Autonomous and Mobile Robotics

Prof. Giuseppe Oriolo

Motion Planning
Probabilistic Methods

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

sampling-based methods

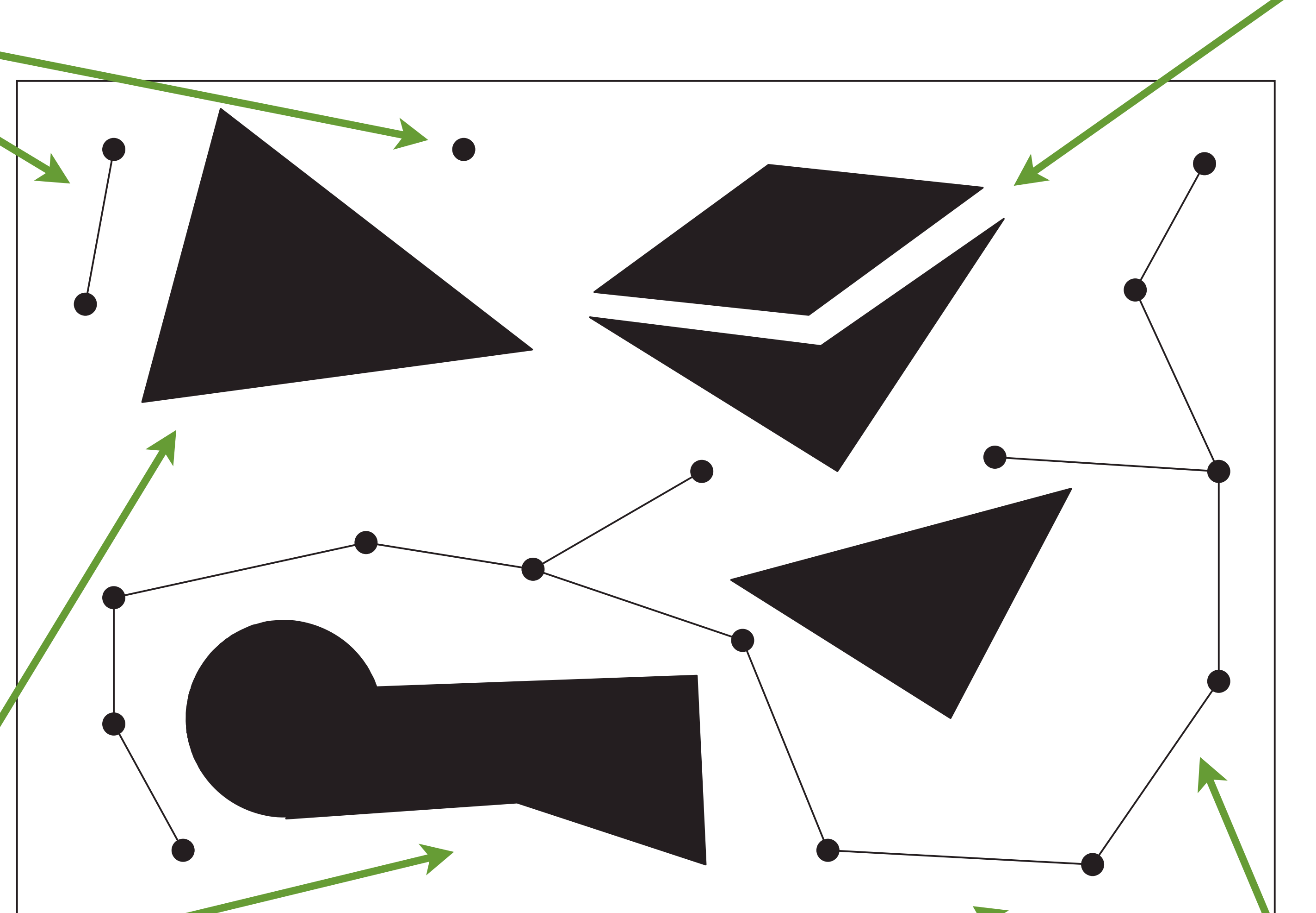
- build a roadmap of the configuration space $\mathcal{C}$ by repeating this basic iteration:
  - extract a sample $\boldsymbol{q}$ of $\mathcal{C}$
  - use forward kinematics to compute the volume $\mathcal{B}(\boldsymbol{q})$ occupied by the robot $\mathcal{B}$ at $\boldsymbol{q}$
  - check collision between $\mathcal{B}(\boldsymbol{q})$ and obstacles $\mathcal{O}_1,\ldots,\mathcal{O}_p$
  - if $\boldsymbol{q} \in \mathcal{C}_{\text{free}}$, add $\boldsymbol{q}$ to the roadmap; else, discard it
- preliminary computation of $\mathcal{CO}$ is completely avoided: an approximate representation of $\mathcal{C}_{\text{free}}$ is directly built as a collection of connected configurations (roadmap)
- different criteria for sampling lead to different methods: in general, randomized outperforms deterministic

PRM (Probabilistic Roadmap)

- basic iteration to build the PRM:
  - extract a sample $\boldsymbol{q}$ of $\mathcal{C}$ with uniform probability distribution
  - compute $\mathcal{B}(\boldsymbol{q})$ and check for collision
  - if $\boldsymbol{q} \in \mathcal{C}_{\rm free}$, add $\boldsymbol{q}$ to the PRM; else, discard it
  - search the PRM for "sufficiently near" configurations $\boldsymbol{q}_{\rm near}$
  - if possible, connect $\boldsymbol{q}$ to $\boldsymbol{q}_{\rm near}$ with a free local path
- the generation of a free path between $\boldsymbol{q}$ and $\boldsymbol{q}_{\rm near}$ is delegated to a procedure called local planner: e.g., throw a linear path and check it for collision
- the chosen metric in $\mathcal{C}$ plays a role in identifying $\boldsymbol{q}_{\rm near}$

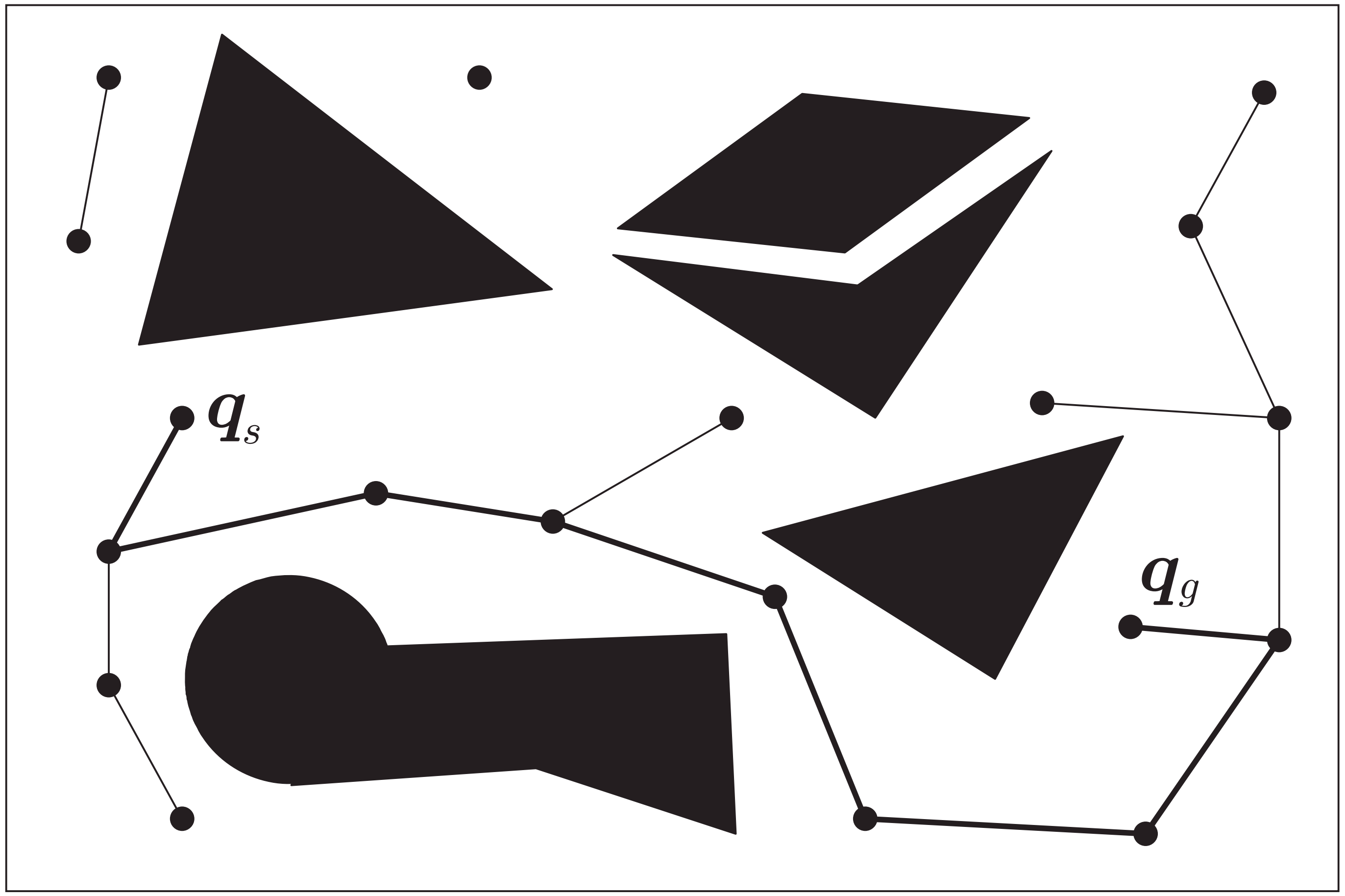
disconnected components

narrow passages are scarcely sampled



$\mathcal{C}$-obstacles are never computed

local paths



- construction of the PRM is arrested when
  1. connected components become less than a threshold, or
  2. a maximum number of iterations is reached
- if $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ can be connected to the same component, a solution can be found by graph search; else, enhance the PRM by performing more iterations

- the PRM method is probabilistically complete, i.e., the probability of finding a solution whenever one exists tends to 1 as the execution time tends to ∞; and is multiple-query (new queries enhance the PRM)

- the main advantage is speed; the time PRM needs to find a solution in high-dimensional spaces can be orders of magnitude smaller than previous planners

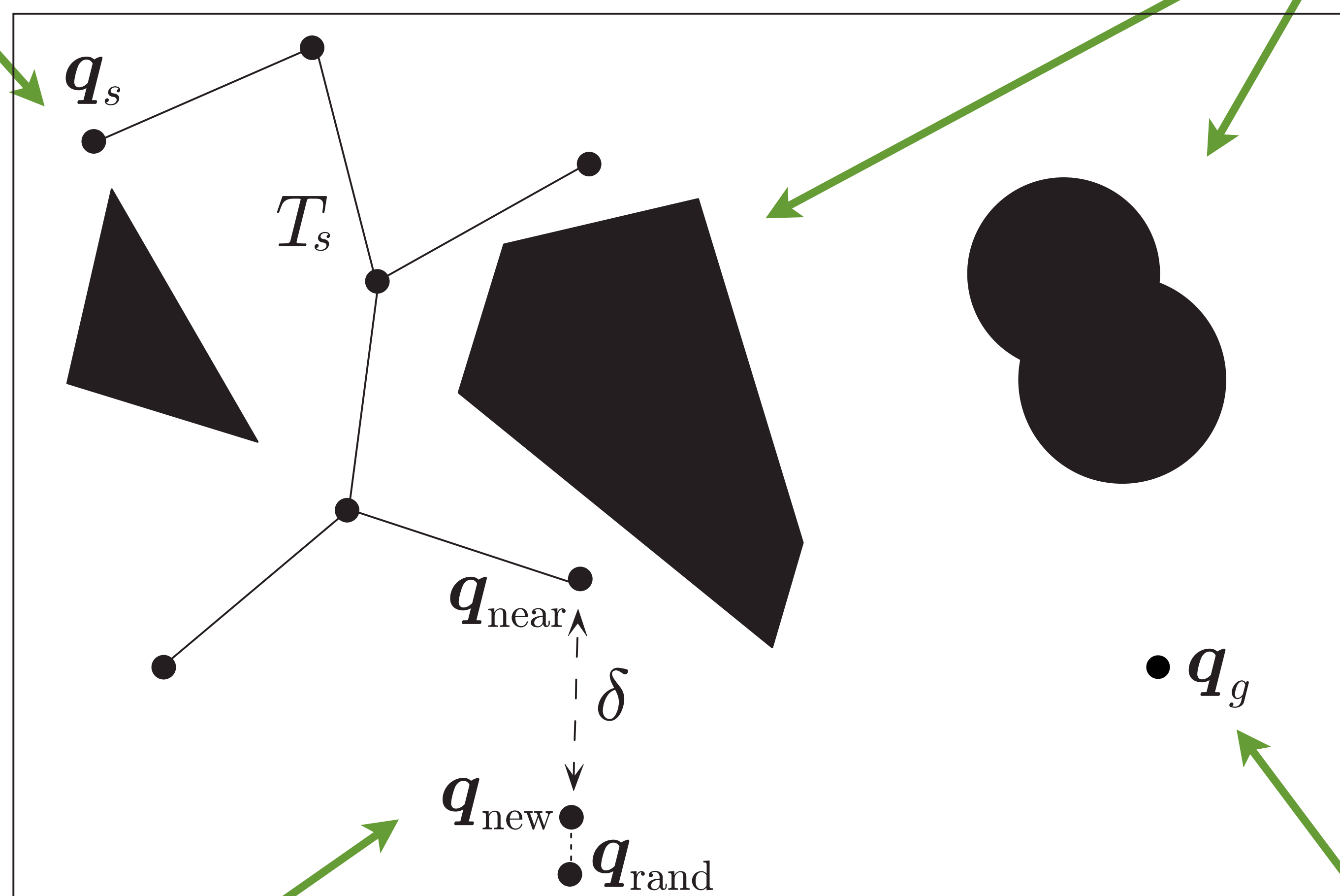
- narrow passages are critical; heuristics may be used to design biased (non-uniform) probability distributions aimed at increasing sampling in such areas

RRT (Rapidly-exploring Random Tree)

- basic iteration to build the tree T_s rooted at $\boldsymbol{q}_s$:
  - generate $\boldsymbol{q}_{\mathrm{rand}}$ in $\mathcal{C}$ with uniform probability distribution
  - search the tree for the nearest configuration $\boldsymbol{q}_{\mathrm{near}}$
  - choose $\boldsymbol{q}_{\mathrm{new}}$ at a distance δ from $\boldsymbol{q}_{\mathrm{near}}$ in the direction of $\boldsymbol{q}_{\mathrm{rand}}$
  - check for collision $\boldsymbol{q}_{\mathrm{new}}$ and the segment from $\boldsymbol{q}_{\mathrm{near}}$ to $\boldsymbol{q}_{\mathrm{new}}$
  - if check is negative, add $\boldsymbol{q}_{\mathrm{new}}$ to T_s (expansion)
- the chosen metric in $\mathcal{C}$ plays a role in identifying $\boldsymbol{q}_{\mathrm{near}}$
- T_s rapidly covers $\mathcal{C}_{\mathrm{free}}$ because the expansion is biased towards unexplored areas (actually, towards larger Voronoi regions)

tree is
rooted at $\boldsymbol{q}_s$

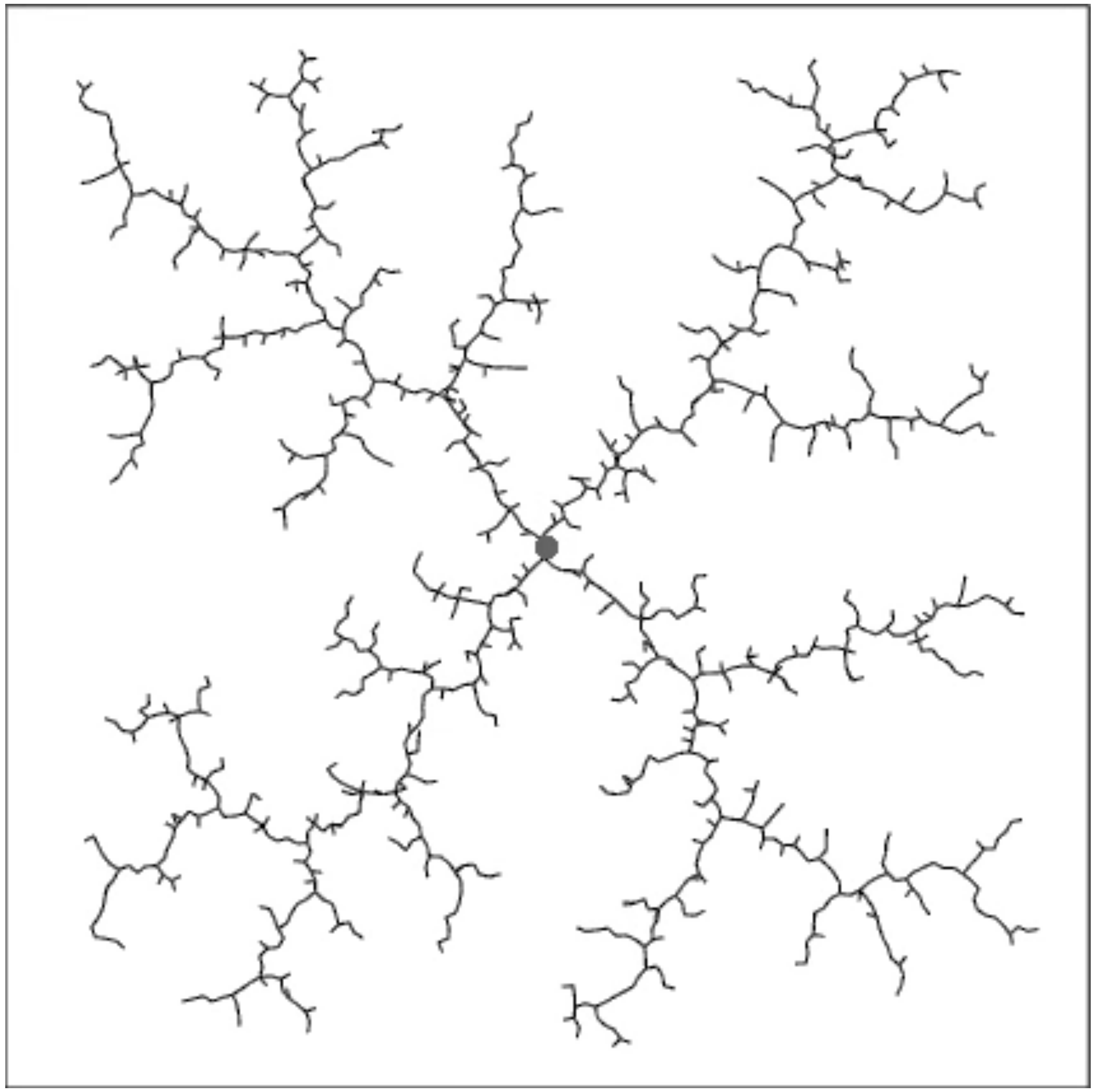
$\mathcal{C}$-obstacles are
never computed



tree
expansion

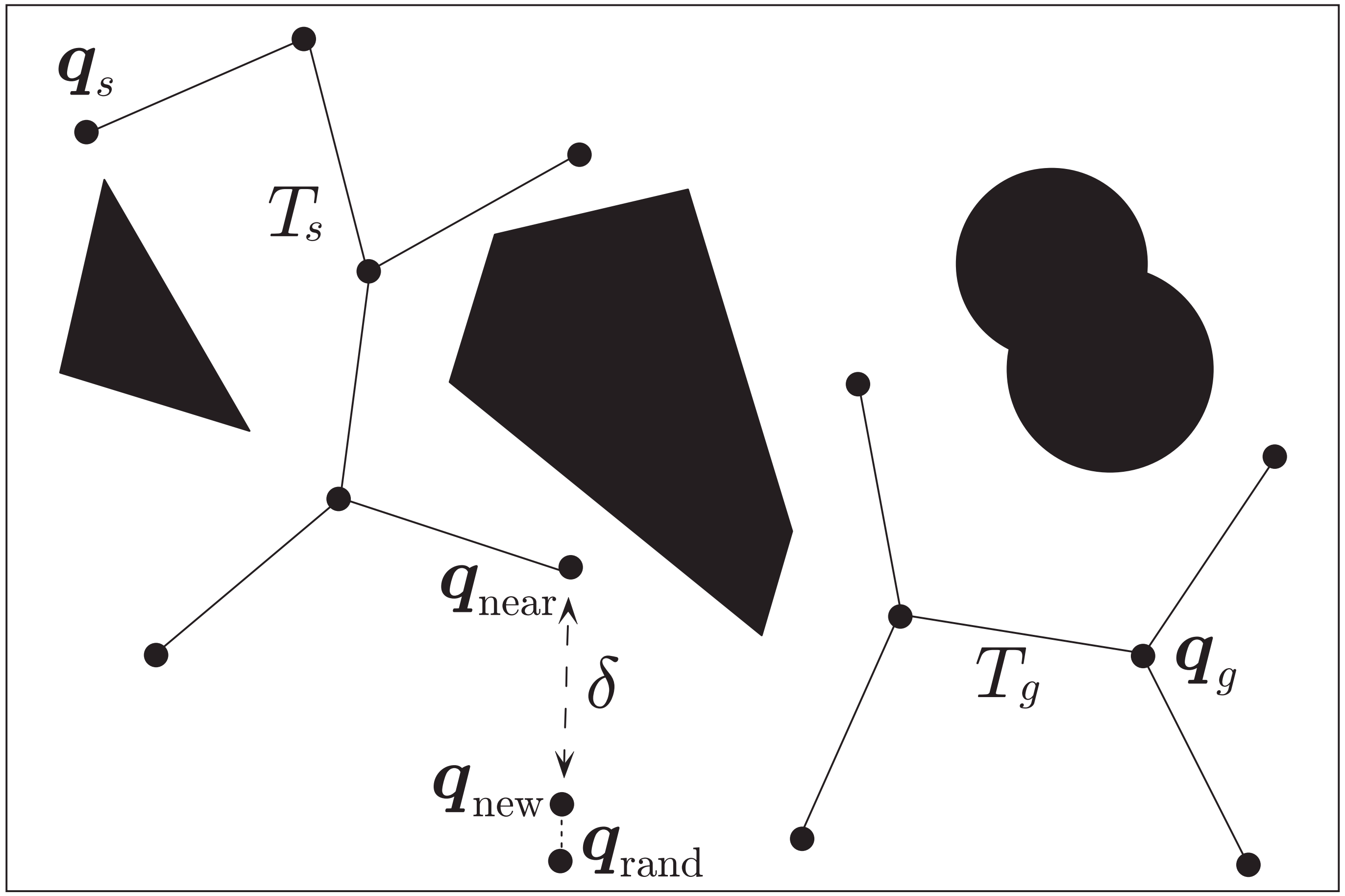
no bias
towards $\boldsymbol{q}_g$

RRT in empty 2D space



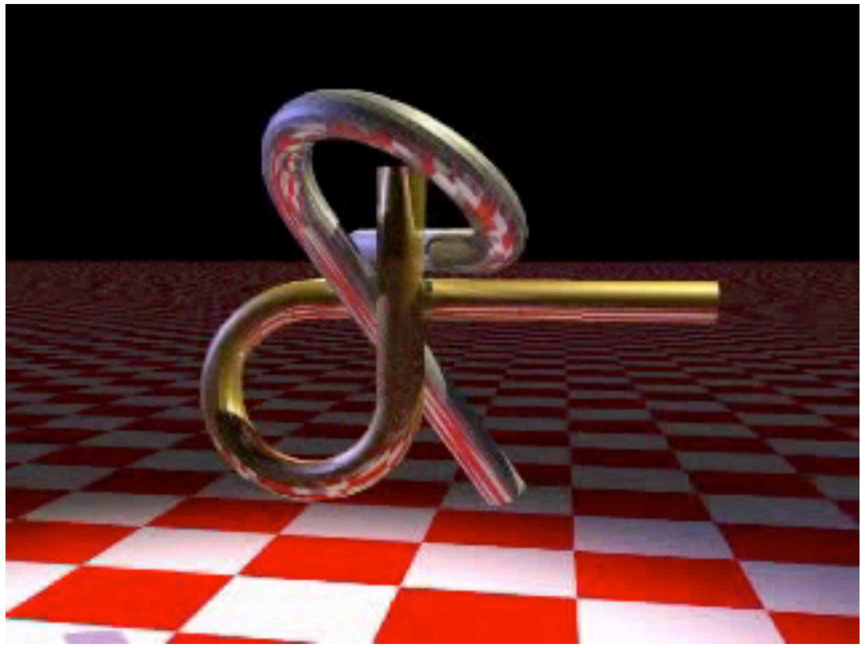
quickly explores all areas, much more efficiently than other simple strategies, e.g., random walks

- to introduce a bias towards $\boldsymbol{q}_g$, one may grow two trees T_s and T_g, respectively rooted at $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ (bidirectional RRT)
- alternate expansion and connection phases: use the last generated $\boldsymbol{q}_{\text{new}}$ of T_s as a $\boldsymbol{q}_{\text{rand}}$ for T_g, and then repeat switching the roles of T_s and T_g



- bidirectional RRT is probabilistically complete and single-query (trees are rooted at $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, and in any case new queries may require significant work)
- as an alternative, one may adopt ε-greedy exploration, to balance exploration ($\boldsymbol{q}_{\rm rand}=\mathrm{random}(\boldsymbol{q})$) and exploitation ($\boldsymbol{q}_{\rm rand}=\boldsymbol{q}_{\rm g}$)
- many variations to basic RRT are possible: e.g., one may use an adaptive stepsize δ to speed up motion in wide open areas
- RRT can be modified to address many extensions of the canonical planning problem, e.g., moving obstacles, nonholonomic constraints, manipulation planning

a benchmark problem: the Alpha Puzzle



- 6-dof configuration space + narrow passages
- solved by bidirectional RRT in few mins (average)
- in practice, this problem is not solvable by classical methods such as retraction or cell decomposition

RRT: extension to nonholonomic robots

- motion planning for a unicycle in $\mathcal{C} = \mathrm{R}^2 \times SO(2)$

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} v + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \omega$$

- linear paths in $\mathcal{C}$ such as those used to connect $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{rand}}$ are not admissible in general

- one possibility is to use motion primitives, i.e., a finite set of admissible local paths, produced by a specific choice of the inputs (kinodynamic RRT)

- for example, one may use (Dubins car)

$$v = \bar{v} \qquad \omega = \{-\bar{\omega}, 0, \bar{\omega}\} \qquad t \in [0, \Delta]$$

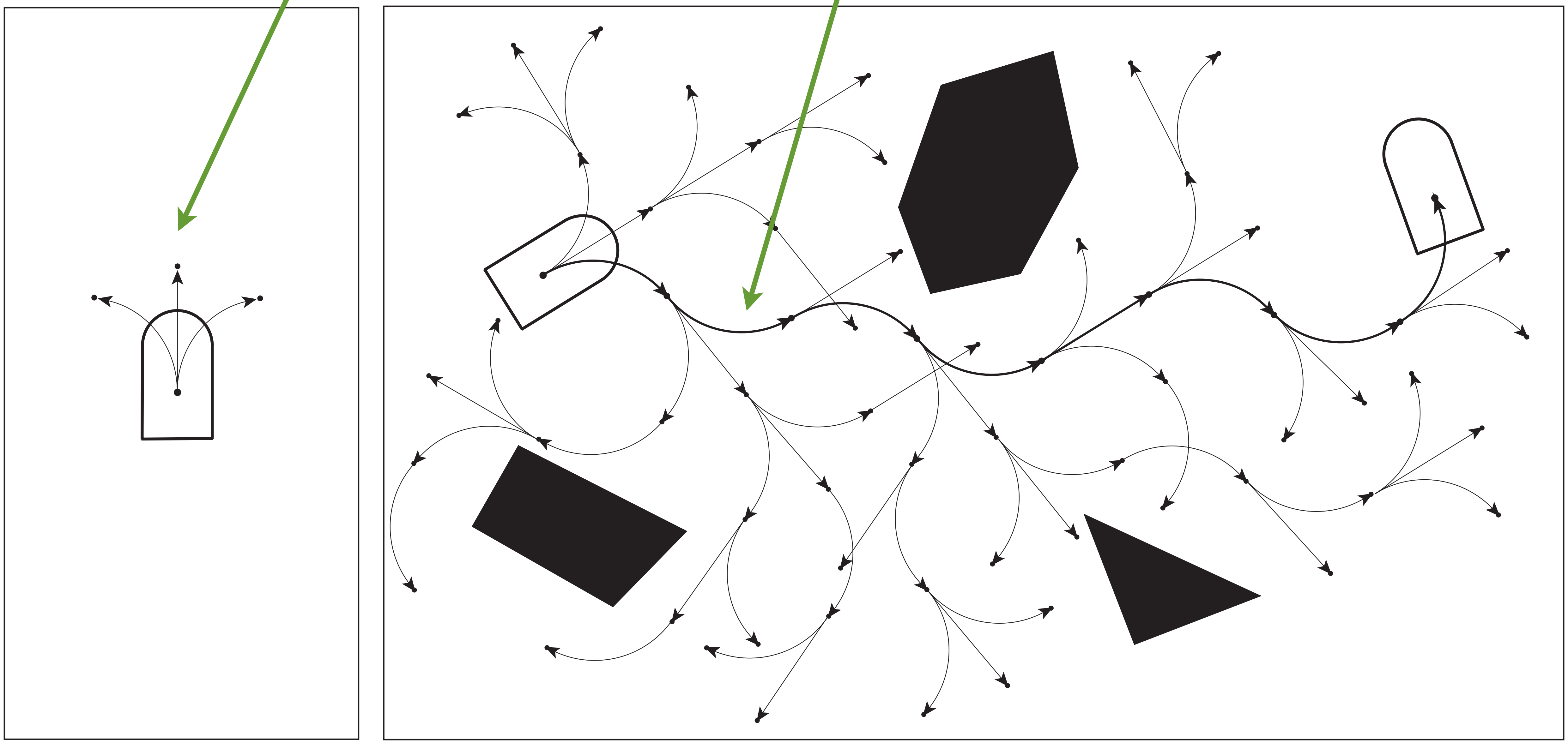
resulting in 3 possible paths in forward motion

- the algorithm is the same with the only difference that $\boldsymbol{q}_{\rm new}$ is generated from $\boldsymbol{q}_{\rm near}$ selecting one of the possible paths (either randomly or as the one that leads the unicycle closer to $\boldsymbol{q}_{\rm rand}$)

- if $\boldsymbol{q}_g$ can be reached from $\boldsymbol{q}_s$ with a collision-free concatenation of primitives, the probability that a solution is found tends to 1 as the time tends to ∞

solution path made by concatenation

primitives



optimal motion planning

- PRM or RRT do not allow to optimize a cost function (length of the path, distance from the obstacles,...)
- running the algorithm for a longer time (or multiple times in succession) may improve the solution, but optimality is not guaranteed; indeed, one can prove that, whatever the cost function, the probability of finding an optimal solution is 0
- one may seek optimal paths using $A^\star$ (or variants) on a previously computed gridmap representation, but running time grows exponentially with its dimension and optimality is only ensured up to grid resolution

the RRT* algorithm

- idea: consider the cost needed to reach a vertex
- new basic iteration to build the tree T_s rooted at $\boldsymbol{q}_s$:
  - generate $\boldsymbol{q}_{\rm rand}$ in $\mathcal{C}$ with uniform probability distribution
  - search the tree for the nearest configuration $\boldsymbol{q}_{\rm near}$
  - choose $\boldsymbol{q}_{\rm new}$ at a distance δ from $\boldsymbol{q}_{\rm near}$ in the direction of $\boldsymbol{q}_{\rm rand}$
  - check for collision $\boldsymbol{q}_{\rm new}$ and the segment from $\boldsymbol{q}_{\rm near}$ to $\boldsymbol{q}_{\rm new}$
  - if check is negative, add $\boldsymbol{q}_{\rm new}$ to T_s (expansion):
    - identify $Q_{\rm near}$, the vertexes of T_s within distance r from $\boldsymbol{q}_{\rm new}$
    - choose parent: parent of $\boldsymbol{q}_{\rm new}$ is the vertex in $Q_{\rm near}$ which allows to reach $\boldsymbol{q}_{\rm new}$ with minimum cost (rather than $\boldsymbol{q}_{\rm near}$)
    - rewire: for each vertex in $Q_{\rm near}$, redefine its parent as $\boldsymbol{q}_{\rm new}$ if this reduces the cost to reach the vertex

RRT*: definition of Q_{near}

- r (size of the ball around $\boldsymbol{q}_{\text{new}}$) depends on the current size n of the tree T_s as well as on the dimension d of the configuration space $\mathcal{C}$

$$r(n) = \gamma \left(\frac{\log n}{n} \right)^{1/(d+1)}$$

volume of $\mathcal{C}_{\text{free}}$

$$\gamma > \gamma * = 2 \left(1 + \frac{1}{d} \right)^{\frac{1}{d}} \left(\frac{\mu(\mathcal{C}_{\text{free}})}{\zeta_d} \right)^{\frac{1}{d}}$$

volume of
unit ball in R^d

- in practice, choose γ 'large enough'

asymptotic optimality

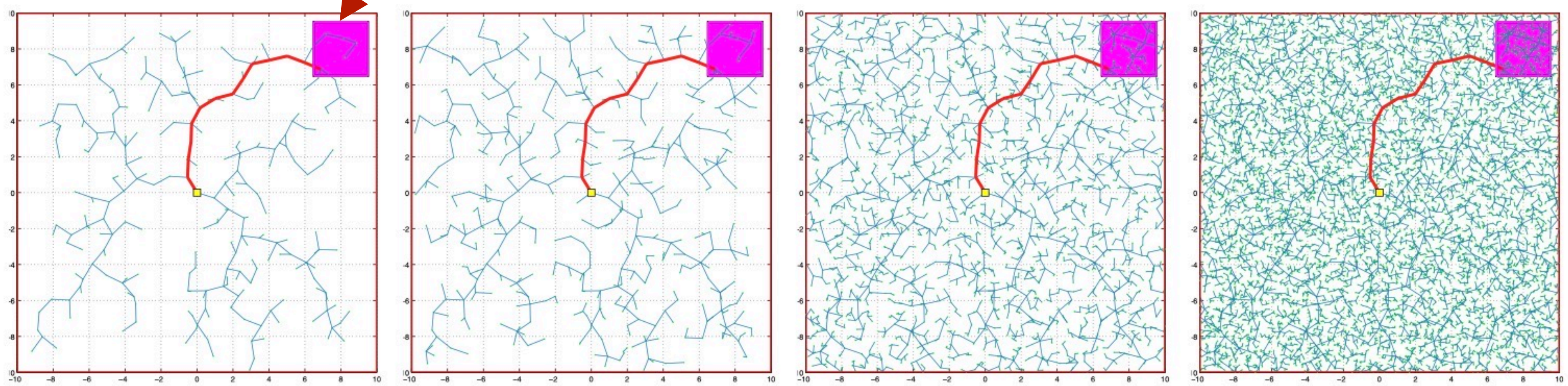
- the choose parent and rewire steps reduce the cost (if possible) of the vertexes in the tree, while still guaranteeing probabilistic completeness

- RRT* is asymptotically optimal: the probability of finding an optimal solution tends to 1 as the number of vertices of the tree (equivalently, the execution time) tends to ∞

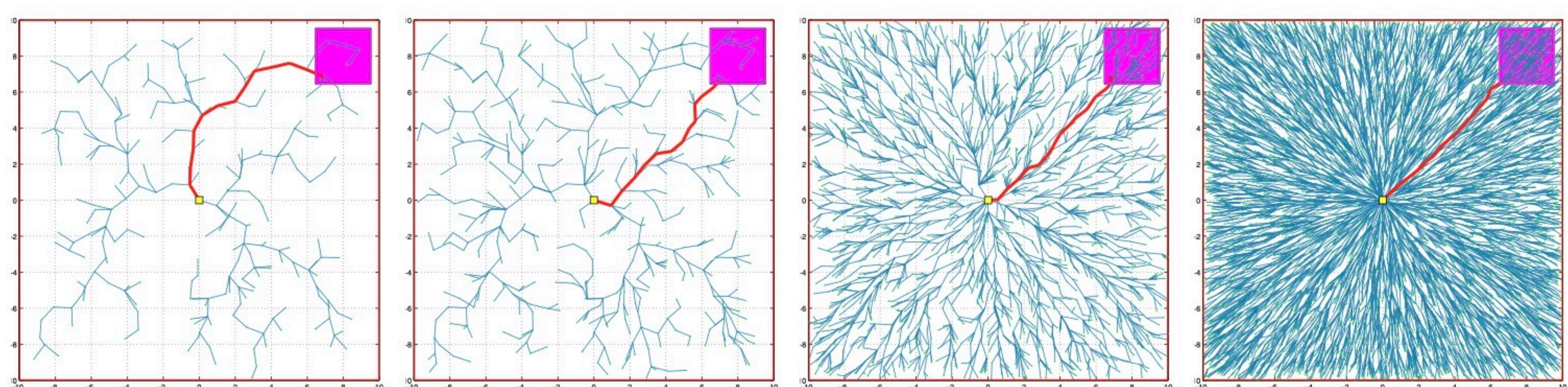
RRT* in empty 2D space

- RRT

goal region

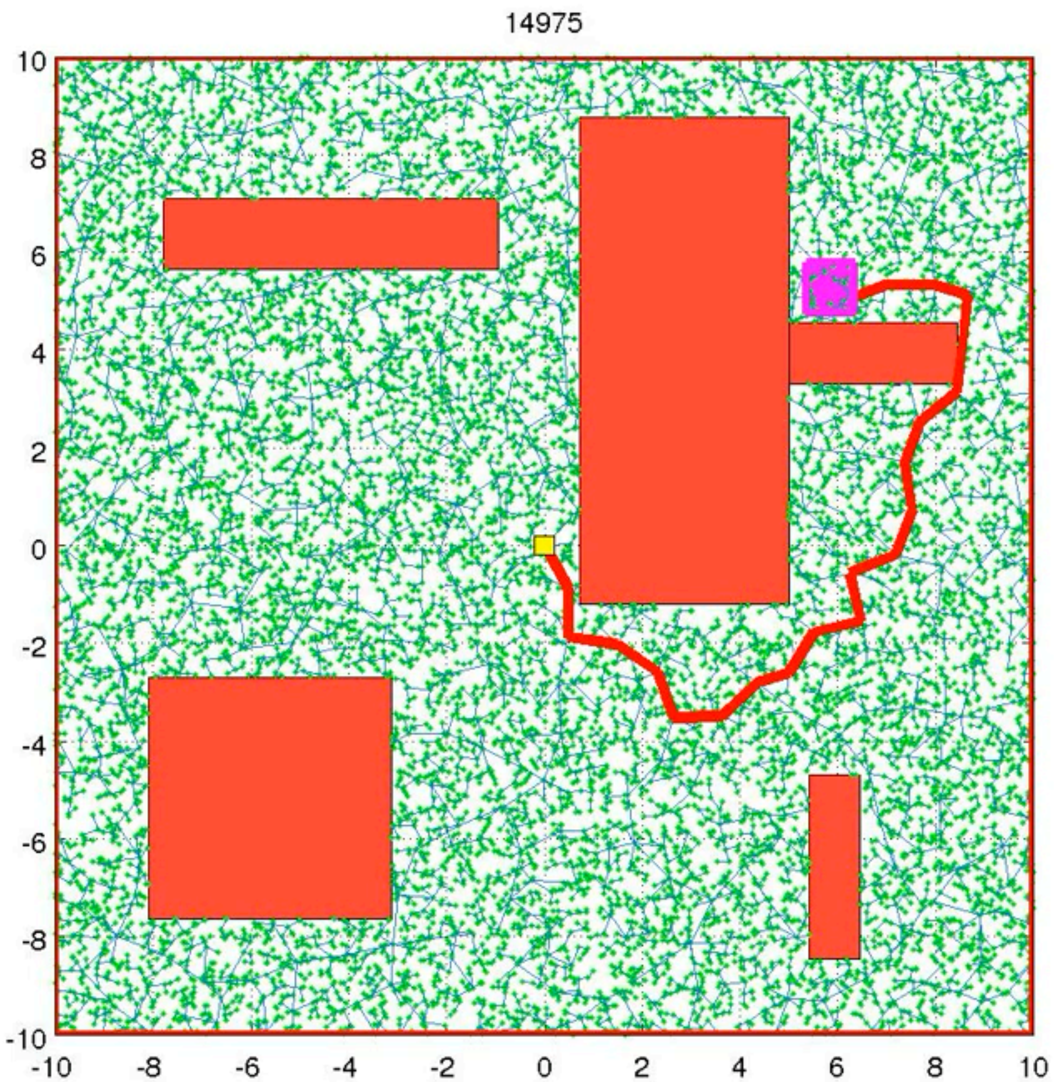


- RRT*

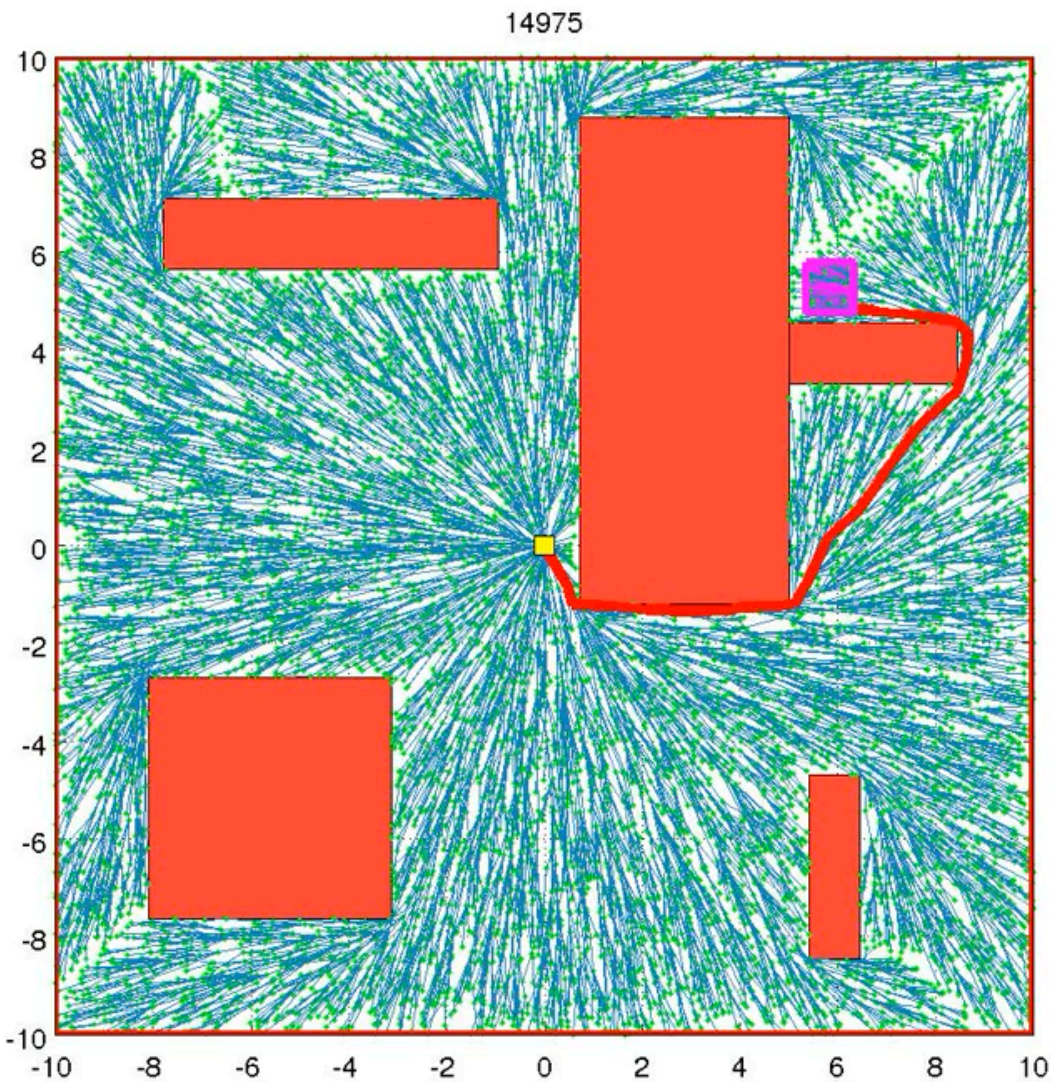


RRT* in 2D space with obstacles

RRT



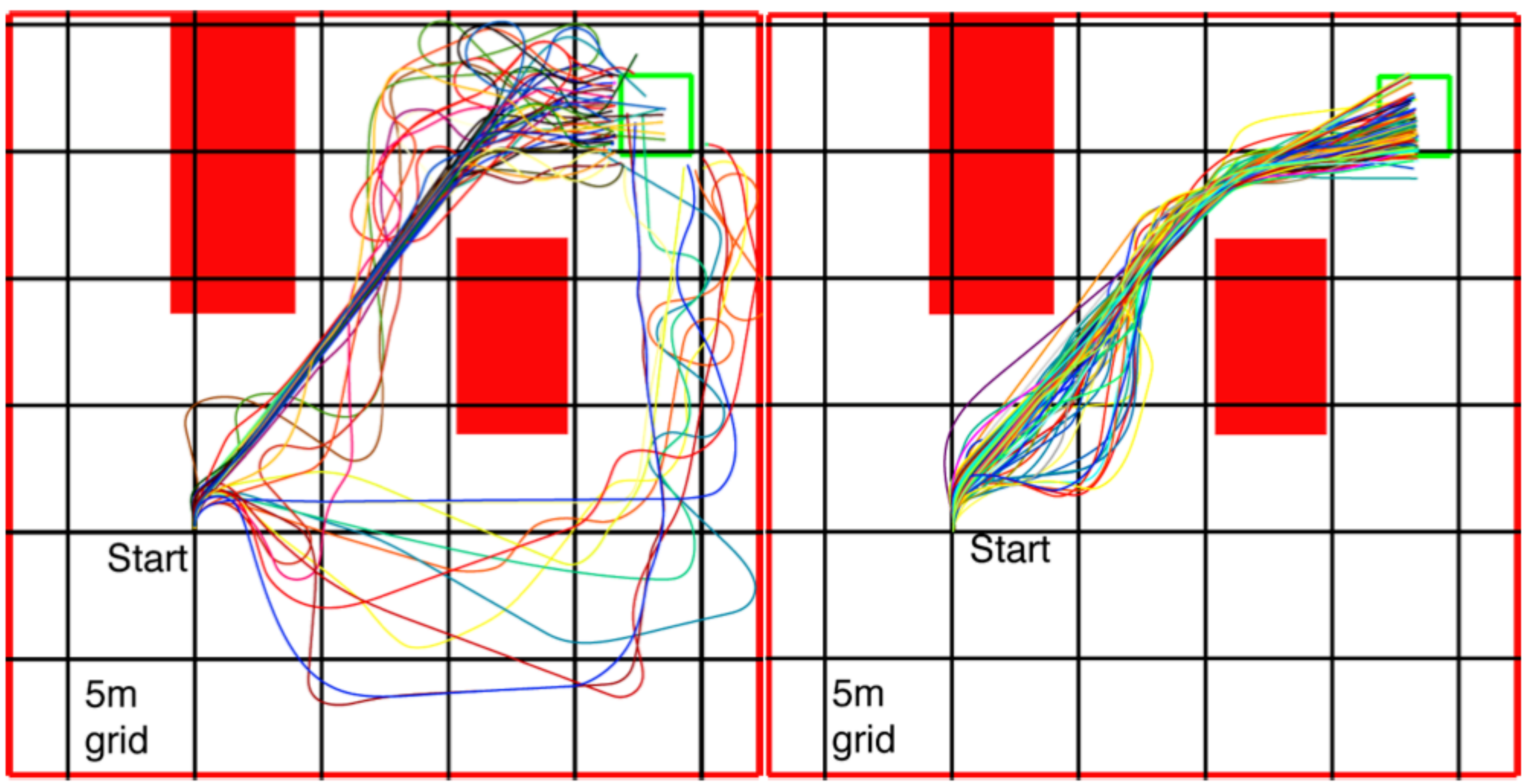
RRT*



RRT* for Dubins car

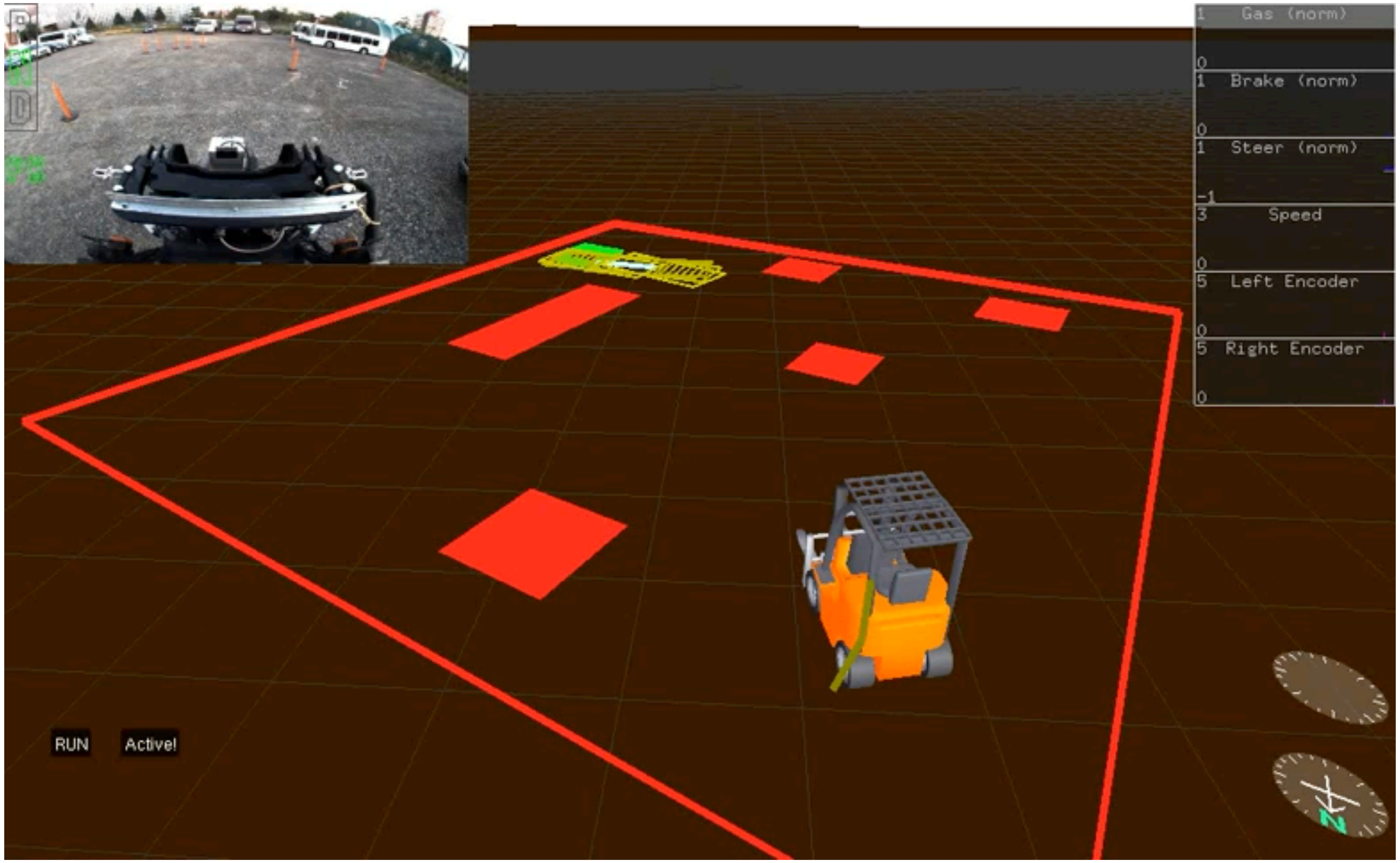
- application to autonomous driving for a forklift

RRT RRT*



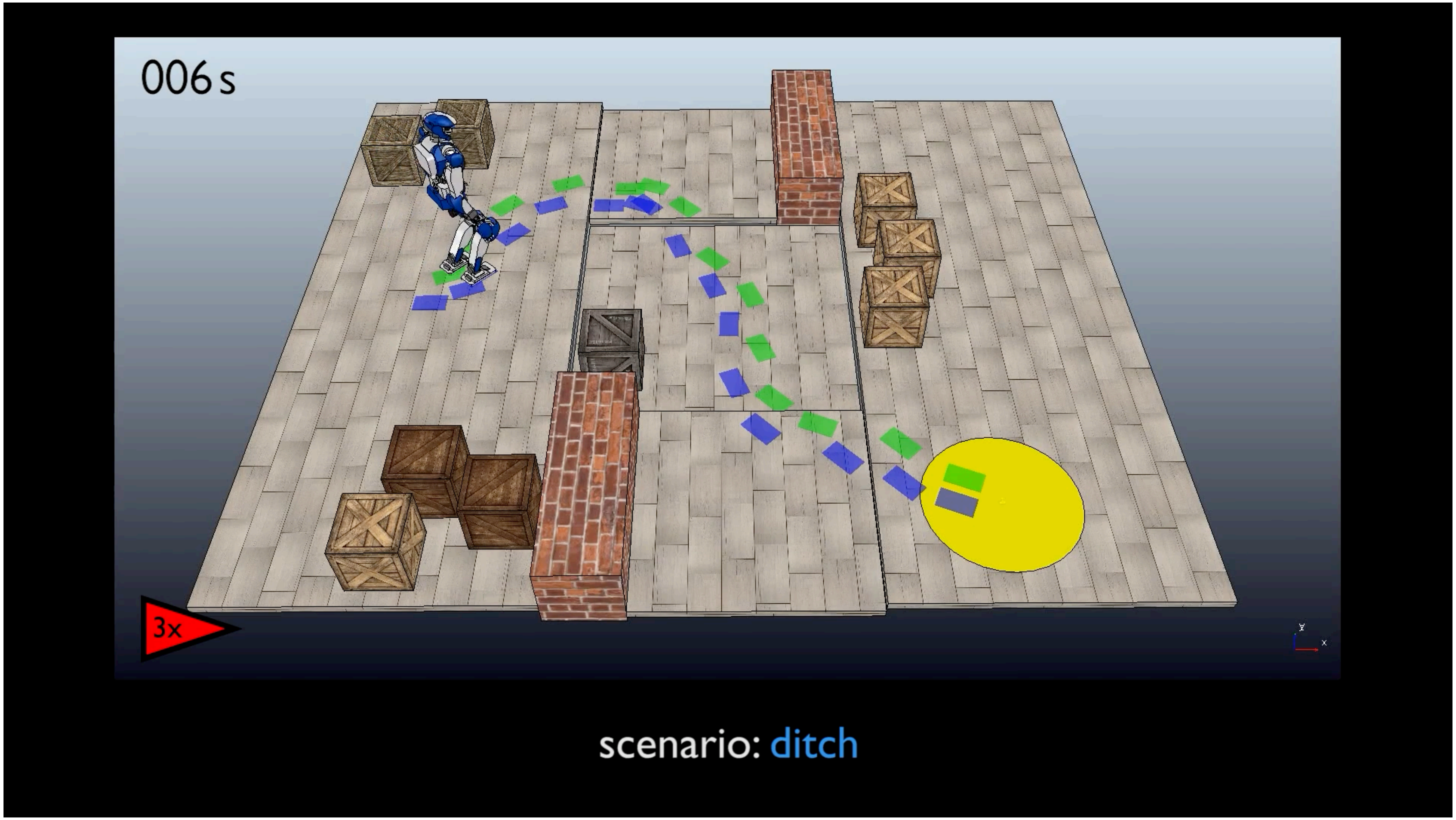
anytime RRT*

- finds an initial solution quickly and then improves it during plan execution by leveraging the asymptotic optimality property of RRT*



RRT*-based footstep planning in humanoids

- generate a sequence of footsteps that leads a humanoid robot to a goal region in a world of stairs



RRT*-based footstep planning in humanoids

- the use of different cost functions leads to footstep plans with different characteristics