CS686:
RRT

Sung-Eui Yoon
(윤성의)

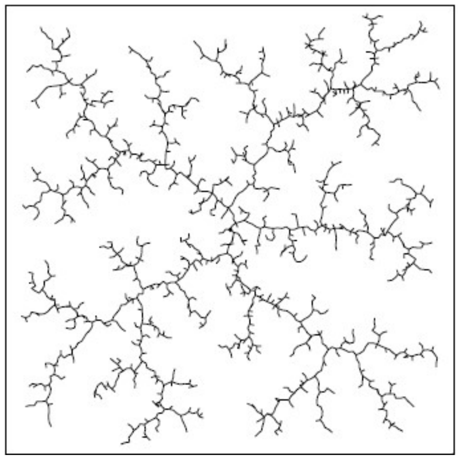
Course URL:
http://sgvr.kaist.ac.kr/~sungeui/MPA

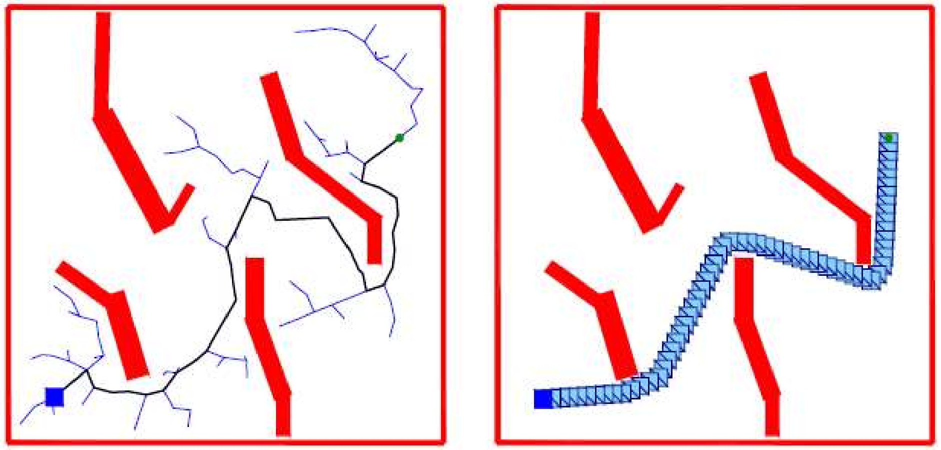
Class Objectives

- **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





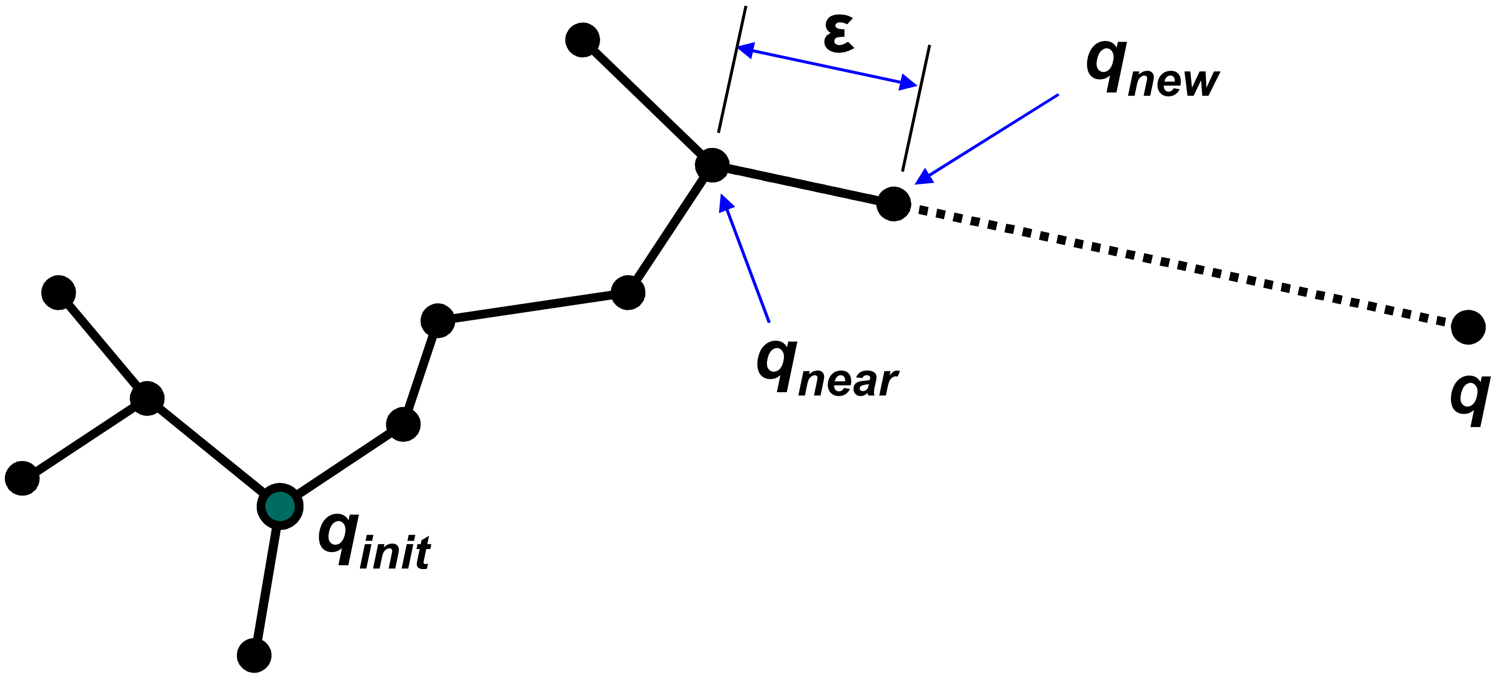
Rapidly-Exploring Random Tree

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

KAIST

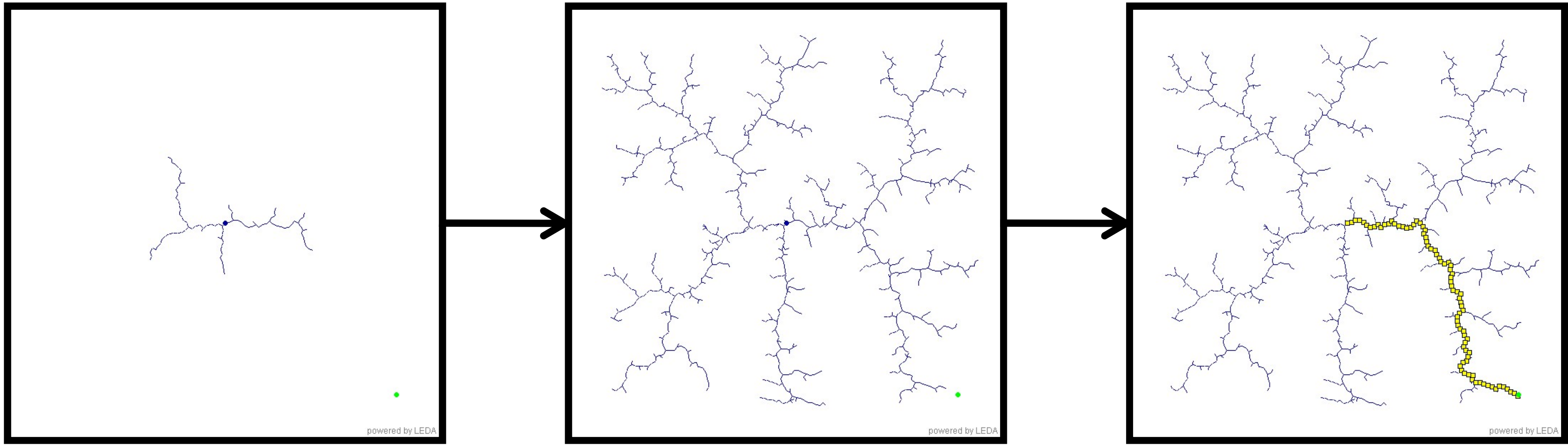
RRT Construction Algorithm

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



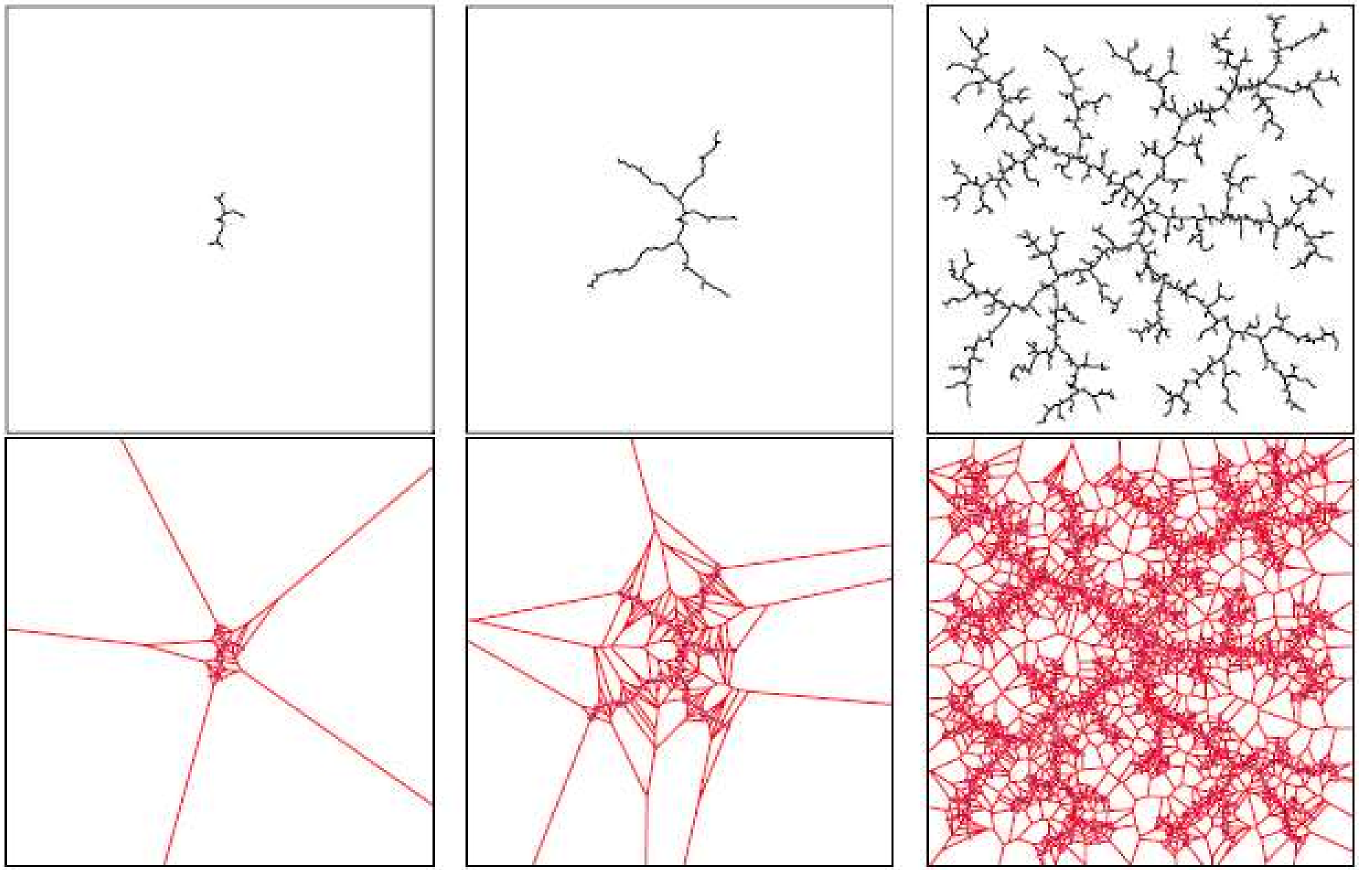
Overview – Planning with RRT

- **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**



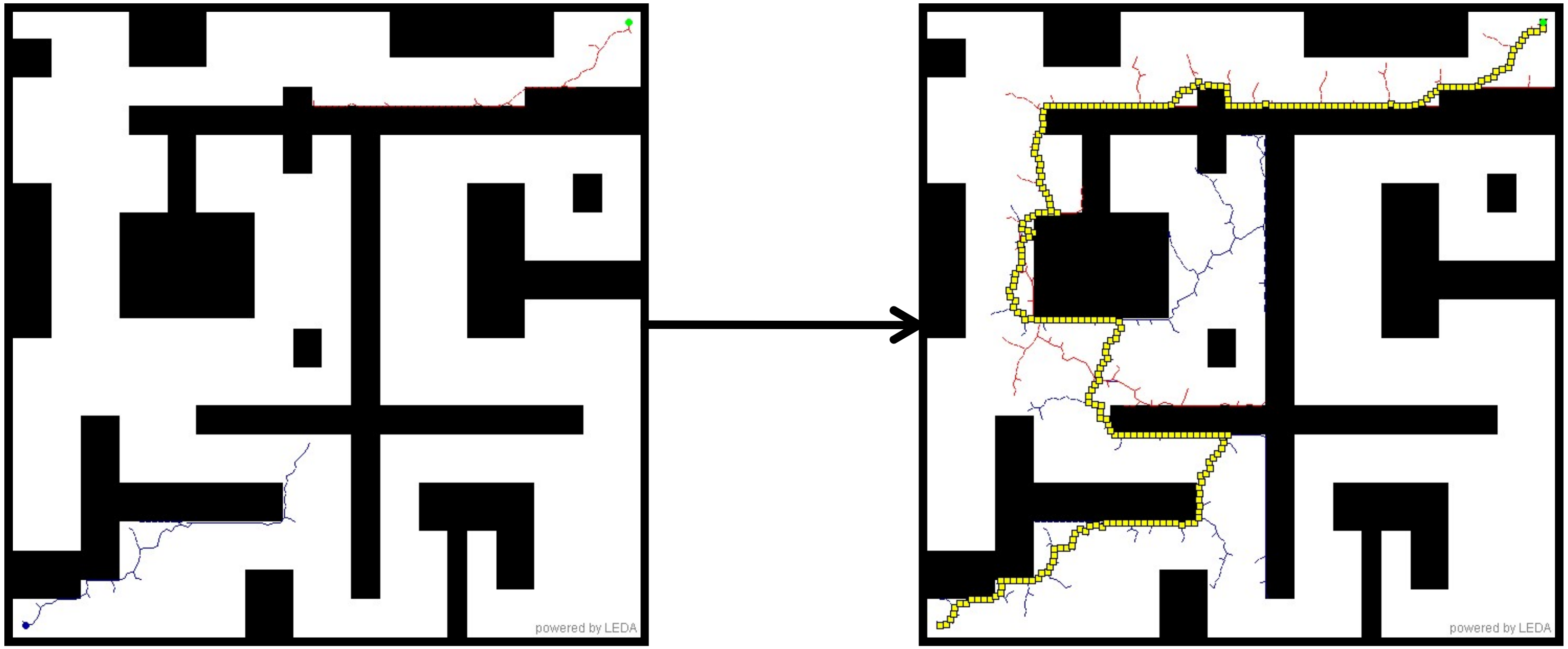
Voronoi Region

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



Overview – With Dual RRT

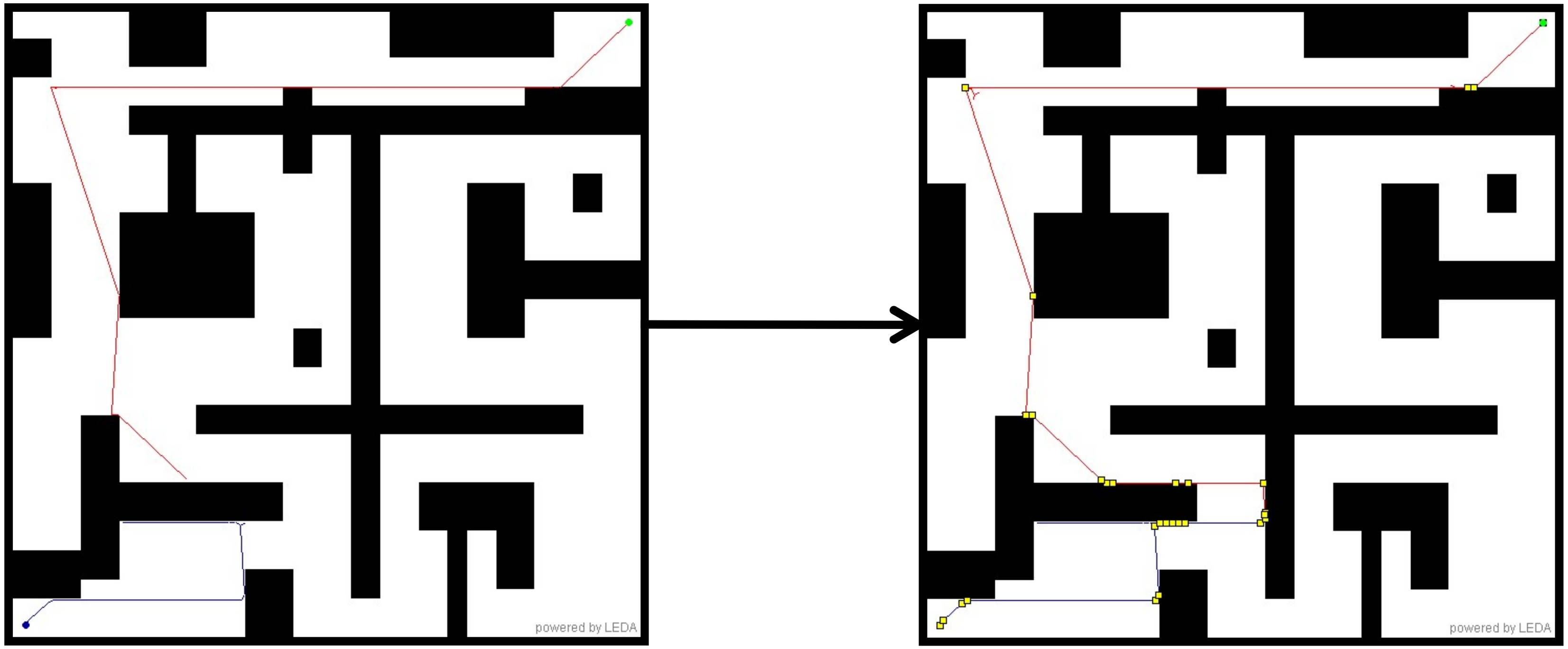
- **Extend RRTs from both initial and goal states**
- **Find path much more quickly**



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})

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

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

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

  EXTEND($\mathcal{T}, q_{rand}$);

Return $\mathcal{T}$

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

$q_{near} \leftarrow$ NEAREST_NEIGHBOR($q, \mathcal{T}$);

if NEW_CONFIG(q, q_{near}, q_{new}) **then**

  $\mathcal{T}$.add_vertex(q_{new});

  $\mathcal{T}$.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

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

until not ($S = \textit{Advanced}$)

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**

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

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

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

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

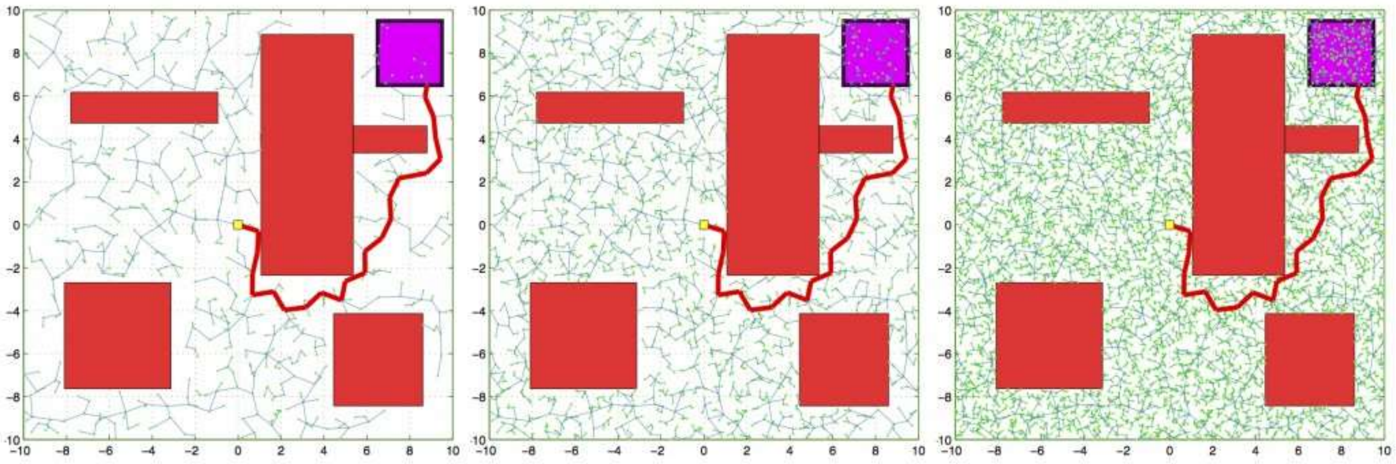
$\quad$SWAP($\mathcal{T}_a, \mathcal{T}_b$);

Return *Failure*

RRT*

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

RRT



(a) RRT in iteration 1,000 (b) RRT in iteration 3,000 (c) RRT in iteration 10,000

RRT*

From Sertac's homepage

RRT*

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

Theorem [Karaman & Frazzoli, IJRR 2011]

(i) The RRT* algorithm is asymptotically optimal

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

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

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

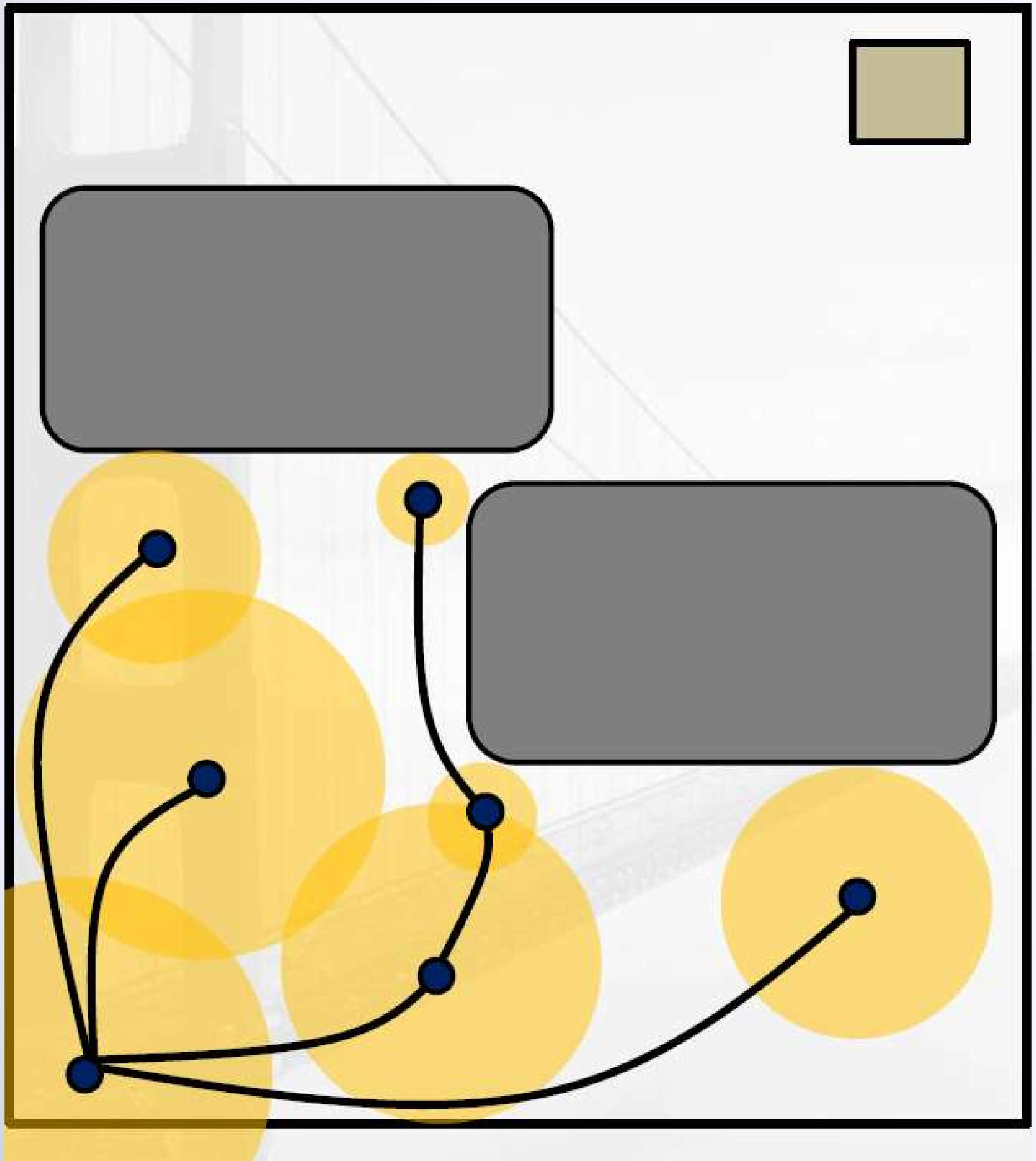
- **Y_n^{RRT*} : cost of the best path in the RRT***
- **c^* : cost of an optimal solution**
- **M_n^{RRT} : # of steps executed by RRT at iteration n**
- **M_n^{RRT*}: # of steps executed by RRT* at iteration n**

From DH's homepage

Key Operation of RRT*

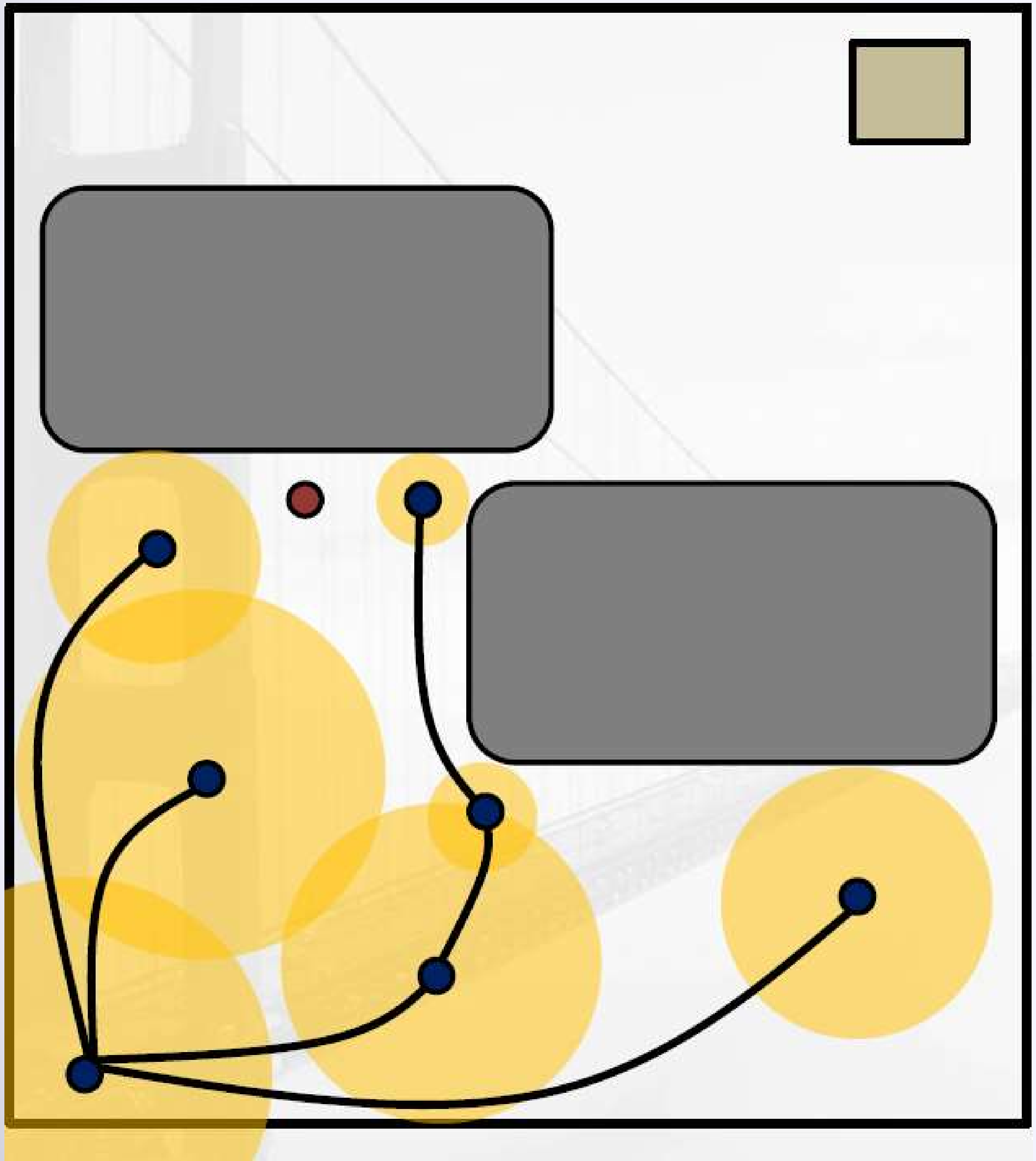
- **RRT**
  - **Just connect a new node to its nearest neighbor node**
- **RRT*: refine the connection with re-wiring operation**
  - **Given a ball, identify neighbor nodes to the new node**
  - **Refine the connection to have a lower cost**

Example: Re-Wiring Operation



From ball tree paper

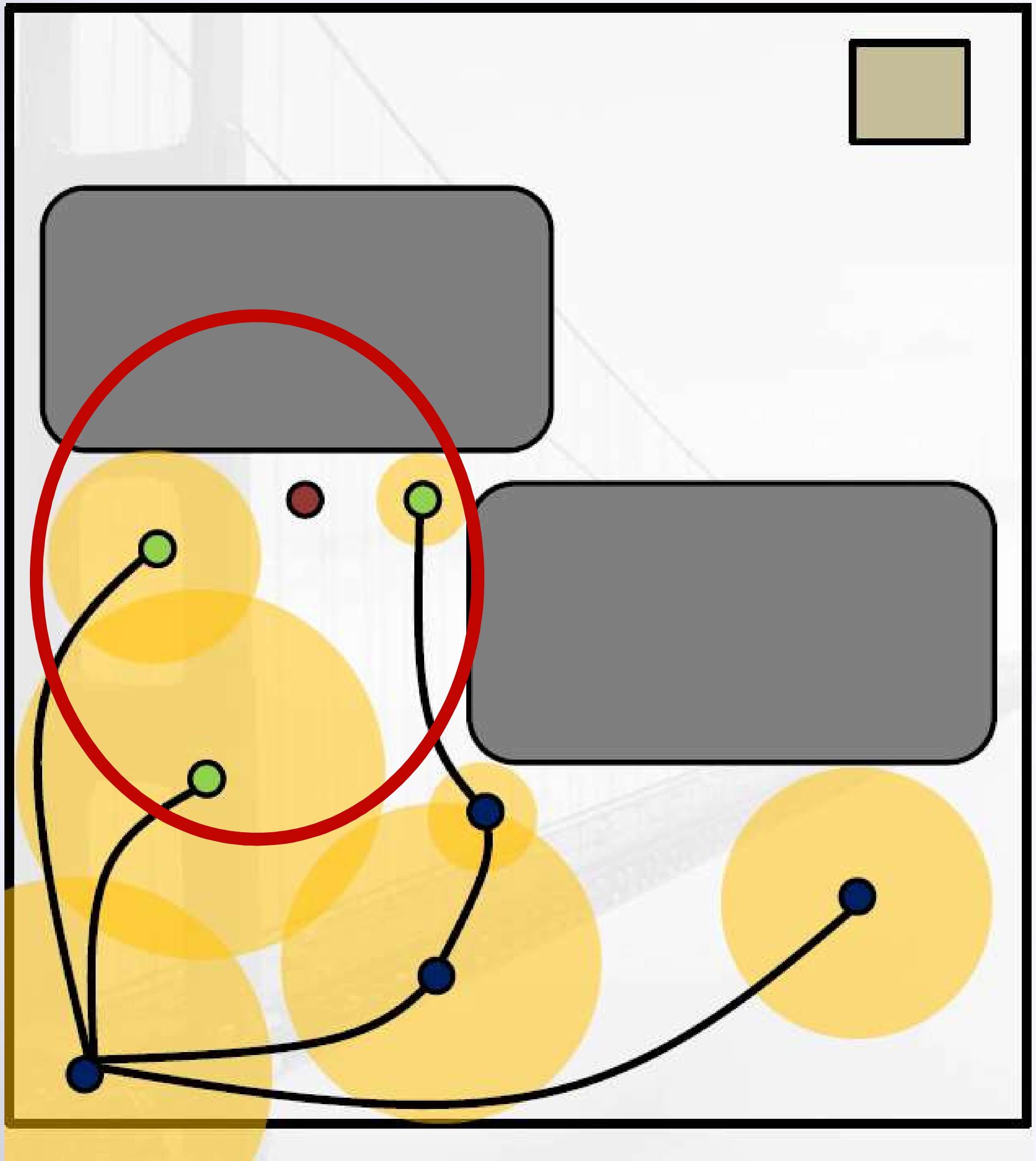
Example: Re-Wiring Operation



Generate a new sample

From ball tree paper

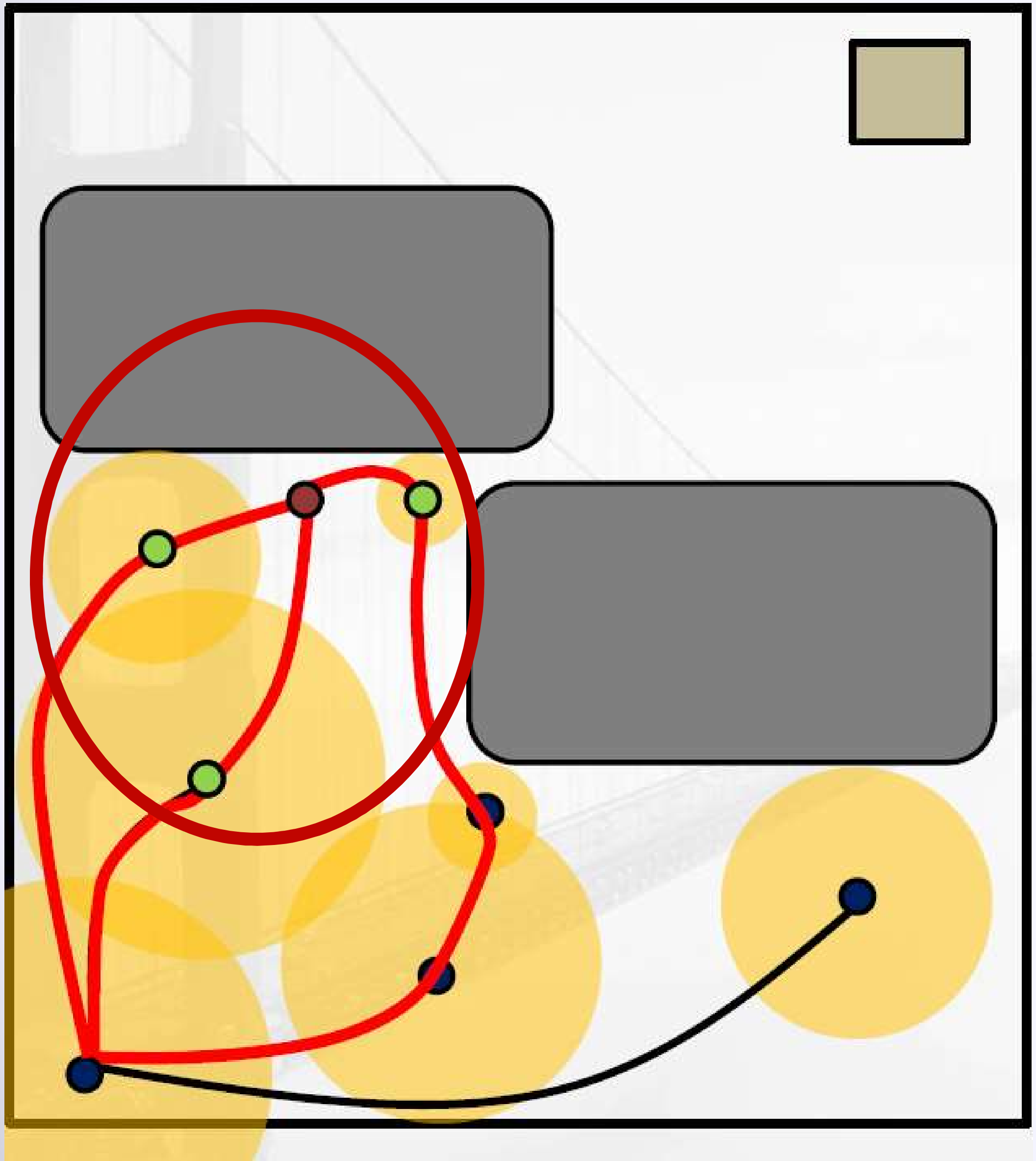
Example: Re-Wiring Operation



Identify nodes in a ball

From ball tree paper

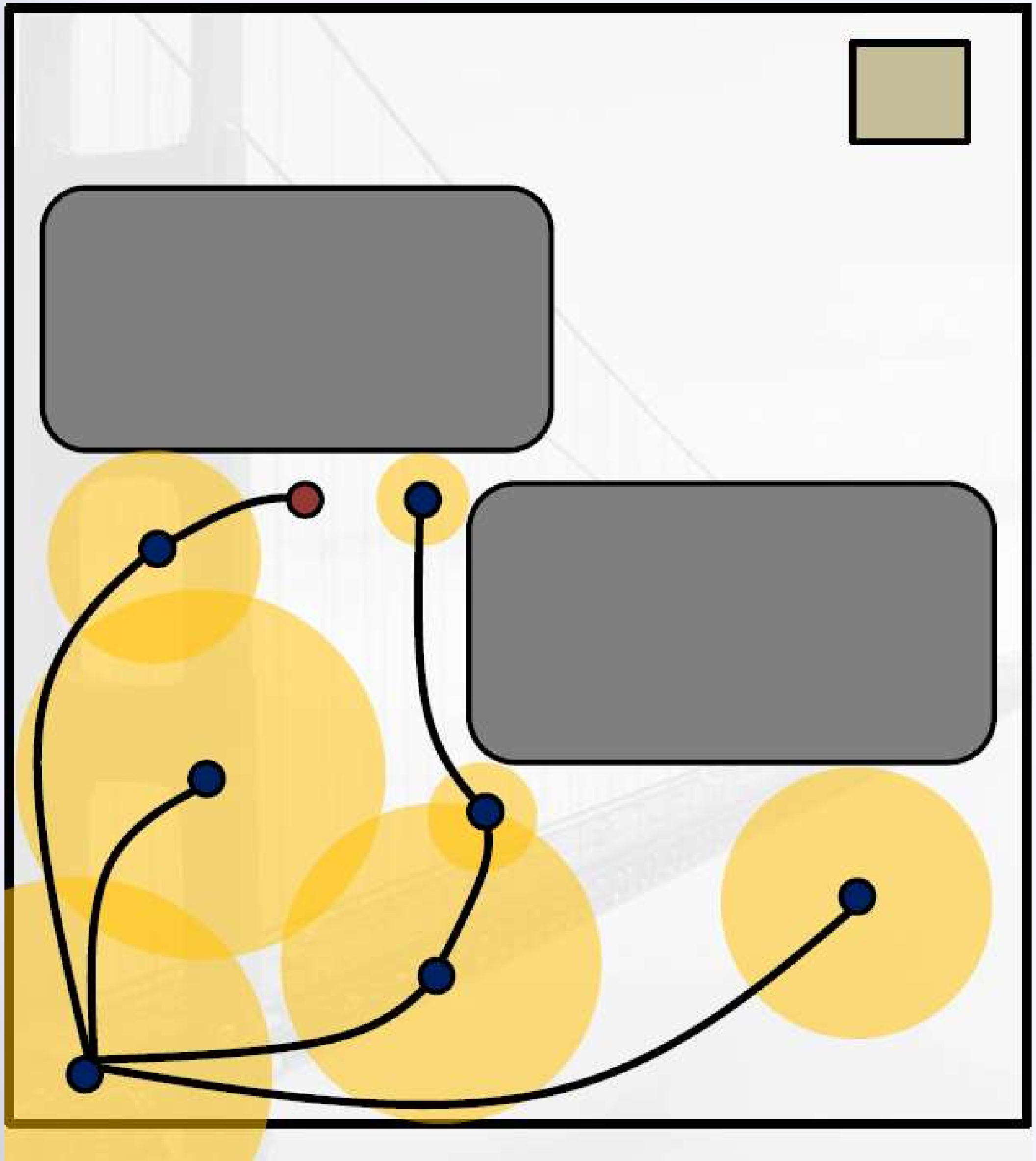
Example: Re-Wiring Operation



Identify which parent gives the lowest cost

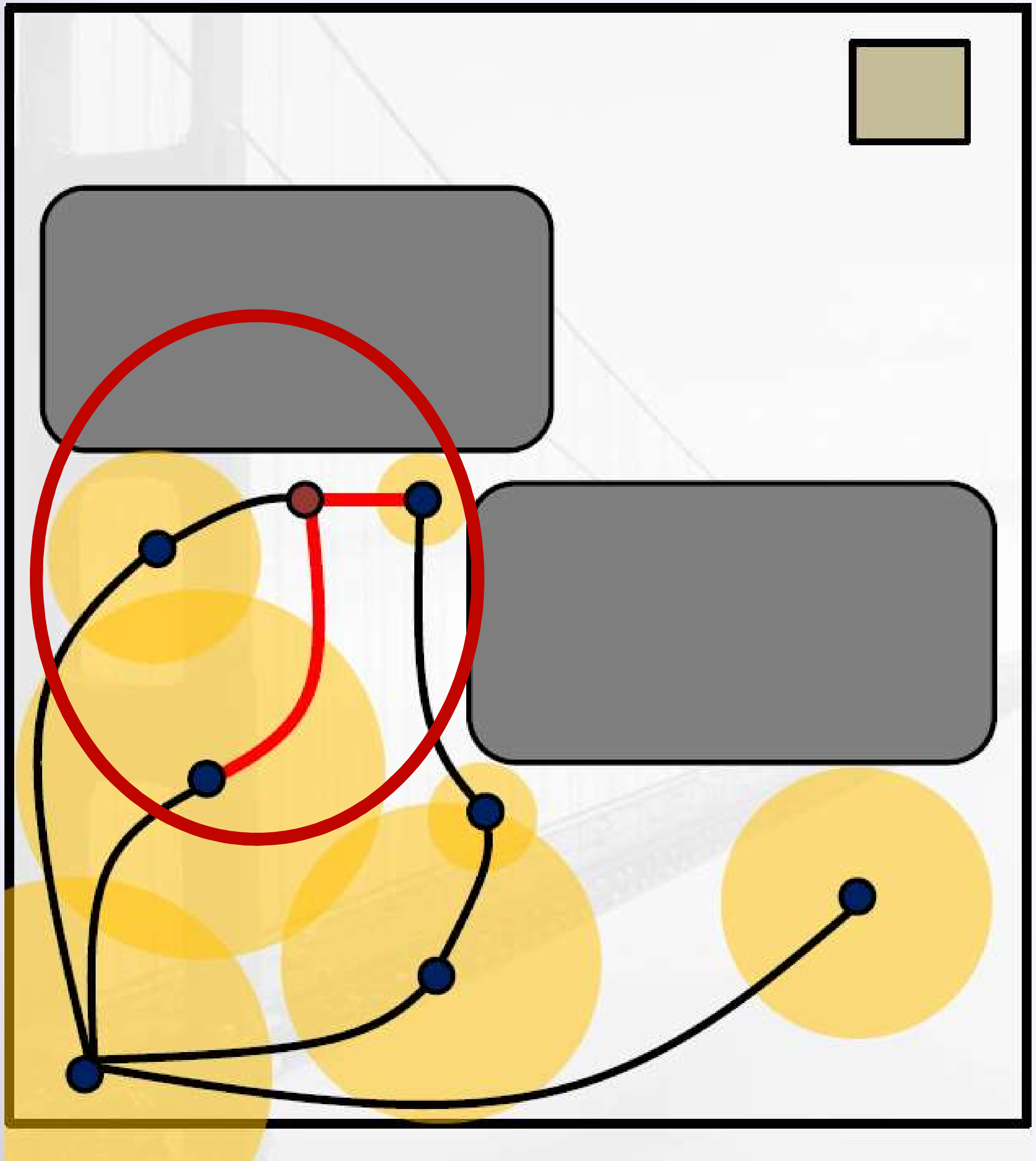
From ball tree paper

Example: Re-Wiring Operation



From ball tree paper

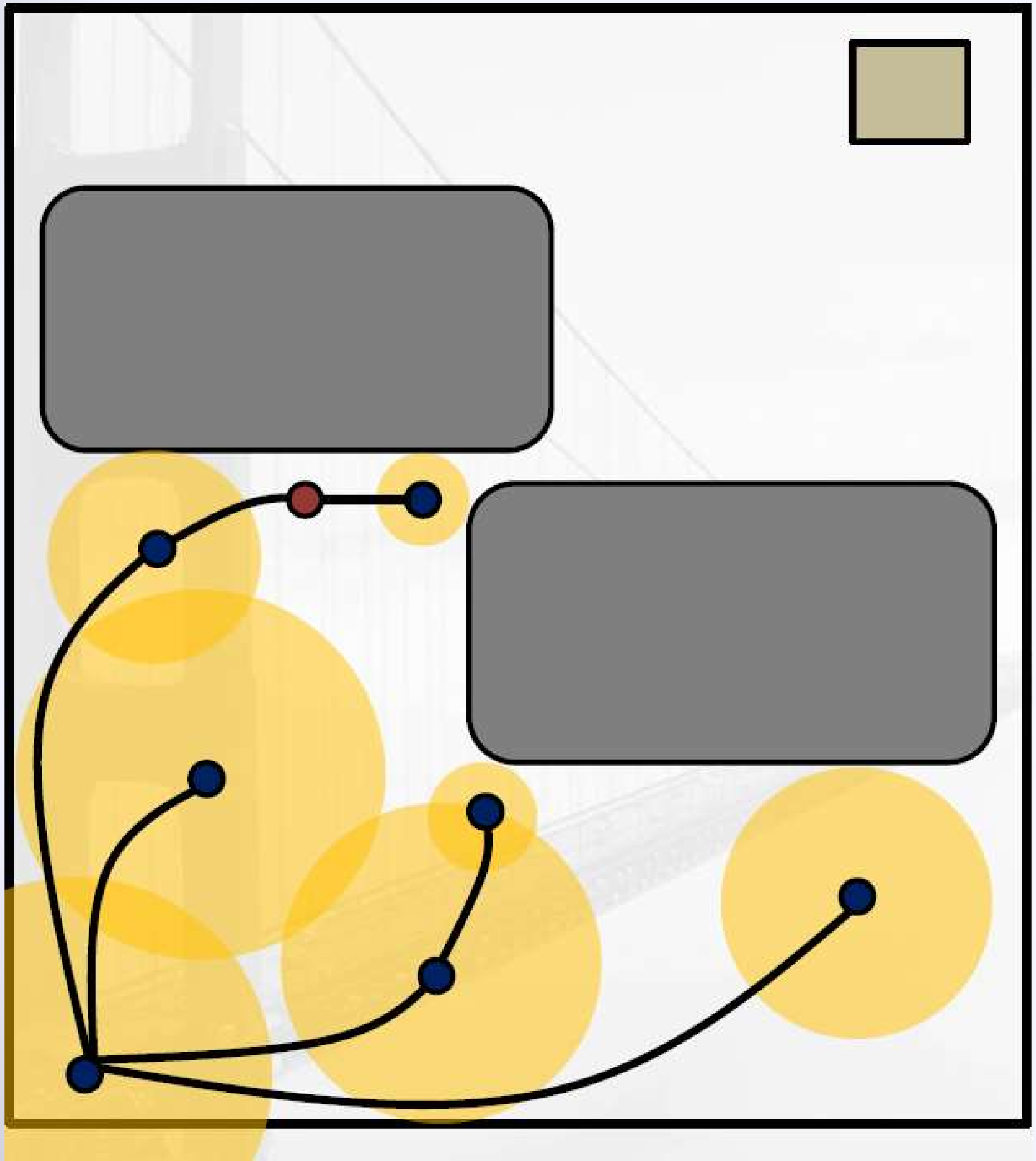
Example: Re-Wiring Operation



Identify which child gives the lowest cost

From ball tree paper

Example: Re-Wiring Operation



Video showing benefits with real robot

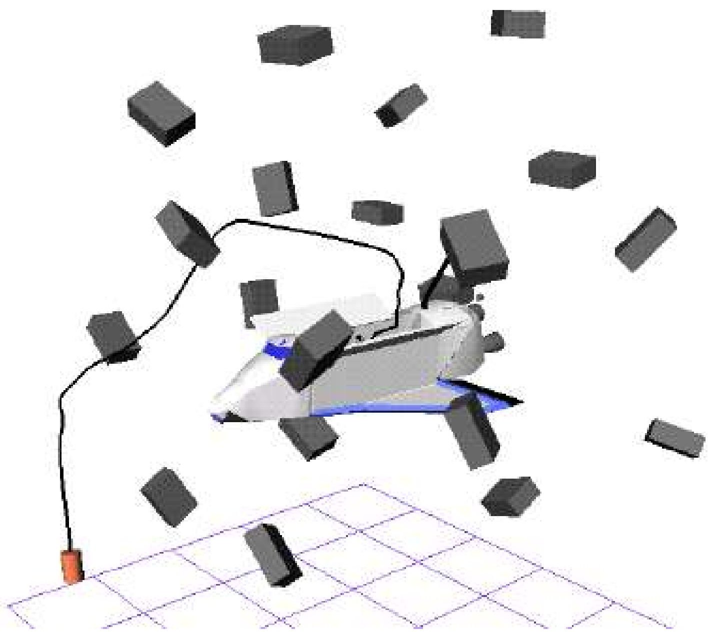
From ball tree paper

Kinodynamic Path Planning

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

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

- **Consider kinematic + dynamic constraints**



State Space Formulation

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

C denote the C-space

X denote the state space

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

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

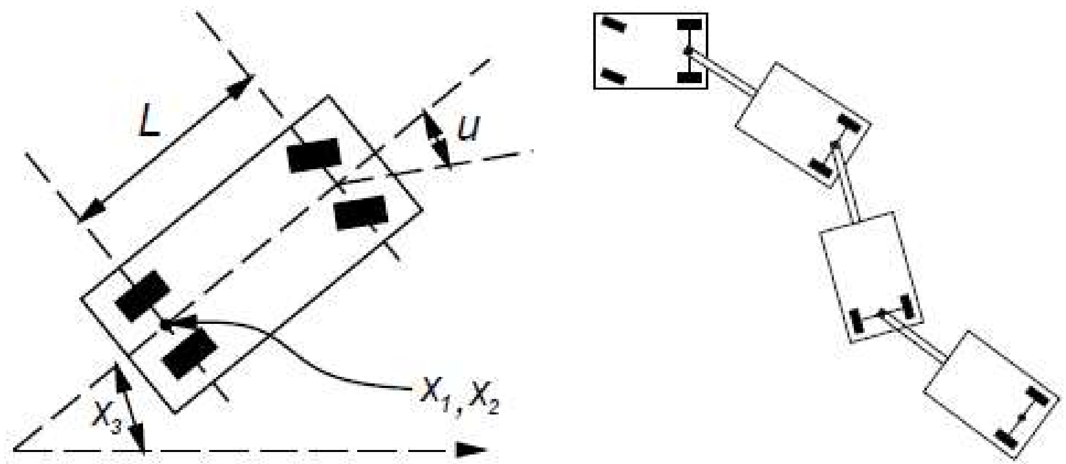
Constraints in State Space

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

- **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**

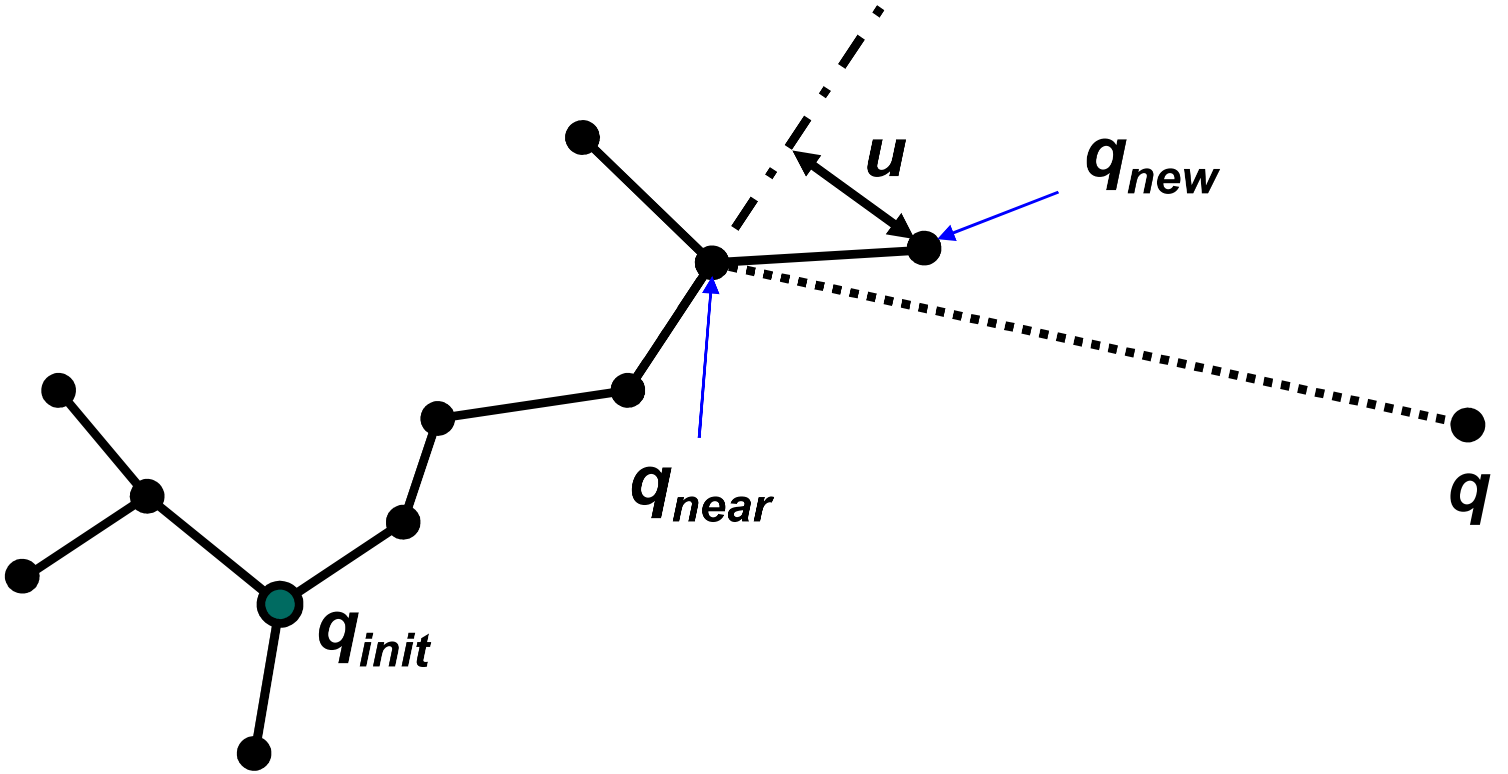
$$\tau : [0, T] \rightarrow X_{free}, \text{ satisfies the constraints}$$

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

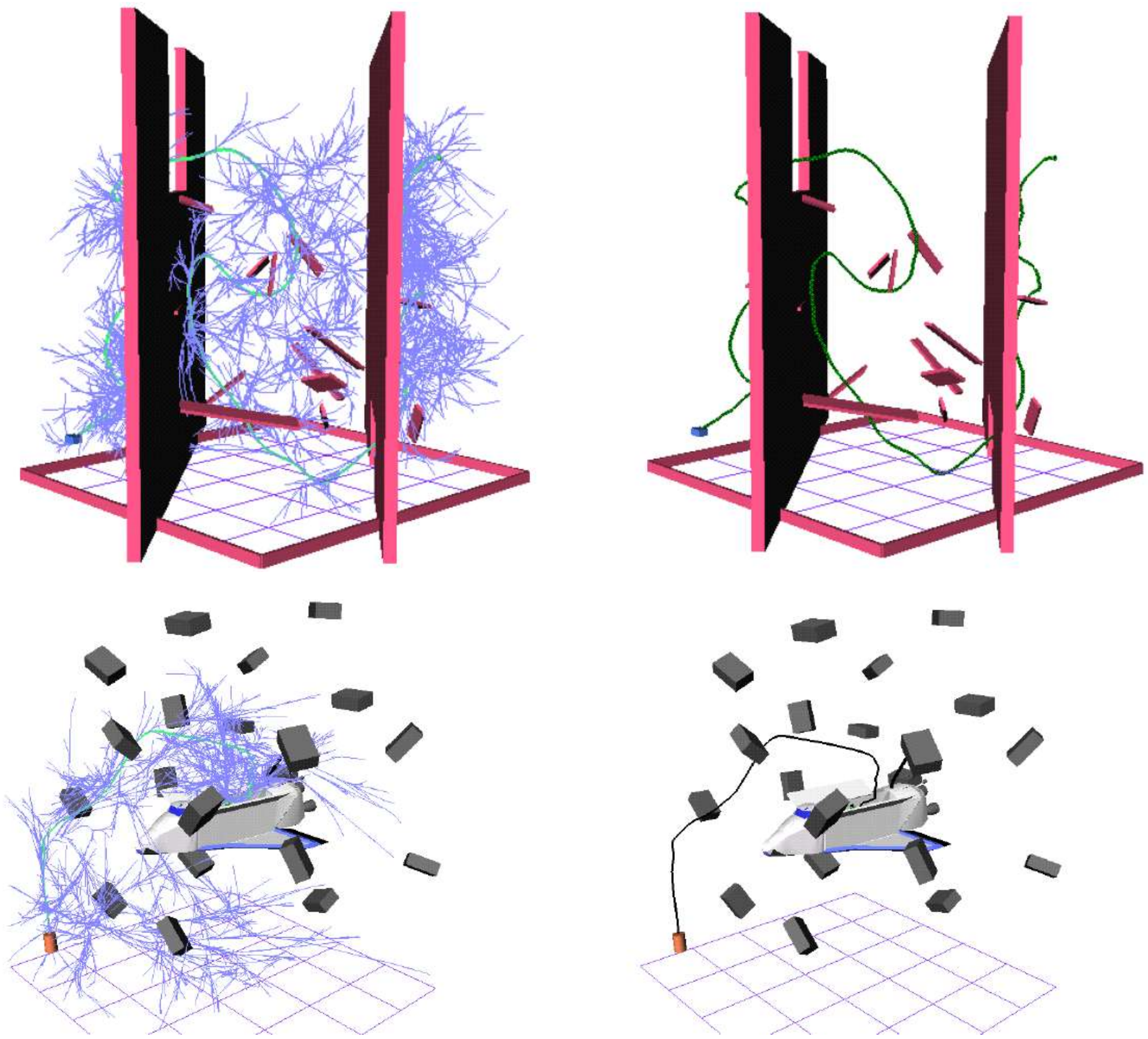
$$u : [0, T] \rightarrow U$$

Rapidly-Exploring Random Tree

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



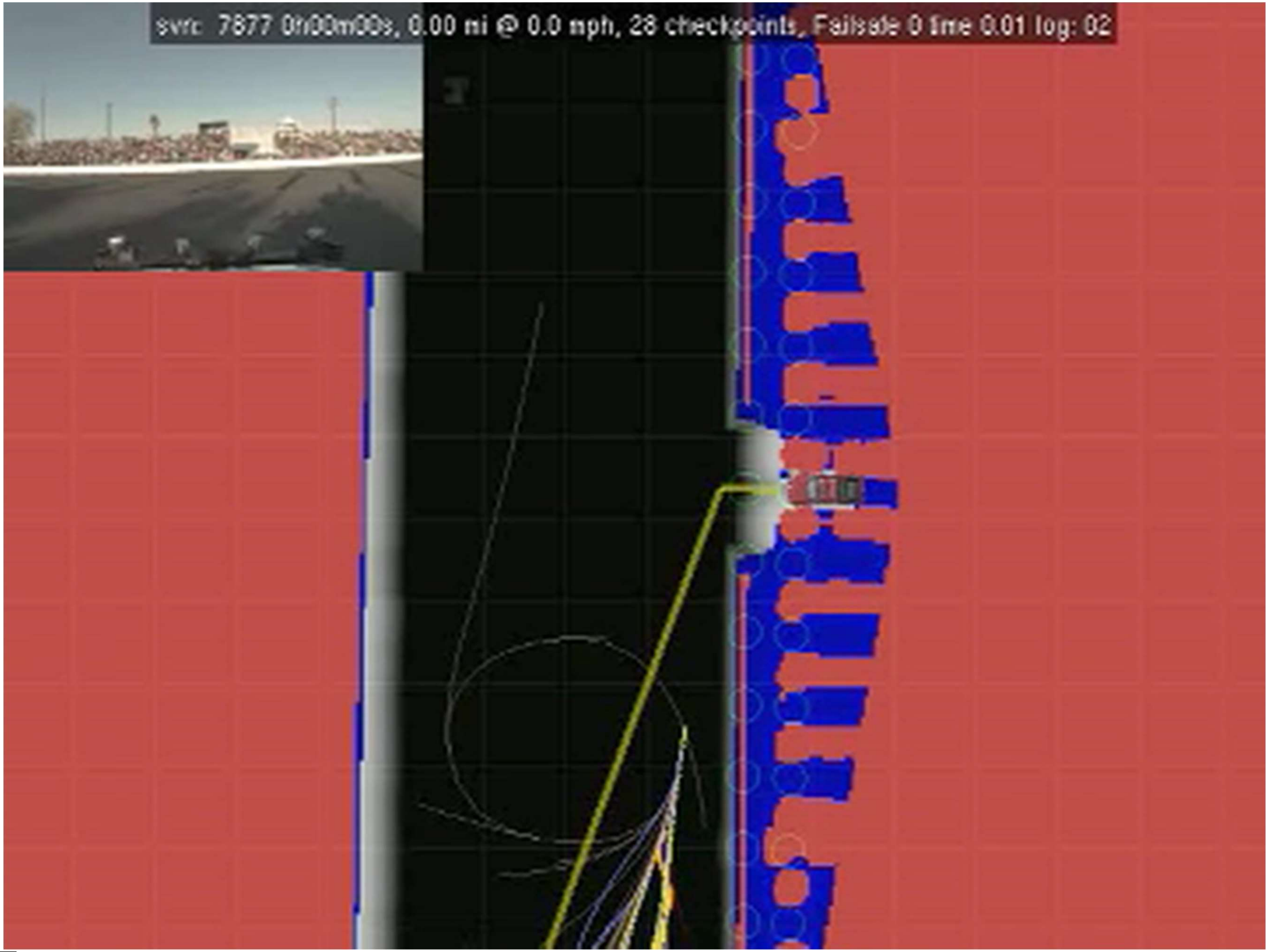
Results – 200MHz, 128MB



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

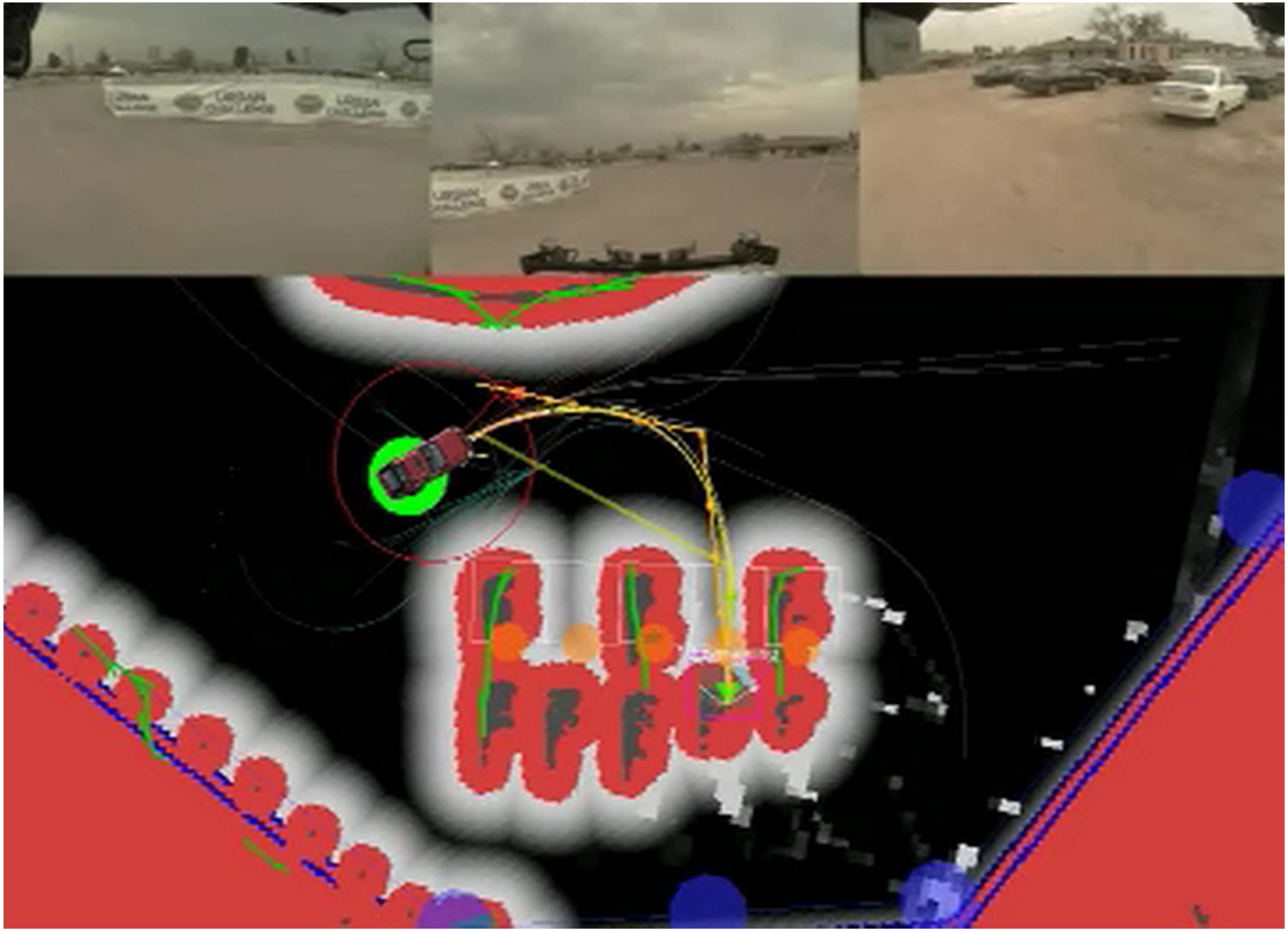
- 3D TR+RO
- X=12 DOF
- 23,800 nodes
- 8.4min

RRT at work: Urban Challenge

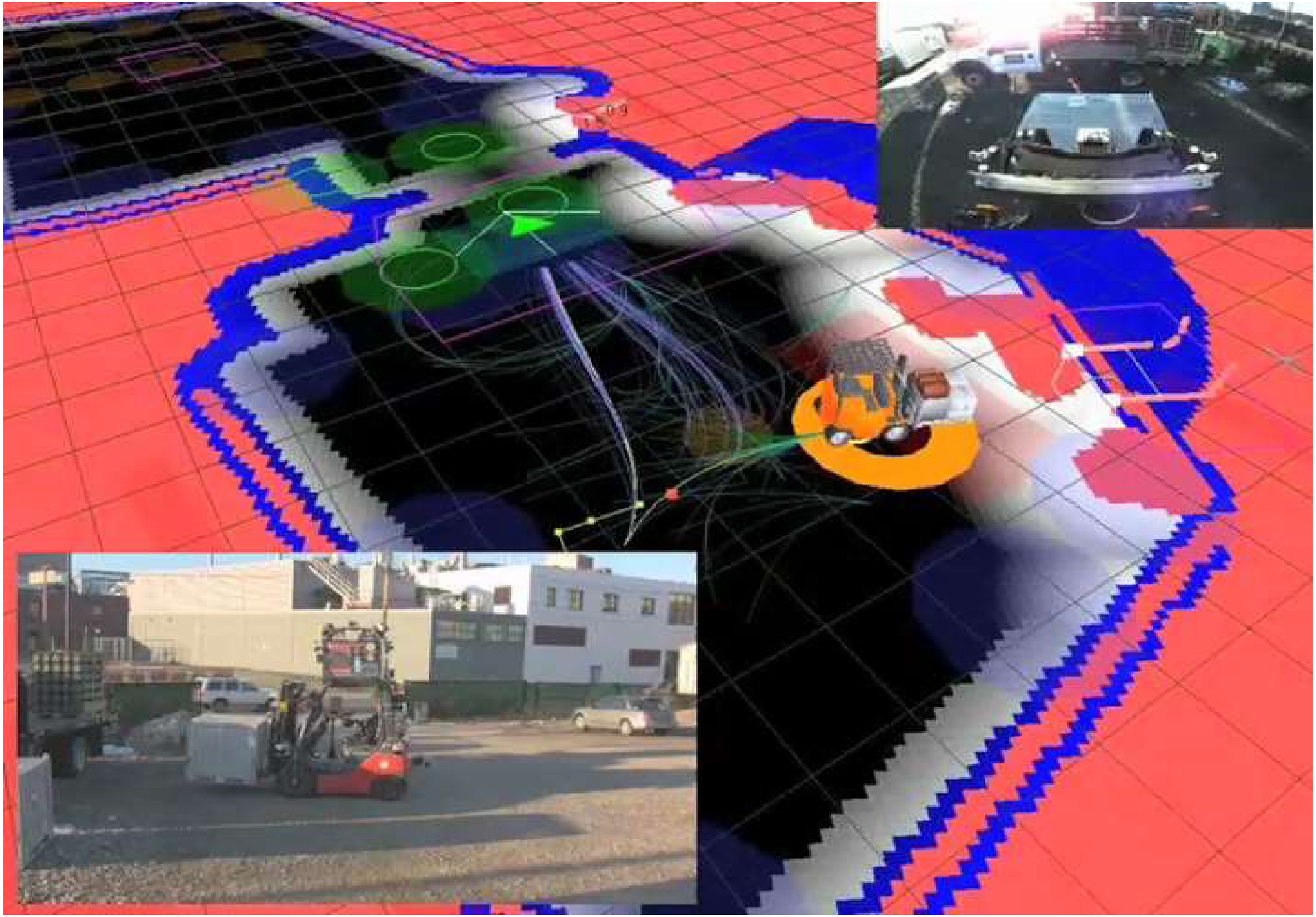


From MIT

Successful Parking Maneuver



RRT at work: Autonomous Forklift



Some 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

Retration-based RRT [Zhang & Manocha 08]

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

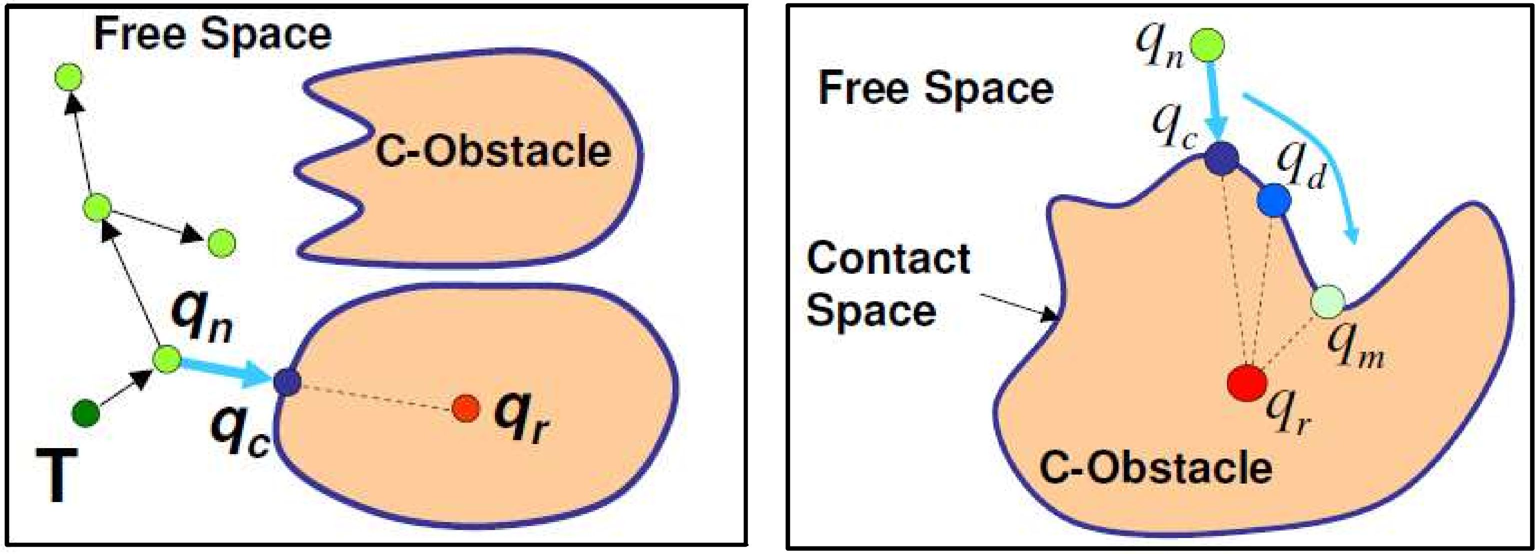
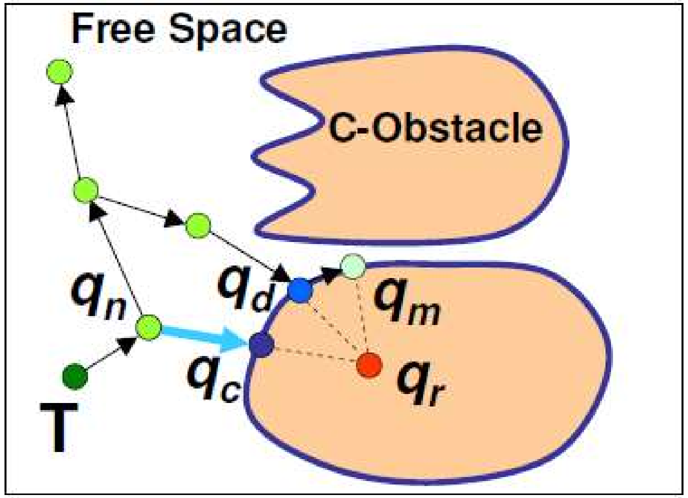


image from [Zhang & Manocha 08]

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

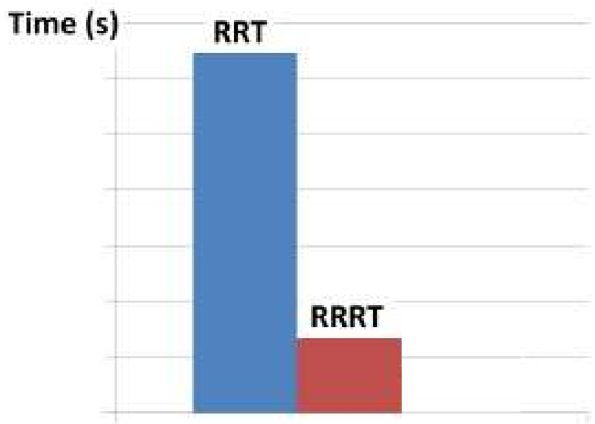
RRRT: Pros and Cons



with narrow passages

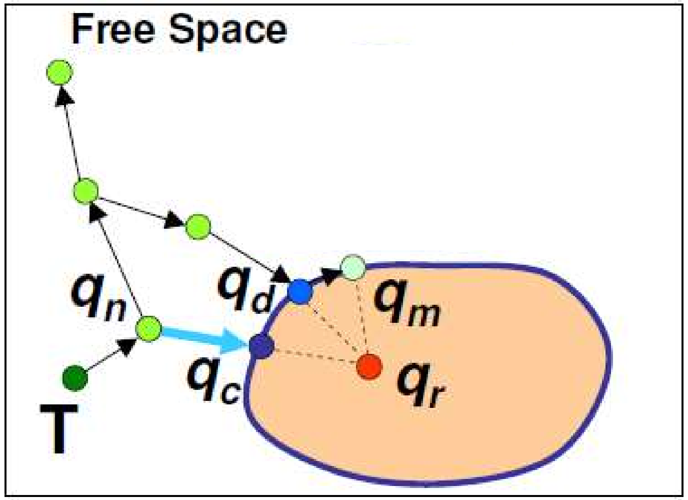
without narrow passages

images from [Zhang & Manocha 08]



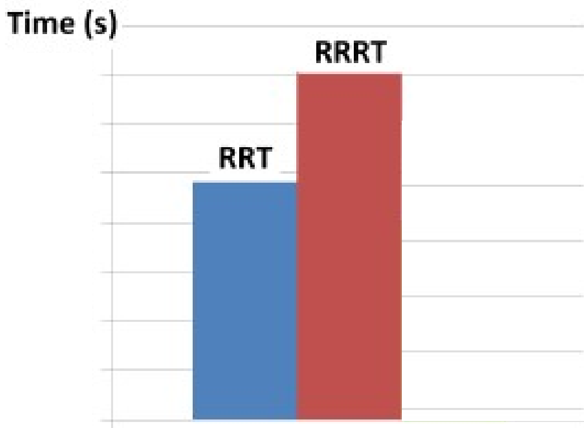
RRRT: Pros and Cons

with narrow passages



without narrow passages

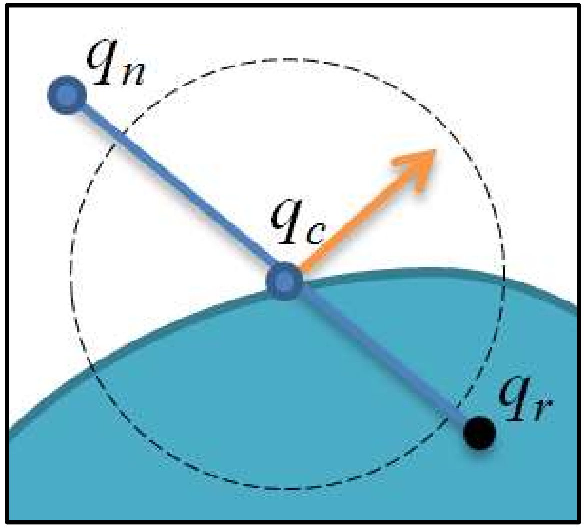
images from [Zhang & Manocha 08]

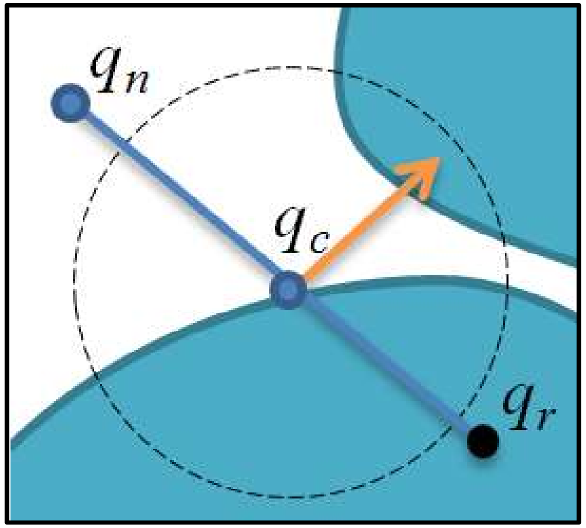


Bridge line-test [Lee et al., T-RO 14]

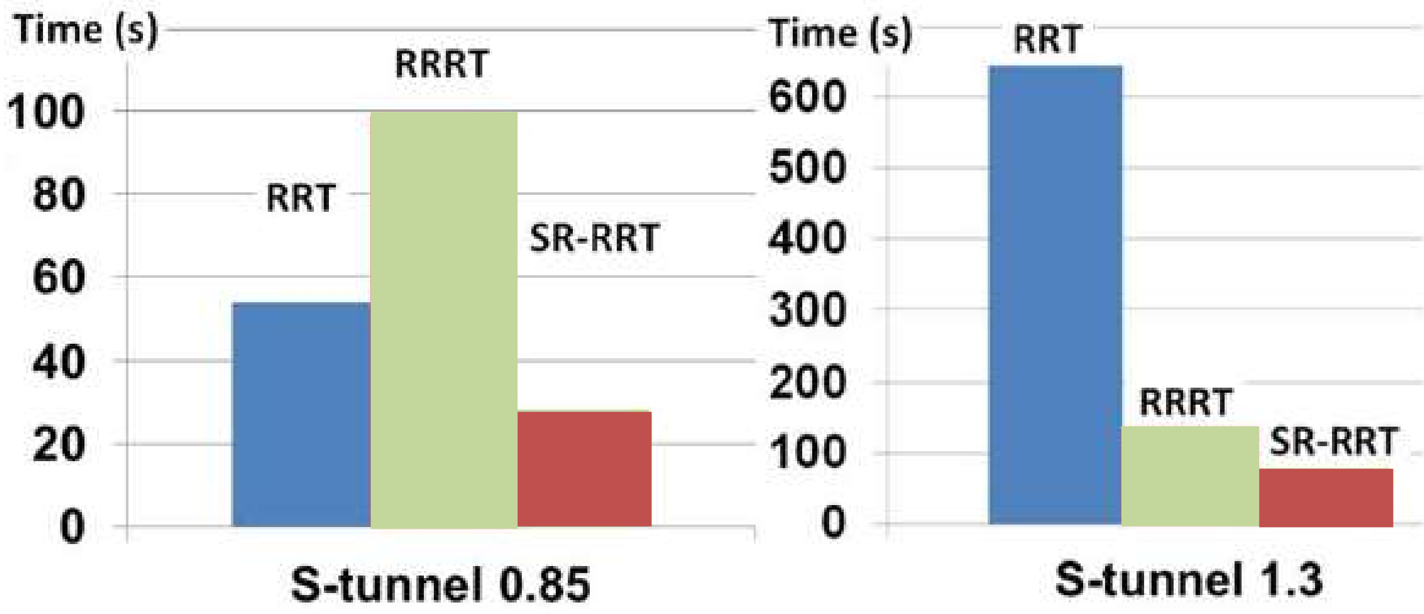
- To identify narrow passage regions

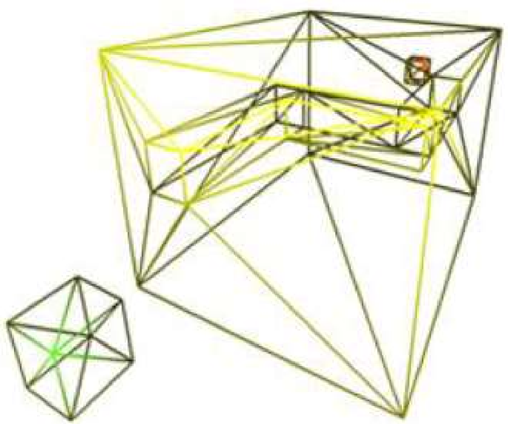
- Bridge line-test
  1. Generate a random line
  2. Check whether the line meets any obstacle





Results





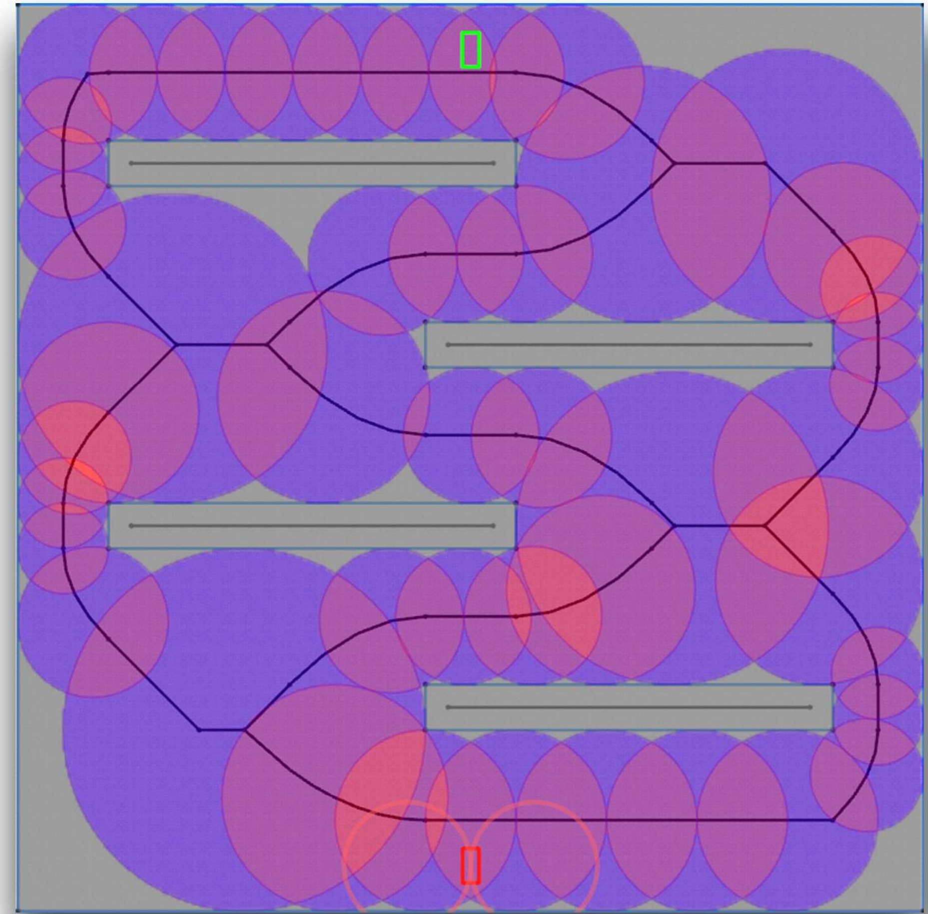
Video

Some Works of Our Group

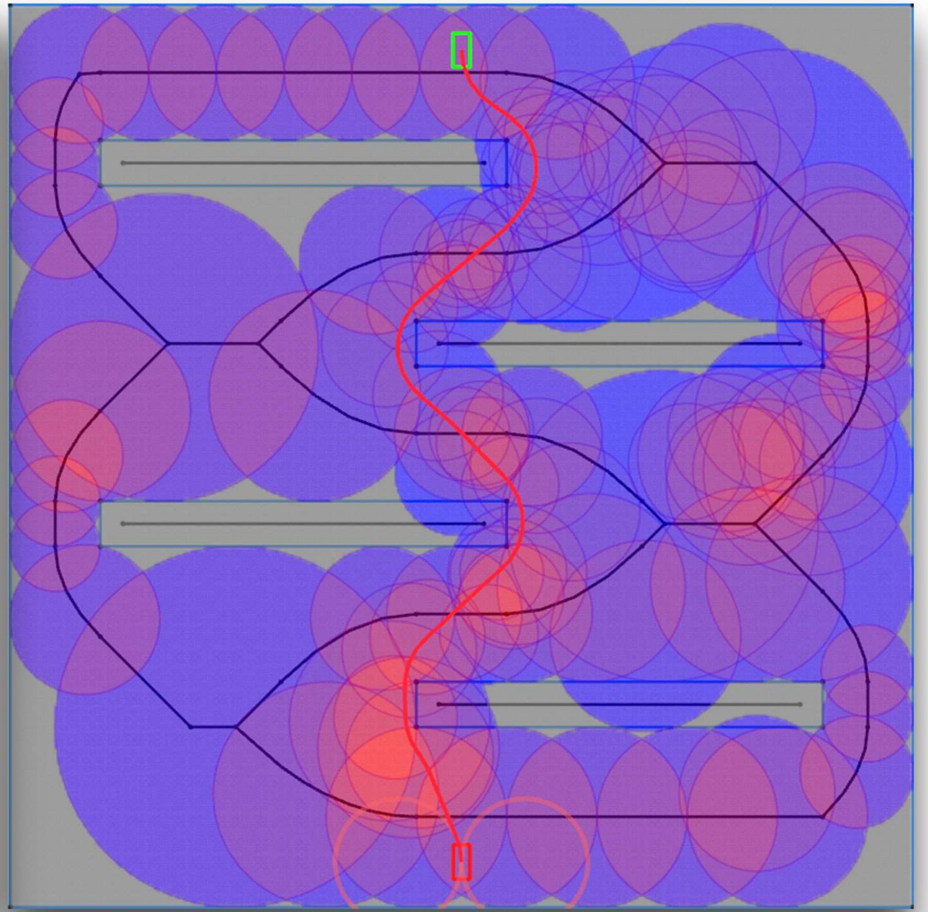
- Handling narrow passages

- 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/

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



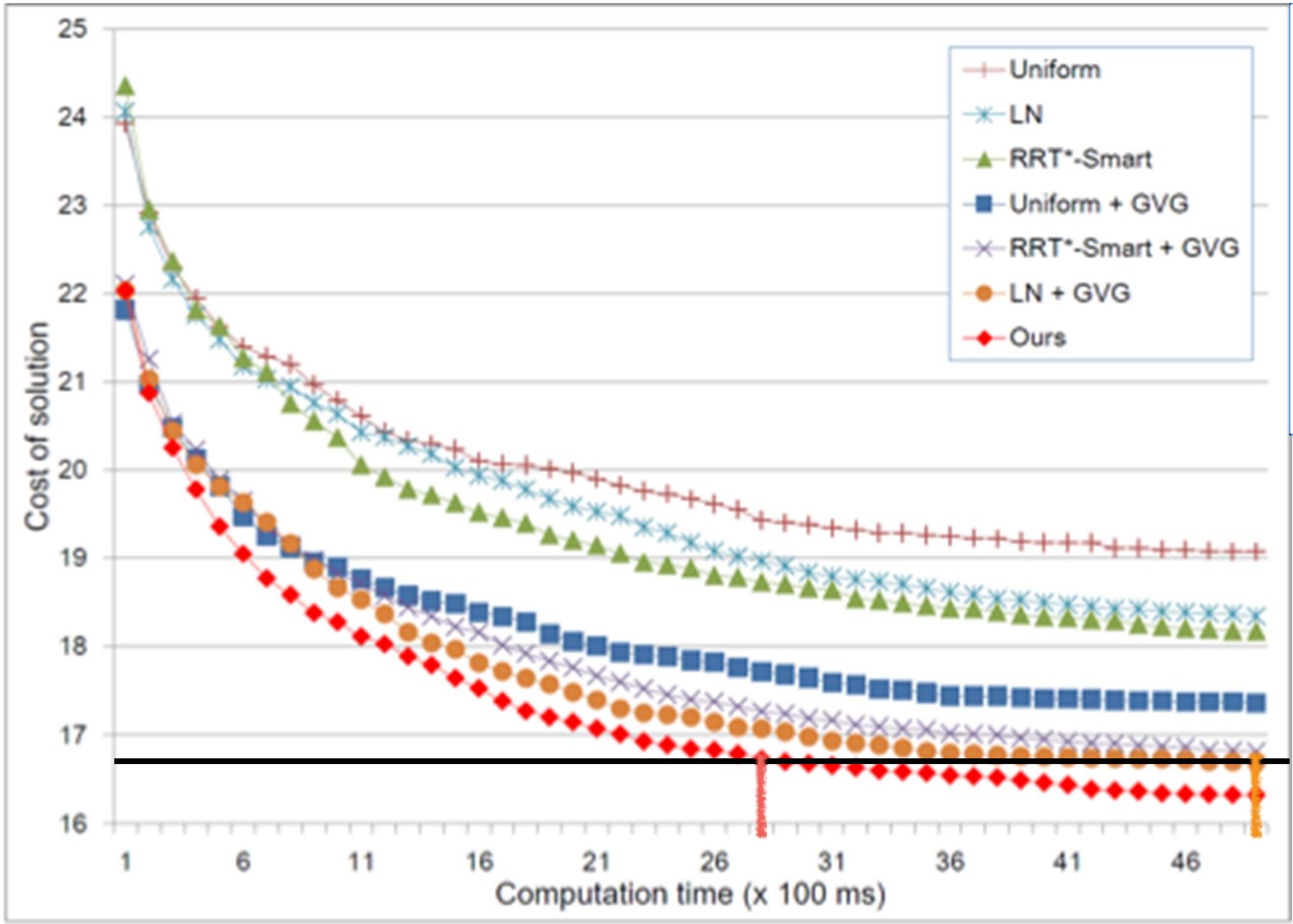
Initial state of sampling cloud

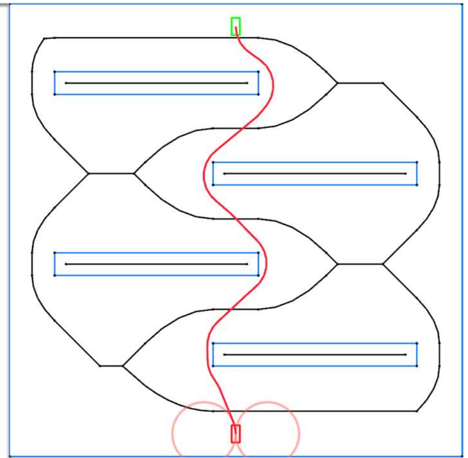


After updated several times

Video

Results: 4 squares





1.8X
improvement

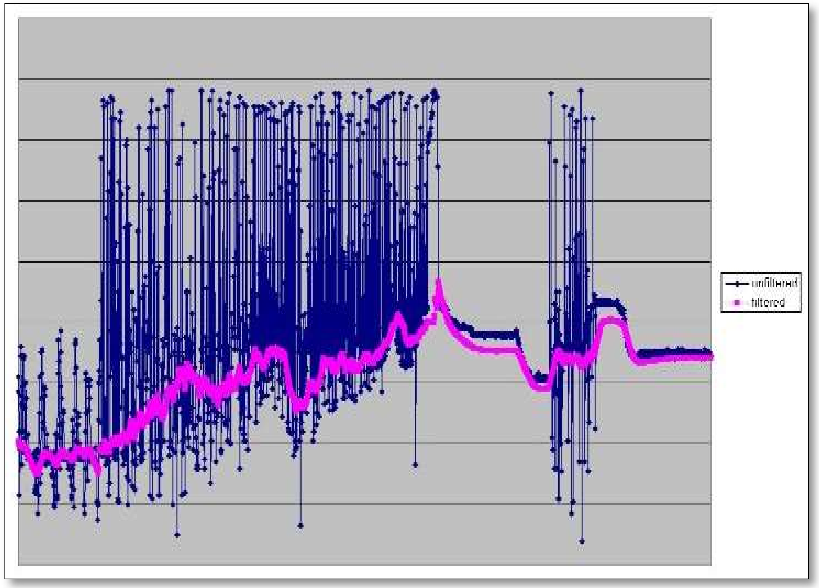
KAIST

Recent Works of Our Group

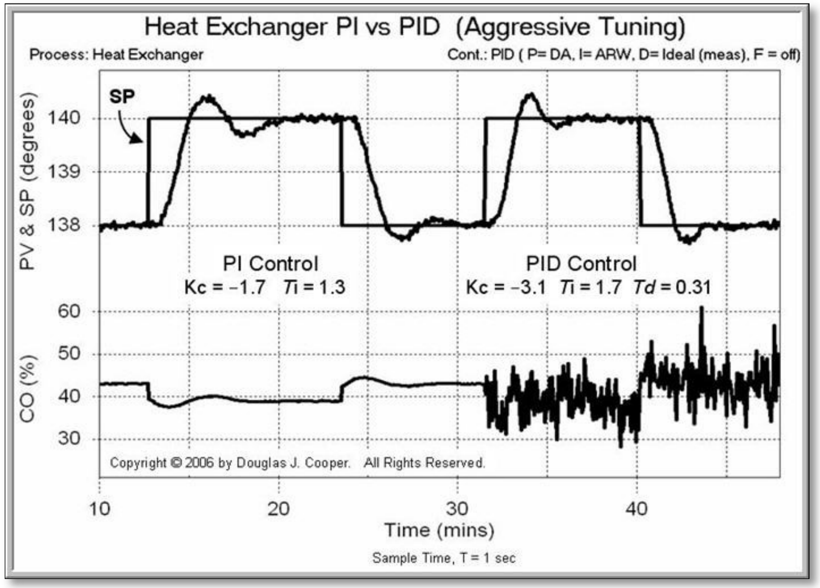
- **Handling narrow passages**
- **Improving low convergence to the optimal solution**
- **Handling uncertainty and dynamic objects**
  - **Anytime RRBT for handling uncertainty and dynamic objects, IROS 16**

Handling Sensor Errors

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



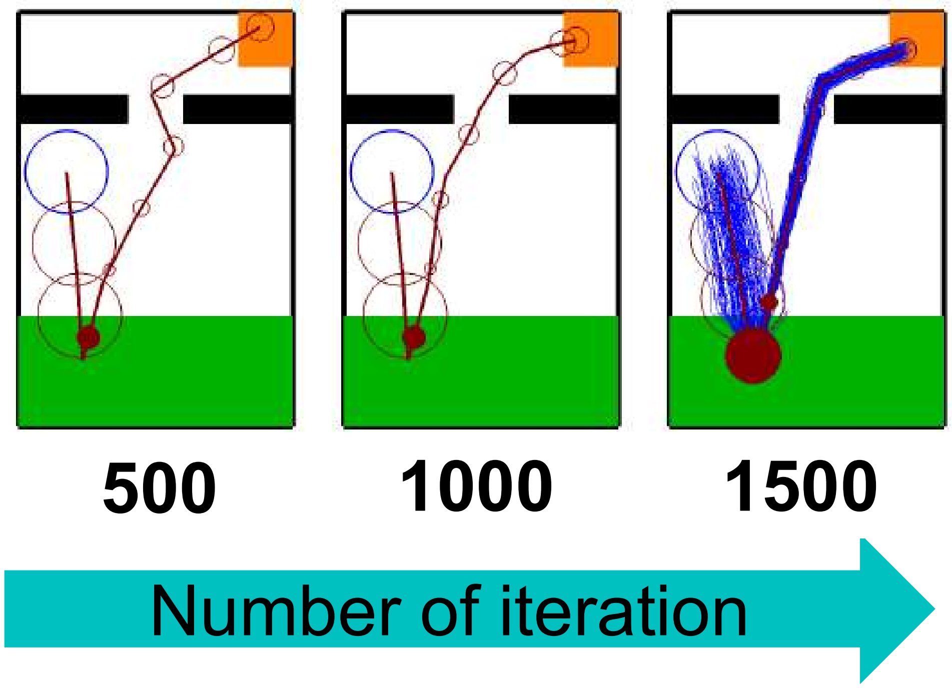
Sensor noise



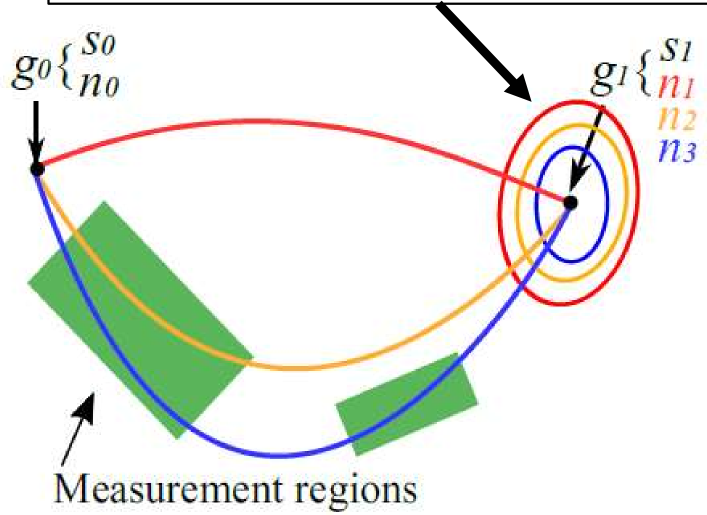
Controller noise

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

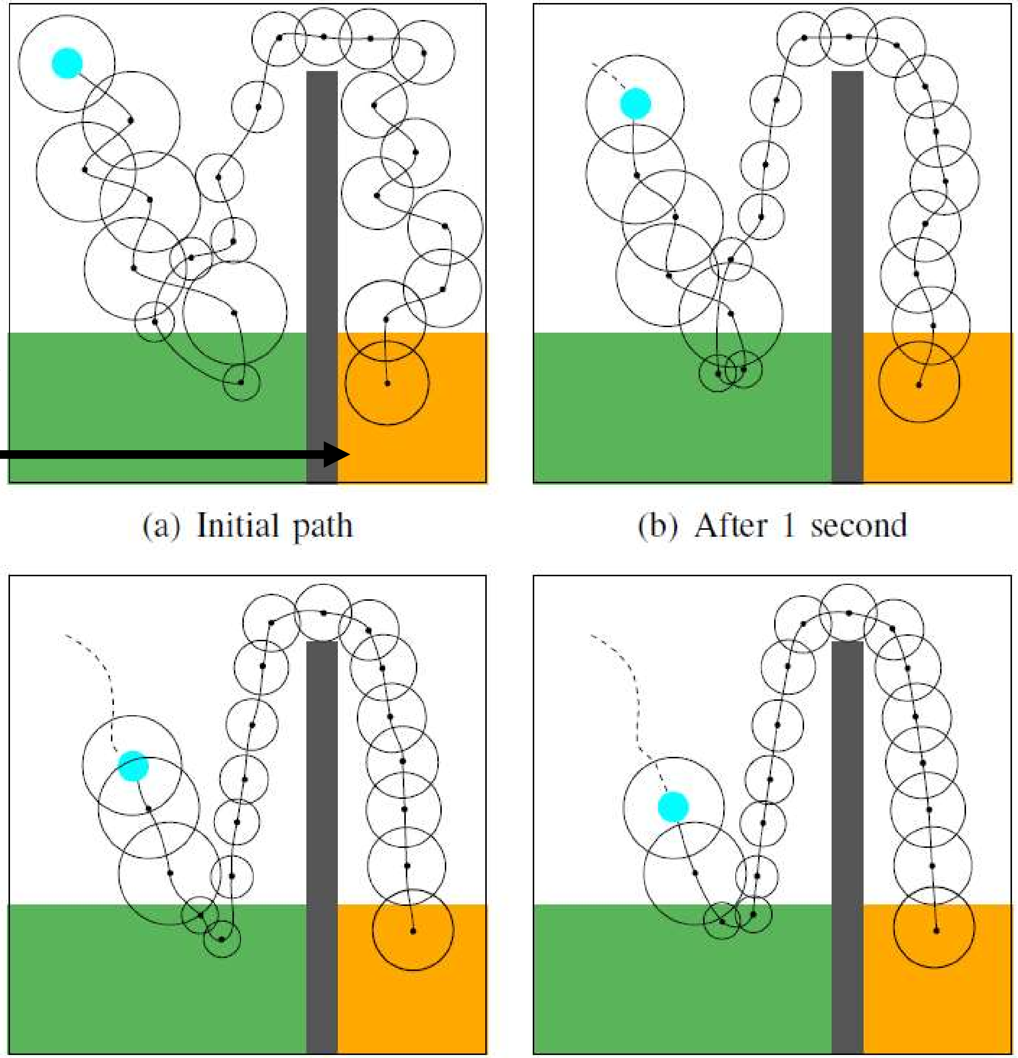


Preserve optimal path

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

Measurement region

Goal region



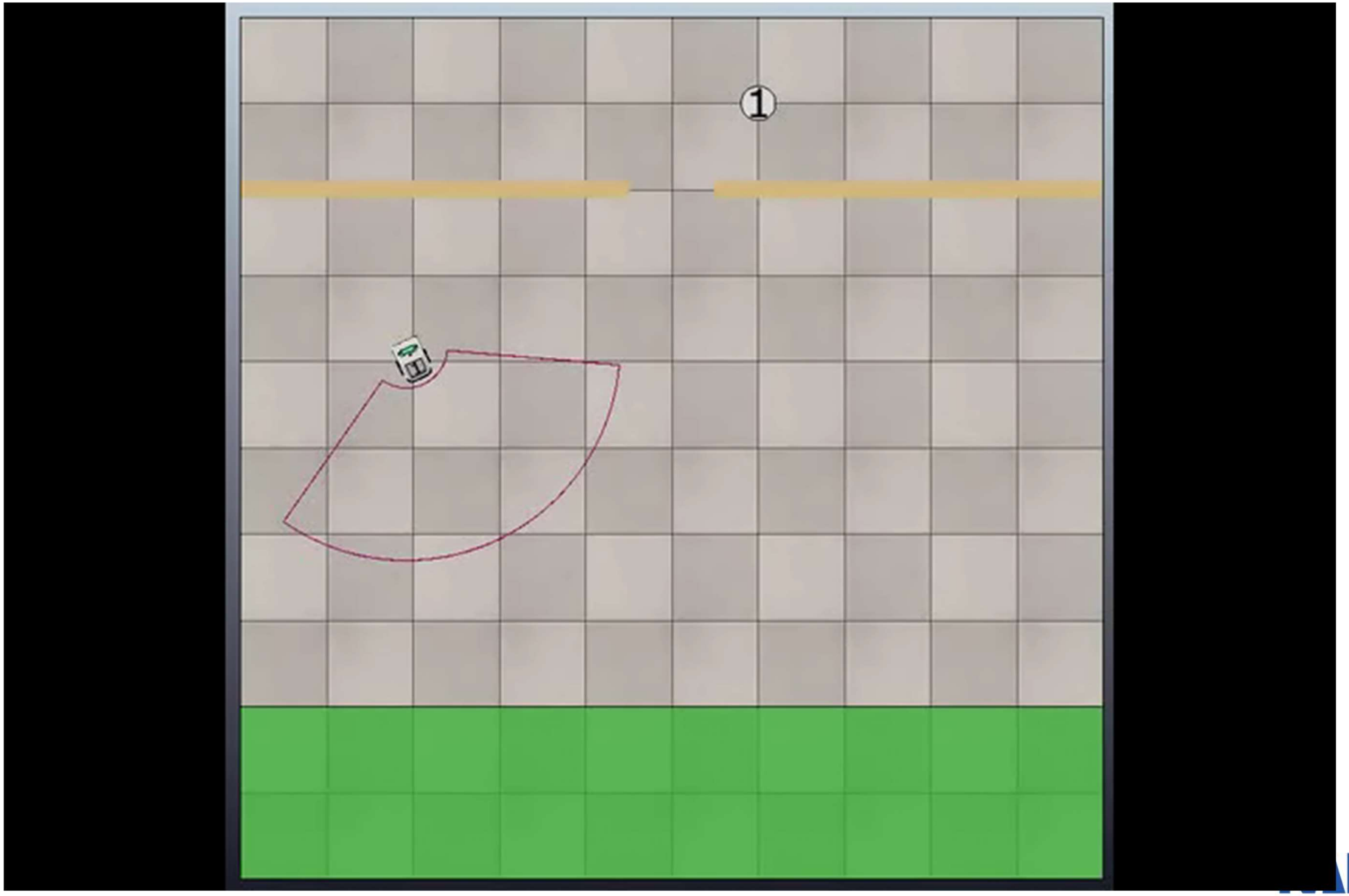
(a) Initial path (b) After 1 second

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

Bigger circle means higher uncertainty

The robot computes better path while executing the path

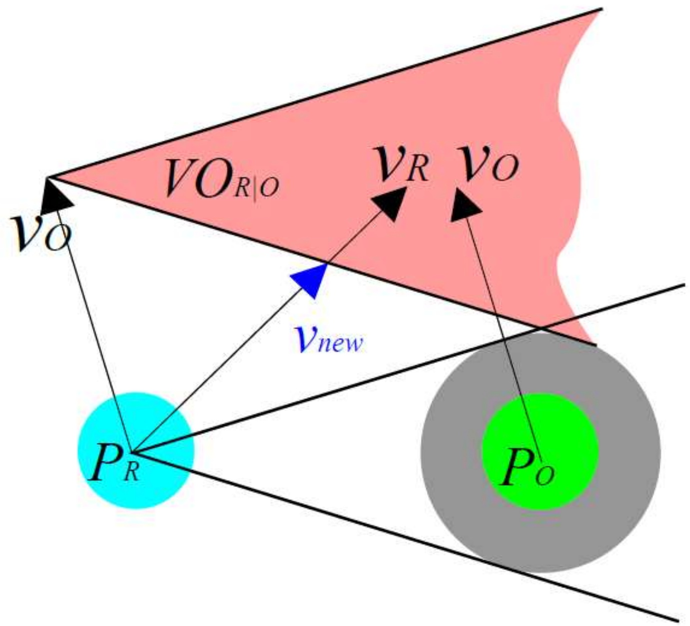
Main Contribution: Anytime Extension



Velocity Obstacle: Local Geometric Analysis

When v for Robot is in the VO, we will have collision

$$VO_{R|O} = \{v \mid \exists t > 0 : t(v - v_O) \in Disc(P_O - P_R, r_R + r_O)\}$$

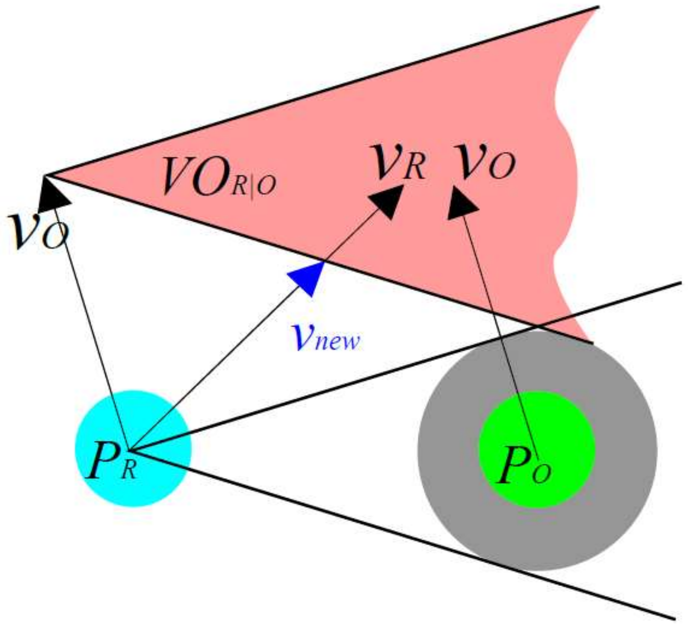


(a) Velocity obstacle

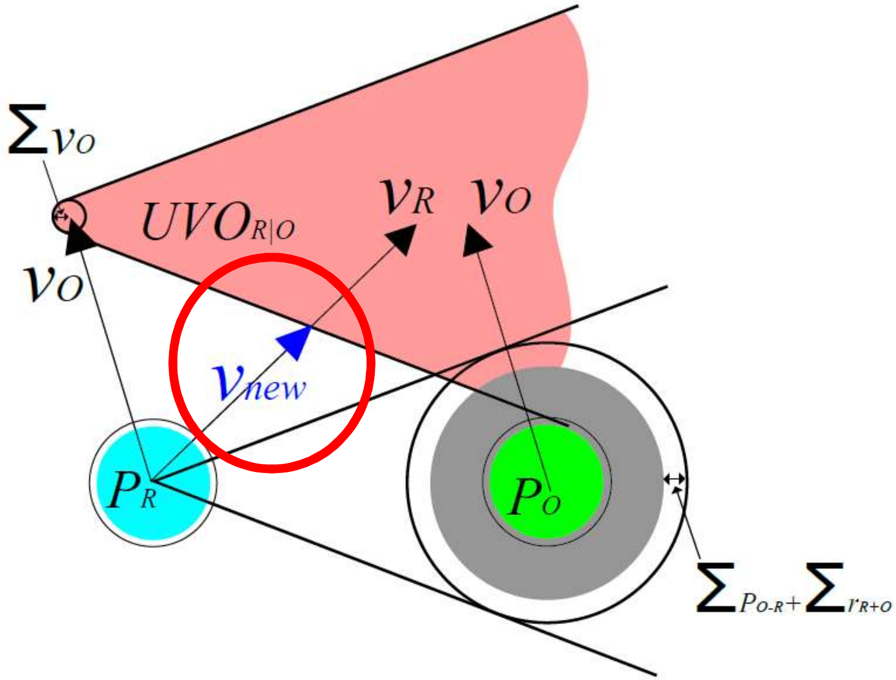
"The hybrid reciprocal velocity obstacle" **TRO11** J Snape, J van den Berg, SJ Guy
"Reciprocal velocity obstacles for real-time multi-agent navigation" J van den Berg
"Generalized Velocity Obstacles" **IROS09**, D Wilkie, J Van den Berg

Uncertainty-aware Velocity Obstacle as Local Geometry Analysis

Conservative collision checking



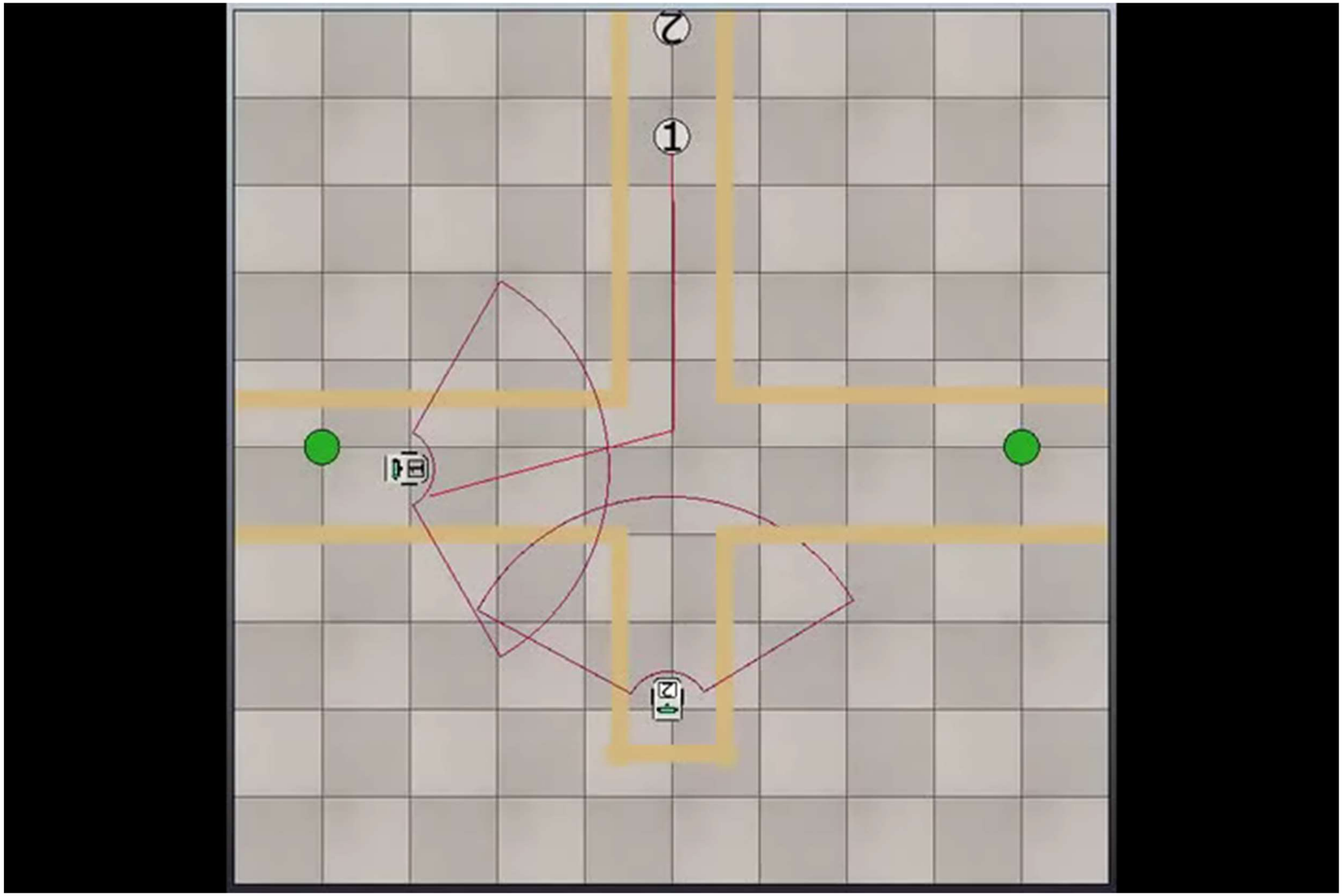
(a) Velocity obstacle



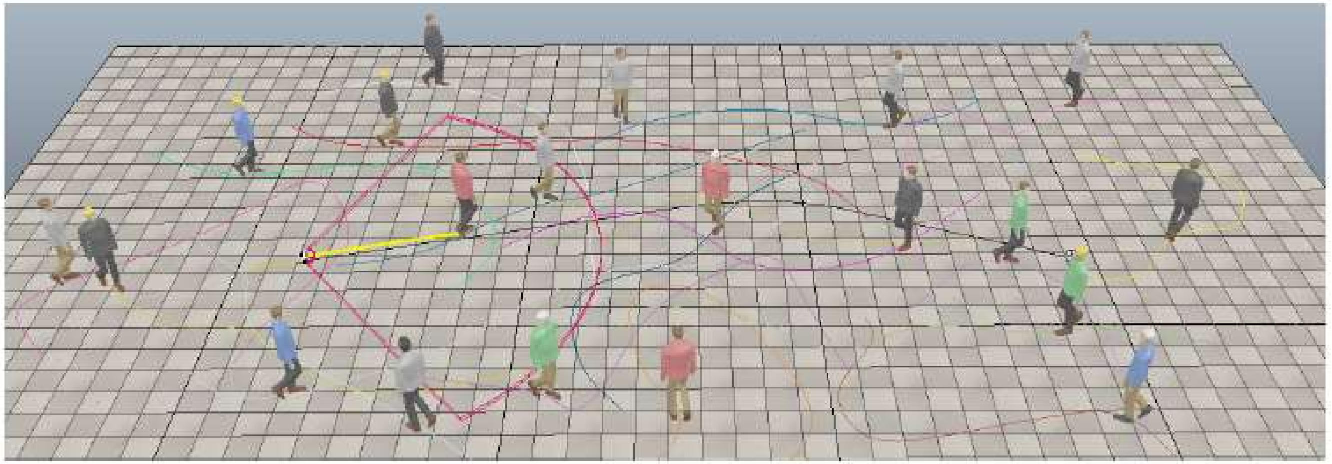
(b) Uncertainty-aware velocity obstacle

"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

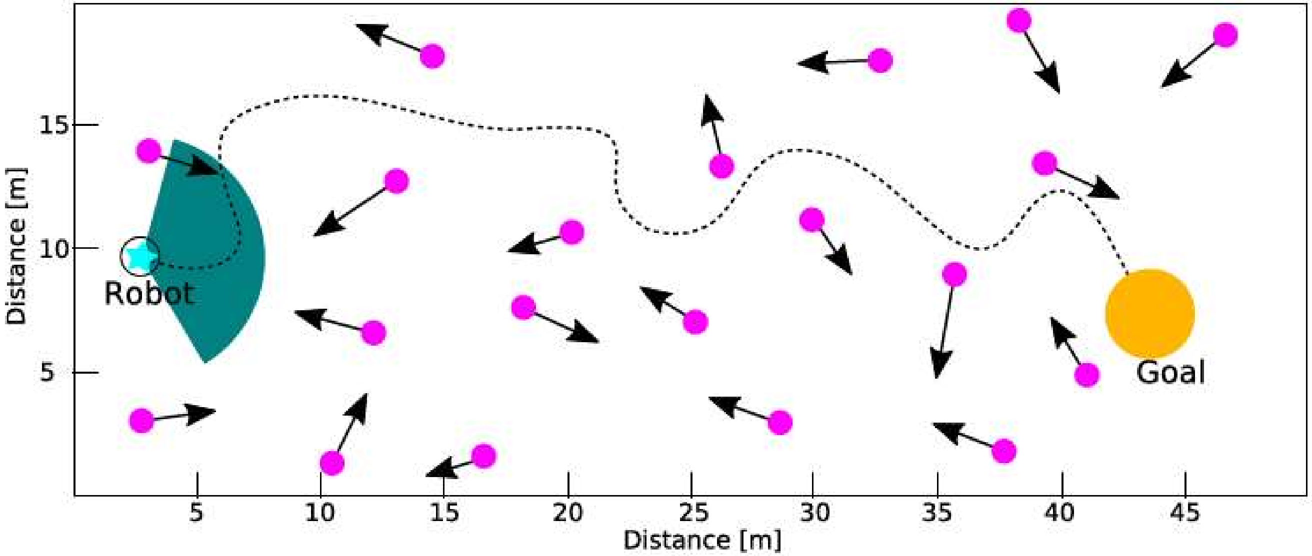
Intersection scene – with UVO



Result – Crowd scene

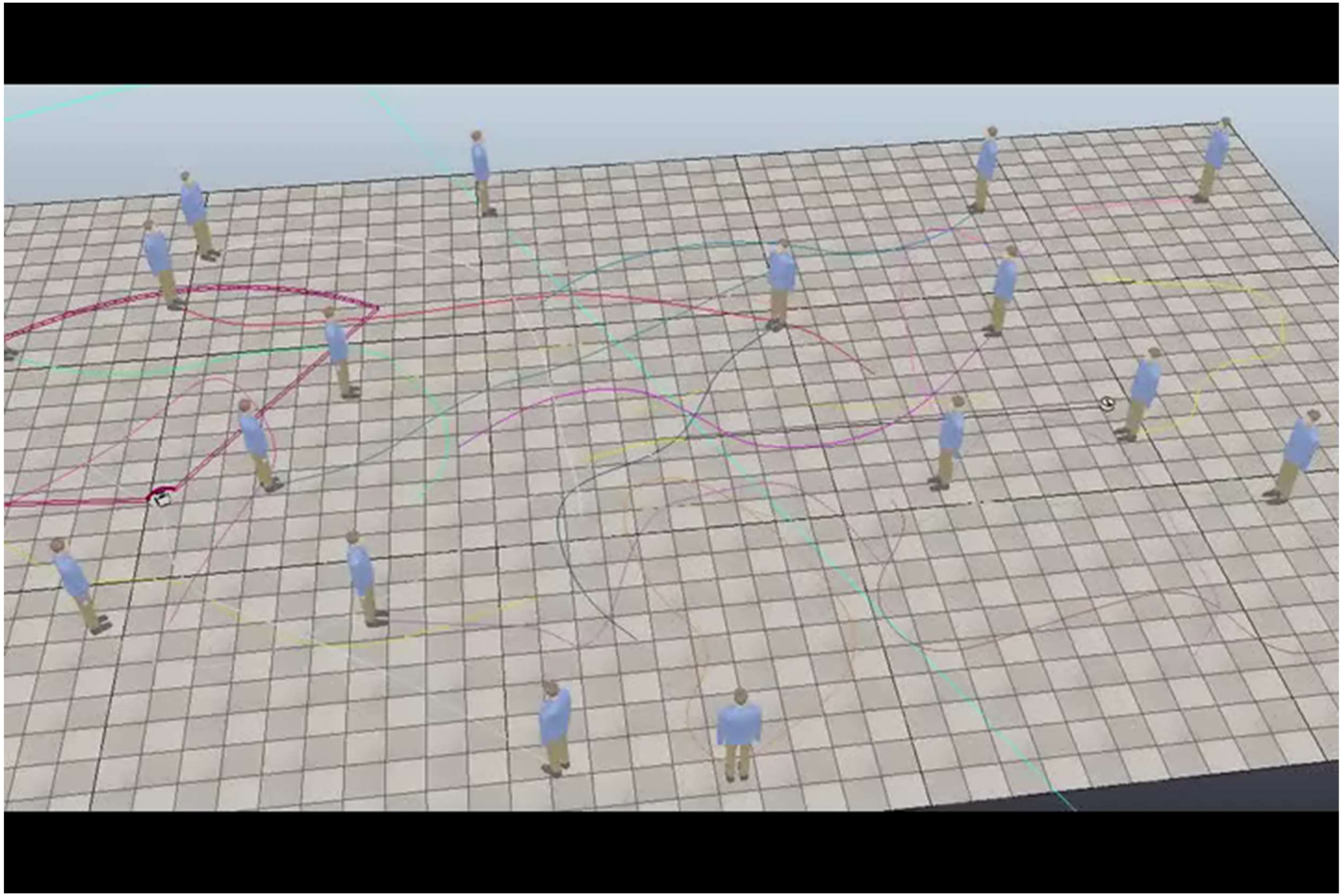


(a)



(b)

Result – Crowd scene



Class Objectives were:

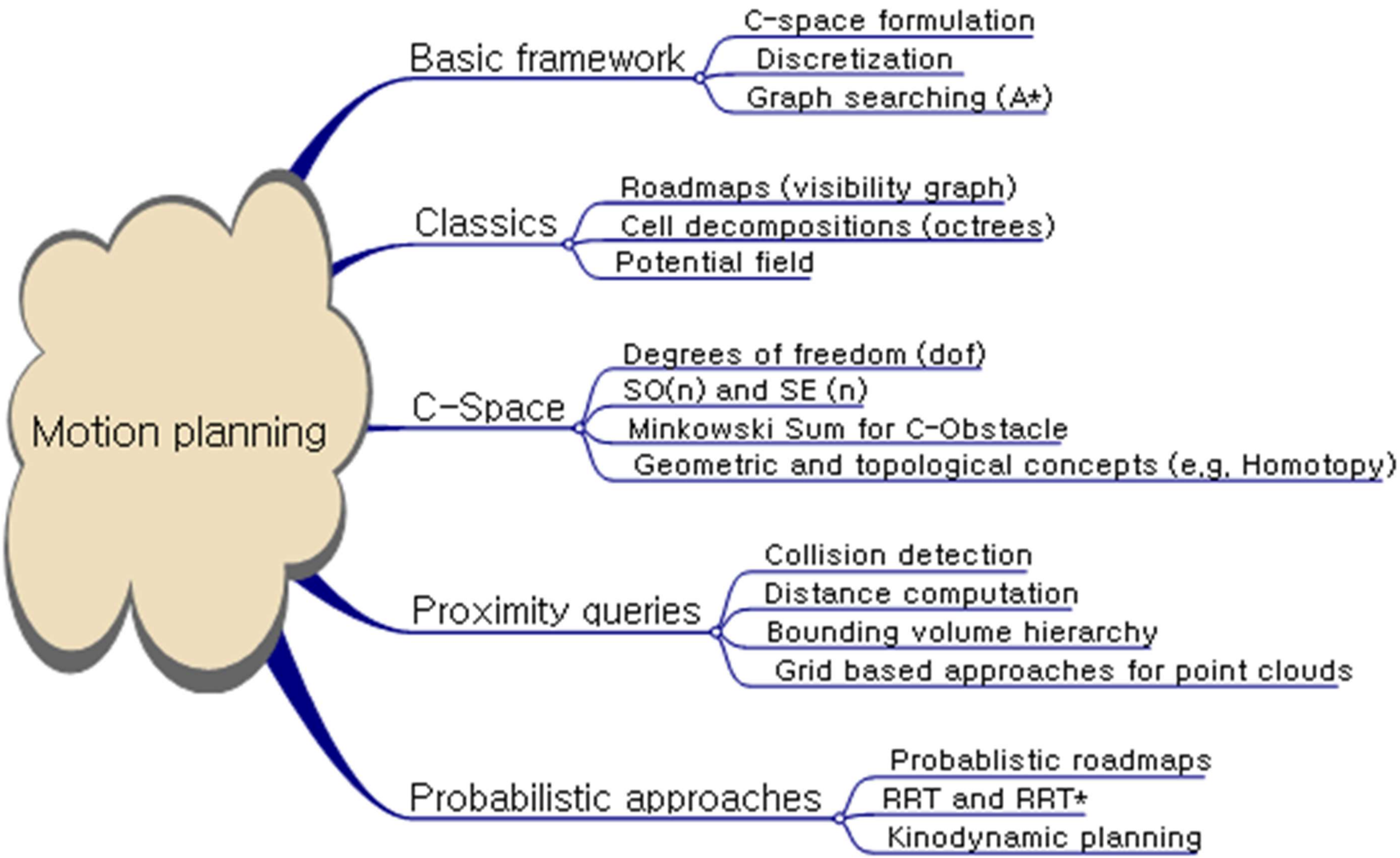
- **Understand the RRT technique and its recent advancements**
  - **RRT* for optimal path planning**
  - **Kinodynamic planning**

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