CS689 - Robot Motion Planning

Motion Planning with Kinematics and Dynamics

Amarda Shehu

Department of Computer Science
George Mason University

Beyond Motion Planning with Geometric Constraints

- Geometric constraints are generally not sufficient to adequately express robot motions

[movie: Moving Car 1]

Beyond Motion Planning with Geometric Constraints

- Geometric constraints are generally not sufficient to adequately express robot motions

[movie: Moving Car 1]

- Are the motions realistic?
- What is missing?

Motion Planning with Kinematic Constraints

- Actual car steering and constraints on velocity may make planned motions more realistic.

[movie: Moving Car 2]

Motion Planning with Kinematic Constraints

- Actual car steering and constraints on velocity may make planned motions more realistic.

[movie: Moving Car 2]

- Are the motions more realistic?
- Can they be made more realistic?
- How?

Motion Planning with Kinematic and Dynamic Constraints

- Geometric constraints are generally not sufficient to adequately express robot motions
- Constraints on velocity, forces, torques, accelerations are needed for better representations

[movie: Moving Car 1 - Geometric]

[movie: Moving Car 1 - Kinematics]

[movie: Moving Car 3 - Dynamics]

Kinematics Constraints == Constraints on Velocity

Illustration:

- C-space $= \mathbb{R}^2 = \{q = (x, y) \in \mathbb{R}^2\}$
- Velocity $\frac{dq}{dt} = \dot{q} = (\dot{x}, \dot{y})$
- Each $(\dot{x}, \dot{y})$ is an element of the tangent space $T_q(\mathbb{R}^2)$, which is a 2D vector space at every (x, y)
- At each $q \in \mathbb{R}^2$, restricting the set of velocities yields some set $U(q) \subset T_q(\mathbb{R}^2)$
- Think about the kinds of constraints imposed on velocity

Kinematics Constraints == Constraints on Velocity

Eamples of interesting yet simple constraints:

Eamples of interesting yet simple constraints:

- $\dot{x} > 0$:

Kinematics Constraints == Constraints on Velocity

Eamples of interesting yet simple constraints:

- $\dot{x} > 0$: Paddling boat on river flowing in positive x direction.

Eamples of interesting yet simple constraints:

- $\dot{x} > 0$: Paddling boat on river flowing in positive x direction.
- $\dot{x} \geq 0$:

Kinematics Constraints == Constraints on Velocity

Eamples of interesting yet simple constraints:

- $\dot{x} > 0$: Paddling boat on river flowing in positive x direction.
- $\dot{x} \geq 0$: Allows to stop in x direction; e.g. following velocity in y.

Eamples of interesting yet simple constraints:

- $\dot{x} > 0$: Paddling boat on river flowing in positive x direction.
- $\dot{x} \geq 0$: Allows to stop in x direction; e.g. following velocity in y.
- $\dot{x} = 0$:

Eamples of interesting yet simple constraints:

- $\dot{x} > 0$: Paddling boat on river flowing in positive x direction.
- $\dot{x} \geq 0$: Allows to stop in x direction; e.g. following velocity in y.
- $\dot{x} = 0$: No motion in x direction; $(0, \dot{y})$ allowed.

Eamples of interesting yet simple constraints:

- $\dot{x} > 0$: Paddling boat on river flowing in positive x direction.
- $\dot{x} \geq 0$: Allows to stop in x direction; e.g. following velocity in y.
- $\dot{x} = 0$: No motion in x direction; $(0, \dot{y})$ allowed.
- $\dot{x} + \dot{y} \leq 1$:

Kinematics Constraints == Constraints on Velocity

Eamples of interesting yet simple constraints:

- $\dot{x} > 0$: Paddling boat on river flowing in positive x direction.
- $\dot{x} \geq 0$: Allows to stop in x direction; e.g. following velocity in y.
- $\dot{x} = 0$: No motion in x direction; $(0, \dot{y})$ allowed.
- $\dot{x} + \dot{y} \leq 1$: Constraint enforces maximum speed.

Kinematics Constraints == Constraints on Velocity

Eamples of interesting yet simple constraints:

- $\dot{x} > 0$: Paddling boat on river flowing in positive x direction.
- $\dot{x} \geq 0$: Allows to stop in x direction; e.g. following velocity in y.
- $\dot{x} = 0$: No motion in x direction; $(0, \dot{y})$ allowed.
- $\dot{x} + \dot{y} \leq 1$: Constraint enforces maximum speed.
- $\dot{x} + \dot{y} \geq 1$:

Kinematics Constraints == Constraints on Velocity

Eamples of interesting yet simple constraints:

- $\dot{x} > 0$: Paddling boat on river flowing in positive x direction.
- $\dot{x} \geq 0$: Allows to stop in x direction; e.g. following velocity in y.
- $\dot{x} = 0$: No motion in x direction; $(0, \dot{y})$ allowed.
- $\dot{x} + \dot{y} \leq 1$: Constraint enforces maximum speed.
- $\dot{x} + \dot{y} \geq 1$: Impossible to stop or slow down.

Implicit vs. Parametric Kinematics Constraints

Implicit and parametric representations are alternative ways to express

$$U(q) \quad \forall q \in \mathbb{R}^2.$$

- Implicit (indirect) representation: expresses velocities that are not allowed.
- Parametric (direct) representation: expresses velocities that are allowed.

Implicit Velocity Constraints

Implicit velocity constraints express velocities that are <u>not allowed</u> and are of the form:

$$g(q, \dot{q}) \bowtie 0$$

where

- $g(q, \dot{q})$ is some function $g : Q \times \dot{Q} \rightarrow \mathbb{R}$
- $\bowtie$ can be any of the symbols $=, <, >, \leq, \geq$

Implicit Velocity Constraints

Implicit velocity constraints express velocities that are not allowed and are of the form:

$$g(q, \dot{q}) \bowtie 0$$

where

- $g(q, \dot{q})$ is some function $g : Q \times \dot{Q} \to \mathbb{R}$
- $\bowtie$ can be any of the symbols $=, <, >, \leq, \geq$

Example of point in plane

- configuration: $q = (x, y) \in \mathbb{R}^2$
- velocity: $\frac{dq}{dt} = \dot{q} = (\dot{x}, \dot{y})$

Implicit Velocity Constraints

Implicit velocity constraints express velocities that are not allowed and are of the form:

$$g(q, \dot{q}) \bowtie 0$$

where

- $g(q, \dot{q})$ is some function $g : Q \times \dot{Q} \rightarrow \mathbb{R}$
- $\bowtie$ can be any of the symbols $=, <, >, \leq, \geq$

Example of point in plane

- configuration: $q = (x, y) \in \mathbb{R}^2$
- velocity: $\frac{dq}{dt} = \dot{q} = (\dot{x}, \dot{y})$

Examples of implicit velocity constraints

- $\dot{x} > 0$, $\dot{x} = 0$, $\dot{x}^2 + \dot{y}^2 \leq 1$, $x = \dot{x}$, etc.

Parametric Velocity Constraints

Parametric velocity constraints express velocities that are allowed and are of the form:

$$\dot{q} = f(q, u)$$

where

- $f(q, u)$ is some function $f : Q \times U \rightarrow \dot{Q}$ that expresses a set of differential equations.

  f is referred to as the configuration transition equation
- u is an input control/action.
- So, $T_q(Q)$ is *parameterized* through u: Given a (sampled?) control/action, one can obtain an *allowed* velocity.

Parametric Velocity Constraints

Parametric velocity constraints express velocities that are allowed and are of the form:

$$\dot{q} = f(q, u)$$

where

- $f(q, u)$ is some function $f : Q \times U \to \dot{Q}$ that expresses a set of differential equations. Why a set?

  f is referred to as the configuration transition equation

- u is an input control/action.

- So, $T_q(Q)$ is *parameterized* through u: Given a (sampled?) control/action, one can obtain an *allowed* velocity.

Parametric Velocity Constraints

Parametric velocity constraints express velocities that are allowed and are of the form:

$$\dot{q} = f(q, u)$$

where

- $f(q, u)$ is some function $f : Q \times U \to \dot{Q}$ that expresses a set of differential equations. Why a set?

  f is referred to as the configuration transition equation

- u is an input control/action.

- So, $T_q(Q)$ is *parameterized* through u: Given a (sampled?) control/action, one can obtain an *allowed* velocity.

Let's work out the *kinematics* of some simple *wheeled* systems.

Kinematics for Wheeled Systems

- Objective 1: Derive configuration transition equation (*do the kinematics*) for wheeled systems (car, differential drive, and unicycle).

  Constrain velocities for more realistic motions.

- Objective 2: Proceed with dynamics after working out kinematics.

  Constrain accelerations for even more realistic motions.

a simple car as opposed to other car variations

Objective: Obtain f as in $\dot{q} = f(q, u)$.

Preliminaries:

- Car cannot drive sideways because

  .

- Parallel parking would be trivial

- Complicated maneuvers arise due to rolling constraints.

a simple car as opposed to other car variations

Objective: Obtain f as in $\dot{q} = f(q, u)$.

Preliminaries:

- Car cannot drive sideways because back wheels roll instead of slide (which is why parallel parking is challenging).
- Parallel parking would be trivial
- Complicated maneuvers arise due to rolling constraints.

Kinematics for Wheeled Systems – A Simple Car

a simple car as opposed to other car variations

Objective: Obtain f as in $\dot{q} = f(q, u)$.

Preliminaries:

- Car cannot drive sideways because back wheels roll instead of slide (which is why parallel parking is challenging).
- Parallel parking would be trivial if all four wheels could simultaneously be turned towards the curb.
- Complicated maneuvers arise due to rolling constraints.

a simple car as opposed to other car variations

Objective: Obtain f as in $\dot{q} = f(q, u)$.

Preliminaries:

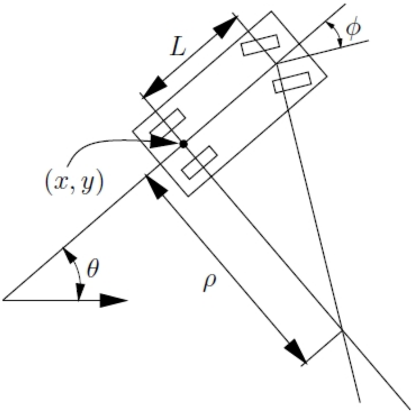
- Car cannot drive sideways because back wheels roll instead of slide (which is why parallel parking is challenging).
- Parallel parking would be trivial if all four wheels could simultaneously be turned towards the curb.
- Complicated maneuvers arise due to rolling constraints.
- Need: model for car

a simple car as opposed to other car variations

Objective: Obtain f as in $\dot{q} = f(q, u)$.

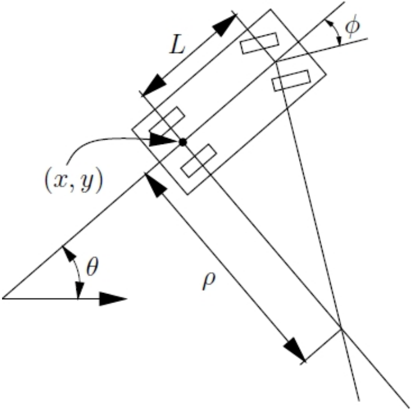
Preliminaries:

- Car cannot drive sideways because back wheels roll instead of slide (which is why parallel parking is challenging).
- Parallel parking would be trivial if all four wheels could simultaneously be turned towards the curb.
- Complicated maneuvers arise due to rolling constraints.
- Need: model for car
- Need: understand way car moves (what do we control?)



- Car: rigid body that moves in plane.
- Car configuration:
  $q = (x, y, \theta) \in \mathbb{R} \times S^1$
- Body frame:
  - Origin is at the center of rear axle
  - x-axis points along main axis of the car
- Velocity (signed speed in x direction of body frame): s
- Steering angle: ϕ
- Distance between front and rear axles: L

Kinematics for Wheeled Systems – A Simple Car

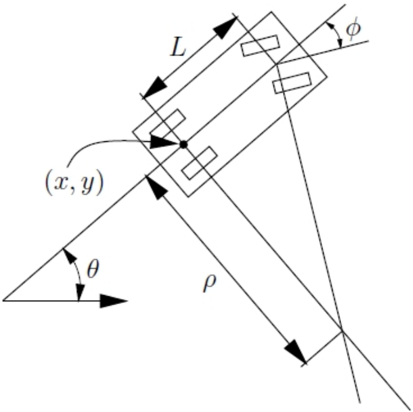


How does the car move?

- If steering angle ϕ is kept fixed, car travels in circular motion.
- Center of circle: intersection between normals to steering axis and car axis.
- Radius of circle: ρ

Need : Express car motions as a set of differential equations

- $\dot{x} = f_1(q, u)$
- $\dot{y} = f_2(q, u)$
- $\dot{\theta} = f_3(q, u)$

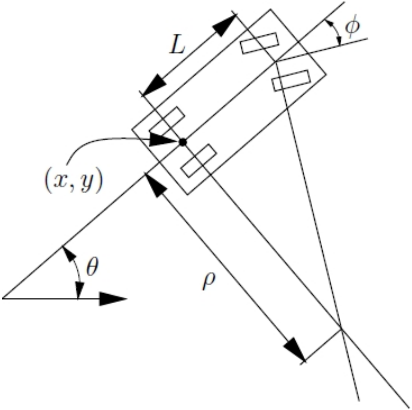


How does the car move?

- If steering angle ϕ is kept fixed, car travels in circular motion.
- Center of circle: intersection between normals to steering axis and car axis.
- Radius of circle: ρ

Need : Express car motions as a set of differential equations

- $\dot{x} = f_1(q, u) = f_1(x, y, \theta, s, \phi)$
- $\dot{y} = f_2(q, u)$
- $\dot{\theta} = f_3(q, u)$

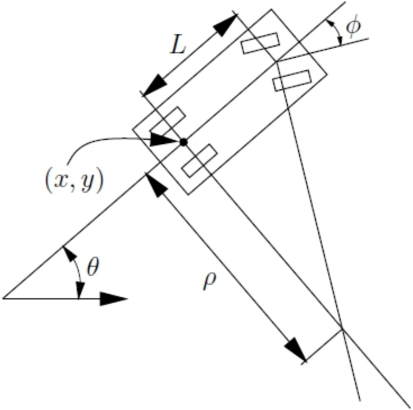


How does the car move?

- If steering angle ϕ is kept fixed, car travels in circular motion.
- Center of circle: intersection between normals to steering axis and car axis.
- Radius of circle: ρ

Need : Express car motions as a set of differential equations

- $\dot{x} = f_1(q, u) = f_1(x, y, \theta, s, \phi)$
- $\dot{y} = f_2(q, u) = f_2(x, y, \theta, s, \phi)$
- $\dot{\theta} = f_3(q, u)$

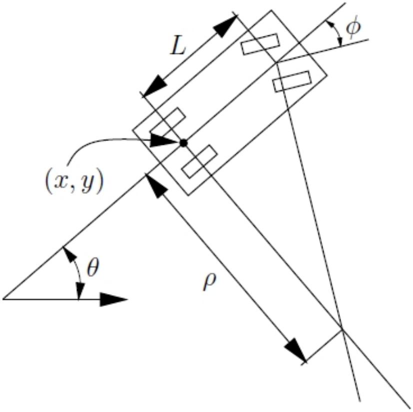


How does the car move?

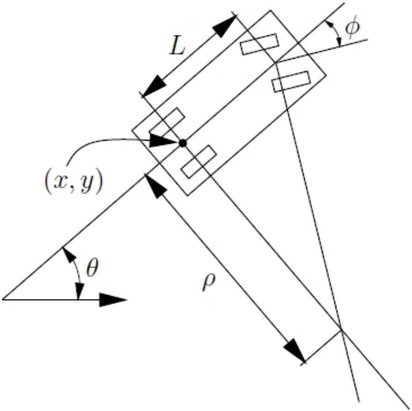
- If steering angle ϕ is kept fixed, car travels in circular motion.
- Center of circle: intersection between normals to steering axis and car axis.
- Radius of circle: ρ

Need : Express car motions as a set of differential equations

- $\dot{x} = f_1(q, u) = f_1(x, y, \theta, s, \phi)$
- $\dot{y} = f_2(q, u) = f_2(x, y, \theta, s, \phi)$
- $\dot{\theta} = f_3(q, u) = f_3(x, y, \theta, s, \phi)$



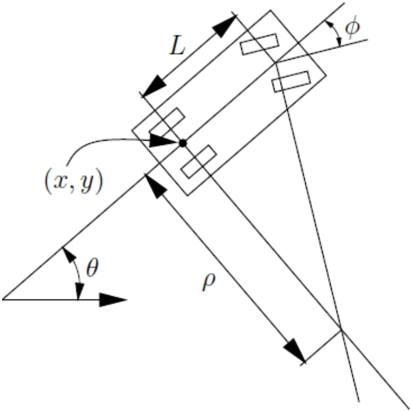
How does the car move?



How does the car move?

- In small time interval Δt, car must move approximately in direction rear wheels are pointing

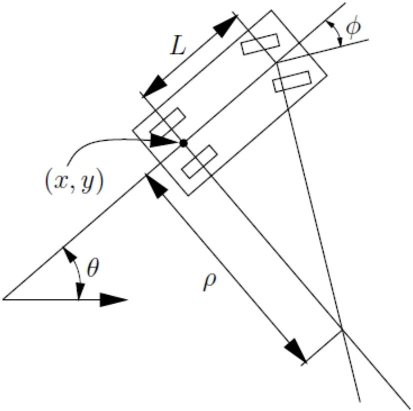
Kinematics for Wheeled Systems – A Simple Car



How does the car move?

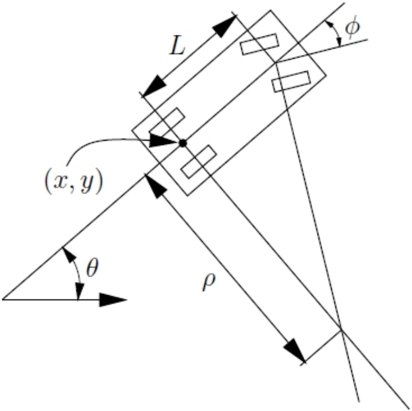
- In small time interval Δt, car must move approximately in direction rear wheels are pointing
- In the limit, as $\Delta t \to 0$, then $\frac{dy}{dx} = \tan\theta$, i.e.,
  $-\dot{x}\sin\theta + \dot{y}\cos\theta = 0$

Kinematics for Wheeled Systems – A Simple Car



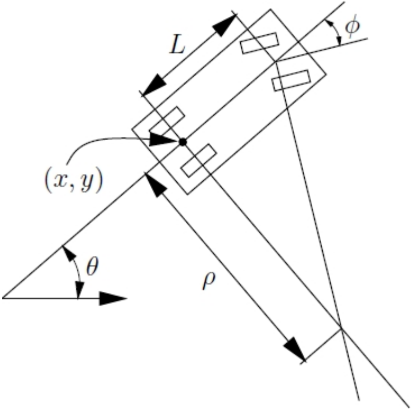
How does the car move?

- In small time interval Δt, car must move approximately in direction rear wheels are pointing
- In the limit, as $\Delta t \to 0$, then $\frac{dy}{dx} = \tan\theta$, i.e., $-\dot{x}\sin\theta + \dot{y}\cos\theta = 0$
- Solution is of the form $\dot{x} = s\cos\theta$ and $\dot{y} = s\sin\theta$

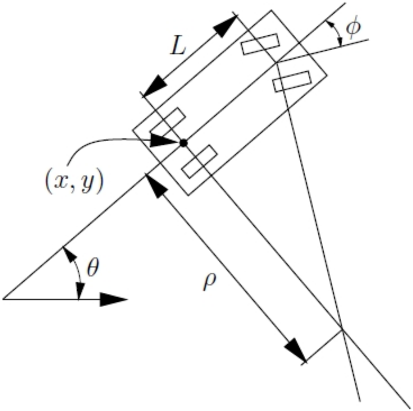


How does the car move?

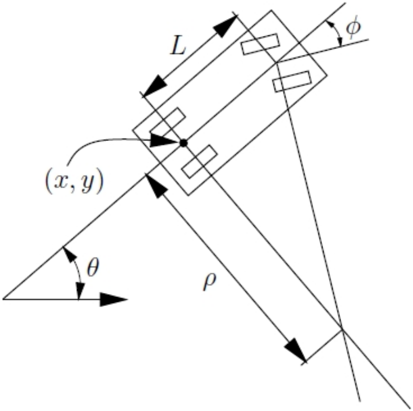
- In small time interval Δt, car must move approximately in direction rear wheels are pointing
- In the limit, as $\Delta t \to 0$, then $\frac{dy}{dx} = \tan\theta$, i.e.,
  $-\dot{x}\sin\theta + \dot{y}\cos\theta = 0$
- Solution is of the form $\dot{x} = s\cos\theta$ and $\dot{y} = s\sin\theta$
- What about $\dot{\theta}$?



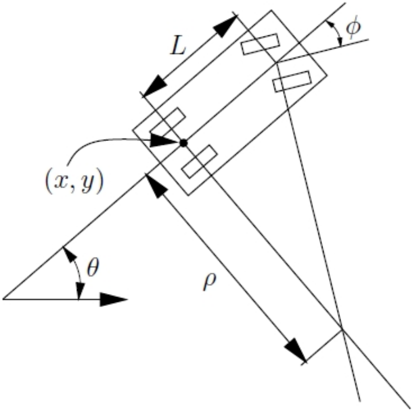
- Need to put car at consecutive configs on circle to see $\Delta\theta$ as $\Delta t \rightarrow 0$.



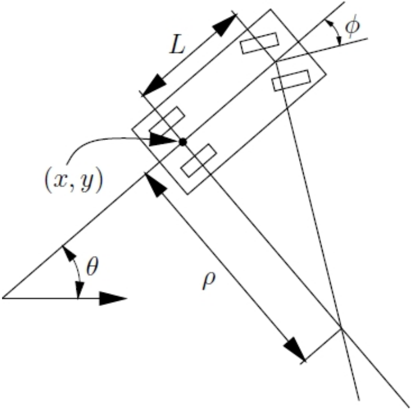
- Need to put car at consecutive configs on circle to see $\Delta\theta$ as $\Delta t \rightarrow 0$.
- $L/\rho = tan(\phi)$ [angle relations]



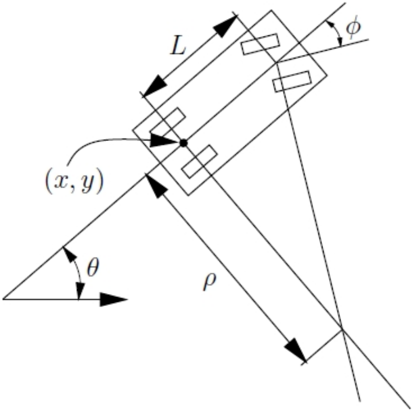
- Need to put car at consecutive configs on circle to see $\Delta\theta$ as $\Delta t \rightarrow 0$.
- $L/\rho = tan(\phi)$ [angle relations]
- Let w be distance traveled on circle



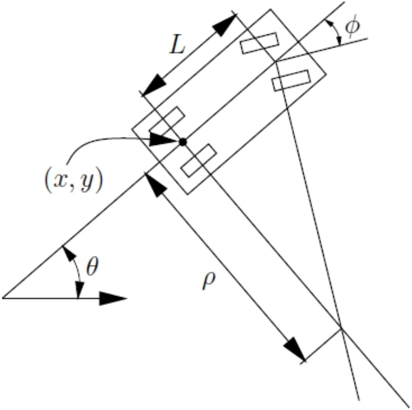
- Need to put car at consecutive configs on circle to see $\Delta\theta$ as $\Delta t \rightarrow 0$.
- $L/\rho = tan(\phi)$ [angle relations]
- Let w be distance traveled on circle
- What is translational velocity dw relative to s?



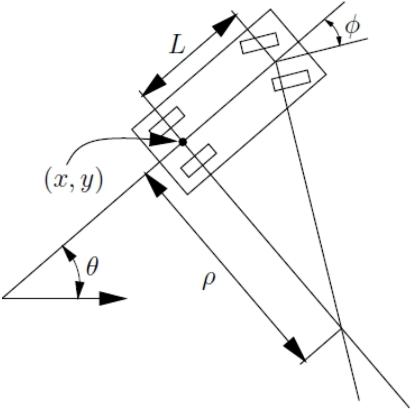
- Need to put car at consecutive configs on circle to see $\Delta\theta$ as $\Delta t \to 0$.
- $L/\rho = tan(\phi)$ [angle relations]
- Let w be distance traveled on circle
- What is translational velocity dw relative to s? $dw = s$.



- Need to put car at consecutive configs on circle to see $\Delta\theta$ as $\Delta t \to 0$.
- $L/\rho = tan(\phi)$ [angle relations]
- Let w be distance traveled on circle
- What is translational velocity dw relative to s? $dw = s$.
- How does translational velocity $dw = s$ relate to angular velocity $d\theta$?



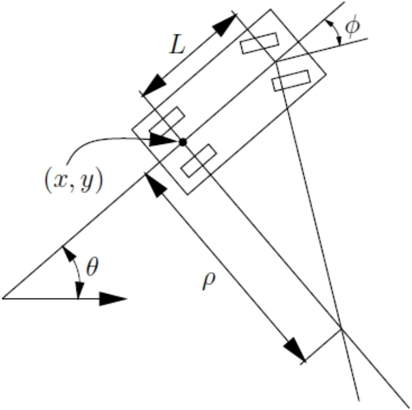
- Need to put car at consecutive configs on circle to see $\Delta\theta$ as $\Delta t \to 0$.
- $L/\rho = tan(\phi)$ [angle relations]
- Let w be distance traveled on circle
- What is translational velocity dw relative to s? $dw = s$.
- How does translational velocity dw $= s$ relate to angular velocity $d\theta$? $dw = \rho d\theta$ [cord length]



- Need to put car at consecutive configs on circle to see $\Delta\theta$ as $\Delta t \rightarrow 0$.
- $L/\rho = tan(\phi)$ [angle relations]
- Let w be distance traveled on circle
- What is translational velocity dw relative to s? $dw = s$.
- How does translational velocity dw $= s$ relate to angular velocity $d\theta$? $dw = \rho d\theta$ [cord length]

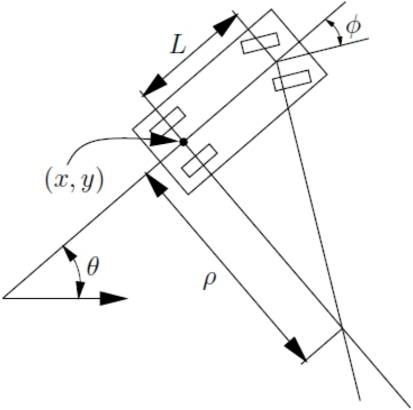
So, putting it all together:

$$d\theta = \frac{\tan\phi}{L} dw = \frac{\tan\phi}{L} s \Rightarrow \dot{\theta} = \frac{s}{L} \tan\phi$$



What we have so far on configuration transition equations:

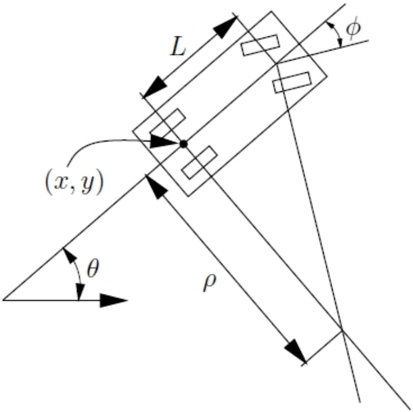
Kinematics for Wheeled Systems – A Simple Car



What we have so far on configuration transition equations:

- $\dot{x} = s \cdot cos(\theta)$
- $\dot{y} = s \cdot sin(\theta)$
- $\dot{\theta} = \frac{s}{L} \tan \phi$

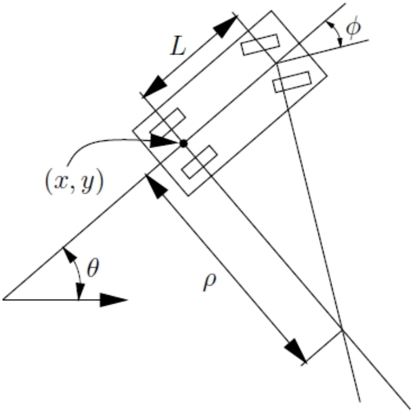
Kinematics for Wheeled Systems – A Simple Car



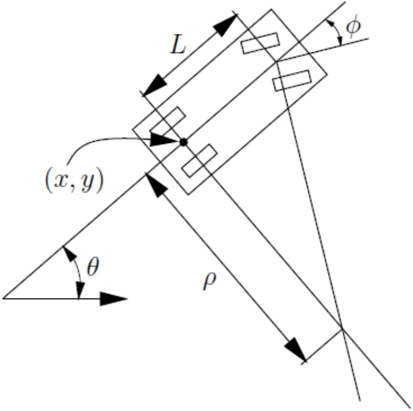
What we have so far on configuration transition equations:

- $\dot{x} = s \cdot cos(\theta)$
- $\dot{y} = s \cdot sin(\theta)$
- $\dot{\theta} = \frac{s}{L} \tan \phi$

How should we control the car? Where are our controls/actions?

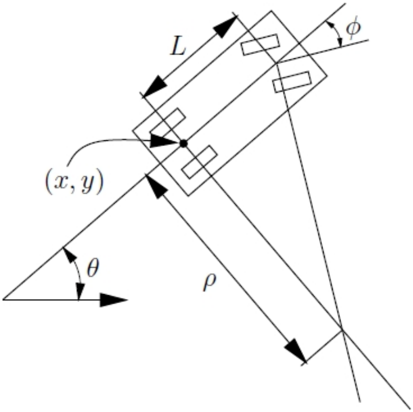


How should we control the car?



How should we control the car?

- Setting the speed s, i.e., $u_s = s$
- Setting the steering angle ϕ, i.e., $u_\phi = \phi$



How should we control the car?

- Setting the speed s, i.e., $u_s = s$
- Setting the steering angle ϕ, i.e., $u_\phi = \phi$

Putting it all together:

- Input controls: u_s (speed) and u_ϕ (steering angle)
- CTE:
  $\dot{x} = u_s \cos\theta$
  $\dot{y} = u_s \sin\theta$
  $\dot{\theta} = \frac{u_s}{L} \tan u_\phi$

Forward Kinematics for Simple Car

Problem formulation when only worrying about geometric constraints:

- Standing at configuration $q = (x, y, \theta)$, what are the workspace coordinates of the control points?

Forward Kinematics for Simple Car

Problem formulation when only worrying about geometric constraints:

- Standing at configuration $q = (x, y, \theta)$, what are the workspace coordinates of the control points?
- Approach: treat car as rigid-body in 2D workspace, use combination of rotation by θ followed by translation by center point (x, y).

Forward Kinematics for Simple Car

Problem formulation when only worrying about geometric constraints:

- Standing at configuration $q = (x, y, \theta)$, what are the workspace coordinates of the control points?
- Approach: treat car as rigid-body in 2D workspace, use combination of rotation by θ followed by translation by center point (x, y).

New problem formulation under kinematic constraints:

- Standing at configuration $q = (x, y, \theta)$ at time t, determine configuration $q^{'} = (x^{'}, y^{'}, \theta^{'})$ at time $t + \delta t$ given controls u_s, u_ϕ
- Approach: use CTE to obtain $\dot{q}$
- Update $q^{'} = q + \delta t \cdot \dot{q}$ [movie: Traj]

Forward Kinematics for Simple Car

Problem formulation when only worrying about geometric constraints:

- Standing at configuration $q = (x, y, \theta)$, what are the workspace coordinates of the control points?
- Approach: treat car as rigid-body in 2D workspace, use combination of rotation by θ followed by translation by center point (x, y).

New problem formulation under kinematic constraints:

- Standing at configuration $q = (x, y, \theta)$ at time t, determine configuration $q^{'} = (x^{'}, y^{'}, \theta^{'})$ at time $t + \delta t$ given controls u_s, u_ϕ
- Approach: use CTE to obtain $\dot{q}$
- Update $q^{'} = q + \delta t \cdot \dot{q}$ [movie: Traj]

How can I generate a random path in configuration space?

Forward Kinematics for Simple Car

Problem formulation when only worrying about geometric constraints:

- Standing at configuration $q = (x, y, \theta)$, what are the workspace coordinates of the control points?
- Approach: treat car as rigid-body in 2D workspace, use combination of rotation by θ followed by translation by center point (x, y).

New problem formulation under kinematic constraints:

- Standing at configuration $q = (x, y, \theta)$ at time t, determine configuration $q^{'} = (x^{'}, y^{'}, \theta^{'})$ at time $t + \delta t$ given controls u_s, u_ϕ
- Approach: use CTE to obtain $\dot{q}$
- Update $q^{'} = q + \delta t \cdot \dot{q}$ [movie: Traj]

How can I generate a random path in configuration space?
Sample values for controls.

Forward Kinematics for Simple Car

Problem formulation when only worrying about geometric constraints:

- Standing at configuration $q = (x, y, \theta)$, what are the workspace coordinates of the control points?
- Approach: treat car as rigid-body in 2D workspace, use combination of rotation by θ followed by translation by center point (x, y).

New problem formulation under kinematic constraints:

- Standing at configuration $q = (x, y, \theta)$ at time t, determine configuration $q^{'} = (x^{'}, y^{'}, \theta^{'})$ at time $t + \delta t$ given controls u_s, u_ϕ
- Approach: use CTE to obtain $\dot{q}$
- Update $q^{'} = q + \delta t \cdot \dot{q}$ [movie: Traj]

How can I generate a random path in configuration space?
Sample values for controls. Should there be bounds?

Variations of the Simple Car Model

- Input controls: u_s (speed) and u_ϕ (steering angle)
- CTE: $\dot{x} = u_s \cos\theta$, $\dot{y} = u_s \sin\theta$, $\dot{\theta} = \frac{u_s}{L} \tan u_\phi$
  What are bounds on steering angle and speed? Why bound speed?

Variations of the Simple Car Model

- Input controls: u_s (speed) and u_ϕ (steering angle)
- CTE:$\dot{x} = u_s \cos\theta$, $\dot{y} = u_s \sin\theta$, $\dot{\theta} = \frac{u_s}{L} \tan u_\phi$
  What are bounds on steering angle and speed? Why bound speed?

Tricycle

- $u_s \in [-1, 1]$ and $u_\phi \in [-\pi/2, \pi/2]$.

Variations of the Simple Car Model

- Input controls: u_s (speed) and u_ϕ (steering angle)
- CTE: $\dot{x} = u_s \cos\theta$, $\dot{y} = u_s \sin\theta$, $\dot{\theta} = \frac{u_s}{L} \tan u_\phi$
  What are bounds on steering angle and speed? Why bound speed?

Tricycle

- $u_s \in [-1, 1]$ and $u_\phi \in [-\pi/2, \pi/2]$. Can it rotate in place?

Variations of the Simple Car Model

- Input controls: u_s (speed) and u_ϕ (steering angle)
- CTE: $\dot{x} = u_s \cos\theta$, $\dot{y} = u_s \sin\theta$, $\dot{\theta} = \frac{u_s}{L} \tan u_\phi$
  What are bounds on steering angle and speed? Why bound speed?

Tricycle

- $u_s \in [-1, 1]$ and $u_\phi \in [-\pi/2, \pi/2]$. Can it rotate in place?

Standard simple car

- $u_s \in [-1, 1]$ and $u_\phi \in (-\phi_{\text{max}}, \phi_{\text{max}})$ for some $\phi_{\text{max}} < \pi/2$.

Variations of the Simple Car Model

- Input controls: u_s (speed) and u_ϕ (steering angle)
- CTE:$\dot{x} = u_s \cos\theta$, $\dot{y} = u_s \sin\theta$, $\dot{\theta} = \frac{u_s}{L} \tan u_\phi$
  What are bounds on steering angle and speed? Why bound speed?

Tricycle

- $u_s \in [-1, 1]$ and $u_\phi \in [-\pi/2, \pi/2]$. Can it rotate in place?

Standard simple car

- $u_s \in [-1, 1]$ and $u_\phi \in (-\phi_{\text{max}}, \phi_{\text{max}})$ for some $\phi_{\text{max}} < \pi/2$.
  What happens at $\pi/2$ and $-\pi/2$?

Variations of the Simple Car Model

- Input controls: u_s (speed) and u_ϕ (steering angle)
- CTE:$\dot{x} = u_s \cos\theta$, $\dot{y} = u_s \sin\theta$, $\dot{\theta} = \frac{u_s}{L} \tan u_\phi$
  What are bounds on steering angle and speed? Why bound speed?

Tricycle

- $u_s \in [-1, 1]$ and $u_\phi \in [-\pi/2, \pi/2]$. Can it rotate in place?

Standard simple car

- $u_s \in [-1, 1]$ and $u_\phi \in (-\phi_{\max}, \phi_{\max})$ for some $\phi_{\max} < \pi/2$.
  What happens at $\pi/2$ and $-\pi/2$?

Reeds-Shepp car

- Variation: $u_s \in \{-1, 0, 1\}$ (i.e., "reverse", "park", "forward")

Variations of the Simple Car Model

- Input controls: u_s (speed) and u_ϕ (steering angle)
- CTE: $\dot{x} = u_s \cos\theta$, $\dot{y} = u_s \sin\theta$, $\dot{\theta} = \frac{u_s}{L} \tan u_\phi$
  What are bounds on steering angle and speed? Why bound speed?

Tricycle

- $u_s \in [-1, 1]$ and $u_\phi \in [-\pi/2, \pi/2]$. Can it rotate in place?

Standard simple car

- $u_s \in [-1, 1]$ and $u_\phi \in (-\phi_{\text{max}}, \phi_{\text{max}})$ for some $\phi_{\text{max}} < \pi/2$.
  What happens at $\pi/2$ and $-\pi/2$?

Reeds-Shepp car

- Variation: $u_s \in \{-1, 0, 1\}$ (i.e., "reverse", "park", "forward")

Dubins car

- Variation: $u_s \in \{0, 1\}$ (i.e., "park", "forward")

Fundamental question:

- Is any configuration reachable from a given configuration in wheeled systems?

Reachability Issues under Kinematic Constraints

Fundamental question:

- Is any configuration reachable from a given configuration in wheeled systems?
- Most often, no. A car cannot move sideways. Specifying an arbitrary goal configuration does not mean that one can always find instataneous controls/one set of velocities.

Reachability Issues under Kinematic Constraints

Fundamental question:

- Is any configuration reachable from a given configuration in wheeled systems?
- Most often, no. A car cannot move sideways. Specifying an arbitrary goal configuration does not mean that one can always find instataneous controls/one set of velocities.
- We say a car is not instantaneously controllable (non-holonomic) but series of maneuvers may exist (small time locally controllable).

Reachability Issues under Kinematic Constraints

Fundamental question:

- Is any configuration reachable from a given configuration in wheeled systems?
- Most often, no. A car cannot move sideways. Specifying an arbitrary goal configuration does not mean that one can always find instataneous controls/one set of velocities.
- We say a car is not instantaneously controllable (non-holonomic) but series of maneuvers may exist (small time locally controllable).
- We have issue of constrained mobility (no instantaneous controls).

Inverse Kinematics for Simple Car

Problem formulation:

- Standing at configuration $q_{\text{start}} = (x_{\text{start}}, y_{\text{start}}, \theta_{\text{start}})$ at time t find path that places robot at $q_{\text{goal}} = (x_{\text{goal}}, y_{\text{goal}}, \theta_{\text{goal}})$ at time $t + \delta t$.

Inverse Kinematics for Simple Car

Problem formulation:

- Standing at configuration $q_{\text{start}} = (x_{\text{start}}, y_{\text{start}}, \theta_{\text{start}})$ at time t find path that places robot at $q_{\text{goal}} = (x_{\text{goal}}, y_{\text{goal}}, \theta_{\text{goal}})$ at time $t + \delta t$.
- How can we increase constrained mobility in non-holonomic systems? Allow a sequence of maneuvers/different velocities. Look for a path rather than a single edge.

Inverse Kinematics for Simple Car

Problem formulation:

- Standing at configuration $q_{\text{start}} = (x_{\text{start}}, y_{\text{start}}, \theta_{\text{start}})$ at time t find path that places robot at $q_{\text{goal}} = (x_{\text{goal}}, y_{\text{goal}}, \theta_{\text{goal}})$ at time $t + \delta t$.
- How can we increase constrained mobility in non-holonomic systems? Allow a sequence of maneuvers/different velocities. Look for a path rather than a single edge.
- If path exists, we say system is small-time locally controllable (maneuvers exist).

Inverse Kinematics for Simple Car

Problem formulation:

- Standing at configuration $q_{\text{start}} = (x_{\text{start}}, y_{\text{start}}, \theta_{\text{start}})$ at time t find path that places robot at $q_{\text{goal}} = (x_{\text{goal}}, y_{\text{goal}}, \theta_{\text{goal}})$ at time $t + \delta t$.
- How can we increase constrained mobility in non-holonomic systems? Allow a sequence of maneuvers/different velocities. Look for a path rather than a single edge.
- If path exists, we say system is small-time locally controllable (maneuvers exist).

New problem formulation: Find series of controls to get to goal.

- Motion planning with kinematic constraints to find *feasible* series of maneuvers.

Constrained Mobility in Non-holonomic Systems

- Simple Car is under-actuated: only 2 controls, but C-space has 3 dimensions.

A robot is non-holonomic if its motion is constrained by a non-integrable equation of the form $f(q, \dot{q}) = 0$.

- Simple Car is non-holonomic because $-\dot{x}sin\theta + \dot{y}cos\theta = 0$.
- Reeds-Shepp car can be maneuvered into an arbitrarily small parking space, provided that a small amount of clearance exists.
  Property called small-time locally controllable (STLC).
- Dubins car is non-holonomic but not small-time controllable.

Constrained Mobility in Non-holonomic Systems

- Simple Car is under-actuated: only 2 controls, but C-space has 3 dimensions.

A robot is non-holonomic if its motion is constrained by a non-integrable equation of the form $f(q, \dot{q}) = 0$.

- Simple Car is non-holonomic because $-\dot{x}sin\theta + \dot{y}cos\theta = 0$.
- Reeds-Shepp car can be maneuvered into an arbitrarily small parking space, provided that a small amount of clearance exists.

  Property called small-time locally controllable (STLC).
- Dubins car is non-holonomic but not small-time controllable. Why?

Constrained Mobility in Non-holonomic Systems

- Simple Car is under-actuated: only 2 controls, but C-space has 3 dimensions.

A robot is non-holonomic if its motion is constrained by a non-integrable equation of the form $f(q, \dot{q}) = 0$.

- Simple Car is non-holonomic because $-\dot{x} sin\theta + \dot{y} cos\theta = 0$.
- Reeds-Shepp car can be maneuvered into an arbitrarily small parking space, provided that a small amount of clearance exists.
  Property called small-time locally controllable (STLC).
- Dubins car is non-holonomic but not small-time controllable. Why?
  Try to parallel park with no reverse gear!

Constrained Mobility in Non-holonomic Systems

- Simple Car is under-actuated: only 2 controls, but C-space has 3 dimensions.

A robot is non-holonomic if its motion is constrained by a non-integrable equation of the form $f(q, \dot{q}) = 0$.

- Simple Car is non-holonomic because $-\dot{x}sin\theta + \dot{y}cos\theta = 0$.
- Reeds-Shepp car can be maneuvered into an arbitrarily small parking space, provided that a small amount of clearance exists.

  Property called small-time locally controllable (STLC).
- Dubins car is non-holonomic but not small-time controllable. Why?

  Try to parallel park with no reverse gear!

  Can do it in an infinitely-large parking lot with no obstacles.

Kinematics for Other Wheeled Systems

Other non-holonomic wheeled systems:

- Differential drive
- Unicycle
- Tractor trailer

Objective 1: Derive CTE for each of them

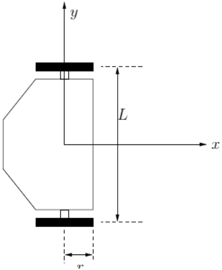
Objective 2: Move on to dynamic constraints

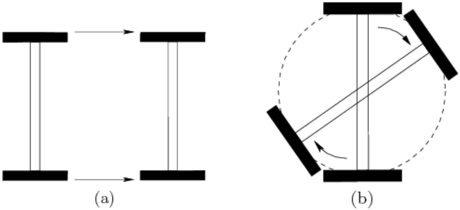
Put it all together for sampling-based motion planning.

Kinematics for Wheeled Systems – Differential Drive

Differential drives

- Most indoor robots are modeled after ddrives.
- Two main wheels, each attached to its own motor.
- Third invisible (caster) wheel in rear to passively roll and prevent falling over.
- Wheels move at same or different angular velocity.
- As a result, ddrive moves ahead or on circle.





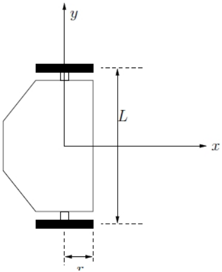
Body frame:

- Origin at center of axle
- x-axis perpendicular to axle
- L: distance between wheels.
- r: wheel radius

(a) Pure translation when both wheels move at same angular velocity

(b) pure rotation when wheels move at opposite velocities.

That is why origin placed at center of axle, so ddrive rotates in place in (b).

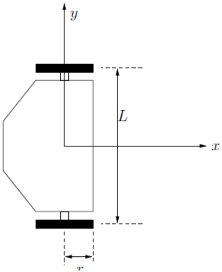


Input controls $v = (v_\ell, v_r)$

- v_ℓ: angular velocity of left wheel
- v_r: angular velocity of right wheel

Body frame:

- Origin at center of axle
- x-axis perpendicular to axle
- L: distance between wheels.
- r: wheel radius



Input controls $v = (v_\ell, v_r)$

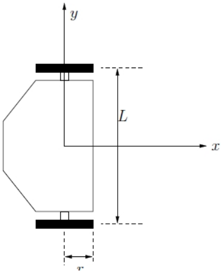
- v_ℓ: angular velocity of left wheel
- v_r: angular velocity of right wheel

How does the ddrive move?

- $v_\ell = v_r \Rightarrow$ moves in direction wheels are pointing

Body frame:

- Origin at center of axle
- x-axis perpendicular to axle
- L: distance between wheels.
- r: wheel radius



Body frame:

- Origin at center of axle
- x-axis perpendicular to axle
- L: distance between wheels.
- r: wheel radius

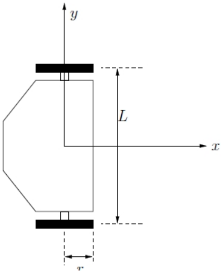
Input controls $v = (v_\ell, v_r)$

- v_ℓ: angular velocity of left wheel
- v_r: angular velocity of right wheel

How does the ddrive move?

- $v_\ell = v_r \Rightarrow$ moves in direction wheels are pointing
- $v_\ell = -v_r \Rightarrow$ rotates in place

Kinematics for Wheeled Systems – Differential Drive



Body frame:

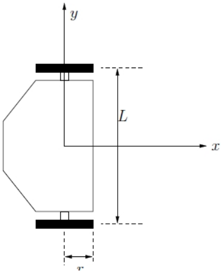
- Origin at center of axle
- x-axis perpendicular to axle
- L: distance between wheels.
- r: wheel radius

Input controls $v = (v_\ell, v_r)$

- v_ℓ: angular velocity of left wheel
- v_r: angular velocity of right wheel

How does the ddrive move?

- $v_\ell = v_r \Rightarrow$ moves in direction wheels are pointing
- $v_\ell = -v_r \Rightarrow$ rotates in place
- CTE for ddrive?



Body frame:

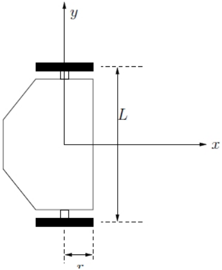
- Origin at center of axle
- x-axis perpendicular to axle
- L: distance between wheels.
- r: wheel radius

Input controls $v = (v_\ell, v_r)$

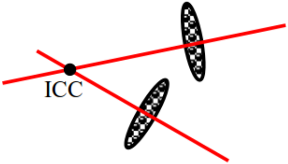
- v_ℓ: angular velocity of left wheel
- v_r: angular velocity of right wheel

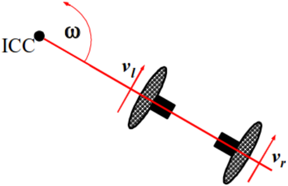
How does the ddrive move?

- $v_\ell = v_r \Rightarrow$ moves in direction wheels are pointing
- $v_\ell = -v_r \Rightarrow$ rotates in place
- CTE for ddrive?
- Variant of car, but need to introduce concept of ICC



There must be a point that lies along common left and right wheel axis, known as ICC – Instantaneous Center of Curvature.



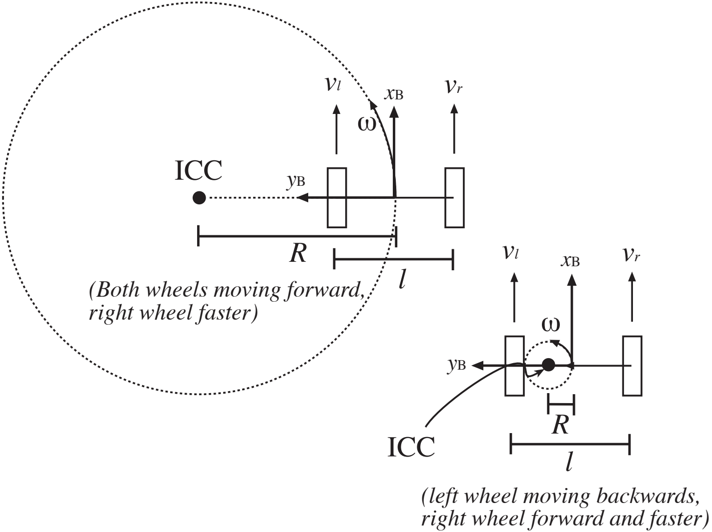


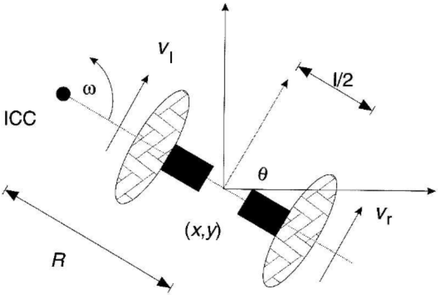
Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r)\cos\theta$
- $\dot{y} = s(v_\ell, v_r)\sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

For ddrive, ICC exists as long as $v_\ell \neq v_r$. Ddrive rotates around ICC.

ICC location on wheel axis changes as v_ℓ and v_r change.



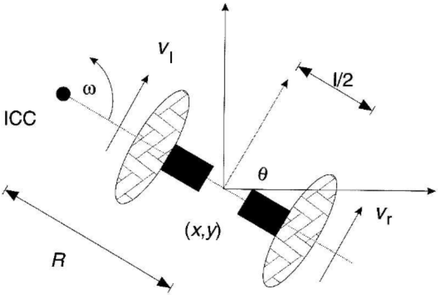


Observations to obtain CTE:

Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r) \cos\theta$
- $\dot{y} = s(v_\ell, v_r) \sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

s := translational velocity
θ := angular velocity



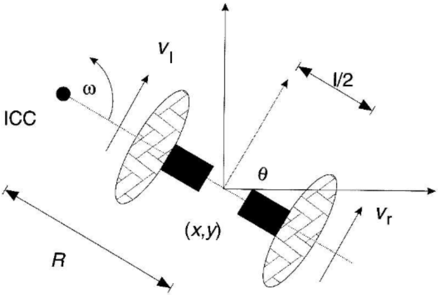
Observations to obtain CTE:

- Ddrive rotates around ICC

Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r) \cos\theta$
- $\dot{y} = s(v_\ell, v_r) \sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

$s :=$ translational velocity
$\theta :=$ angular velocity



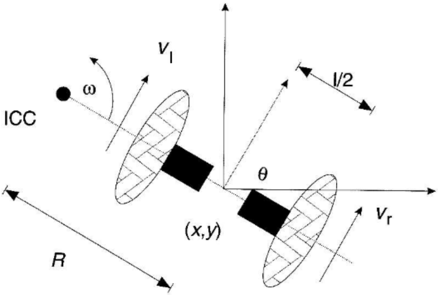
Observations to obtain CTE:

- Ddrive rotates around ICC
- Center and wheels rotate on concentric circles with radii R, $R - L/2$ (left), and $R + L/2$ (right), respectively.

Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r)\cos\theta$
- $\dot{y} = s(v_\ell, v_r)\sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

s := translational velocity
θ := angular velocity



Ddrive is variant of simple car:

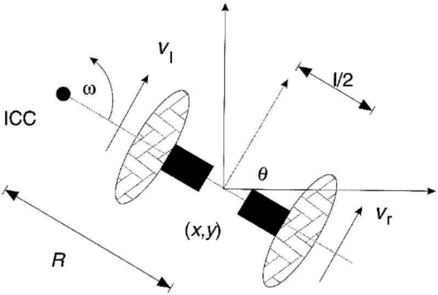
- $\dot{x} = s(v_\ell, v_r) \cos\theta$
- $\dot{y} = s(v_\ell, v_r) \sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

$s :=$ translational velocity
$\theta :=$ angular velocity

Observations to obtain CTE:

- Ddrive rotates around ICC
- Center and wheels rotate on concentric circles with radii R, $R - L/2$ (left), and $R + L/2$ (right), respectively.
- Let ω be angular vel. around ICC

Arclengths traveled per unit of time by left and right wheel:



Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r)\cos\theta$
- $\dot{y} = s(v_\ell, v_r)\sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

s := translational velocity
θ := angular velocity

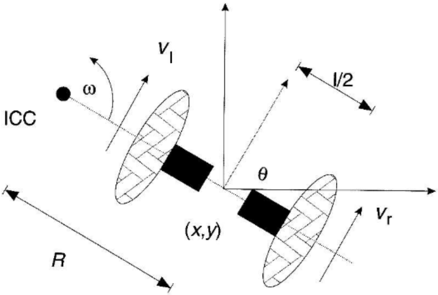
Observations to obtain CTE:

- Ddrive rotates around ICC
- Center and wheels rotate on concentric circles with radii R, $R - L/2$ (left), and $R + L/2$ (right), respectively.
- Let ω be angular vel. around ICC

Arclengths traveled per unit of time by left and right wheel:

- $w \cdot (R - L/2)$

Kinematics for Wheeled Systems – Differential Drive



Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r)\cos\theta$
- $\dot{y} = s(v_\ell, v_r)\sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

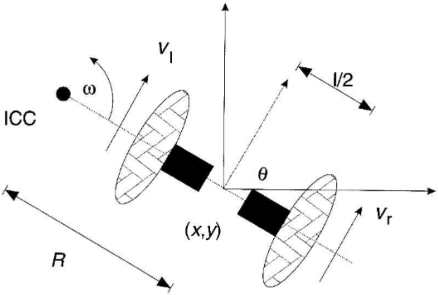
s := translational velocity
θ := angular velocity

Observations to obtain CTE:

- Ddrive rotates around ICC
- Center and wheels rotate on concentric circles with radii R, $R - L/2$ (left), and $R + L/2$ (right), respectively.
- Let ω be angular vel. around ICC

Arclengths traveled per unit of time by left and right wheel:

- $w \cdot (R - L/2) = v_\ell \cdot r$



Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r) \cos\theta$
- $\dot{y} = s(v_\ell, v_r) \sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

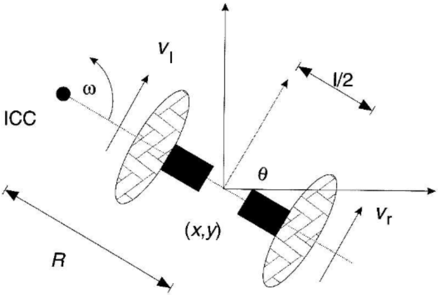
s := translational velocity
θ := angular velocity

Observations to obtain CTE:

- Ddrive rotates around ICC
- Center and wheels rotate on concentric circles with radii R, $R - L/2$ (left), and $R + L/2$ (right), respectively.
- Let ω be angular vel. around ICC

Arclengths traveled per unit of time by left and right wheel:

- $w \cdot (R - L/2) = v_\ell \cdot r$
- $w \cdot (R + L/2)$



Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r)\cos\theta$
- $\dot{y} = s(v_\ell, v_r)\sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

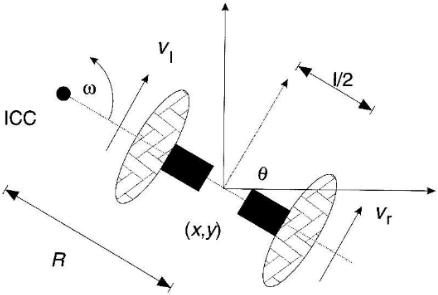
s := translational velocity
θ := angular velocity

Observations to obtain CTE:

- Ddrive rotates around ICC
- Center and wheels rotate on concentric circles with radii R, $R - L/2$ (left), and $R + L/2$ (right), respectively.
- Let ω be angular vel. around ICC

Arclengths traveled per unit of time by left and right wheel:

- $w \cdot (R - L/2) = v_\ell \cdot r$
- $w \cdot (R + L/2) = v_r \cdot r$



Arclengths traveled per unit of time by left and right wheel:

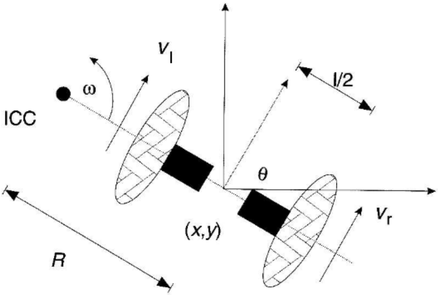
- $w \cdot (R - L/2) = v_\ell \cdot r$
- $w \cdot (R + L/2) = v_r \cdot r$

Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r) \cos\theta$
- $\dot{y} = s(v_\ell, v_r) \sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

s := translational velocity
θ := angular velocity

Kinematics for Wheeled Systems – Differential Drive



Arclengths traveled per unit of time by left and right wheel:

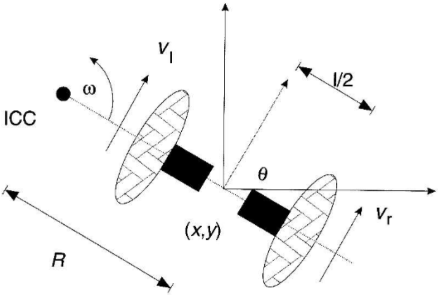
- $w \cdot (R - L/2) = v_\ell \cdot r$
- $w \cdot (R + L/2) = v_r \cdot r$

Solving for w and R, we obtain:

Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r) \cos \theta$
- $\dot{y} = s(v_\ell, v_r) \sin \theta$
- $\dot{\theta} = f(v_\ell, v_r)$

s := translational velocity
θ := angular velocity



Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r)\cos\theta$
- $\dot{y} = s(v_\ell, v_r)\sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

$s :=$ translational velocity

$\theta :=$ angular velocity

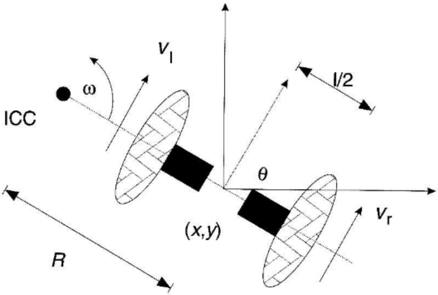
Arclengths traveled per unit of time by left and right wheel:

- $w \cdot (R - L/2) = v_\ell \cdot r$
- $w \cdot (R + L/2) = v_r \cdot r$

Solving for w and R, we obtain:

- $w = r/L \cdot (v_r - v_\ell)$

Kinematics for Wheeled Systems – Differential Drive



Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r)\cos\theta$
- $\dot{y} = s(v_\ell, v_r)\sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

$s :=$ translational velocity
$\theta :=$ angular velocity

Arclengths traveled per unit of time by left and right wheel:

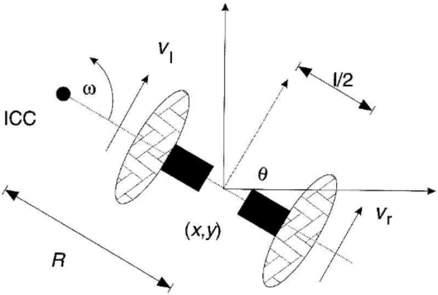
- $w \cdot (R - L/2) = v_\ell \cdot r$
- $w \cdot (R + L/2) = v_r \cdot r$

Solving for w and R, we obtain:

- $w = r/L \cdot (v_r - v_\ell)$
- $R = L/2 \cdot \frac{v_\ell + v_r}{v_r - v_\ell} r$

Question: how does ω relate to $\dot{\theta}$?

Kinematics for Wheeled Systems – Differential Drive



Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r)\cos\theta$
- $\dot{y} = s(v_\ell, v_r)\sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

$s :=$ translational velocity
$\theta :=$ angular velocity

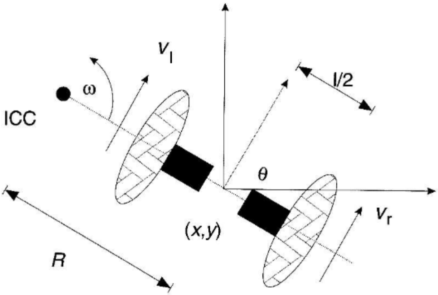
Arclengths traveled per unit of time by left and right wheel:

- $w \cdot (R - L/2) = v_\ell \cdot r$
- $w \cdot (R + L/2) = v_r \cdot r$

Solving for w and R, we obtain:

- $w = r/L \cdot (v_r - v_\ell)$
- $R = L/2 \cdot \frac{v_\ell + v_r}{v_r - v_\ell} r$

Question: how does ω relate to $\dot{\theta}$?
Answer: They are one and the same.



So:

- $\dot{\theta} = r/L \cdot (v_r - v_\ell)$
- What about s?

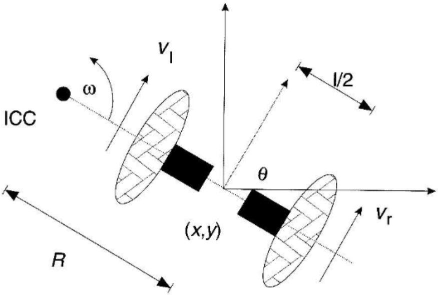
s is translational velocity

Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r) \cos \theta$
- $\dot{y} = s(v_\ell, v_r) \sin \theta$
- $\dot{\theta} = f(v_\ell, v_r)$

s := translational velocity
θ := angular velocity

Kinematics for Wheeled Systems – Differential Drive



Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r)\cos\theta$
- $\dot{y} = s(v_\ell, v_r)\sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

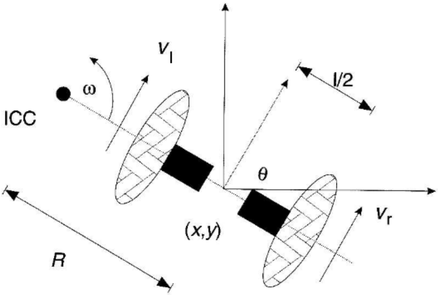
s := translational velocity
θ := angular velocity

So:

- $\dot{\theta} = r/L \cdot (v_r - v_\ell)$
- What about s?

s is translational velocity
(of center of axle)

- When $v_l = v_r$, ddrive moves forward but not at twice the speed: suggests: $s = r/2 \cdot (v_\ell + v_r)$



Ddrive is variant of simple car:

- $\dot{x} = s(v_\ell, v_r)\cos\theta$
- $\dot{y} = s(v_\ell, v_r)\sin\theta$
- $\dot{\theta} = f(v_\ell, v_r)$

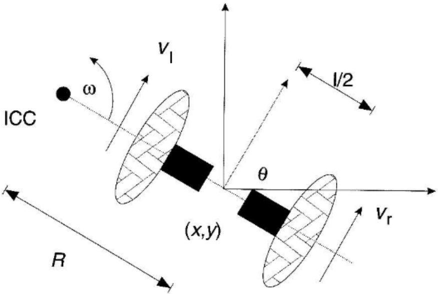
$s :=$ translational velocity
$\theta :=$ angular velocity

So:

- $\dot{\theta} = r/L \cdot (v_r - v_\ell)$
- What about s?

s is translational velocity
(of center of axle)

- When $v_l = v_r$, ddrive moves forward but not at twice the speed: suggests: $s = r/2 \cdot (v_\ell + v_r)$
- $s = R \cdot \omega \Rightarrow s = r/2 \cdot (v_\ell + v_r)$



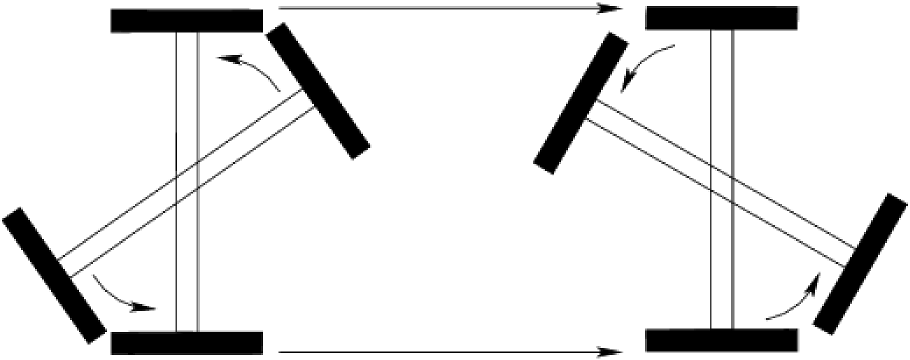
CTE (using u_ℓ and u_r for controls on wheel (angular) velocities:

- $\dot{x} = \frac{r}{2}(u_\ell + u_r)\cos\theta$
- $\dot{y} = \frac{r}{2}(u_\ell + u_r)\sin\theta$
- $\dot{\theta} = \frac{r}{L}(u_r - u_\ell)$

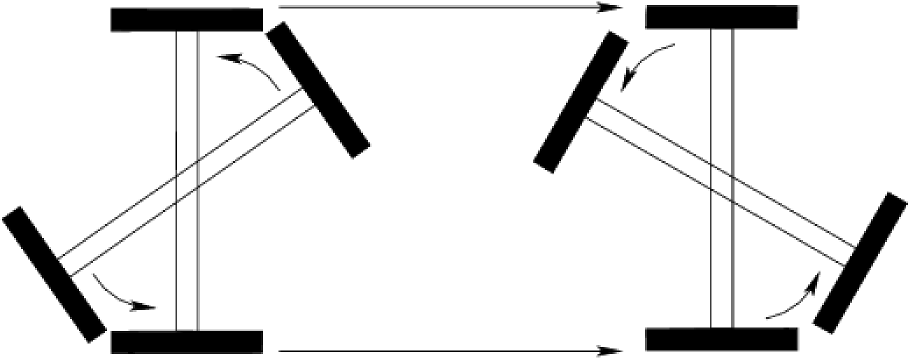
Interesting questions:

- What happens when either u_ℓ or u_r (not both) are set to 0?
- Can ddrive simulate motions of simple car?
- Is ddrive non-holonomic?
- Is it STLC?

Can ddrive move between any two configurations?

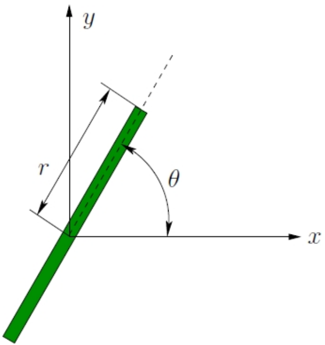


Can ddrive move between any two configurations?

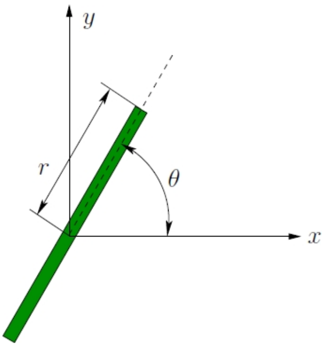


Point wheels as in destination. Translate. Rotate to desired orientation.

Kinematics for Wheeled Systems – Unicycle



- Rider can set pedaling speed and orientation of the wheel with respect to the z-axis
- r: wheel radius
- σ: pedaling angular velocity
- $s = r\sigma$: speed of unicycle
- ω: rotational velocity in the xy plane controlled directly.

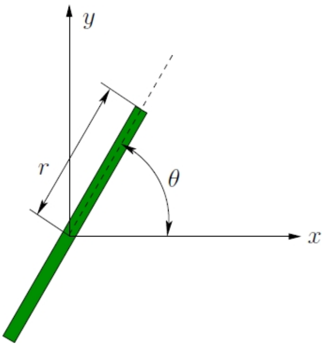


CTE:

- $\dot{x} = u_\sigma r \cos\theta$
- $\dot{y} = u_\sigma r \sin\theta$
- $\dot{\theta} = u_\omega$

- Rider can set pedaling speed and orientation of the wheel with respect to the z-axis
- r: wheel radius
- σ: pedaling angular velocity
- $s = r\sigma$: speed of unicycle
- ω: rotational velocity in the xy plane controlled directly.
- Note: ddrive with $L = 1$, $u_s = ru_\sigma$

Kinematics for Wheeled Systems – Unicycle

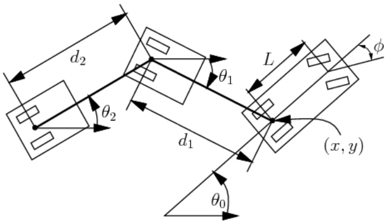


CTE:

- $\dot{x} = u_\sigma r \cos\theta$
- $\dot{y} = u_\sigma r \sin\theta$
- $\dot{\theta} = u_\omega$

- Rider can set pedaling speed and orientation of the wheel with respect to the z-axis
- r: wheel radius
- σ: pedaling angular velocity
- $s = r\sigma$: speed of unicycle
- ω: rotational velocity in the xy plane controlled directly.
- Note: ddrive with $L = 1$, $u_s = ru_\sigma$
- Ddrive can simulate a unicycle. Unicycle can simulate simple car. Unicycle == tricycle.

Tractor Trailer



CTE:

- $\dot{x} = s \cos \theta$
- $\dot{y} = s \sin \theta$
- $\dot{\theta}_0 = s/L \tan \phi$
- $\dot{\theta}_1 = s/d_1 \sin(\theta_1 - \theta_0)$
- ...
- $\dot{\theta}_i = s/d_j(\Pi_{j=1}^{i-1} \cos(\theta_{j-1} - \theta_j)) \sin(\theta_{i-1} - \theta_i)$

- Simple car pulling k trailers, each attached to rear axle of body in front of it.
- New: hitch length, d_i, distance from center of rear axle of trailer i to point at which trailer is hitched to next body.
- Car itself contributes $\mathbb{R}^2 \times S^1$ to C, and each trailer contributes an S^1. So, $|\mathcal{C}| = k + 1$.
- Configuration transition equation is hard to get right. Shown one here is adapted from Murray, Sastry, IEEE Trans Autom Control, 1993.

[movie: strailer4]

Beyond Kinematics: Dynamical Systems

- Involve acceleration $\ddot{q}$ in addition to velocity $\dot{q}$ and configuration q
- Control acceleration directly
- Implicit constraints:
$$g(\ddot{q}, \dot{q}, q) = 0$$
- Parametric constraints:
$$\ddot{q} = f(\dot{q}, q, u)$$

State Space: Reducing Degree by Increasing Dimension

Example: $y \in \mathbb{R}$ is a scalar variable and

$$\ddot{y} - 3\dot{y} + y = 0 \qquad (1)$$

Example: $y \in \mathbb{R}$ is a scalar variable and

$$\ddot{y} - 3\dot{y} + y = 0 \qquad (1)$$

Let $x = (x_1, x_2)$ denote a state vector, where $x_1 = y$ and $x_2 = \dot{y}$.

State Space: Reducing Degree by Increasing Dimension

Example: $y \in \mathbb{R}$ is a scalar variable and

$$\ddot{y} - 3\dot{y} + y = 0 \qquad (1)$$

Let $x = (x_1, x_2)$ denote a state vector, where $x_1 = y$ and $x_2 = \dot{y}$.
Then:

$$\dot{x}_2 - 3x_2 + x_1 = 0 \qquad (2)$$

Are (1) and (2) equivalent?

State Space: Reducing Degree by Increasing Dimension

Example: $y \in \mathbb{R}$ is a scalar variable and

$$\ddot{y} - 3\dot{y} + y = 0 \qquad (1)$$

Let $x = (x_1, x_2)$ denote a state vector, where $x_1 = y$ and $x_2 = \dot{y}$.

Then:

$$\dot{x}_2 - 3x_2 + x_1 = 0 \qquad (2)$$

Are (1) and (2) equivalent?

- Yes, if we also add the constraint $x_2 = \dot{x}_1$.

Example: $y \in \mathbb{R}$ is a scalar variable and

$$\ddot{y} - 3\dot{y} + y = 0 \qquad (1)$$

Let $x = (x_1, x_2)$ denote a state vector, where $x_1 = y$ and $x_2 = \dot{y}$.
Then:

$$\dot{x}_2 - 3x_2 + x_1 = 0 \qquad (2)$$

Are (1) and (2) equivalent?

- Yes, if we also add the constraint $x_2 = \dot{x}_1$.

Thus, (1) can be rewritten as two constraints

- $\dot{x}_1 = x_2$
- $\dot{x}_2 = 3x_2 - x_1$

Extending Models by Adding Integrators

Suppose equations of motions are given as:

$$\dot{x} = f(x, u)$$

Let n denote the dimension. Then:

Suppose equations of motions are given as:

$$\dot{x} = f(x, u)$$

Let n denote the dimension. Then:

1. Select an input control u_i

Extending Models by Adding Integrators

Suppose equations of motions are given as:

$$\dot{x} = f(x, u)$$

Let n denote the dimension. Then:

1. Select an input control u_i
2. Rename the input control as a new state variable $x_{n+1} = u_i$

Extending Models by Adding Integrators

Suppose equations of motions are given as:

$$\dot{x} = f(x, u)$$

Let n denote the dimension. Then:

1. Select an input control u_i
2. Rename the input control as a new state variable $x_{n+1} = u_i$
3. Define a new input control u_i' that takes the place of u_i

Extending Models by Adding Integrators

Suppose equations of motions are given as:

$$\dot{x} = f(x, u)$$

Let n denote the dimension. Then:

1. Select an input control u_i
2. Rename the input control as a new state variable $x_{n+1} = u_i$
3. Define a new input control u'_i that takes the place of u_i
4. Extend the equations of motions by one dimension by introducing $\dot{x}_{n+1} = u'_i$

Extending Models by Adding Integrators

Suppose equations of motions are given as:

$$\dot{x} = f(x, u)$$

Let n denote the dimension. Then:

1. Select an input control u_i
2. Rename the input control as a new state variable $x_{n+1} = u_i$
3. Define a new input control u_i' that takes the place of u_i
4. Extend the equations of motions by one dimension by introducing $\dot{x}_{n+1} = u_i'$

Procedure referred to as placing an integrator in front of u_i

Putting it all together: Car

Kinematic (first-order) model

Dynamics (second-order) model

Config $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_s, u_\phi)$

- Signed speed $u_s \in \mathbb{R}$
- Steering angle $u_\phi \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_s \cos\theta \\ u_s \sin\theta \\ \frac{u_s}{L} \tan u_\phi \end{bmatrix}$$

Putting it all together: Car

Kinematic (first-order) model

Config $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_s, u_\phi)$

- Signed speed $u_s \in \mathbb{R}$
- Steering angle $u_\phi \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_s \cos\theta \\ u_s \sin\theta \\ \frac{u_s}{L} \tan u_\phi \end{bmatrix}$$

Dynamics (second-order) model

State $s = (x, y, \theta, \sigma, \phi)$

- Signed speed $\sigma \in \mathbb{R}$
- Steering angle $\phi \in \mathbb{R}$

Putting it all together: Car

Kinematic (first-order) model

Config $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_s, u_\phi)$

- Signed speed $u_s \in \mathbb{R}$
- Steering angle $u_\phi \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_s \cos\theta \\ u_s \sin\theta \\ \frac{u_s}{L} \tan u_\phi \end{bmatrix}$$

Dynamics (second-order) model

State $s = (x, y, \theta, \sigma, \phi)$

- Signed speed $\sigma \in \mathbb{R}$
- Steering angle $\phi \in \mathbb{R}$

Control inputs $u = (u_1, u_2)$

- Transl. accel. $u_1 \in \mathbb{R}$
- Steering rotational vel. $u_2 \in \mathbb{R}$

Putting it all together: Car

Kinematic (first-order) model

Config $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_s, u_\phi)$

- Signed speed $u_s \in \mathbb{R}$
- Steering angle $u_\phi \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_s \cos\theta \\ u_s \sin\theta \\ \frac{u_s}{L} \tan u_\phi \end{bmatrix}$$

Dynamics (second-order) model

State $s = (x, y, \theta, \sigma, \phi)$

- Signed speed $\sigma \in \mathbb{R}$
- Steering angle $\phi \in \mathbb{R}$

Control inputs $u = (u_1, u_2)$

- Transl. accel. $u_1 \in \mathbb{R}$
- Steering rotational vel. $u_2 \in \mathbb{R}$

CTE $\dot{s} = f(s, u)$:

$$\dot{s} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\sigma} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \sigma \cos\theta \\ \sigma \sin\theta \\ \frac{\sigma}{L} \tan\phi \\ u_1 \\ u_2 \end{bmatrix}$$

[movie: SCar]

Putting it all together: Differential Drive

Kinematic (first-order) model

Config. $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_\ell, u_r)$

- Angular velocities $u_\ell, u_r \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}(u_\ell + u_r)\cos\theta \\ \frac{r}{2}(u_\ell + u_r)\sin\theta \\ \frac{r}{L}(u_r - u_\ell) \end{bmatrix}$$

Dynamics (second-order) model

Putting it all together: Differential Drive

Kinematic (first-order) model

Config. $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_\ell, u_r)$

- Angular velocities $u_\ell, u_r \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}(u_\ell + u_r)\cos\theta \\ \frac{r}{2}(u_\ell + u_r)\sin\theta \\ \frac{r}{L}(u_r - u_\ell) \end{bmatrix}$$

Dynamics (second-order) model

State $s = (x, y, \theta, v_\ell, v_r)$

- Angular velocities $v_\ell, v_r \in \mathbb{R}$

Putting it all together: Differential Drive

Kinematic (first-order) model

Config. $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_\ell, u_r)$

- Angular velocities $u_\ell, u_r \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}(u_\ell + u_r)\cos\theta \\ \frac{r}{2}(u_\ell + u_r)\sin\theta \\ \frac{r}{L}(u_r - u_\ell) \end{bmatrix}$$

Dynamics (second-order) model

State $s = (x, y, \theta, v_\ell, v_r)$

- Angular velocities $v_\ell, v_r \in \mathbb{R}$

Control inputs $u = (u_1, u_2)$

- Left wheel ang. accel. $u_1 \in \mathbb{R}$
- Right wheel ang. accel. $u_2 \in \mathbb{R}$

Putting it all together: Differential Drive

Kinematic (first-order) model

Config. $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_\ell, u_r)$

- Angular velocities $u_\ell, u_r \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}(u_\ell + u_r)\cos\theta \\ \frac{r}{2}(u_\ell + u_r)\sin\theta \\ \frac{r}{L}(u_r - u_\ell) \end{bmatrix}$$

Dynamics (second-order) model

State $s = (x, y, \theta, v_\ell, v_r)$

- Angular velocities $v_\ell, v_r \in \mathbb{R}$

Control inputs $u = (u_1, u_2)$

- Left wheel ang. accel. $u_1 \in \mathbb{R}$
- Right wheel ang. accel. $u_2 \in \mathbb{R}$

CTE $\dot{s} = f(s, u)$:

$$\dot{s} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_\ell \\ \dot{v}_r \end{bmatrix} = \begin{bmatrix} \frac{r}{2}(v_\ell + v_r)\cos\theta \\ \frac{r}{2}(v_\ell + v_r)\sin\theta \\ \frac{r}{L}(v_r - v_\ell) \\ u_1 \\ u_2 \end{bmatrix}$$

[movie: SDDrive]

Kinematic (first-order) model

Config. $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_\sigma, u_\omega)$

- Translational velocity $u_\sigma \in \mathbb{R}$
- Rotational velocity $u_\omega \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_\sigma r \cos\theta \\ u_\sigma r \sin\theta \\ u_\omega \end{bmatrix}$$

Dynamics (second-order) model

Putting it all together: Unicycle

Kinematic (first-order) model

Config. $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_\sigma, u_\omega)$

- Translational velocity $u_\sigma \in \mathbb{R}$
- Rotational velocity $u_\omega \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_\sigma r \cos\theta \\ u_\sigma r \sin\theta \\ u_\omega \end{bmatrix}$$

Dynamics (second-order) model

State $s = (x, y, \theta, \sigma, \omega)$

- Translational velocity $\sigma \in \mathbb{R}$
- Rotational velocity $\omega \in \mathbb{R}$

Putting it all together: Unicycle

Kinematic (first-order) model

Config. $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_\sigma, u_\omega)$

- Translational velocity $u_\sigma \in \mathbb{R}$
- Rotational velocity $u_\omega \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_\sigma r \cos\theta \\ u_\sigma r \sin\theta \\ u_\omega \end{bmatrix}$$

Dynamics (second-order) model

State $s = (x, y, \theta, \sigma, \omega)$

- Translational velocity $\sigma \in \mathbb{R}$
- Rotational velocity $\omega \in \mathbb{R}$

Control inputs $u = (u_1, u_2)$

- Translat. accel. $u_1 \in \mathbb{R}$
- Rotational accel. $u_2 \in \mathbb{R}$

Putting it all together: Unicycle

Kinematic (first-order) model

Config. $q = (x, y, \theta)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$

Control inputs $u = (u_\sigma, u_\omega)$

- Translational velocity $u_\sigma \in \mathbb{R}$
- Rotational velocity $u_\omega \in \mathbb{R}$

CTE $\dot{q} = f(q, u)$:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_\sigma r \cos\theta \\ u_\sigma r \sin\theta \\ u_\omega \end{bmatrix}$$

Dynamics (second-order) model

State $s = (x, y, \theta, \sigma, \omega)$

- Translational velocity $\sigma \in \mathbb{R}$
- Rotational velocity $\omega \in \mathbb{R}$

Control inputs $u = (u_1, u_2)$

- Translat. accel. $u_1 \in \mathbb{R}$
- Rotational accel. $u_2 \in \mathbb{R}$

CTE $\dot{s} = f(s, u)$:

$$\dot{s} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\sigma} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \sigma r \cos\theta \\ \sigma r \sin\theta \\ \omega \\ u_1 \\ u_2 \end{bmatrix}$$

Robot motions obtained by applying input controls and integrating equations of motions

Consider

- a starting state s_0
- an input control u
- motion equations $\dot{s} = f(s, u)$

Let $s(t)$ denote the state at time t. Then,

$$s(t) = s_0 + \int_{h=0}^{h=t} f(s(h), u)dh$$

Robot motions obtained by applying input controls and integrating equations of motions

Consider

- a starting state s_0
- an input control u
- motion equations $\dot{s} = f(s, u)$

Let $s(t)$ denote the state at time t. Then,

$$s(t) = s_0 + \int_{h=0}^{h=t} f(s(h), u)dh$$

Computation can be carried out by

- Closed-form integration when available or
- Numerical integration

Numerical Integration – Euler Method

Let Δt denote a small time step. We would like to compute $s(\Delta t)$ as

$$s(\Delta t) = s(0) + \int_{h=0}^{h=\Delta t} f(s(h), u)dh$$

Euler Approximation

$$f(s(t), u) = \dot{s}(t) = \frac{ds(t)}{dt} \approx \frac{s(\Delta t) - s(0)}{\Delta t}$$

Therefore,

$$s(\Delta t) \approx s(0) + \Delta t\, f(s(t), u)$$

For example, Euler integration of the kinematic model of unicycle yields:

$$s(\Delta t) \approx \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix} + \Delta t \begin{bmatrix} u_\sigma r \cos\theta \\ u_\sigma r \sin\theta \\ u_\omega \end{bmatrix}$$

- Advantage: Simple and efficient
- Disadvantage: Not very accurate (first-order approximation)

Let Δt denote a small time step. We would like to compute $s(\Delta t)$ as

$$s(\Delta t) = s(0) + \int_{h=0}^{h=\Delta t} f(s(h), u) dh$$

Fourth-order Runge-Kutta integration:

$$s(\Delta t) \approx s(0) + \frac{\Delta t}{6}(w_1 + w_2 + w_3 + w_4)$$

where

$$w_1 = f(s(0), u)$$

$$w_2 = f(s(0) + \frac{\Delta t}{2} w_1, u)$$

$$w_3 = f(s(0) + \frac{\Delta t}{2} w_2, u)$$

$$w_4 = f(s(0) + \Delta t\, w_3, u)$$

Motion-Planning Problem for Systems with Kinodynamics

Given

- State space S
- Control space U
- Equations of motions as differential equations $f : S \times U \rightarrow \dot{S}$
- State-validity function $\textsc{valid} : S \rightarrow \{\texttt{true}, \texttt{false}\}$ for collisions
- Goal function $\textsc{goal} : S \rightarrow \{\texttt{true}, \texttt{false}\}$
- Initial state s_0

Compute a control trajectory $u : [0, T] \rightarrow U$ so resulting state trajectory $s : [0, T] \rightarrow S$ obtained by integration is valid and reaches the goal, i.e.,

$$s(t) = s_0 + \int_{h=0}^{h=t} f(s(t), u(t))dh \qquad (1)$$

$$\forall t \in [0, T] : \textsc{valid}(s(t)) = \texttt{true} \qquad (2)$$

$$\exists t \in [0, T] : \textsc{goal}(s(t)) = \texttt{true} \qquad (3)$$

Decoupled approach

1. Compute a geometric solution path ignoring differential constraints
2. Transform the geometric path into a trajectory that satisfies the differential constraints

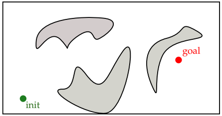
Sampling-based Motion Planning

- Take the differential constraints into account during motion planning

Roadmap Approaches

0. Initialization

add s_{init} and s_{goal} to roadmap vertex set V



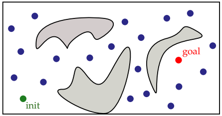
1. Sampling

repeat several times

$s \leftarrow$ StateSample()

if IsStateValid(s) = true

add s to roadmap vertex set V



Roadmap Approaches

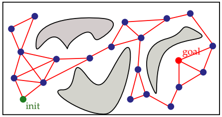
2. Connect Samples

for each pair of neighboring samples
$(s_a, s_b) \in V \times V$

$\lambda \leftarrow \textsc{GenerateLocalTrajectory}(s_a, s_b)$

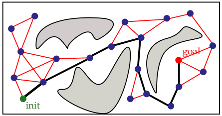
if $\textsc{IsTrajectoryValid}(\lambda) = \texttt{true}$

add (s_a, s_b) to roadmap edge set E



3. Graph Search

search graph (V, E) for path from s_{init} to s_{goal}



$s \leftarrow \textsc{StateSample}()$

- generate random values for all the state components

$\textsc{IsStateValid}(s)$

- place the robot in the configuration specified by the position and orientation components of the state
- check if the robot collides with the obstacles
- check if velocity and other state components are within bounds

$\textsc{IsTrajectoryValid}(\lambda)$

- use subdivision or incremental approach to check intermediate states

Implementation of Roadmap Approaches

$\lambda \leftarrow \textsc{GenerateLocalTrajectory}(s_a, s_b)$

- linear interpolation between s_a and s_b won't work as it does not respect underlying differential constraints
- need to find control function $u : [0, T] \rightarrow U$ such that trajectory obtained by applying u to s_a for T time units ends at s_b
- known as two-point boundary value problem: cannot always be solved analytically, and numerical solutions increase computational cost

Tree Approaches with Differential Constraints

RRT

$\mathcal{T} \leftarrow$ create tree rooted at s_{init}
while solution not found **do**
▷ *select state from tree*
    $s_{\mathrm{rand}} \leftarrow \textsc{StateSample}()$
    $s_{\mathrm{near}} \leftarrow$ nearest configuration in $\mathcal{T}$ to q_{rand} according to distance ρ
▷ *add new branch to tree from selected configuration*
    $\lambda \leftarrow \textsc{GenerateLocalTrajectory}(s_{\mathrm{near}}, s_{\mathrm{rand}})$
    if $\textsc{IsSubTrajectoryValid}(\lambda, 0, \texttt{step})$ **then**
        $s_{\mathrm{new}} \leftarrow \lambda(\texttt{step})$
        add configuration s_{new} and edge $(s_{\mathrm{near}}, s_{\mathrm{new}})$ to $\mathcal{T}$
▷ *check if a solution is found*
        if $\rho(s_{\mathrm{new}}, s_{\mathrm{goal}}) \approx 0$ **then**
            return solution trajectory from root to s_{new}

✓ StateSample(): random values for state components
✓ $\rho : S \times S \rightarrow \mathbb{R}^{\geq 0}$: distance metric between states
✓ IsSubTrajectoryValid($\lambda, 0,$ step): incremental approach

$\lambda \leftarrow$ GenerateLocalTrajectory($s_{\mathrm{near}}, s_{\mathrm{rand}}$)

- will it not create the same two-boundary value problems as in PRM?
- is it necessary to connect to s_{rand}?
- would it suffice to just come close to s_{rand}?

Avoiding Two-Boundary Value Problem

Rather than computing a trajectory from s_{near} to s_{rand} compute a trajectory that starts at s_{near} and extends toward s_{rand}

Approach 1 – extend according to random control

- Sample random control u in U
- Integrate equations of motions when applying u to s_{near} for Δt units of time, i.e.,

$$\lambda \to s(t) = s_{\mathrm{near}} + \int_{h=0}^{h=\Delta t} f(s(t), u)dh$$

Approach 2 – find the best-out-of-many random controls

1. for $i = 1, \ldots, m$ do
   1. $u_i \leftarrow$ sample random control in U
   2. $\lambda_i \to s(t) = s_{\mathrm{near}} + \int_{h=0}^{h=\Delta t} f(s(t), u_i)dh$
   3. $d_i \leftarrow \rho(s_{\mathrm{rand}}, \lambda_i(\Delta t))$
2. return λ_i with minimum d_i

[movie: Traj]

Tree approaches require only the ability to simulate robot motions

- Physics engines can be used to simulate robot motions
- Physics engines provide greater simulation accuracy
- Physics engines can take into account friction, gravity, and interactions of the robot with objects in the evironment

BULLET PHYSICS LIBRARY

OPEN DYNAMICS ENGINE™

[movie: PhysicsTricycle]

[movie: PhysicsBug]