CS686:
RRT

Sung-Eui Yoon
(윤성의)

Course URL:
http://sglab.kaist.ac.kr/~sungeui/MPA

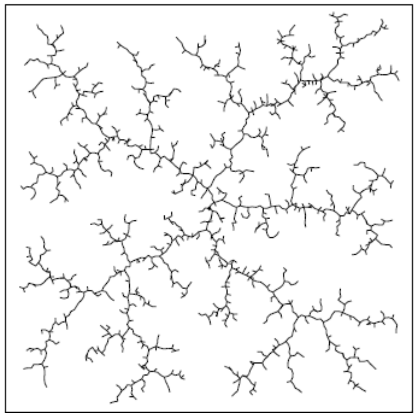
KAIST

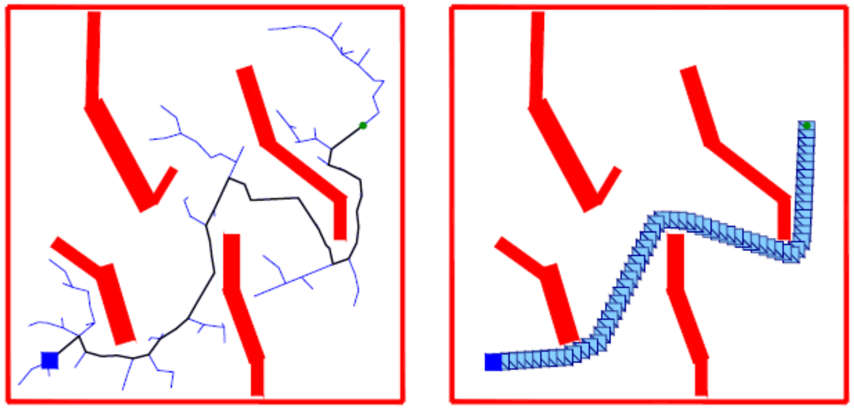
Class Objectives

- **Understand the RRT technique and its recent advancements**
  - **RRT***
  - **Kinodynamic planning**

Rapidly-exploring Random Trees (RRT) [LaValle 98]

- **Present an efficient randomized path planning algorithm for single-query problems**
  - **Converges quickly**
  - **Probabilistically complete**
  - **Works well in high-dimensional C-space**



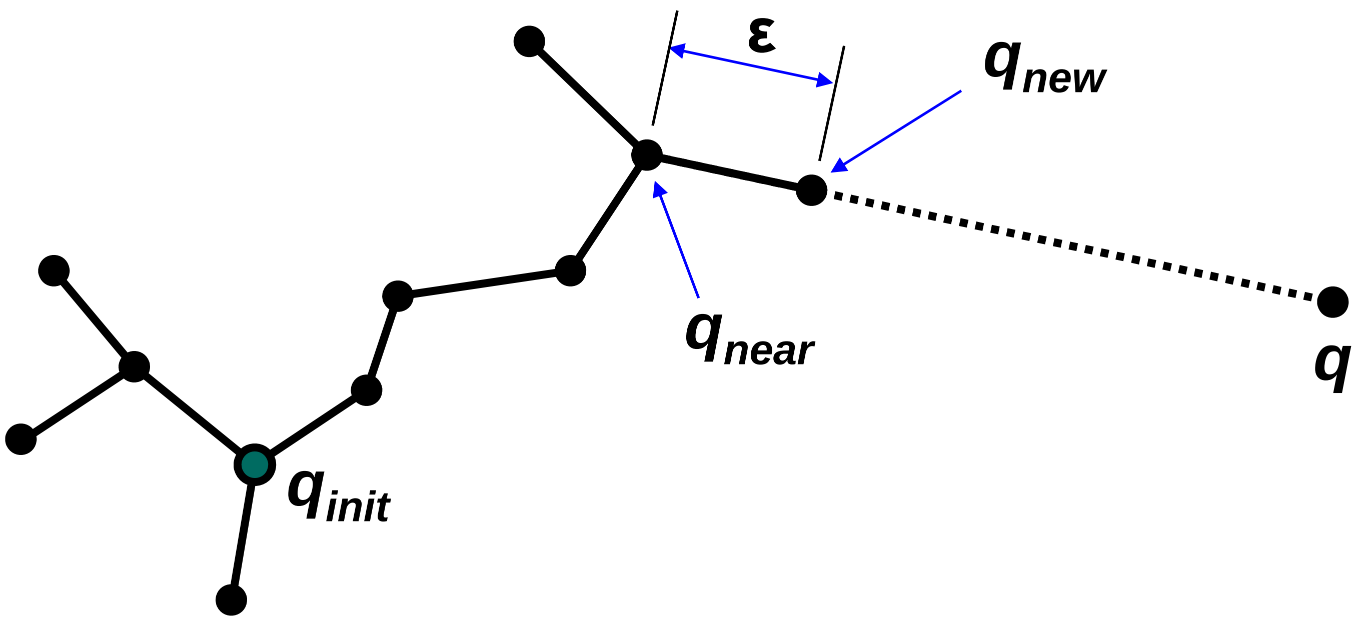


Rapidly-Exploring Random Tree

- **A growing tree from an initial state**

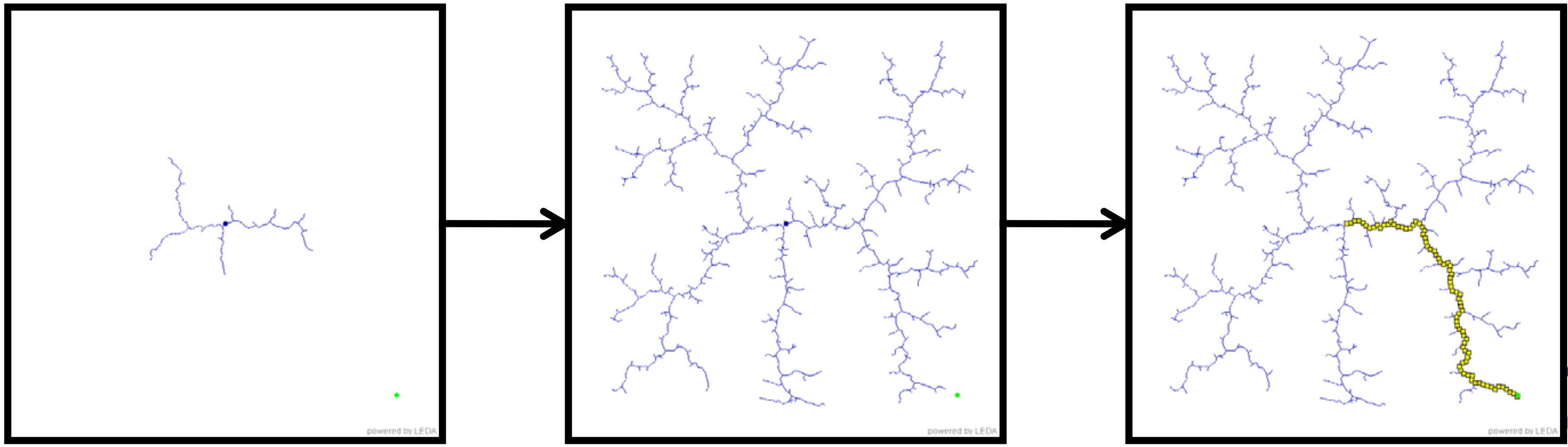
RRT Construction Algorithm

- **Extend a new vertex in each iteration**



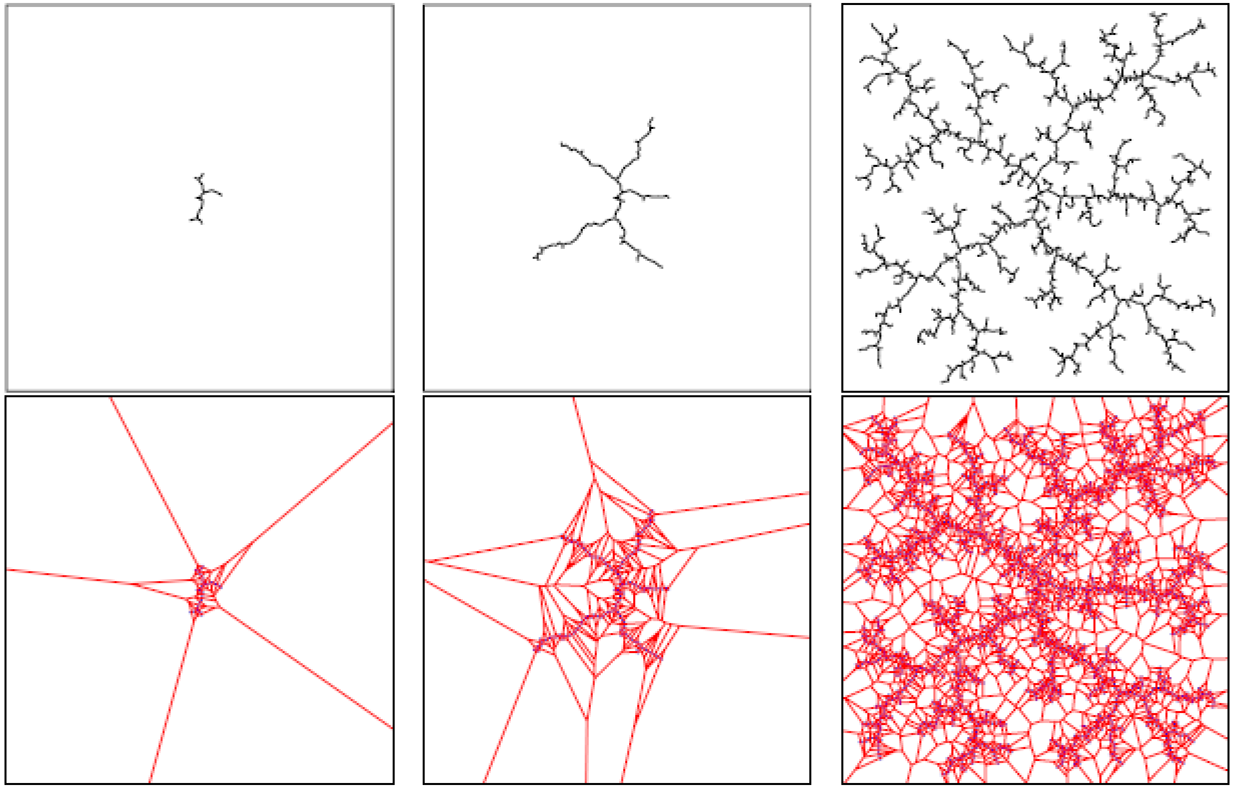
Overview – Planning with RRT

- **Extend RRT until a nearest vertex is close enough to the goal state**
  - **Biased toward unexplored space**
  - **Can handle nonholonomic constraints and high degrees of freedom**
- **Probabilistically complete, but does not converge**



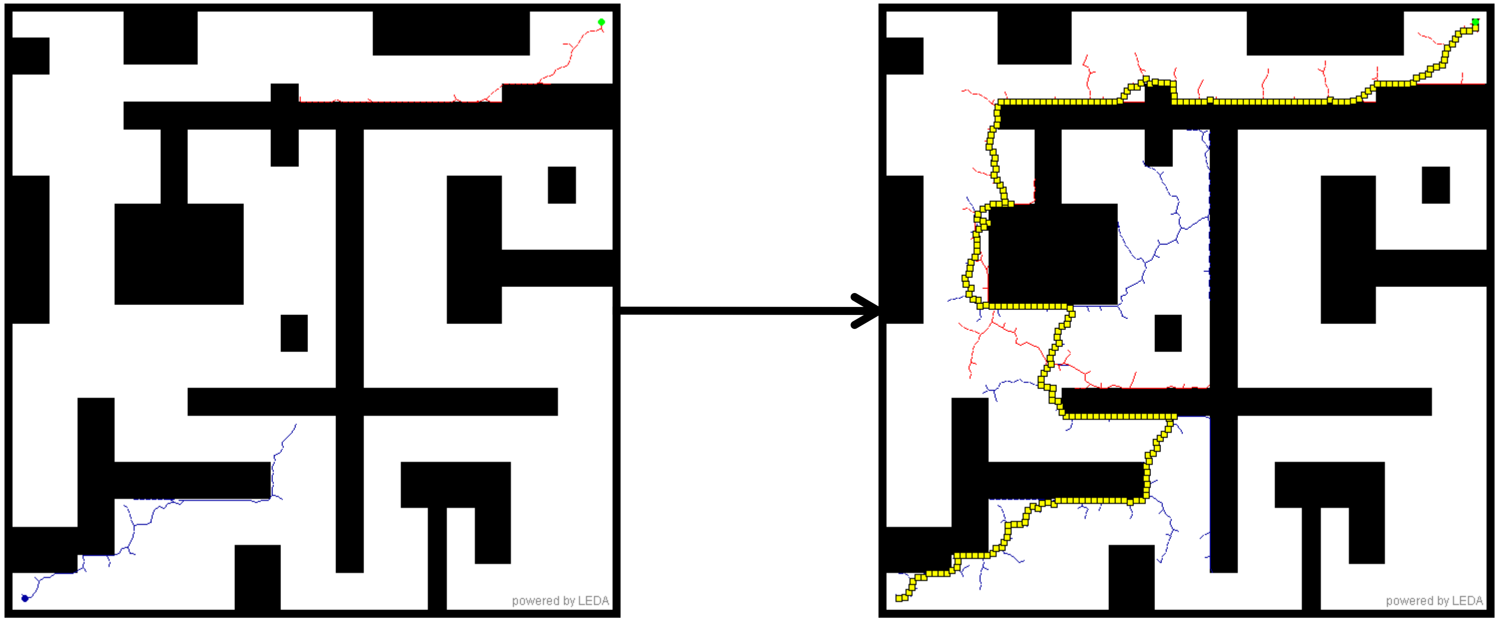
Voronoi Region

- An RRT is biased by large Voronoi regions to rapidly explore, before uniformly covering the space



Overview – With Dual RRT

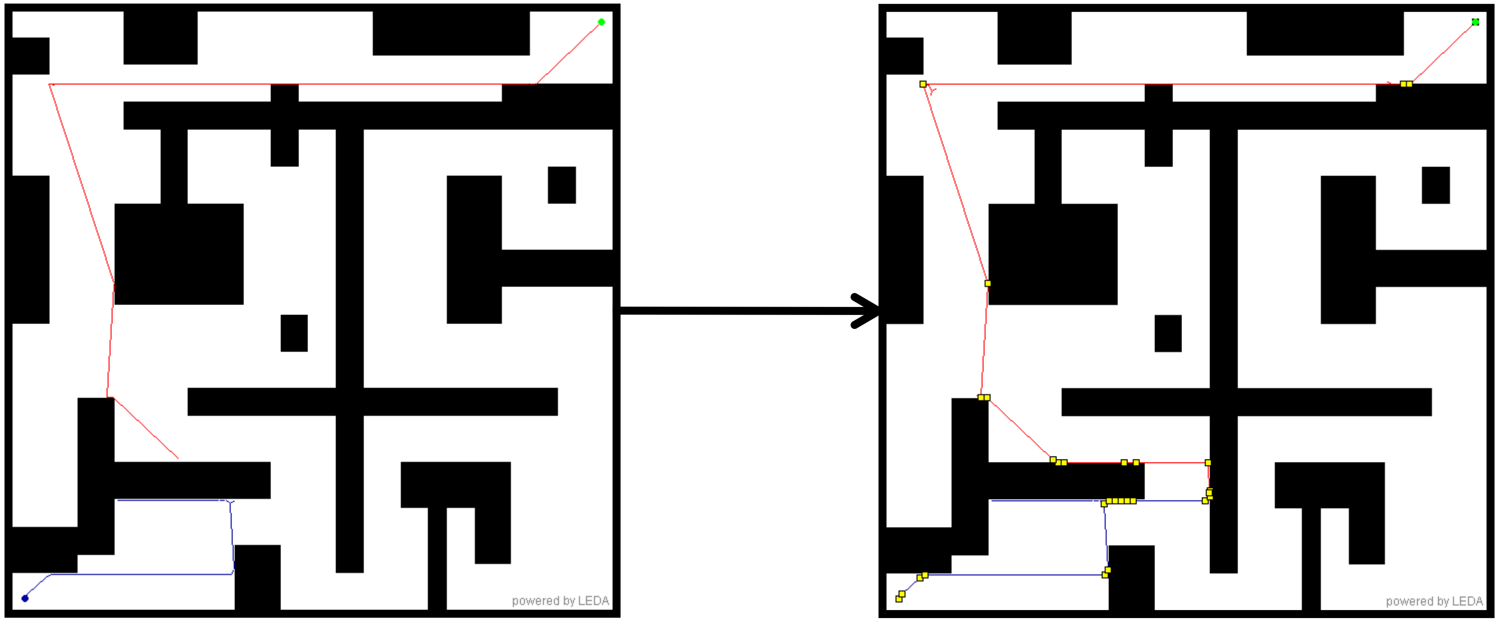
- **Extend RRTs from both initial and goal states**
- **Find path much more quickly**



737 nodes are used

Overview – With RRT-Connect

- **Aggressively connect the dual trees using a greedy heuristic**
- **Extend & connect trees alternatively**



42 nodes are used

KAIST

RRT Construction Algorithm

```
BUILD_RRT(q_init)
    T.init(q_init);
    for k = 1 to K do
        q_rand ← RANDOM_CONFIG();
        EXTEND(T, q_rand);
    Return T
```

```
EXTEND(T, q)
    q_near ← NEAREST_NEIGHBOR(q, T);
    if NEW_CONFIG(q, q_near, q_new) then
        T.add_vertex(q_new);
        T.add_edge(q_near, q_new);
        if q_new = q then
            Return Reached;
        else
            Return Advanced;
    Return Trapped;
```

RRT Connect Algorithm

CONNECT($\mathcal{T}, q$)

repeat

    $S \leftarrow$ EXTEND($\mathcal{T}, q$);

until not ($S = Advanced$)

Return S;

RRT_CONNECT_PLANNER(q_{init}, q_{goal})

$\mathcal{T}_a$.init(q_{init}); $\mathcal{T}_b$.init(q_{goal});

for $k = 1$ **to** K **do**

    $q_{rand} \leftarrow$ RANDOM_CONFIG();

    if not (EXTEND($\mathcal{T}_a, q_{rand}$) $= Trapped$) **then**

        if (CONNECT($\mathcal{T}_b, q_{new}$) $= Reached$) **then**

            Return PATH($\mathcal{T}_a, \mathcal{T}_b$);

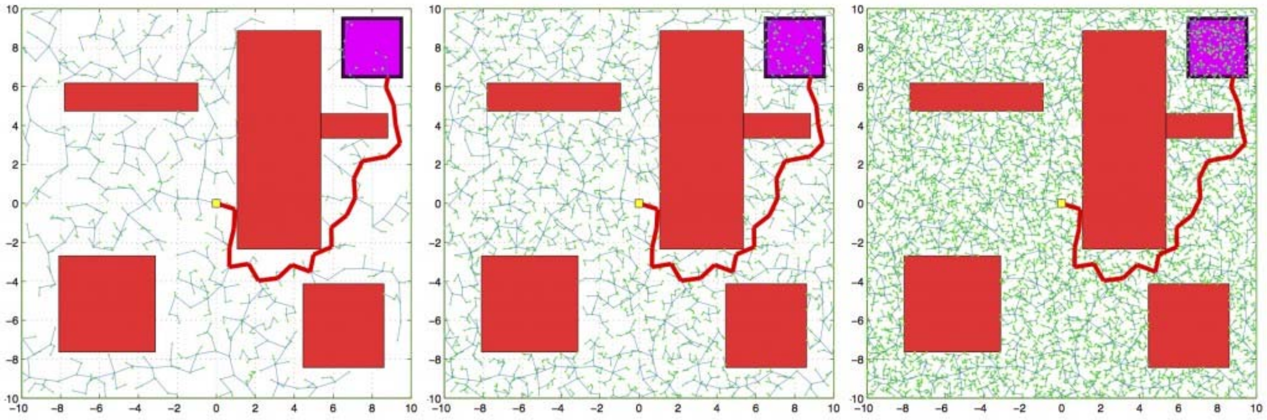
    SWAP($\mathcal{T}_a, \mathcal{T}_b$);

Return *Failure*

RRT*

- **RRT does not converge to the optimal solution**

RRT



RRT*

From Sertac's homepage

KAIST

RRT*

- **Asymptotically optimal without a substantial computational overhead**

Theorem [Karaman & Frazzoli, IJRR 2011]
(i) The RRT* algorithm is asymptotically optimal

$$\mathbb{P}\Big(\big\{ \lim_{n\to\infty} Y_n^{\mathrm{RRT}^*} = c^* \big\}\Big) = 1$$

(ii) RRT* algorithm has no substantial computational overhead when compared to the RRT:

$$\lim_{n\to\infty} \mathbb{E}\left[\frac{M_n^{\mathrm{RRT}^*}}{M_n^{\mathrm{RRT}}}\right] = \mathrm{constant}$$

- **Y_n^{RRT*} : cost of the best path in the RRT***
- **c^* : cost of an optimal solution**
- **M_n^{RRT} : # of steps executed by RRT at iteration n**
- **M_n^{RRT*}: # of steps executed by RRT* at iteration n**

From DH's homepage

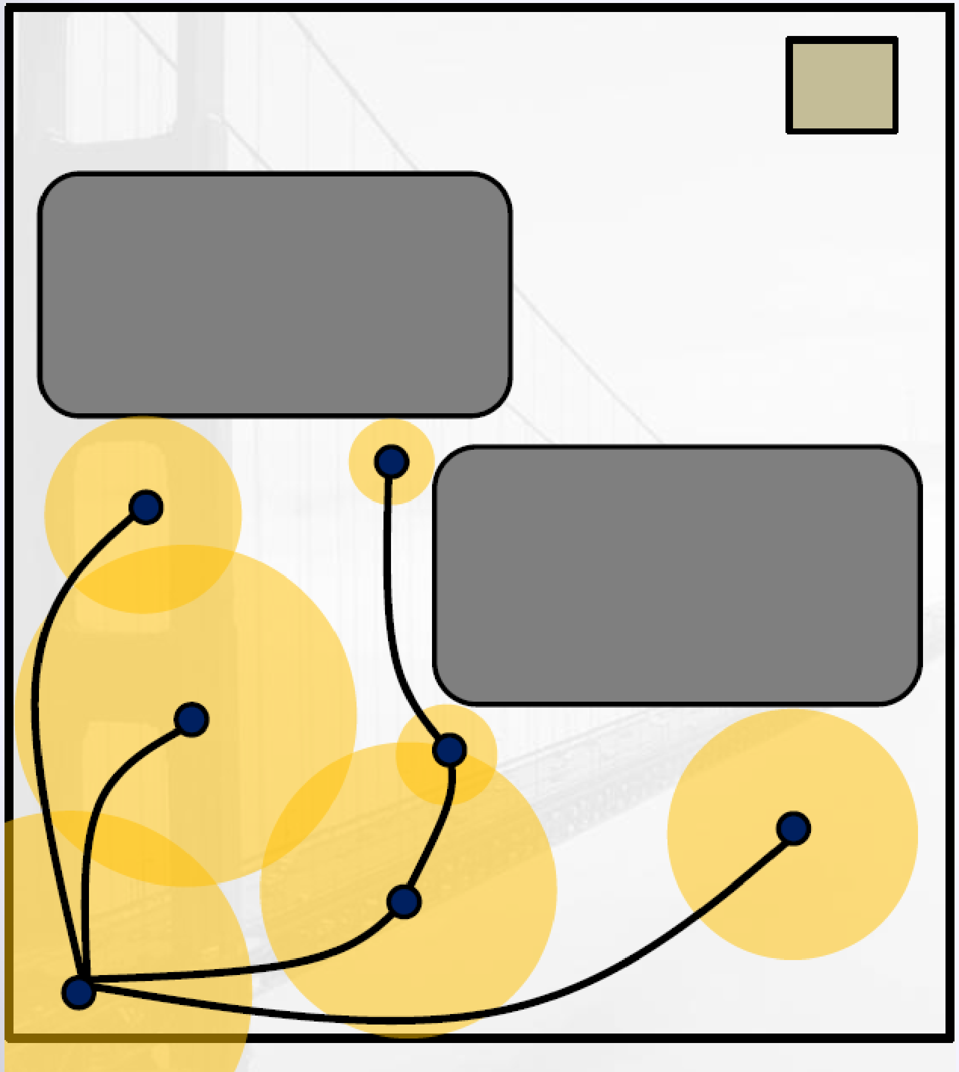
KAIST

Key Operation of RRT*

- RRT
  - Just connect a new node to its nearest neighbor node
- RRT*: refine the connection with re-wiring operation
  - Given a ball, identify neighbor nodes to the new node
  - Refine the connection to have a lower cost

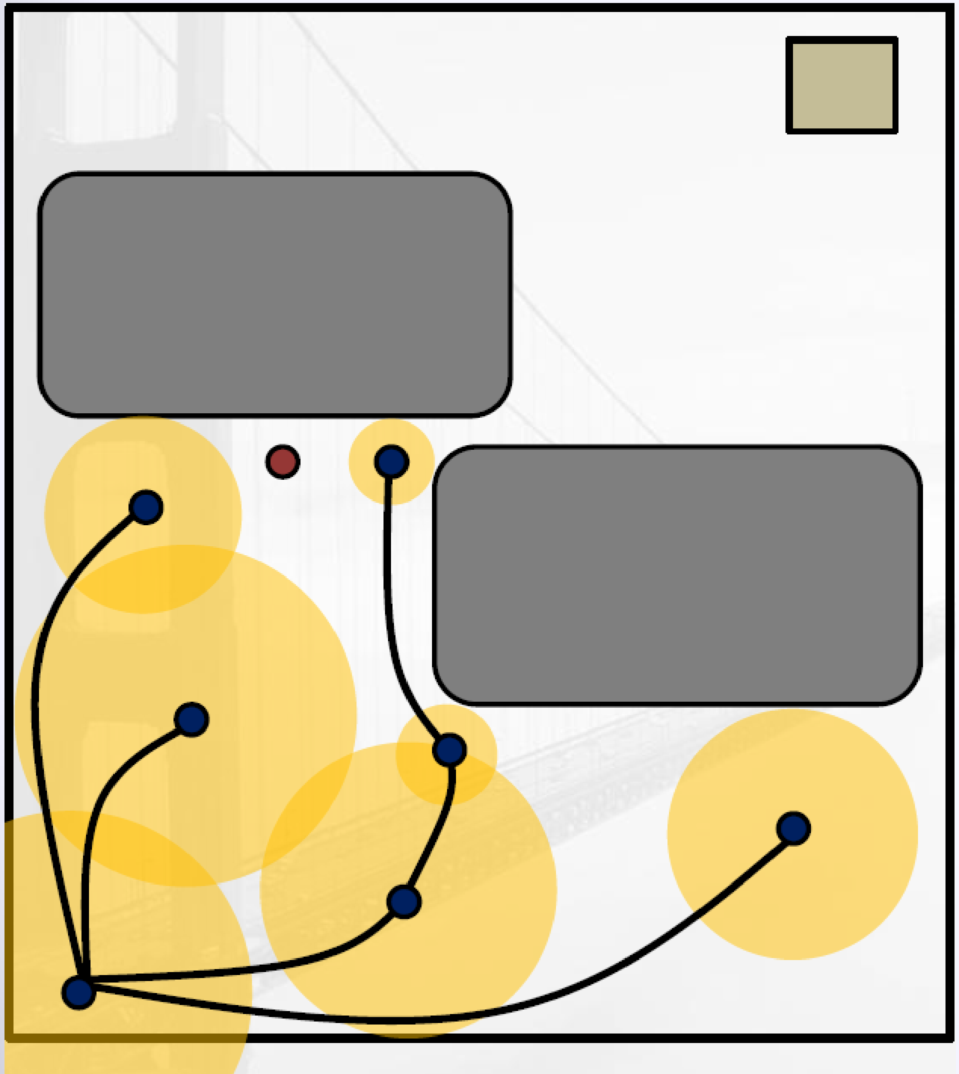
KAIST

Example: Re-Wiring Operation



From ball tree paper

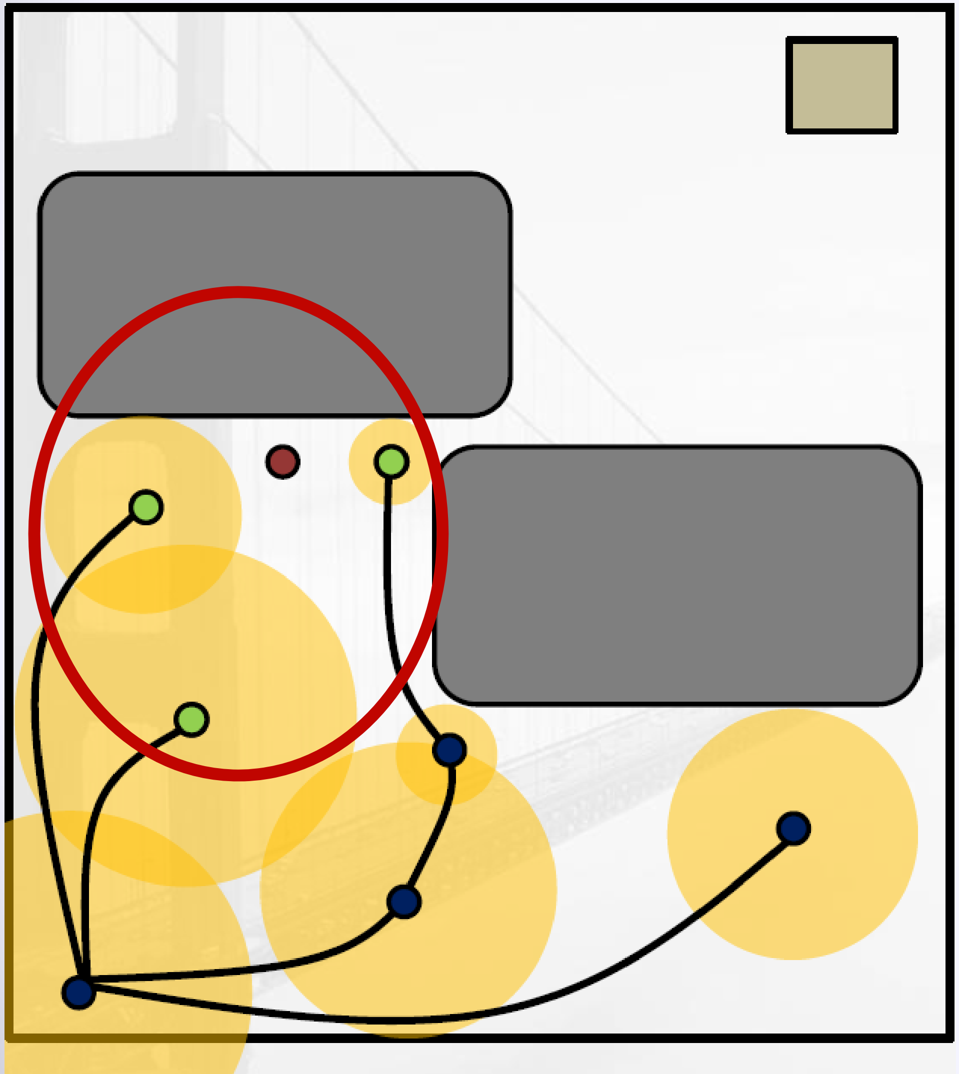
Example: Re-Wiring Operation



Generate a new sample

From ball tree paper

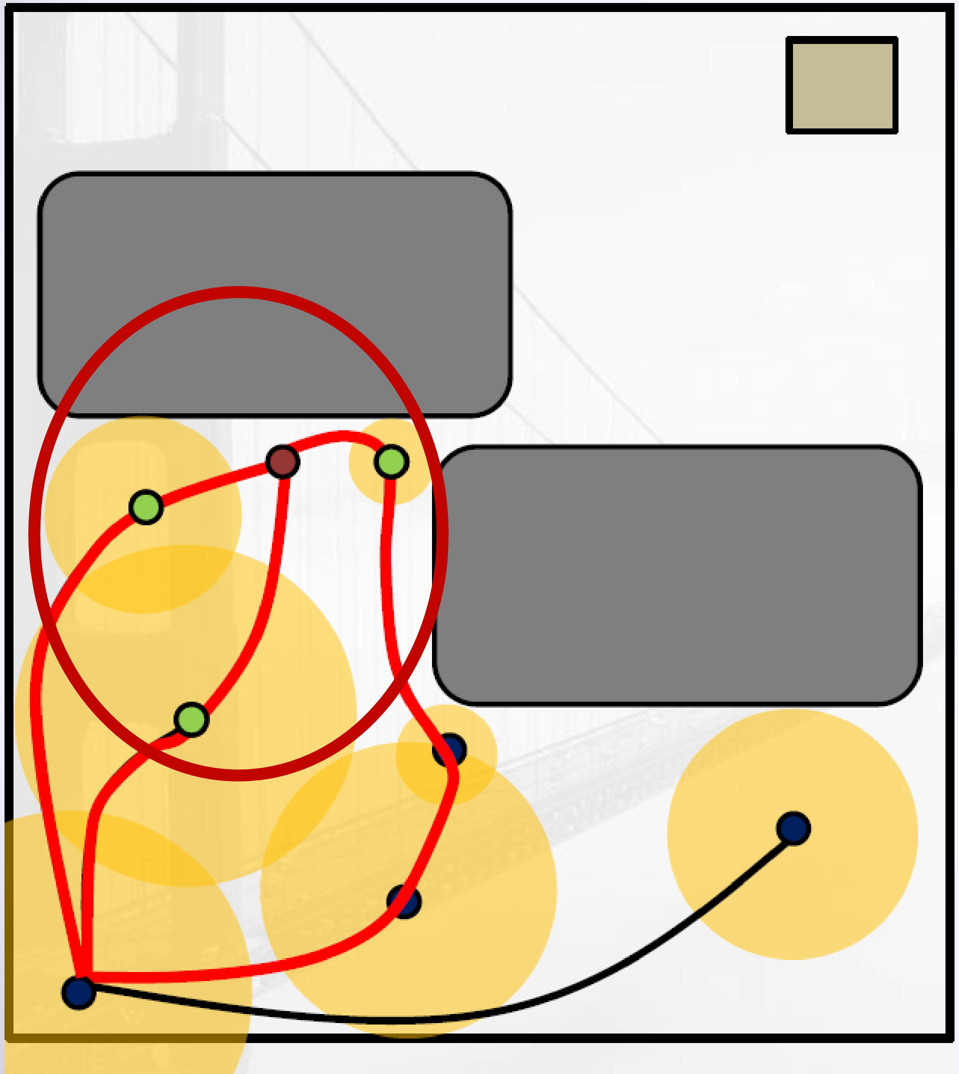
Example: Re-Wiring Operation



Identify nodes in a ball

From ball tree paper

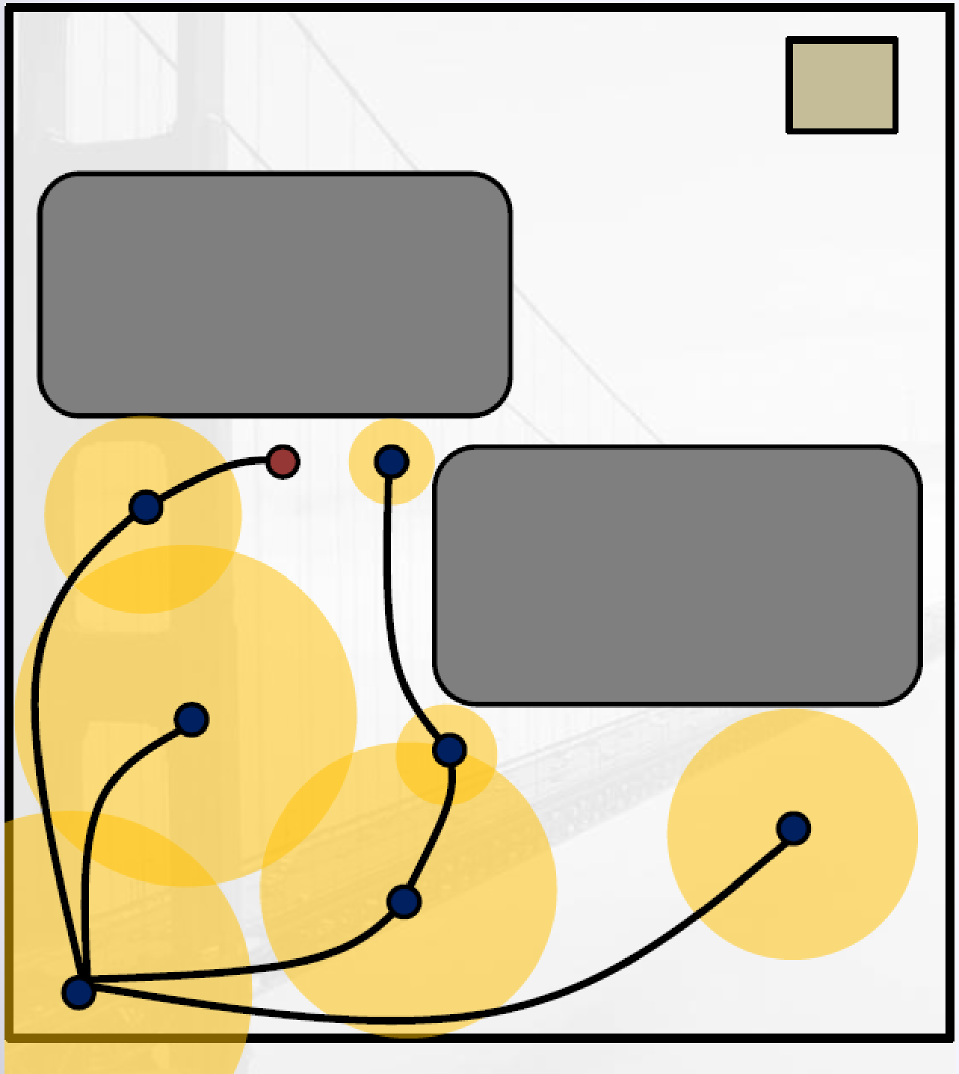
Example: Re-Wiring Operation



Identify which parent gives the lowest cost

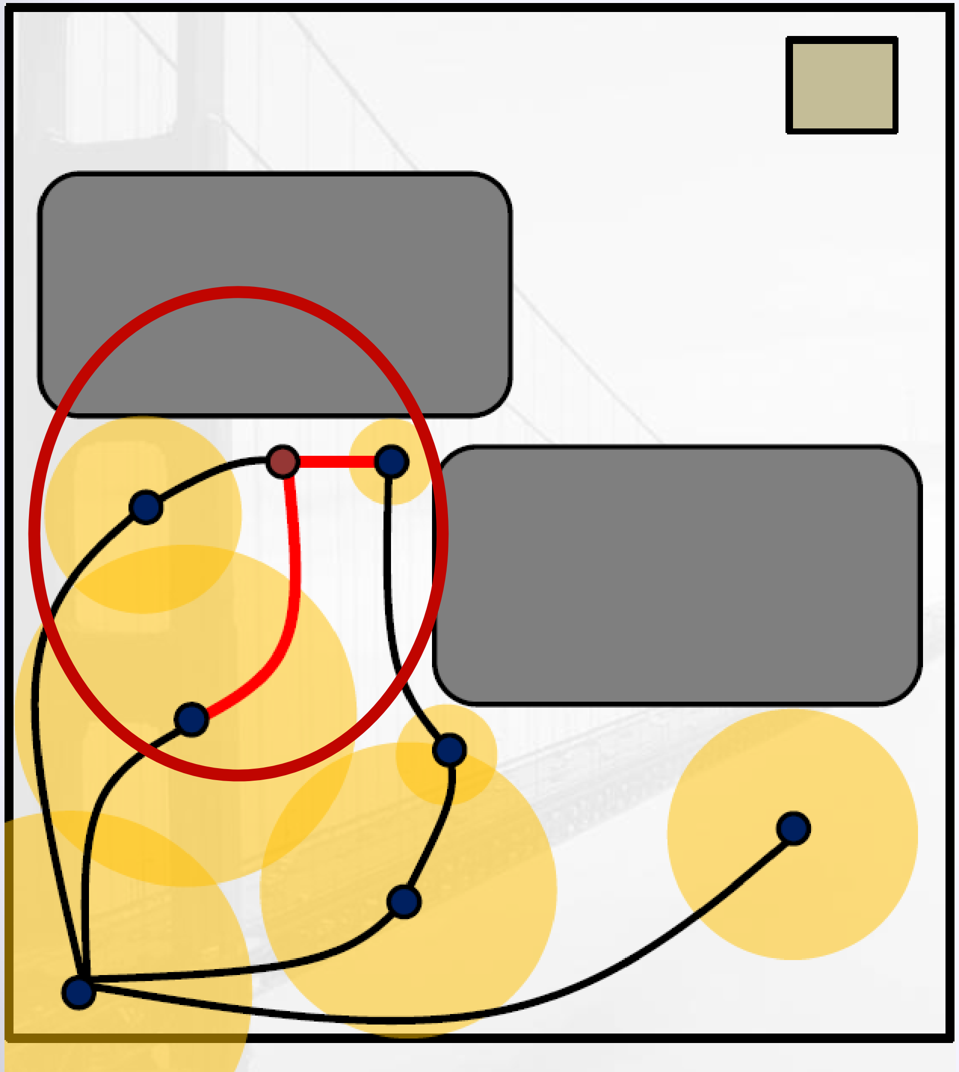
From ball tree paper

Example: Re-Wiring Operation



From ball tree paper

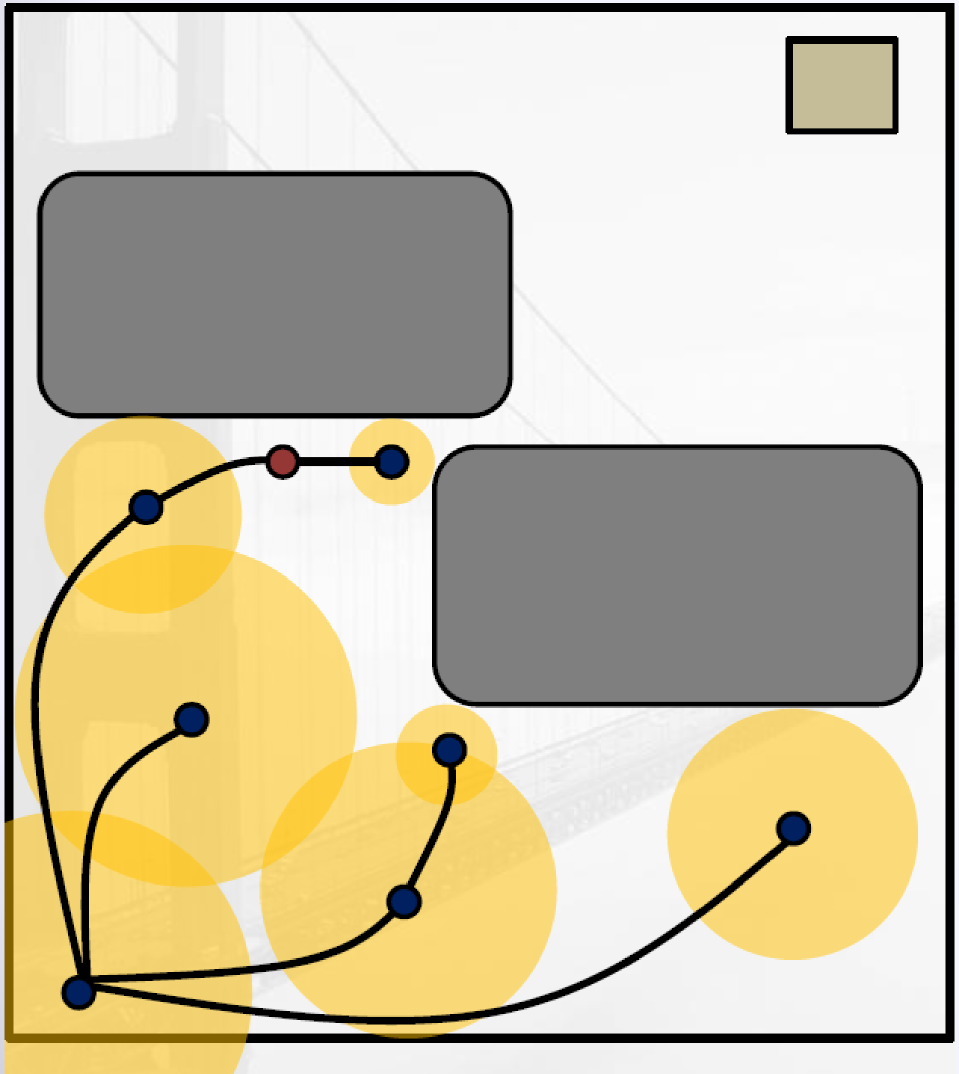
Example: Re-Wiring Operation



Identify which child gives the lowest cost

From ball tree paper

Example: Re-Wiring Operation



Video showing benefits with real robot

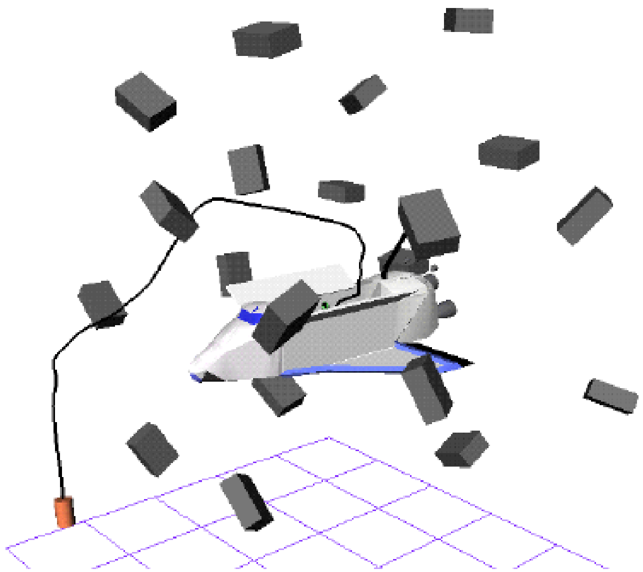
From ball tree paper

Kinodynamic Path Planning

ALSO GIVEN: $h_i(q, \dot{q}, \ddot{q}) \leq 0,\ h_i(q, \dot{q}, \ddot{q}) = 0,\ \ldots$

FIND: τ that satisfies $f_i(q),\ g_i(q, \dot{q}),\ h_i(q, \dot{q}, \ddot{q})$

- **Consider kinematic + dynamic constraints**



State Space Formulation

- **Kinodynamic planning → 2n-dimensional state space**

C denote the C-space

X denote the state space

$$x = (q, \dot{q}), \text{ for } q \in C, x \in X$$

$$x = [q_1 \;\; q_2 \;\; \dots \;\; q_n \;\; \frac{dq_1}{dt} \;\; \frac{dq_2}{dt} \;\; \dots \;\; \frac{dq_n}{dt}]$$

KAIST

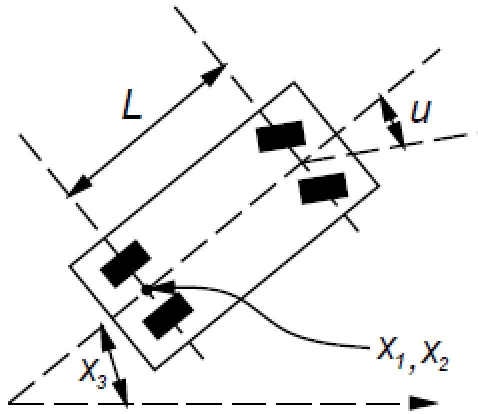
Constraints in State Space

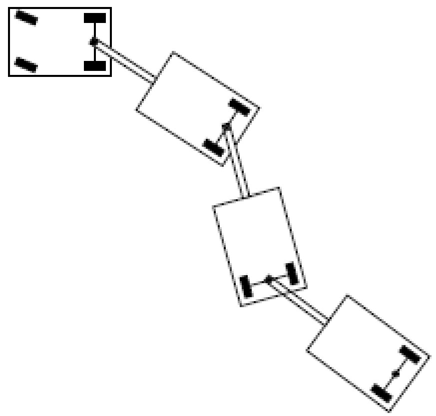
$$h_i(q,\dot{q},\ddot{q}) = 0 \text{ becomes } G_i(x,\dot{x}) = 0,$$
$$\text{for } i = 1,\ldots,m \text{ and } m < 2n$$

- **Constraints can be written in:**

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

$$u \in U, \quad U: \text{Set of allowable controls or inputs}$$





Solution Trajectory

- **Defined as a time-parameterized continuous path**

$$\tau : [0, T] \rightarrow X_{free}, \text{ satisfies the constraints}$$

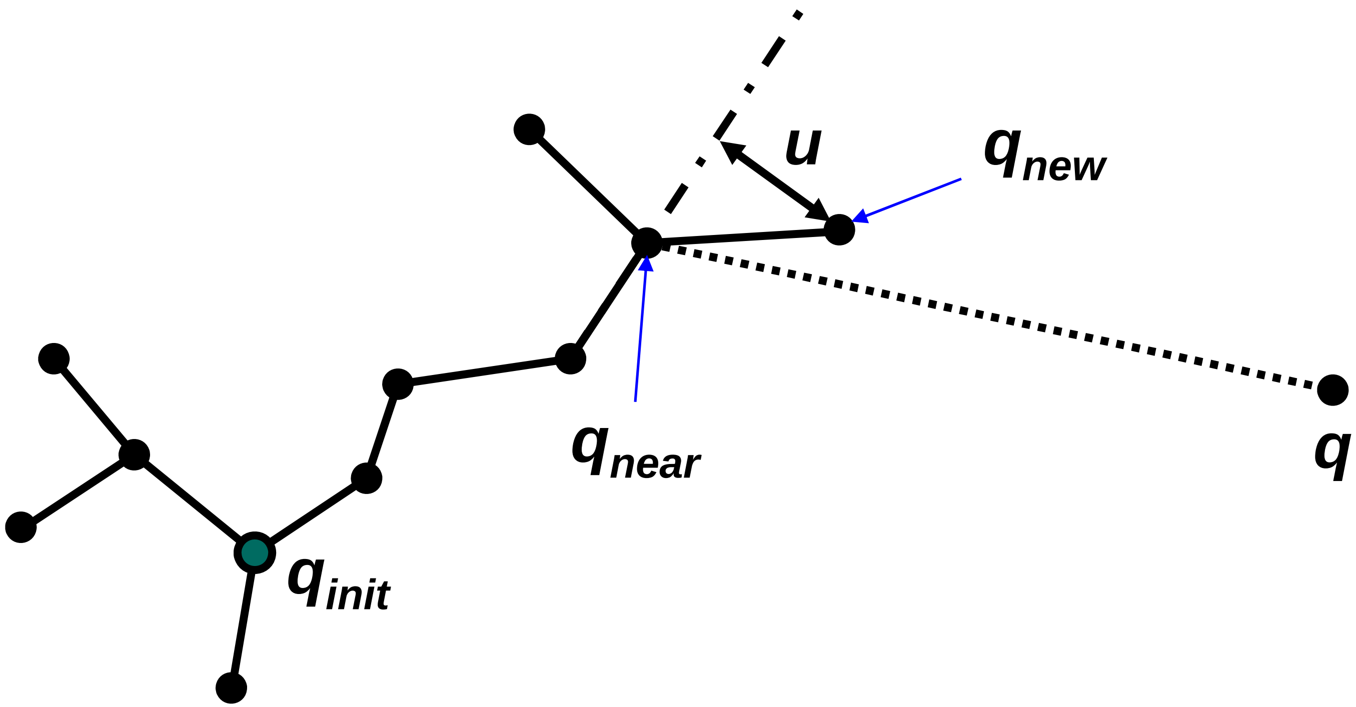
- **Obtained by integrating** $\dot{x} = f(x, u)$
- **Solution: Finding a control function**

$$u : [0, T] \rightarrow U$$

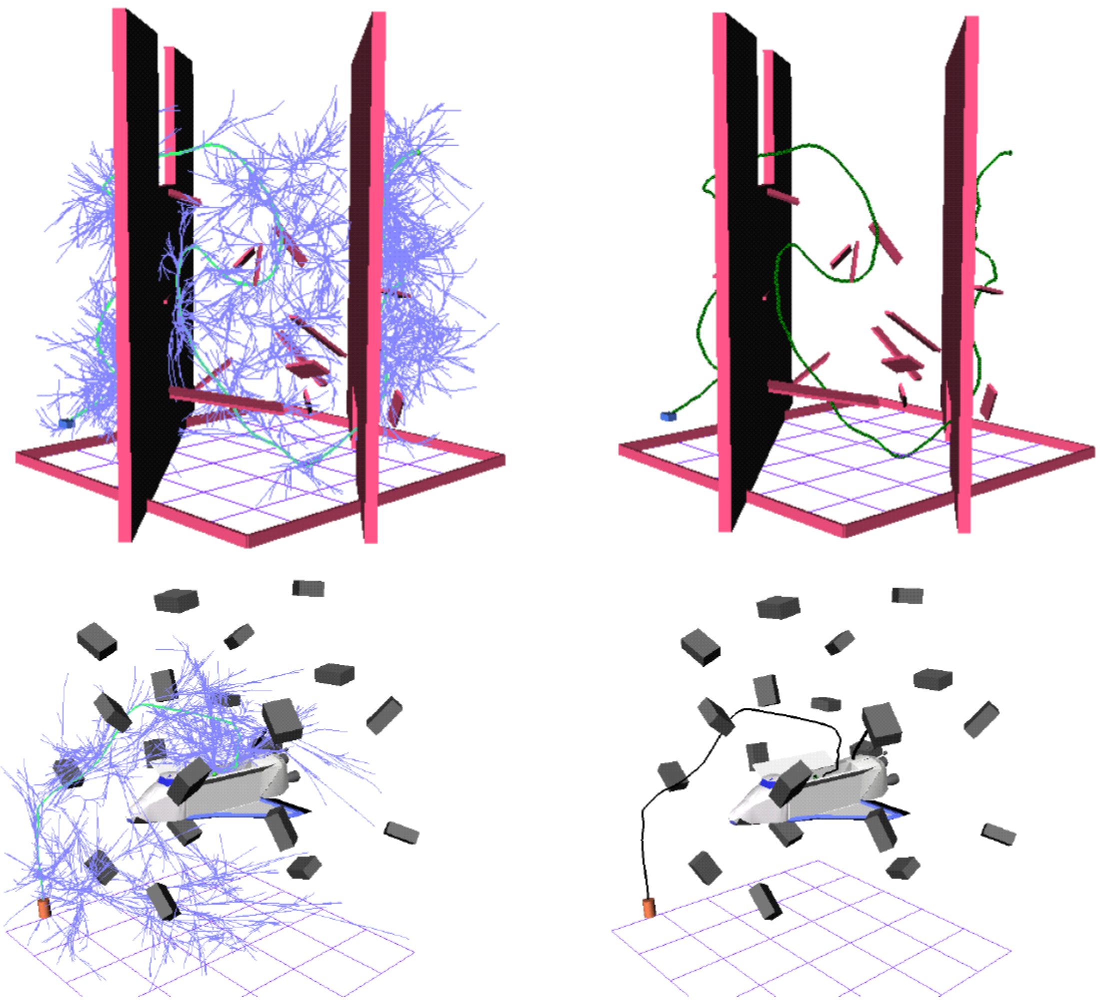
KAIST

Rapidly-Exploring Random Tree

- **Extend a new vertex in each iteration**



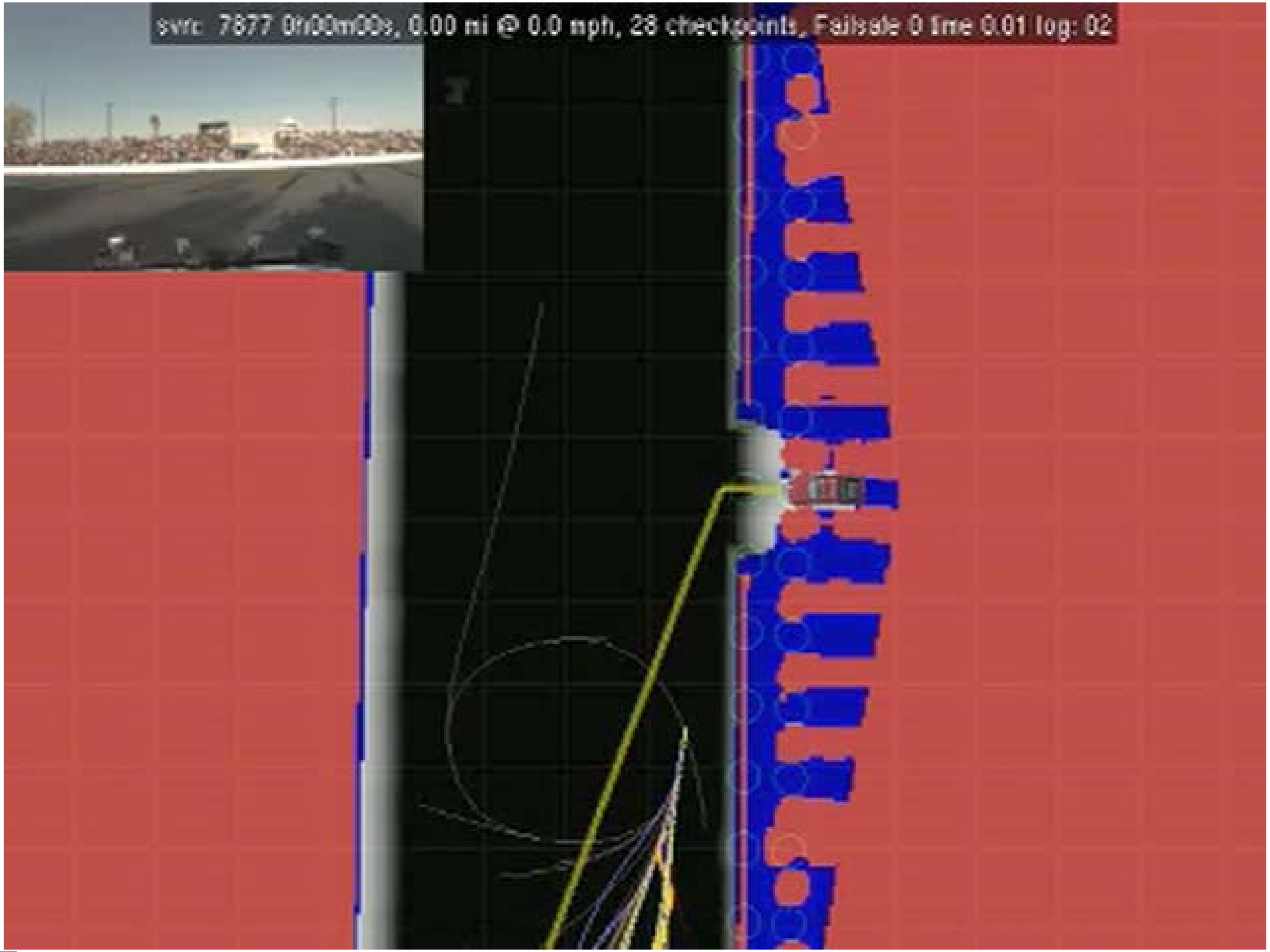
Results – 200MHz, 128MB



- 3D translating
- X=6 DOF
- 16,300 nodes
- 4.1min

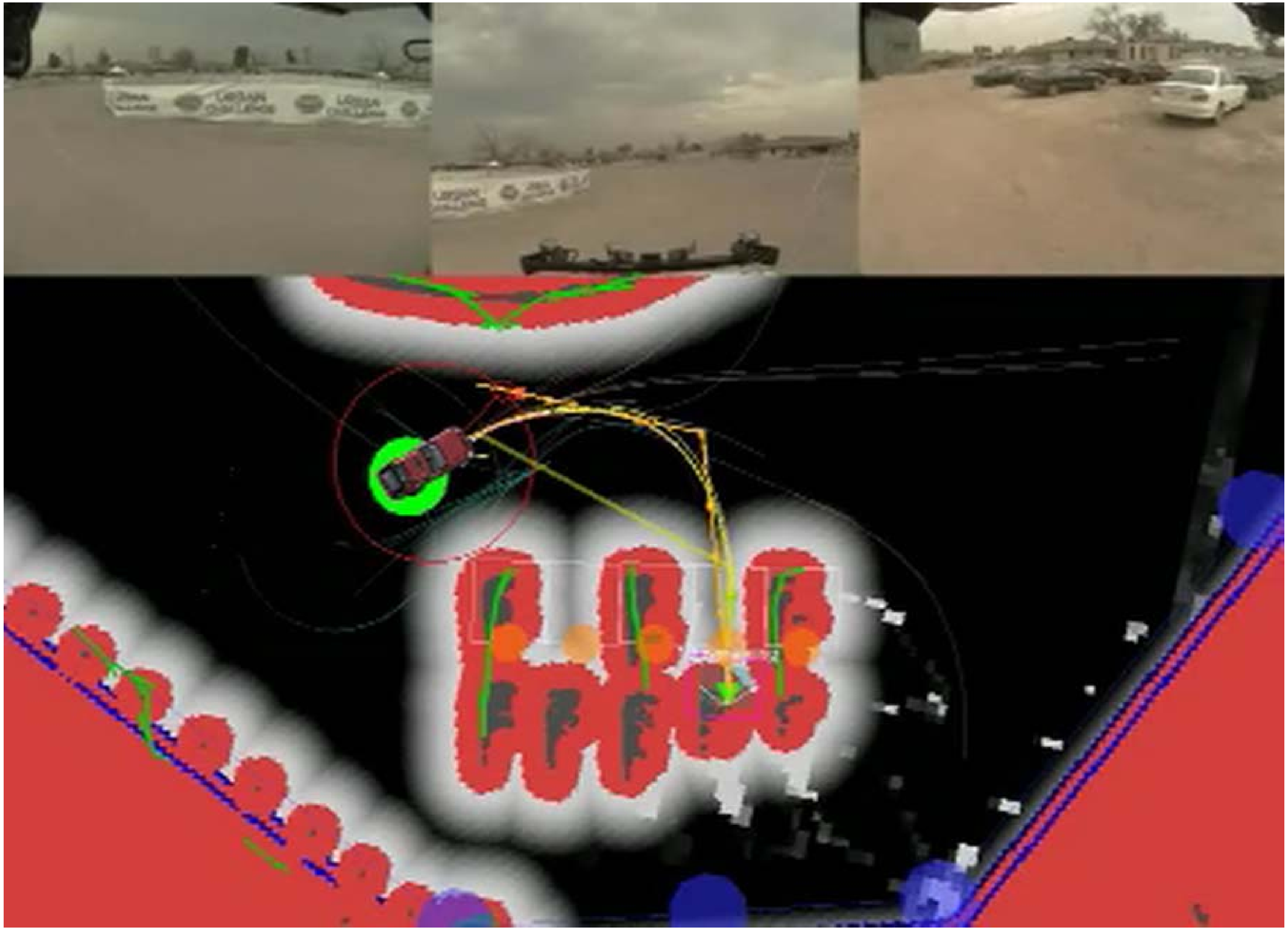
- 3D TR+RO
- X=12 DOF
- 23,800 nodes
- 8.4min

RRT at work: Urban Challenge

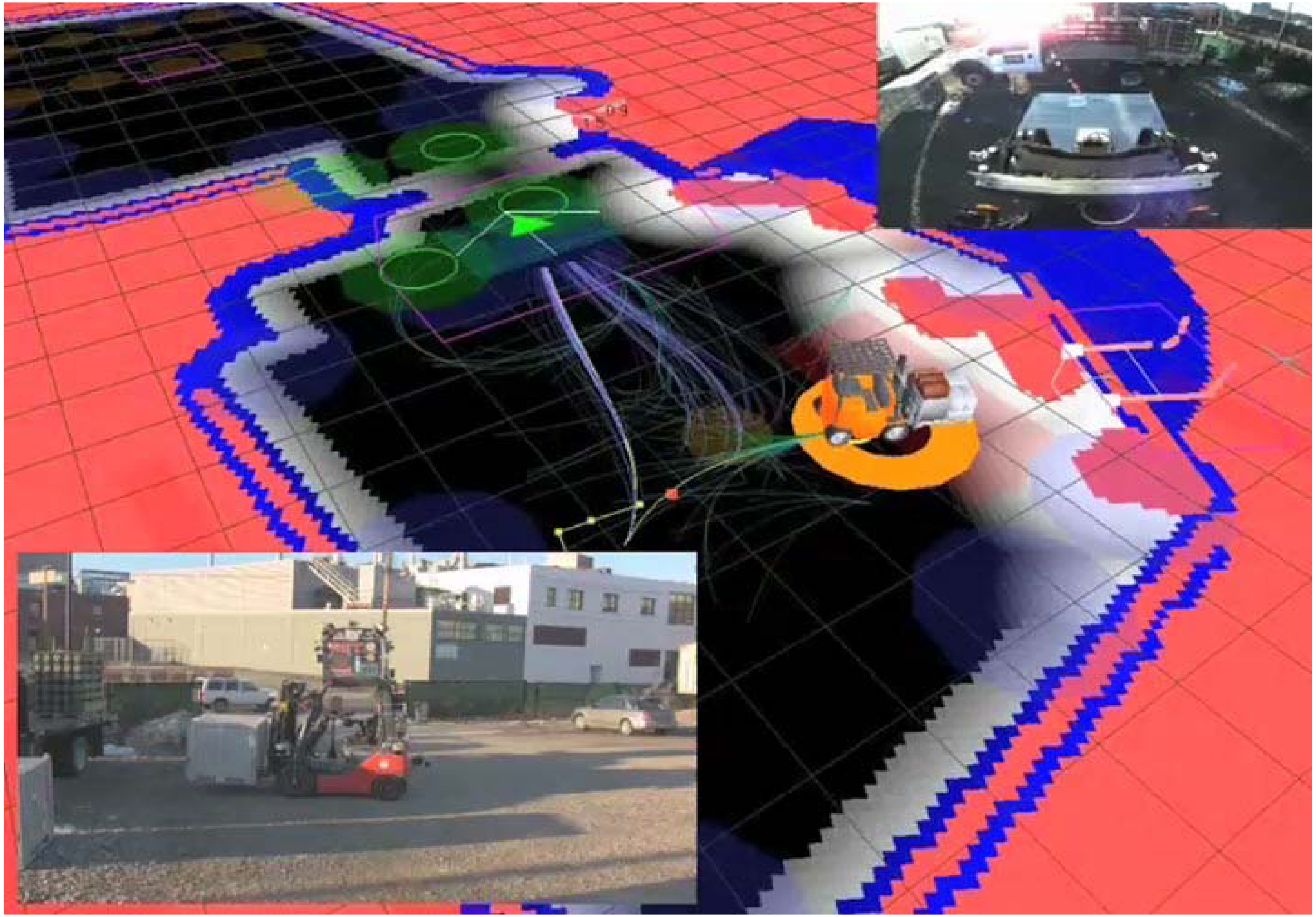


From MIT

Successful Parking Maneuver



RRT at work: Autonomous Forklift



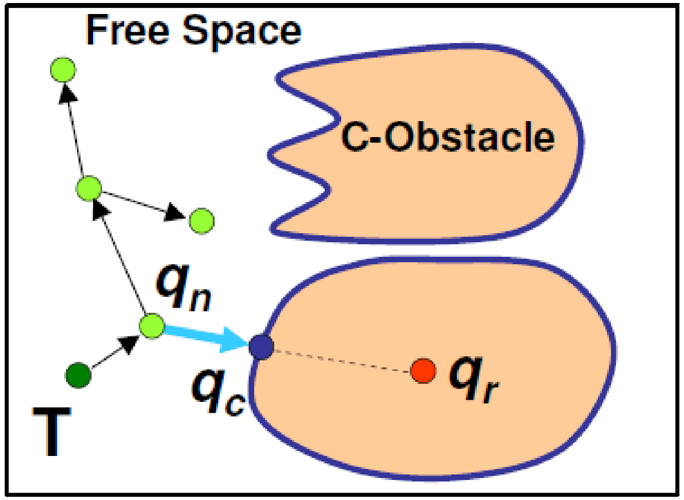
Recent Works of Our Group

- **Narrow passages**
  - **Identify narrow passage with a simple one-dimensional line test, and selectively explore such regions**
  - **Selective retraction-based RRT planner for various environments, Lee et al., T-RO 14**
  - **http://sglab.kaist.ac.kr/SRRRT/T-RO.html**

KAIST

Retration-based RRT [Zhang & Manocha 08]

- **Retraction-based RRT technique handling narrow passages**



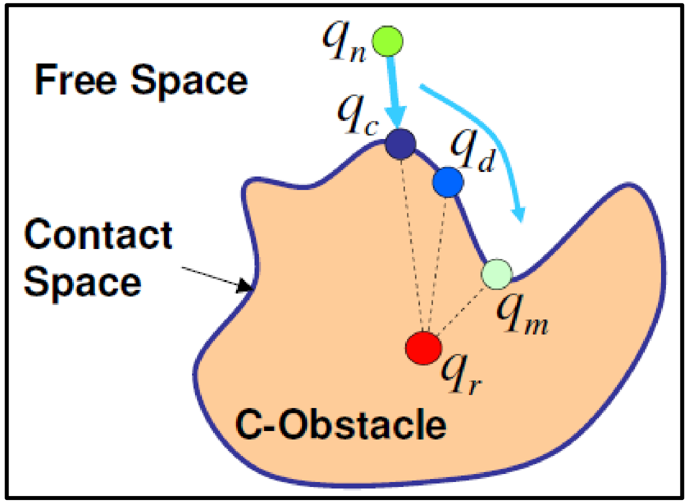
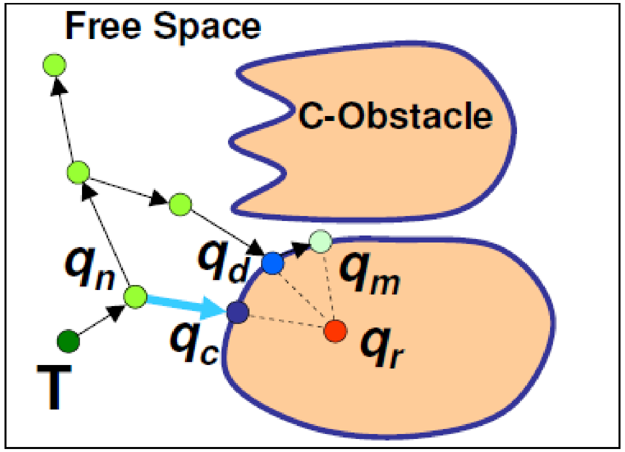


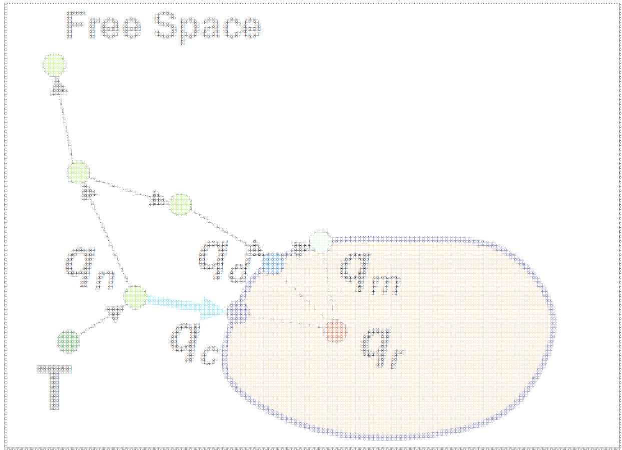
image from [Zhang & Manocha 08]

- **General characteristic**:
  Generates more samples near the boundary of obstacles

RRRT: Pros and Cons

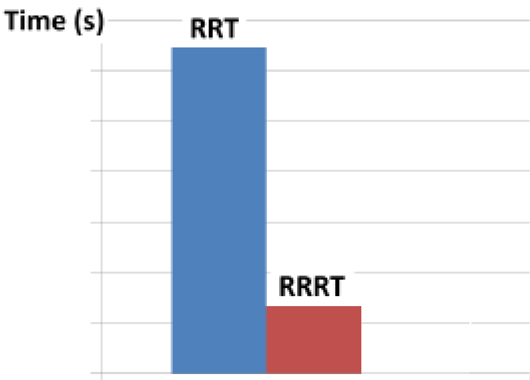


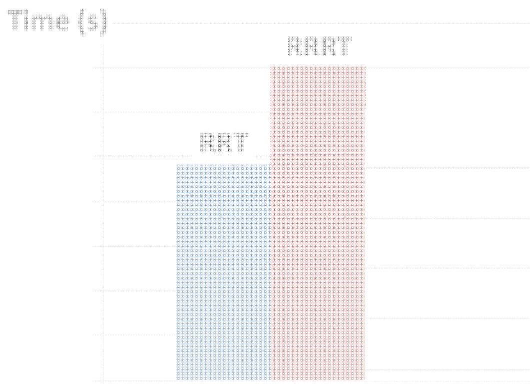
with narrow passages



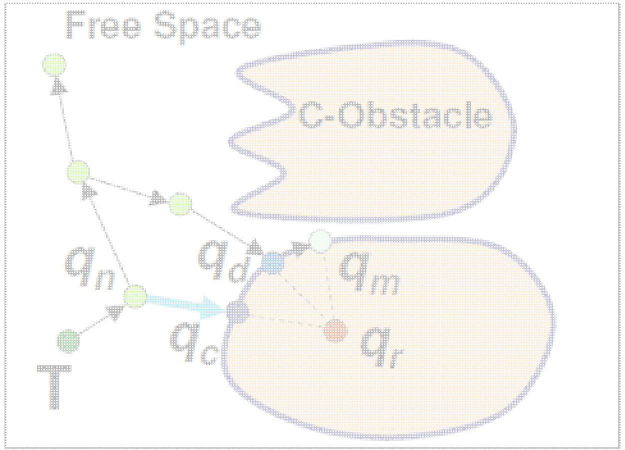
without narrow passages

images from [Zhang & Manocha 08]

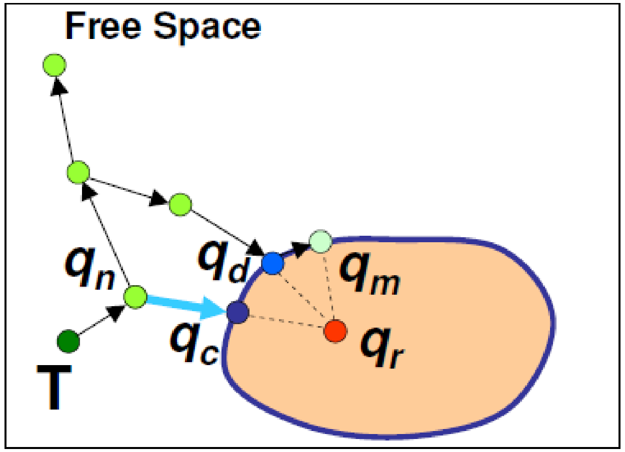




RRRT: Pros and Cons

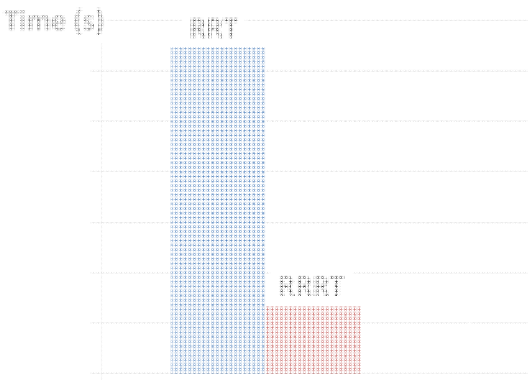


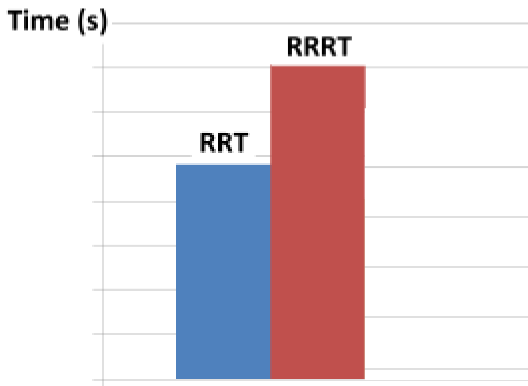
with narrow passages



without narrow passages

images from [Zhang & Manocha 08]

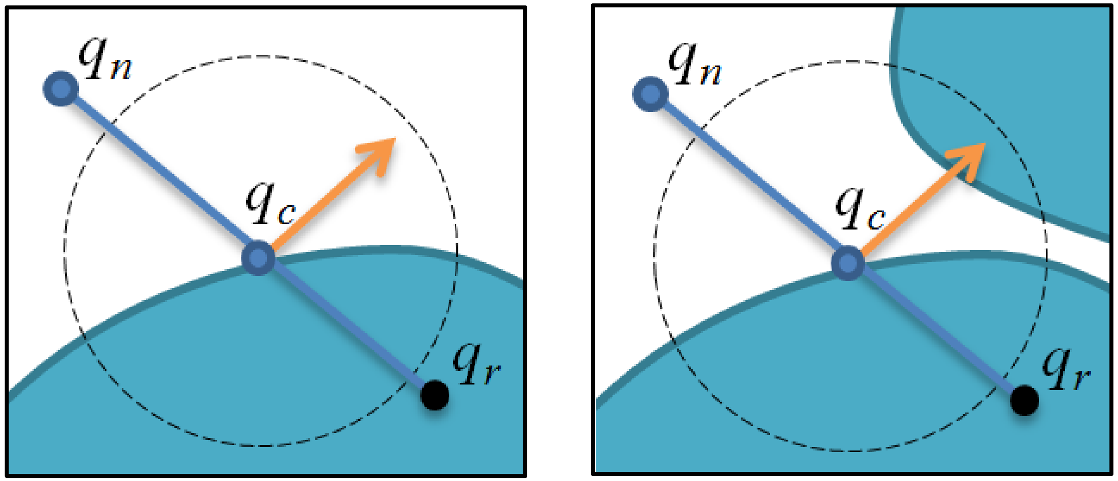




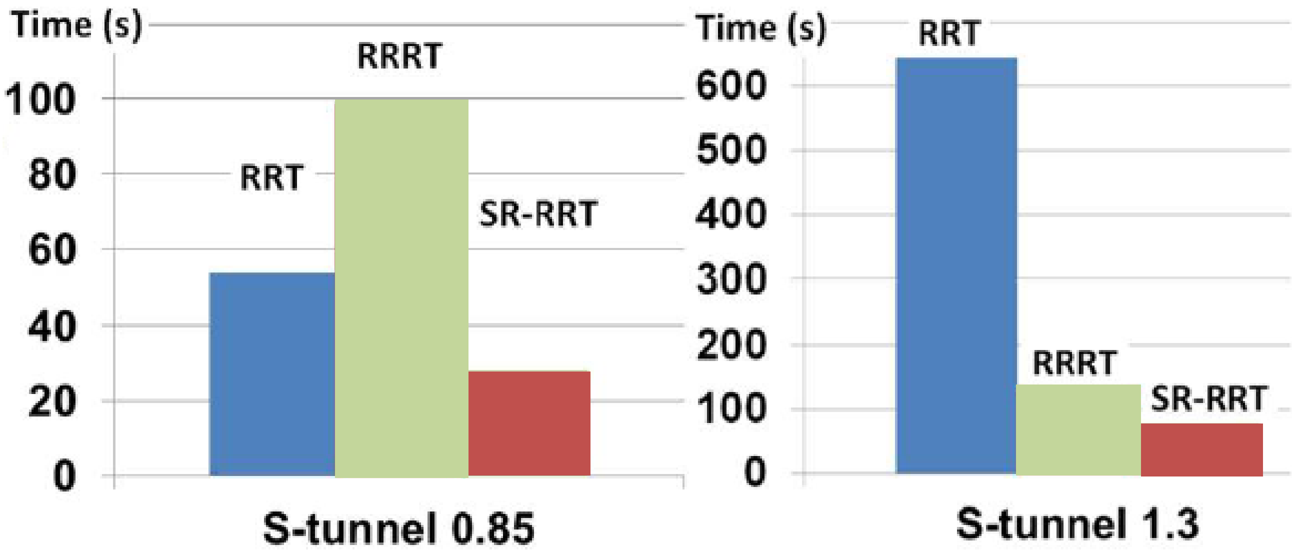
Bridge line-test [Lee et al., T-RO 14]

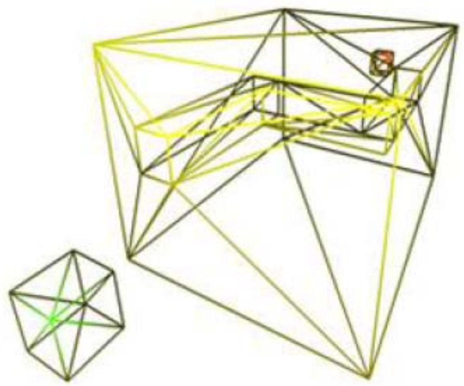
- To identify narrow passage regions

- Bridge line-test
  1. Generate a random line
  2. Check whether the line meets any obstacle



Results





Video

KAIST

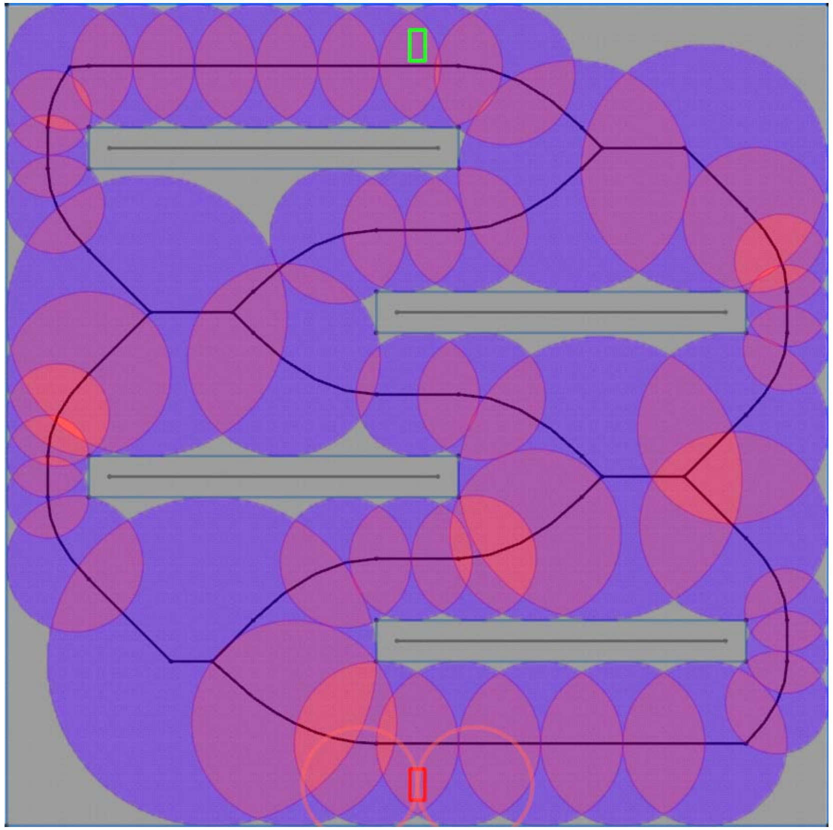
Recent Works of Our Group

- Handling narrow passages

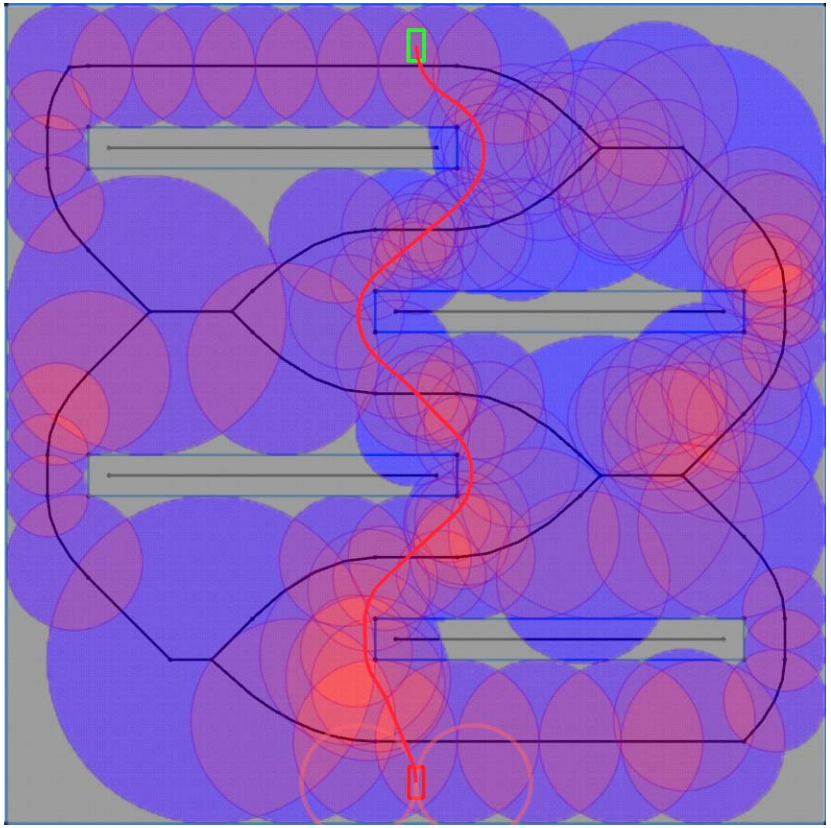
- Improving low convergence to the optimal solution
  - Use the sampling cloud to indicate regions that lead to the optimal path
  - Cloud RRT* : Sampling Cloud based RRT*, Kim et al., ICRA 14
  - http://sglab.kaist.ac.kr/CloudRRT/

KAIST

Examples of Sampling Cloud [Kim et al., ICRA 14]



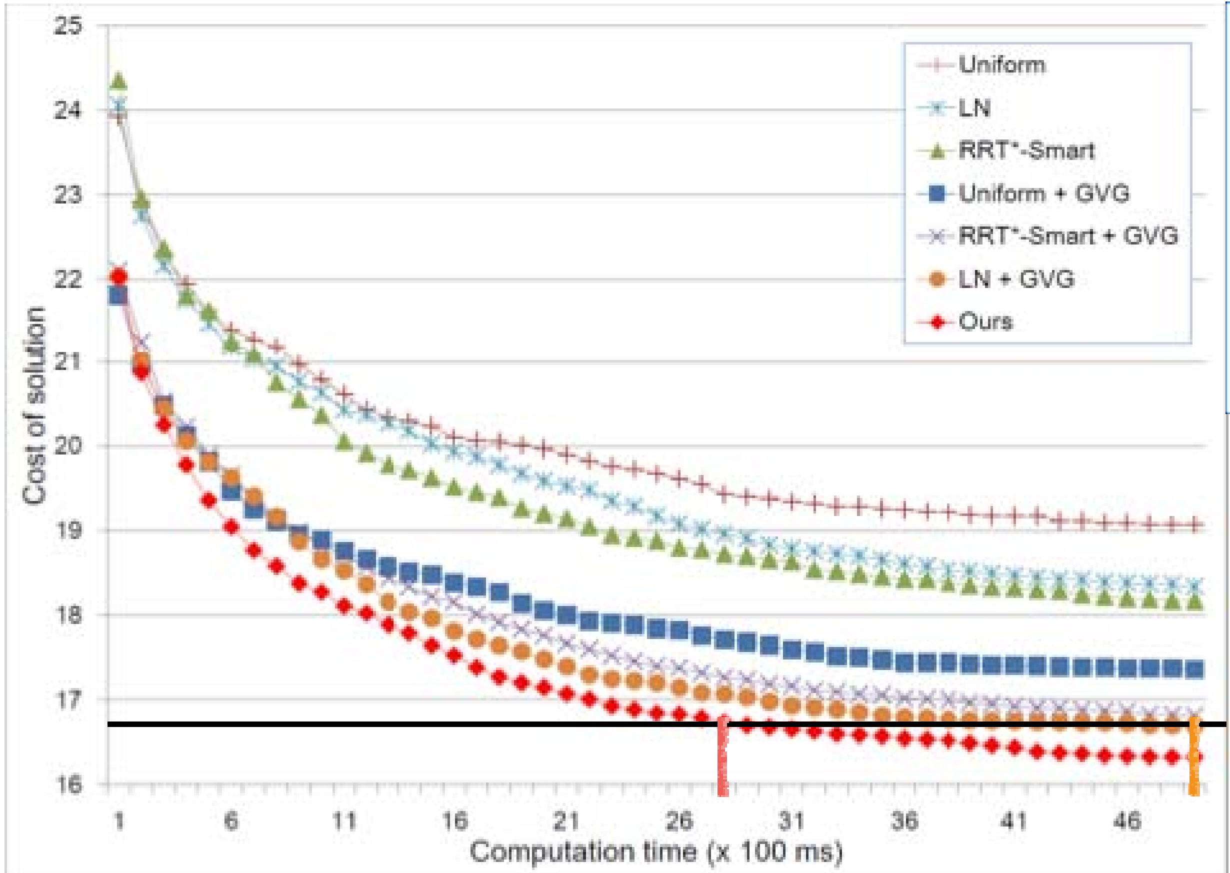
Initial state of sampling cloud

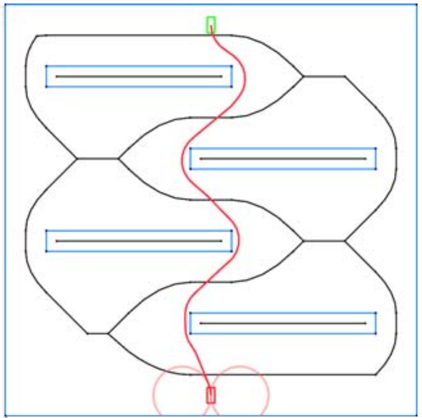


After updated several times

Video

Results: 4 squares





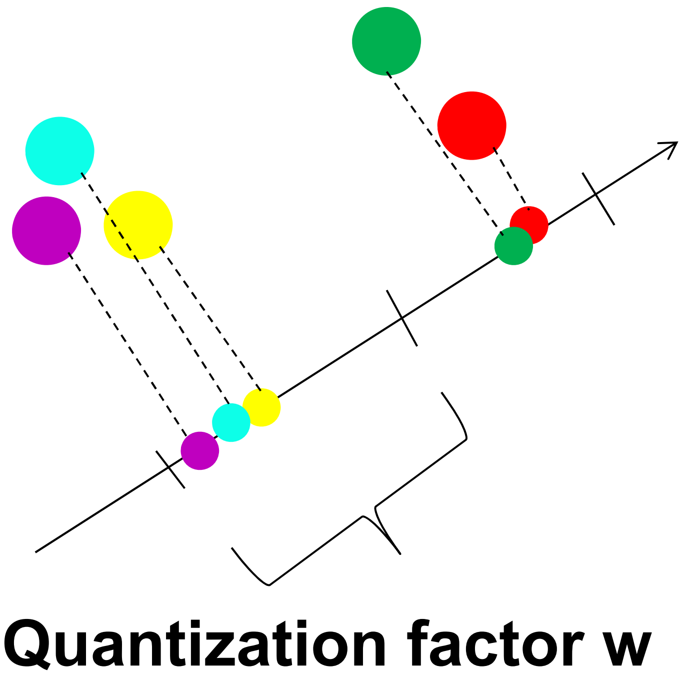
1.8X improvement

Recent Works of Our Group

- Handling narrow passages
- Improving low convergence to the optimal solution
- Accelerating nearest neighbor search
  - VLSH: Voronoi-based Locality Sensitive Hashing, Loi et al., IROS 13

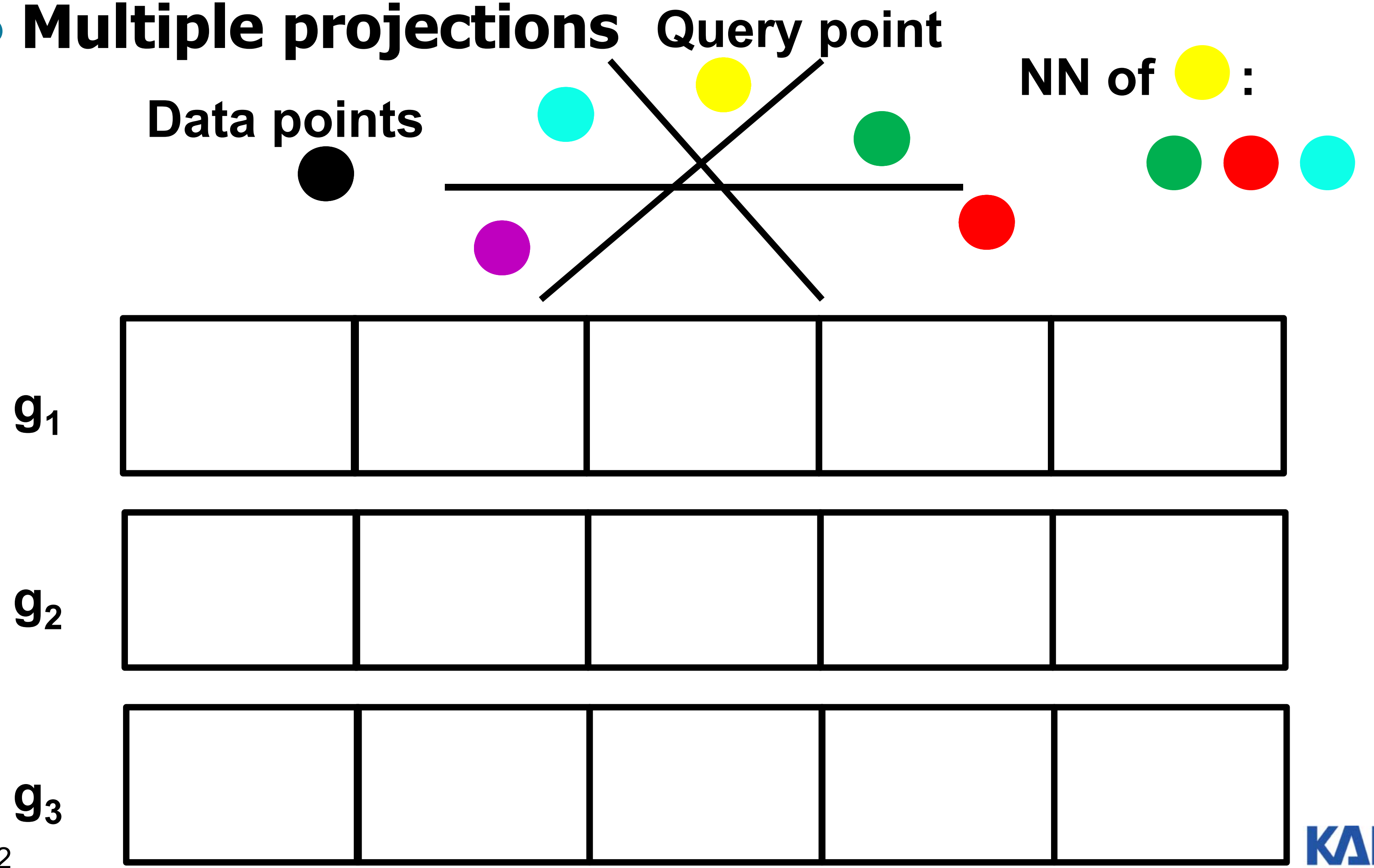
Background on Locality Sensitive Hashing (LSH)

- **Randomly generate a projection vector**
- **Project points onto vector**
- **Bin the projected points to a segment, whose width is w, i.e. quantization factor**
- **All the data in a bin has the same hash code**

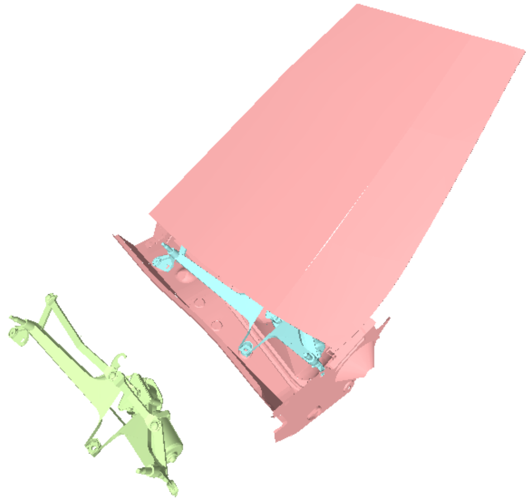


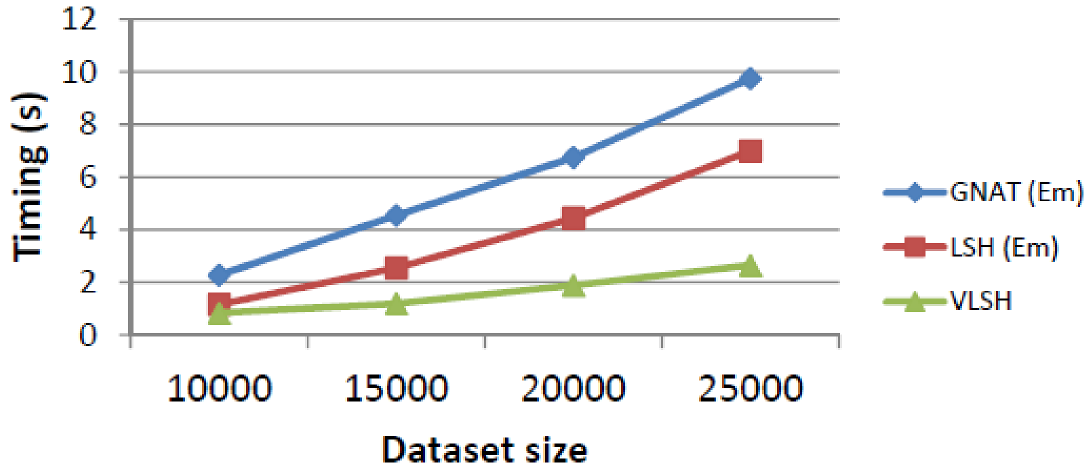
Background on LSH

- **Multiple projections**



Wiper: Performance Evaluation



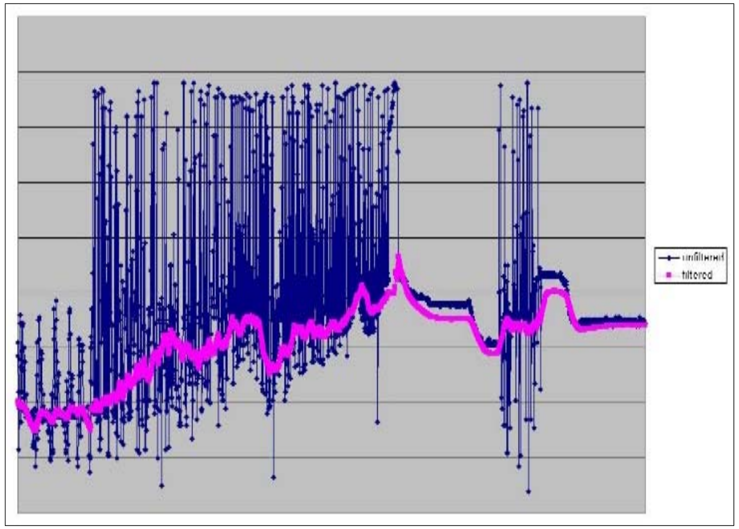


- **VLSH vs. GNAT (Em):**
  - **3.7x faster**
- **VLSH vs. LSH (Em):**
  - **2.6x faster**

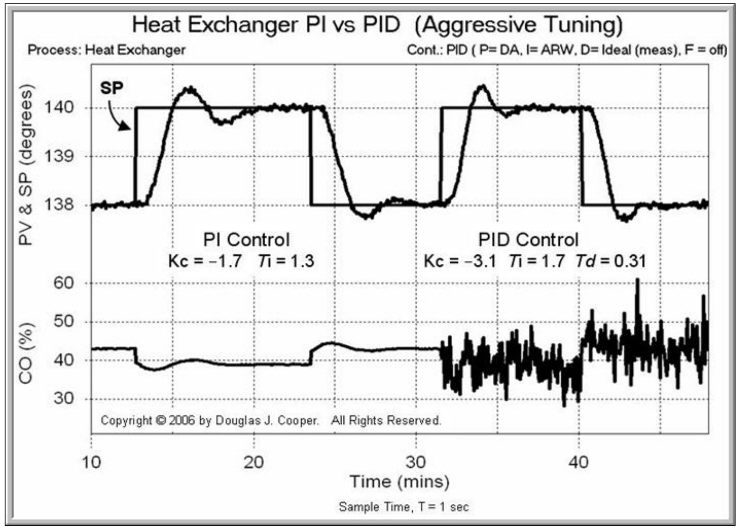
KAIST

Handling Sensor Errors

- **Uncertainty caused by:**
  - **Various sensors**
  - **Low-level controllers**



Sensor noise

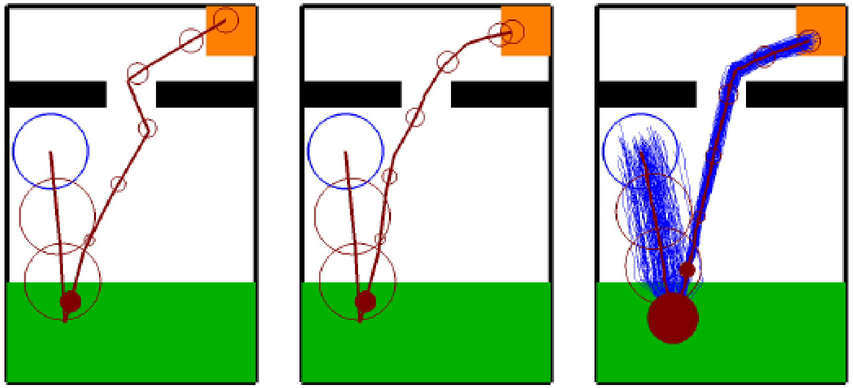


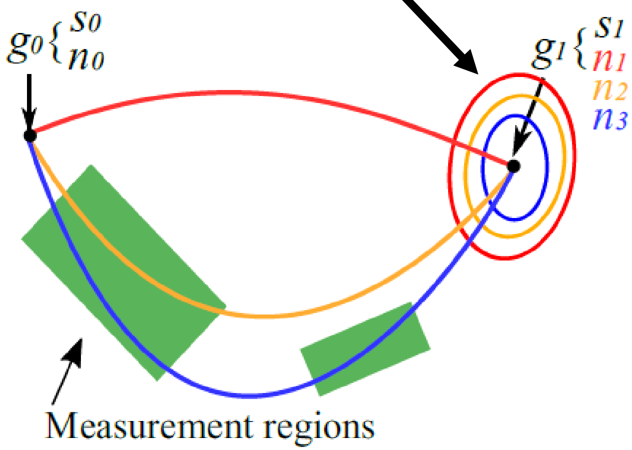
Controller noise

Rapidly-exploring Random Belief Tree
[Bry et al., ICRA 11]

- Use Kalman filter to propagate Gaussian states
- Improve solutions toward optimal

Multiple belief nodes in the same vertex



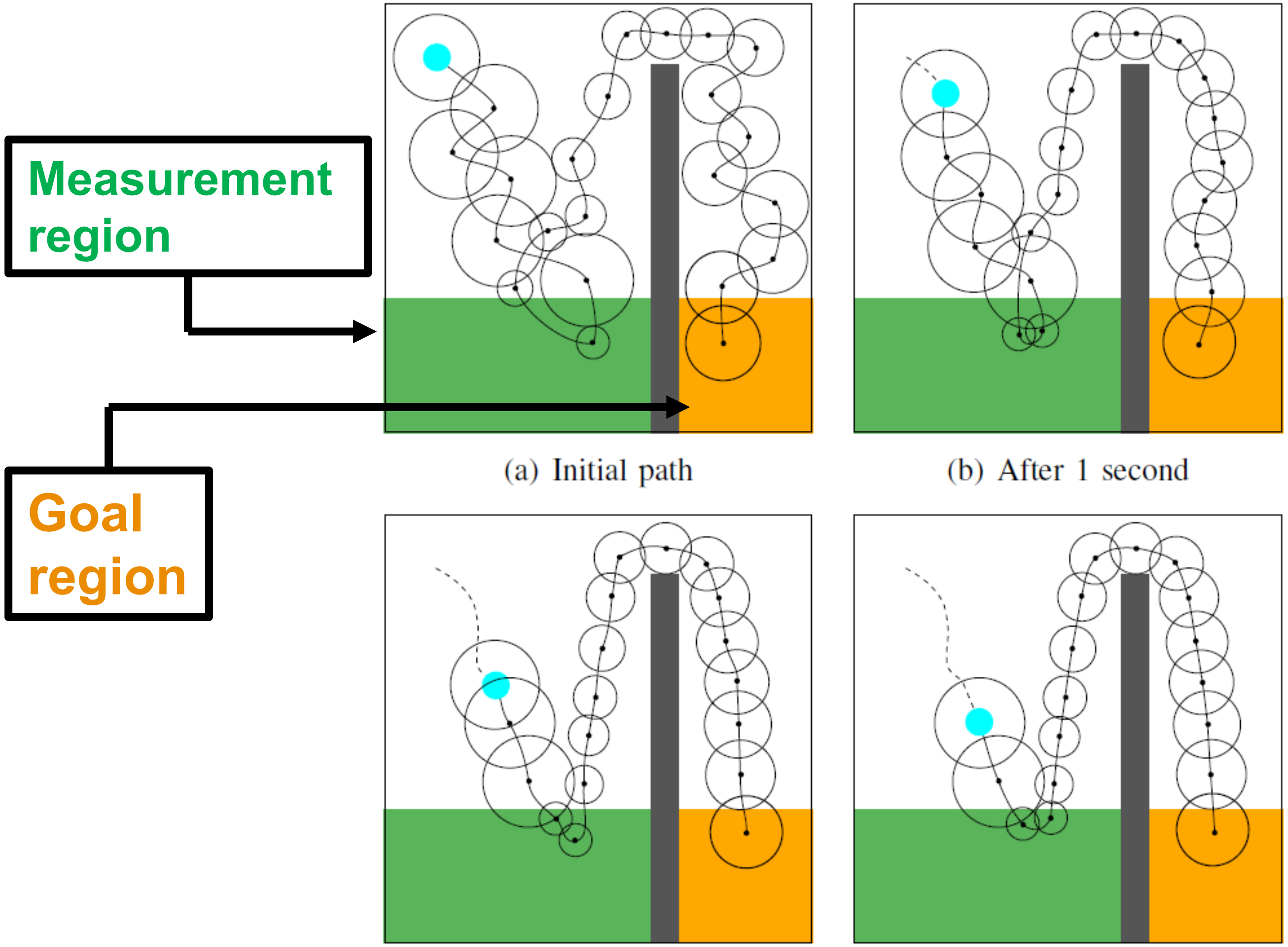


500 **1000** **1500**

Number of iteration

Preserve optimal path

Main Contribution: Anytime Extension [Yang et al.,IROS 16]



(a) Initial path (b) After 1 second

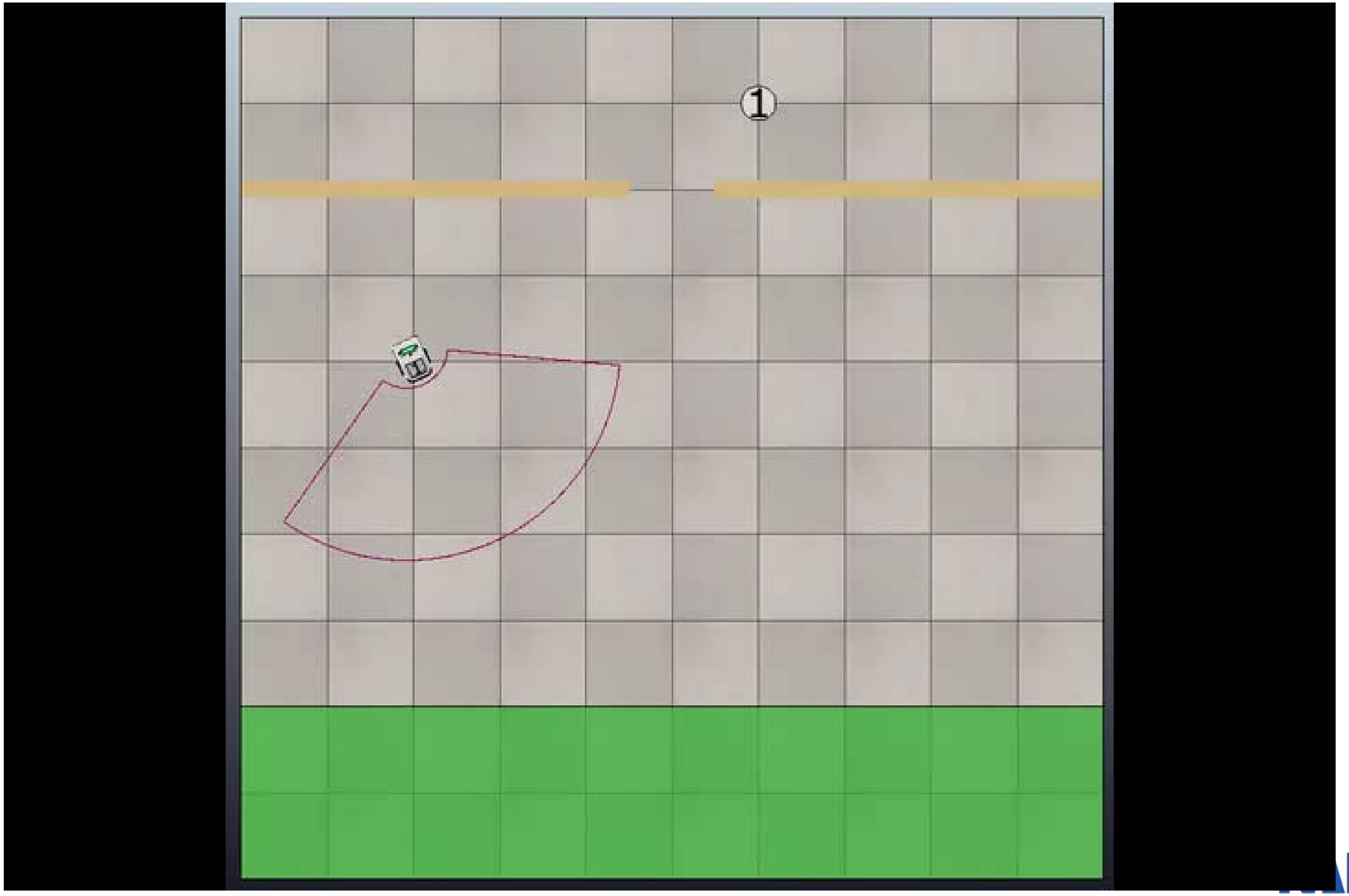
(c) After 3 seconds (d) After 5 seconds

Bigger circle means higher uncertainty

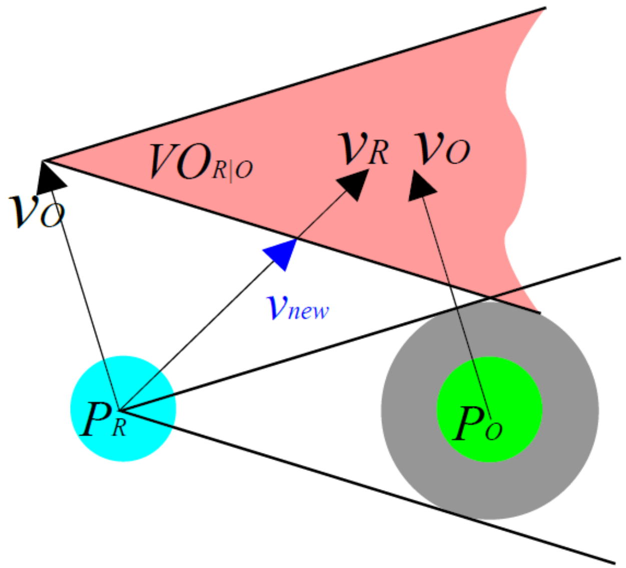
The robot computes better path while executing the path

KAIST

Main Contribution: Anytime Extension



Velocity Obstacle: Local Geometric Analysis

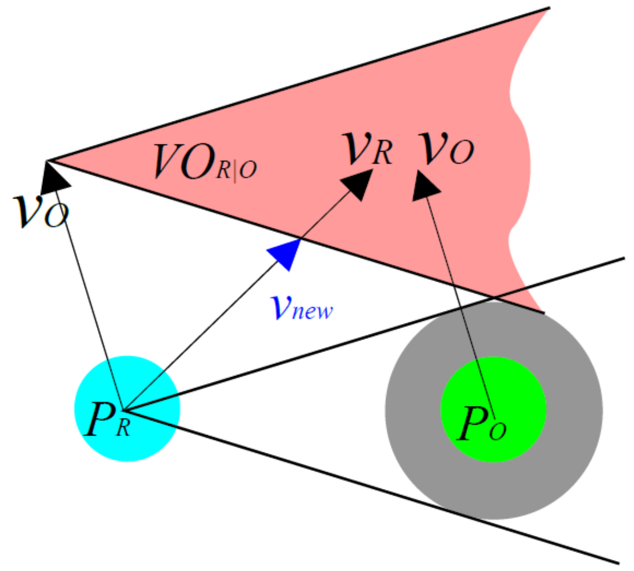


(a) Velocity obstacle

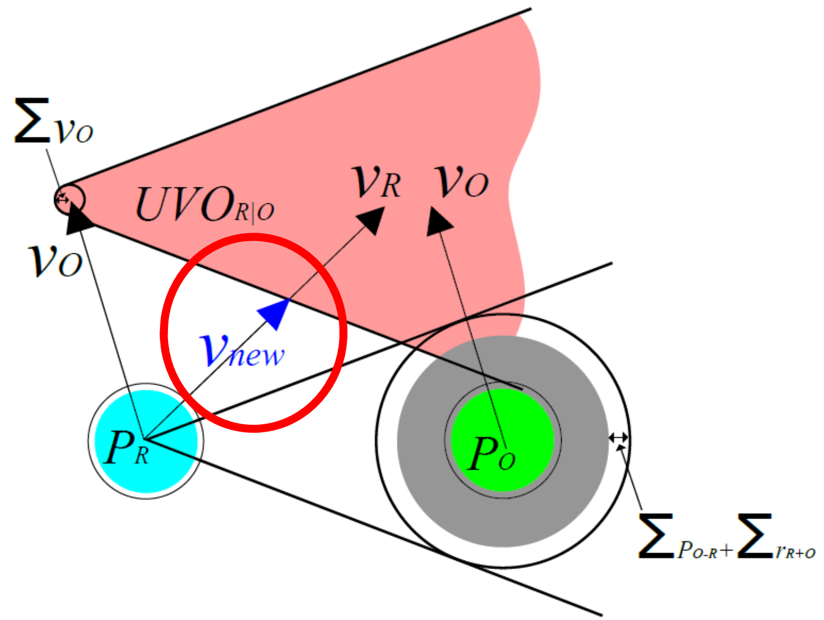
"The hybrid reciprocal velocity obstacle" **TRO11** J Snape, J van den Berg, SJ Guy
"Reciprocal velocity obstacles for real-time multi-agent navigation" J van den Berg
"Generalized Velocity Obstacles" **IROS09**, D Wilkie, J Van den Berg

Uncertainty-aware Velocity Obstacle as Local Geometry Analysis

Conservative collision checking



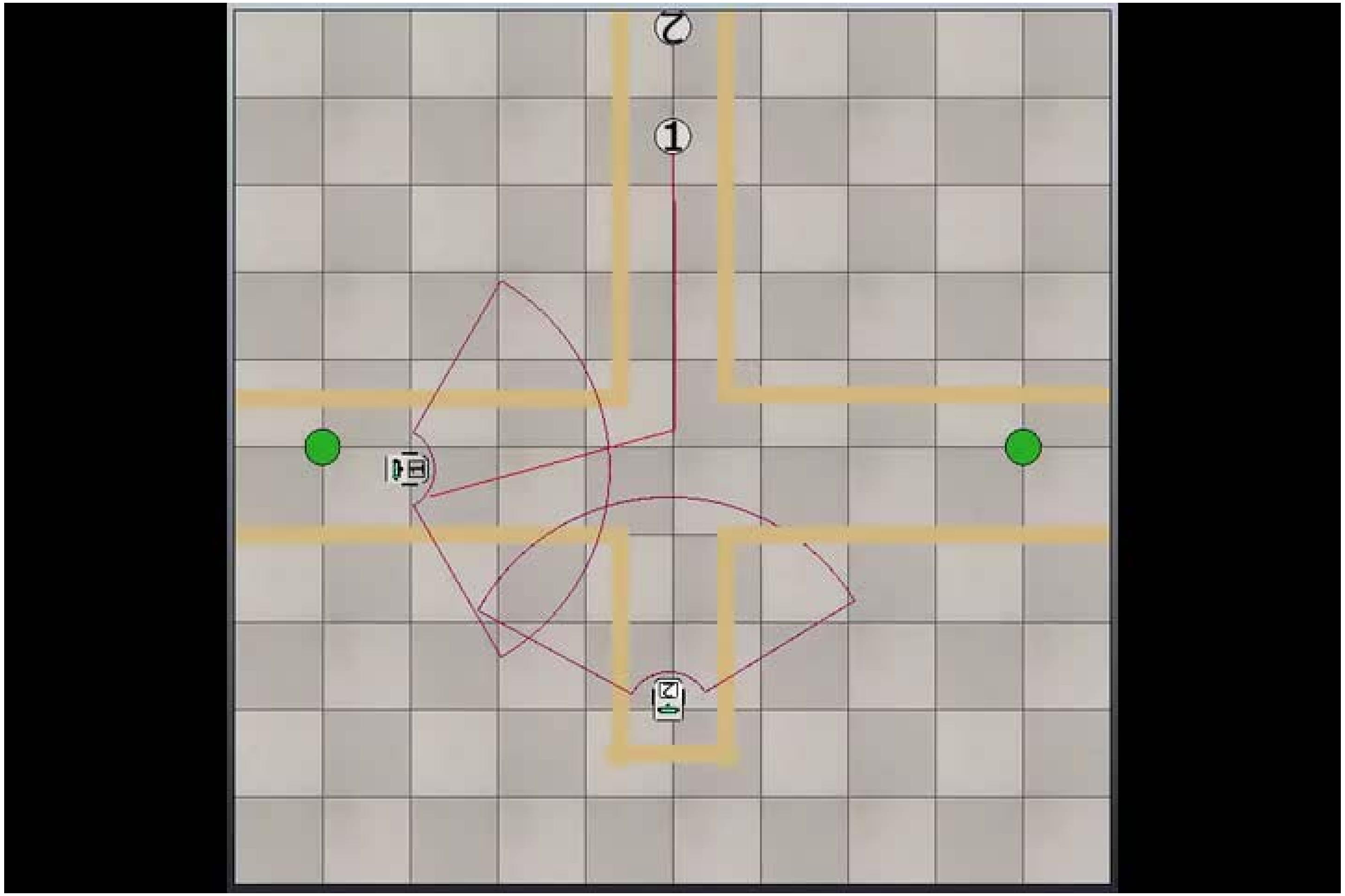
(a) Velocity obstacle



(b) Uncertainty-aware velocity obstacle

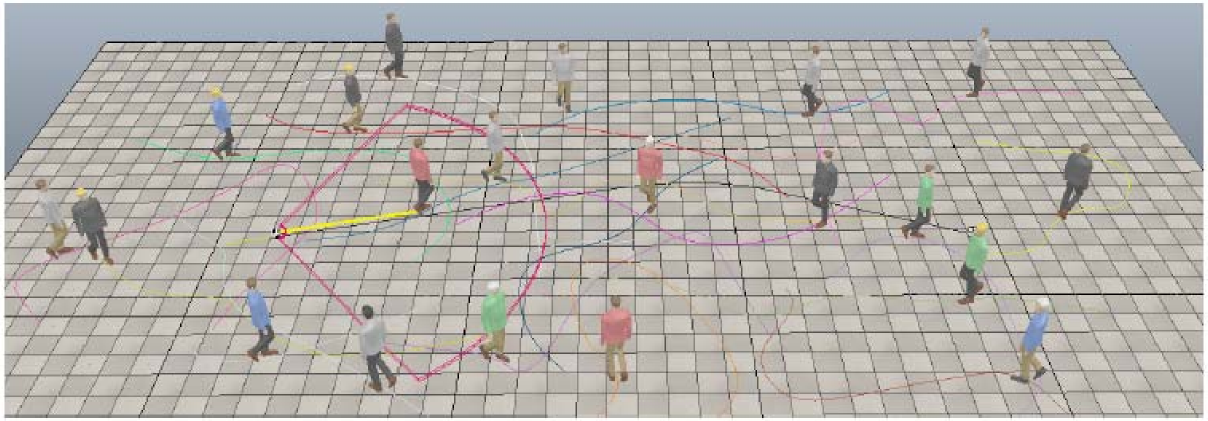
"The hybrid reciprocal velocity obstacle" **TRO11** J Snape, J van den Berg, SJ Guy
"Reciprocal velocity obstacles for real-time multi-agent navigation" J van den Berg
"Generalized Velocity Obstacles" **IROS09**, D Wilkie, J Van den Berg

Intersection scene – with UVO

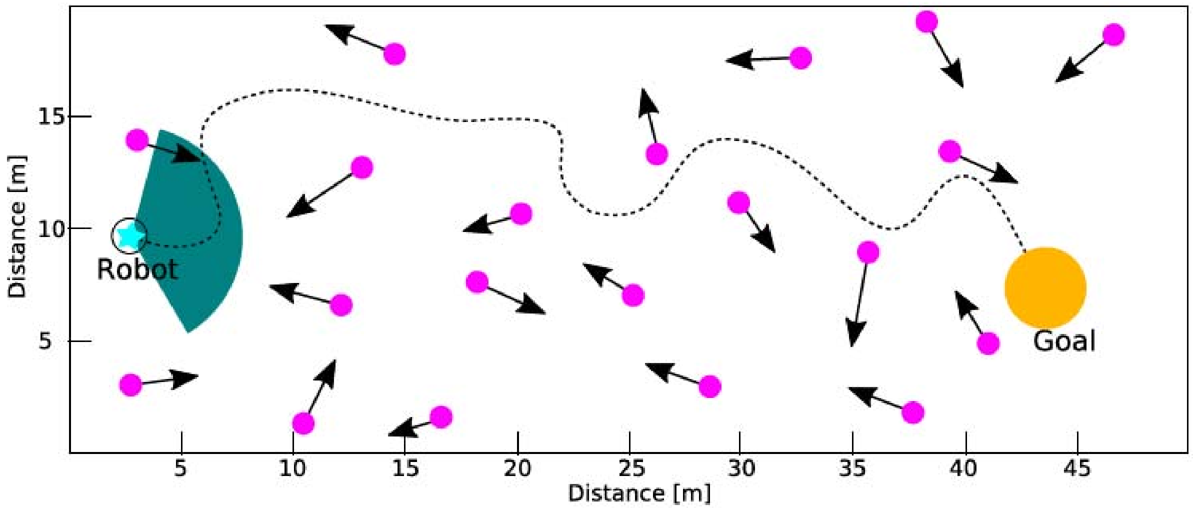


KAIST

Result – Crowd scene

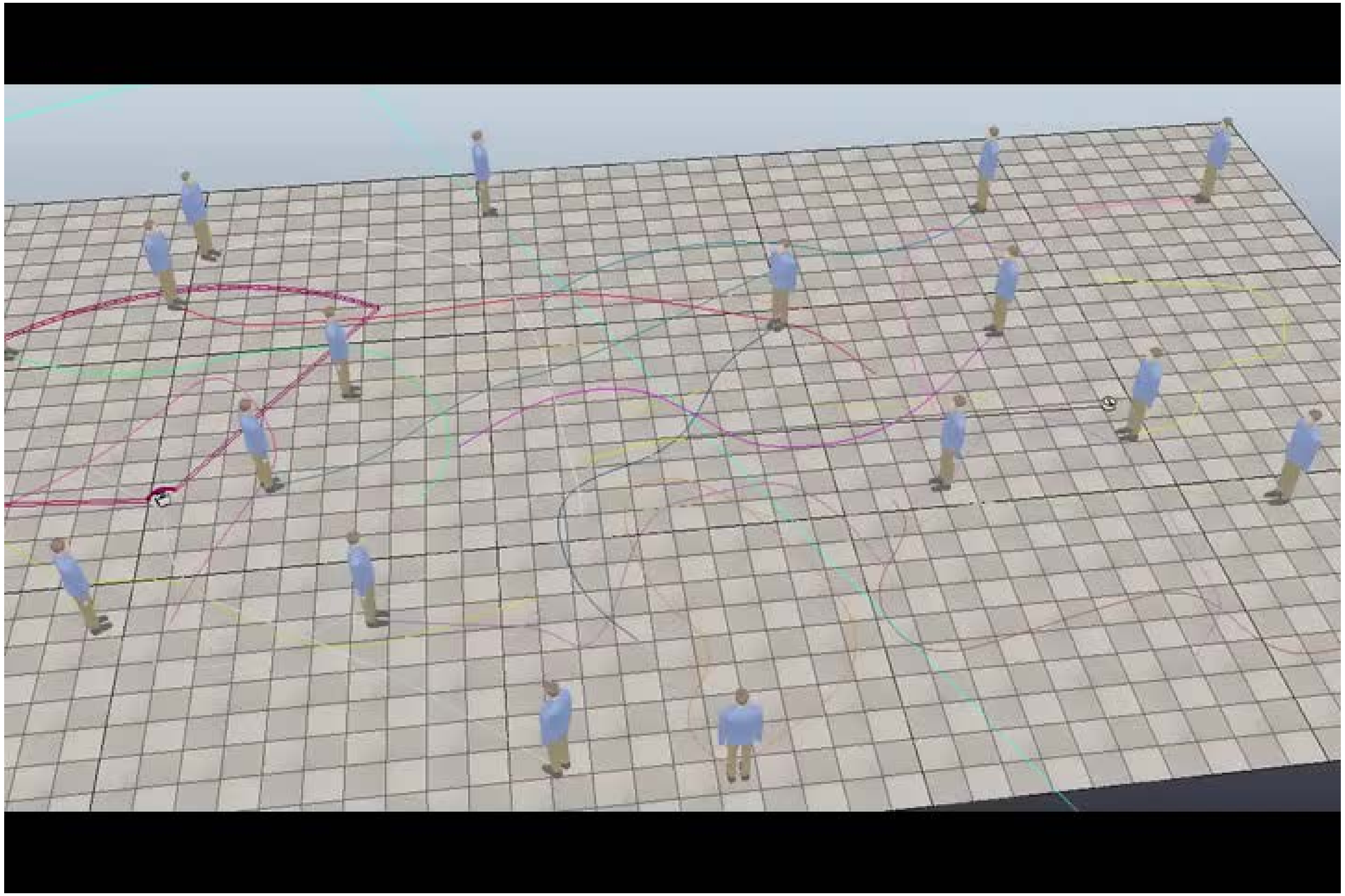


(a)



(b)

Result – Crowd scene



KAIST

Class Objectives were:

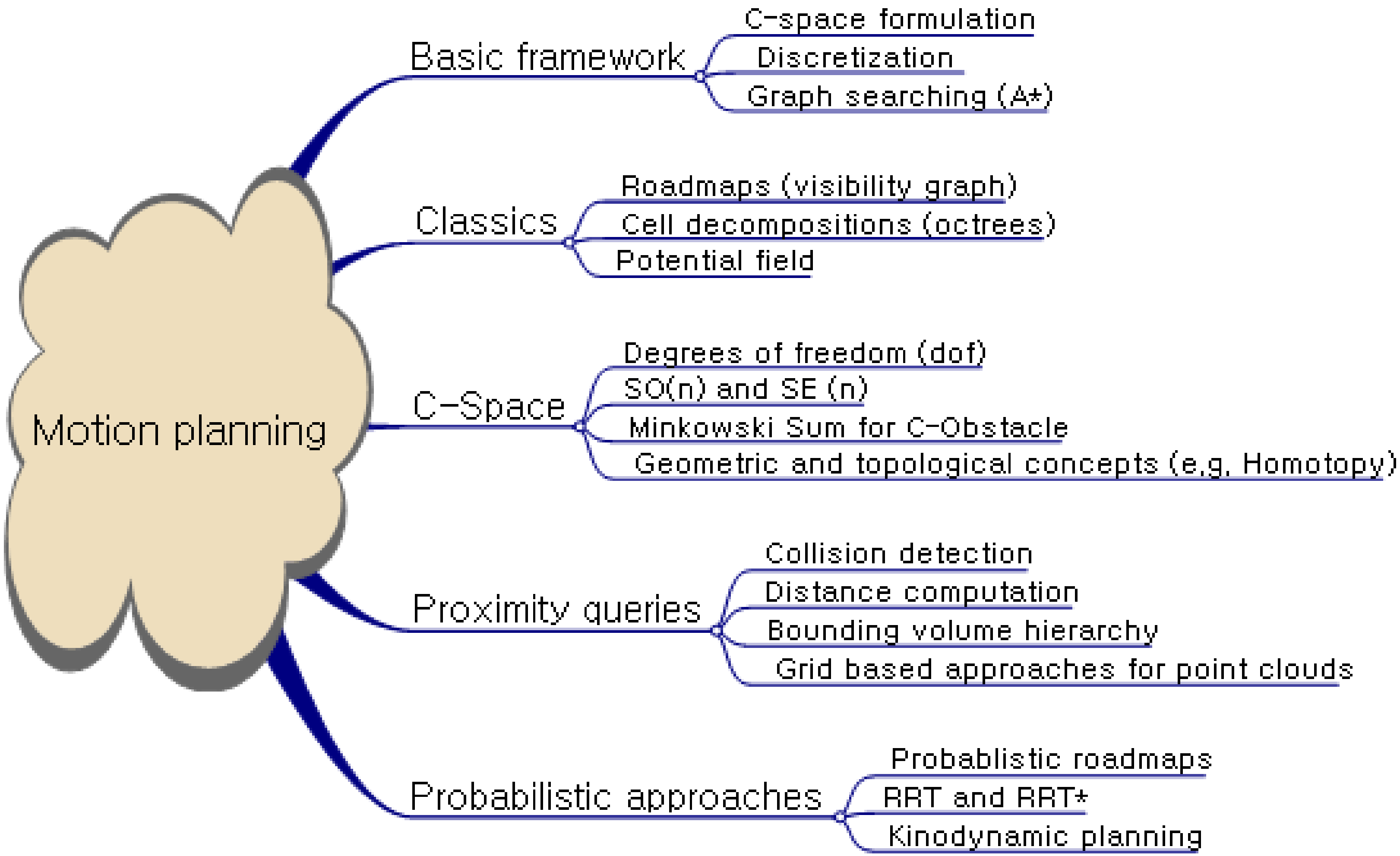
- **Understand the RRT technique and its recent advancements**
  - **RRT* for optimal path planning**
  - **Kinodynamic planning**

No More HWs on:

- **Paper summary and questions submissions**

- **Instead:**
  - **Focus on your paper presentation and project progress!**

Summary



Motion planning
Basic framework
C-space formulation
Discretization
Graph searching (A*)
Classics
Roadmaps (visibility graph)
Cell decompositions (octrees)
Potential field
C-Space
Degrees of freedom (dof)
SO(n) and SE (n)
Minkowski Sum for C-Obstacle
Geometric and topological concepts (e.g. Homotopy)
Proximity queries
Collision detection
Distance computation
Bounding volume hierarchy
Grid based approaches for point clouds
Probabilistic approaches
Probablistic roadmaps
RRT and RRT*
Kinodynamic planning