[CS686] 모션 플래닝 및 응용

# Sampling-based Kinodynamic Planning

**Seungwook Lee**
(이승욱)

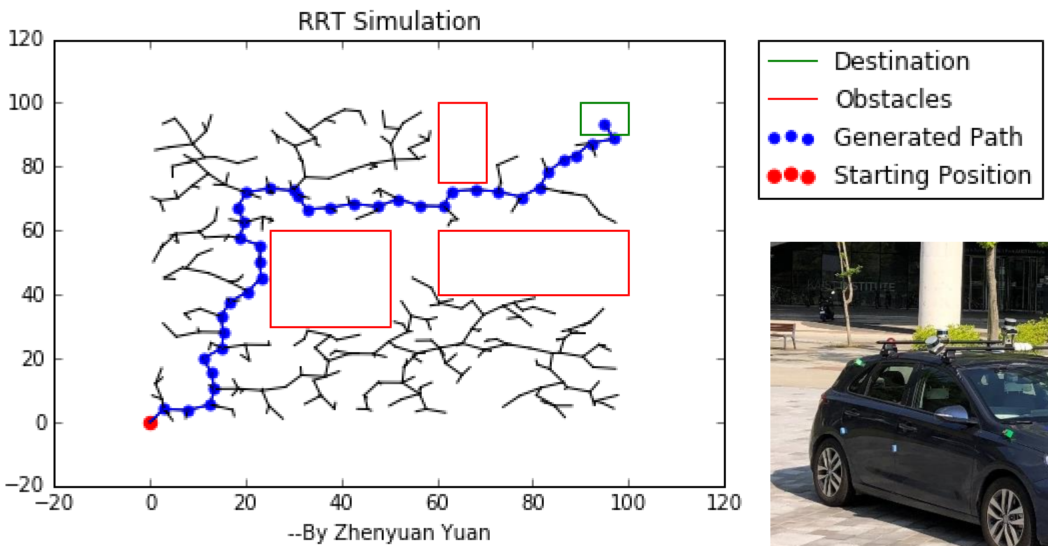
**2019. 11. 12**

# Index

- Motivations
- Approaches
- Paper 1
- Paper 2

# Motivation

- **Real-world implementation**
  - **How to follow the jerky path…**

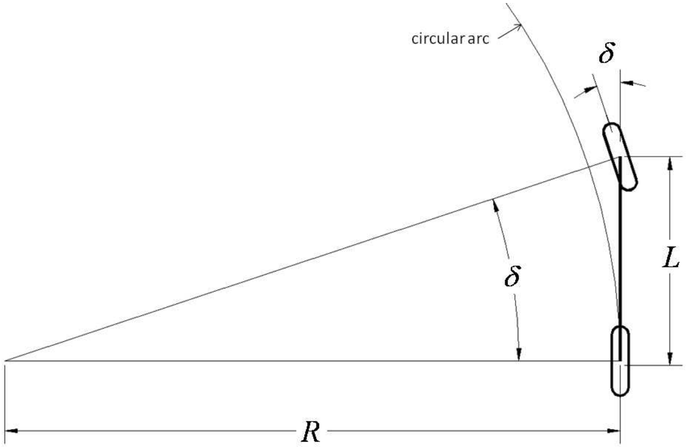



# Motivation

- Constraints exist in real-world
  - May face dynamic environments
  - Inertia
  - Limited controllability
  - Limited sensors
  - Limited actuators
  - Example: for cars, steering angle and its derivative are finite.

# Motivation

- **Kinematic constraints**
  - **Mechanum wheeled robot vs. Car-like robot**
  - **Cannot perform translation to the sides.**




- **Dynamic constraints**
  - **Actuation force is limited**
  - **Limited a=F/m -> limited v -> limited x**

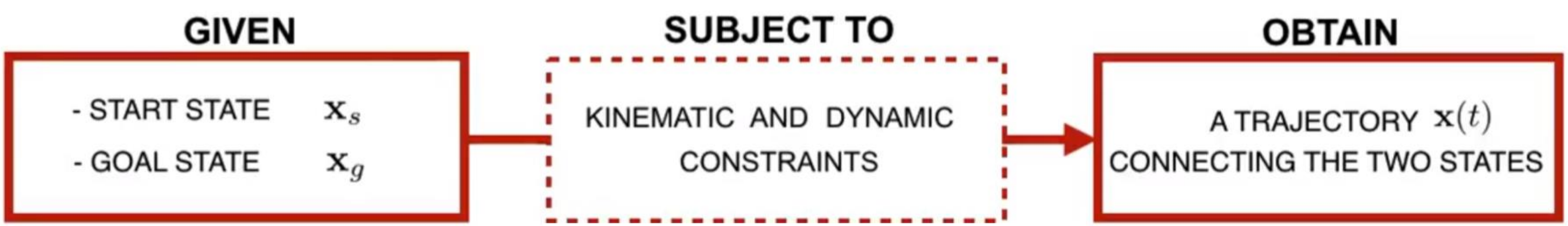
Bicycle model:

$$\delta = \operatorname{atan}\frac{L}{R}$$

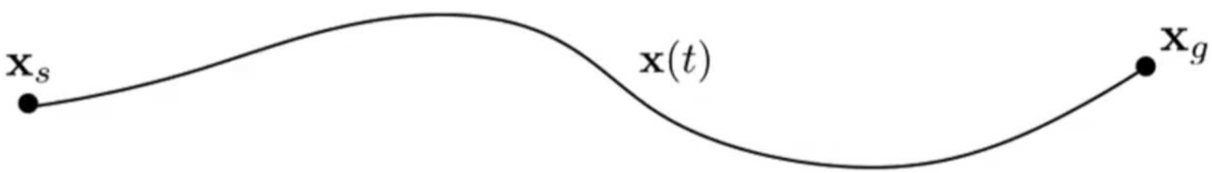
$\delta$: steering angle
L: car length
R: turn radius

# Motivation

- **Problem Statement**




$$x(t+1) = f(x(t), u(t))$$
$$x(t) \leq c$$




where, $\boldsymbol{x(t)} = [p, \dot{p}, \dots, \theta, \dot{\theta}, \dots]$



*Randomized Kinodynamic Planning for Constrained Systems, ICRA 2018 Youtube

# Previous Researches

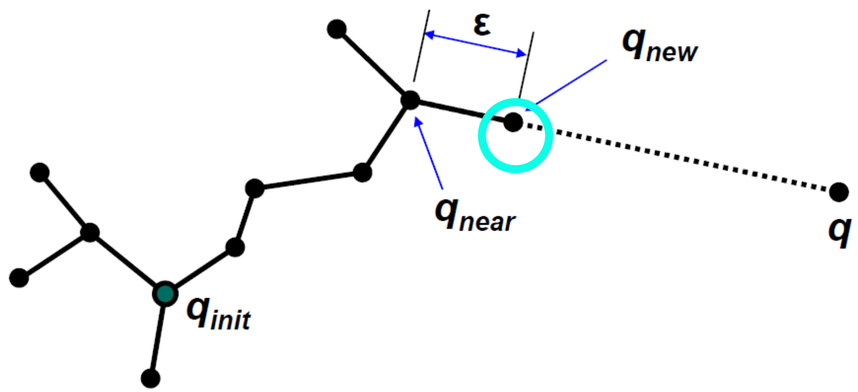
- **LaValle** and **Kuffner**(2001): Randomized Kinodynamic Planning
- **Webb** and **van den Berg**(2013): Kinodynamic RRT*: Asymptotically Optimal Motion Planning for Robots with Linear Dynamics
- **Allen** and **Pavone**(2016): The Real-Time Framework for Kinodynamic Planning Applied to Quadrotor Obstacle Avoidance

# Approaches

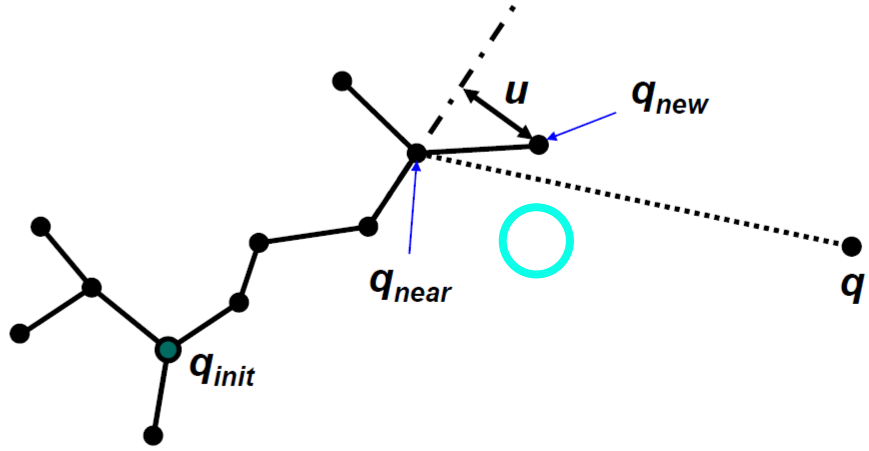
- **KCRSS: Kinematic Constraints based Random State Search**
- **Closed-loop predictions**

# KCRSS(Kinematic Constraints based Random State Search)

- **No constraints**
  - Define **q_new** directly

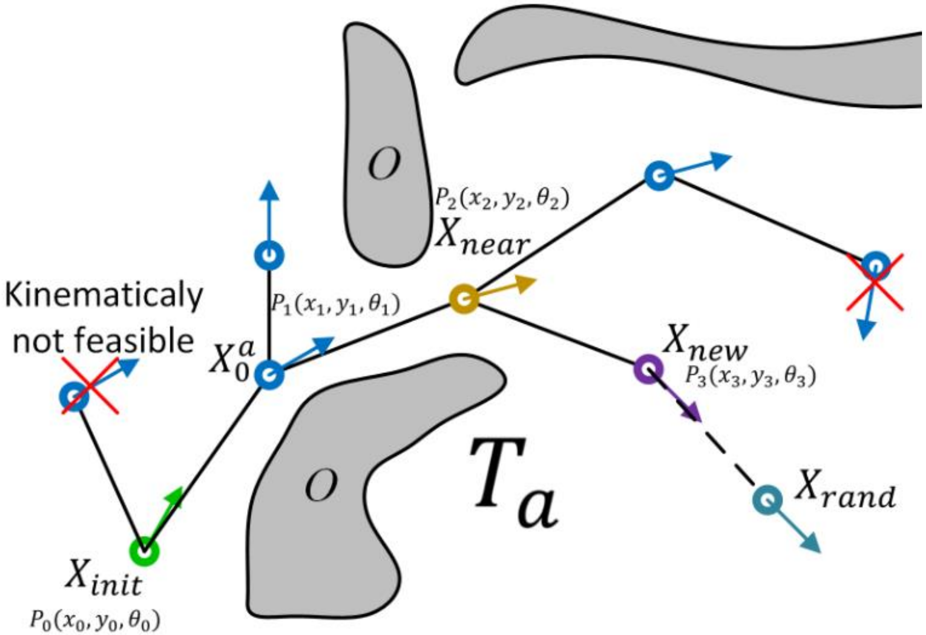


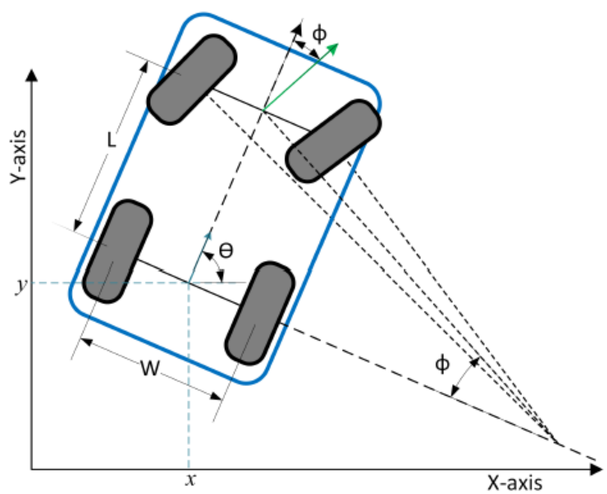

- **With constraints**
  - Define **q_new** as far as **u** permits



# KCRSS(Kinematic Constraints based Random State Search)

- **Impose kinematic constraints in the node generation process.**
- **Add only the kinematically feasible nodes -> reduction of nodes.**



# KCRSS(Kinematic Constraints based Random State Search)

- **Identify deviation of orientation $\alpha$**
- **Propagate states based on kinematic constraints**
- **Collision check**

**Algorithm 1** KCRSS $(X_c, X_r)$

**Input:** $u, L, \delta t, \delta s$
**Output:** $x_{new}, y_{new}, \theta_{new}$
$length = 0$
**while** $\| (x,y)_c - (x,y)_r \| \leq v * \delta t$ **and** $length \leq \delta s$ **do**
  $\alpha = \theta_c - atan2(y_r - y_c, x_r - x_c)$
  **if** $abs(\alpha) > \phi$ **then**
    $$\alpha = \begin{cases} \phi & if \alpha > 0 \\ -\phi & if \alpha < 0 \end{cases}$$
  **end if**
  $x_{new} = x_c + v * \delta t * cos(\theta_c + \omega * \delta t)$
  $y_{new} = y_c + v * \delta t * sin(\theta_c + \omega * \delta t)$
  $\theta_{new} = \theta_c - (v/L) * tan(\alpha) * \delta t$
  $CurrentCell = Cell(x_{new}, y_{new})$
  **if** $CurrentCell \in O$ **then**
    **return false**
  **else**
    $length = length + |v| * \delta t$
  **end if**
**end while**

# Closed-loop Predictions

- **Simulate -> obtain output x**
  - $u(t) = g\big(r(t)\big)$
  - $x(t+1) = f(x(t), u(t))$
- **Dynamically feasible by construction.**

# Closed-loop Predictions

- **Sample an output point y_rand**
- **Improve solution**
- **Extract a reference with lowest-cost trajectory.**

**Algorithm 1:** The CL-RRT$^{\#}$ Algorithm

**CL-RRT$^{\#}$**$(x_{\text{init}}, X_{\text{goal}}, X)$<br>
  $Y_{\text{goal}} :=$ `OutputMap`$(X_{\text{goal}})$;<br>
  $\mathcal{S} \leftarrow$ `Initialize`$(x_{\text{init}}, Y_{\text{goal}})$;<br>
  **for** $k = 1$ *to* $N$ **do**<br>
    $y_{\text{rand}} \leftarrow$ `Sample`$(k)$;<br>
    $\mathcal{S} \leftarrow$ `Extend`$(\mathcal{S}, Y_{\text{goal}}, y_{\text{rand}})$;<br>
    $\mathcal{S} \leftarrow$ `Replan`$(\mathcal{S})$;<br>
  $\mathcal{T}_x \leftarrow$ `ConstrSolution`$(\mathcal{S})$;<br>
  **return** $\mathcal{T}_x$;

# Paper 1

**Author**: Ghosh

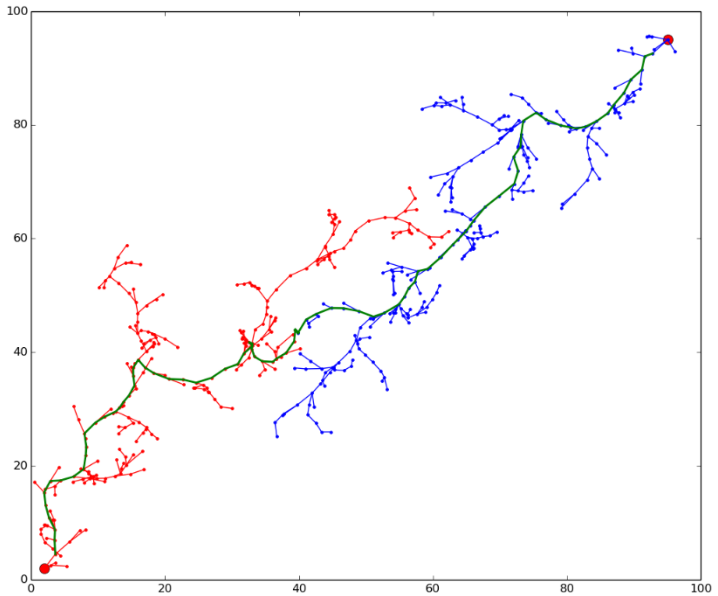
**Title**: Kinematic Constraints Based Bi-directional RRT (KB-RRT) with Parameterized Trajectories for Robot Path Planning in Cluttered Environment

**Conference**: ICRA 2019

- **KCRSS**
- **BI-RRT**
- **Improved time(iterations) and memory usage**

# BI-RRT(Bidirectional RRT)

- **Effective in narrow environments**
- **Efficient computing**
- **Grow two trees**



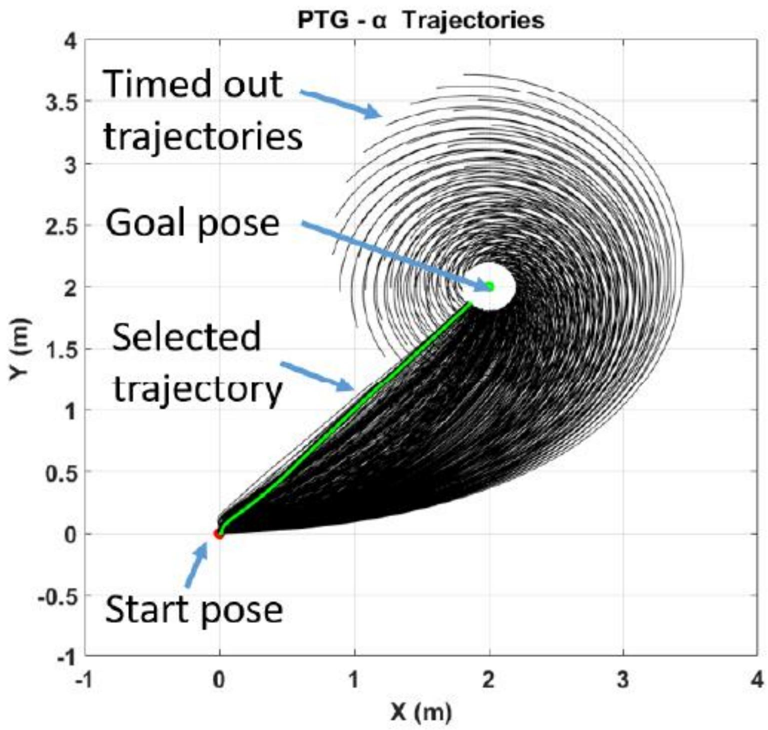

$$T_a = \{X_{init}, X_a^0, X_a^1, X_a^2, \dots, X_a^{n-1}\}$$
$$T_b = \{X_{goal}, X_b^0, X_b^1, X_b^2, \dots, X_b^{m-1}\}$$

$$\Downarrow$$

$$T_p = \{X_{init}, X_a^0, X_a^1, \dots, X_a^{n-1}, X_b^{m-1}, \dots, X_b^0, X_{goal}\}$$

# Trajectory Generation

- **Resulting trajectory may not be optimal – need preprocessing**
- **Parametrized Trajectory Generator (PTG−α)**



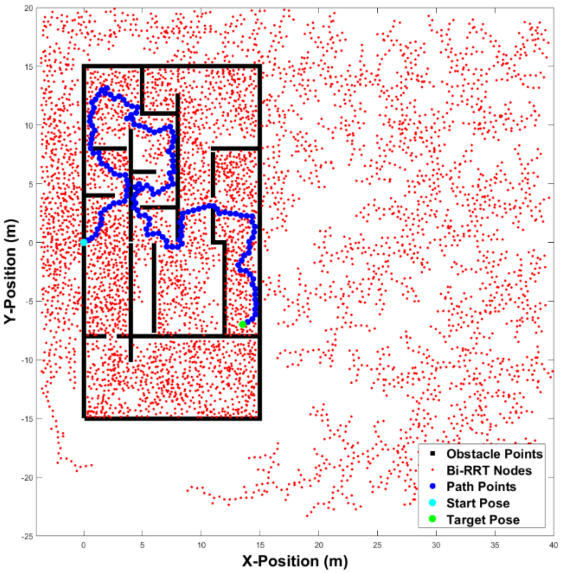



KAIST

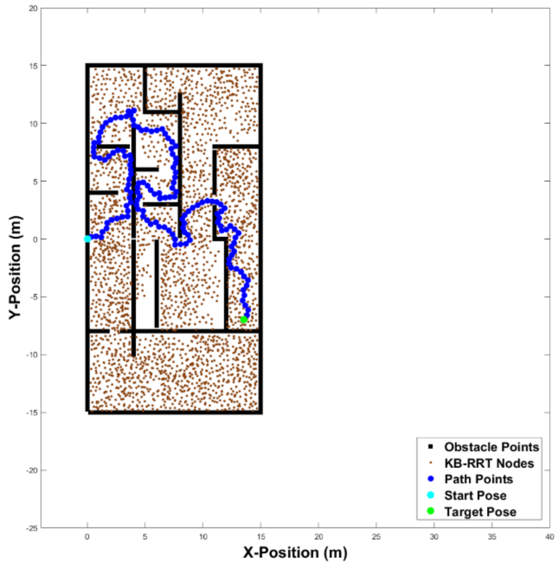
# Experiment Scenarios

- **Scenario 1: Maze**(one open)
- **Scenario 2: Tunnel**
- **Scenario 3: Maze**(no open)

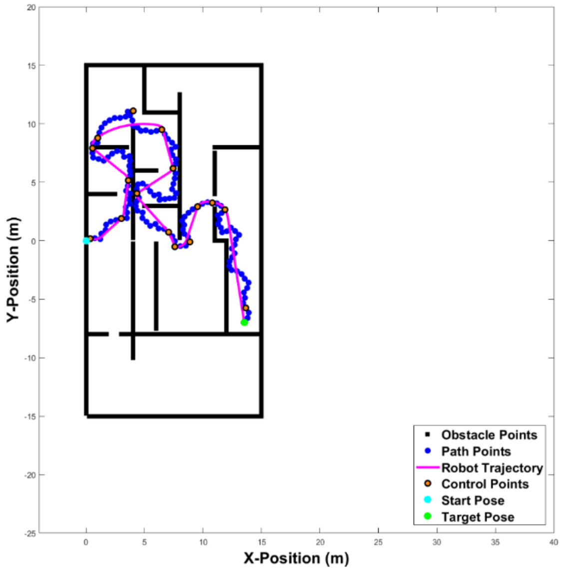
# Scenario 1




(a) Scenario 1: Bi-RRT

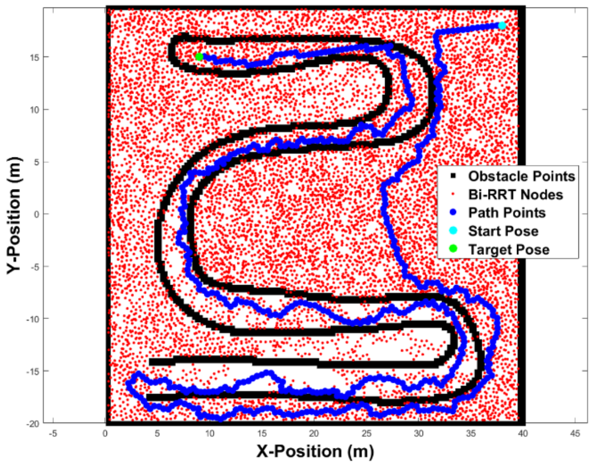



(b) Scenario 1: KB-RRT

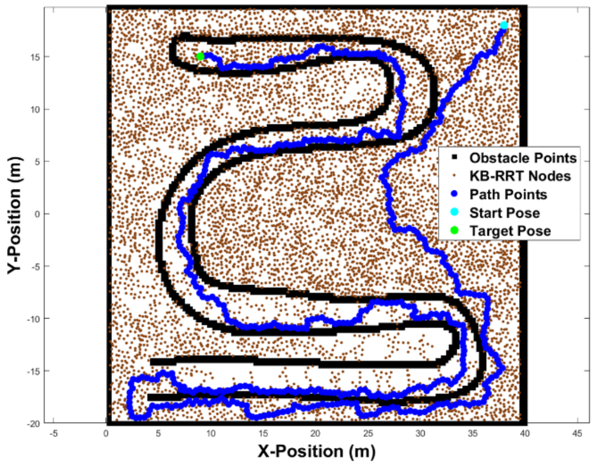



(c) Scenario 1: Generated trajectory

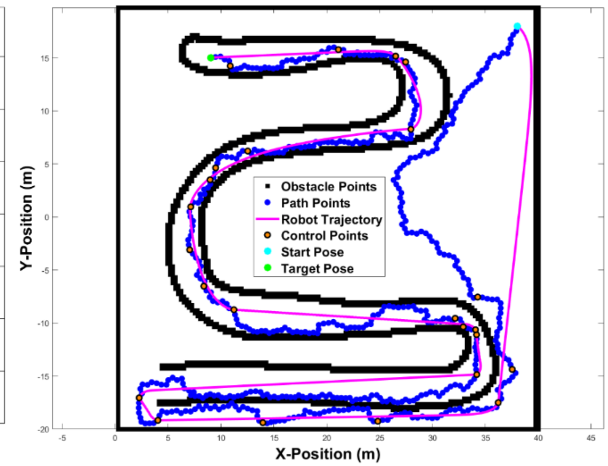
# Scenario 2



(d) Scenario 2: Bi-RRT

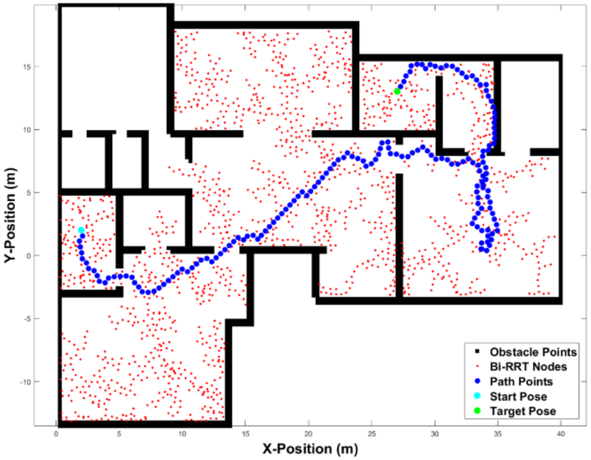


(e) Scenario 2: KB-RRT

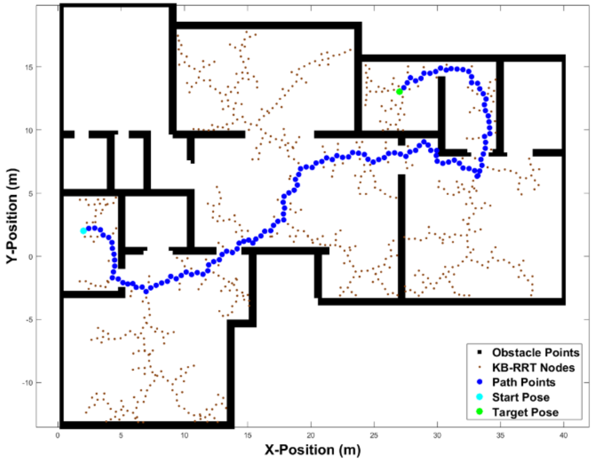


(f) Scenario 2: Generated trajectory

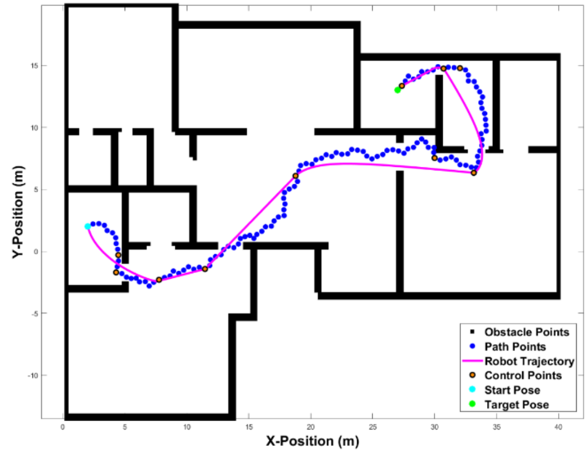
# Scenario 3




(g) Scenario 3: Bi-RRT




(h) Scenario 3: KB-RRT




(i) Scenario 3: Generated trajectory

TABLE I: Details of the test scenarios

| Sl. No | Start Point $(x,y)$ | Target Point $(x,y)$ | No of Obstacles |
|---|---|---|---|
| Scenario 1 | $(0,0)$ | $(13.50,-7.0)$ | 1031 |
| Scenario 2 | $(3.80,18.0)$ | $(9.0,15.0)$ | 2695 |
| Scenario 3 | $(0,2.0)$ | $(27.0,13.0)$ | 1581 |

TABLE II: Performance comparison of Bi-RRT and KB-RRT

| Sl. No | Algorithm | No. of Nodes | Iterations | Memory (KB) |
|---|---|---|---|---|
| Scenario 1 | KB-RRT | 3061 | 16842 | 1635 |
| | Bi-RRT | 6079 | 57456 | 3150 |
| Scenario 2 | KB-RRT | 8759 | 13133 | 1595 |
| | Bi-RRT | 10437 | 55376 | 3600 |
| Scenario 3 | KB-RRT | 787 | 1171 | 274.3 |
| | Bi-RRT | 1707 | 17196 | 1521 |

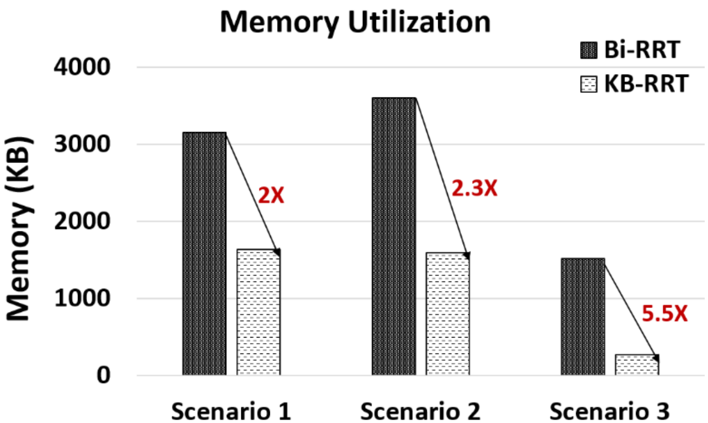



Fig. 8: Comparison of memory utilization

# Paper 2

**Author**: Arslan

**Title**: Sampling-based Algorithms for Optimal Motion Planning Using Closed-loop Prediction

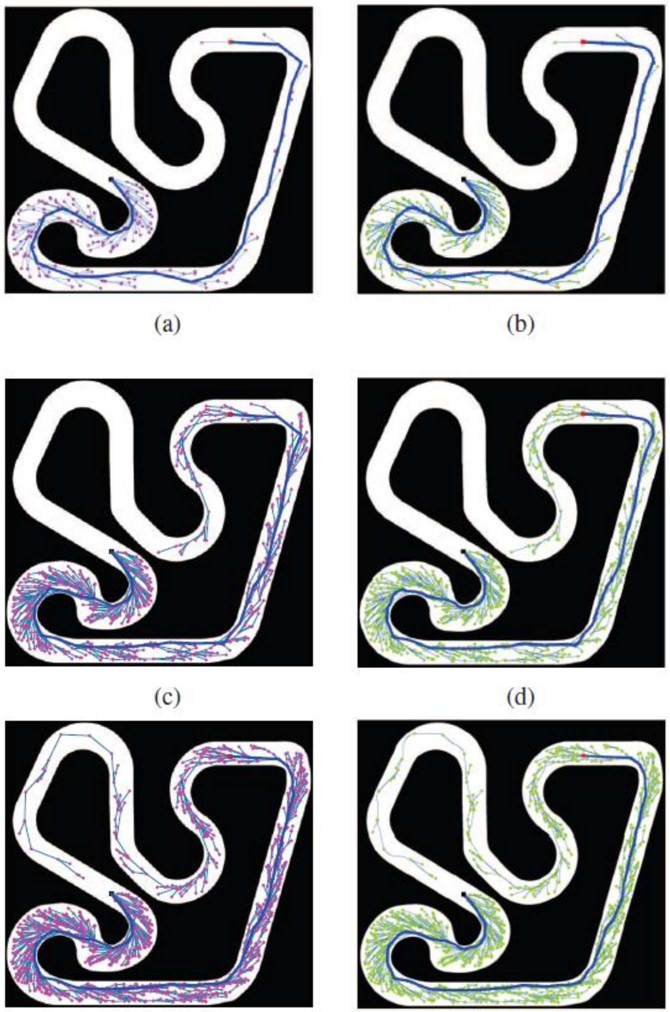
**Conference**: ICRA 2017

- **Closed-loop prediction**
- **RRT#**

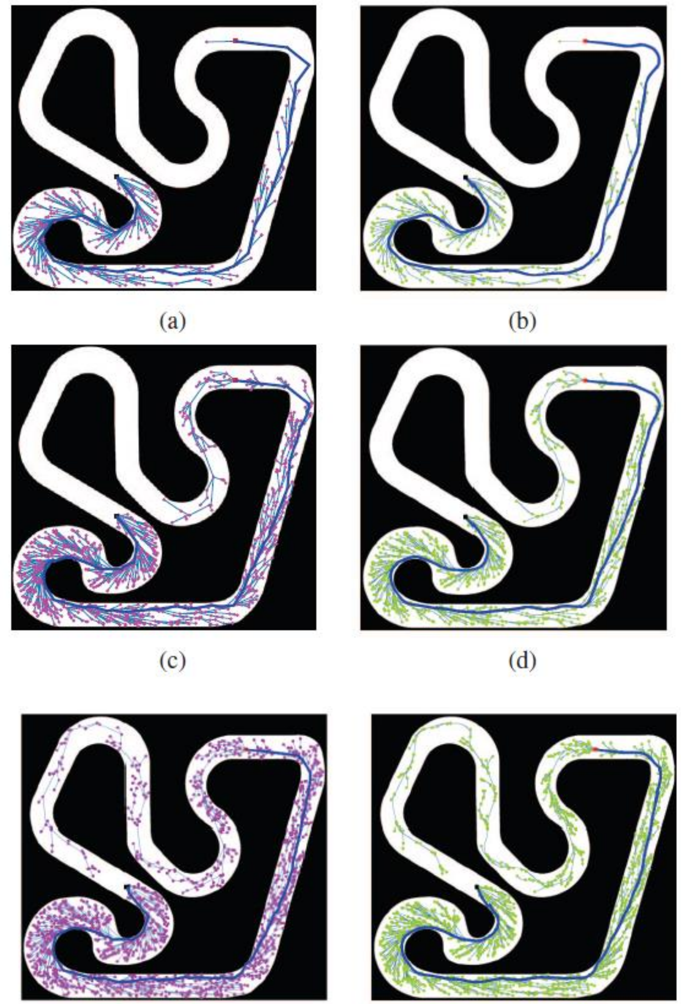
# RRT* vs. RRT#

- **Classify vertices: 4 types of cost-to-come**
- **Can utilize promising neighbor vertex**
- **No expansion of non-promising vertices -> better speed**

# RRT* vs. RRT#



(a) (b)
(c) (d)
(e) (f)



(a) (b)
(c) (d)
(e) (f)

# RRT* vs. RRT#

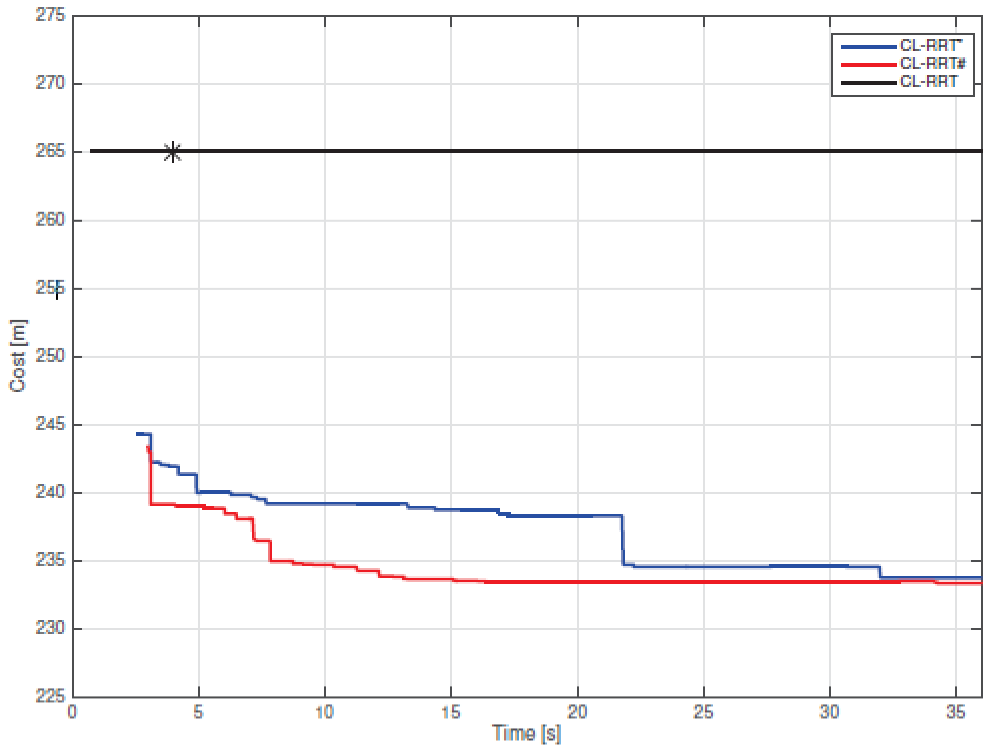
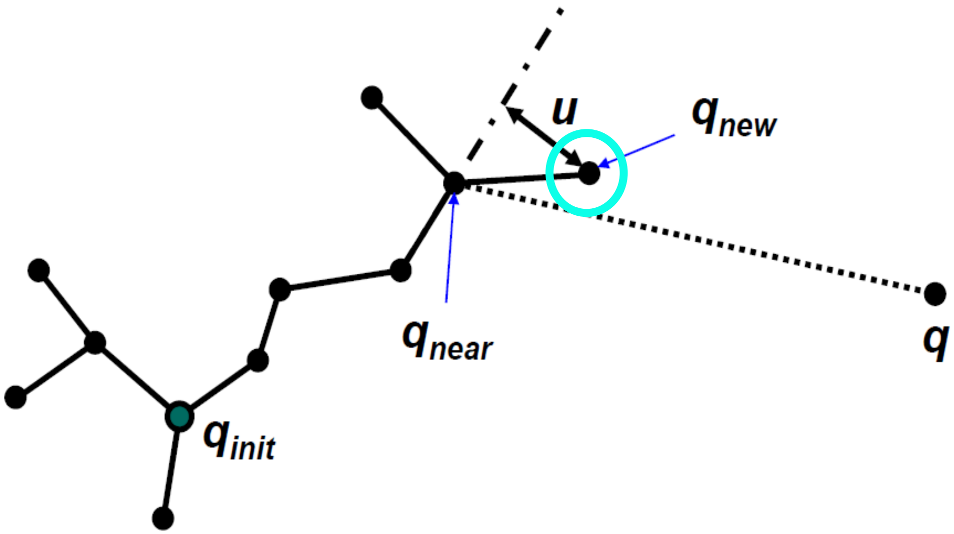



Fig. 3: Value of the cost function over the computation time. The black marker indicates the end of the 3000 iterations for CL-RRT.

# Summary

- **Define kinematics and dynamics of the robot**
- **Simulate forward**
- **Keep only the feasible nodes**



**Thank you**

# Quiz

- **Q1. When implemented to a real-world robotic system, planning and control are irrelevant to each other. (T/F)**

- **Q2. Kinodynamic feasibility is achieved by propagating the states forward in time. (T/F)**