4-9 샘플링 기법 7:
Kinodynamic RRT

강의 요약

01

RRT-Connect

- Bidirectional Tree
- Single-query
- Rewire

- Probabilistic Completeness
- Narrow Passage

02

알고리즘

03

코드 분석

Geometric vs. Dynamic Planning

Geometric

- Kinematics와 Dynamics를 고려하지 않음
- 시간 사이의 움직임을 선형적으로 해석
- Manipulator

Geometric
vs.
Dynamics

Dynamic

- Dynamics를 고려함
- 시간 사이의 움직임을 dynamics를 고려하여 해석
- 움직임에 추가적인 제약이 있음
- 자동차, 자전거

Geometric vs. Dynamic Planning

Geometric

- Kinematics와 Dynamics를 고려하지 않음
- 시간 사이의 움직임을 선형적으로 해석
- Manipulator

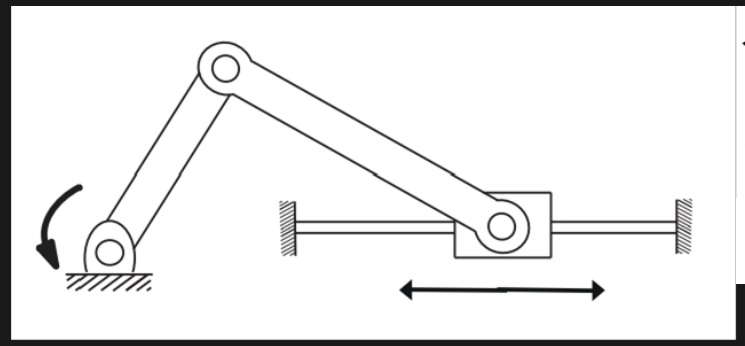
- PRM, OB-PRM, Lazy-PRM
- RRT, RRT*, RRT-Connect

Geometric
vs.
Dynamics

Dynamic

- Dynamics를 고려함
- 시간 사이의 움직임을 dynamics를 고려하여 해석
- 움직임에 추가적인 제약이 있음
- 자동차, 자전거

Dynamics 복습

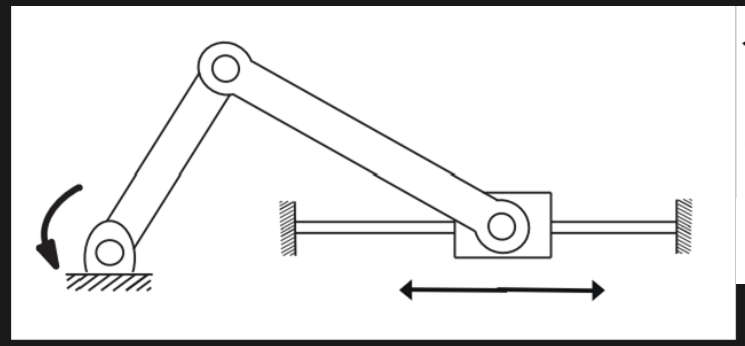


상태(state) 정의: $x = [q, \dot{q}]$

입력(input): $u = \tau$ (토크)

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

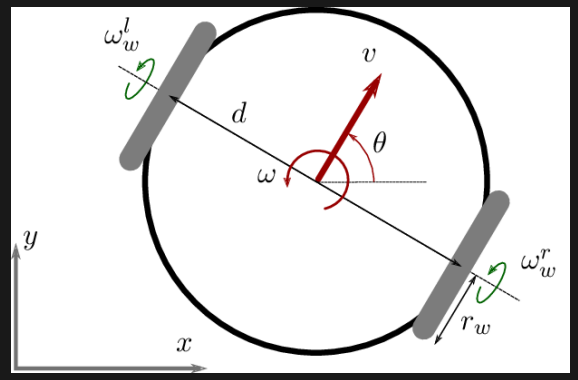
Dynamics 복습



상태(state) 정의: $x = [q, \dot{q}]$

입력(input): $u = \tau$ (토크)

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



$$x = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \omega \end{bmatrix}$$

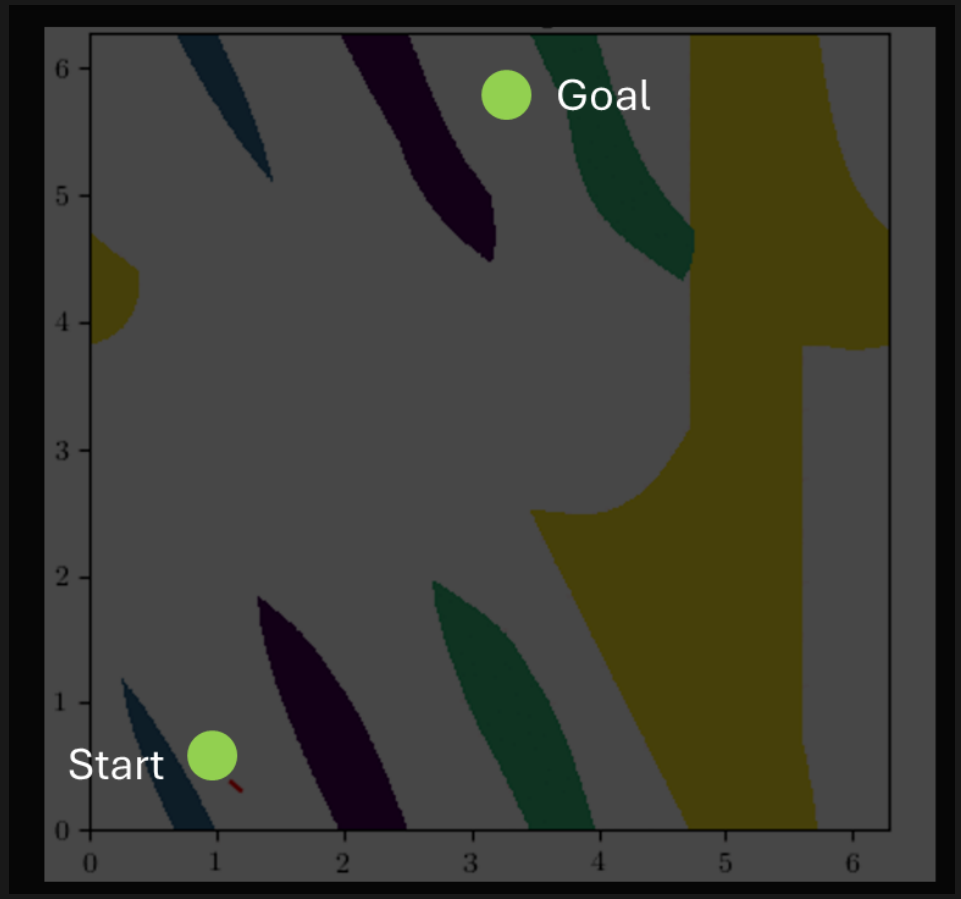
$$u = \begin{bmatrix} F_L \\ F_R \end{bmatrix}$$

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

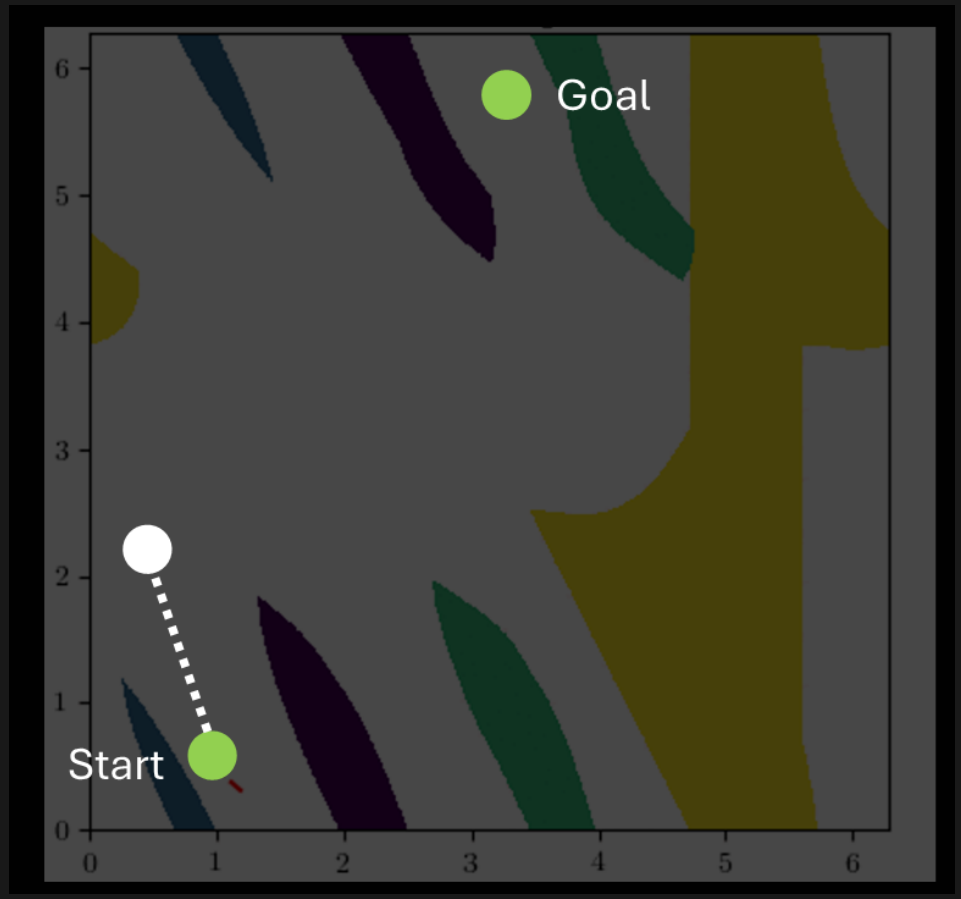
$$f(x,u) = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ \frac{1}{m}(F_L + F_R) \\ \frac{d}{2I_z}(F_R - F_L) \end{bmatrix}$$

$$\dot{x} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix}$$

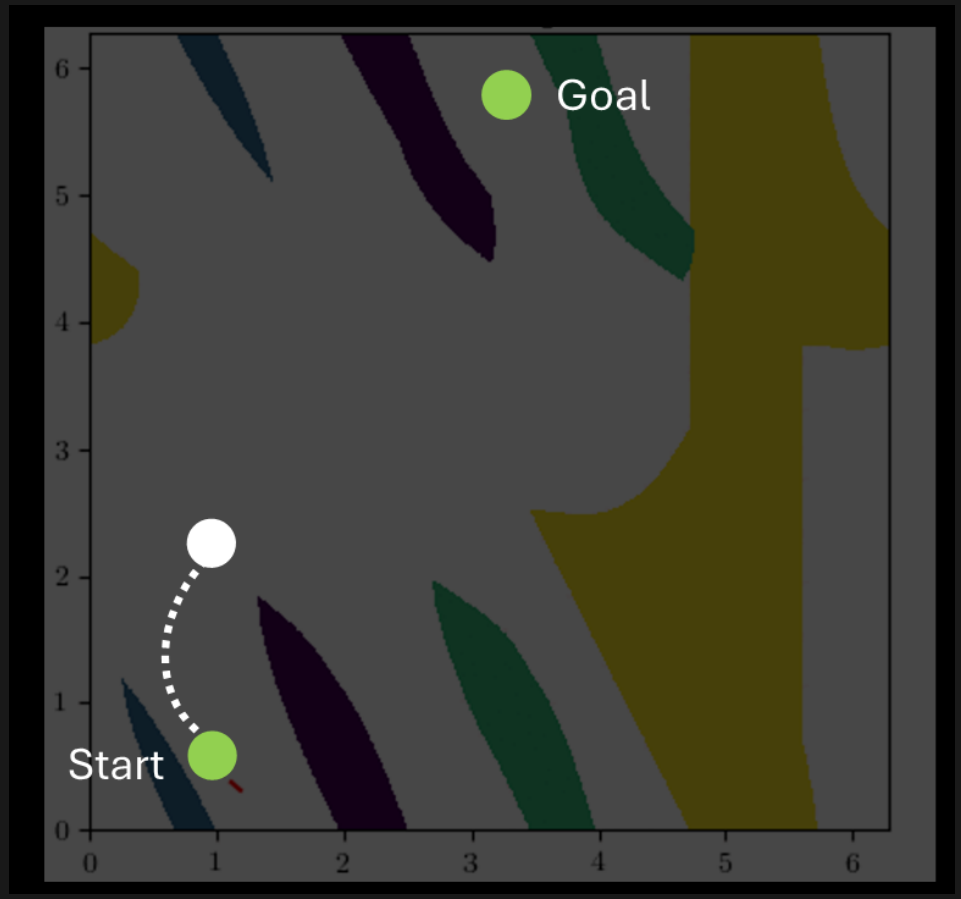
Kinodynamic RRT



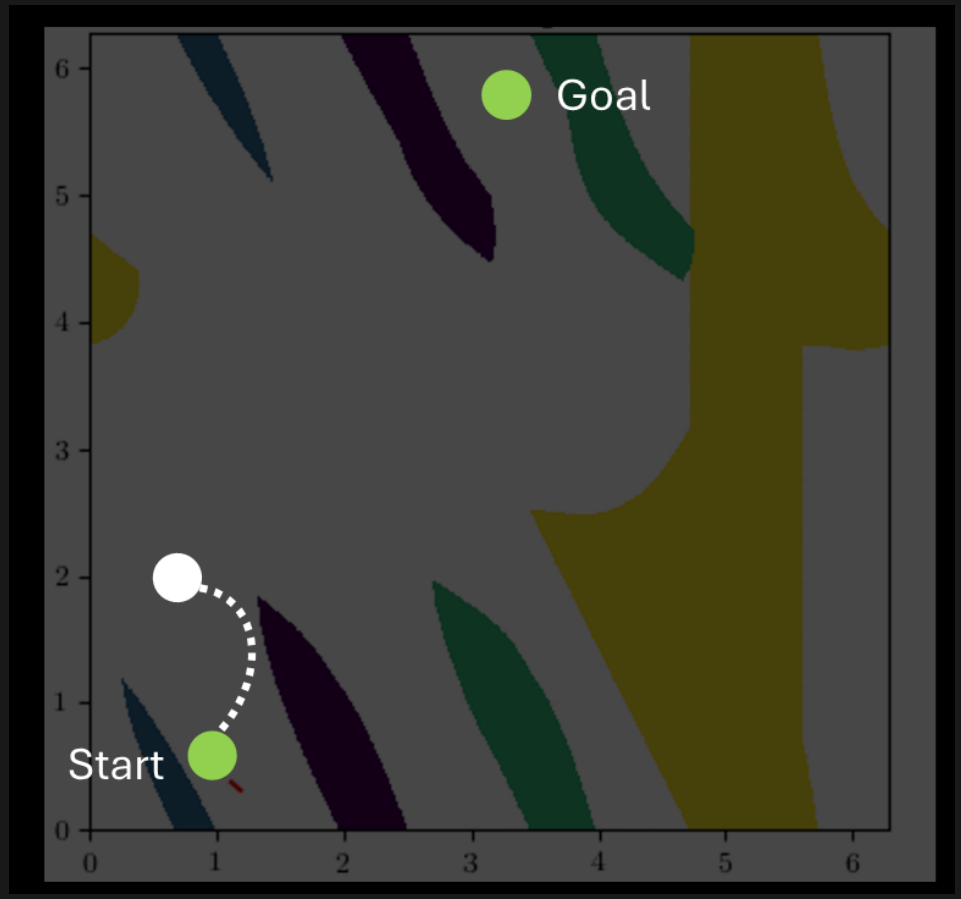
Kinodynamic RRT



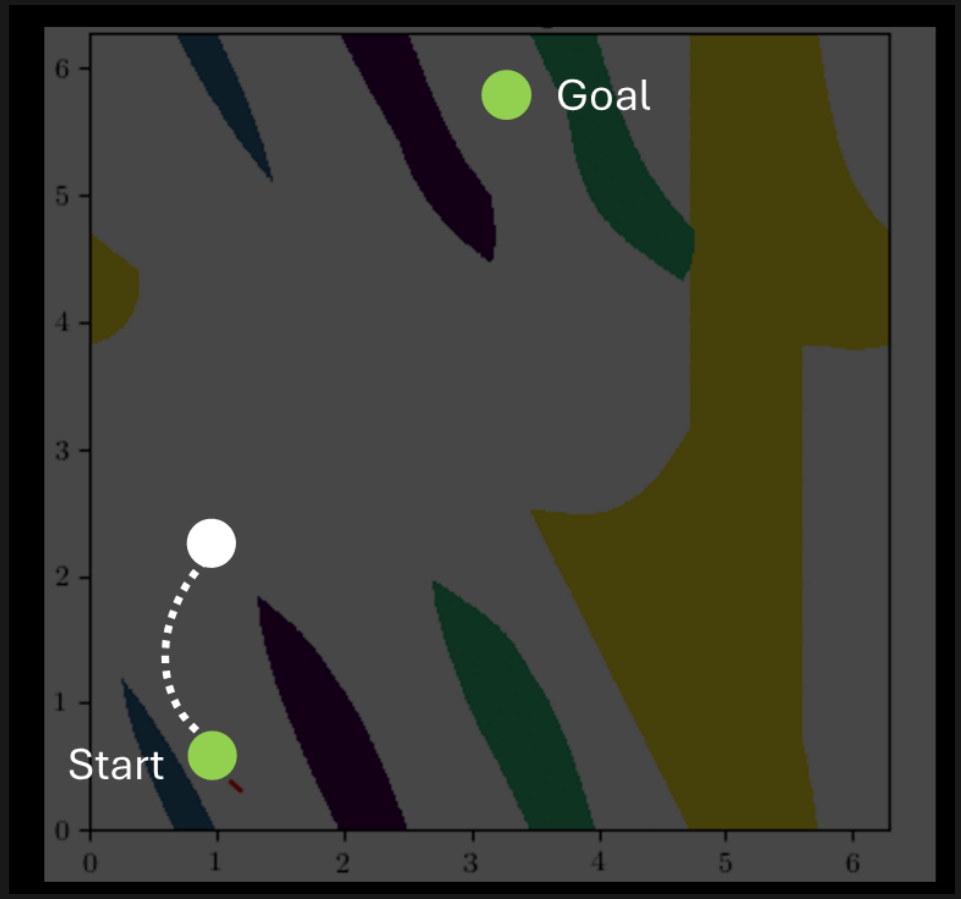
Kinodynamic RRT



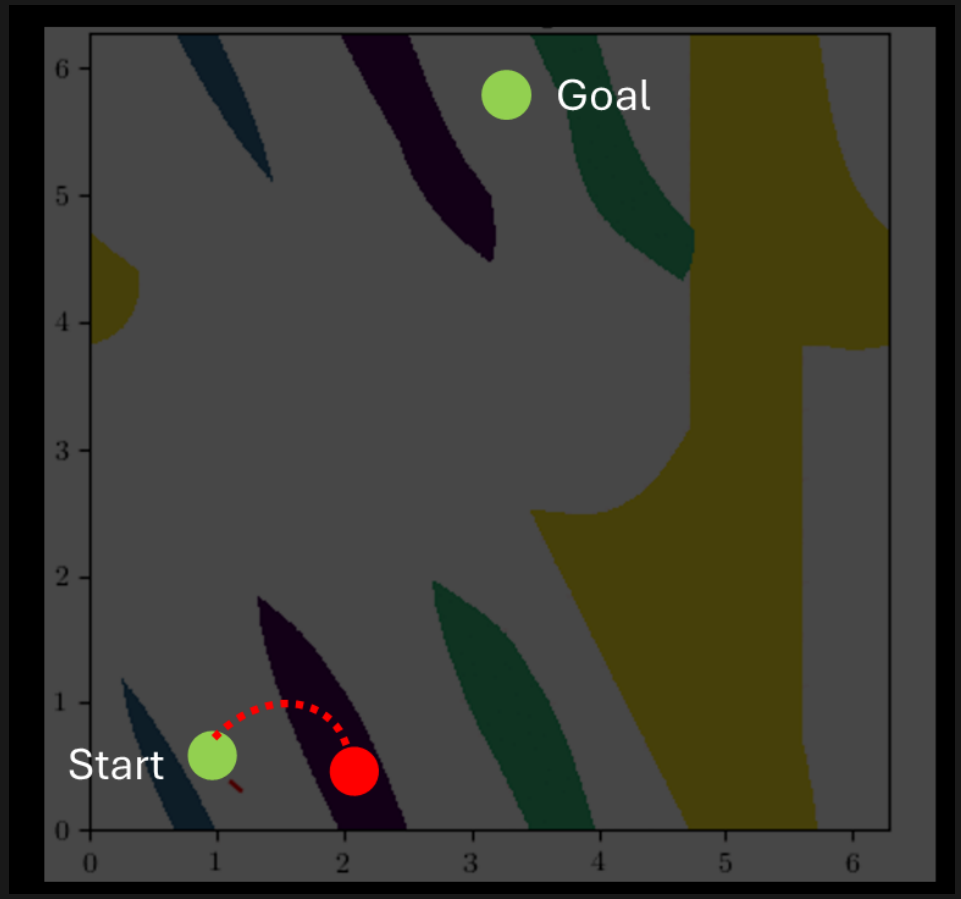
Kinodynamic RRT



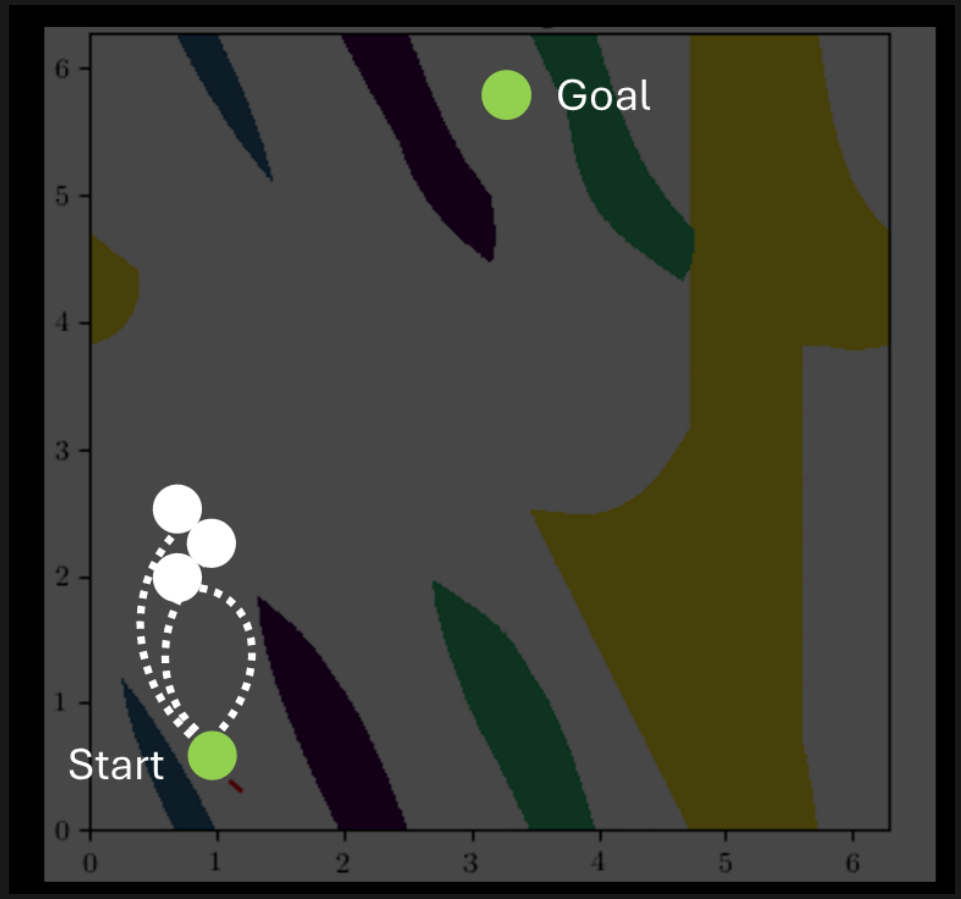
Kinodynamic RRT



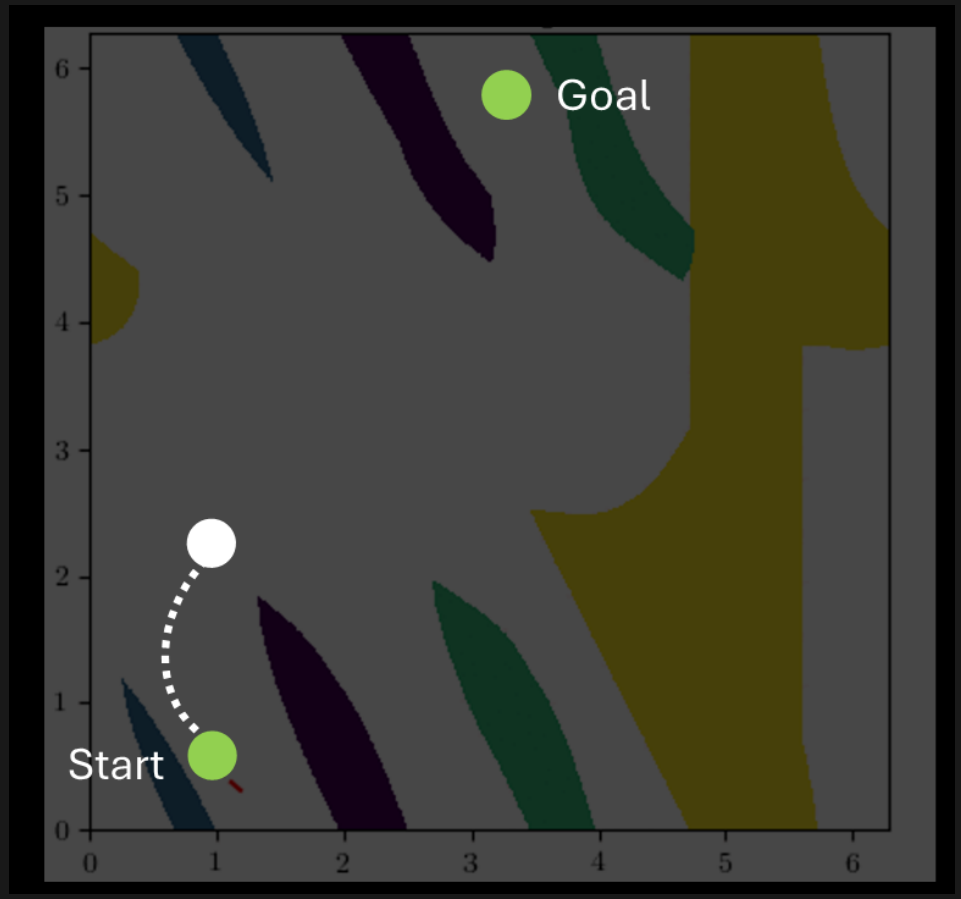
Kinodynamic RRT



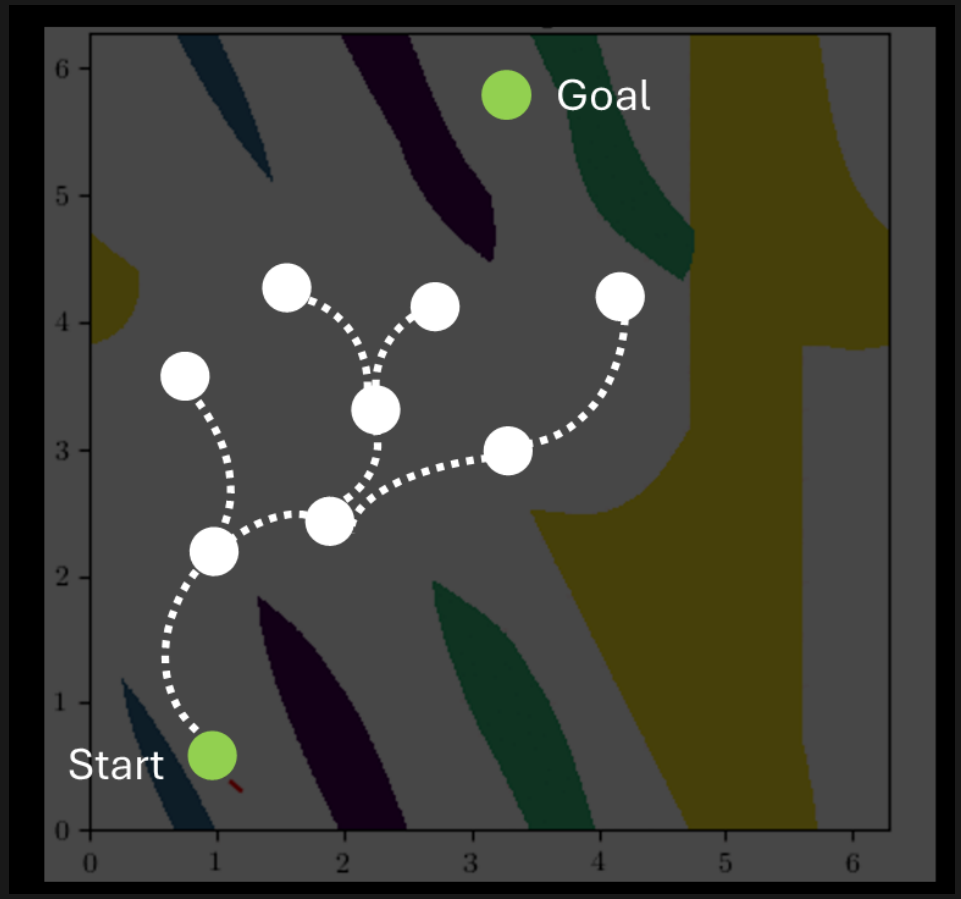
Kinodynamic RRT



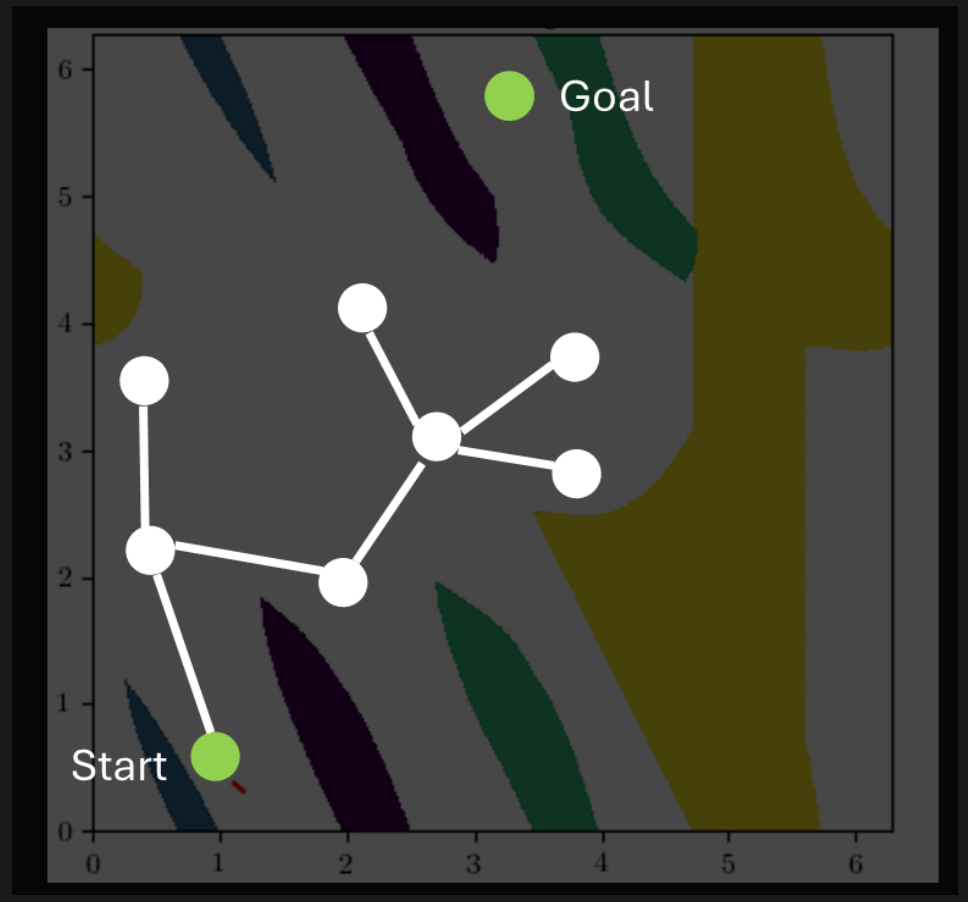
Kinodynamic RRT

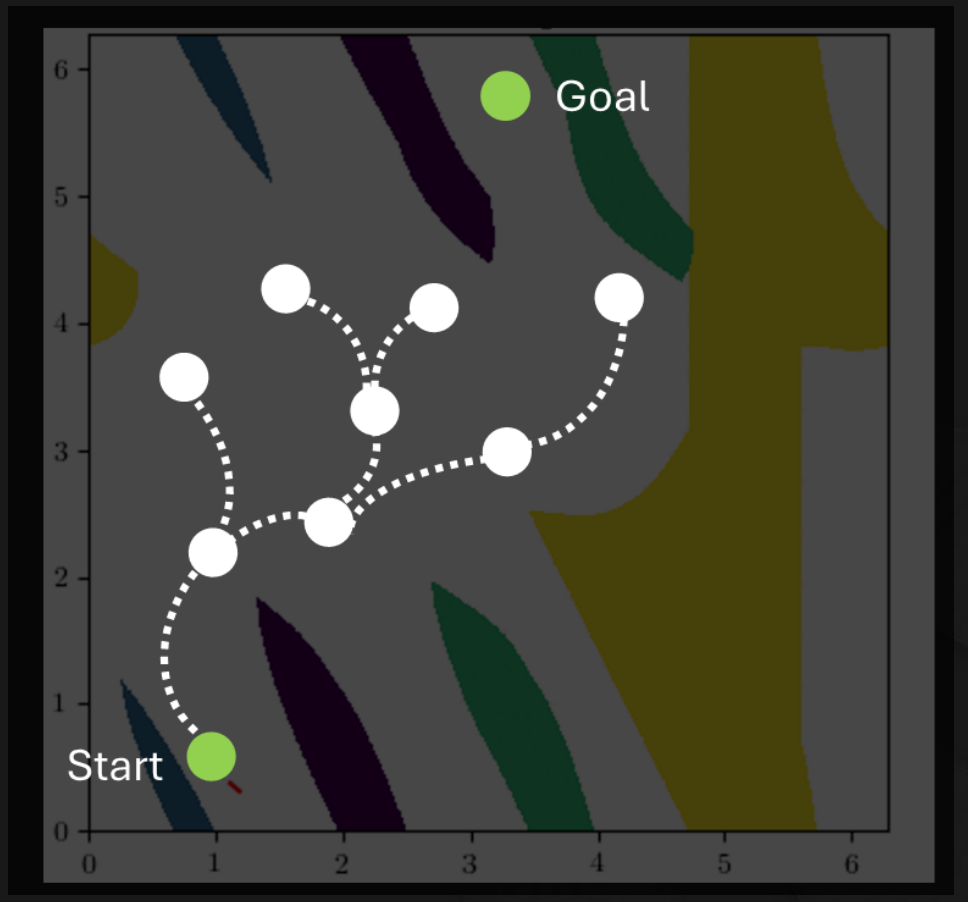


Kinodynamic RRT

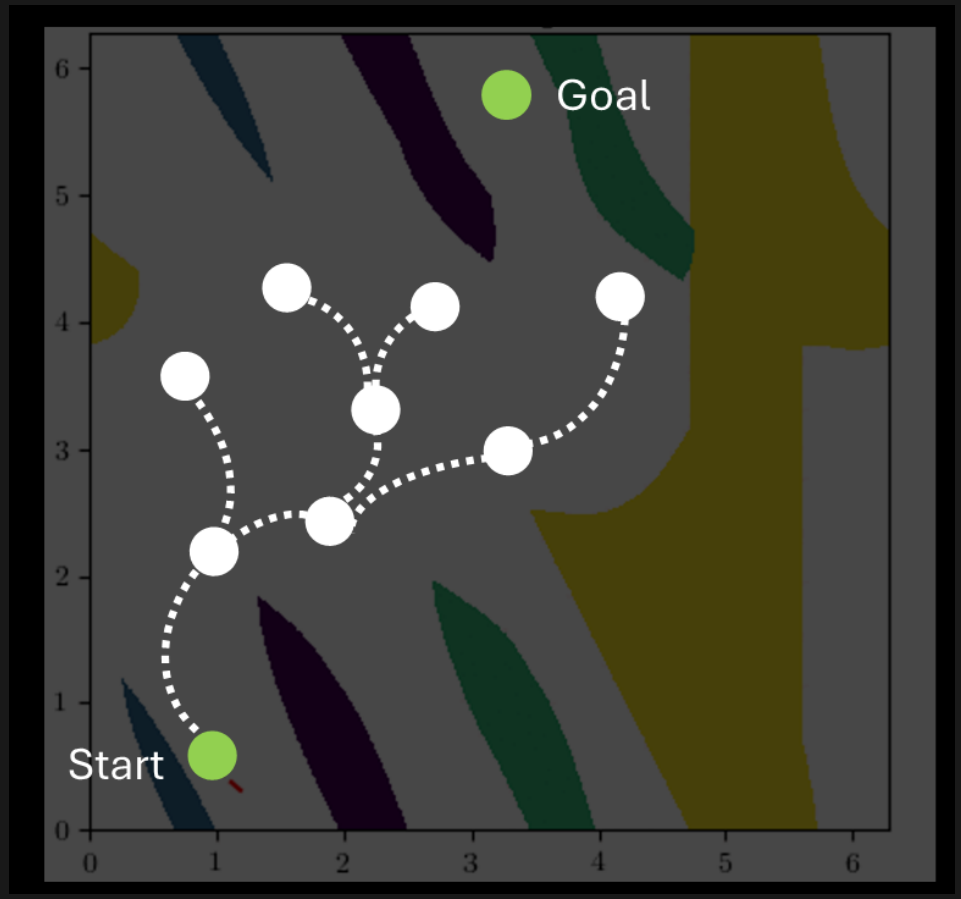


Kinodynamic RRT

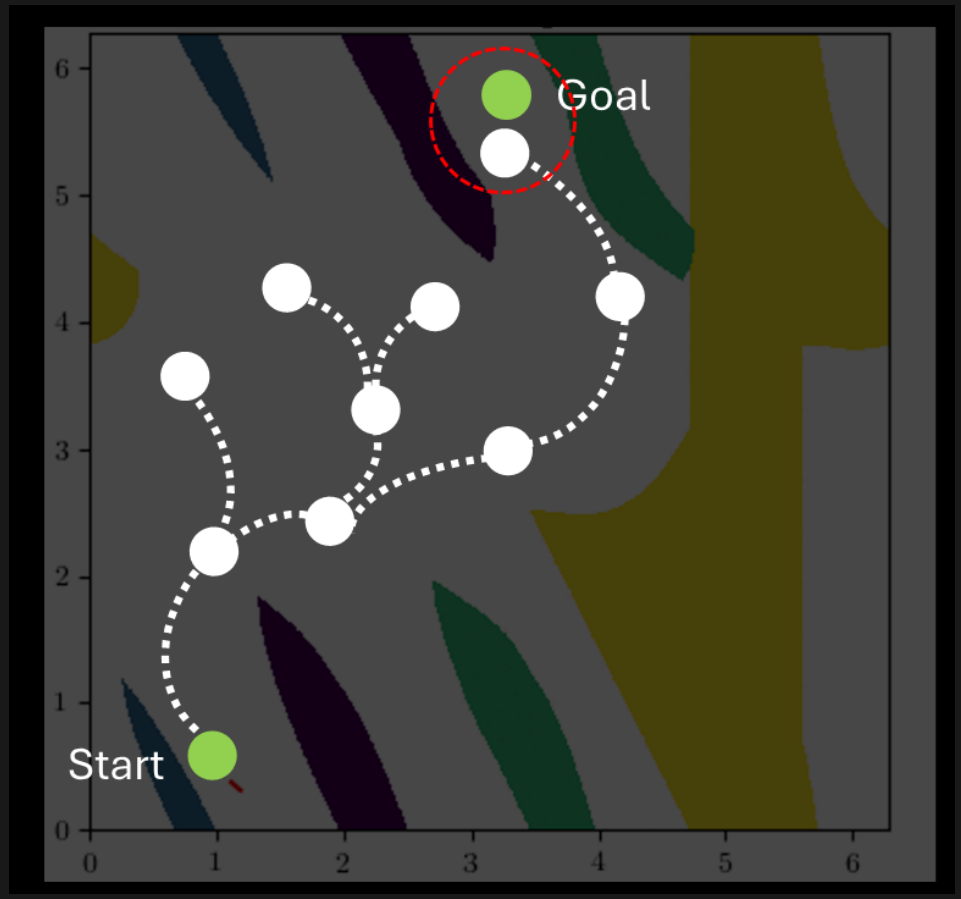




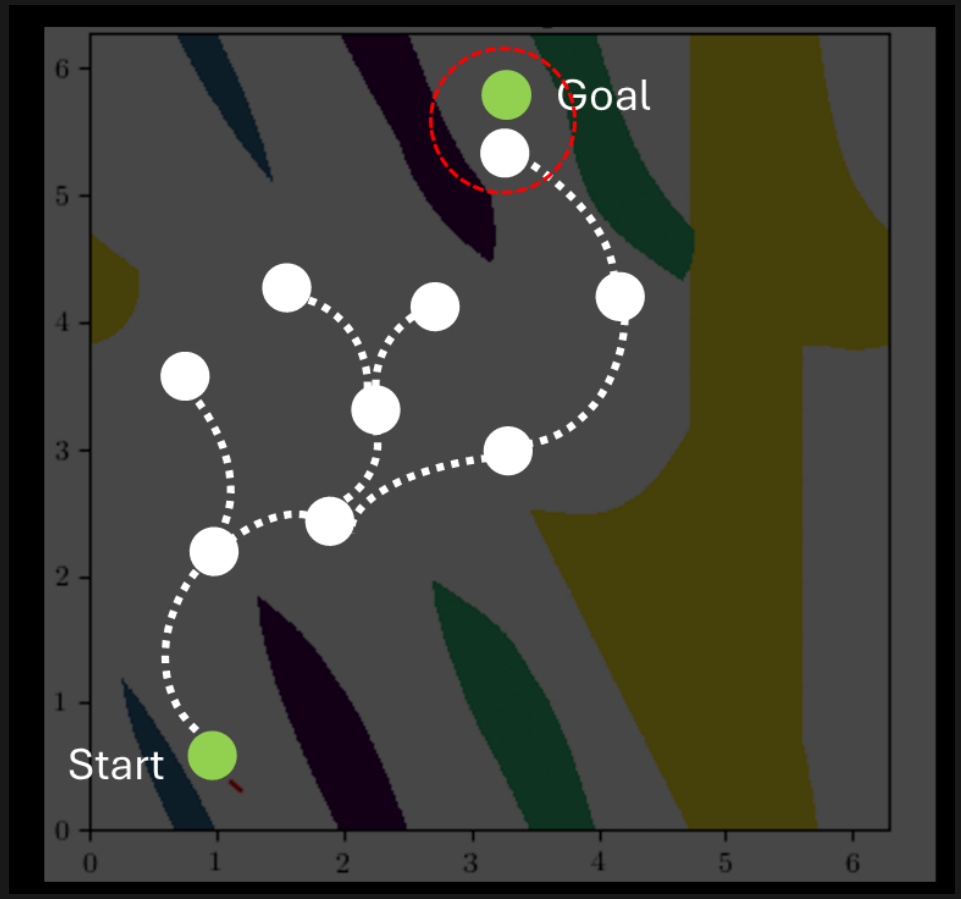
Kinodynamic RRT



Kinodynamic RRT

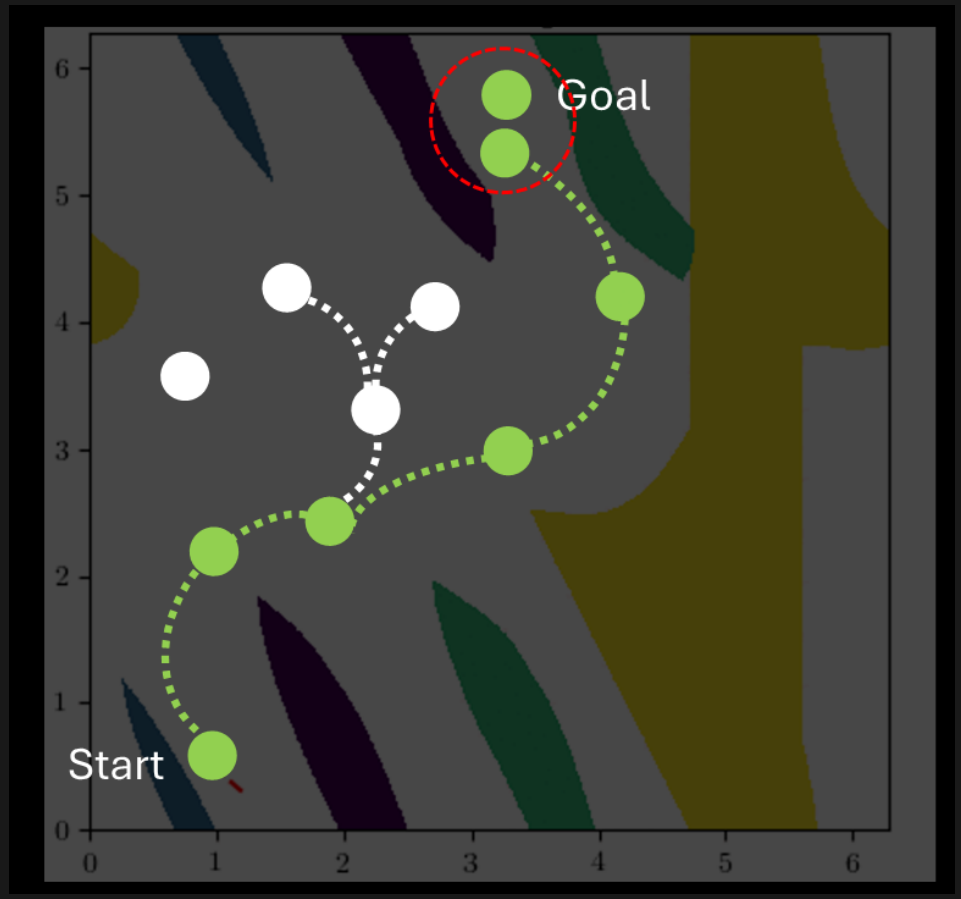


Kinodynamic RRT



Kinodynamic RRT

- 주요 특징
  - Single-query
  - Narrow Passage
  - 최적의 경로 보장 X
    도착 지점이 일치하지 않을 수 있음
  - Probabilistically Complete



Kinodynamic RRT

Algorithm 4 Rapidly-exploring Random Tree (RRT)

Require: Maximum iterations N, step size δ, start q_{start}, goal q_{goal}
Ensure: A path from q_{start} to q_{goal}, if one exists

Initialize tree $T \leftarrow \{q_{\text{start}}\}$
for $i = 1$ to N **do**
  Sample random configuration q_{rand}
  $q_{\text{near}} \leftarrow \text{Nearest}(T, q_{\text{rand}})$
  $q_{\text{new}} \leftarrow \text{Steer}(q_{\text{near}}, q_{\text{rand}}, \delta)$
  if collision-free$(q_{\text{near}}, q_{\text{new}})$ **then**
    Add q_{new} to T with edge from q_{near}
    if $q_{\text{new}} \approx q_{\text{goal}}$ **then**
      return Extract path from q_{start} to q_{goal}
    end if
  end if
end for
return Failure (no path found)

Algorithm 7 Kinodynamic RRT

Require: Maximum iterations N, set of control inputs $\mathcal{U}$
  time step Δt, system dynamics $f(x, u)$
  start state x_{start}, goal state x_{goal}
Ensure: A dynamically feasible trajectory from x_{start} to x_{goal}

Initialize tree $T \leftarrow \{x_{\text{start}}\}$
for $i = 1$ to N **do**
  Sample random state x_{rand}
  $x_{\text{near}} \leftarrow \text{Nearest}(T, x_{\text{rand}})$
  Sample control input $u \in \mathcal{U}$ and duration τ
  $x_{\text{new}} \leftarrow \text{ForwardSimulate}(x_{\text{near}}, u, \tau, f)$
  if collision-free trajectory from x_{near} to x_{new} **then**
    Add x_{new} to T with edge $(x_{\text{near}}, x_{\text{new}})$
    if $x_{\text{new}} \approx x_{\text{goal}}$ **then**
      return Extract trajectory to x_{goal}
    end if
  end if
end for
return Failure (no feasible trajectory found)

RRT vs. Kinodynamic RRT

RRT

- Tree
- Single-query
- Probabilistic Completeness
- Narrow Passage Problem
- Dynamics를 고려하지 않음 (geometric planning)

RRT vs. Kinodynamic RRT

Kinodynamic RRT

- Tree
- Single-query
- Probabilistic Completeness
- Narrow Passage Problem
- Dynamics를 고려함

강의 요약

01

Kinodynamic RRT

- Dynamics를 고려한 RRT
- 최적의 경로 보장 X 도착 지점이 일치하지 않을 수 있음
- Probabilistically Complete

02

알고리즘

03

코드 분석