Introduction to Robotics

Motion Planning with Kinematics and Dynamics

Erion Plaku

Department of Electrical Engineering and Computer Science
Catholic University of America

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

[movie: geometric]

[movie: kinematic]

[movie: dynamics]

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 $=, <, >, \leq, \geq$

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: $\frac{dq}{dt} = \dot{q} = (\dot{x}, \dot{y})$

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

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

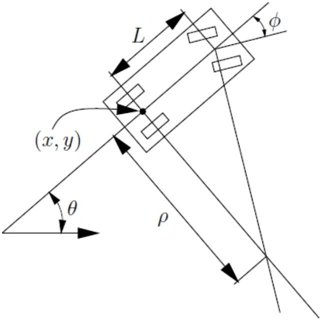
Parametric Velocity Constraints

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
- u is an input control

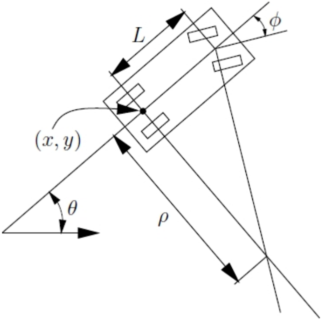


- Car configuration: $q = (x, y, \theta) \in \mathbb{R} \times S^1$
- Body frame
  - Origin is at the center of rear axle
  - x-axis points along main axis of the car
- Velocity (signed speed): s
- Steering angle: ϕ

How does the car move?

Express car motions as a set of differential equations

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



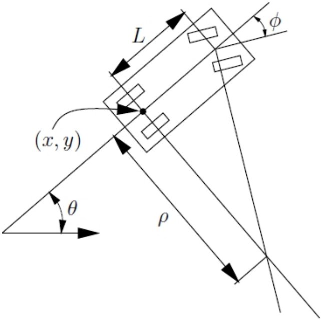
- Car configuration: $q = (x, y, \theta) \in \mathbb{R} \times S^1$
- Body frame
  - Origin is at the center of rear axle
  - x-axis points along main axis of the car
- Velocity (signed speed): s
- Steering angle: ϕ

How does the car move?

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

Express car motions as a set of differential equations

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



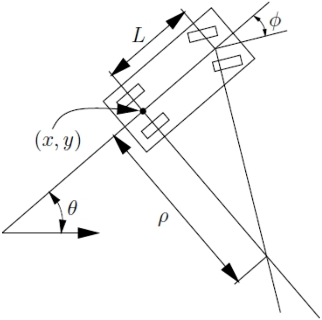
- Car configuration: $q = (x, y, \theta) \in \mathbb{R} \times S^1$
- Body frame
  - Origin is at the center of rear axle
  - x-axis points along main axis of the car
- Velocity (signed speed): s
- Steering angle: ϕ

How does the car move?

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

Express car motions as a set of differential equations

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



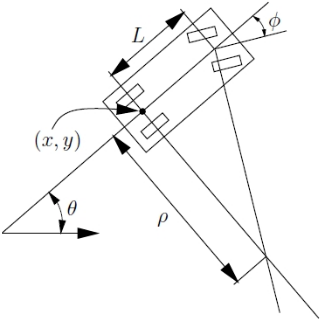
- Car configuration: $q = (x, y, \theta) \in \mathbb{R} \times S^1$
- Body frame
  - Origin is at the center of rear axle
  - x-axis points along main axis of the car
- Velocity (signed speed): s
- Steering angle: ϕ

How does the car move?

- In a small time interval Δt, the car must move approximately in the direction that the 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 $\dot{y} = s\sin\theta$

Express car motions as a set of differential equations

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



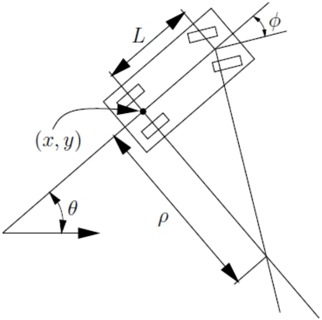
- Car configuration: $q = (x, y, \theta) \in \mathbb{R} \times S^1$
- Body frame
  - Origin is at the center of rear axle
  - x-axis points along main axis of the car
- Velocity (signed speed): s
- Steering angle: ϕ

Express car motions as a set of differential equations

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

How does the car move?

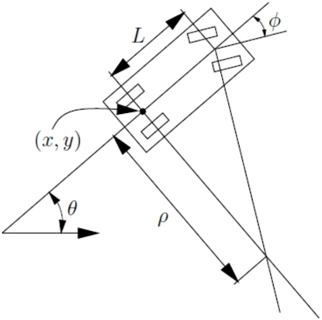
- In a small time interval Δt, the car must move approximately in the direction that the rear wheels are pointing
- In the limit, as $\Delta t \to 0$, then $\frac{dy}{dx} = \tan\theta$, i.e., $-\dot{x}\sin\theta + \dot{y}\cos\theta = 0$
- Solution is of the form $\dot{x} = s\cos\theta$ and $\dot{y} =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



- w: distance traveled by the car
- $dw = \rho d\theta$

  If the steering angle is fixed at ϕ, the car travels in circular motion, in which the radius of the circle is ρ
- $\rho = L / \tan \phi$

  where L is the distance from front to rear axles

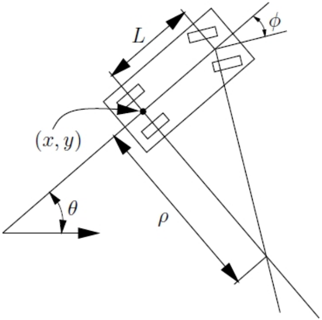


- w: distance traveled by the car
- $dw = \rho d\theta$

  If the steering angle is fixed at ϕ, the car travels in circular motion, in which the radius of the circle is ρ
- $\rho = L/\tan\phi$

  where L is the distance from front to rear axles
- Therefore

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



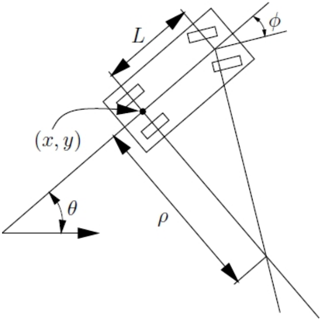
How should we control the car?

- w: distance traveled by the car
- $dw = \rho d\theta$

  If the steering angle is fixed at ϕ, the car travels in circular motion, in which the radius of the circle is ρ
- $\rho = L / \tan\phi$

  where L is the distance from front to rear axles
- Therefore

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



- w: distance traveled by the car
- $dw = \rho d\theta$

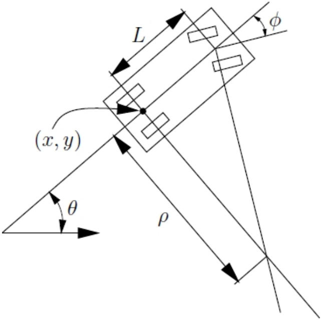
  If the steering angle is fixed at ϕ, the car travels in circular motion, in which the radius of the circle is ρ
- $\rho = L / \tan\phi$

  where L is the distance from front to rear axles
- Therefore

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

How should we control the car?

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



- w: distance traveled by the car
- $dw = \rho d\theta$

  If the steering angle is fixed at ϕ, the car travels in circular motion, in which the radius of the circle is ρ
- $\rho = L/\tan\phi$

  where L is the distance from front to rear axles
- Therefore

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

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)
- Equations of motions: $\dot{x} = u_s \cos\theta$ $\quad \dot{y} = u_s \sin\theta$ $\quad \dot{\theta} = \frac{u_s}{L}\tan u_\phi$

Variations of the Simple Car Model

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

What are the bounds on the steering angle?

What are the bounds on the speed?

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

What are the bounds on the steering angle?

What are the bounds on the speed?

Tricycle

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

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

What are the bounds on the steering angle?

What are the bounds on the speed?

Tricycle

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

Standard simple car

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

Variations of the Simple Car Model

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

What are the bounds on the steering angle?

What are the bounds on the speed?

Tricycle

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

Standard simple car

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

Reeds-Shepp car

- $u_s \in \{-1, 0, 1\}$ (i.e., "reverse", "park", "forward")
- u_ϕ same as in the standard simple car

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

What are the bounds on the steering angle?

What are the bounds on the speed?

Tricycle

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

Standard simple car

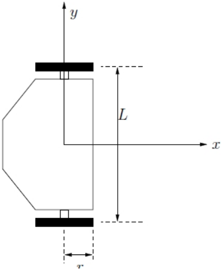
- $u_s \in [-1, 1]$
- $u_\phi \in (-\phi_{\max}, \phi_{\max})$ for some $\phi_{\max} < \pi/2$

Reeds-Shepp car

- $u_s \in \{-1, 0, 1\}$ (i.e., "reverse", "park", "forward")
- u_ϕ same as in the standard simple car

Dubins car

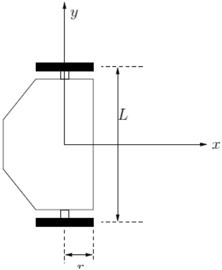
- $u_s \in \{0, 1\}$ (i.e., "park", "forward")
- u_ϕ same as in the standard simple car



Input controls $u = (u_\ell, u_r)$

- u_ℓ: angular velocity of left wheel
- u_r: angular velocity of right wheel

Kinematics for Wheeled Systems – Differential Drive

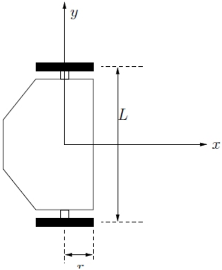


Input controls $u = (u_\ell, u_r)$

- u_ℓ: angular velocity of left wheel
- u_r: angular velocity of right wheel

How does the robot move?

Kinematics for Wheeled Systems – Differential Drive



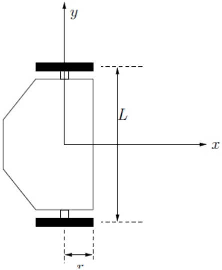
Input controls $u = (u_\ell, u_r)$

- u_ℓ: angular velocity of left wheel
- u_r: angular velocity of right wheel

How does the robot move?

- $u_\ell = u_r \Rightarrow$ moves forward in the direction the wheels are pointing

  speed proportional to wheel radius r



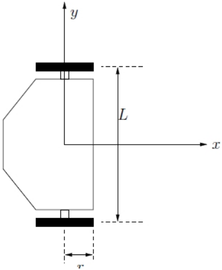
Input controls $u = (u_\ell, u_r)$

- u_ℓ: angular velocity of left wheel
- u_r: angular velocity of right wheel

How does the robot move?

- $u_\ell = u_r \Rightarrow$ moves forward in the direction the wheels are pointing

  speed proportional to wheel radius r
- $u_\ell = -u_r \Rightarrow$ rotates clockwise because wheels are turning in opposite directions



Input controls $u = (u_\ell, u_r)$

- u_ℓ: angular velocity of left wheel
- u_r: angular velocity of right wheel

How does the robot move?

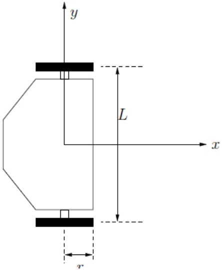
- $u_\ell = u_r \Rightarrow$ moves forward in the direction the wheels are pointing

  speed proportional to wheel radius r
- $u_\ell = -u_r \Rightarrow$ rotates clockwise because wheels are turning in opposite directions

Where is the body frame placed?

- origin at the center of the axle between the wheels

Kinematics for Wheeled Systems – Differential Drive



Equations of motions

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

Input controls $u = (u_\ell, u_r)$

- u_ℓ: angular velocity of left wheel
- u_r: angular velocity of right wheel

How does the robot move?

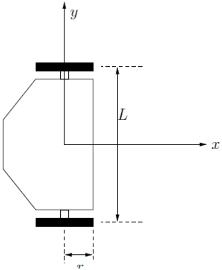
- $u_\ell = u_r \Rightarrow$ moves forward in the direction the wheels are pointing

  speed proportional to wheel radius r
- $u_\ell = -u_r \Rightarrow$ rotates clockwise because wheels are turning in opposite directions

Where is the body frame placed?

- origin at the center of the axle between the wheels

Kinematics for Wheeled Systems – Differential Drive



Equations of motions

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

Different way of representing equations of motions

- $u_\omega = (u_\ell + u_r)/2$ (rotate)
- $u_\psi = (u_r - u_\ell)$ (translate)

Input controls $u = (u_\ell, u_r)$

- u_ℓ: angular velocity of left wheel
- u_r: angular velocity of right wheel

How does the robot move?

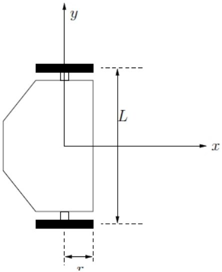
- $u_\ell = u_r \Rightarrow$ moves forward in the direction the wheels are pointing

  speed proportional to wheel radius r
- $u_\ell = -u_r \Rightarrow$ rotates clockwise because wheels are turning in opposite directions

Where is the body frame placed?

- origin at the center of the axle between the wheels

Kinematics for Wheeled Systems – Differential Drive



Equations of motions

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

Different way of representing equations of motions

- $u_\omega = (u_\ell + u_r)/2$ (rotate)
- $u_\psi = (u_r - u_\ell)$ (translate)

Input controls $u = (u_\ell, u_r)$

- u_ℓ: angular velocity of left wheel
- u_r: angular velocity of right wheel

How does the robot move?

- $u_\ell = u_r \Rightarrow$ moves forward in the direction the wheels are pointing

  speed proportional to wheel radius r
- $u_\ell = -u_r \Rightarrow$ rotates clockwise because wheels are turning in opposite directions

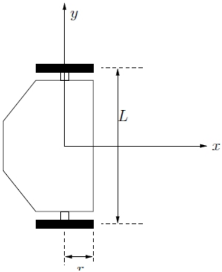
Where is the body frame placed?

- origin at the center of the axle between the wheels

Then

- $\dot{x} = r u_\omega \cos\theta$
- $\dot{y} = r u_\omega \sin\theta$
- $\dot{\theta} = r u_\psi / L$

Kinematics for Wheeled Systems – Differential Drive



Equations of motions

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

Different way of representing equations of motions

- $u_\omega = (u_\ell + u_r)/2$ (rotate)
- $u_\psi = (u_r - u_\ell)$ (translate)

Input controls $u = (u_\ell, u_r)$

- u_ℓ: angular velocity of left wheel
- u_r: angular velocity of right wheel

How does the robot move?

- $u_\ell = u_r \Rightarrow$ moves forward in the direction the wheels are pointing

  speed proportional to wheel radius r
- $u_\ell = -u_r \Rightarrow$ rotates clockwise because wheels are turning in opposite directions

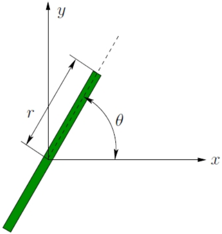
Where is the body frame placed?

- origin at the center of the axle between the wheels

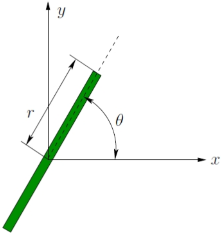
Then

- $\dot{x} = r u_\omega \cos\theta$
- $\dot{y} = r u_\omega \sin\theta$
- $\dot{\theta} = r u_\psi / L$

Can the differential drive move between any two configurations?



- rider can set the pedaling speed and the 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

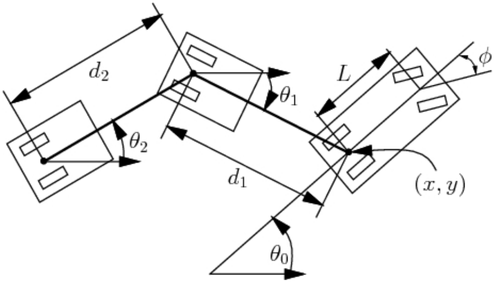


Equations of motions

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

- rider can set the pedaling speed and the 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

Tractor Trailer

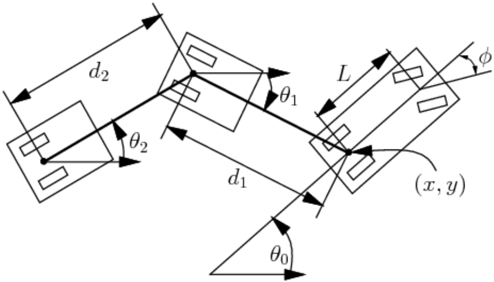


Equations of motions:

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

- Consider a simple car pulling k trailers (similar to an airport luggage cart).
- Each trailer is attached to rear axle of body in front of it.
- New parameter here is hitch length, d_i, the distance from the center of the rear axle of trailer i to the point at which the trailer is hitched to next body.
- The car itself contributes $\mathbb{R}^2 \times S^1$ to C, and each trailer contributes an S^1. So, $|\mathcal{C}| = k + 1$.
- The configuration transition equation is somewhat of an art to get right. The one here is adapted from Murray, Sastry, IEE Trans Autom Control, 1993.

[movie: strailer4]



Equations of motions:

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

- Consider a simple car pulling k trailers (similar to an airport luggage cart).
- Each trailer is attached to rear axle of body in front of it.
- New parameter here is hitch length, d_i, the distance from the center of the rear axle of trailer i to the point at which the trailer is hitched to next body.
- The car itself contributes $\mathbb{R}^2 \times S^1$ to C, and each trailer contributes an S^1. So, $|\mathcal{C}| = k + 1$.
- The configuration transition equation is somewhat of an art to get right. The one here is adapted from Murray, Sastry, IEE Trans Autom Control, 1993.

[movie: strailer4]

How about acceleration?

Dynamical Systems

- Involve acceleration $\ddot{q}$ in addition to velocity $\dot{q}$ and configuration q
- Implicit constraints

$$g(\ddot{q}, \dot{q}, q) = 0$$

- Parametric constraints

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

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

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

Phase Space: Reducing Degree by Increasing Dimension

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

$$\ddot{y} - 3\dot{y} + y = 0 \tag{1}$$

Let $x = (x_1, x_2)$ denote a phase vector, where

- $x_1 = y$
- $x_2 = \dot{y}$

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 phase vector, where

- $x_1 = y$
- $x_2 = \dot{y}$

Then

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

Are (1) and (2) equivalent?

Phase 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 phase vector, where

- $x_1 = y$
- $x_2 = \dot{y}$

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 phase vector, where

- $x_1 = y$
- $x_2 = \dot{y}$

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

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

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

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 $x_{n+1} = u_i$

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 $\dot{x}_{n+1} = u_i'$

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

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

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

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

- Translational velocity $u_s \in \mathbb{R}$
- Steering angle $u_\phi \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

Putting it all together: Car

Kinematic (first-order) model

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

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

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

- Translational velocity $u_s \in \mathbb{R}$
- Steering angle $u_\phi \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

State $s = (x, y, \theta, \mathbf{s}, \phi)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$
- Translational velocity $\mathbf{s} \in \mathbb{R}$
- Steering angle $\phi \in \mathbb{R}$

Putting it all together: Car

Kinematic (first-order) model

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

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

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

- Translational velocity $u_s \in \mathbb{R}$
- Steering angle $u_\phi \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

State $s = (x, y, \theta, \mathbf{s}, \phi)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$
- Translational velocity $\mathbf{s} \in \mathbb{R}$
- Steering angle $\phi \in \mathbb{R}$

Control inputs $u = (u_1, u_2)$

- Translational acceleration $u_1 \in \mathbb{R}$
- Steering rotational velocity $u_2 \in \mathbb{R}$

Putting it all together: Car

Kinematic (first-order) model

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

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

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

- Translational velocity $u_s \in \mathbb{R}$
- Steering angle $u_\phi \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

State $s = (x, y, \theta, \mathbf{s}, \phi)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$
- Translational velocity $\mathbf{s} \in \mathbb{R}$
- Steering angle $\phi \in \mathbb{R}$

Control inputs $u = (u_1, u_2)$

- Translational acceleration $u_1 \in \mathbb{R}$
- Steering rotational velocity $u_2 \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

Kinematic (first-order) model

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

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

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

- Translational velocity $u_s \in \mathbb{R}$
- Steering angle $u_\phi \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

State $s = (x, y, \theta, \mathbf{s}, \phi)$

- Position $(x, y) \in \mathbb{R}^2$
- Orientation $\theta \in S^1$
- Translational velocity $\mathbf{s} \in \mathbb{R}$
- Steering angle $\phi \in \mathbb{R}$

Control inputs $u = (u_1, u_2)$

- Translational acceleration $u_1 \in \mathbb{R}$
- Steering rotational velocity $u_2 \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

[movie: SCar]

Kinematic (first-order) model

Dynamics (second-order) model

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

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

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

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

Motion equations $\dot{s} = f(s, u)$, where

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

Putting it all together: Differential Drive

Kinematic (first-order) model

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

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

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

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

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

State $s = (x, y, \theta, s_\ell, s_r)$

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

Putting it all together: Differential Drive

Kinematic (first-order) model

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

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

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

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

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

State $s = (x, y, \theta, s_\ell, s_r)$

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

Control inputs $u = (u_1, u_2)$

- Angular acceleration for left wheel, $u_1 \in \mathbb{R}$
- Angular acceleration for right wheel, $u_2 \in \mathbb{R}$

Kinematic (first-order) model

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

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

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

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

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

State $s = (x, y, \theta, s_\ell, s_r)$

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

Control inputs $u = (u_1, u_2)$

- Angular acceleration for left wheel, $u_1 \in \mathbb{R}$
- Angular acceleration for right wheel, $u_2 \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

[movie: SDDrive]

Kinematic (first-order) model

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

- 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 $u_\omega \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

Putting it all together: Unicycle

Kinematic (first-order) model

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

- 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 $u_\omega \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

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

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

Putting it all together: Unicycle

Kinematic (first-order) model

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

- 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 $u_\omega \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

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

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

Control inputs $u = (u_1, u_2)$

- Translational acceleration $u_1 \in \mathbb{R}$
- Rotational acceleration $u_2 \in \mathbb{R}$

Putting it all together: Unicycle

Kinematic (first-order) model

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

- 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 $u_\omega \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

Dynamics (second-order) model

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

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

Control inputs $u = (u_1, u_2)$

- Translational acceleration $u_1 \in \mathbb{R}$
- Rotational acceleration $u_2 \in \mathbb{R}$

Motion equations $\dot{s} = f(s, u)$, where

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

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

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

$$\begin{aligned} w_1 &= f(s(0), u) \\ w_2 &= f(s(0) + \frac{\Delta t}{2} w_1, u) \\ w_3 &= f(s(0) + \frac{\Delta t}{2} w_2, u) \\ w_4 &= f(s(0) + \Delta t\, w_3, u) \end{aligned}$$

Motion-Planning Problem for Systems with Differential Constraints

Given

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

Compute a control trajectory $u : [0, T] \rightarrow U$ such that the 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

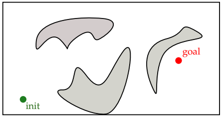
Sampling-based Motion Planning

- Take the differential constraints into account during motion planning

Roadmap Approaches

0. Initialization

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



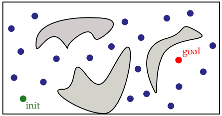
1. Sampling

repeat several times

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

if $\textsc{IsStateValid}(s) = \texttt{true}$

add s to roadmap vertex set V



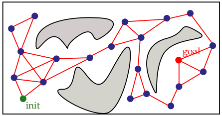
2. Connect Samples

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

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

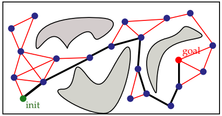
if $\textsc{IsTrajectoryValid}(\lambda) = \texttt{true}$

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 desired bounds

$\textsc{IsTrajectoryValid}(\lambda)$

- use subdivision or incremental approach to check if intermediate states are valid

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

- 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

RRT

```
1: T ← create tree rooted at s_init
2: while solution not found do
▷select state from tree
3:    s_rand ← StateSample()
4:    s_near ← nearest configuration in T to q_rand according to distance ρ
▷add new branch to tree from selected configuration
5:    λ ← GenerateLocalTrajectory(s_near, s_rand)
6:    if IsSubTrajectoryValid(λ, 0, step) then
7:       s_new ← λ(step)
8:       add configuration s_new and edge (s_near, s_new) to T
▷check if a solution is found
9:       if ρ(s_new, s_goal) ≈ 0 then
10:         return solution trajectory from root to s_new
```

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

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

- 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 $s_{\rm near}$ to $s_{\rm rand}$ compute a trajectory that starts at $s_{\rm near}$ and extends toward $s_{\rm rand}$

Approach 1 – extend according to random control

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

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

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. $\lambda_i \rightarrow s(t) = s_{\rm near} + \int_{h=0}^{h=\Delta t} f(s(t), u_i)dh$
   3. $d_i \leftarrow \rho(s_{\rm rand}, \lambda_i(\Delta t))$
2. return λ_i with minimum d_i

[movie: Traj]

Sampling-based Motion Planning with Physics-Based Simulations

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]