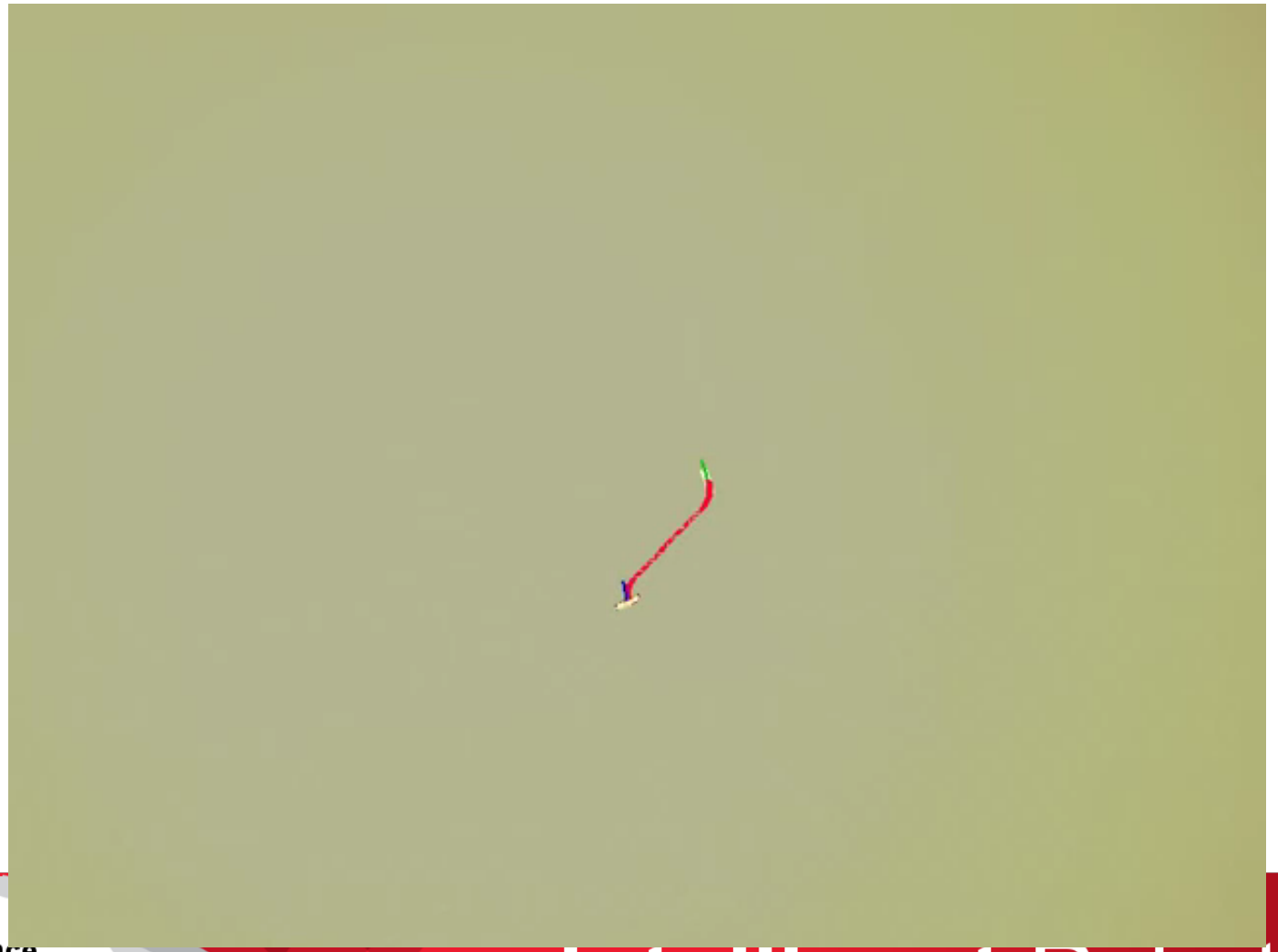
Special case of simple car: Dubins car

- Can only go forward at a constant speed

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

- You control the angular velocity, but rate is limited

$$0 \leq |v_x| < const$$

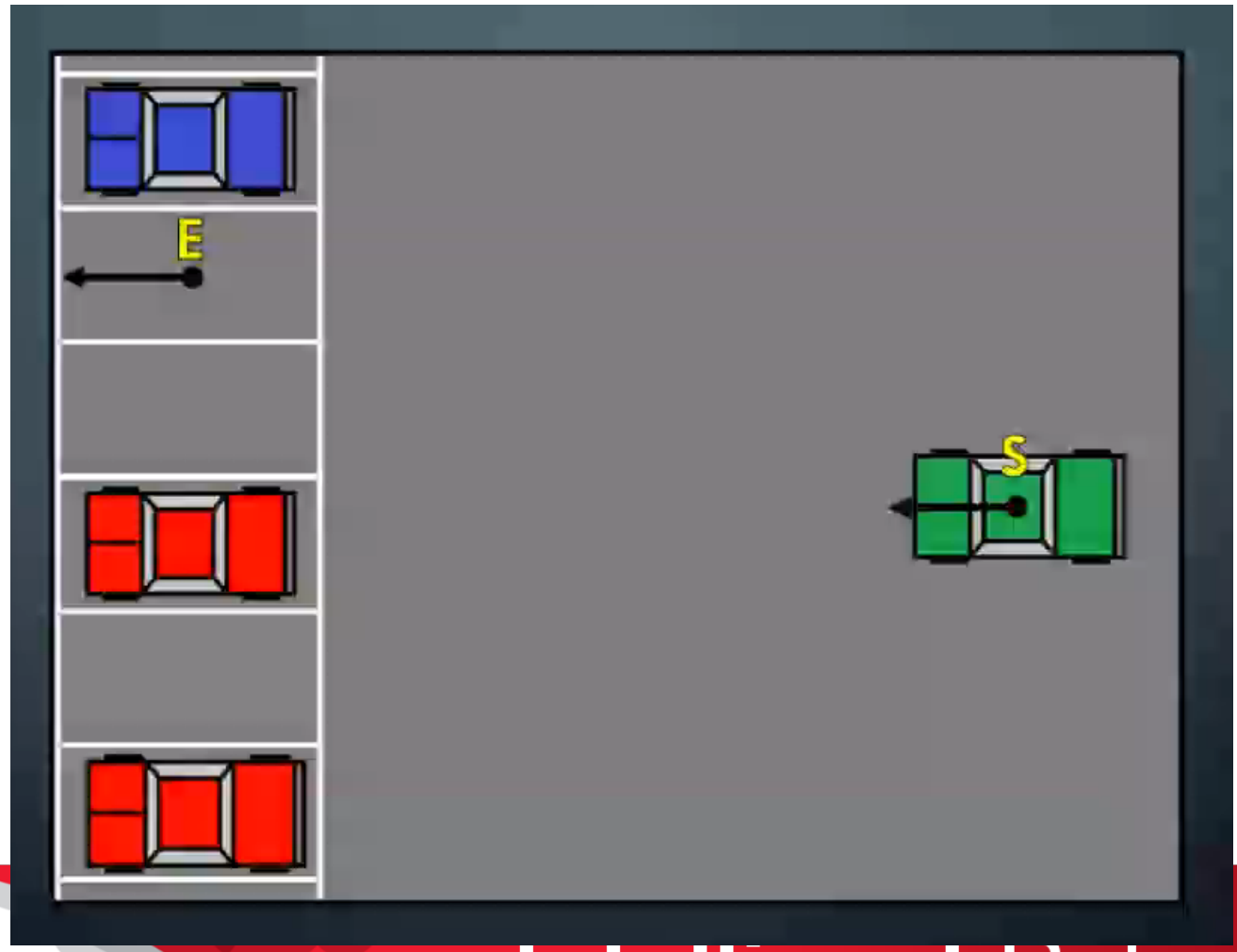


Special case of simple car: Dubins car

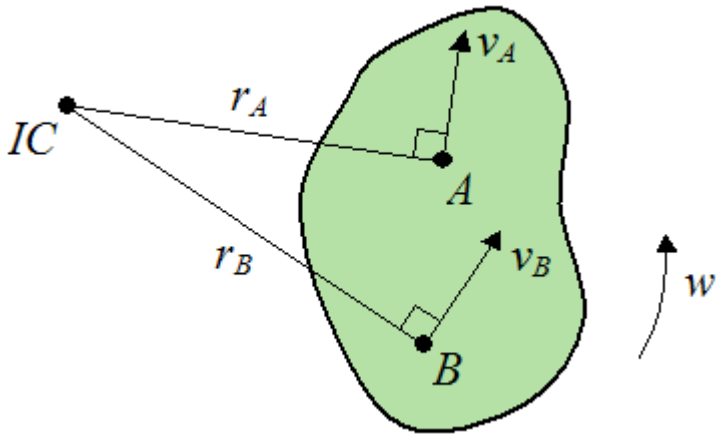
- Can only go forward
- Constant speed

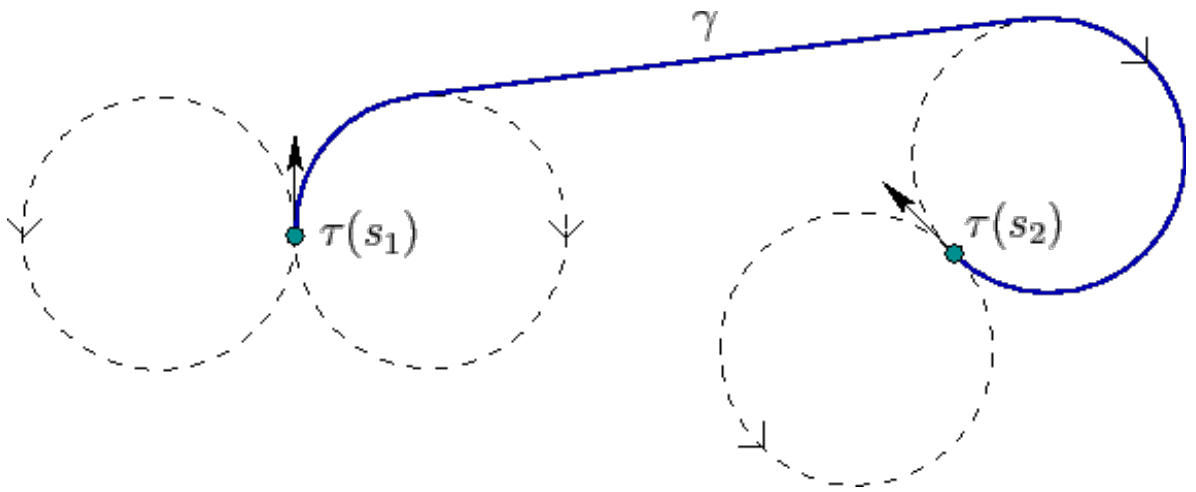
$$^{C}v_x = \text{const} > 0$$

- You only control the angular velocity



Instantaneous Center of Rotation

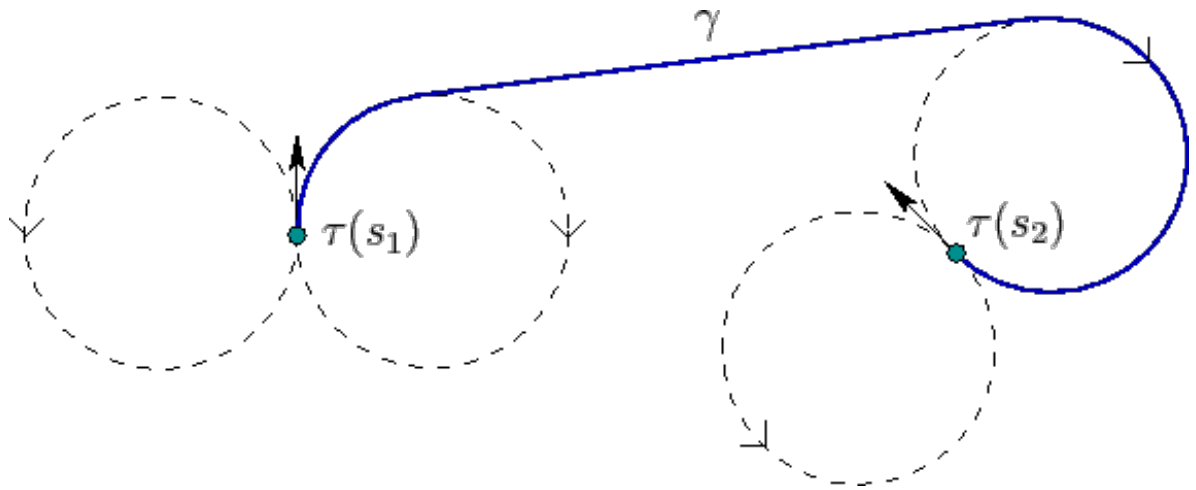




IC = Instantaneous Center of Rotation
The center of the circle circumscribed by the turning path.
Undefined for straight path segments.

Dubins car: motion primitives

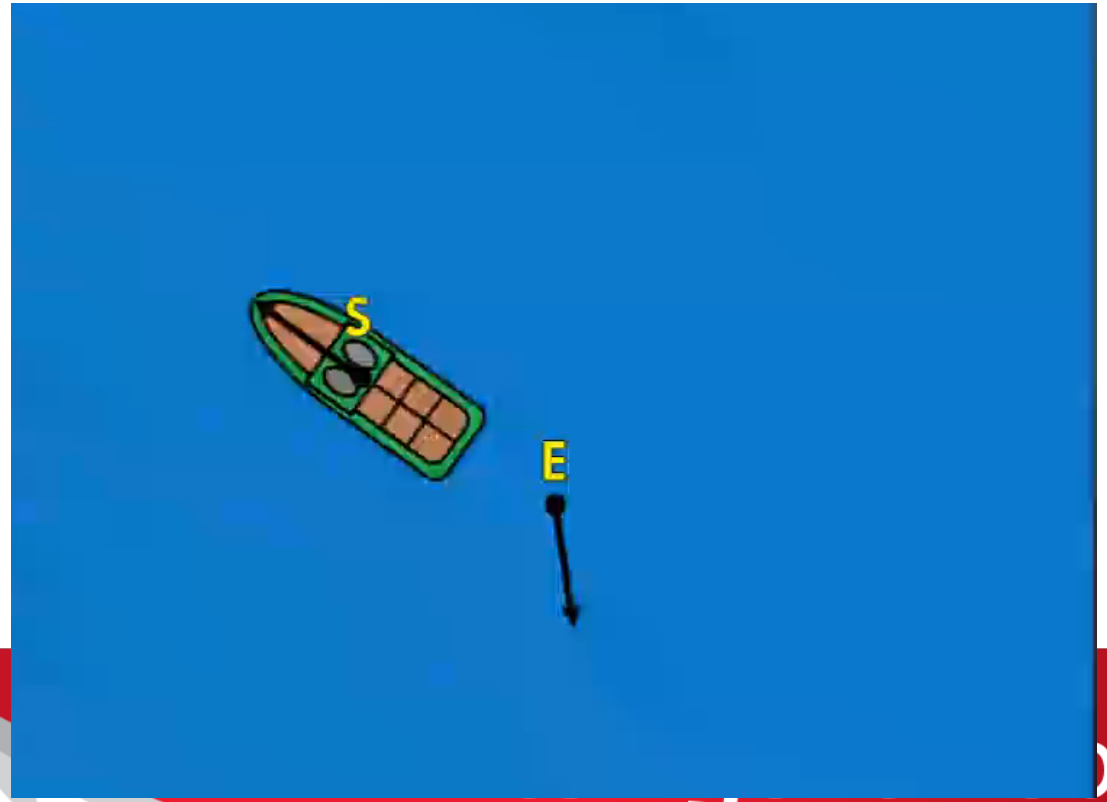
- Optimal paths of the car can be decomposed to L(eft), R(ight), S(traight) segments.
- Planning paths of this nature for various start/end is already an interesting challenge worthy of some thinking (often an assignment)



RSR path

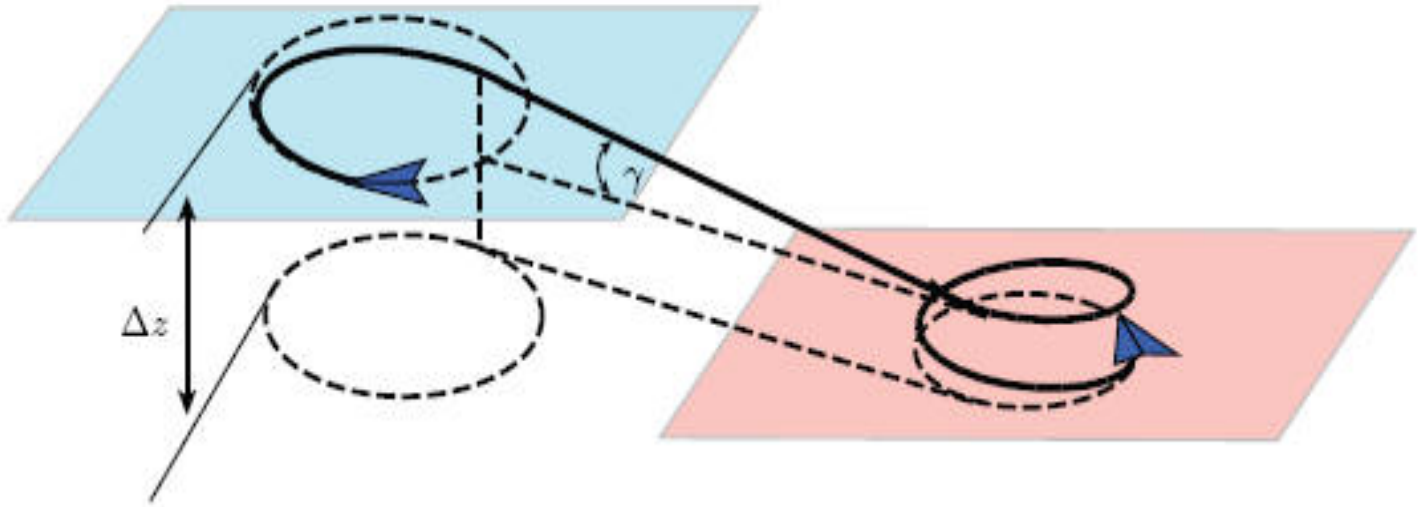
Dubins car ⟶ Dubins boat

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



Dubins car ⟶ Dubins airplane in 3D

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



(a) The 3D view of the path.

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

θ is yaw

ϕ is pitch

Holonomic constraints

- Equality constraints on the state of the system, but not on the higher-order derivatives:

$$\mathbf{f}(\mathbf{x}, t) = 0$$

- For example, if you want to constrain the state to lie on a circle:

$$||\mathbf{x}||^2 - 1 = 0$$

- Another example: train tracks are a holonomic constraint.

Holonomic constraints

- Under only holonomic constraints, all of the local neighborhood is reachable

- We will have to search for time-varying shortest paths in the state space to move farther

$$\mathbf{f}(\mathbf{x}, t) = 0$$

Non-holonomic constraints

- Equality constraints that involve the derivatives of the state (e.g. velocity) in a way that it cannot be integrated out into holonomic constraints, i.e.

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, t) = 0$$

but not

$$\mathbf{f}(\mathbf{x}, t) = 0$$

Non-holonomic constraints

- With the time derivative (first or higher order) in the constraint, we can only reach a sub-set of the state-space neighborhood immediately
- Optimal paths lie on a manifold with more complex geometry which leads to more interesting estimation, planning and control
- We will primarily be interested with algorithms that can handle non-holonomic constraints

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, t) = 0$$

The Dubins car is non-holonomic

- Dubins car is constrained to move straight towards the direction it is currently heading. It cannot move sideways. It needs to

  "parallel park" to move laterally.

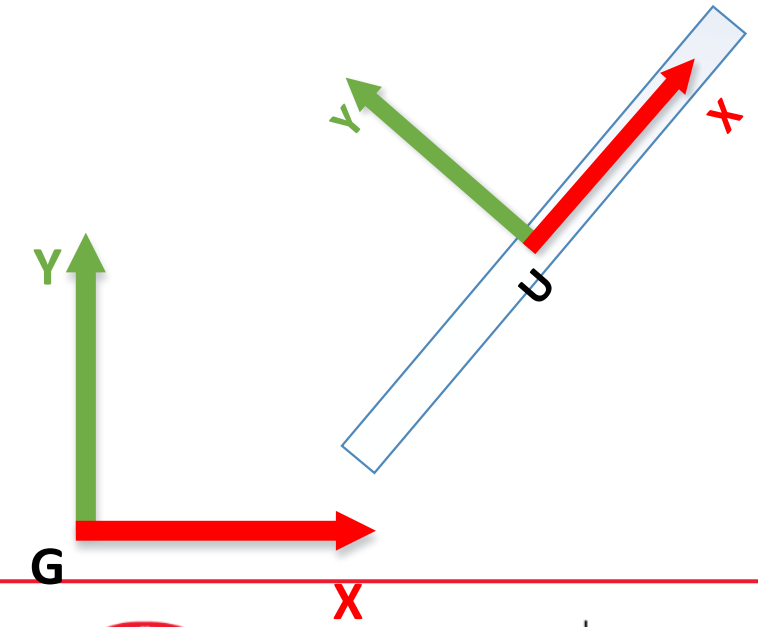
- In a small time interval dt the vehicle is going to move by δp_x and δp_y in the global frame of reference. Then from the dynamical system:

$$\begin{aligned} \delta p_x \sin(\theta) &= v_x \cos(\theta)\sin(\theta) \\ \delta p_y \cos(\theta) &= v_x \sin(\theta)\cos(\theta) \end{aligned} \Rightarrow \delta p_x \sin(\theta) = \delta p_y \cos(\theta) \Rightarrow v_x \sin(\theta) = v_y \cos(\theta)$$

Car is constrained to move along the line of current heading, i.e. non-holonomic

The state of a unicycle

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



Top view of a unicycle

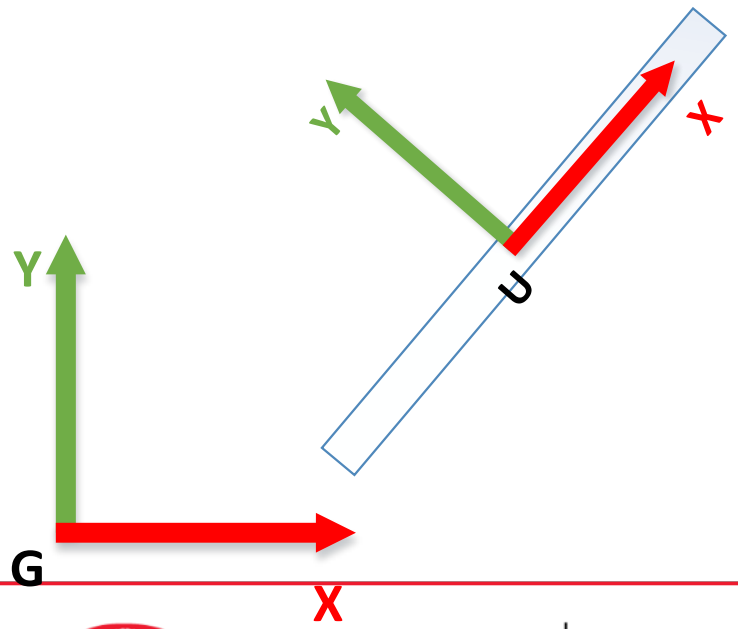
State = [Position, Orientation]
Position of the unicycle's frame of reference U with respect to a fixed frame of reference G, expressed in coordinates of frame G. Angle is the orientation of frame U with respect to frame G.

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

The state of a unicycle

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

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

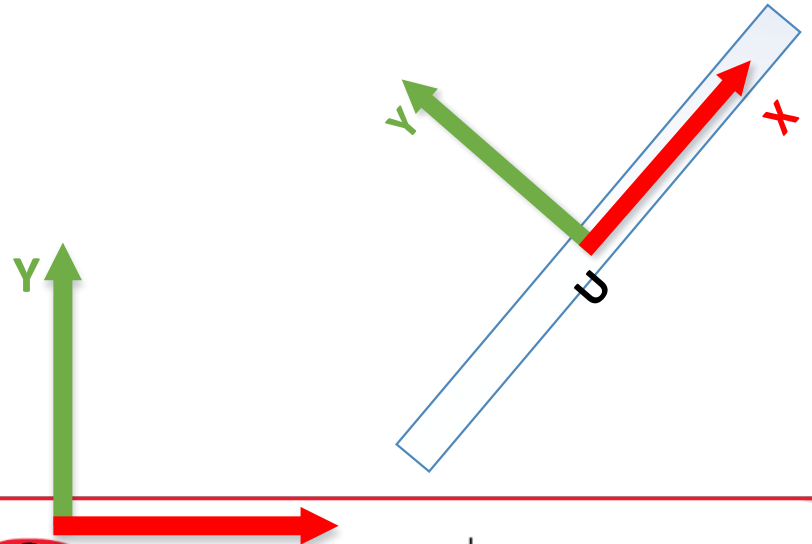


Top view of a unicycle.

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

Controls of a unicycle

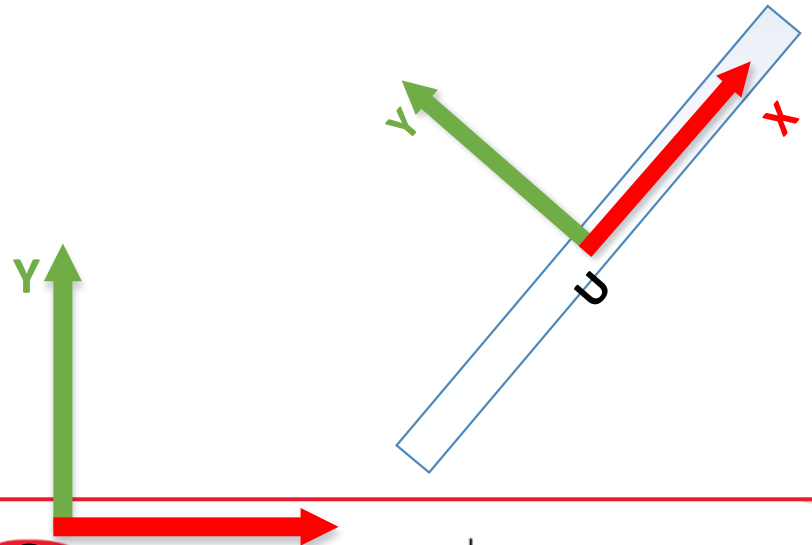
$$\mathbf{u} = [{}^{U}\omega_z, {}^{U}\omega_y]$$



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

Velocities of a unicycle

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

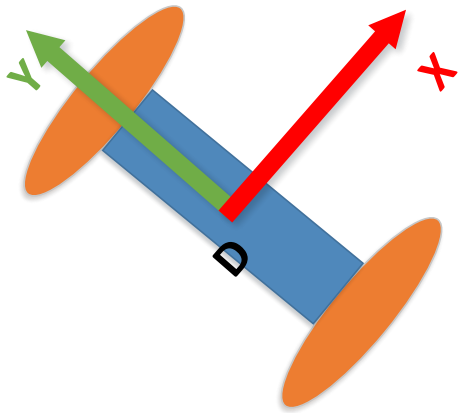


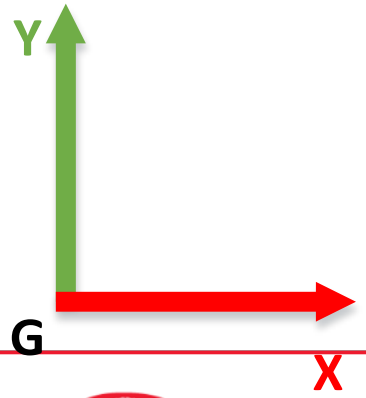
r = the radius of the wheel

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

The state of a differential drive vehicle

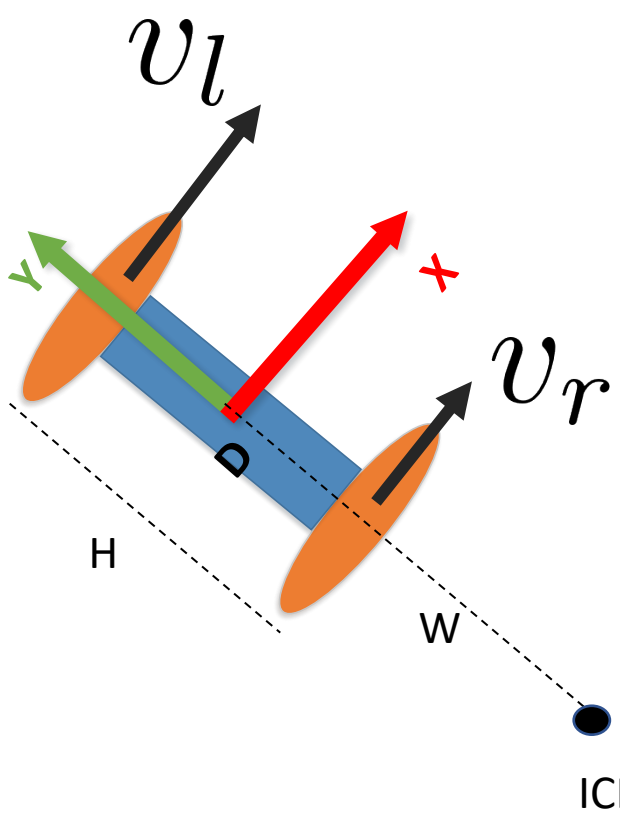
$$\mathbf{x} = [{}^{G}p_x, {}^{G}p_y, {}^{G}\theta]$$





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

Controls of a differential drive vehicle



$$\mathbf{u} = \left[{}^{D}\omega_l, {}^{D}\omega_r\right]$$

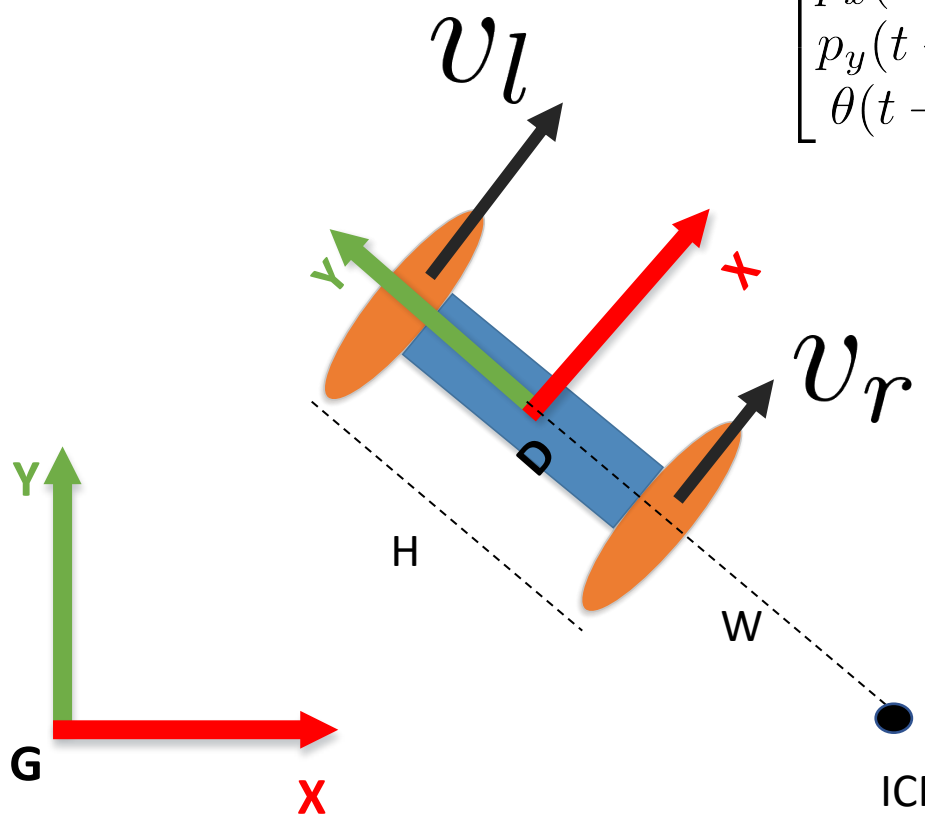
Controls = [Left wheel and right wheel turning rates]
Wheel turning rates determine the linear velocities of the respective wheels of the vehicle's frame of reference D with respect to a fixed frame of reference G, expressed in coordinates of D.

$$v_l = (W - H/2)\omega_l$$

$$v_r = (W + H/2)\omega_r$$

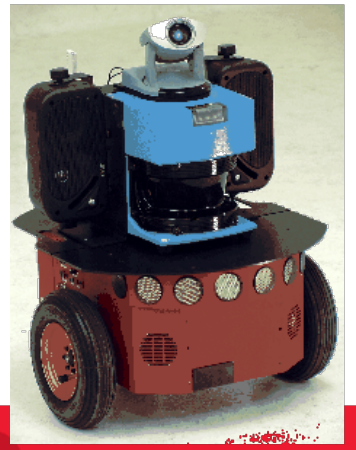
$$v_x = (v_l + v_r)/2$$

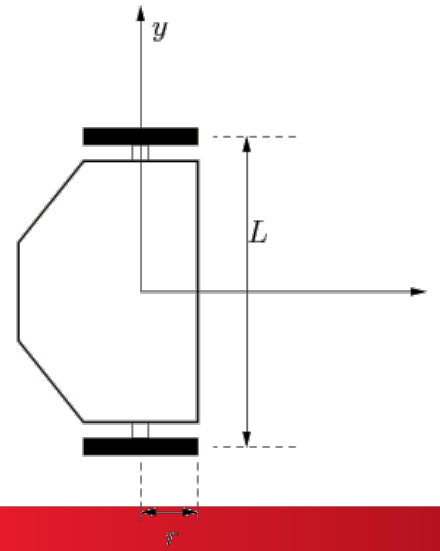
Velocities of a differential drive vehicle



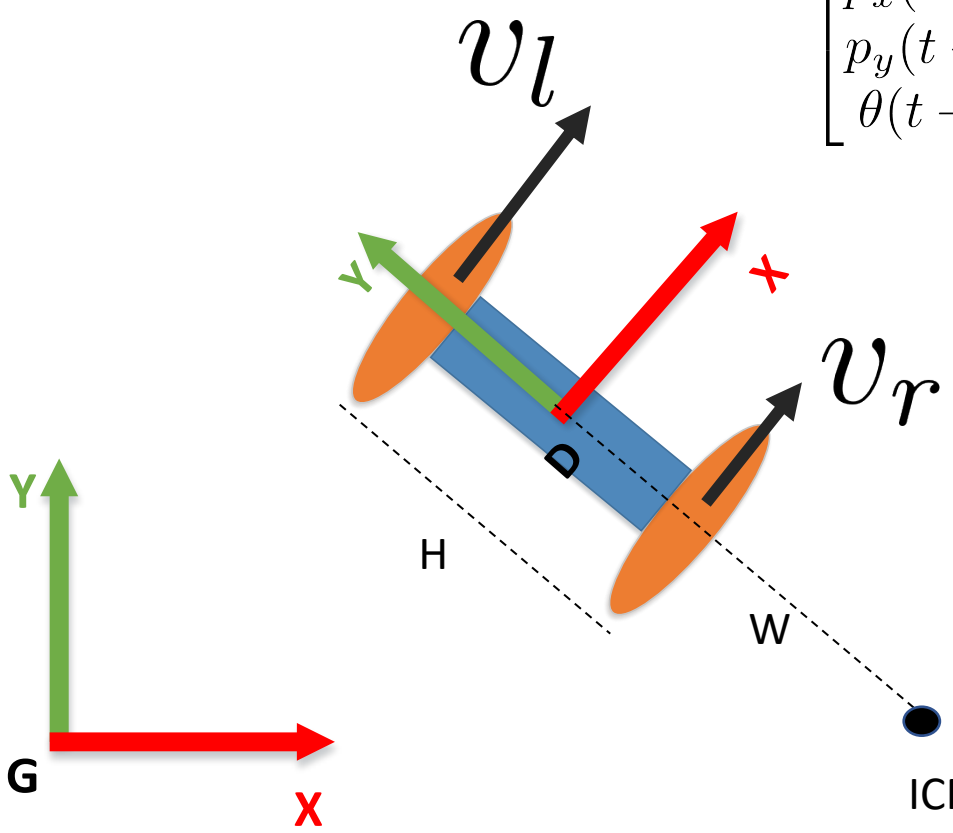
$$\begin{bmatrix} p_x(t+1) \\ p_y(t+1) \\ \theta(t+1) \end{bmatrix} = \begin{bmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x(t) - \mathrm{ICR}_x \\ p_y(t) - \mathrm{ICR}_y \\ \theta(t) \end{bmatrix}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

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





Dynamics of a differential drive vehicle



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

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

Special cases:

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

Kinematics and Dynamics as algorithms: both have a forward and inverse query

- Kinematics:
  - Forward – what robot position results from a given configuration?
  - Inverse – which configuration(s) gives a desired position?

- Dynamics:
  - Forward – what robot motion will result from given input forces?
  - Inverse – which input forces will result in a desired robot motion?

Forward and Inverse Kinematics of the differential drive car

- Kinematics:
  - Forward – after rotating the left wheel by 360 deg/sec and right wheel by 720 deg/sec for one second, where is the centre of the vehicle, and at what angle?
  - Inverse – suppose we want to maintain the same centre location, but rotate the body by 180 degrees, and accomplish this in one second, what fixed rates should be used for the left and right wheel?

More Important Properties of State Spaces and Kinematics

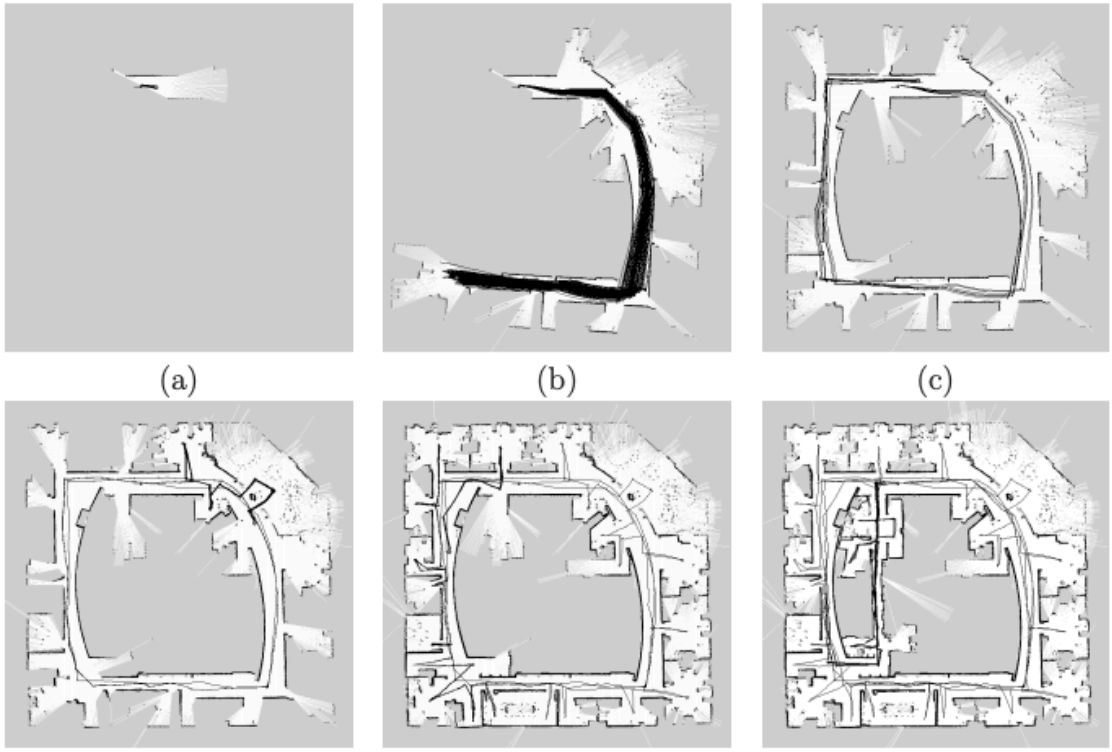
- Dimensionality: What minimum-sized vector is needed to define the state?

- Connectedness: Can every point in the space be reached from every other point?
  - If disconnected, the state space can have 2 or many components

- Topology: Are there ”holes” in the space, defined by the inability to continuously deform path A into path B

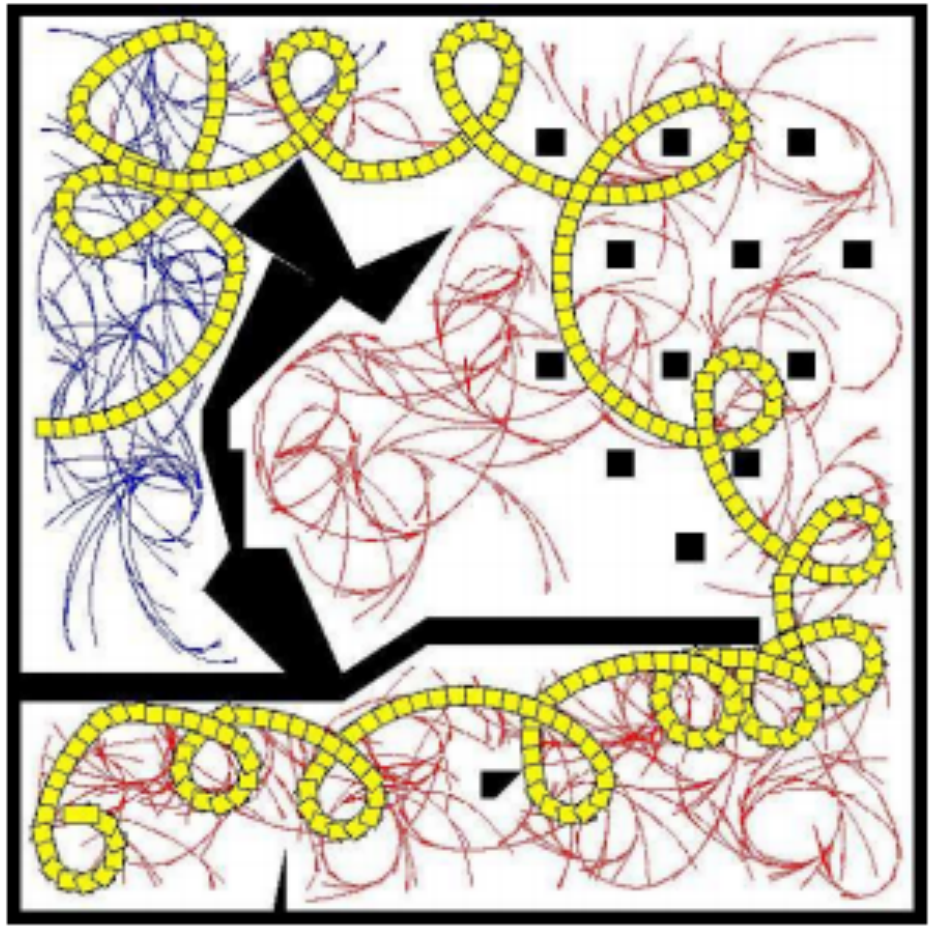
Why are these representations important? To build Maps and Plans

- A map of the environment adds external constraints on the where the robot can be
- Maps can be given to the robot (such as an architect's floorplan) or built by the robot using its sensors
- To be useful, a map should relate to the robot's state space



Why are these representations important? To build Maps and Plans

- A ***plan*** is a sequence of states for a robot within a map
- A plan is ***feasible*** if each state in the sequence, and the motion between consecutive pairs are all feasible under the kinematics and the map
- Planning algorithms can be:
  - ***Correct*** if every plan they output is feasible
  - ***Complete*** if they find a plan whenever one exists



Consider the planning algorithm for the Dubins Car robot

- Input:
  - Current state of the robot
  - Map
  - Goal state
- Output:
  - A plan, as a sequence of states that follow Dubins constraints, if one exists
  - A “fail” flag, if not
- How would you code this solution? We’ll cover many ideas in our planning section!