
RRT and Recent Advancements

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-Connect: An Efficient Approach to Single-Query Path Planning

James Kuffner, Steven LaValle

ICRA 2000

of citation: more 600

Goal

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

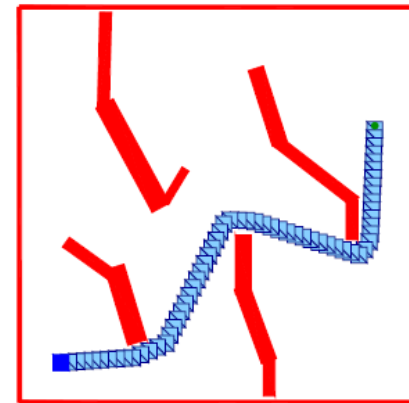
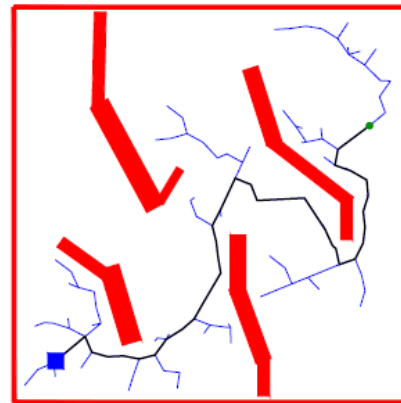
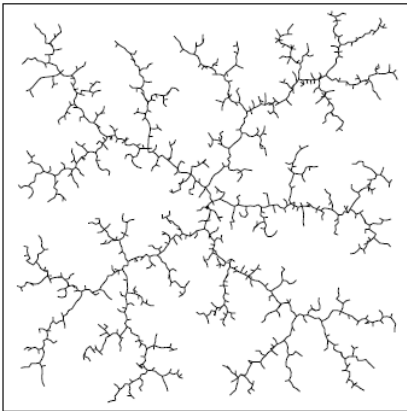


Motivation – Performance vs. Reliability

- **Complete algorithms [Schwartz and M. Sharir 83, Canny 88]**
 - Most reliable, needs high computational power
 - Only used to low-dimensional C-space
- **Randomized potential field [Barraquand and Latombe 91]**
 - Greedy & relaxation approach
 - Fast in many cases, but not in every case
- **Probabilistic roadmap [Kavraki et al. 96]**
 - Reliable, but needs preprocessing
 - Good for multiple-query problems

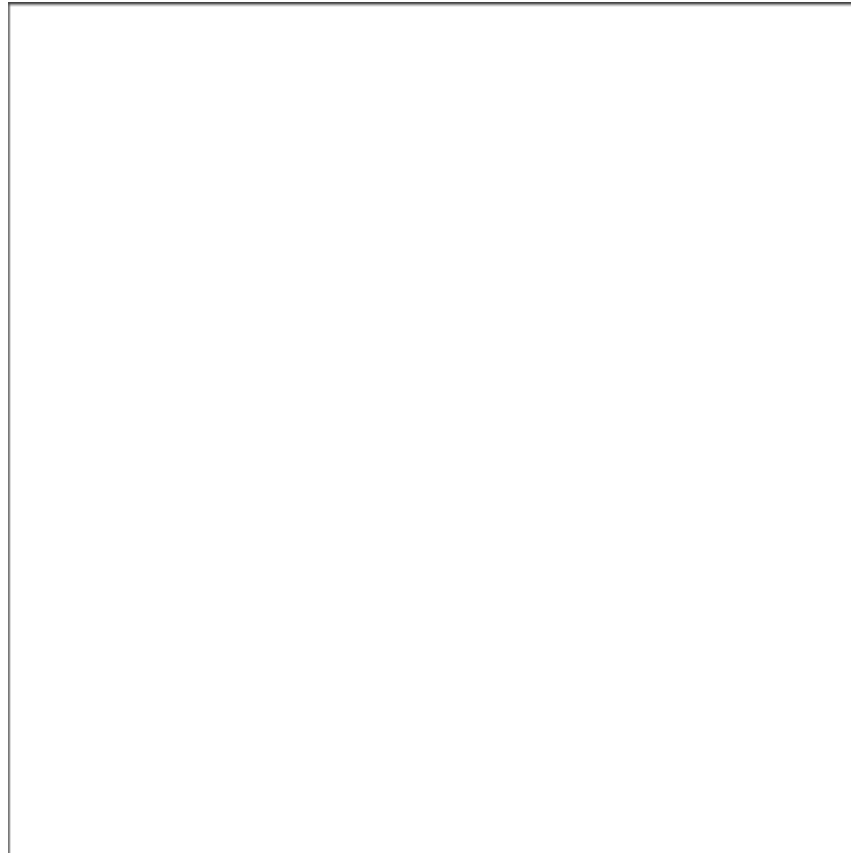
Approach

- Design a simple, reliable, and fast algorithm for single-query problems
 - Use RRT (Rapidly-exploring Random Trees) [LaValle 98] for reliability
 - Develop a greedy heuristic to converge quickly



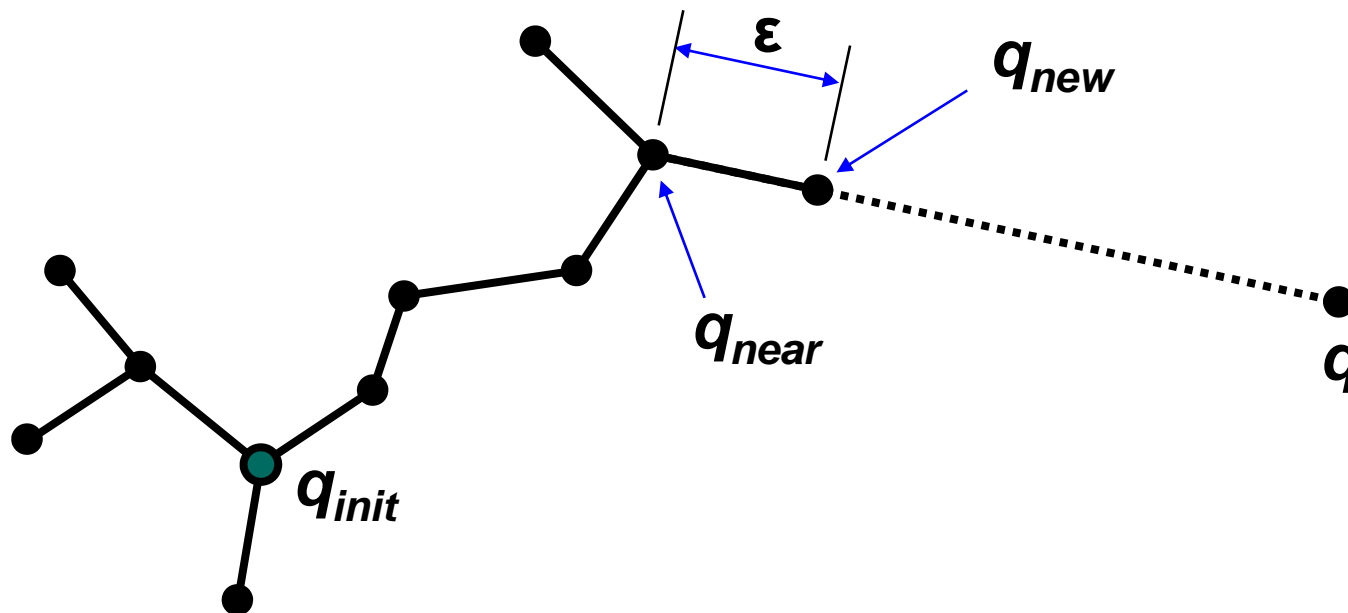
Rapidly-Exploring Random Tree

- A growing tree from an initial state



RRT Construction Algorithm

- Extend a new vertex in each iteration



Advantages of RRT

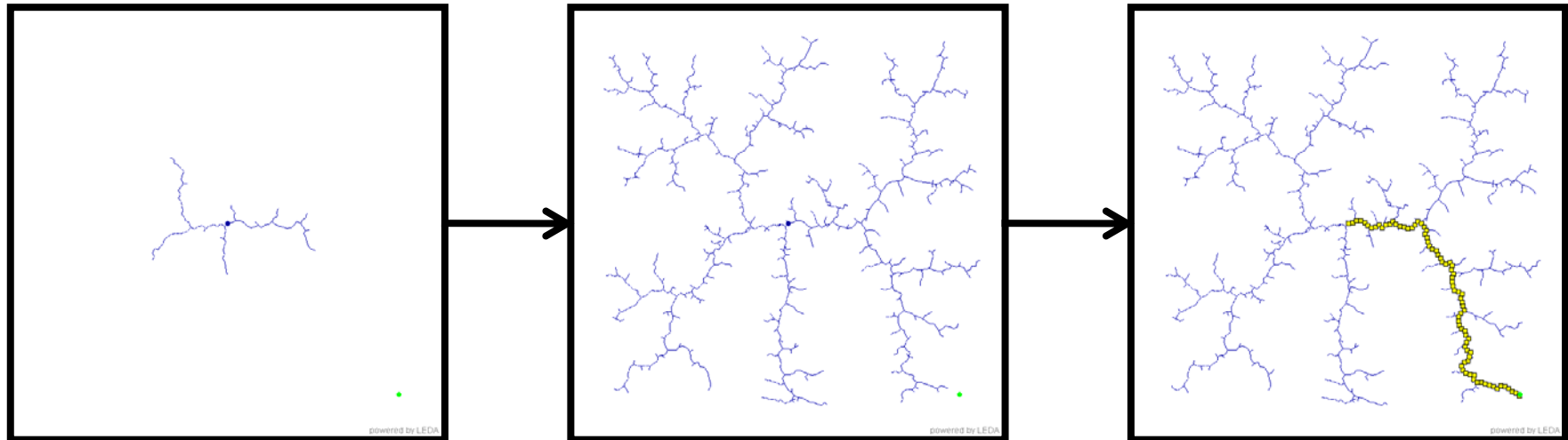
- Biased toward unexplored space
- Probabilistically complete
- Always connected
- Can handle nonholonomic constraints and high degrees of freedom

Outline

- Introduction
- Rapidly-exploring Random Tree
- Overview
- **RRT-Connect Algorithm**
- Demo
- Results
- Conclusion

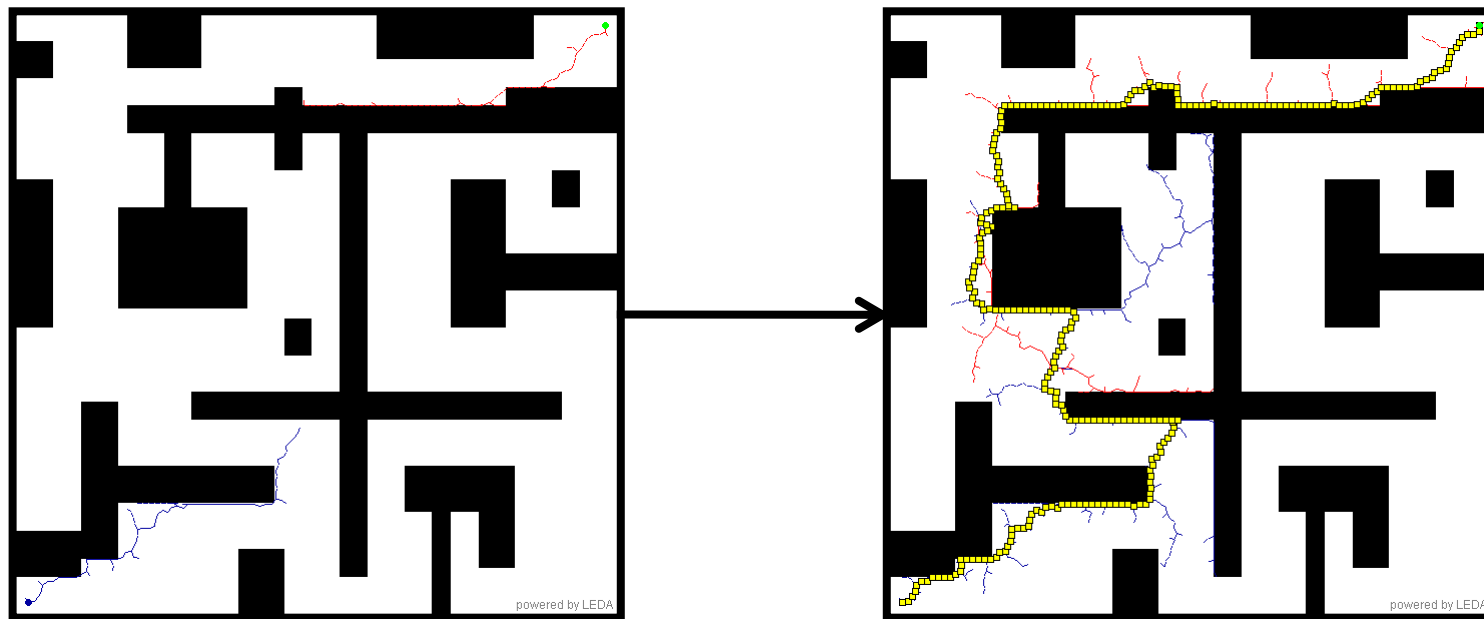
Overview – Planning with RRT

- Extend RRT until a nearest vertex is close enough to the goal state
- Probabilistically complete, but converge slowly



Overview – With Dual RRT

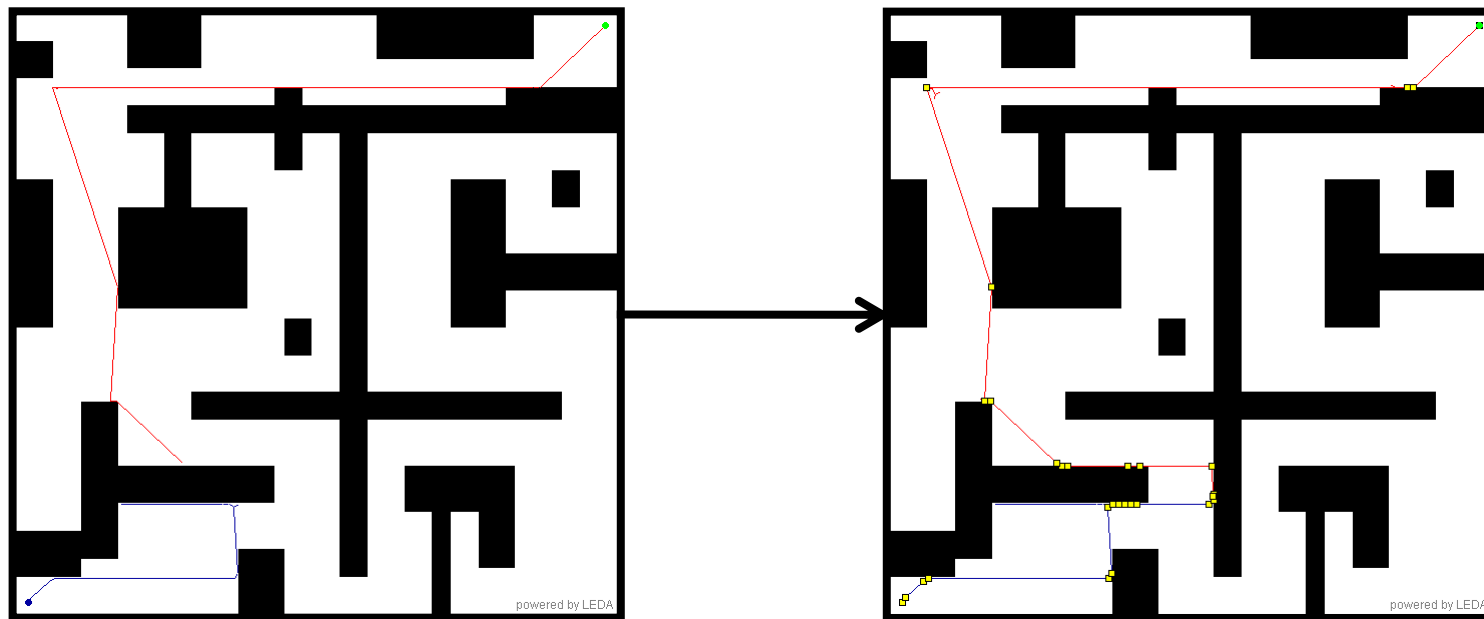
- Extend RRTs from both initial and goal states
- Find path much more quickly



737 nodes are used

Overview – With RRT-Connect

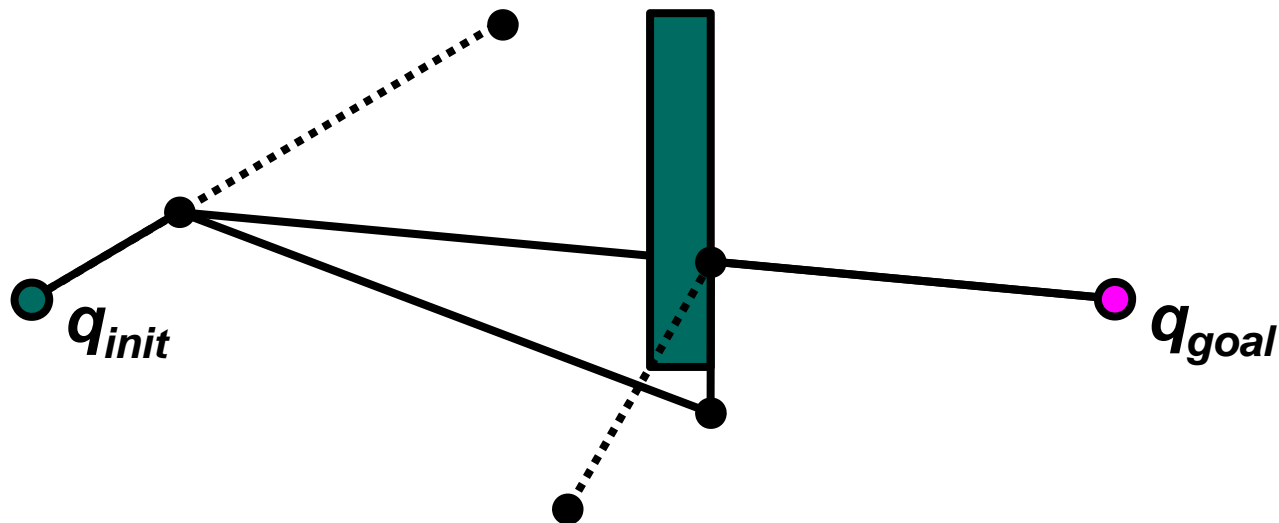
- Aggressively connect the dual trees using a greedy heuristic
- Extend & connect trees alternatively



42 nodes are used

RRT-Connect Algorithm

- Starting from both initial and goal states
- Extend a tree and try to connect the new vertex and another tree
- Alternatively repeat until two trees are actually connect

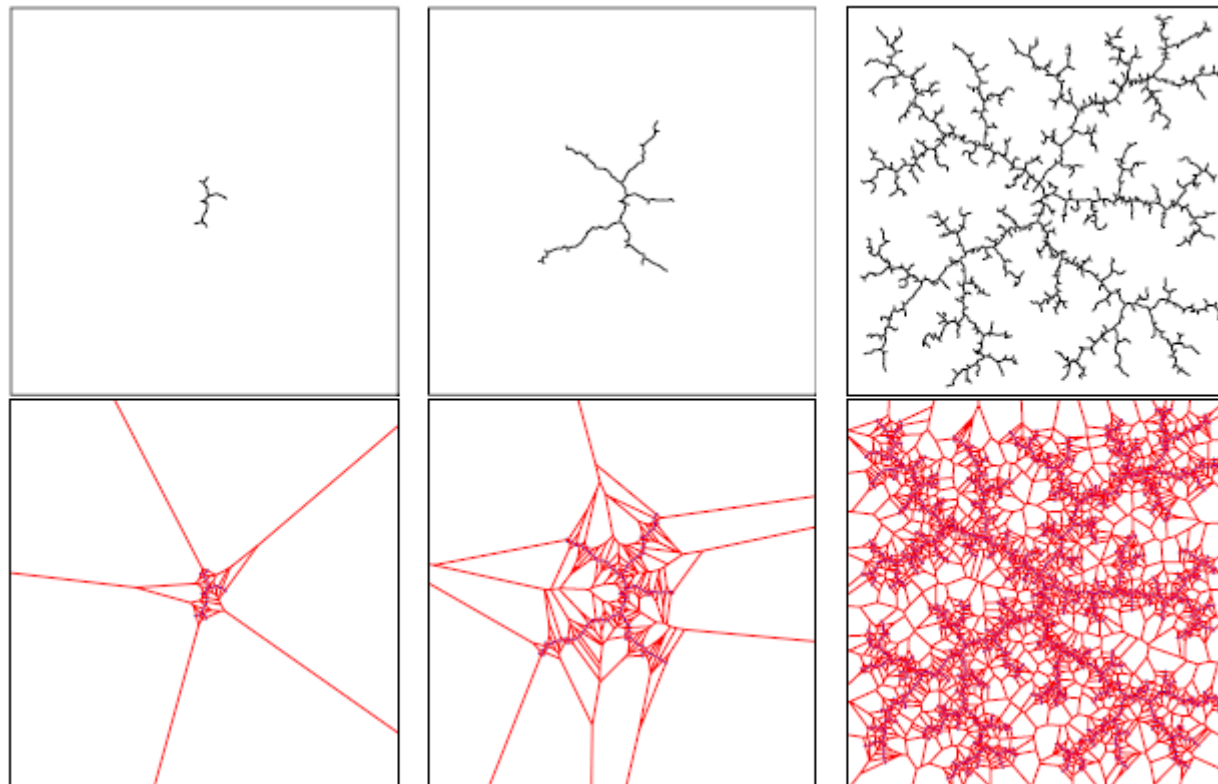


Variations of RRT-Connect

- **Extend & Extend**
 - Less aggressive, but works well on nonholonomic constrains
- **Connect & Connect**
 - Stronger greedy

Voronoi Region

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



RRT Construction Algorithm

```
BUILD_RRT( $q_{init}$ )
1   $\mathcal{T}$ .init( $q_{init}$ );
2  for  $k = 1$  to  $K$  do
3       $q_{rand} \leftarrow \text{RANDOM\_CONFIG}()$ ;
4      EXTEND( $\mathcal{T}, q_{rand}$ );
5  Return  $\mathcal{T}$ 
```

```
EXTEND( $\mathcal{T}, q$ )
1   $q_{near} \leftarrow \text{NEAREST\_NEIGHBOR}(q, \mathcal{T})$ ;
2  if NEW_CONFIG( $q, q_{near}, q_{new}$ ) then
3       $\mathcal{T}$ .add_vertex( $q_{new}$ );
4       $\mathcal{T}$ .add_edge( $q_{near}, q_{new}$ );
5      if  $q_{new} = q$  then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

RRT Connect Algorithm

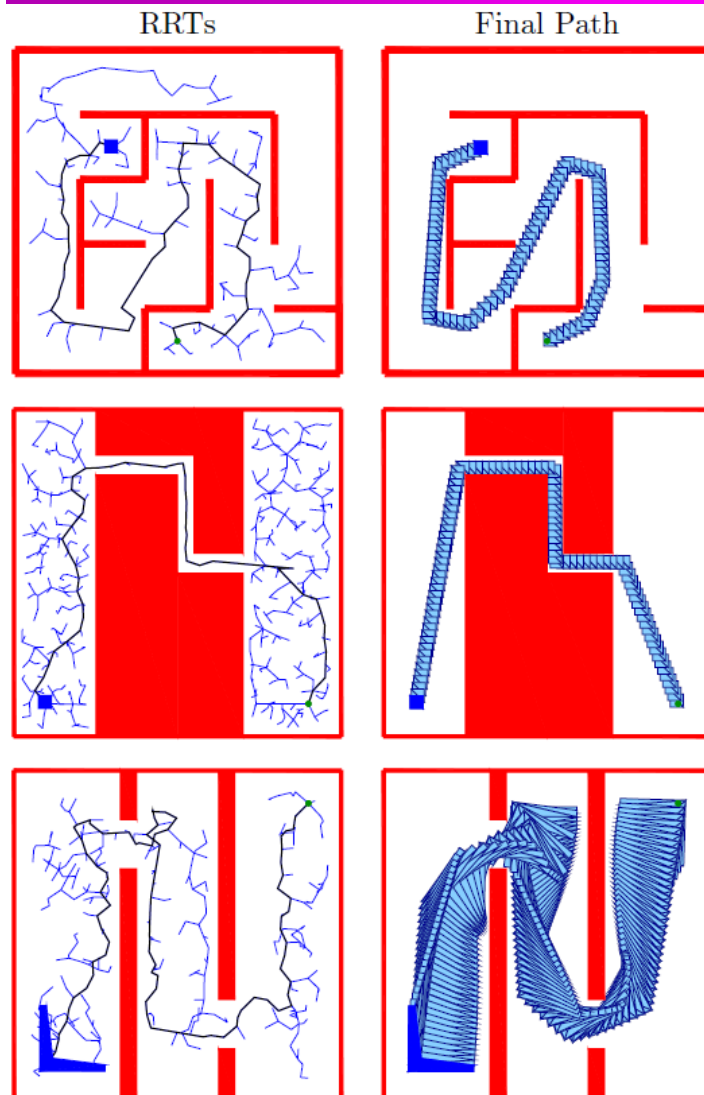
CONNECT(\mathcal{T}, q)

```
1  repeat
2       $S \leftarrow \text{EXTEND}(\mathcal{T}, q)$ ;
3  until not ( $S = \text{Advanced}$ )
4  Return  $S$ ;
```

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

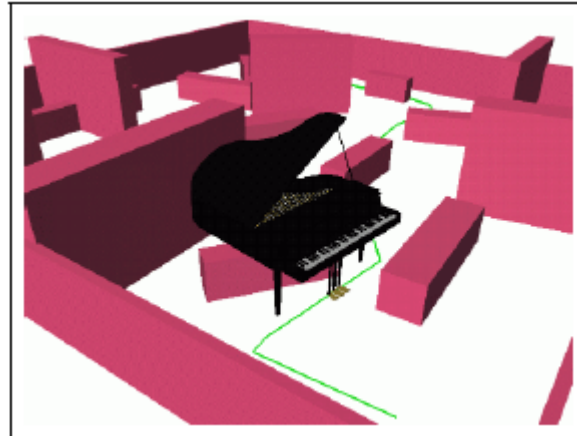
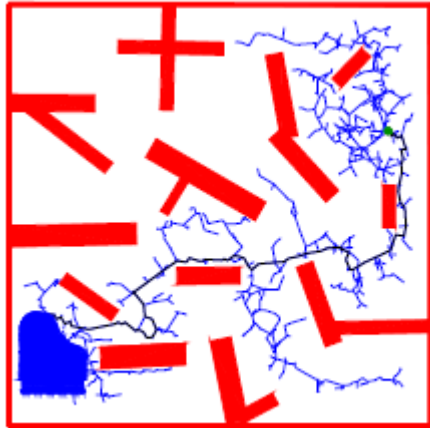
```
1   $\mathcal{T}_a.\text{init}(q_{init})$ ;  $\mathcal{T}_b.\text{init}(q_{goal})$ ;
2  for  $k = 1$  to  $K$  do
3       $q_{rand} \leftarrow \text{RANDOM\_CONFIG}()$ ;
4      if not ( $\text{EXTEND}(\mathcal{T}_a, q_{rand}) = \text{Trapped}$ ) then
5          if ( $\text{CONNECT}(\mathcal{T}_b, q_{new}) = \text{Reached}$ ) then
6              Return  $\text{PATH}(\mathcal{T}_a, \mathcal{T}_b)$ ;
7      SWAP( $\mathcal{T}_a, \mathcal{T}_b$ );
8  Return Failure
```

Results



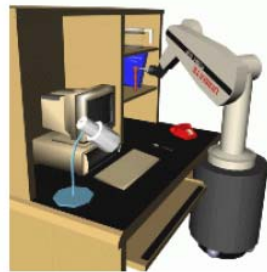
- 0.13s, 1.52s, and 1.02s on 270MHz
- Improves performance by a factor of three or four in uncluttered environments
- Slightly improves in very cluttered environments

Results



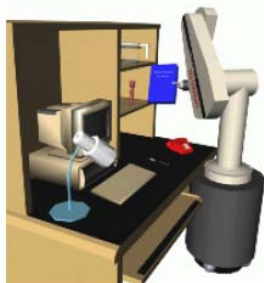
- Translations & rotations

- 12s



- 6-DOF

- 4s



Conclusions

- Reasonably balanced path planning between greedy exploration (as in a potential field) and uniform exploration (as in a probabilistic roadmap)
- Simple and practical method
- The huge performance improvements happen in relatively open spaces only
- Theoretical convergence ratio is not given

Randomized Kinodynamic Planning

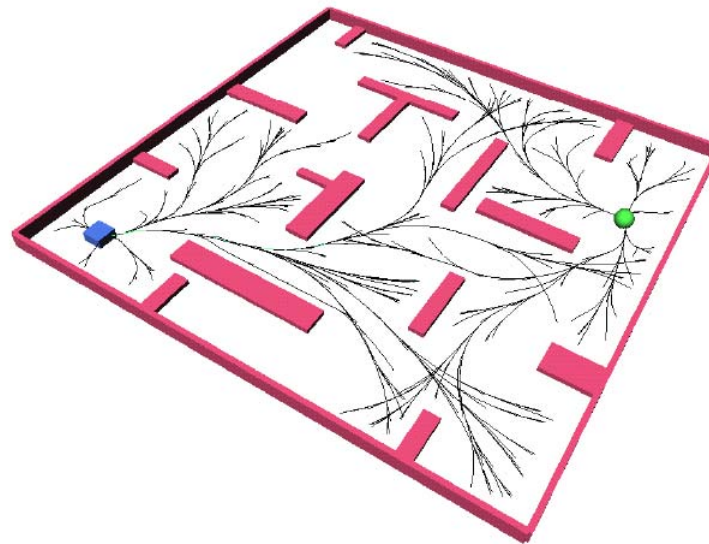
Steven LaValle James Kuffner

ICRA 1999

of citation: more than 400

Goal

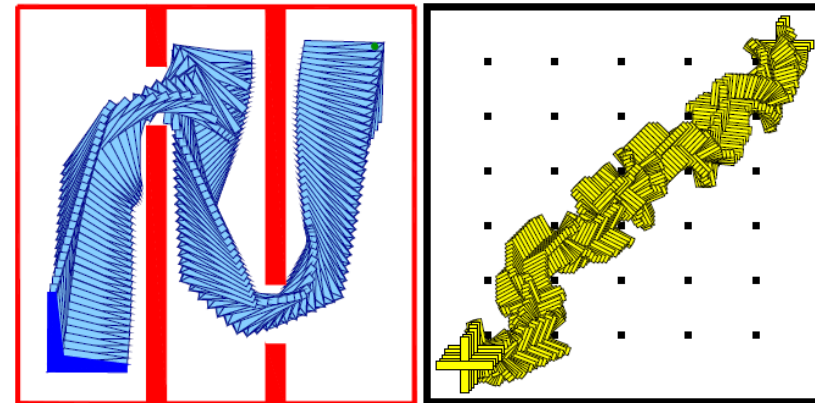
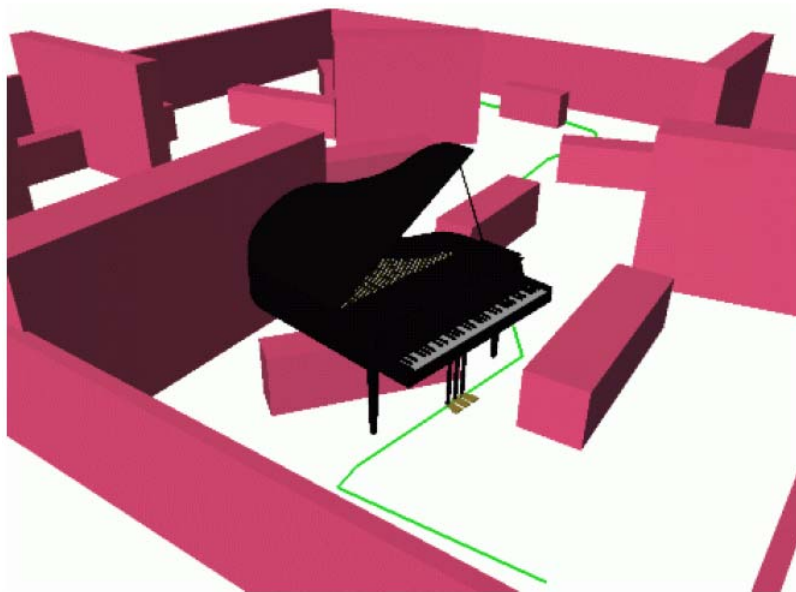
- Present an efficient randomized path planning algorithm on the kinodynamic planning problem



Holonomic Path Planning

GIVEN: \mathcal{A} (robot), \mathcal{C} (C-space), $f_i(q) \leq 0$ (obstacles), ...

FIND: Continuous path, τ , that satisfies $f_i(q)$ constraints

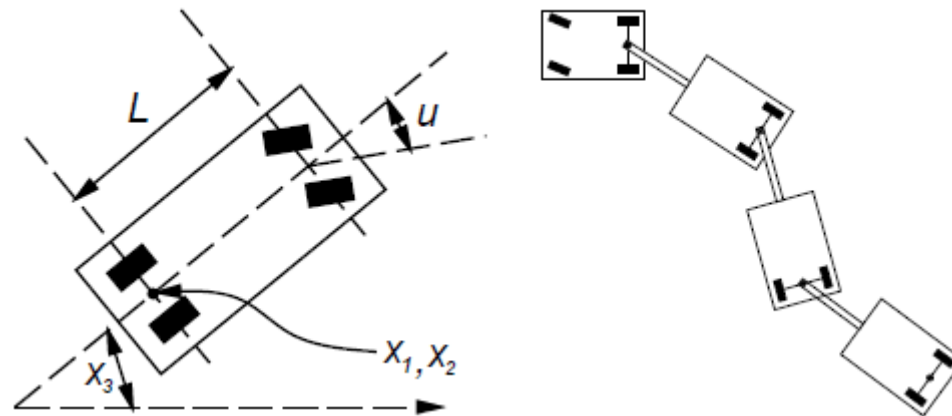


Nonholonomic Path Planning

ALSO GIVEN: $g_i(q, \dot{q}) \leq 0, g_i(q, \dot{q}) = 0, \dots$

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

- Consider kinematic constraints

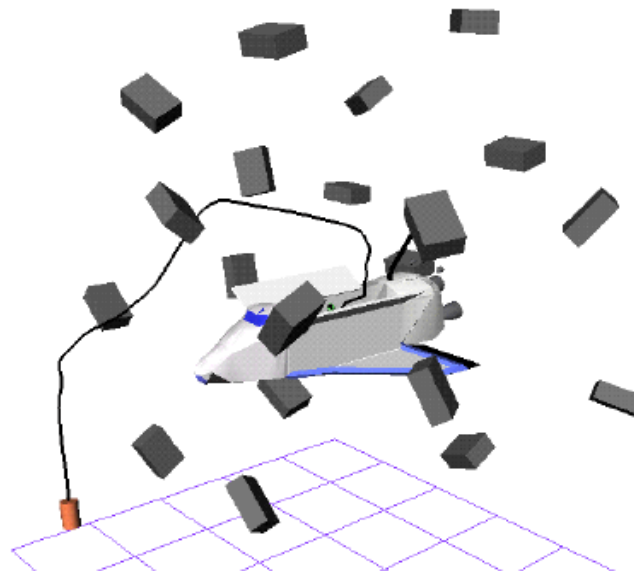


Kinodynamic Path Planning

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

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

- Consider kinematic + dynamic constraints



Kinodynamic Path Planning

- **Conventional planning: Decouple problems**
 - Solve basic path planning
 - Find trajectory and controller that satisfies the dynamics and follows the path
 - [Bobrow et al. 85, Latombe 91, Shiller and Dubowsky 91]
- **PSPACE-hard in general [Reif 79]**

Outline

- Introduction
- Kinodynamic Planning
- **Problem Formulation**
- **Randomized Kinodynamic Planning**
- **Rapidly-Exploring Random Trees (RRTs)**
- Demo
- Results
- Conclusion

State Space Formulation

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

C denote the C -space

X denote the state space

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

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

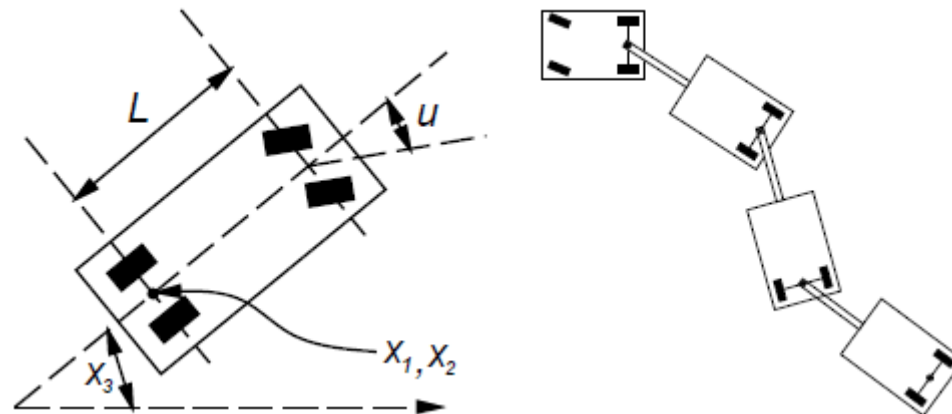
Constraints in State Space

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

- Constraints can be written in:

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

$u \in U$, U : 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**

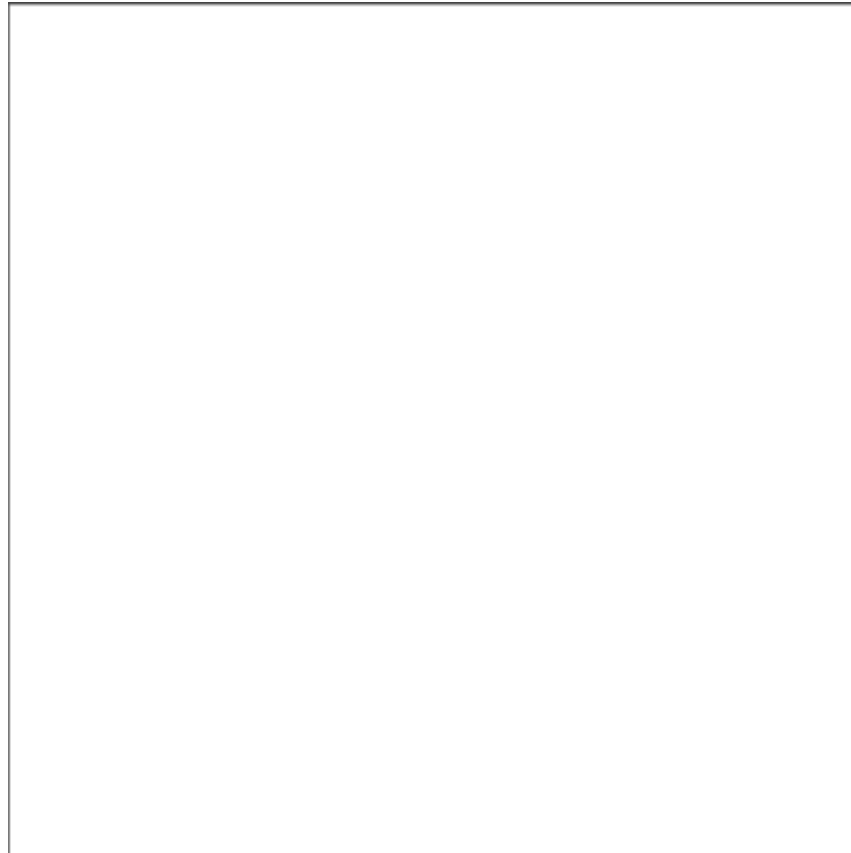
$$u : [0, T] \rightarrow U$$

Randomized Kinodynamic Planning

- **Randomized potential fields**
 - [Barraquand and Latombe 91, Challou et al. 95]
 - Set u which reduces the potential
 - Leads oscillations
 - Hard to design good potential fields
- **Randomized roadmap**
 - [Amato and Wu 96, Kavraki et al. 96]
 - Hard to connect two configurations (or states), except for specific environments [Svestka and Overmars 95, Reeds and Schepp 90, Bushnell et al. 95...]

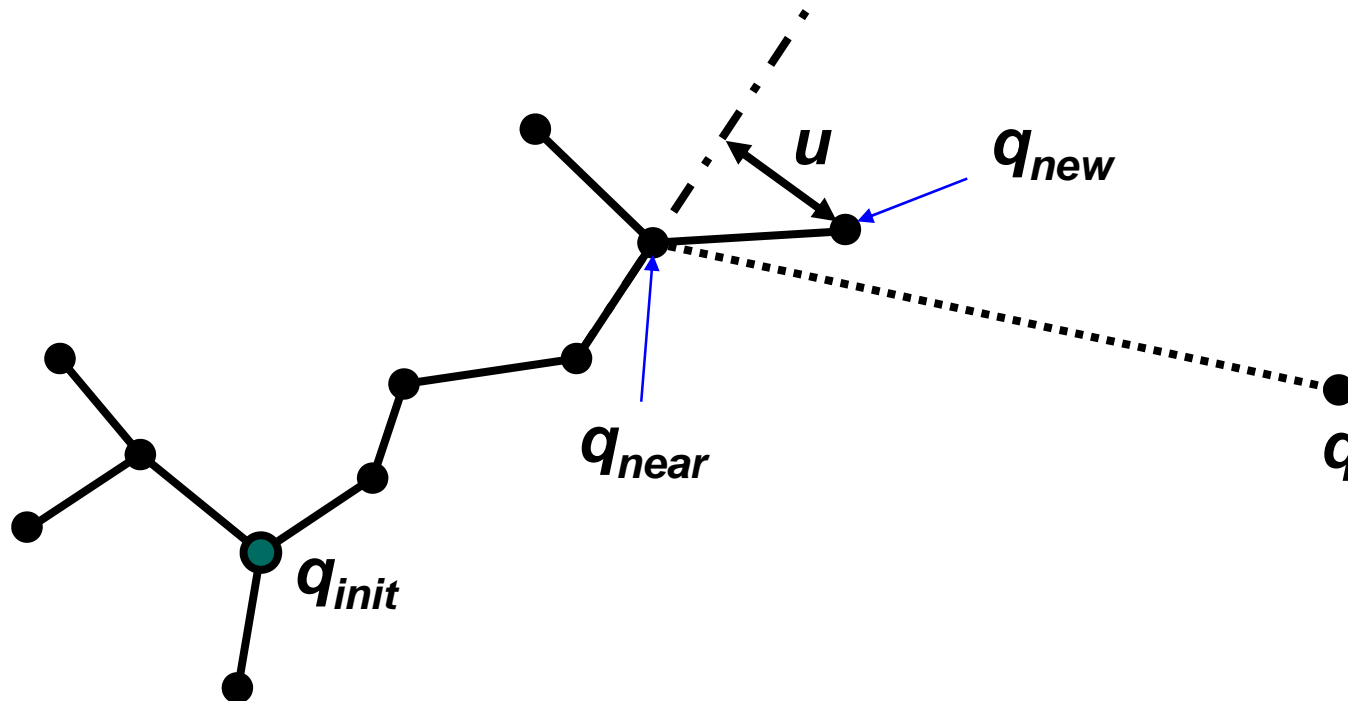
Rapidly-Exploring Random Tree

- A growing tree from initial state



Rapidly-Exploring Random Tree

- Extend a new vertex in each iteration

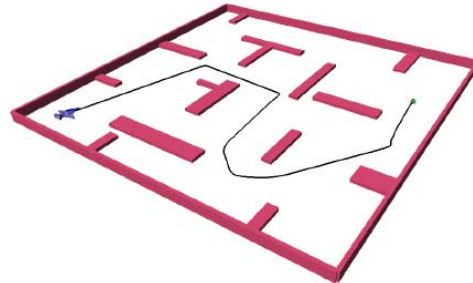
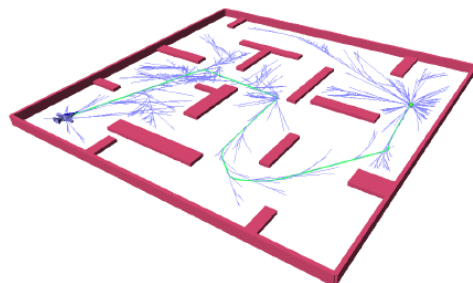
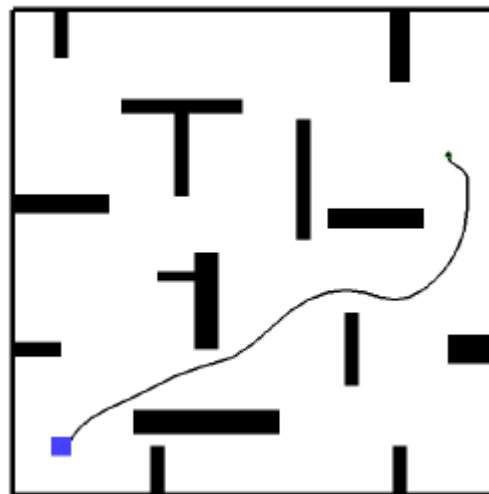
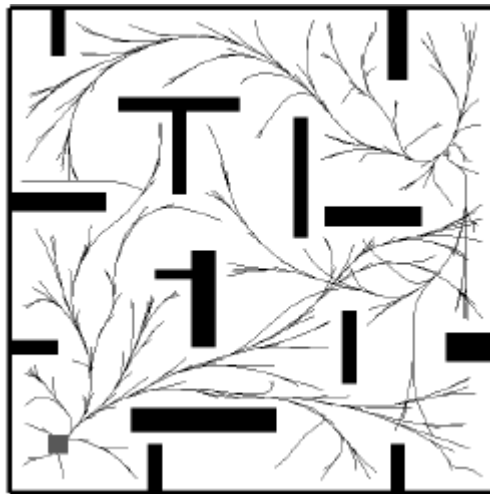


Planning with RRTs

- Use dual RRTs from both initial and goal states
- Extend & Extend (No connect heuristic)

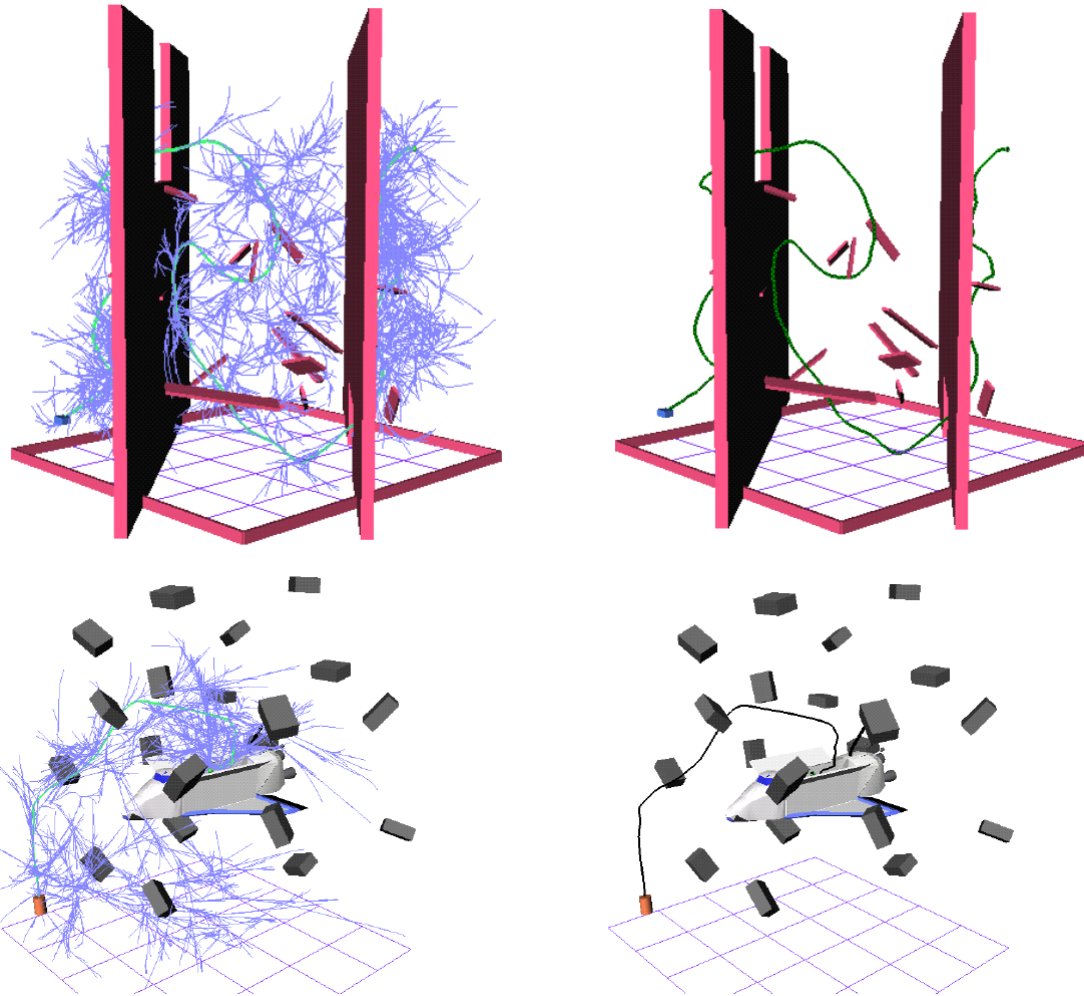
```
GROW_RANDOM_TREES()
1  InsertState( $\mathcal{T}_{init}$ , nil,  $x_{init}$ );
2  InsertState( $\mathcal{T}_{goal}$ , nil,  $x_{goal}$ );
3  while continuePlan do
4       $x_{rand} \leftarrow$  RandomState();
5       $x_i \leftarrow$  GEN_STATE( $\mathcal{T}_{init}$ ,  $x_{rand}$ , FWD);
6      if  $x_i \neq nil$  and NearbyState( $\mathcal{T}_{goal}$ ,  $x_i$ ) then
7          record candidate solution  $\tau$  connecting
               $\mathcal{T}_{init}$  and  $\mathcal{T}_{goal}$  through  $x_i$ ;
8       $x_g \leftarrow$  GEN_STATE( $\mathcal{T}_{goal}$ ,  $x_{rand}$ , BACK);
9      if  $x_g \neq nil$  and NearbyState( $\mathcal{T}_{init}$ ,  $x_g$ ) then
10         record candidate solution  $\tau$  connecting
               $\mathcal{T}_{init}$  and  $\mathcal{T}_{goal}$  through  $x_g$ ;
```

Results – 200MHz, 128MB



- Planar translating
 - $X=4$ DOF
- Four different controls: up, down, left, right forces
- 500 ~ 2,500 nodes
- 5 ~ 15sec
- Planar TR+RO
- $X=6$ DOF
- 13,600 nodes
- 4.2min

Results – 200MHz, 128MB



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

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

Conclusions

- **Take advantages from both randomized potential fields and roadmaps**
 - “Drives forward” like potential fields
 - Quickly and uniformly explores like roadmaps
- **Efficient and reliable method**
 - **Practical!**

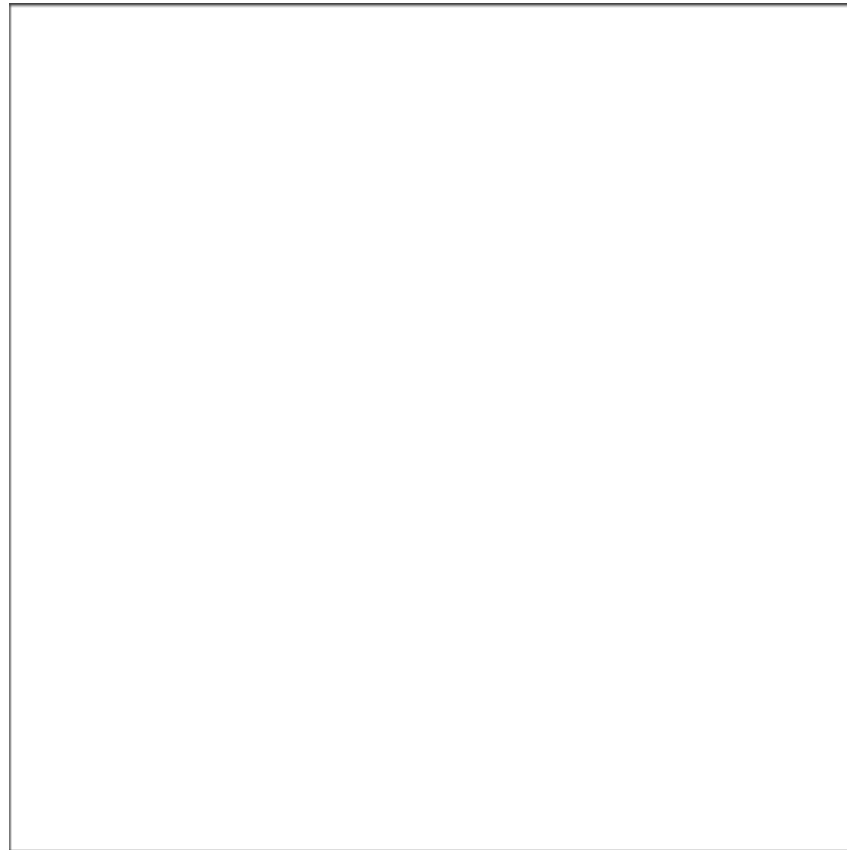
Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain

Anna Yershova Léonard Jaillet Thierry Siméon Steven M. LaValle

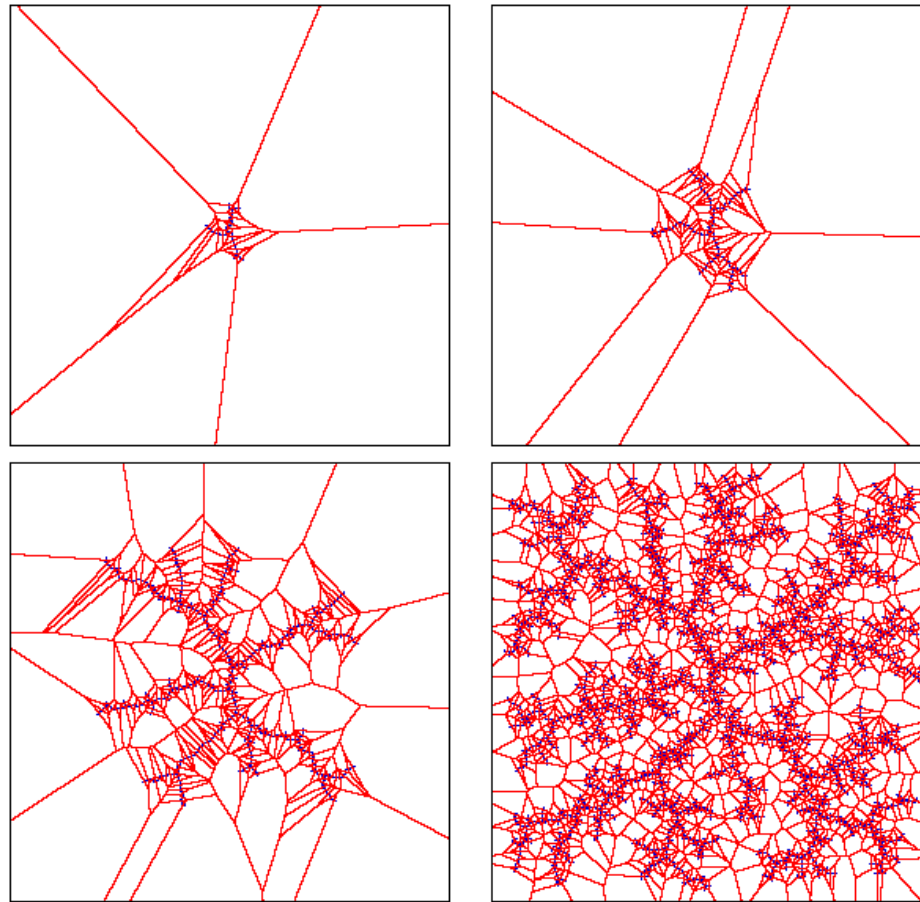
ICRA 05

Citation: more than 80

A Rapidly-exploring Random Tree (RRT)

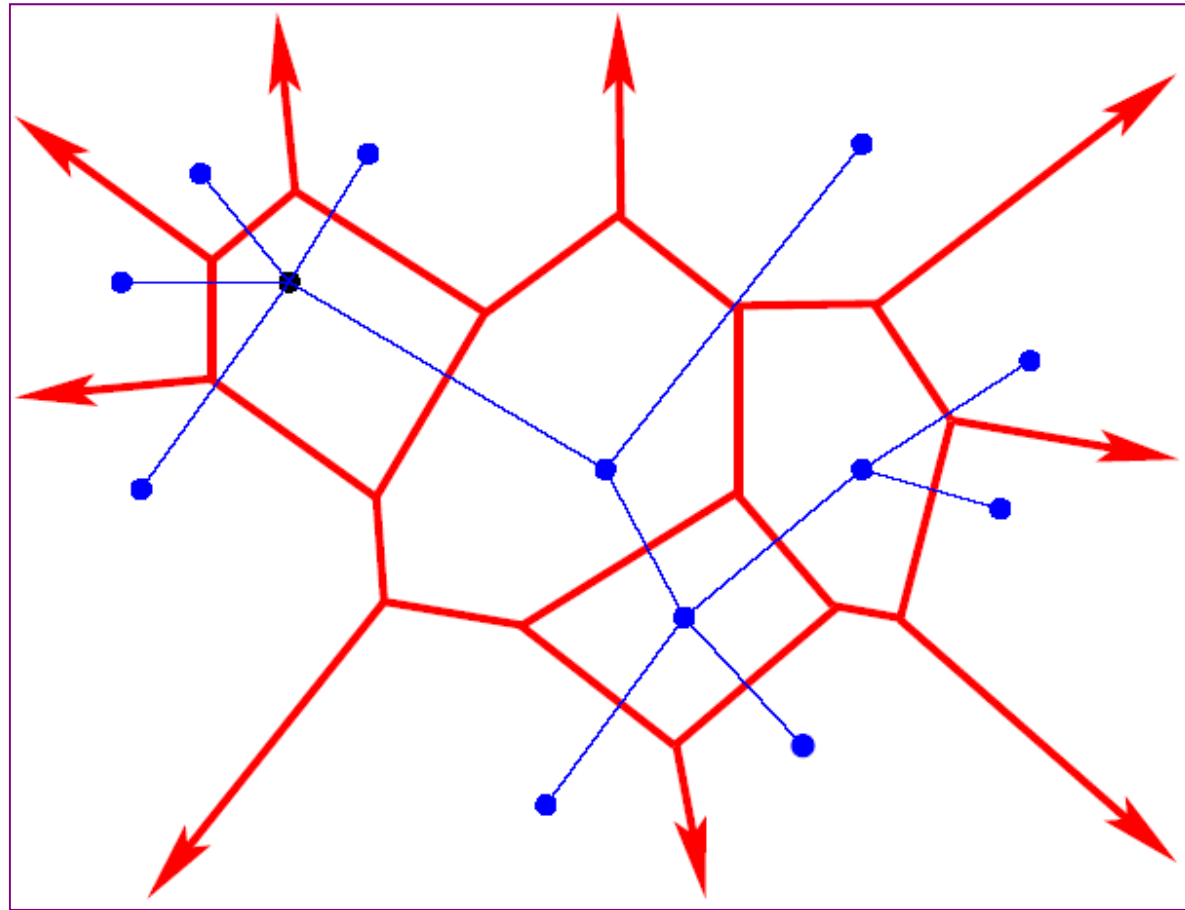


Voronoi Biased Exploration

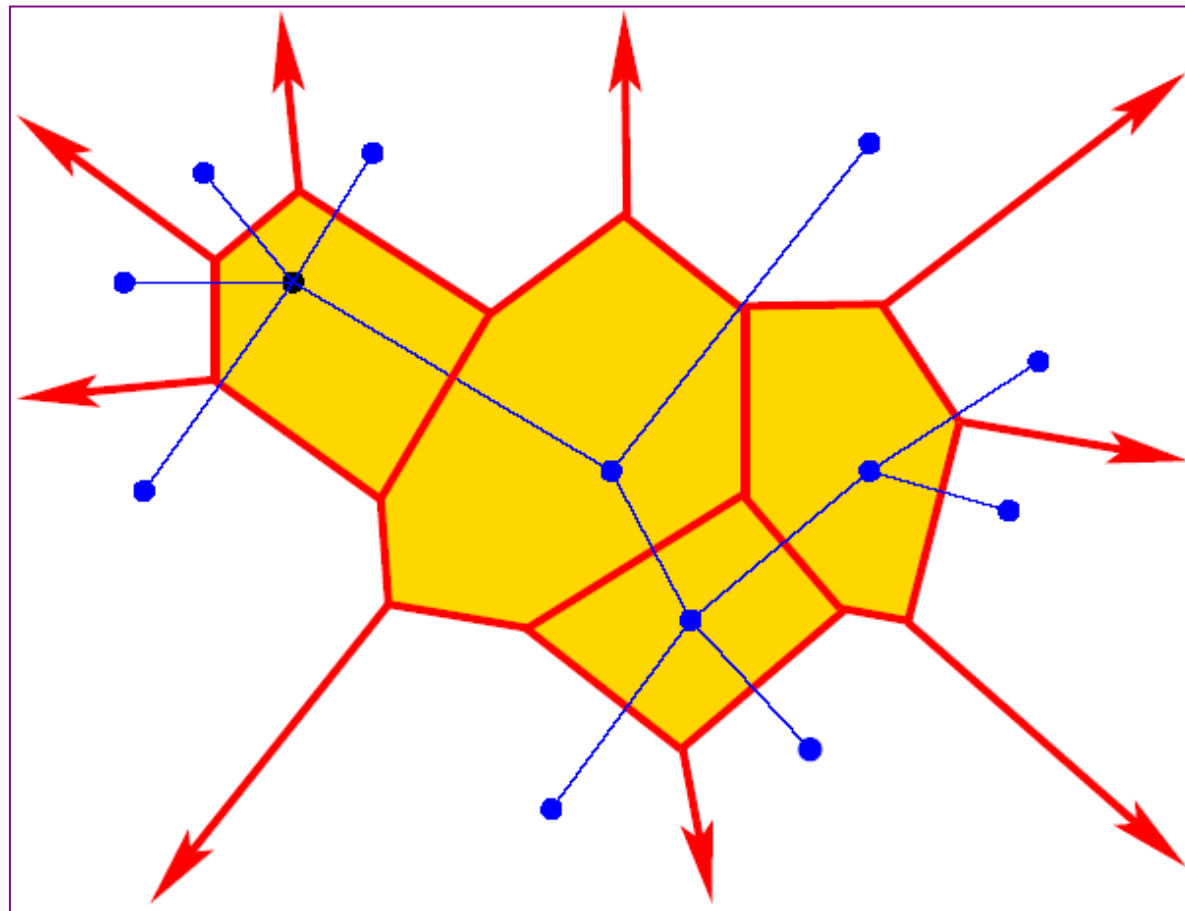


Is this always a good idea?

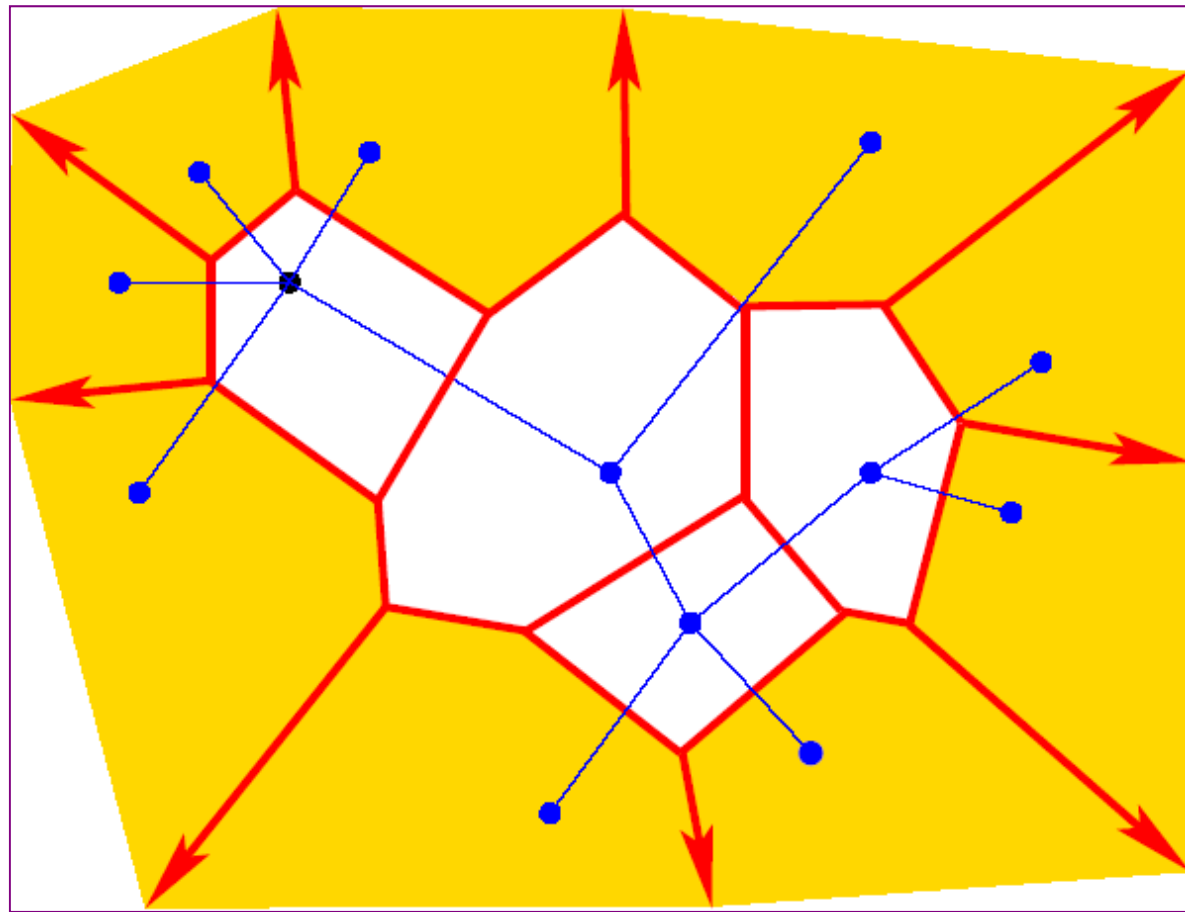
Voronoi Diagram in \mathbb{R}^2



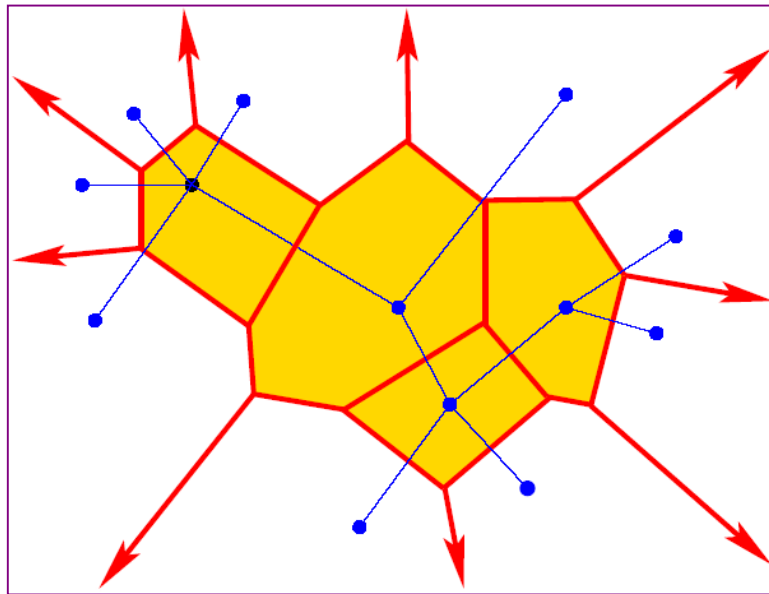
Voronoi Diagram in \mathbb{R}^2



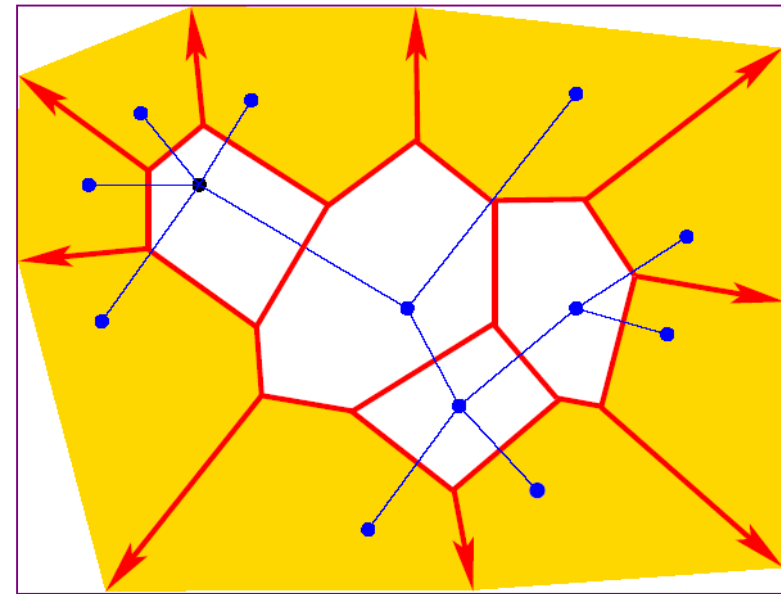
Voronoi Diagram in \mathbb{R}^2



Refinement vs. Expansion



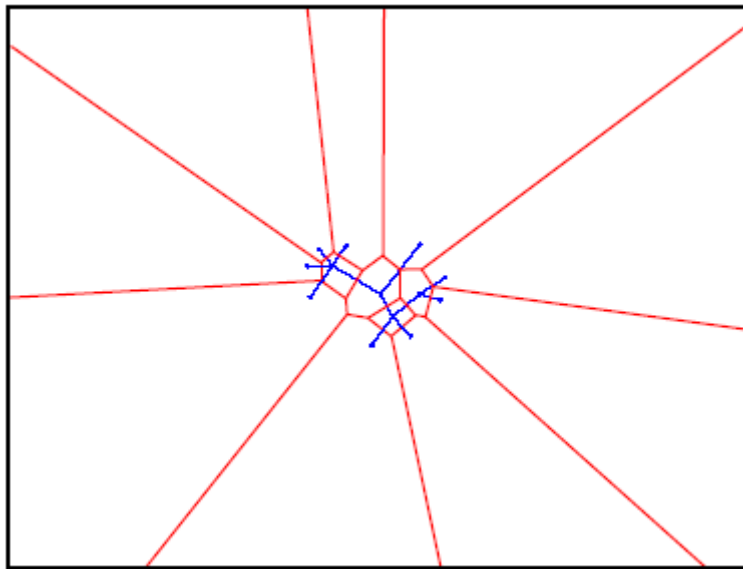
refinement



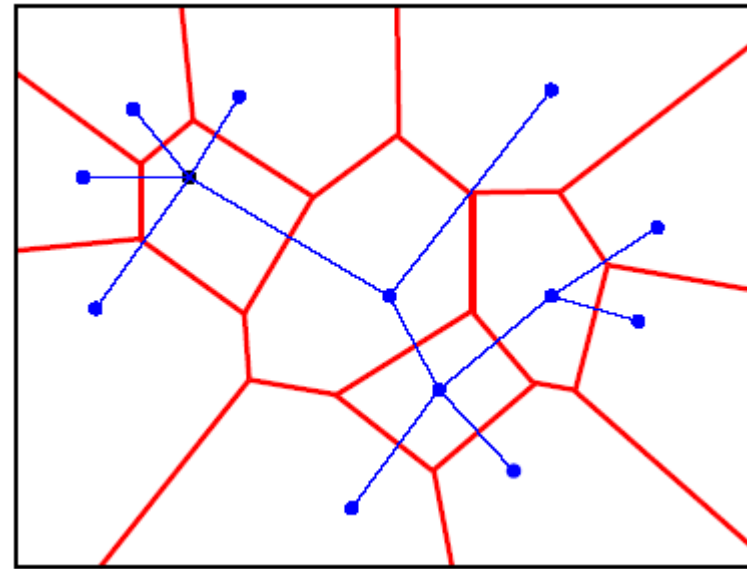
expansion

Where will the random sample fall? How to control the behavior of RRT?

Determining the Boundary



Expansion
dominates



Balanced refinement and
expansion

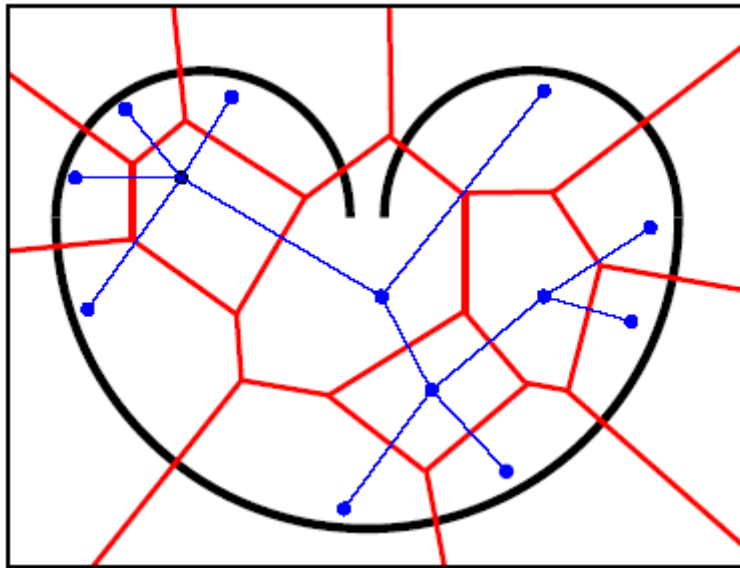
Controlling the Voronoi Bias

- Refinement is good when multiresolution search is needed
- Expansion is good when the tree can grow and not blocked by obstacles

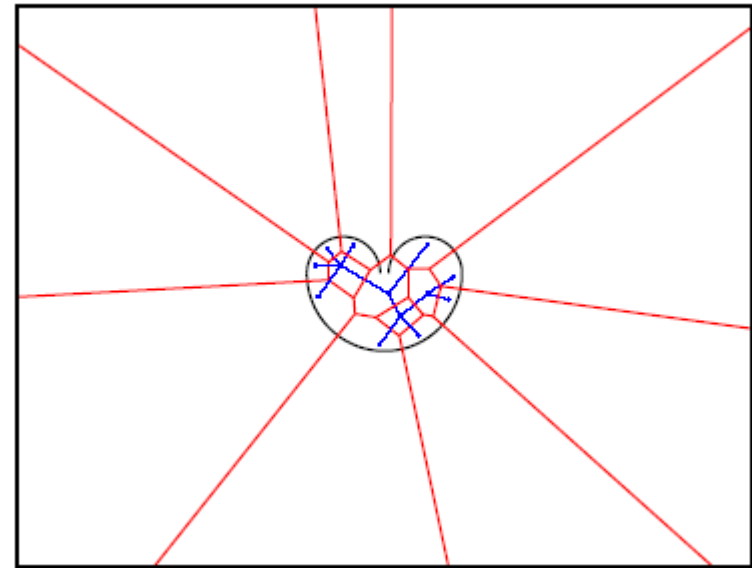
Main motivation:

- Voronoi bias does not take into account obstacles
- How to incorporate the obstacles into Voronoi bias?

Bug Trap



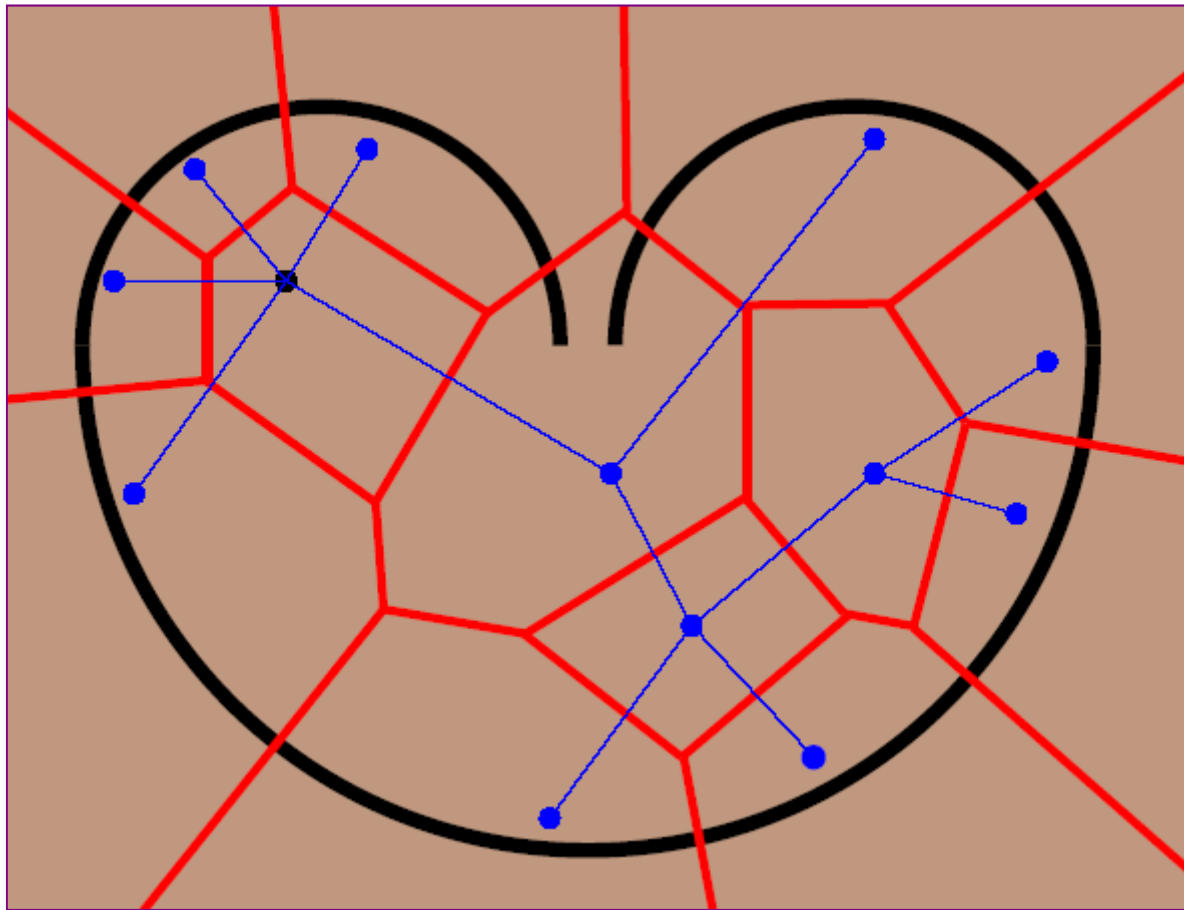
Small Bounding Box



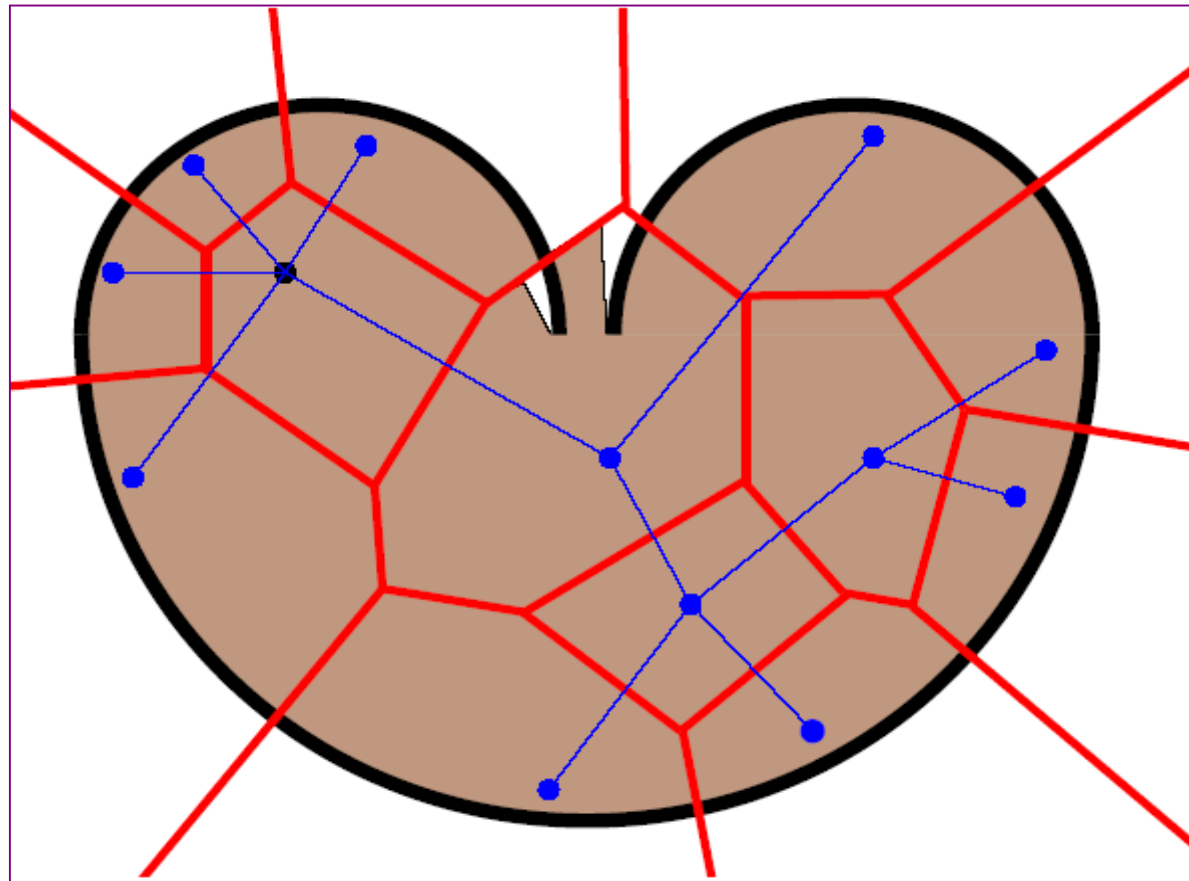
Large Bounding Box

Which one will perform better?

Voronoi Bias for the Original RRT

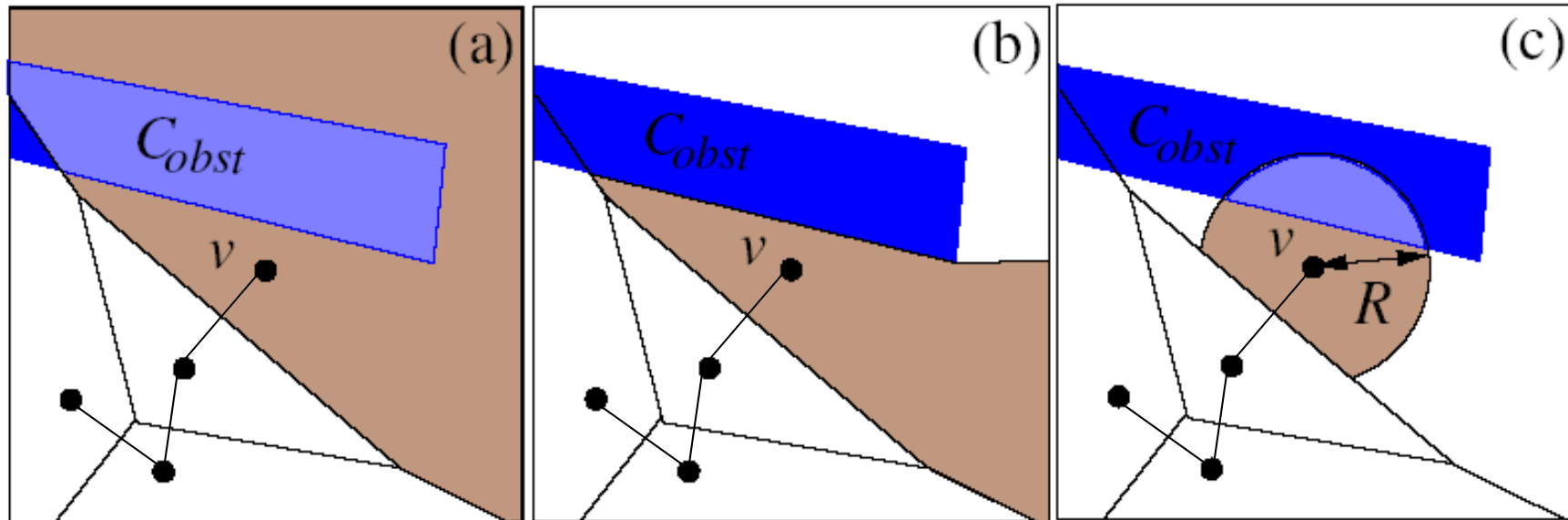


Visibility-Based Clipping of the Voronoi Regions



Nice idea, but how can this be done in practice?
Even better: Voronoi diagram for obstacle-based metric

A Boundary Node

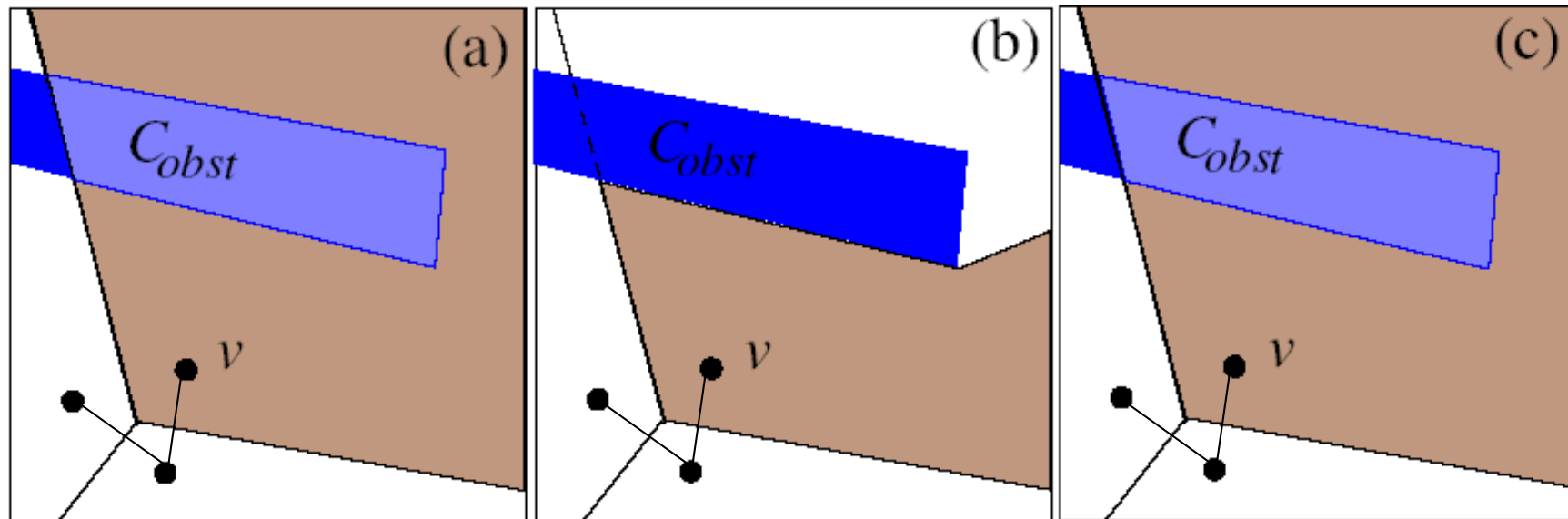


(a) Regular RRT, unbounded Voronoi region

(b) Visibility region

(c) Dynamic domain

A Non-Boundary Node

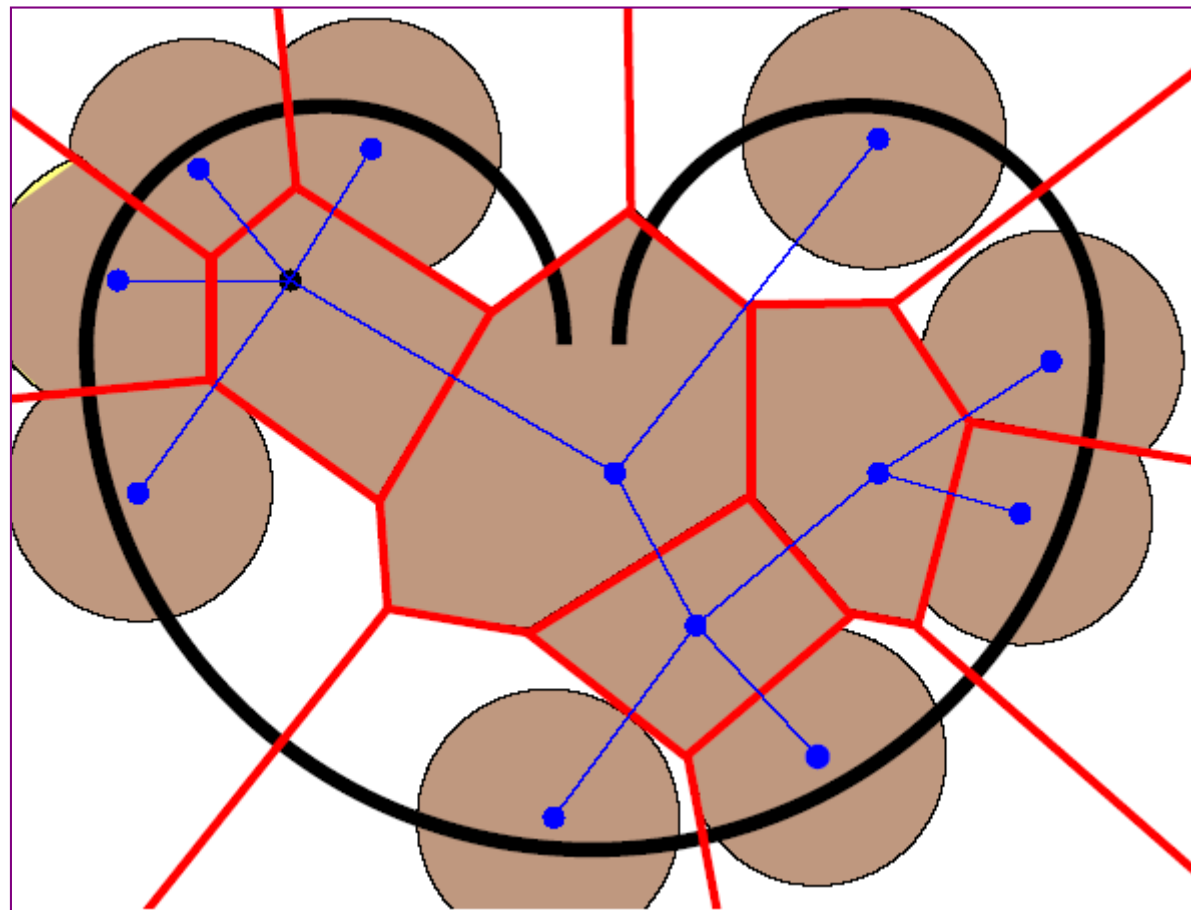


(a) Regular RRT, unbounded Voronoi region

(b) Visibility region

(c) Dynamic domain

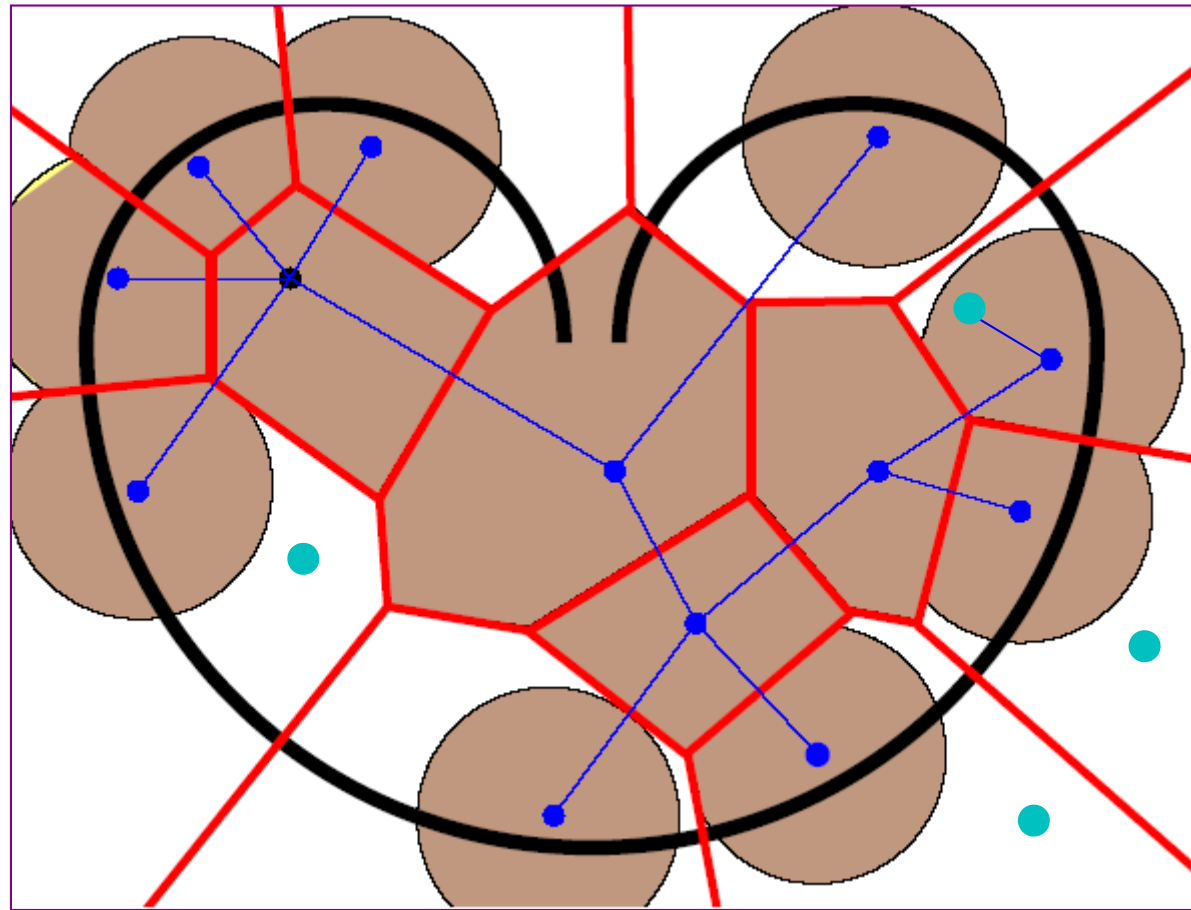
Dynamic-Domain RRT Bias



Dynamic-Domain RRT Construction

```
BUILD_DYNAMIC_DOMAIN_RRT( $q_{init}$ )
1   $\mathcal{T}.init(q_{init});$ 
2  for  $k = 1$  to  $K$  do
3    repeat
4       $q_{rand} \leftarrow \text{RANDOM\_CONFIG}();$ 
5       $q_{near} \leftarrow \text{NEAREST\_NEIGHBOR}(q_{rand}, \mathcal{T});$ 
6    until  $dist(q_{near}, q_{rand}) < q_{near}.radius$ 
7    if  $\text{CONNECT}(\mathcal{T}, q_{rand}, q_{near}, q_{new})$ 
8       $q_{new}.radius = \infty;$ 
9       $\mathcal{T}.add\_vertex(q_{new});$ 
10      $\mathcal{T}.add\_edge(q_{near}, q_{new});$ 
11   else
12      $q_{near}.radius = R;$ 
13  Return  $\mathcal{T};$ 
```

Dynamic-Domain RRT Bias



Tradeoff between nearest neighbor calls and collision detection calls

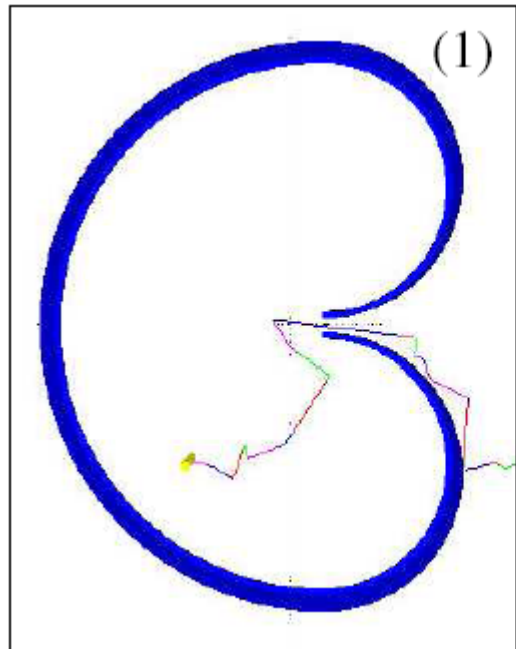
Experiments

- 333 Mhz machine

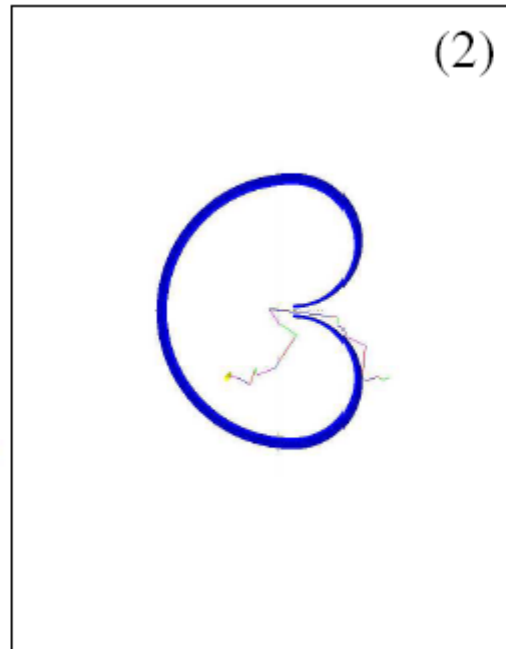
Two kinds of experiments:

- Controlled experiments for toy problems
- Challenging benchmarks from industry and biology

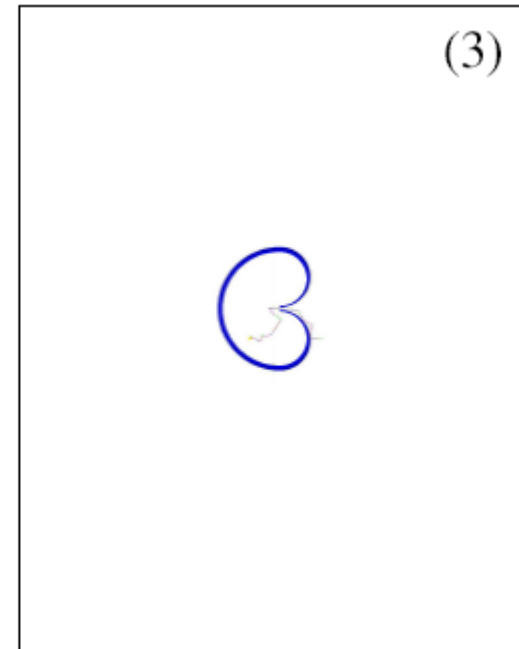
Shrinking Bug Trap



Large



Medium

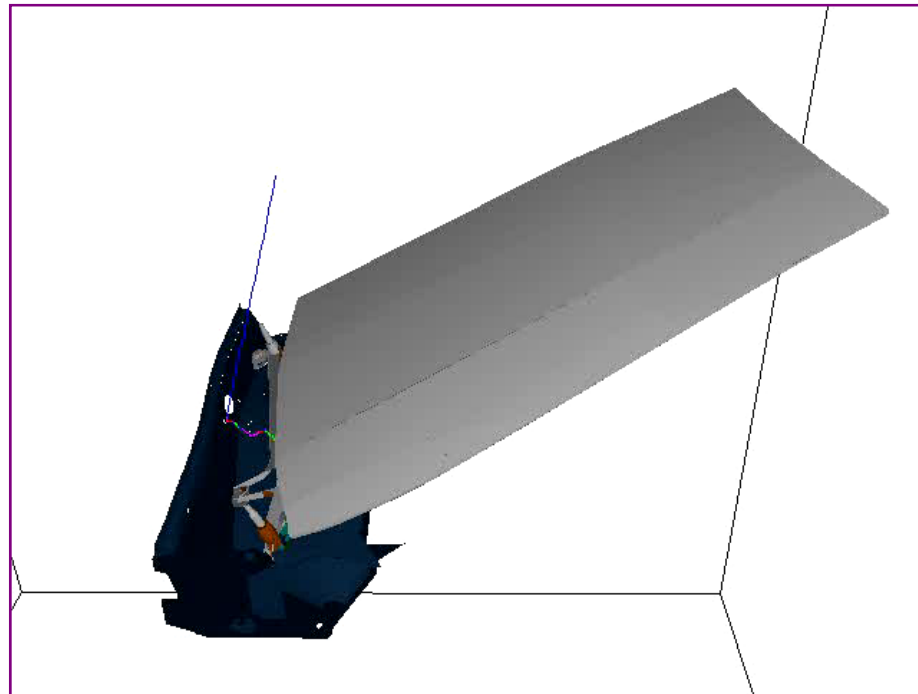


Small

Shrinking Bug Trap

Trap Size	Statistic	Dynamic-Domain bi-RRT	bi-RRT
Large	time (1)	0.4 sec	0.1 sec
	no. nodes (1)	253	37
	CD calls (1)	618	54
Medium	time (2)	2.5 sec	379 sec
	no. nodes (2)	1607	6924
	CD calls (2)	3751	781530
Small	time (3)	1.6 sec	> 80000 sec
	no. nodes (3)	1301	–
	CD calls (3)	3022	–

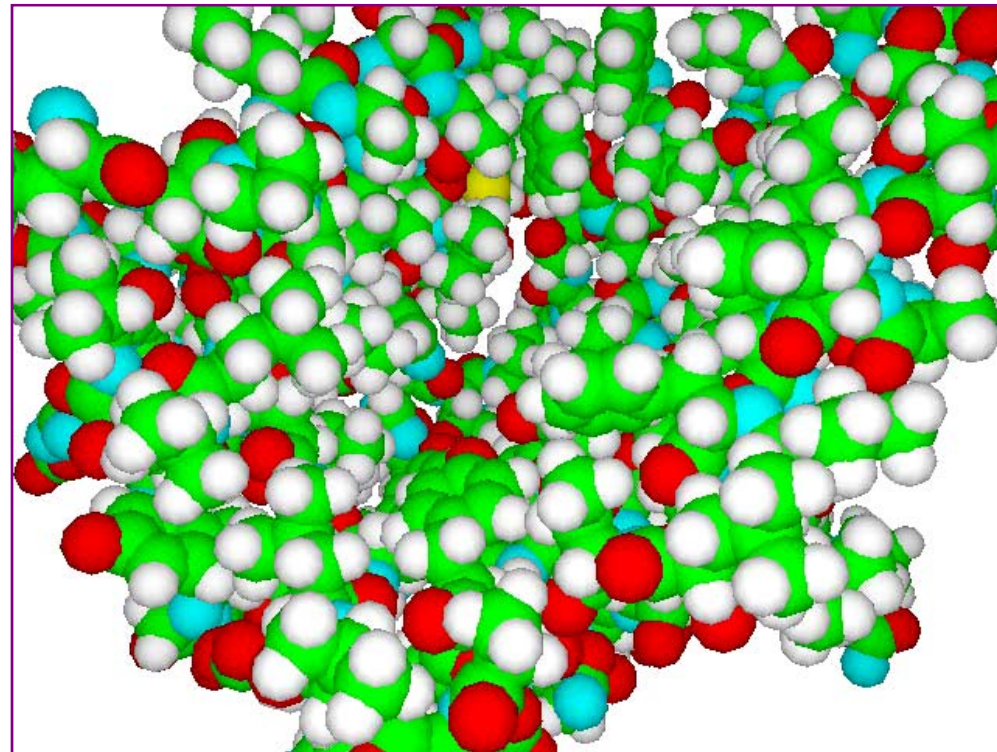
Wiper Motor (courtesy of KINEO)



- ❑ 6 dof problem
- ❑ CD calls are expensive

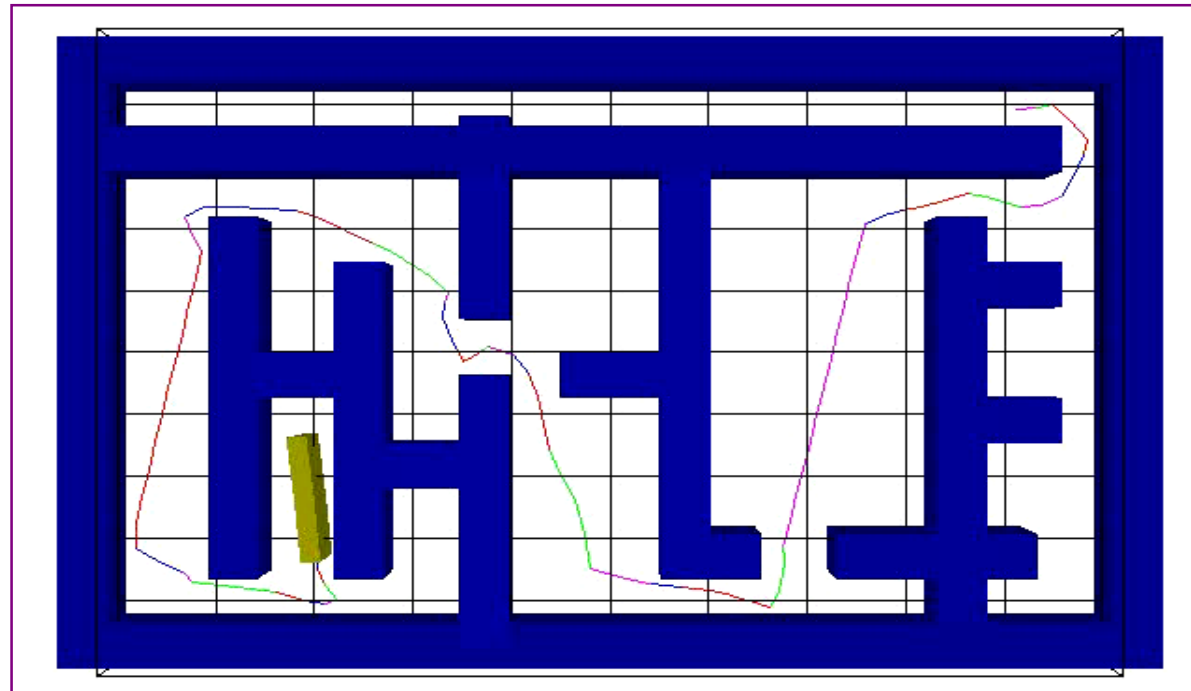
	Dynamic-Domain bi-RRT	bi-RRT
time	217 sec	> 80000 sec
no. nodes	219	–
CD calls	30443	–

Molecule



	Dynamic-Domain bi-RRT	bi-RRT
time	70 sec	2926 sec
no. nodes	1358	428
CD calls	47710	1257055

Labyrinth



- ❑ 3 dof problem
- ❑ CD calls are not expensive

	Dynamic-Domain bi-RRT	bi-RRT
time	161 sec	237 sec
no. nodes	25483	20392
CD calls	604503	464137

Conclusions

- **Controlling Voronoi bias is important in RRTs**
- **Provides dramatic performance improvements on some problems**
- **Does not incur much penalty for unsuitable problems**

Work in Progress:

- **There is a radius parameter; adaptive tuning is possible**

Class Objectives were:

- Understand the RRT technique and its recent advancements