CS491/691: Introduction to Aerial Robotics

Topic: Motion Planning Recap

Dr. Kostas Alexis (CSE)

Fundamental motion planning problem

- Consider a dynamical control system defined by an ODE of the form:

$$\frac{dx}{dt} = f(x, u), x(0) = x_{init} \text{ (1)}$$

- Where is x the state, u is the control.
- Given an obstacle set X_{obs}, and a goal set X_{goal}, the objective of the motion planning problem is to find, if it exists, a control signal u such that the solution of (1) satisfies $x(t) \notin X_{obs}$ for all $t \in R^+$, and $x(t) \in X_{goal}$ for all $t > T$, for some finite $T \geq 0$. Return failure if no such control signal exists.

- Basic problem in robotics
- Provably hard: a basic version of it (the Generalized Piano Mover's problem) is known to be PSPACE-hard.

Motion planning in practice

- Many methods have been proposed to solve such problems in practical applications:
  - **Algebraic planners:** Explicit representation of obstacles. Use complicated algebra (visibility computations/projections) to find the path. Complete, but impractical.
  - **Discretization + graph search:** Analytic/grid-based methods do not scale well to high dimensions. Graph search methods (A*, D*, etc.) can be sensitive to graph size. Resolution complete.
  - **Potential fields/navigation functions:** Virtual attractive forces towards the goal, repulsive forces away from the obstacles. No completeness guarantees; unless "navigation functions" are available – very hard to compute in general.
- These algorithms achieve tractability by foregoing completeness altogether, or achieving weaker forms of it, e.g. resolution completeness.

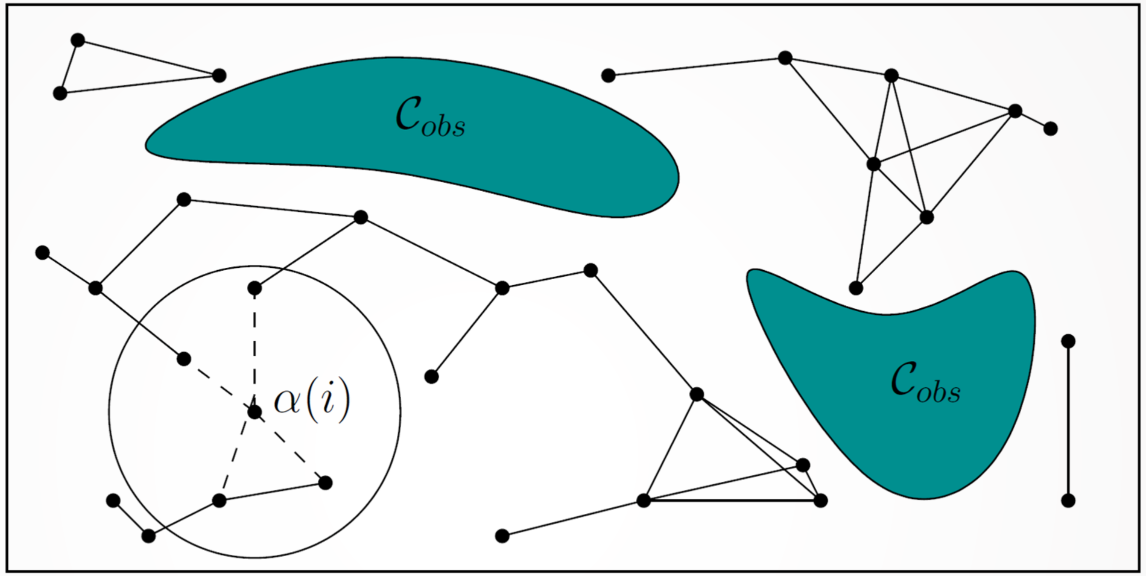
Sampling-based algorithms

- A recently proposed class of motion planning algorithms that has been very successful in practice is based on (batch or incremental) sampling methods:
  - S**olutions are computed based on samples drawn from some distribution.** Sampling algorithms retain some form of completeness, e.g., probabilistic or resolution completeness.
- **Incremental sampling methods are particularly attractive:**
  - Incremental sampling algorithms lend themselves easily to real-time, on-line implementation.
  - Applicable to very generic dynamical systems.
  - Do not require the explicit enumeration of constraints.
  - Adaptively multi-resolution methods (i.e. make your own grid as you go along, up to the necessary resolution).

Probabilistic RoadMaps (PRM)

- Introduced by Kavraki and Latombe in 1994.
- Mainly geared towards "multi-query" motion planning problems.
- **Idea:** build (offline) a graph (i.e., the roadmap) representing the "connectivity" of the environment – use this roadmap to find paths quickly at run-time.
- **Learning/pre-processing phase:**
  - Sample n points from $X_{free} = [0,1]^d \backslash X_{obs}$.
  - Try to connect these points using a fast "local planner" (e.g. ignore obstacles).
  - If connection successful (i.e. no collisions), add an edge between the points.
- **At run-time:**
  - Connect the start and end goal to the closest nodes in the roadmap.
  - Find a path on the roadmap.
- *First planner ever to demonstrate the ability to solve generic planning problems in > 4-5 dimensions!*

Probabilistic RoadMap example



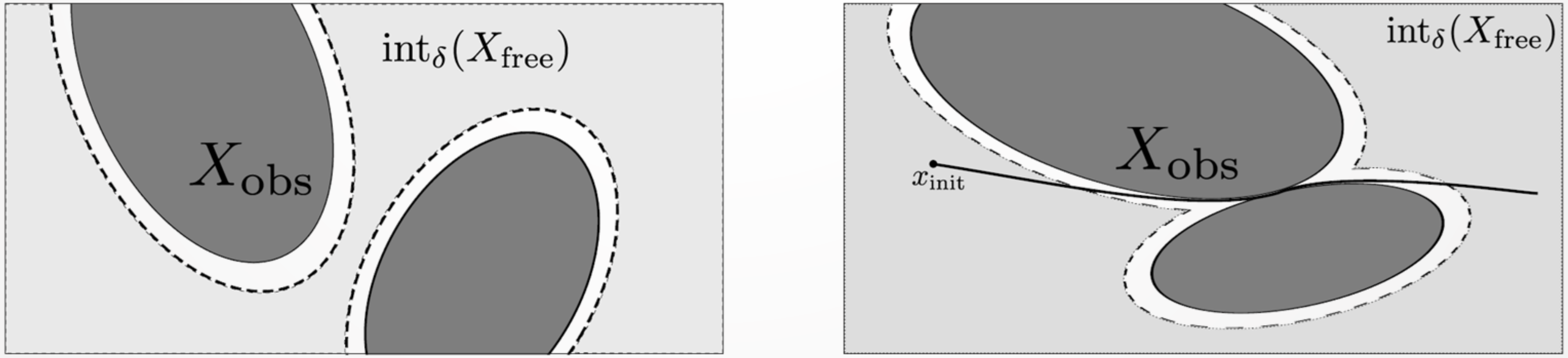
- **"Practical" algorithm:**
  - Incremental construction.
  - Connects points within a radius r, starting from "closest" ones.
  - Do not attempt to connect points that are already on the same connected component of the RPM.
- *What kind of properties does this algorithm have? Will it find a solution if there is one? Will that be an optimal solution? What is the complexity of the algorithm?*

Probabilistic Completeness

- Definition – Probabilistic Completeness:
- An algorithm ALG is probabilistically complete if, for any robustly feasible motion planning problem defined by $P = (X_{free}, x_{init}, X_{goal})$, then:

$$\lim_{N\to\infty} \Pr(ALG\ returns\ a\ solution\ P) = 1$$

- A "relaxed" notion of completeness
- Applicable to motion planning problems with a robust solution. A robust solutions remains a valid solution even when the obstacles are "dilated" by small small δ.



robust

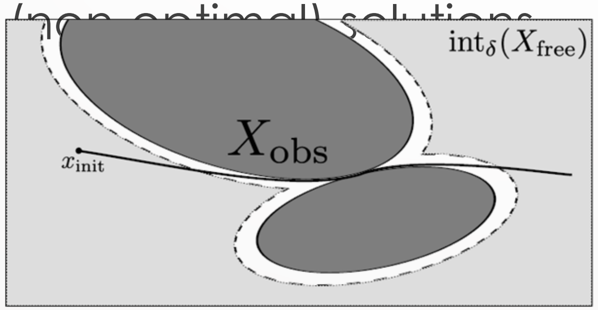
NOT robust

Asymptotic Optimality

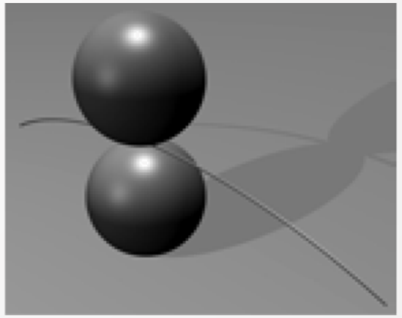
- Definition – Asymptotic Optimality:
- An algorithm ALG is asymptotically optimal if, for any motion planning problem defined by $P = \left(X_{free}, x_{init}, X_{goal}\right)$ and function c that admit a robust optimal solution with finite cost c^*,

$$P\left(\left\{\lim_{i \to \infty} Y_i^{ALG} = c^*\right\}\right) = 1$$

- The function c associates to each path σ a non-negative $c(\sigma)$, e.g. $c(\sigma) = \int_{\sigma} \mathrm{X}(s)ds$
- The definition is applicable to optimal motion planning problem with a robust optimal solution. A robust optimal solution is such that it can be obtained as a limit of robust (non-optimal) solutions.



NOT robust


robust

Simple PRM (sPRM)

sPRM Algorithm

$V \leftarrow \{x_{init}\} \cup \{SampleFree_i\}_{i=1,,\ldots,N-1}; E \leftarrow 0;$
foreach $v \in V$ **do**:
    $U \leftarrow Near(G = (V, E), v, r)\backslash\{v\};$
    foreach $u \in U$ **do**:
        if $CollisionFree(v, u)$ **then** $E \leftarrow E \cup \{(v, u)\}, (u, v)\}$
return $G = (V, E);$

- The simplified version of the PRM algorithm has been shown to be probabilistically complete.
- Moreover, the probability of success goes to 1 exponentially fast, if the environment satisfies “good visibility” conditions.
- New key concept: combinatorial complexity vs “visibility”.

Remarks on PRM

- sPRM is **probabilistically complete** and asymptotically optimal.
- PRM is probabilistically complete **but NOT asymptotically optimal**.
- **Complexity for N samples:** $O(N^2)$.

Rapidly-exploring Random Trees

- Introduced by LaValle and Kuffner in 1998.
- Appropriate for single-query planning problems.
- **Idea:** build (online) a tree, exploring the region of the state space that can be reached from the initial condition.

- **At each step:** sample one point from X_{free}, and try to connect it to the closest vertex in the tree.
- Very effective in practice, “Voronoi bias”.

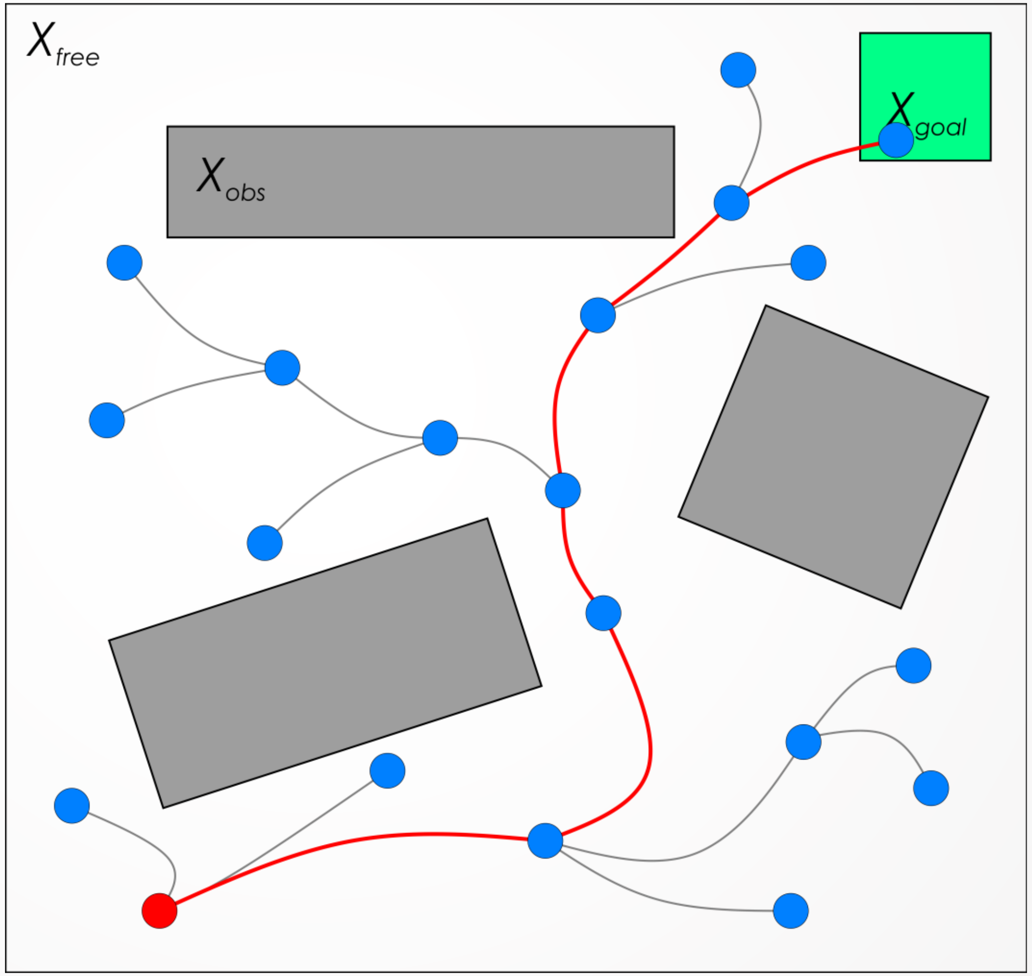
Rapidly-exploring Random Trees

RRT

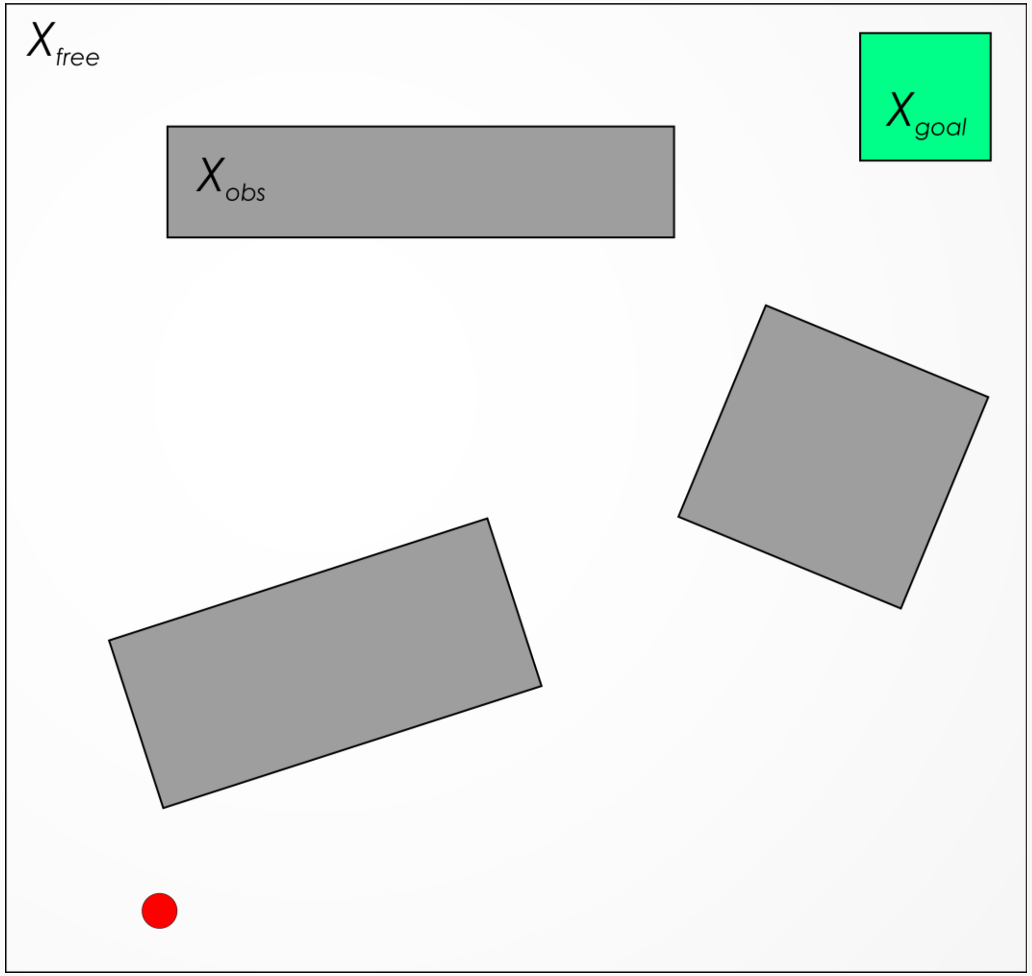
$V \leftarrow \{x_{init}\}; E \leftarrow 0;$
for i=1,…,N **do**:
  $x_{rand} \leftarrow SampleFree;$
  $x_{nearest} \leftarrow Nearest(G = (V, E), x_{rand});$
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand});$
  if $ObstacleFree(x_{nearest}, x_{new})$ **then**:
    $V \leftarrow V \cup \{x_{new}\}; E \leftarrow E \cup \{(x_{nearest}, x_{new})\};$
return $G = (V, E);$

- **The RRT algorithm is probabilistically complete**
- The probability of success goes to 1 exponentially fast, if the environment satisfies certain "good visibility" conditions.

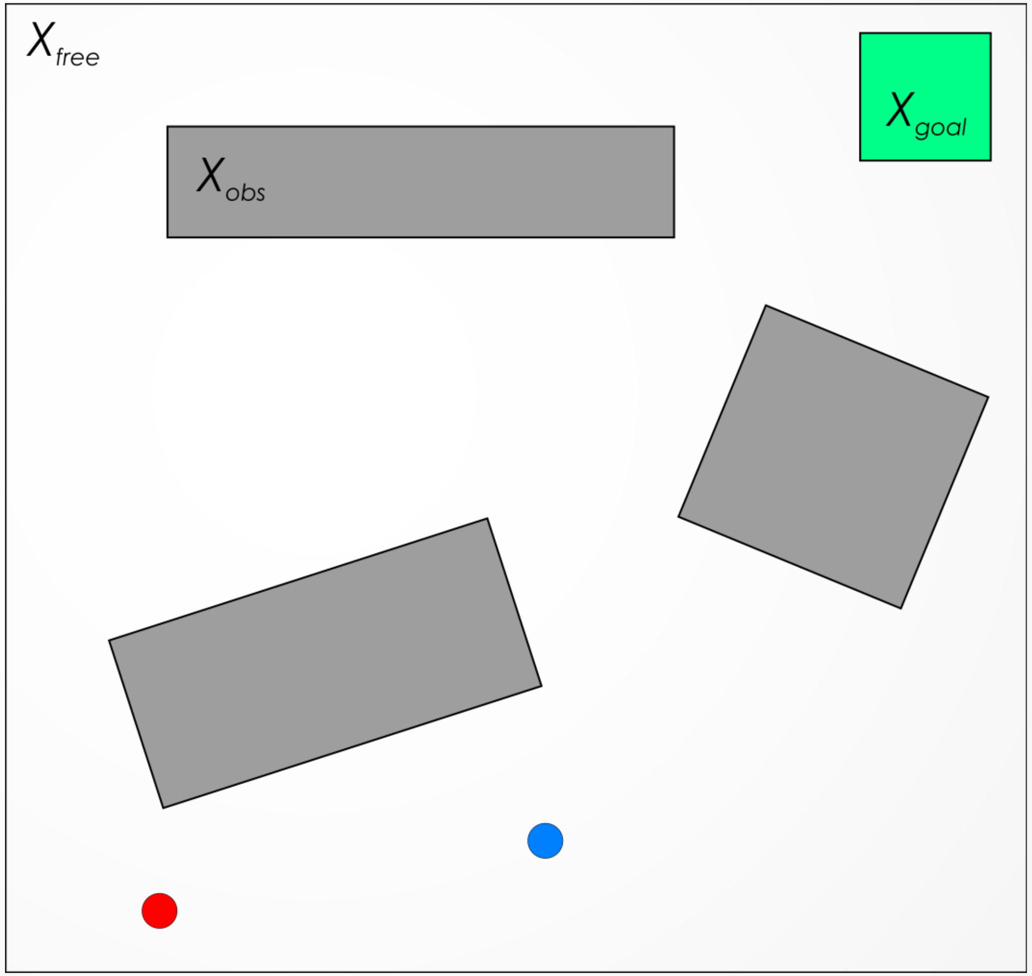
Rapidly-exploring Random Trees (RRTs)



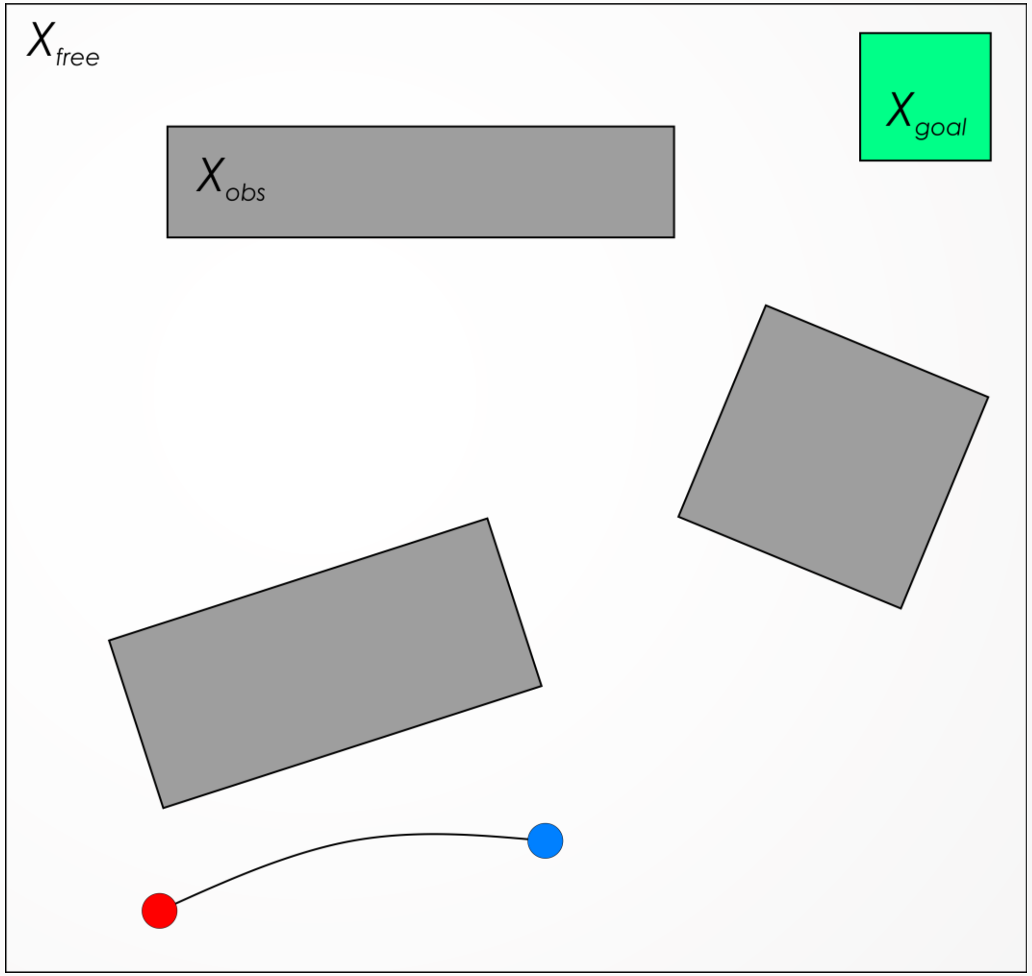
Rapidly-exploring Random Trees (RRTs)



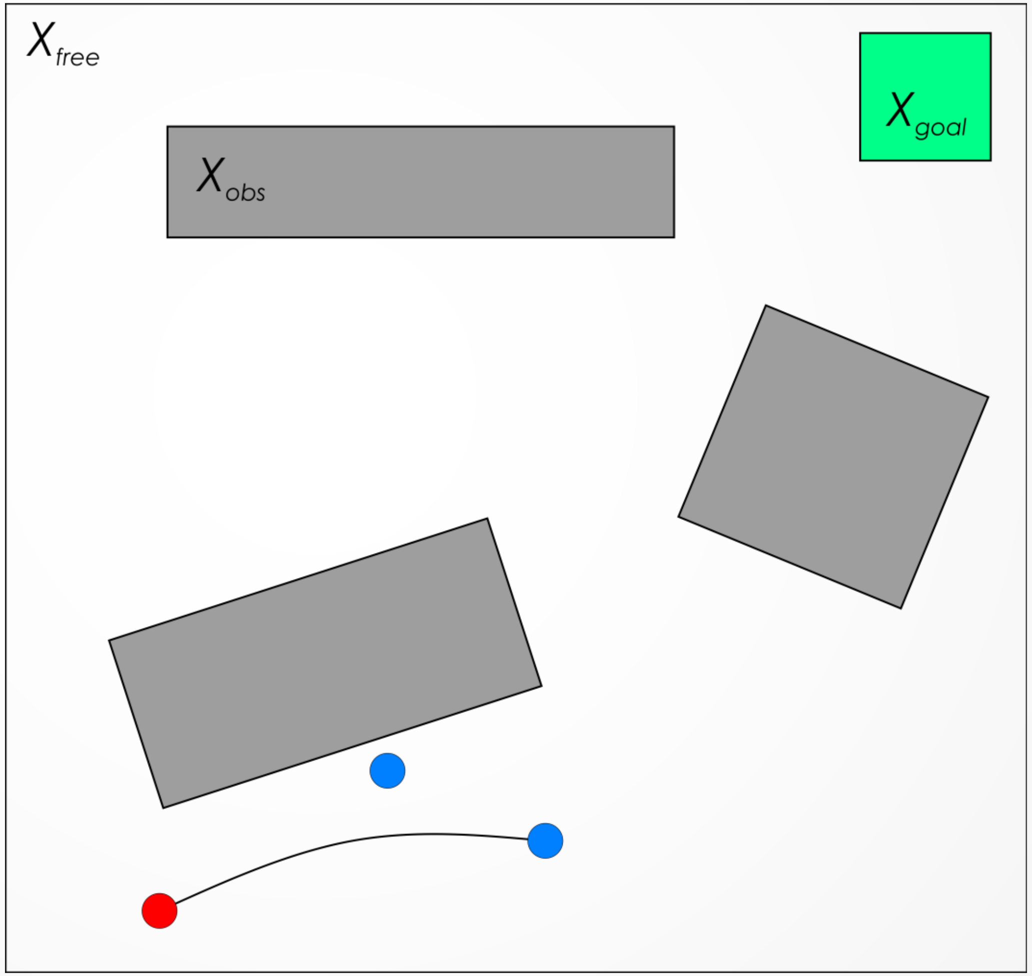
Rapidly-exploring Random Trees (RRTs)



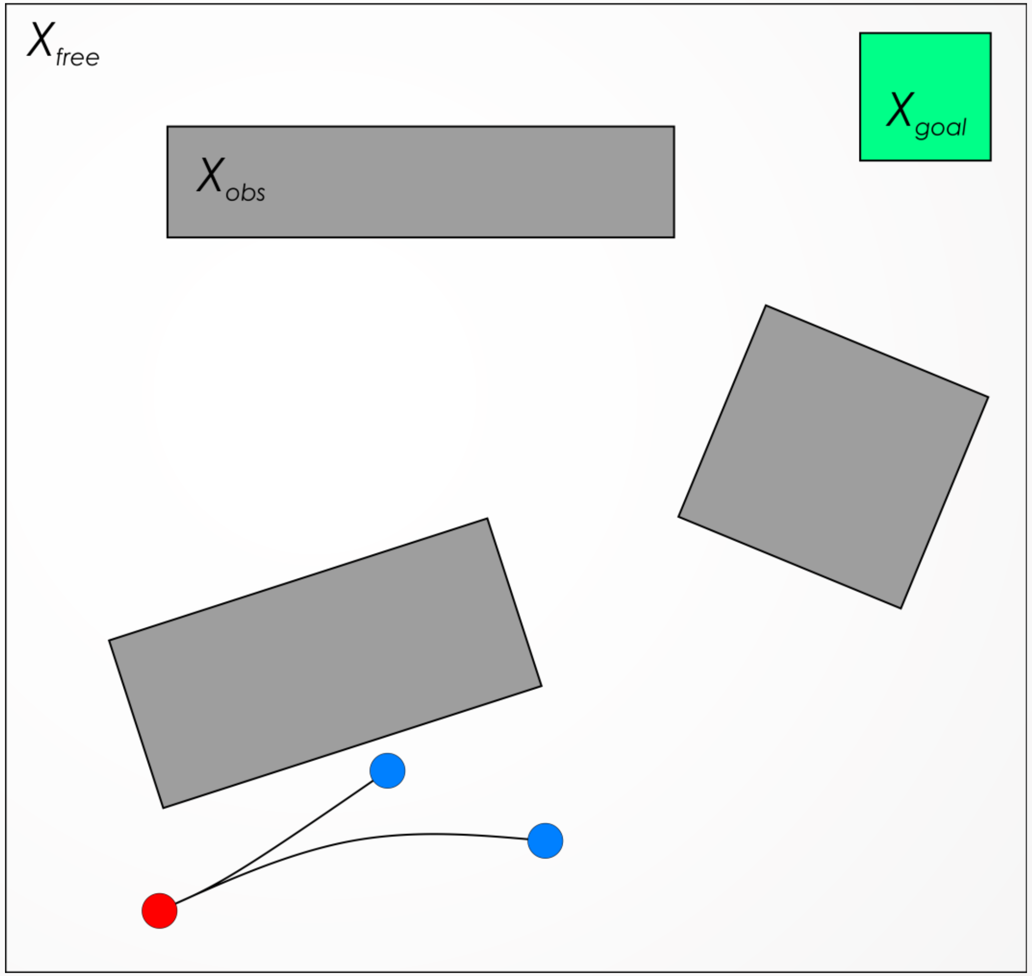
Rapidly-exploring Random Trees (RRTs)



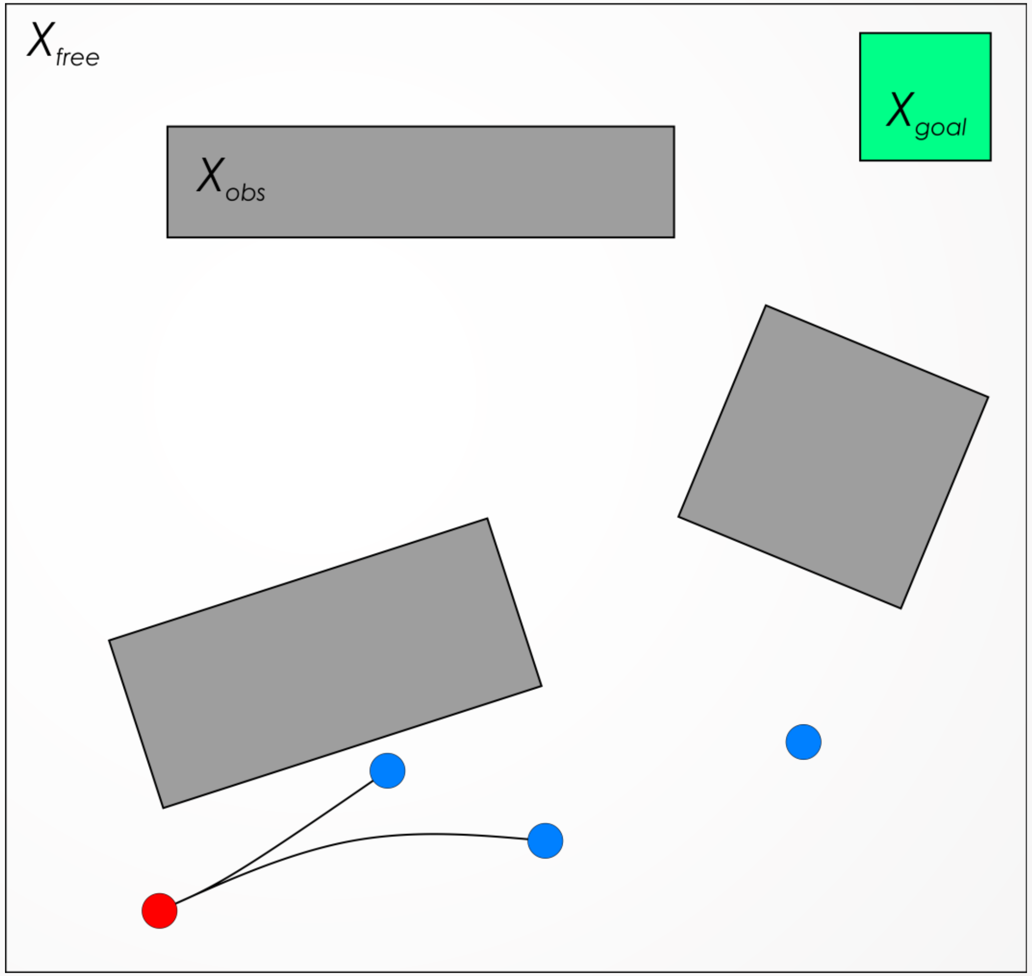
Rapidly-exploring Random Trees (RRTs)



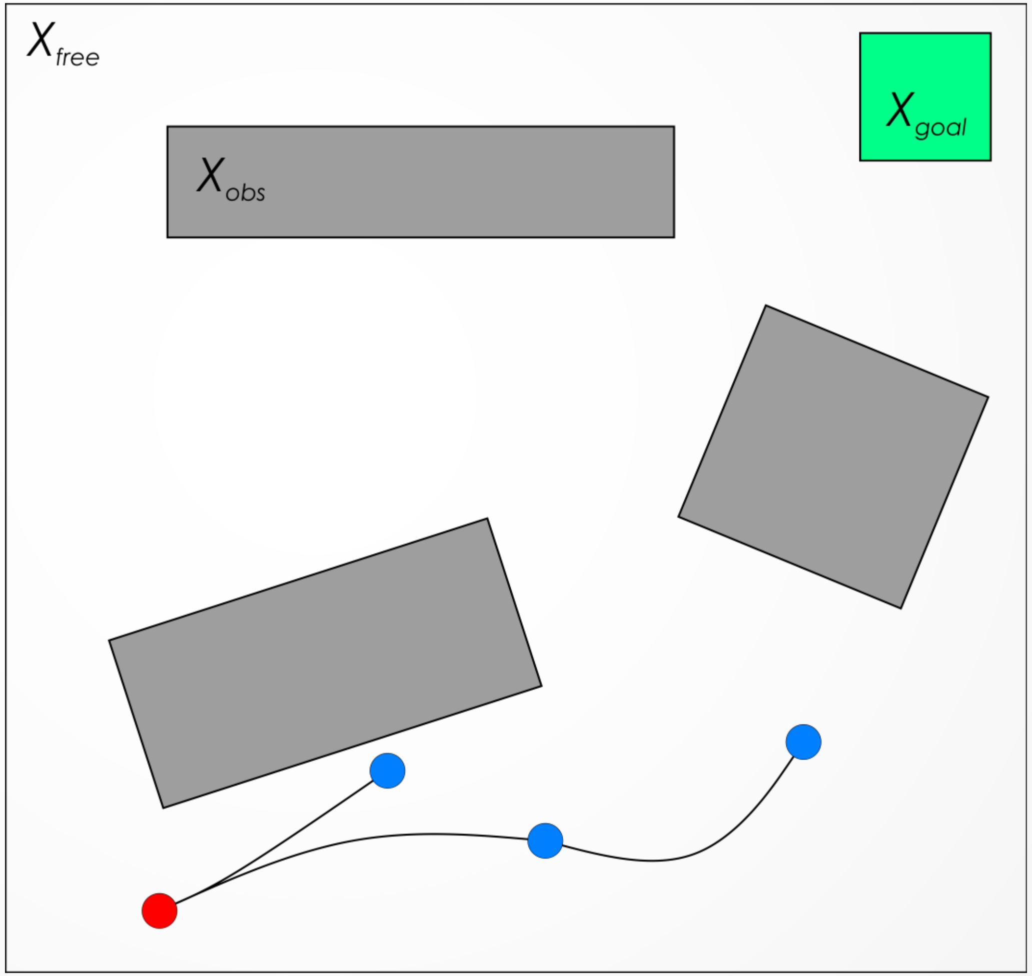
Rapidly-exploring Random Trees (RRTs)



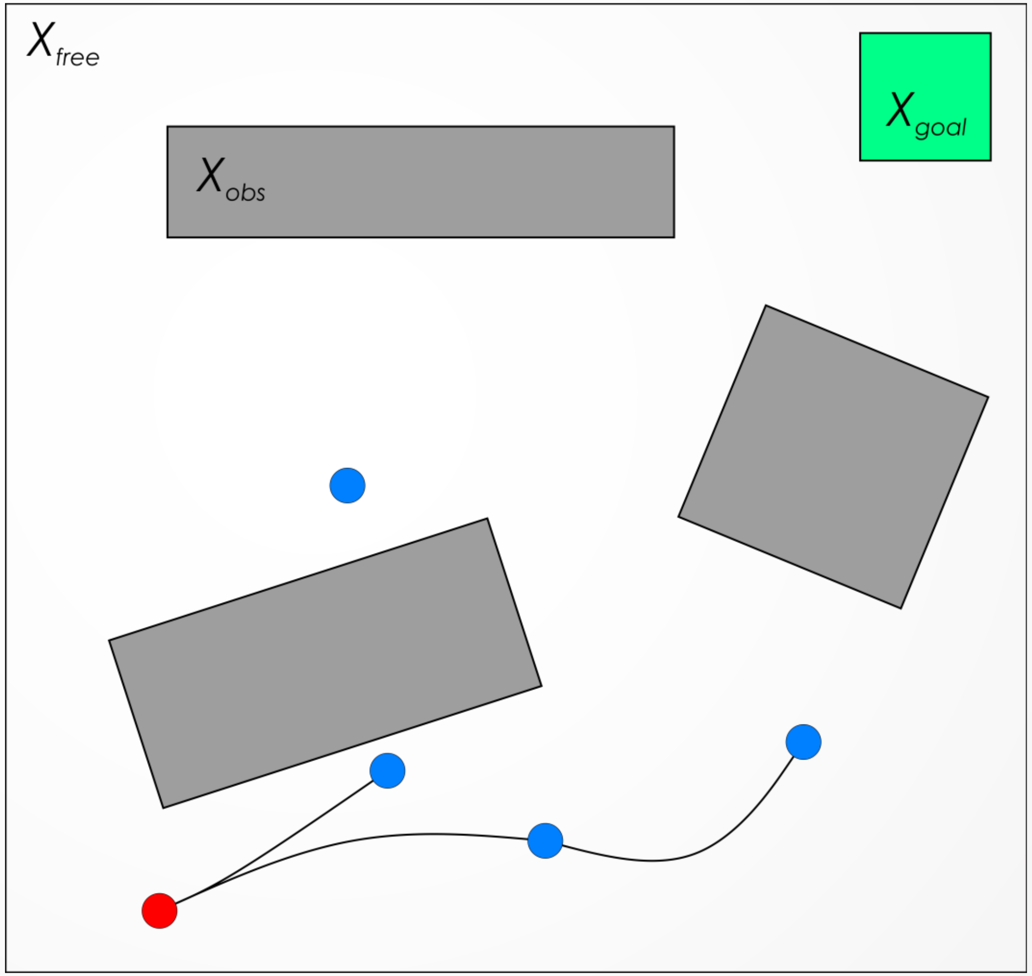
Rapidly-exploring Random Trees (RRTs)



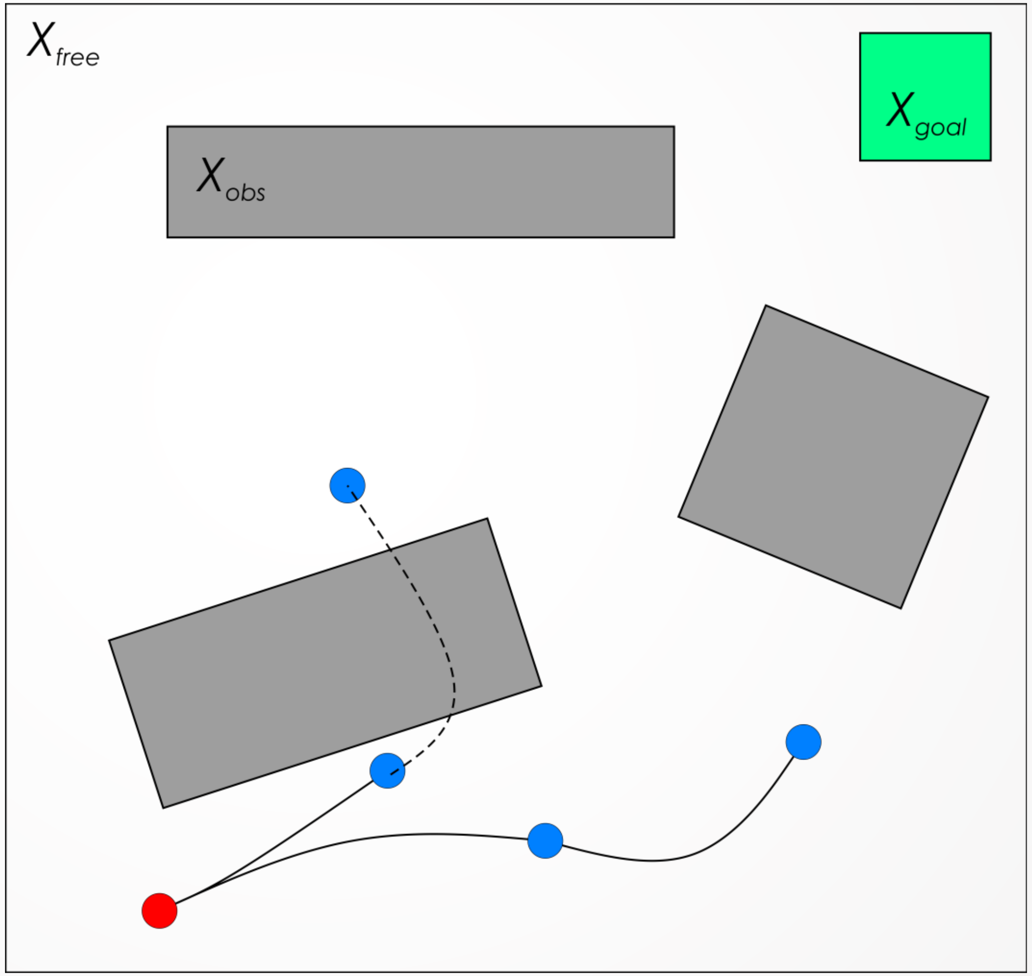
Rapidly-exploring Random Trees (RRTs)



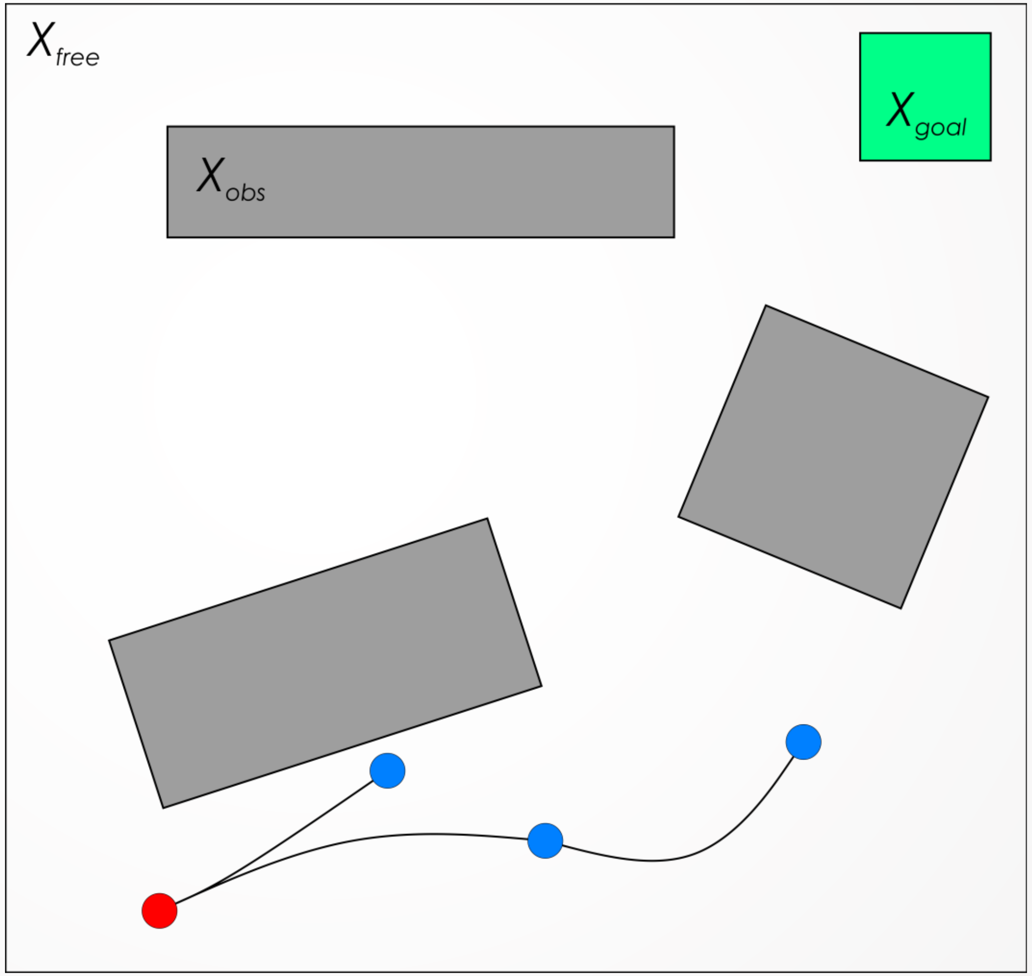
Rapidly-exploring Random Trees (RRTs)



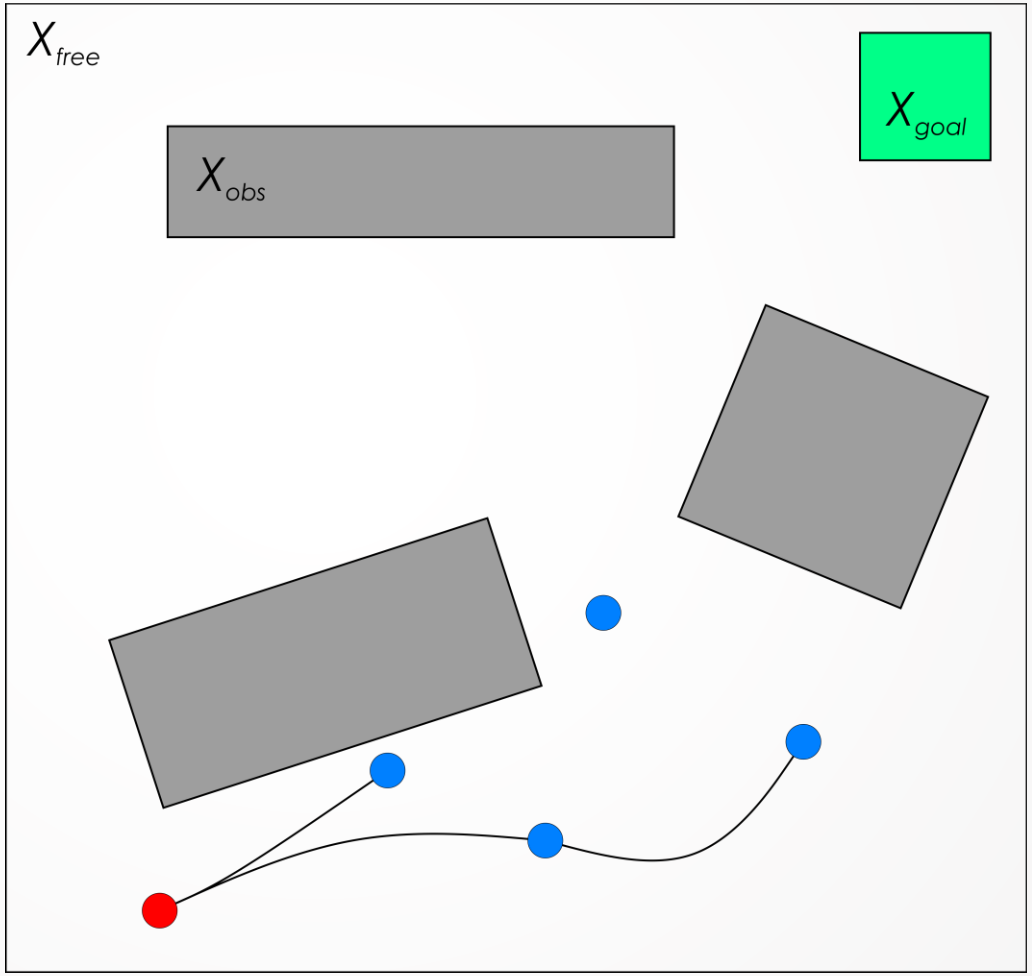
Rapidly-exploring Random Trees (RRTs)



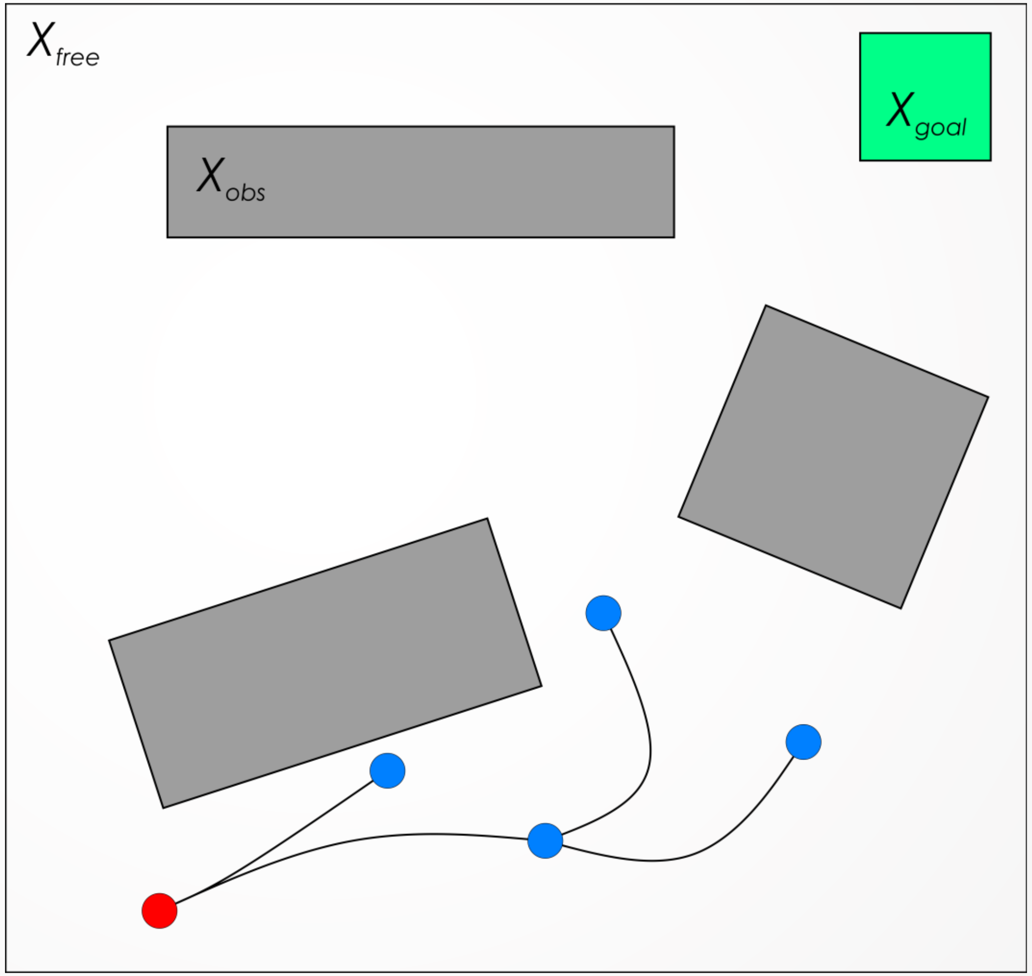
Rapidly-exploring Random Trees (RRTs)



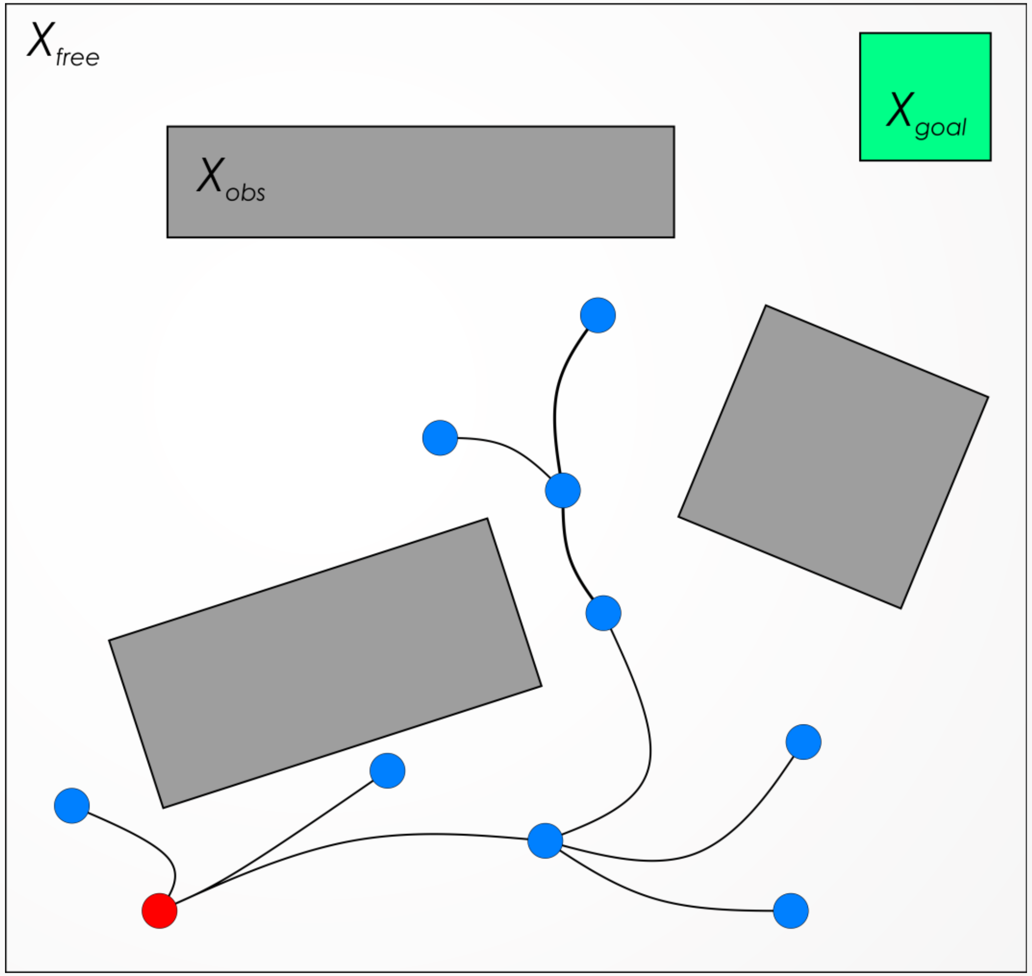
Rapidly-exploring Random Trees (RRTs)



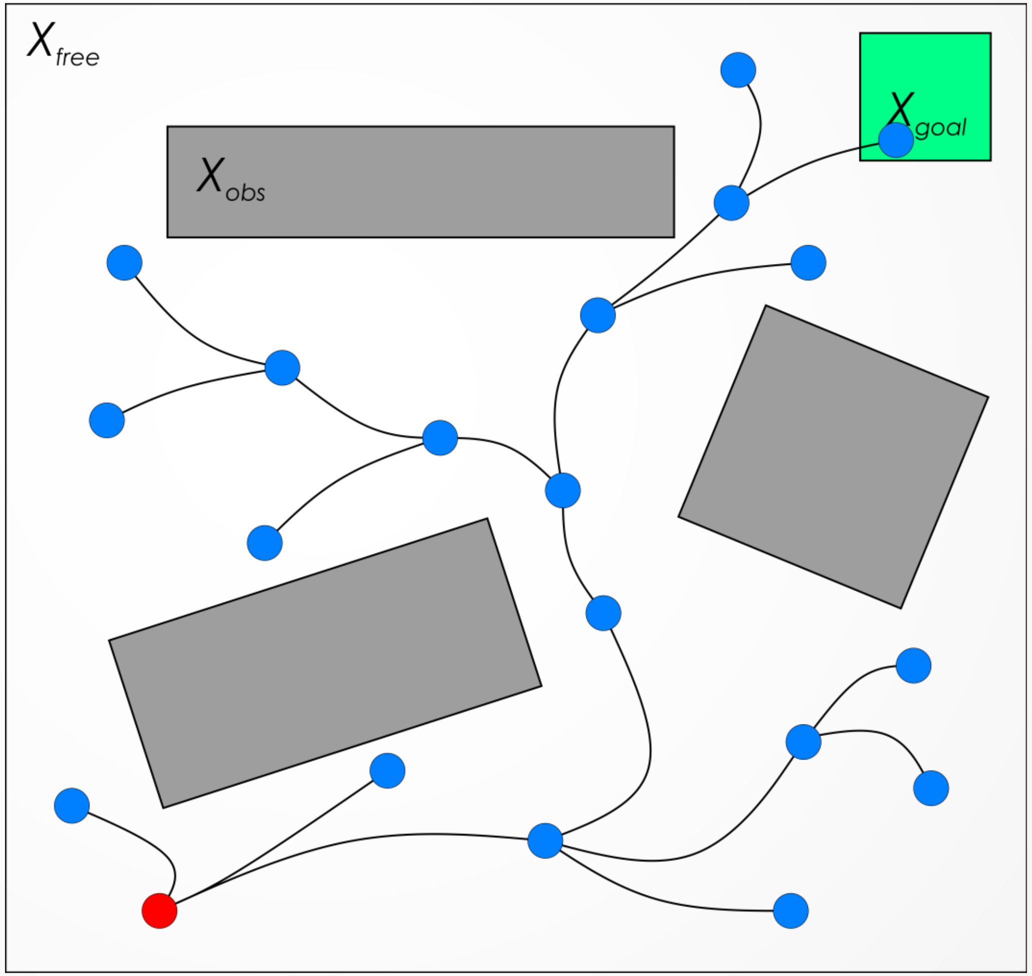
Rapidly-exploring Random Trees (RRTs)



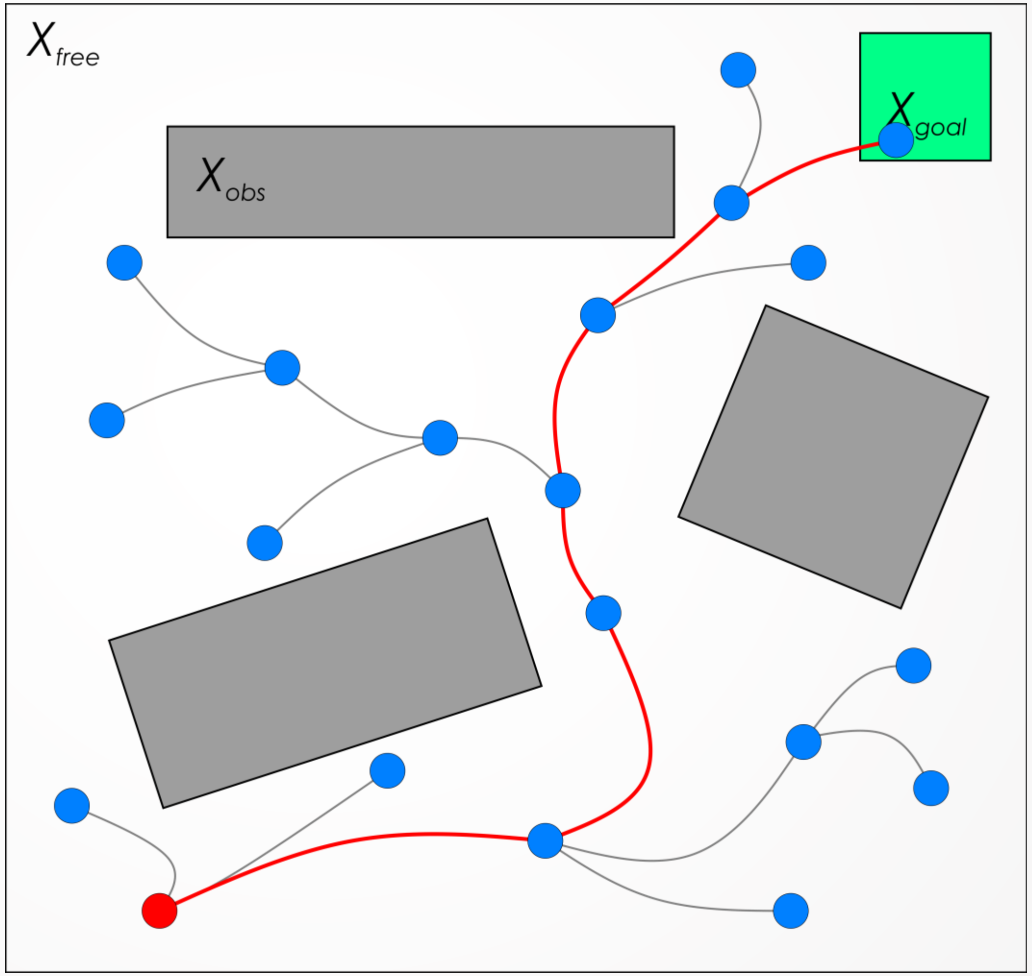
Rapidly-exploring Random Trees (RRTs)



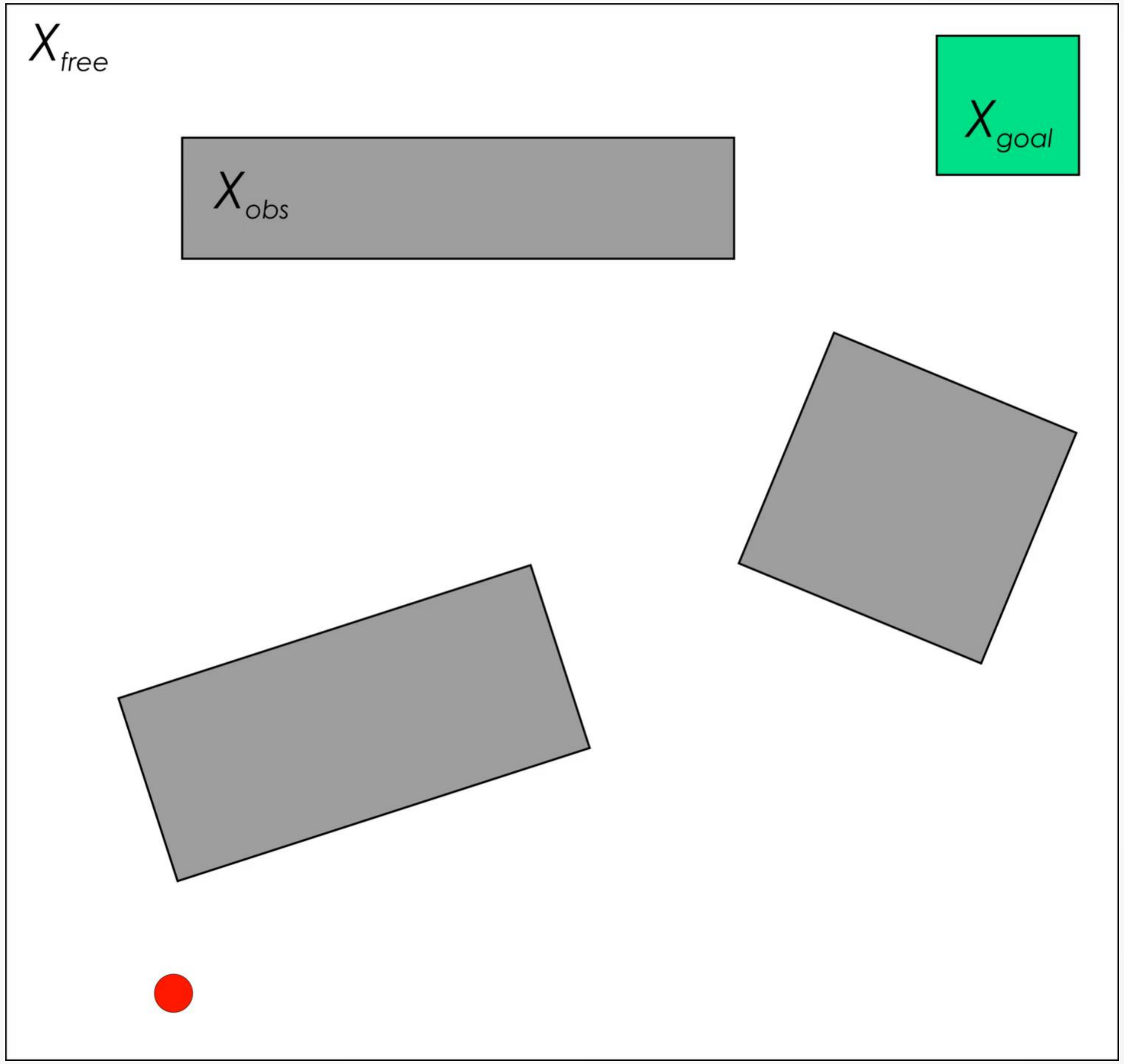
Rapidly-exploring Random Trees (RRTs)



Rapidly-exploring Random Trees (RRTs)



Rapidly-exploring Random Trees (RRTs)



Some remarks on that negative result

- **Intuition:** RRT does not satisfy a necessary condition for asymptotic optimality, i.e., that the root node has infinitely many subtrees that extend at least a distance ϵ away from x_{init}.
- The RRT algorithm "traps" itself by disallowing new better paths to emerge.
- **Heuristics such as**
  - Running the RRT multiple times
  - Running multiple times concurrently
  - Deleting and rebuilding parts of the tree etc.

  Work better than the standard RRT, but cannot remove the sub-optimal behavior.

- **How can we do better?**

Rapidly-exploring Random Graphs (RRGs)

RRG Algorithm

$V \leftarrow \{x_{init}\}; E \leftarrow 0;$
for i=1,…,N **do**:
  $x_{rand} \leftarrow SampleFree;$
  $x_{nearest} \leftarrow Nearest(G = (V, E), x_{rand});$
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand});$
  if $ObstacleFree(x_{nearest}, x_{new})$ **then**:
    $X_{near} \leftarrow Near\left(G = (V, E), x_{new}, \min\{\gamma_{RRG}\left(\frac{\log(card\, V)}{card\, V}\right)^{1/d}, \eta\}\right);$
    $V \leftarrow V \cup \{x_{new}\}; E \leftarrow E \cup \{(x_{nearest}, x_{new}), (x_{nearest}, x_{new})\};$
    foreach $x_{near} \in X_{near}$ **do:**
      if $CollisionFree(x_{near}, x_{new})$ then $E \leftarrow E \cup \{(x_{near}, x_{new}), (x_{near}, x_{new})\};$
return $G = (V, E);$

- At each iteration, the RRG tries to connect to the new sample all vertices in a ball radius r_n centered at it. (Or simply default to the nearest one if such a ball is empty).
- In general, the RRG builds graphs with cycles.

Properties of RRGs

Theorem – Probabilistic completeness

Since $V_n^{RRG} = V_n^{RRT}$, for all n, it follows that RRG has the same completeness properties of RRT, i.e.

$$\Pr\left[V_n^{RRG} \cap X_{goal} = 0\right] = O(e^{-bn})$$

Theorem – Asymptotic optimality

If the $Near$ procedure returns all nodes in V within a ball of volume

$$Vol = \gamma \frac{\log n}{n}, \gamma > 2^d(1 + {}^1/_d),$$

Under some additional technical assumptions (e.g., on the sampling distribution, on the ϵ clearance of the optimal path, and on the continuity of the cost function), the best path in the RRG converges to an optimal solution almost surely, i.e.:

$$\Pr[Y_\infty^{RRG} = c^*] = 1$$

Computational complexity

- At each iteration, the RRG algorithm executes $O(\log n)$ extra calls to $ObstacleFree$ when compared to the RRT.
- However, the complexity of the $Nearest$ procedure is $\Omega(\log n)$. Achieved if using, e.g., a Balanced-Box Decomposition (BBD) Tree.

Theorem – Asymptotic (Relative) Complexity

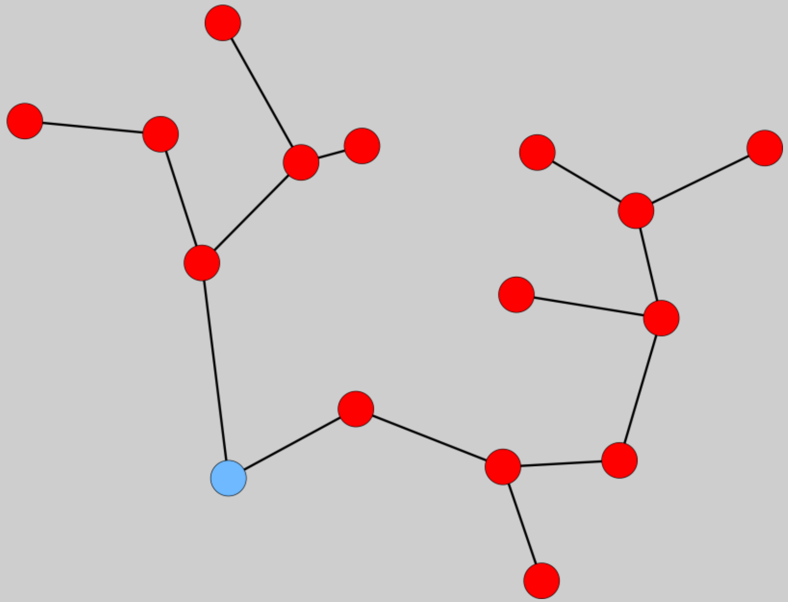
There exists a constant $\beta \in \mathbb{R}_+$ such that

$$\lim_{i\to\infty} sup\, E\left[\frac{OPS_i^{RRG}}{OPS_i^{RRT}}\right] \leq \beta$$

- In other words, the RRG algorithm has **no substantial computational overhead** over RRT, and ensures asymptotic optimality.

RRT*: A tree version of the RRG

- RRT algorithm can account for nonholonomic dynamics and modeling errors.
- RRG requires connecting the nodes exactly, i.e., the *Steer* procedure has to be exact. Exact steering methods are not available for general dynamic systems.

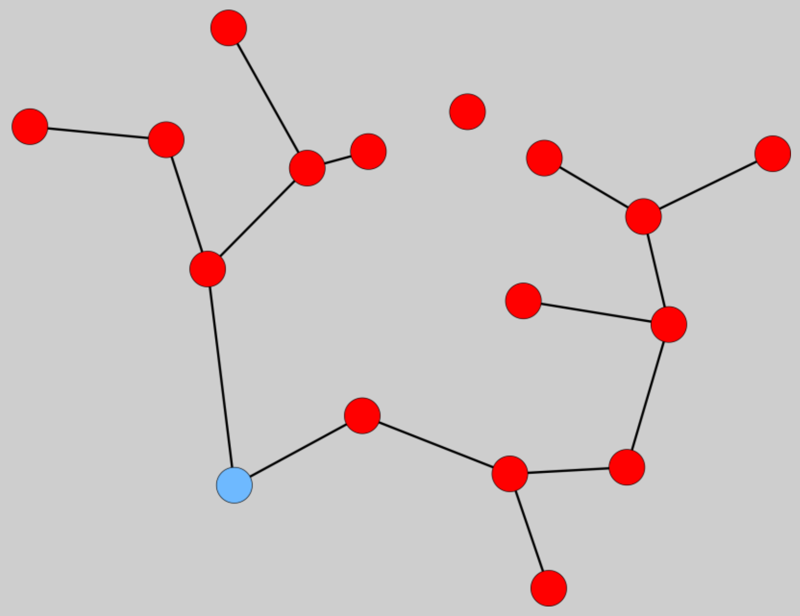
RRT* Algorithm	
• RRT* is a variant of RRG that essentially "rewires" the tree as better paths are discovered. • After rewiring the cost has to be propagated along the leaves. • If steering errors occur, subtrees can be re-computed. • The RRT* algorithm inherits the asymptotic optimality and rapid exploration properties of the RRG and RRT.	

RRT*: A tree version of the RRG

- RRT algorithm can account for nonholonomic dynamics and modeling errors.
- RRG requires connecting the nodes exactly, i.e., the $Steer$ procedure has to be exact. Exact steering methods are not available for general dynamic systems.

RRT* Algorithm

- RRT* is a variant of RRG that essentially "rewires" the tree as better paths are discovered.
- After rewiring the cost has to be propagated along the leaves.
- If steering errors occur, subtrees can be re-computed.
- The RRT* algorithm inherits the asymptotic optimality and rapid exploration properties of the RRG and RRT.

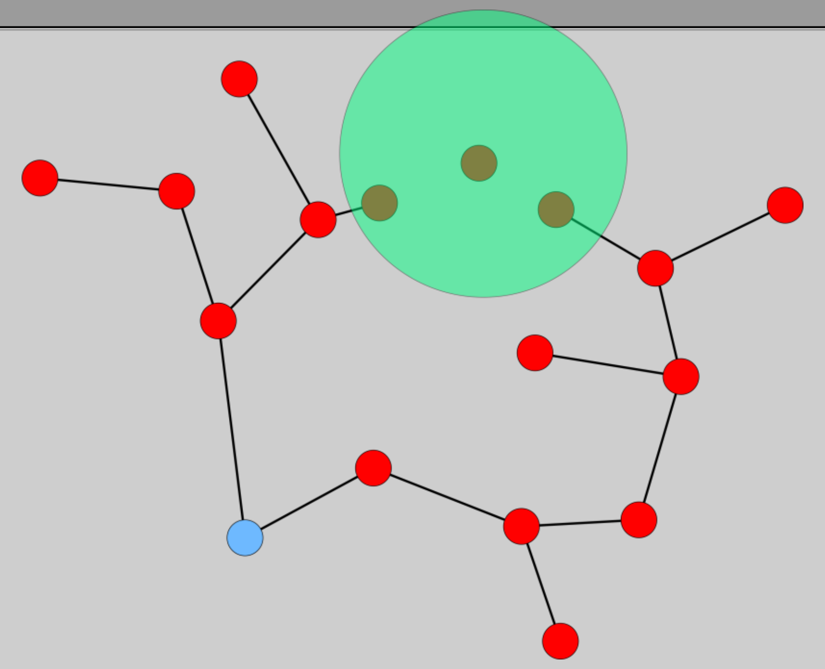


RRT*: A tree version of the RRG

- RRT algorithm can account for nonholonomic dynamics and modeling errors.
- RRG requires connecting the nodes exactly, i.e., the $Steer$ procedure has to be exact. Exact steering methods are not available for general dynamic systems.

RRT* Algorithm

- RRT* is a variant of RRG that essentially "rewires" the tree as better paths are discovered.
- After rewiring the cost has to be propagated along the leaves.
- If steering errors occur, subtrees can be re-computed.
- The RRT* algorithm inherits the asymptotic optimality and rapid exploration properties of the RRG and RRT.

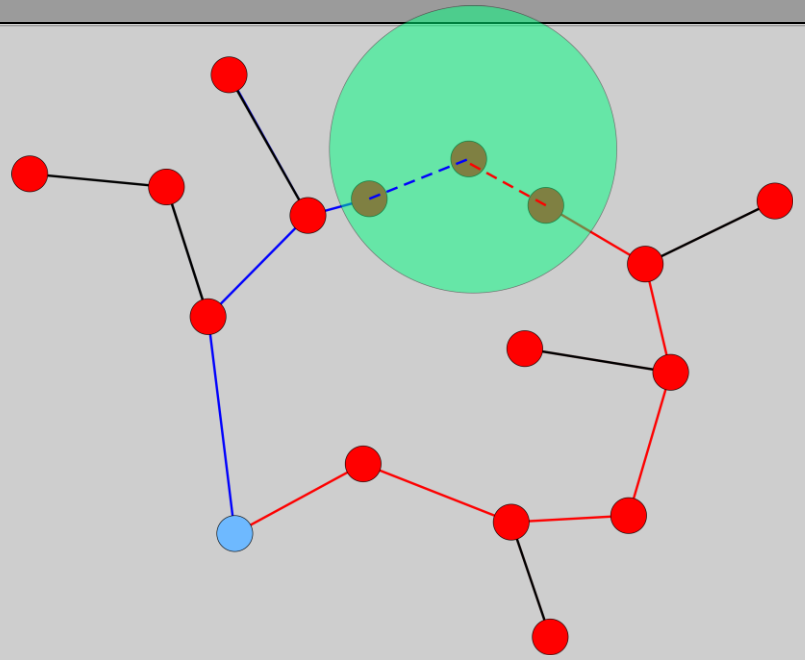


RRT*: A tree version of the RRG

- RRT algorithm can account for nonholonomic dynamics and modeling errors.
- RRG requires connecting the nodes exactly, i.e., the $Steer$ procedure has to be exact. Exact steering methods are not available for general dynamic systems.

RRT* Algorithm

- RRT* is a variant of RRG that essentially "rewires" the tree as better paths are discovered.
- After rewiring the cost has to be propagated along the leaves.
- If steering errors occur, subtrees can be re-computed.
- The RRT* algorithm inherits the asymptotic optimality and rapid exploration properties of the RRG and RRT.

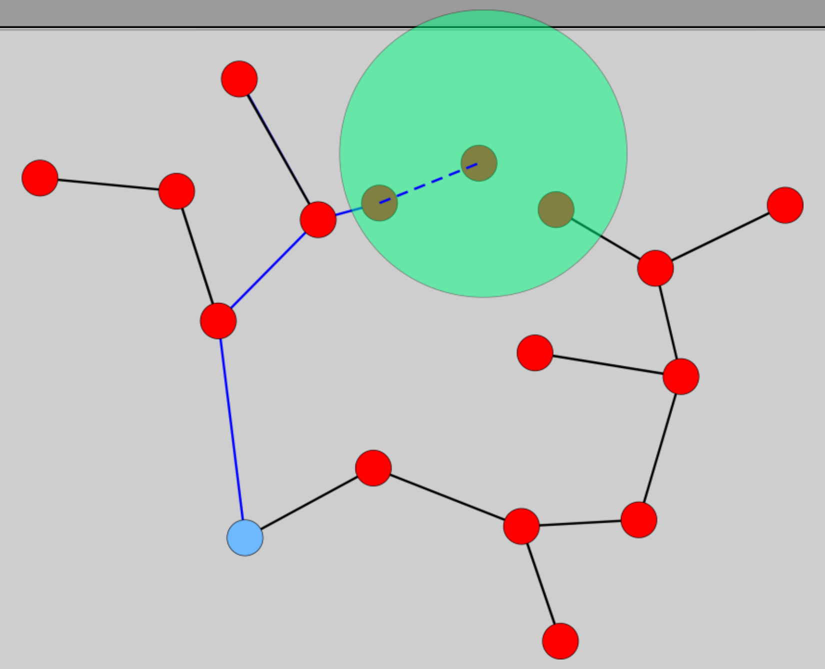


RRT*: A tree version of the RRG

- RRT algorithm can account for nonholonomic dynamics and modeling errors.
- RRG requires connecting the nodes exactly, i.e., the $Steer$ procedure has to be exact. Exact steering methods are not available for general dynamic systems.

RRT* Algorithm

- RRT* is a variant of RRG that essentially "rewires" the tree as better paths are discovered.
- After rewiring the cost has to be propagated along the leaves.
- If steering errors occur, subtrees can be re-computed.
- The RRT* algorithm inherits the asymptotic optimality and rapid exploration properties of the RRG and RRT.

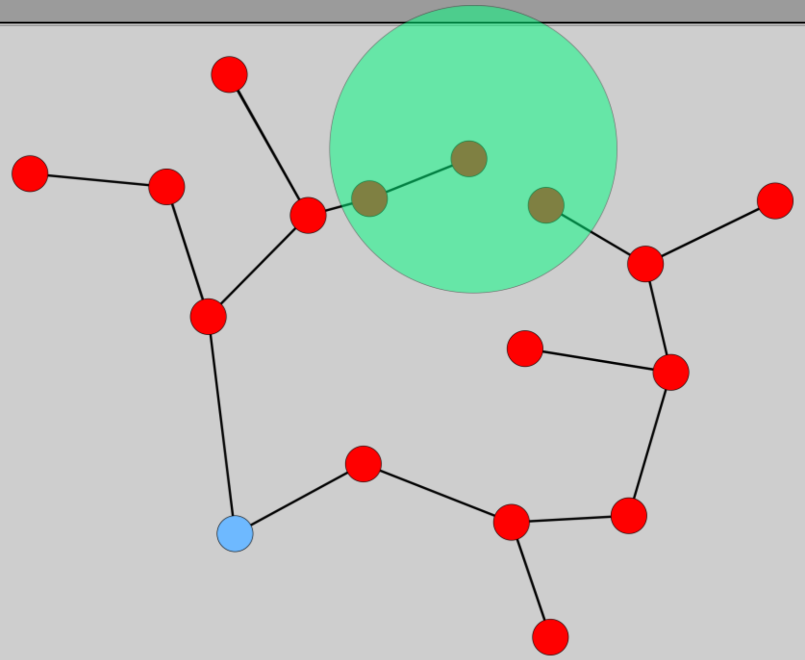


RRT*: A tree version of the RRG

- RRT algorithm can account for nonholonomic dynamics and modeling errors.
- RRG requires connecting the nodes exactly, i.e., the *Steer* procedure has to be exact. Exact steering methods are not available for general dynamic systems.

RRT* Algorithm

- RRT* is a variant of RRG that essentially "rewires" the tree as better paths are discovered.
- After rewiring the cost has to be propagated along the leaves.
- If steering errors occur, subtrees can be re-computed.
- The RRT* algorithm inherits the asymptotic optimality and rapid exploration properties of the RRG and RRT.

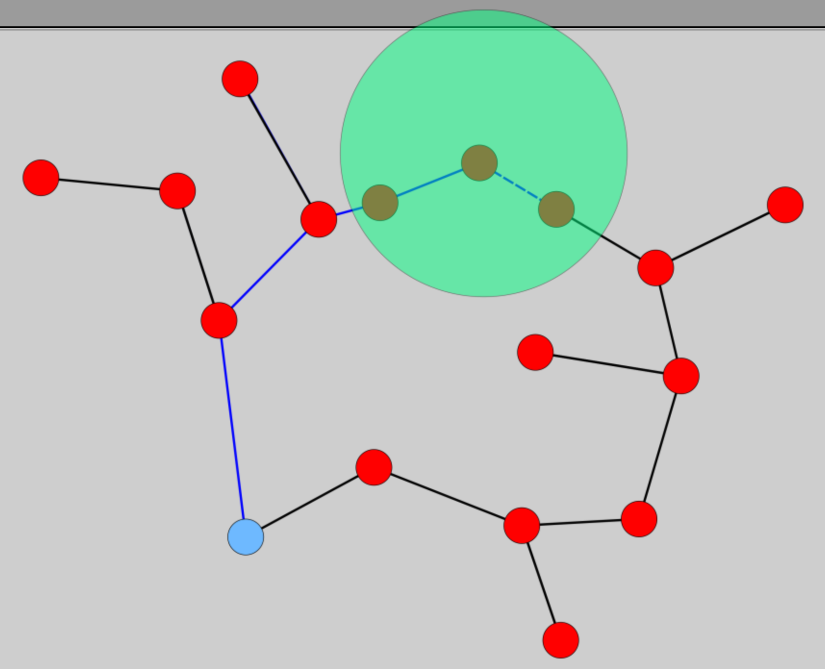


RRT*: A tree version of the RRG

- RRT algorithm can account for nonholonomic dynamics and modeling errors.
- RRG requires connecting the nodes exactly, i.e., the $Steer$ procedure has to be exact. Exact steering methods are not available for general dynamic systems.

RRT* Algorithm

- RRT* is a variant of RRG that essentially "rewires" the tree as better paths are discovered.
- After rewiring the cost has to be propagated along the leaves.
- If steering errors occur, subtrees can be re-computed.
- The RRT* algorithm inherits the asymptotic optimality and rapid exploration properties of the RRG and RRT.

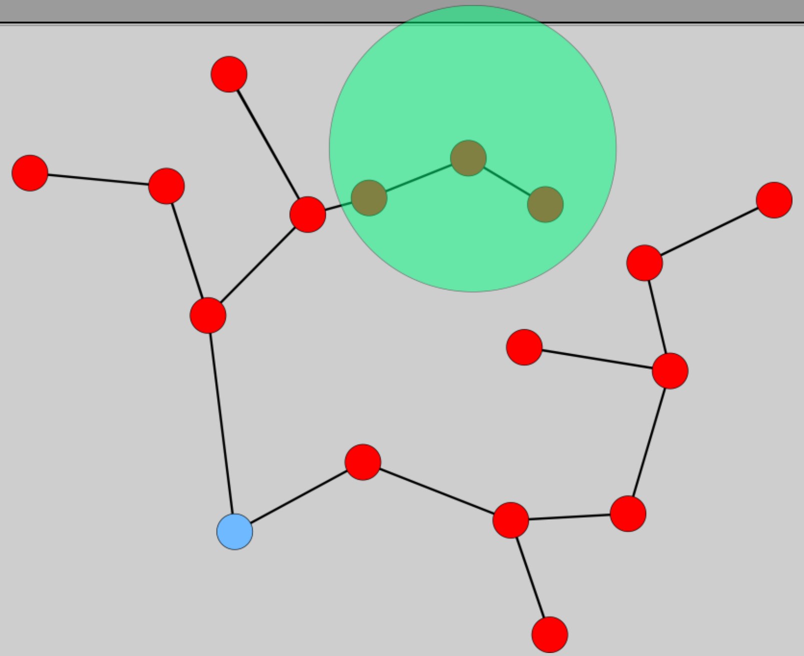


RRT*: A tree version of the RRG

- RRT algorithm can account for nonholonomic dynamics and modeling errors.
- RRG requires connecting the nodes exactly, i.e., the *Steer* procedure has to be exact. Exact steering methods are not available for general dynamic systems.

RRT* Algorithm

- RRT* is a variant of RRG that essentially "rewires" the tree as better paths are discovered.
- After rewiring the cost has to be propagated along the leaves.
- If steering errors occur, subtrees can be re-computed.
- The RRT* algorithm inherits the asymptotic optimality and rapid exploration properties of the RRG and RRT.

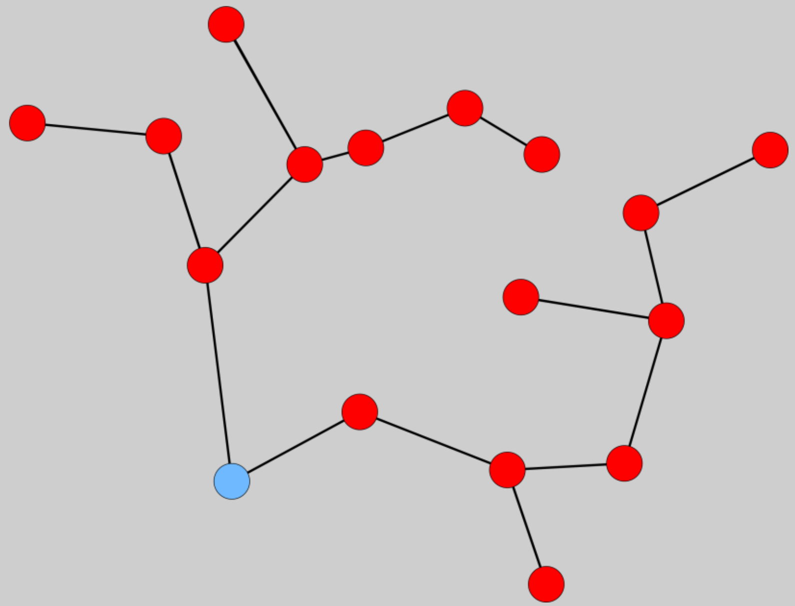


RRT*: A tree version of the RRG

- RRT algorithm can account for nonholonomic dynamics and modeling errors.
- RRG requires connecting the nodes exactly, i.e., the $Steer$ procedure has to be exact. Exact steering methods are not available for general dynamic systems.

RRT* Algorithm

- RRT* is a variant of RRG that essentially "rewires" the tree as better paths are discovered.
- After rewiring the cost has to be propagated along the leaves.
- If steering errors occur, subtrees can be re-computed.
- The RRT* algorithm inherits the asymptotic optimality and rapid exploration properties of the RRG and RRT.



Rapidly-exploring Random Tree-star (RRT*)

RRT* Algorithm

$V \leftarrow \{x_{init}\}; E \leftarrow 0;$

for i=1,…,N **do**:

$x_{rand} \leftarrow SampleFree;$

$x_{nearest} \leftarrow Nearest(G = (V, E), x_{rand});$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand});$

if $ObstacleFree(x_{nearest}, x_{new})$ **then**:

$X_{near} \leftarrow Near\left(G = (V, E), x_{new}, \min\{\gamma_{RRG}\left(\frac{\log(card\ V)}{card\ V}\right)^{1/d}, \eta\}\right);$

$V \leftarrow V \cup \{x_{new}\};$

$x_{min} \leftarrow x_{nearest};\ c_{\min} \leftarrow Cost(x_{nearest}) + c(Line(x_{nearest}, x_{new}))$

foreach $x_{near} \in X_{near}$ **do:**

if $CollisionFree(x_{near}, x_{new}) \wedge Cost(x_{near}) + c\big(Line(x_{near}, x_{new})\big) < Cost(x_{near})$ **then:**

$x_{parent} \leftarrow Parent(x_{near});$

$E \leftarrow E \backslash \{(x_{parent}, x_{near})\} \cup \{(x_{new}, x_{near})\};$

return $G = (V, E);$

Summary

- **Key idea in RRG/RRT*:** to combine optimality and computational efficiency, it is necessary to attempt connection to $\Theta(\log N)$ nodes at each iteration.
  - Reduce volume of the "connection ball" as $\log(N)/N$;
  - Increase the number of connections as $\log N$
- These principles can be used to obtain "optimal" versions of PRM etc.

Algorithm	Probabilistic Completeness	Asymptotic Optimality	Computational Complexity
sPRM	YES	YES	$O(N)$
k-nearest PRM	NO	NO	$O(logN)$
RRT	YES	NO	$O(logN)$
PRM*	YES	YES	$O(logN)$
k-nearest PRM*	YES	YES	$O(logN)$
RRG	YES	YES	$O(logN)$
k-nearest RRG	YES	YES	$O(logN)$
RRT*	YES	YES	$O(logN)$
k-nearest RRG	YES	YES	$O(logN)$

The inspection path planning problem

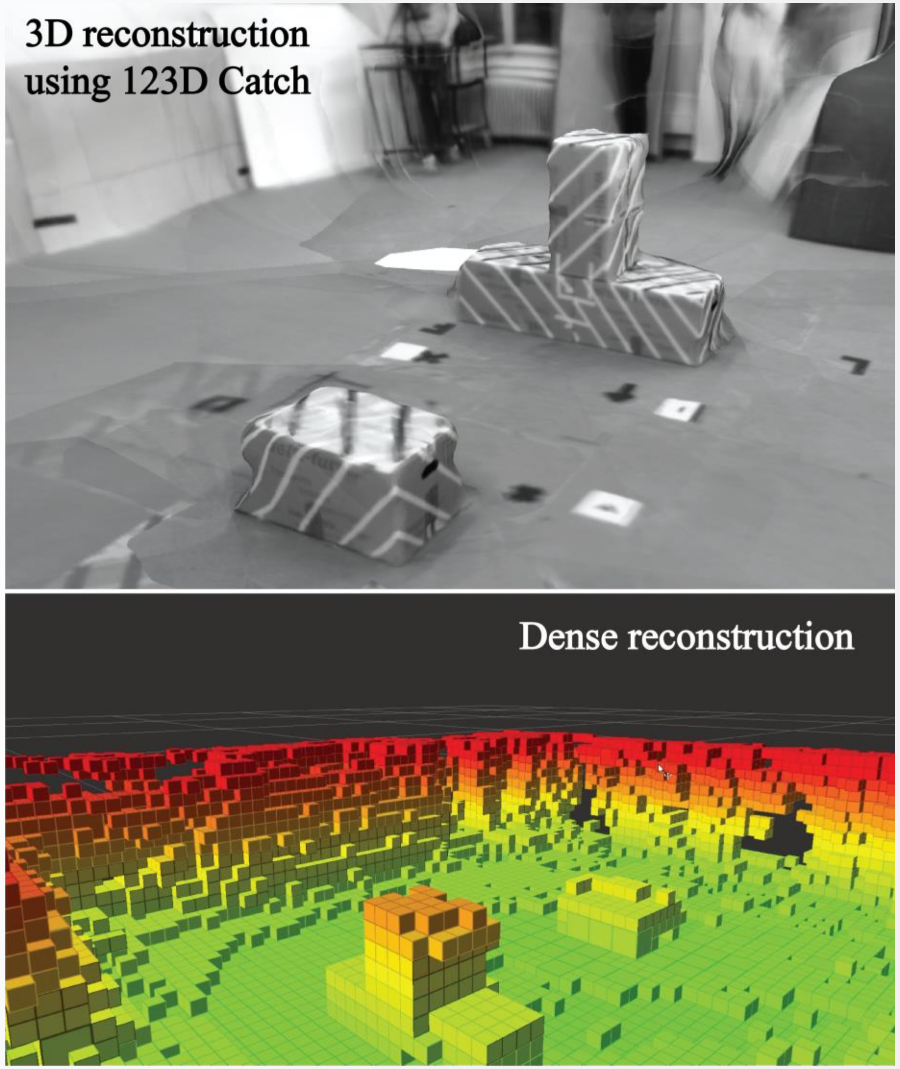
- Consider a dynamical control system defined by an ODE of the form:

$$\frac{dx}{dt} = f(x,u), x(0) = x_{init}$$

- Where is x the state, u is the control. As well as a sensor model of field of view $FOV = [F_H, F_V]$ and maximum range d.
- Given an obstacle set X_{obs}, and a inspection manifold S_I, the objective of the motion planning problem is to find, if it exists, a path $\boldsymbol{r}$ that provides the viewpoints to the sensor such that the whole surface of S_I is perceived, the vehicle dynamics are respected and the cost of the path (distance, time, etc) is minimized.

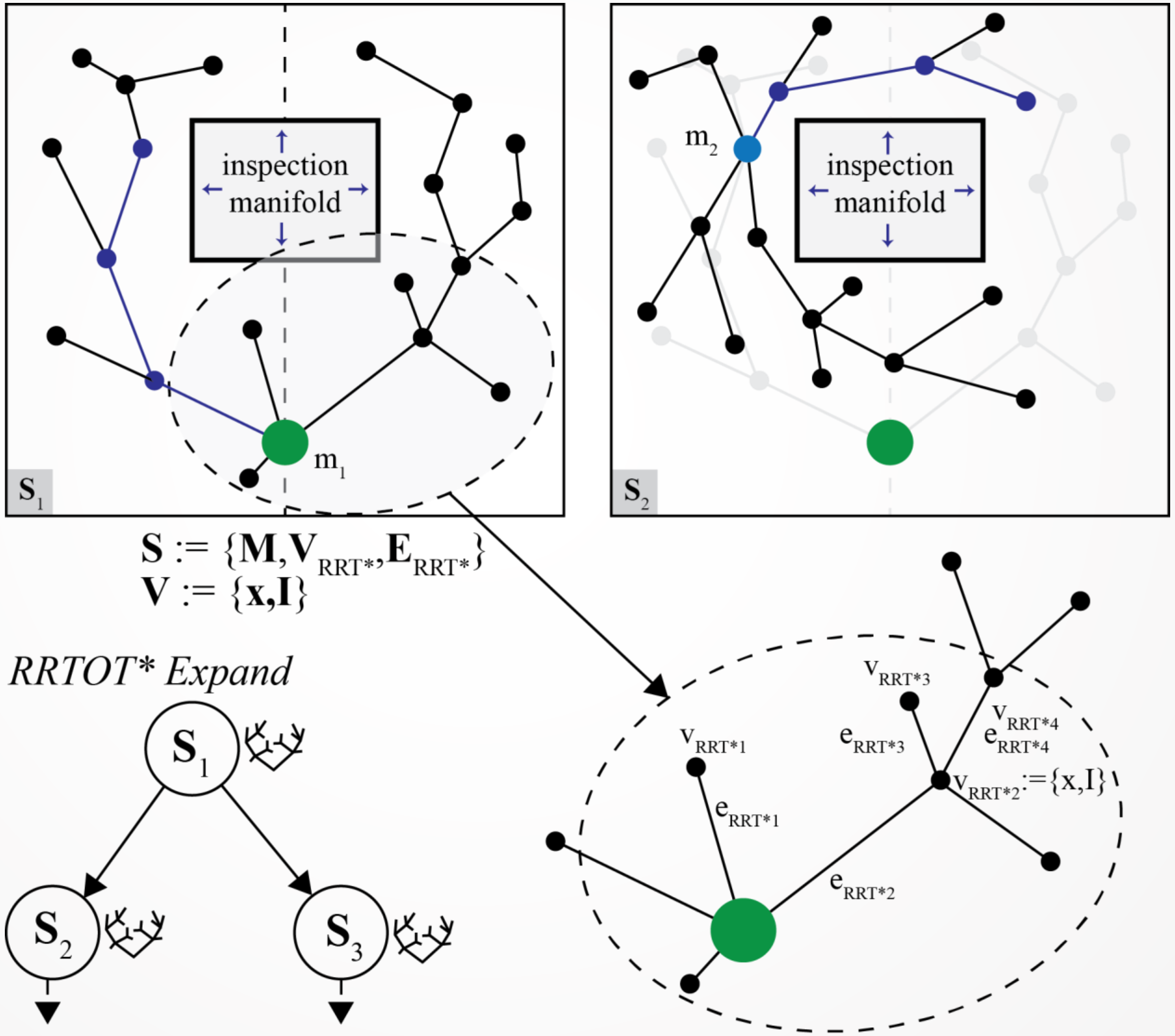
Rapidly-exploring Random Tree-Of-Trees (RRTOT)

- **Problem:** given a representation of the structure find the optimal coverage path.
- **Challenges:** can we find the optimal path? Can we converge asymptotically to that solution?
- **Goal:** Provide an algorithm that can incrementally derive the optimal solution and be able to provide admissible paths "anytime".



RRTOT: Functional Principle

Overcome the limitations of motion planners designed for navigation problems.

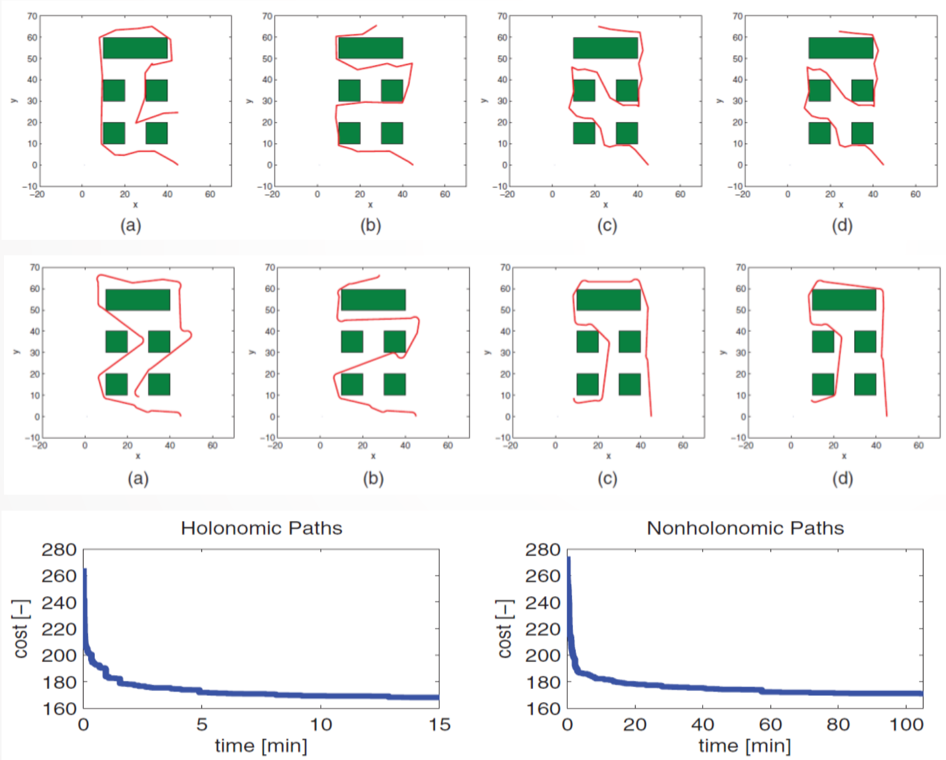


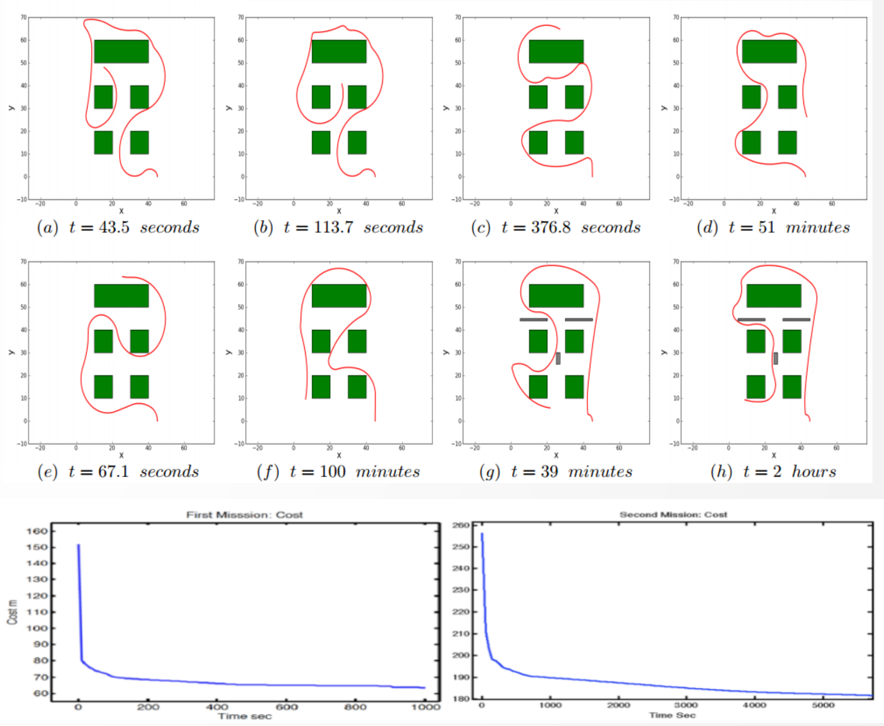
Vary the solution topology – be able to find the optimal solution. X`

Overcome the limitations of SIP but in a computationally very expensive way.

RRTOT: Functional Principle

- **Comparison with the state-of-the-art:** RRTOT seems to be able to provide solutions faster.

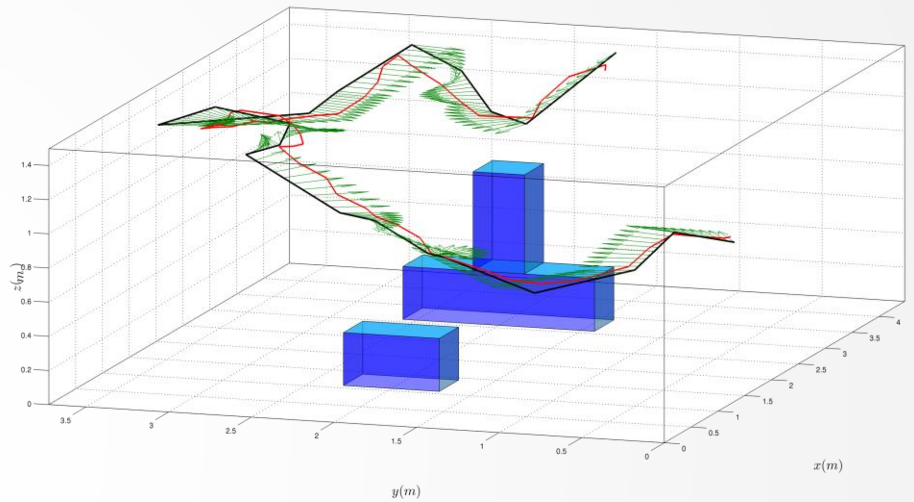




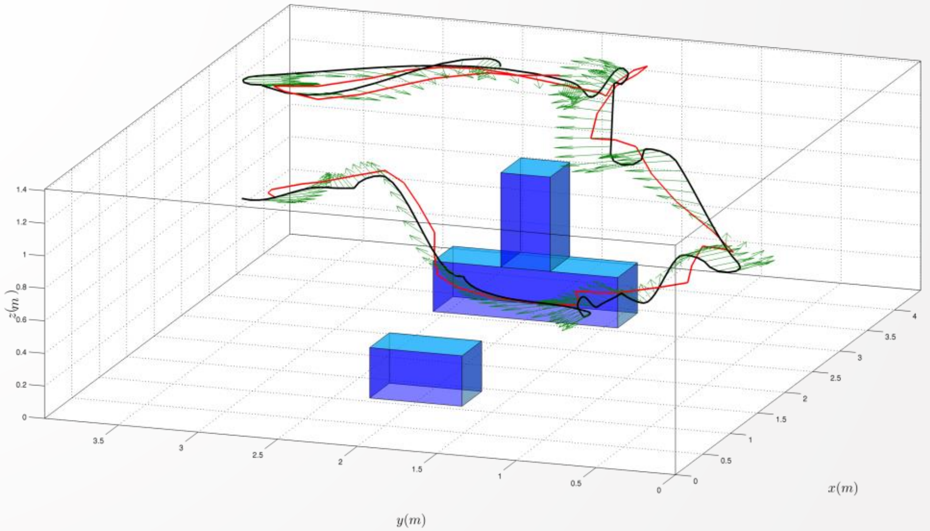
- **Comparison against:** G Papadopoulos, H Kurniawati, N Patrikalakis, “Asymptotically optimal path planning and surface reconstruction for inspection”, IEEE International Conference on Robotics and Automation (ICRA) 2013.

RRTOT: Indicative Solutions

- **Holonomic**

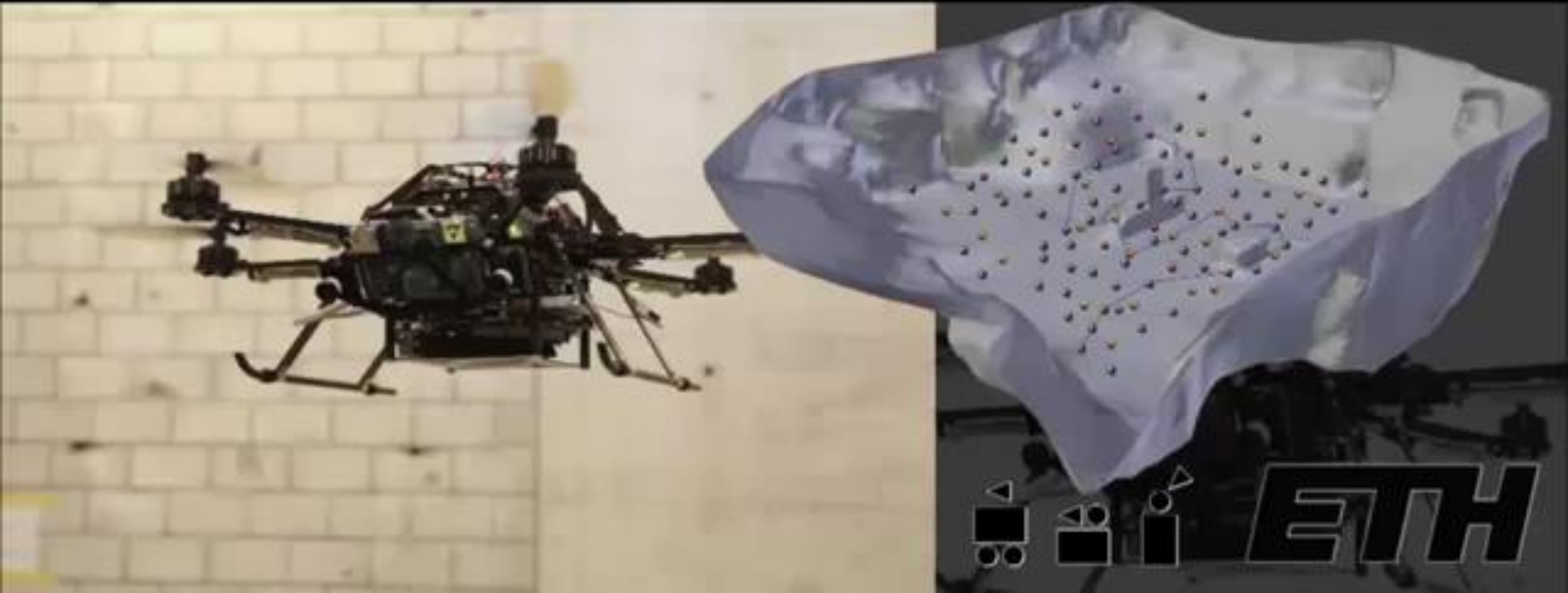


- **Nonholonomic**



An Incremental Sampling-based approach to Inspection Planning: the Rapidly-exploring Random Tree Of Trees

Andreas Bircher, Kostas Alexis, Ulrich Schwesinger, Sammy Omari, Michael Burri and Roland Siegwart



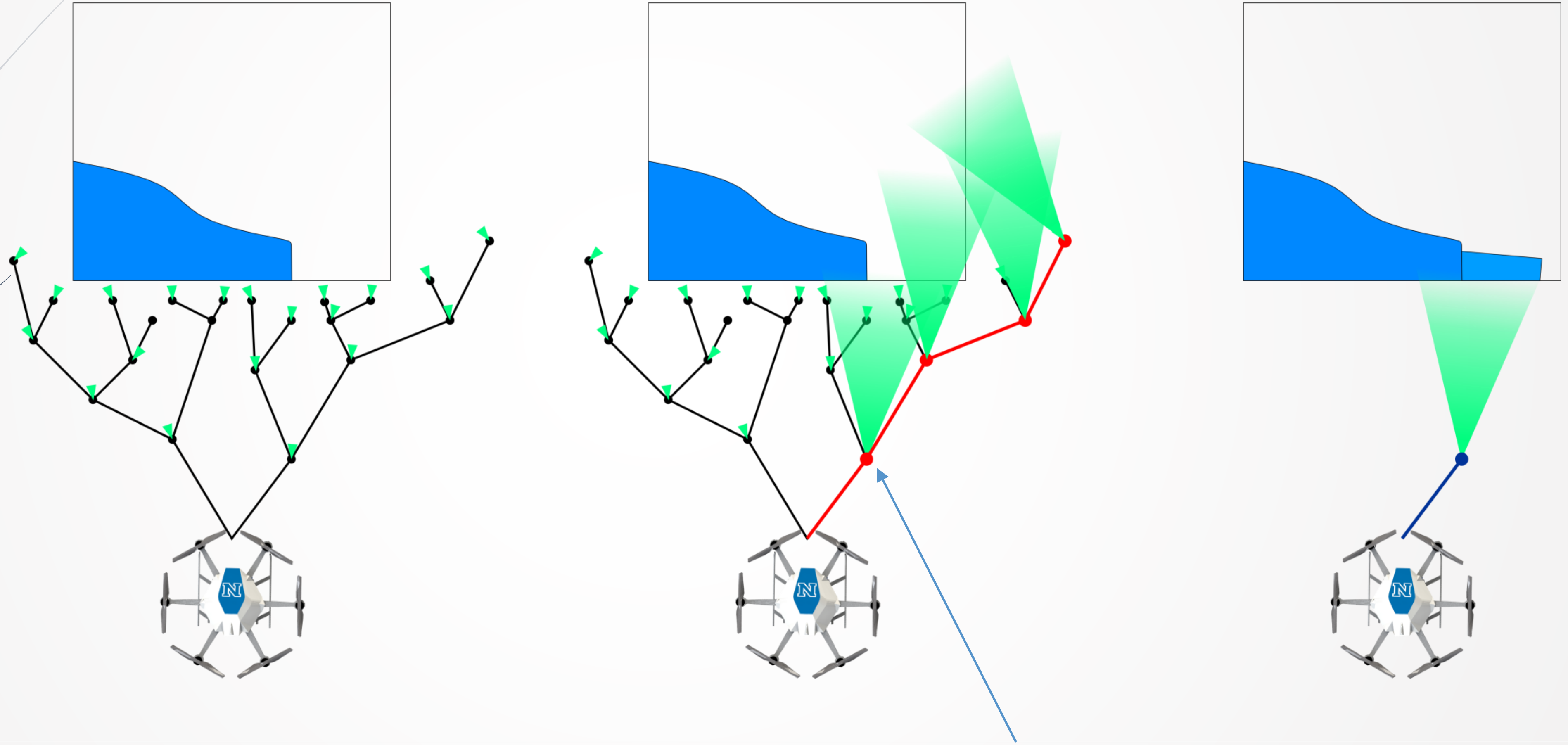
The Exploration path planning problem

Problem Definition

The exploration path planning problem consists in exploring a bounded 3D space $V \subset \mathbb{R}^3$. This is to determine which parts of the initially unmapped space $V_{unm} = V$ are free $V_{free} \subset V$ or occupied $V_{occ} \subset V$. The operation is subject to vehicle kinematic and dynamic constraints, localization uncertainty and limitations of the employed sensor system with which the space is explored.

- As for most sensors the perception stops at surfaces, hollow spaces or narrow pockets can sometimes not be explored with a given setup. This residual space is denoted as V_{res}. The problem is considered to be fully solved when $V_{free} \cup V_{occ} = V \backslash V_{res}$.
- Due to the nature of the problem, a suitable path has to be computed online and in real-time, as free space to navigate is not known prior to its exploration.

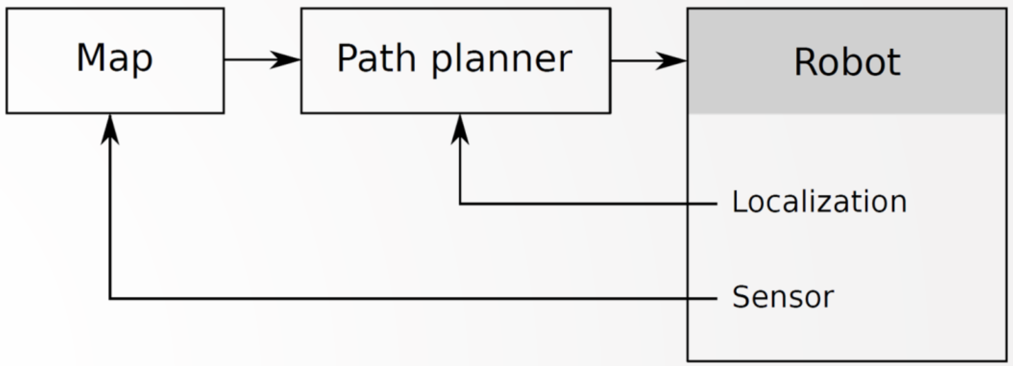
RH-NBVP Functional Principle



$$\mathbf{Gain}(n_k) = \mathbf{Gain}(n_{k-1}) + \mathbf{Visible}(\mathcal{M}, \xi_k) e^{-\lambda c(\sigma_{k-1}^{k})}$$

RH-NBVP Approach

- **Environment representation:** Occupancy Map dividing space V into $m \in M$ cubical volumes (voxels) that can be marked either as free, occupied or unmapped.
- Array of voxels is saved in an octree structure to enable computationally efficient access and search.
- Paths are planned only within the free space V_{free} and collision-free point-to-point navigation is inherently supported.
- At each viewpoint/configuration of the environment ξ, the amount of space that is visible is computed as $Visible(M, \xi)$



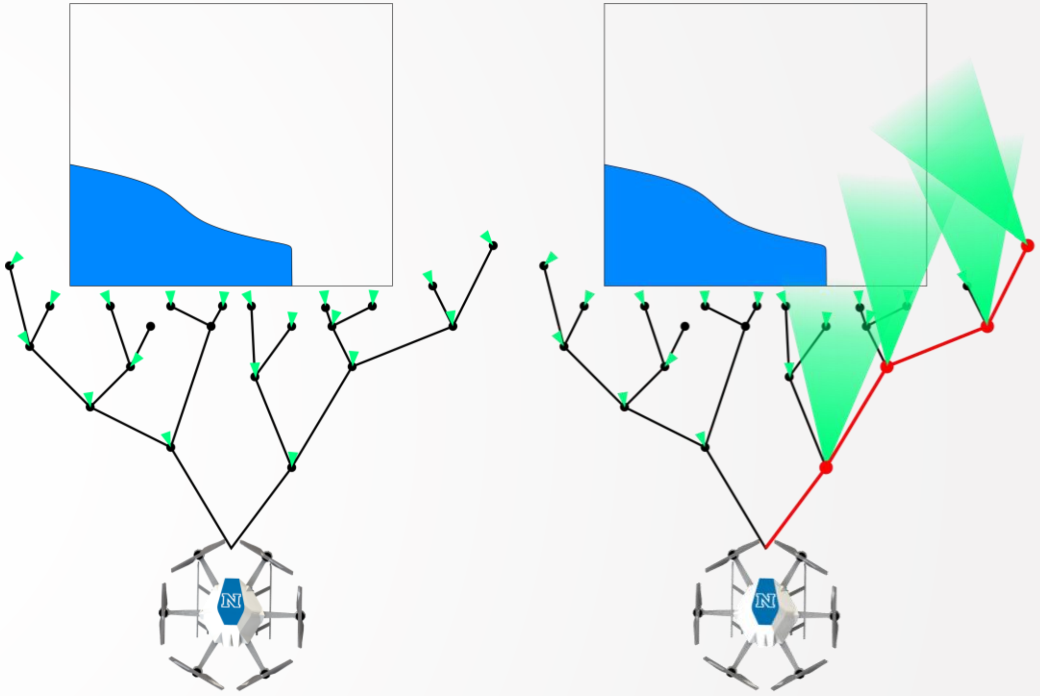
The Receding Horizon Next-Best-View Exploration Planner relies on the real-time update of the 3D map of the environment.

RH-NBVP Approach

- **Tree-based exploration:** At every iteration, RH-NBVP spans a random tree of finite depth. Each vertex of the tree is annotated regarding the collected Information Gain – a metric of how much new space is going to be explored.

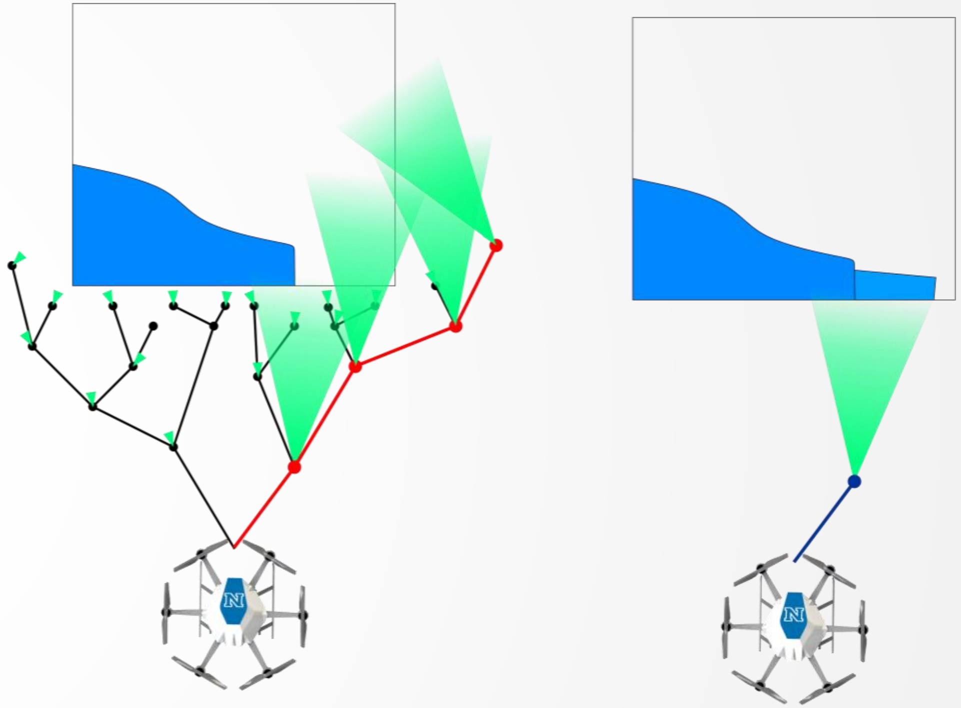
$$\mathbf{Gain}(n_k) = \mathbf{Gain}(n_{k-1}) + \mathbf{Visible}(\mathcal{M}, \xi_k) e^{-\lambda c(\sigma_{k-1}^k)}$$

- Within the sampled tree, evaluation regarding the path that overall leads to the highest information gain is conducted. This corresponds to the **best path** for the given iteration. It is a sequence of next-best-views as sampled based on the vertices of the spanned random tree.



RH-NBVP Approach

- **Receding Horizon:** For the extracted best path of viewpoints, only the first viewpoint is actually executed.
- The system moves to the first viewpoint of the path of best viewpoints.
- Subsequently, the whole process is repeated within the next iteration. This gives rise to a receding horizon operation.



RH-NBVP Algorithm

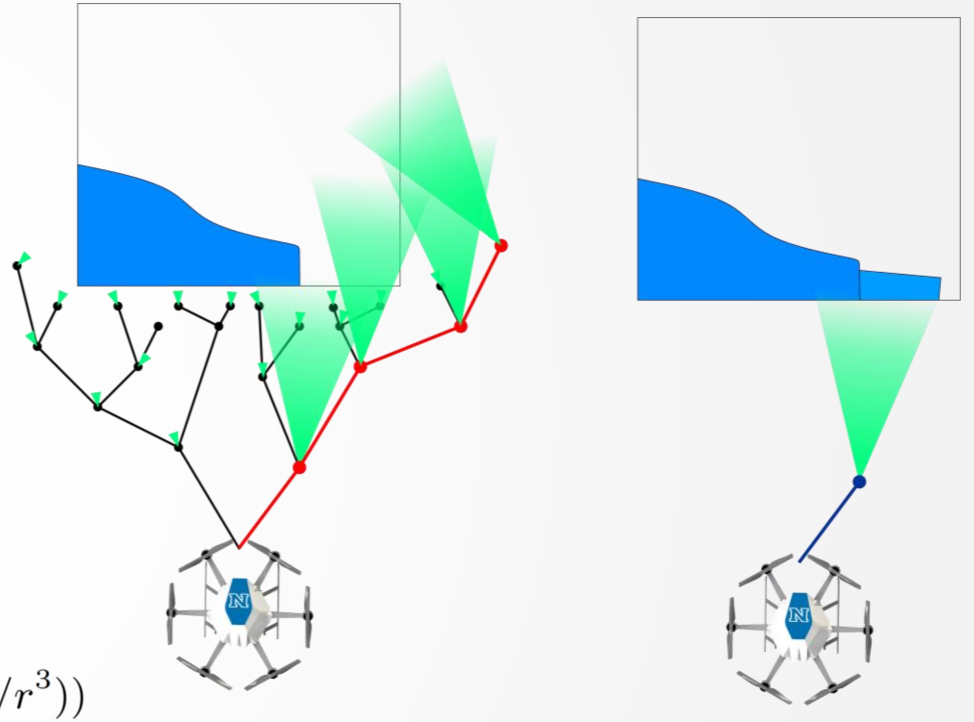
NBVP Iterative Step

- ξ_0 ←current vehicle configuration
- Initialize $\boldsymbol{T}$ with ξ_0 and, unless first planner call, also previous best branch
- $g_{best} \leftarrow 0$ // Set best gain to zero
- $n_{best} \leftarrow n_0(\xi_0)$ // Set best node to root
- N_T ←Number of nodes in $\boldsymbol{T}$
- **while** $N_T < N_{max}$ or $g_{best} == 0$ **do**
  - Incrementally build T by adding $n_{new}(\xi_{new})$
  - $N_T \leftarrow N_T + 1$
  - **if** $Gain(n_{new}) > g_{best}$ **then**
    - $n_{best} \leftarrow n_{new}$
    - $g_{best} \leftarrow Gain(n_{new})$
  - **if** $N_T > N_{TOT}$ **then**
    - Terminate exploration
- $\sigma \leftarrow \boldsymbol{ExtractBestPathSegment}(n_{best})$
- Delete $\boldsymbol{T}$
- **return** σ

RH-NBVP Remarks

- **Inherently Collision-free:** As all paths of NBVP are selected along branches within RRT-based spanned trees, all paths are inherently collision-free.
- **Computational Cost:** NBVP has a thin structure and most of the computational cost is related with collision-checking functionalities. The formula that expresses the complexity of the algorithm takes the form:

$$\mathcal{O}(N_{\mathbb{T}} \log(N_{\mathbb{T}}) + N_{\mathbb{T}}/r^3 \log(V/r^3) + N_{\mathbb{T}}(d_{\max}^{\text{planner}}/r)^4 \log(V/r^3))$$



Find out more

- http://www.kostasalexis.com/autonomous-navigation-and-exploration.html
- http://www.kostasalexis.com/holonomic-vehicle-bvs.html
- http://www.kostasalexis.com/dubins-airplane.html
- http://www.kostasalexis.com/collision-free-navigation.html
- http://www.kostasalexis.com/structural-inspection-path-planning.html

- http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-410-principles-of-autonomy-and-decision-making-fall-2010/lecture-notes/
- http://ompl.kavrakilab.org/
- http://moveit.ros.org/
- http://planning.cs.uiuc.edu/

References

- A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, R. Siegwart, "Structural Inspection Path Planning via Iterative Viewpoint Resampling with Application to Aerial Robotics", IEEE International Conference on Robotics & Automation, May 26-30, 2015 (ICRA 2015), Seattle, Washington, USA
- Kostas Alexis, Christos Papachristos, Roland Siegwart, Anthony Tzes, "Uniform Coverage Structural Inspection Path-Planning for Micro Aerial Vehicles", Multiconference on Systems and Control (MSC), 2015, Novotel Sydney Manly Pacific, Sydney Australia. 21-23 September, 2015
- K. Alexis, G. Darivianakis, M. Burri, and R. Siegwart, "Aerial robotic contact-based inspection: planning and control", Autonomous Robots, Springer US, DOI: 10.1007/s10514-015-9485-5, ISSN: 0929-5593, http://dx.doi.org/10.1007/s10514-015-9485-5
- A. Bircher, K. Alexis, U. Schwesinger, S. Omari, M. Burri and R. Siegwart "An Incremental Sampling–based approach to Inspection Planning: the Rapidly–exploring Random Tree Of Trees", accepted at the Robotica Journal (awaiting publication)
- A. Bircher, M. Kamel, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, R. Siegwart, "Three-dimensional Coverage Path Planning via Viewpoint Resampling and Tour Optimization for Aerial Robots", Autonomous Robots, Springer US, DOI: 10.1007/s10514-015-9517-1, ISSN: 1573-7527
- A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, R. Siegwart, "Receding Horizon "Next-Best-View" Planner for 3D Exploration", IEEE International Conference on Robotics and Automation 2016 (ICRA 2016), Stockholm, Sweden (Accepted - to be presented)

Thank you!

Please ask your question!