

Sampling-based Planning and Control

Lecture 12

Principles of Safe Autonomy ECE498SMA

Sayan Mitra

Based on lectures by Emilio Frazzoli



Motion planning problem

- Get from point A to point B avoiding obstacles
- Last lecture we saw how search for collision free trajectories can be converted to graph search using hybrid A*
- Earlier we saw how vehicle motion can be modeled by differential equations
- Now we will study the motion planning/search problem can incorporate additional dynamical constraints

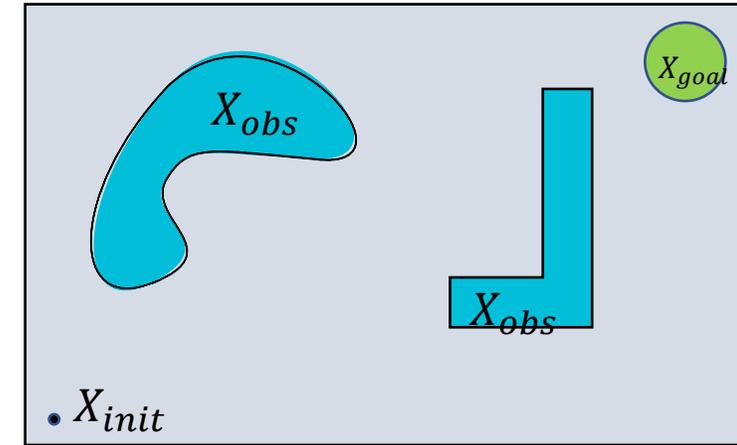


Motion planning, specifically trajectory planning problem

Consider a dynamical control system defined by an ODE of the form $\frac{dx}{dt} = f(x, u)$, $x(0) = x_{init}$ (1)

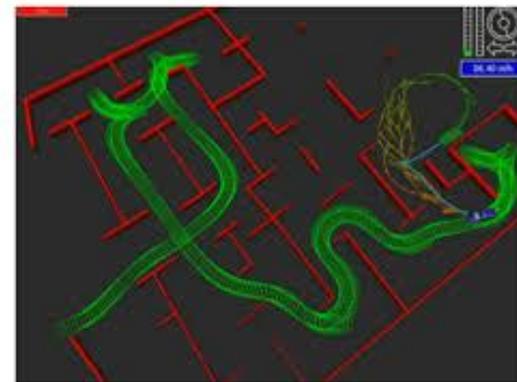
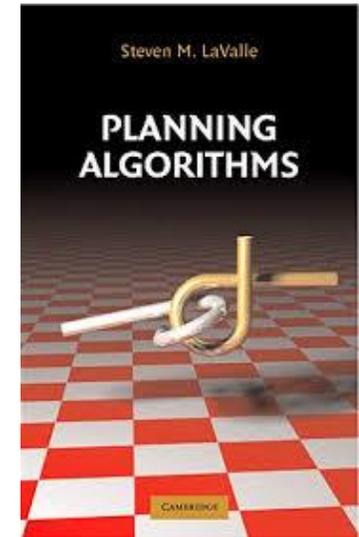
where x is the state (or a point in the configuration space), u is the control. Given an obstacle set $X_{obst} \subset R^d$, and a goal set $X_{goal} \subset R^d$, the objective of the motion planning problem is to find, if it exists, a control signal u such that the solution of (1) satisfies

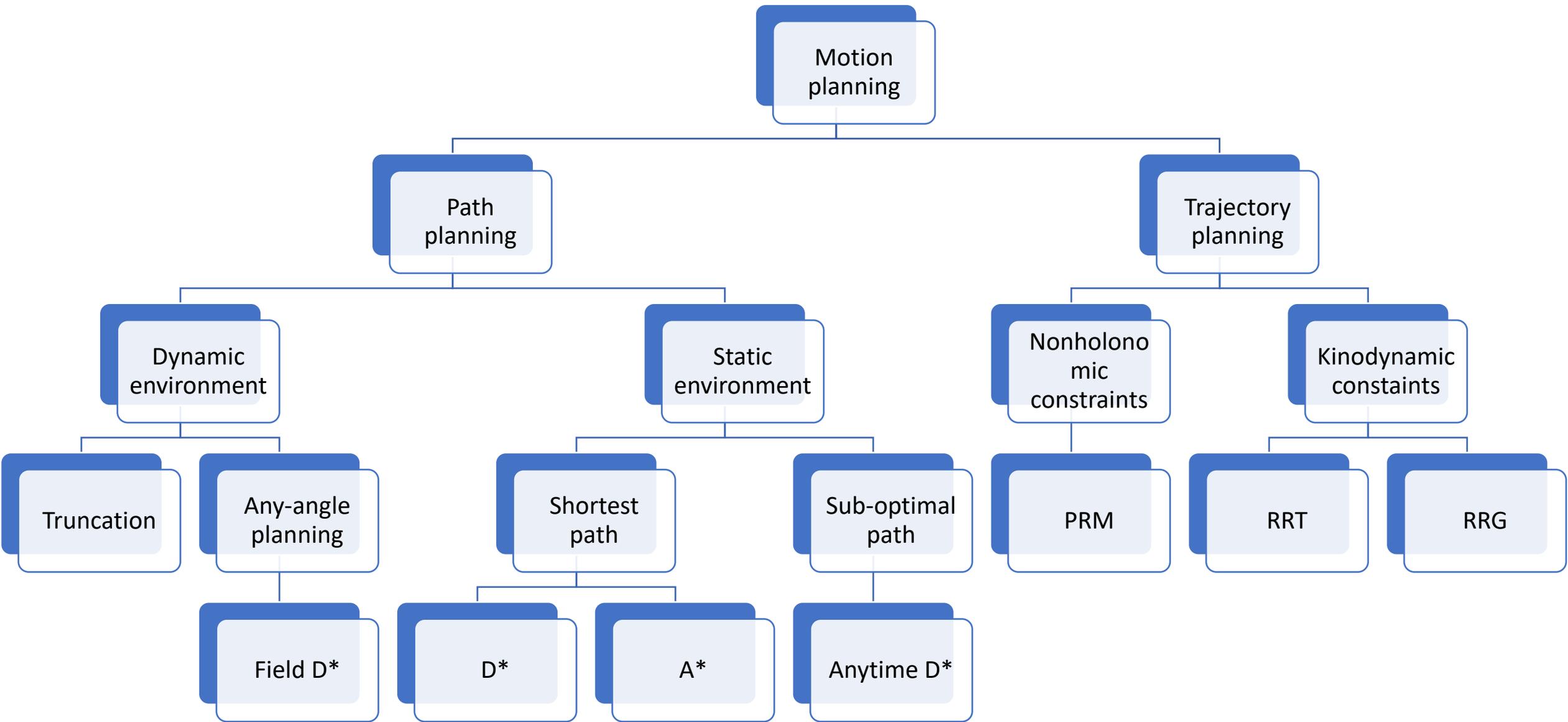
- for all $t \in R_{\geq 0}$, $x(t) \notin X_{obs}$, and
- for some finite $T \geq 0$, for all $t > T$, $x(t) \in X_{goal}$, 0
- Return failure if no such control signal exists.



Canonical problem

- Basic problem in robotics
 - Autonomous vehicles
 - Puzzles
- Provably very hard: a basic version (the Generalized Piano Mover's problem) is known to be PSPACE-hard [Reif, '79].
- The Sea Squirt, or Tunicate, is an organism capable of mobility until it finds a suitable rock to cement itself in place. Once it becomes stationary, it digests its own cerebral ganglion, or "eats its own brain" and develops a thick covering, a "tunic" for self defense. [S. Soatto, 2010, R. Bajcsy, 1988] <http://en.wikipedia.org/wiki/File:Sea-tulip.jpg>.





Types of planners

- **Discretization + graph search:** Analytic/grid-based methods do not scale well to high dimensions.
 - A^* , D^* , etc. can be sensitive to graph size. [Resolution complete](#).
- **Algebraic planners:** Explicit representation of obstacles.
 - Use algebra (visibility computations/projections) to find the path. Complete, but often impractical.
- **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.



Sampling-based algorithms

Solutions are computed based on samples from some distribution.

Retain some form of completeness, e.g., probabilistic completeness

Incremental sampling methods

- Lend themselves to real-time, on-line implementations
- Can work with very general dynamics
- Do not require explicit constraints



Outline

- Probabilistic Roadmaps
- Rapidly expanding random trees (RRT)
- RRG



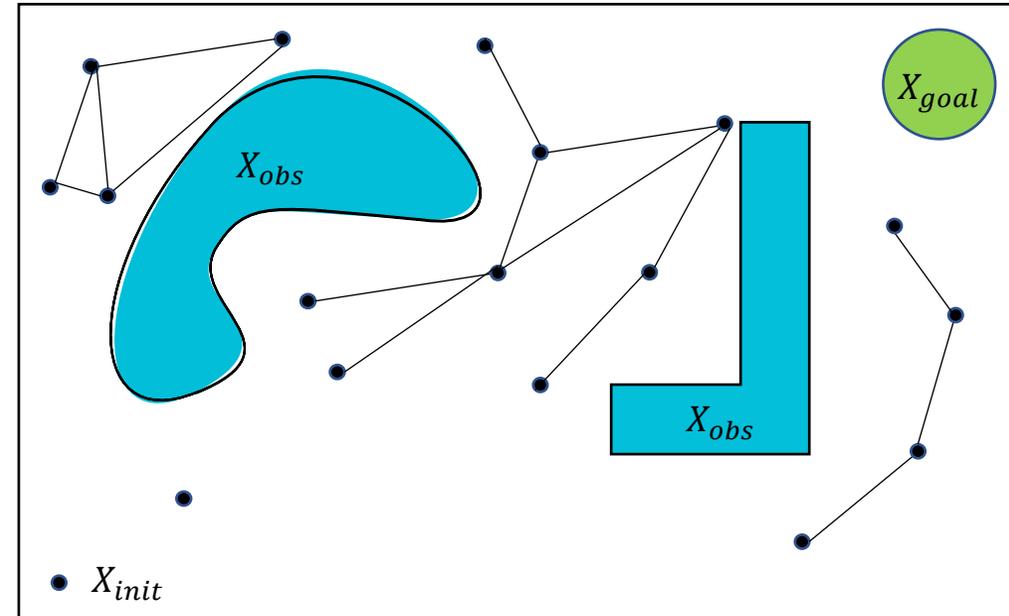
Probabilistic RoadMaps (PRM)

- 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 figure out paths quickly at run time.
- Learning/pre-processing phase:
 - Sample n points from $X_{free} = [0, 1]^d \setminus X_{obs}$
 - Try to connect these points using a fast “local planner” (e.g., ignore obstacles).
 - If connection is 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 general planning problems in > 4 -5 dimensions!



Probabilistic RoadMap

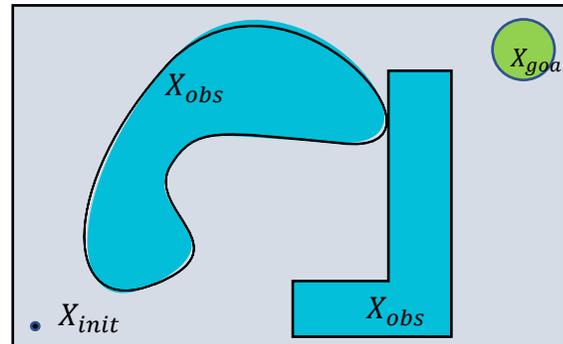
- Connect points within a radius r , starting from “closest” ones
- Do not attempt to connect points already on the same connected component of PRM
- What properties does this algorithm have?
 - Will it find a solution if one exists?
 - Is this an optimal solution?
 - What is the complexity?



Probabilistic completeness

Definition. A motion planning problem $P = (X_{free}, x_{init}, X_{goal})$ is **robustly feasible** if there exists some small $\delta > 0$ such that a solution remains a solution if obstacles are “dilated” by δ .

Fig. not robust.



Definition. An algorithm ALG is **probabilistically complete** if, for any **robustly feasible** motion planning problem defined by $P = (X_{free}, x_{init}, X_{goal})$, $\lim_{N \rightarrow \infty} \Pr(\text{ALG returns a solution to } P) = 1$.

- Applicable to motion planning problems with a robust solution.



Asymptotic optimality

Suppose we have a cost function c that associates to each path σ a non-negative cost $c(\sigma)$, e.g., $c(\sigma) = \int_{\sigma} \chi(s) ds$.

Definition. An algorithm ALG is *asymptotically optimal* if, for any motion planning problem $P = (X_{free}, x_{init}, X_{goal})$ and cost function c that admit a robust optimal solution with finite cost c^* ,

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



Simple PRM construction: Step 1

PRM(N,k)

$V \leftarrow \{x_{\text{init}}\} \cup \{\text{SampleFree}_i\}_{i=1,\dots,N-1}$

$E \leftarrow \emptyset$

foreach $v \in V$ **do**

$U \leftarrow \text{Near}(V, v, k)$

// selects the k nearest neighbors of v in V

foreach $u \in U$ **do**

if $\text{CollisionFree}(v, u)$ **then**

$E \leftarrow E \cup \{(v, u), (u, v)\}$

return $G = (V, E)$



PRM planning: Step 2

```
PRM( $\mathbf{v}_{init}$ ,  $\mathbf{v}_{goal}$ ,  $\mathbf{G}$ )
```

```
// input may not be in roadmap
```

```
Find  $k$  nearest neighbors of  $\mathbf{v}_{init}$ ,  $\mathbf{v}_{goal}$  in  $\mathbf{G}$ 
```

```
foreach  $v \in V$  do
```

```
     $U \leftarrow \text{Near}(V, v, k)$ 
```

```
    // selects the  $k$  nearest neighbors of  $v$  in  $V$ 
```

```
    foreach  $u \in U$  do
```

```
        if CollisionFree( $v, u$ ) then
```

```
            // Check if the st line path is collision free
```

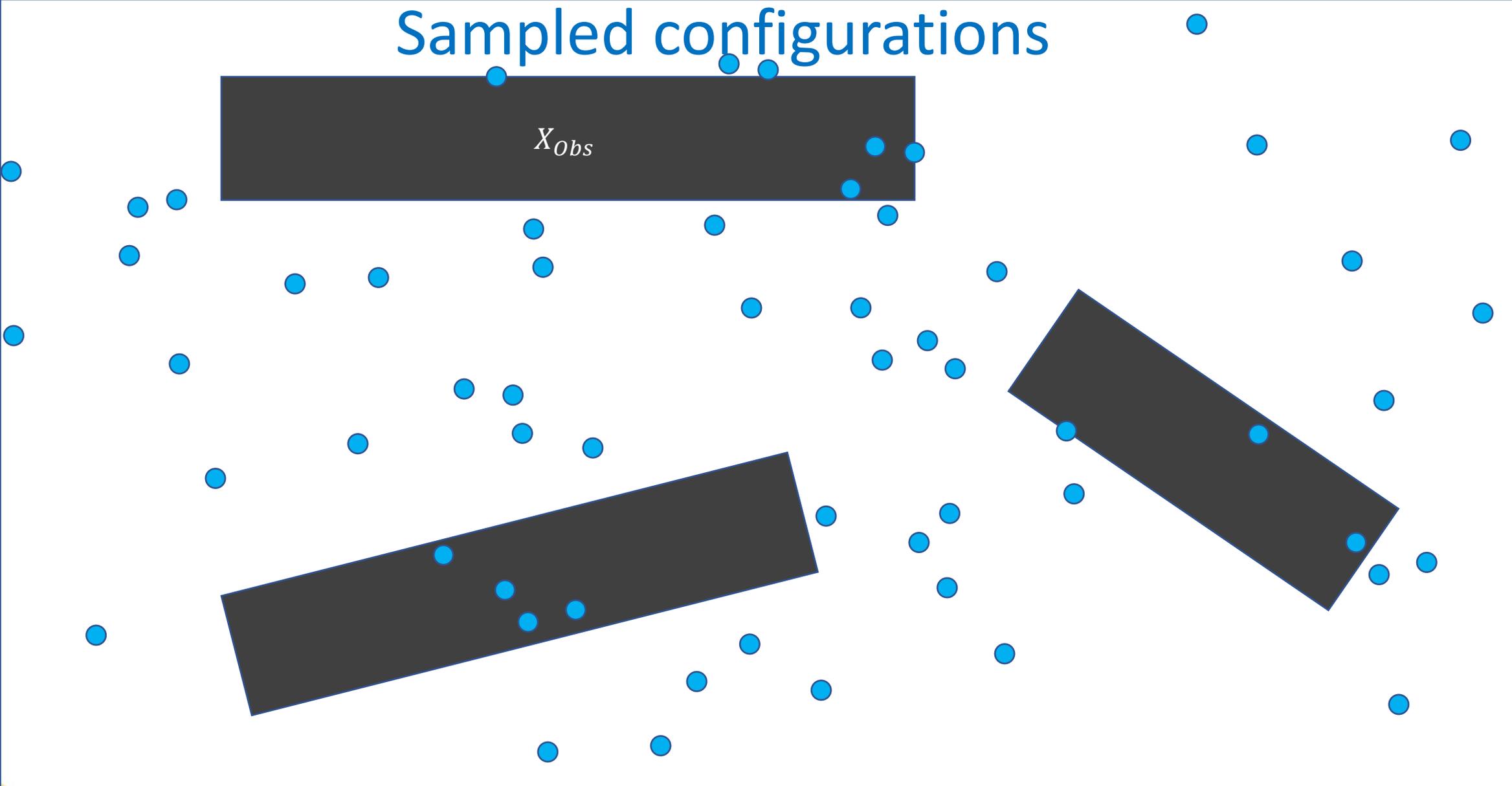
```
             $E \leftarrow E \cup \{(v, u), (u, v)\}$ 
```

```
return  $G = (V, E)$ 
```



Sampled configurations

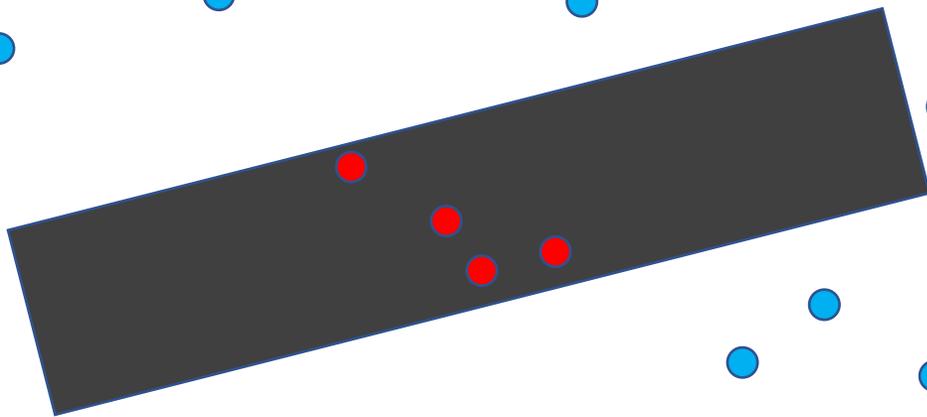
X_{Obs}



The figure displays a set of sampled configurations, represented by blue dots, scattered across a white rectangular area. Three dark gray rectangular regions are overlaid on the plot. The top-most region is labeled X_{Obs} . The other two regions are tilted and positioned in the lower-left and lower-right areas of the plot. The dots are distributed throughout the white area, with some falling within the gray regions.

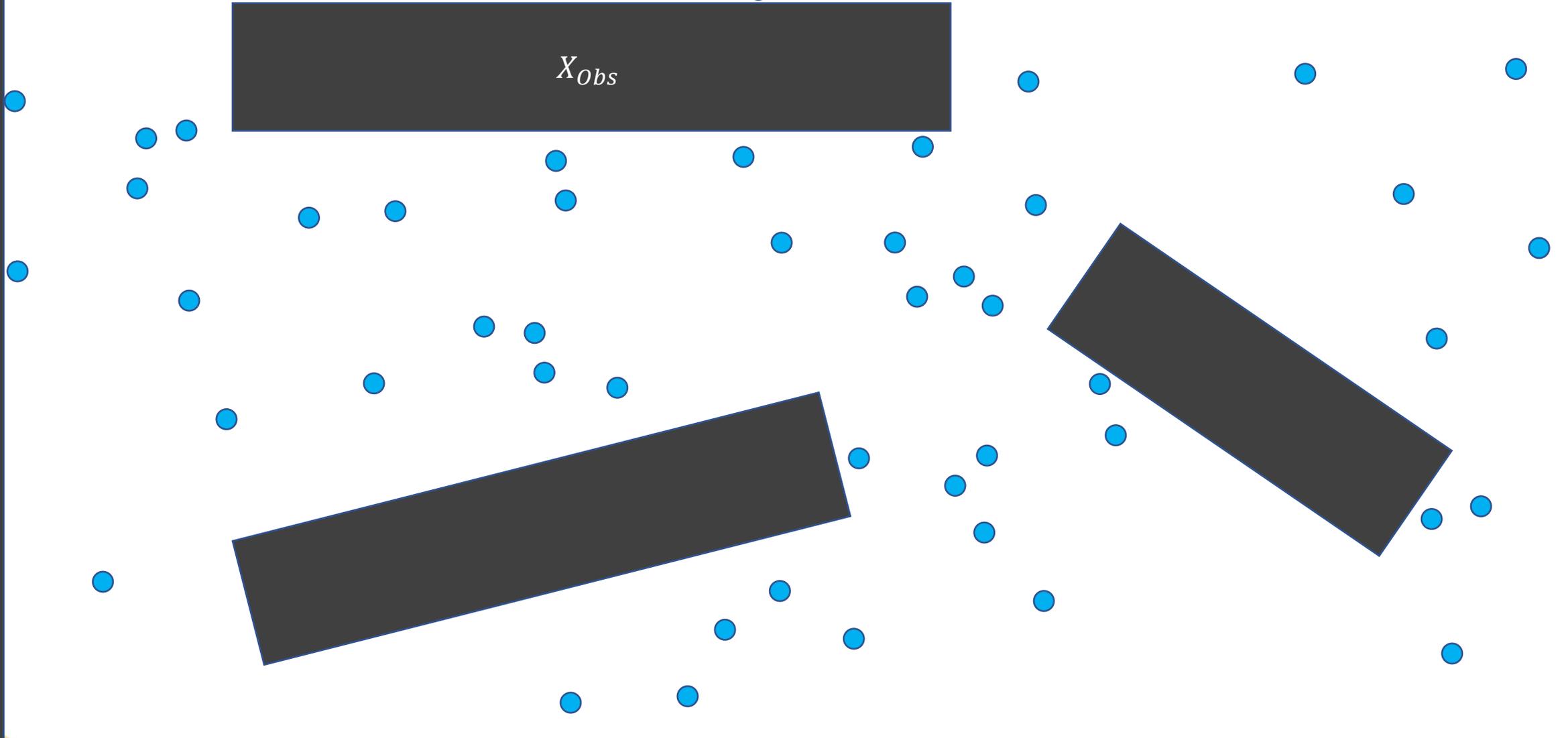
Sampled configurations tested for collisions

X_{Obs}



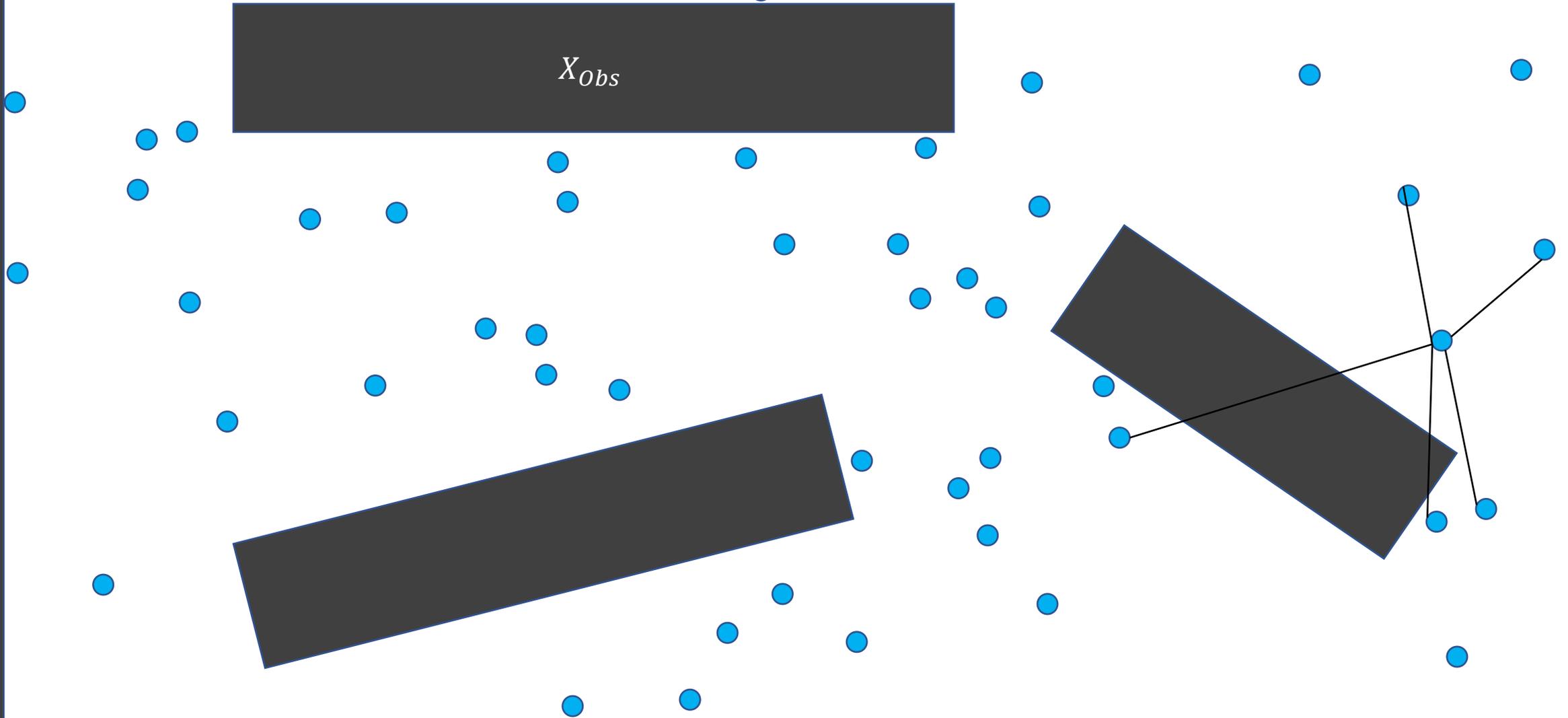
Collision-free configurations retained as milestones

X_{Obs}



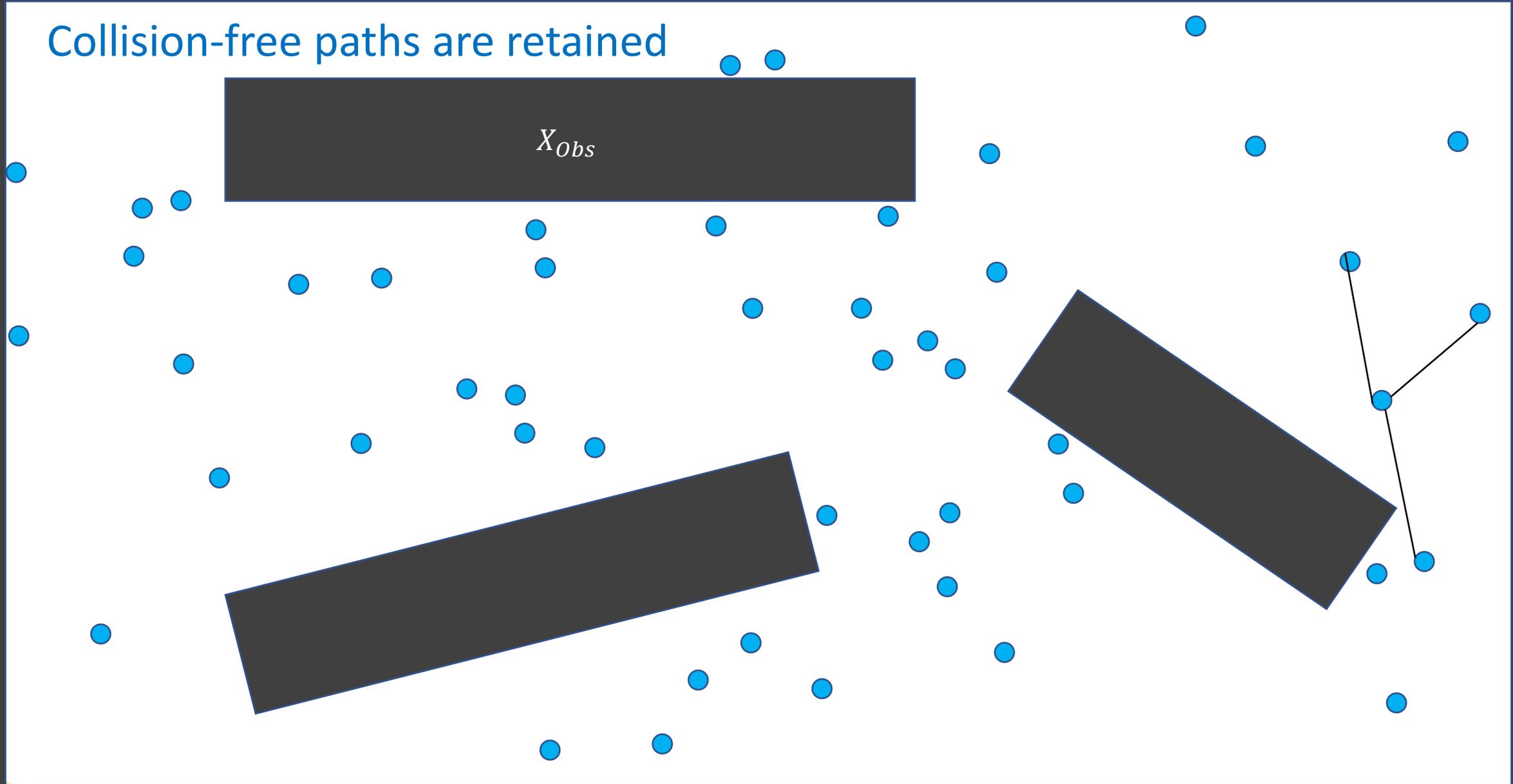
Milestones connected to k nearest neighbors by straight line paths

X_{Obs}



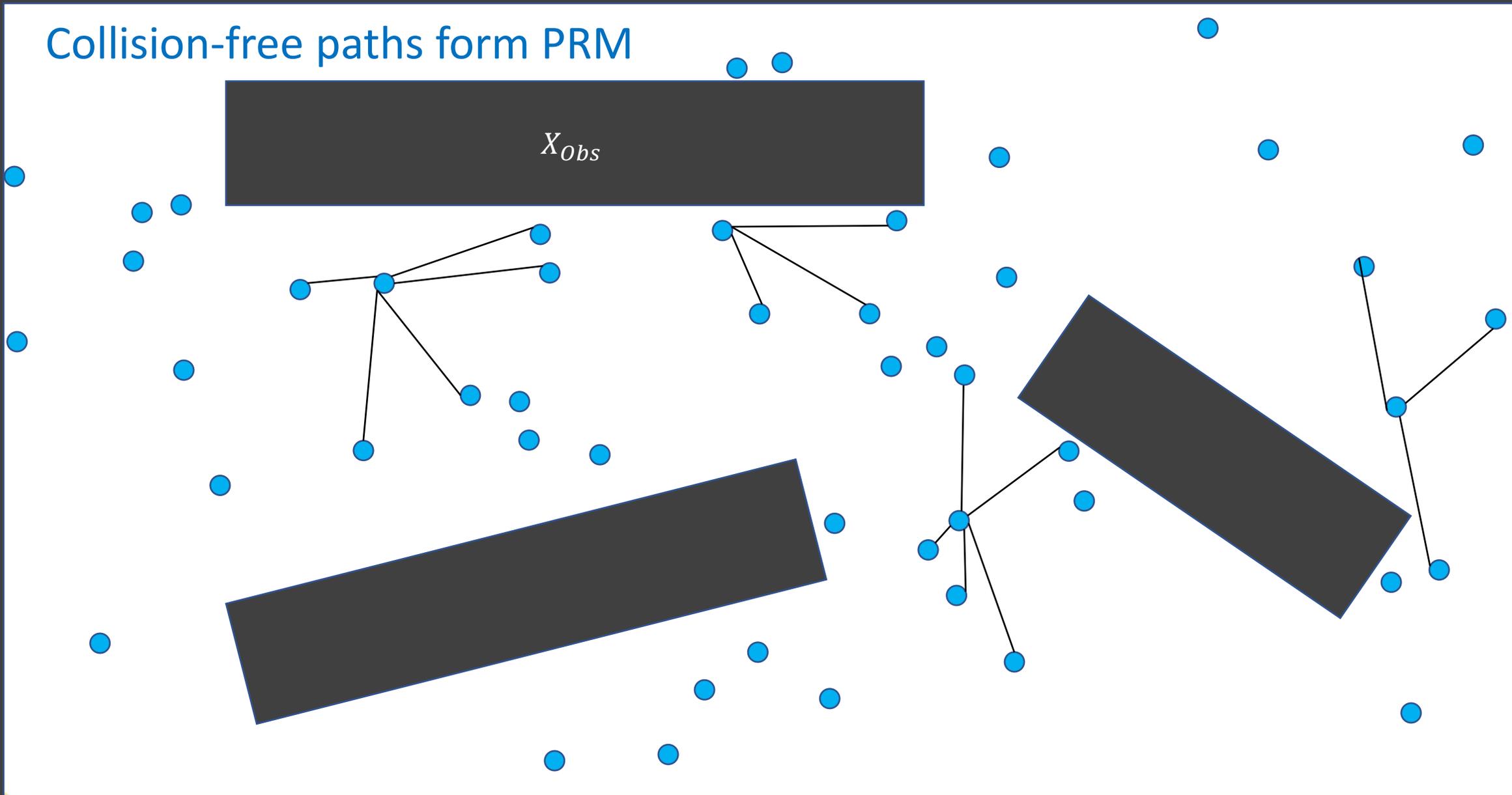
Collision-free paths are retained

X_{Obs}



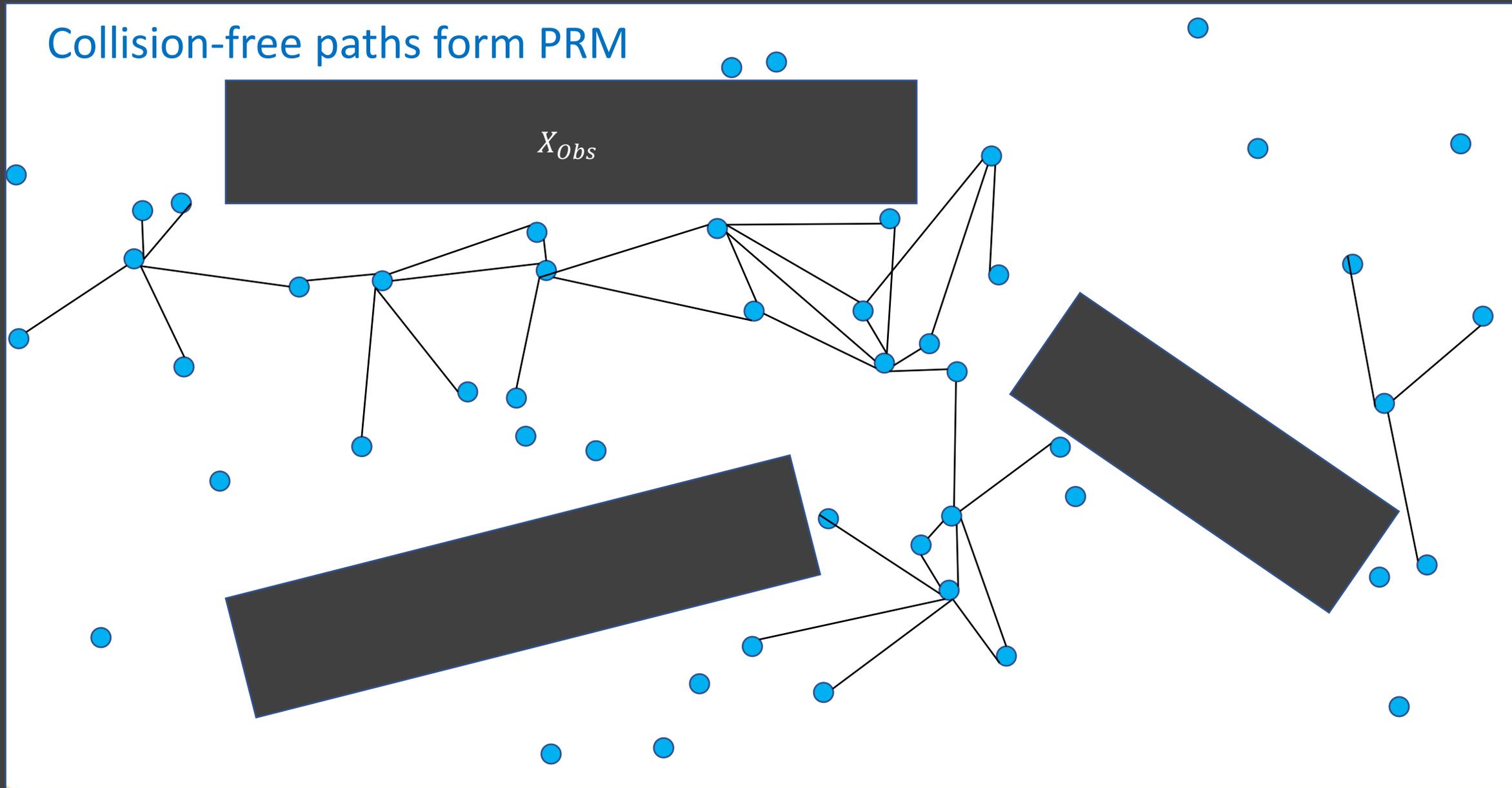
Collision-free paths form PRM

X_{Obs}



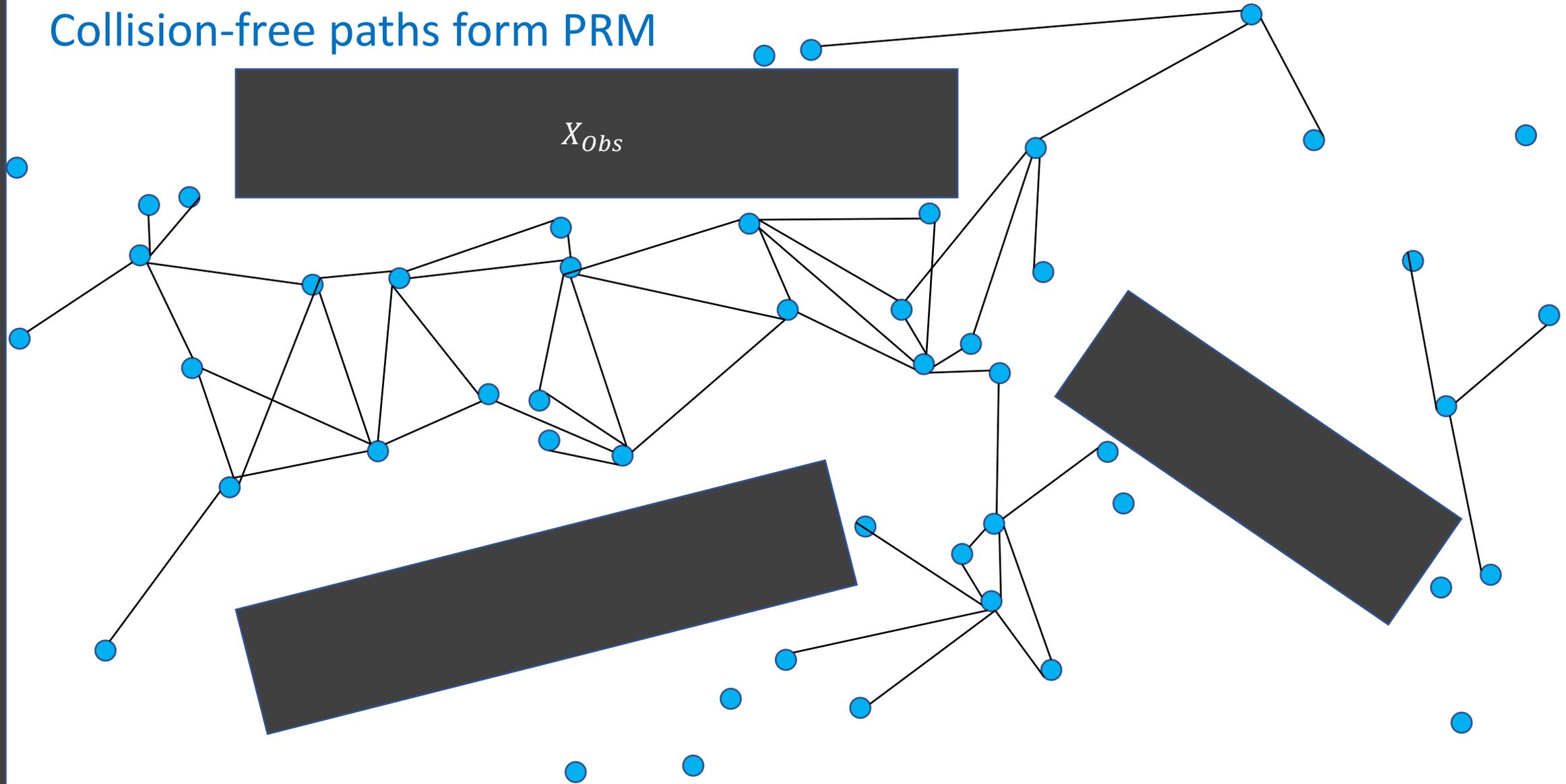
Collision-free paths form PRM

X_{Obs}



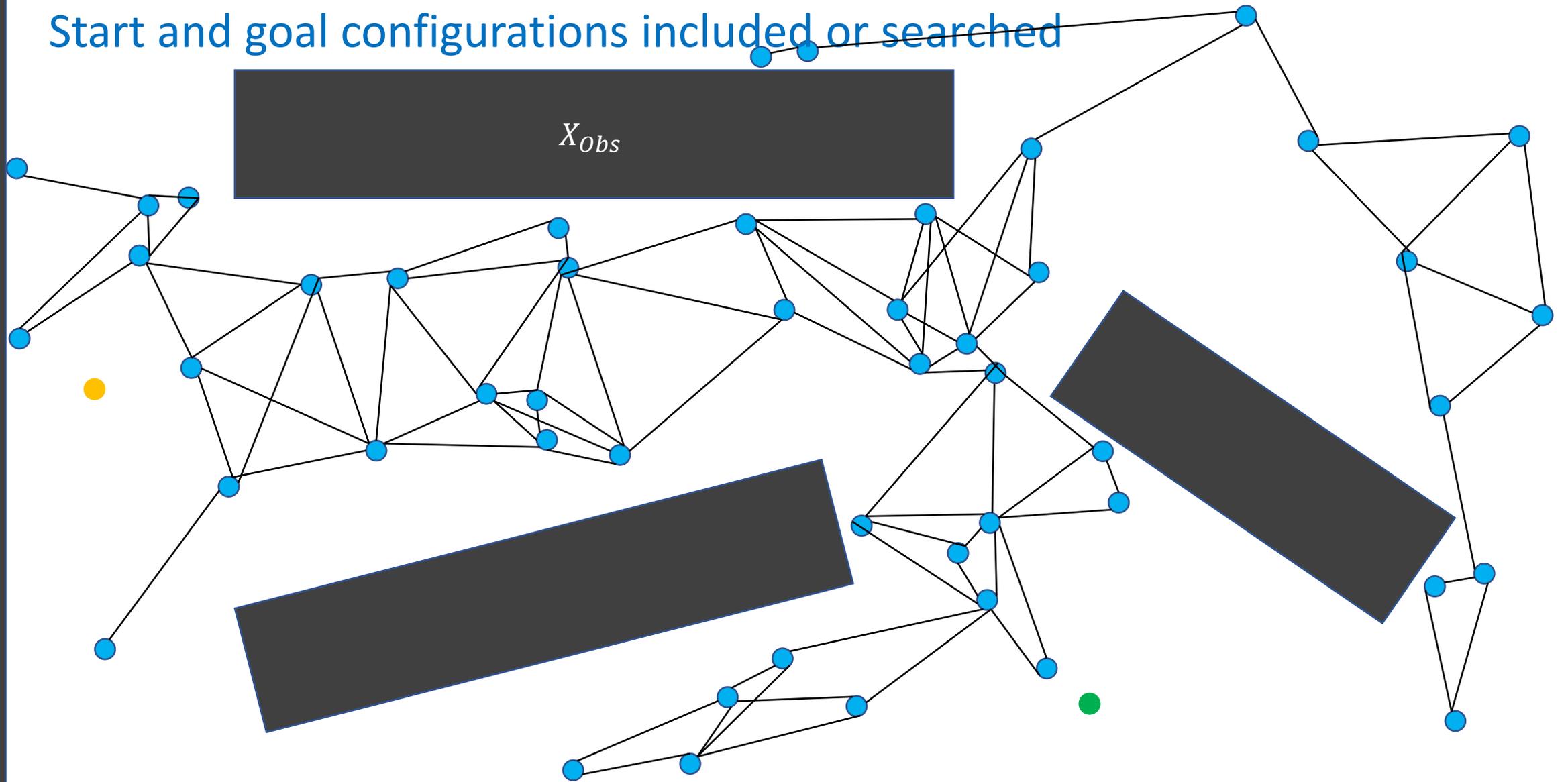
Collision-free paths form PRM

X_{Obs}



Start and goal configurations included or searched

X_{Obs}



Remarks on PRM

Pros.

- PRM has been shown to be ***probabilistically complete***
 - Moreover, the probability of success goes to 1 exponentially fast, if the environment satisfies certain “good visibility” conditions.

Cons:

- NOT asymptotically optimal
- Builds graph without particular focus on generating path
- Required to solve 2-point collision detection problem



Complexity of Sampling-based Algorithms

- How can we measure complexity for an algorithm that does not necessarily terminate?
 - Treat the number of samples as “the size of the input.” (Everything else stays the same)
 - Complexity per sample: how much work (time/memory) is needed to process one sample.
 - Useful for comparison of sampling-based algorithms. Not for deterministic, complete algorithms.
- Complexity of PRM for N samples $\Theta(N^2)$
- Practical complexity reduction tricks
 - k -nearest neighbors: connect to the k nearest neighbors. Complexity $\Theta(N \log N)$. (Finding nearest neighbors takes $\log N$ time.)
 - Bounded degree: connect at most k neighbors among those within radius r .
 - Variable radius: change the connection radius r as a function of N . How?



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



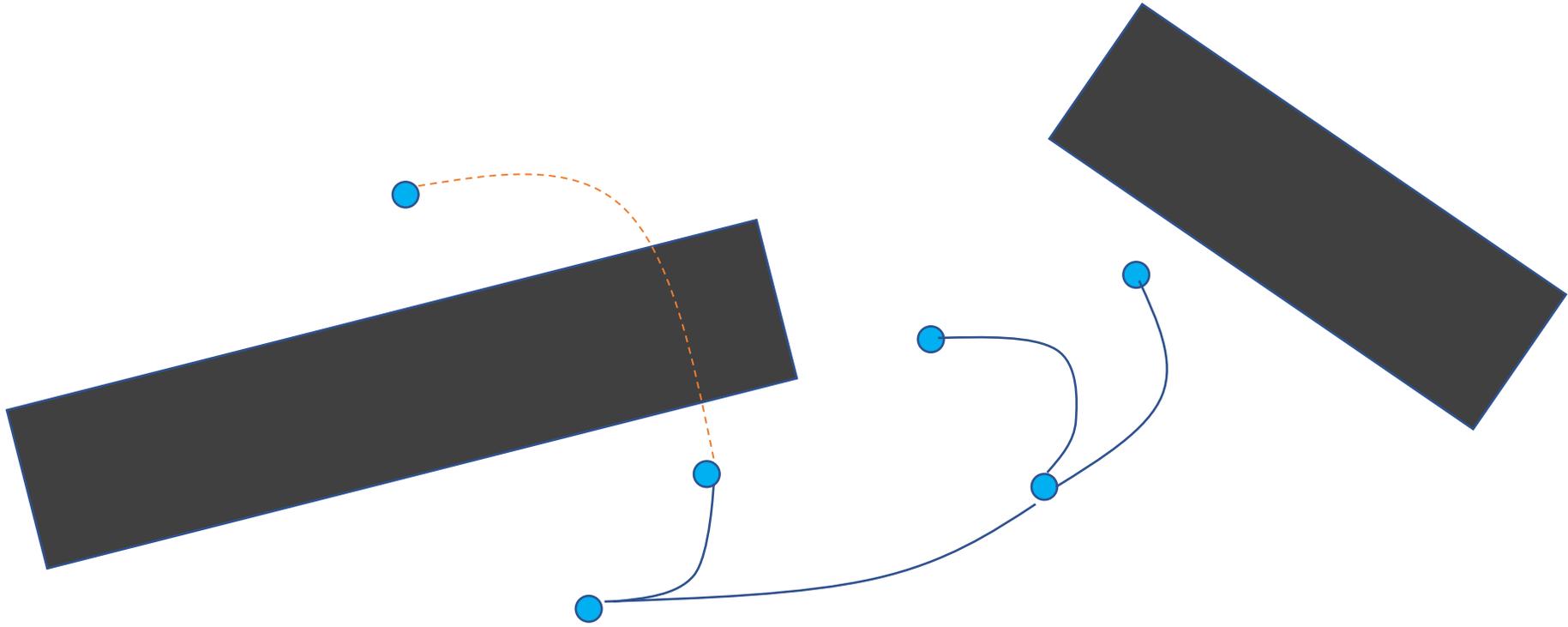
RRT

```
V ← {xinit}; E ← ∅  
for i = 1, . . . , N do  
  xrand ← SampleFreei  
  xnearest ← Nearest(G = (V, E), xrand) // Find node in G that is closest to xrand  
  xnew ← Steer(xnearest, xrand) // Use local controller to steer xnearest to xrand  
  
  if ObstacleFree(xnearest, xnew) then  
    V ← V ∪ {xnew}  
    E ← E ∪ {(xnearest, xnew)}  
return G = (V, E)
```



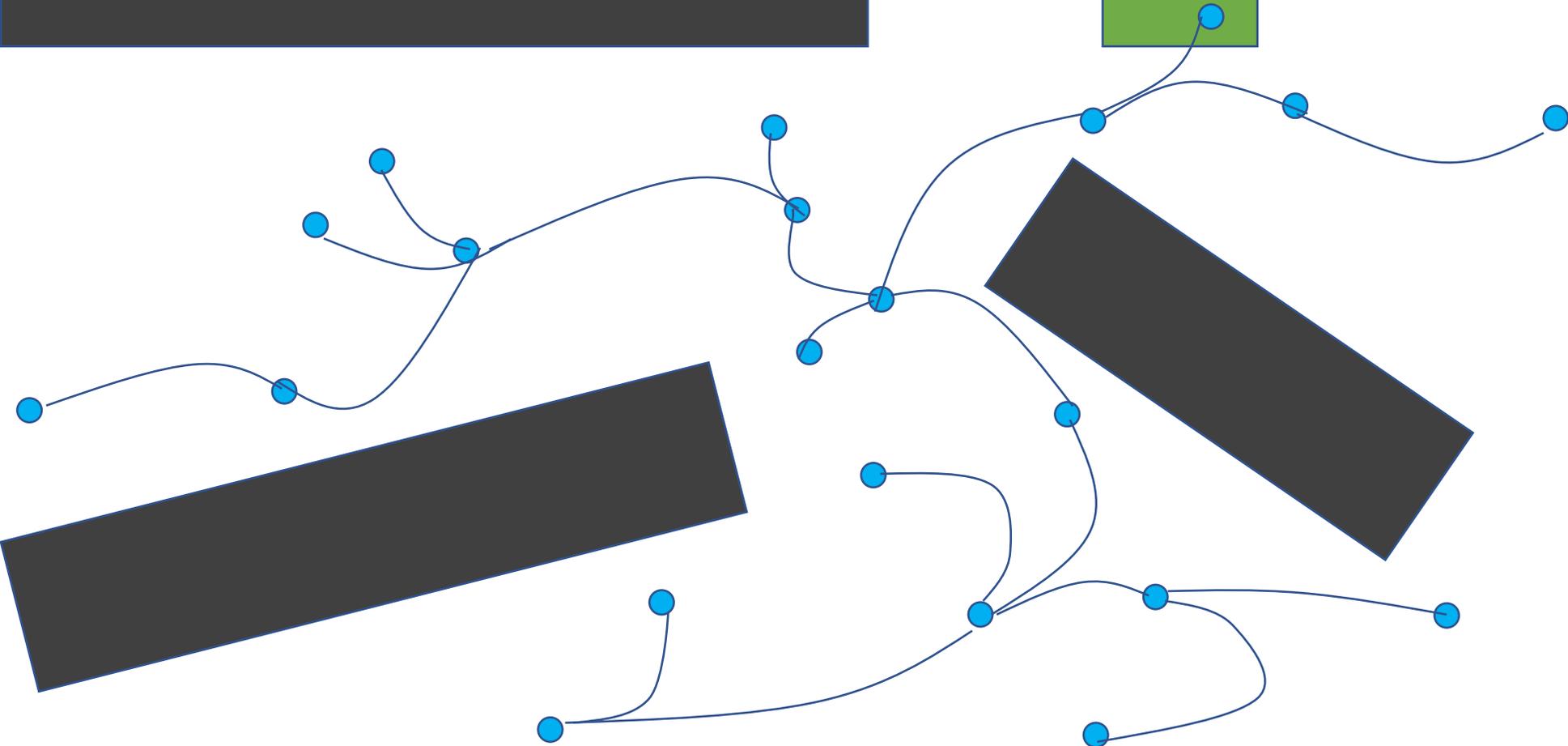
X_{Obs}

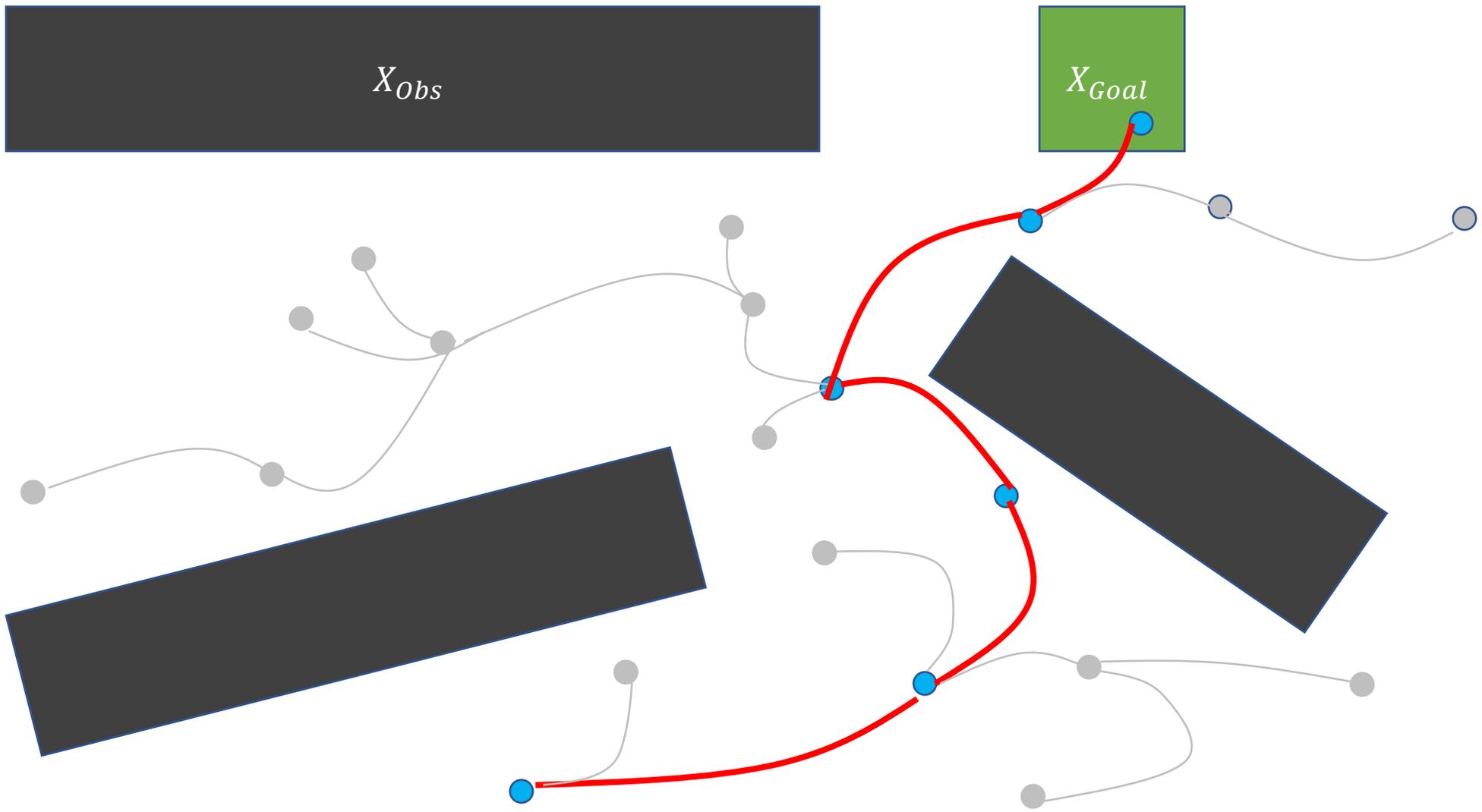
X_{Goal}



X_{Obs}

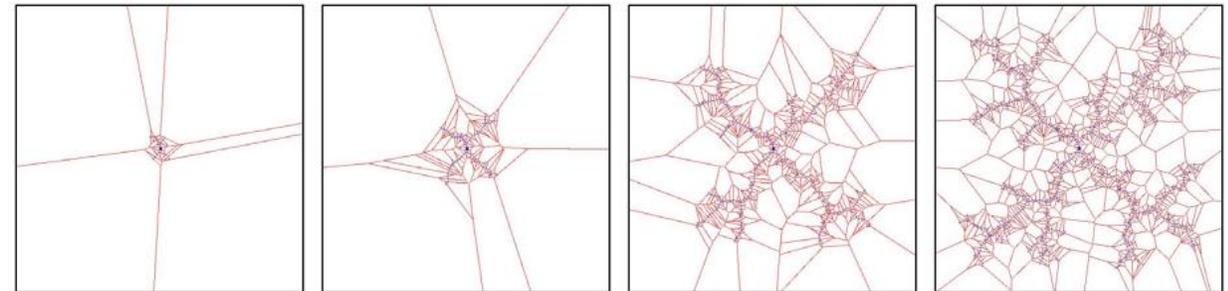
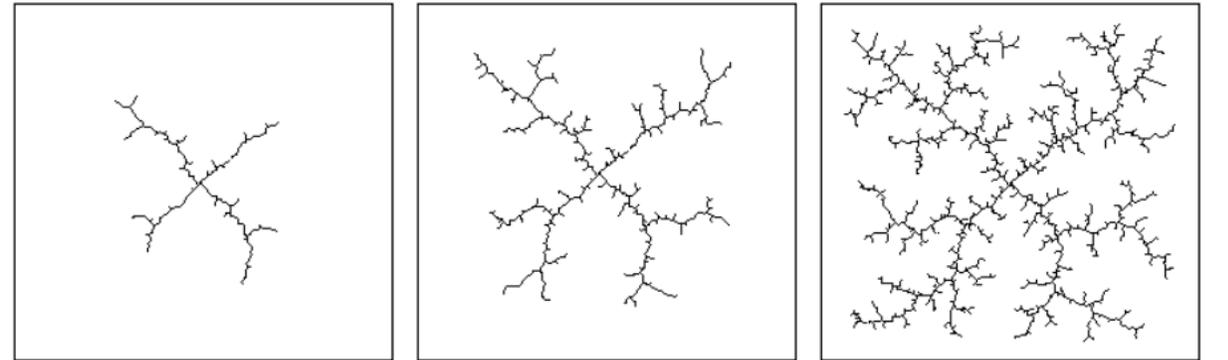
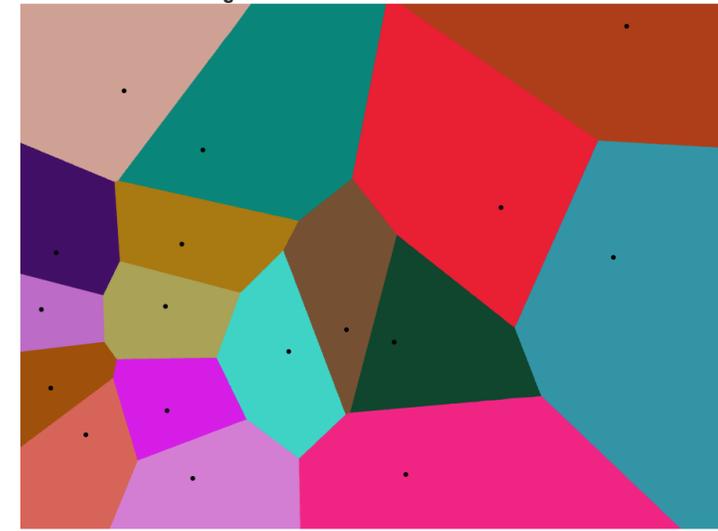
X_{Goal}





Voronoi bias

- Given n points in d dimensions, the **Voronoi diagram** of the sites is a partition of \mathbb{R}^d into regions, one region per point, such that all points in the interior of each region lie closer to that regions site than to any other site.
- Try it: <http://alexbeutel.com/webgl/voronoi.html>
- **Voronoi bias.** Vertices of the RRT that are more “isolated” (e.g., in unexplored areas, or at the boundary of the explored area) have larger Voronoi regions—and are more likely to be selected for extension.
- <http://lavalle.pl/rrt/>



RRTs and Asymptotic Optimality*

Pros.

- RRTs are ***probabilistically complete***
- RRTs are great at finding feasible trajectories quickly

Cons.

- Terrible at finding good trajectories. Why?
 - Let Y_n^{RRT} be the cost of the best path in the RRT at the end of iteration n .
 - It is shown that Y_n^{RRT} converges (to a random variable), $\lim_{n \rightarrow \infty} Y_n^{RRT} = Y_\infty^{RRT}$.
 - The random variable Y_∞^{RRT} is sampled from a distribution with zero mass at the optimum
 - **Theorem [Karaman & Frizzoli`10]** (Almost sure suboptimality of RRTs) If the set of sampled optimal paths has measure zero, the sampling distribution is absolutely continuous with positive density in X_{free} , and $d \geq 2$, then the best path in the RRT converges to a sub-optimal solution almost surely, i.e.,

$$\Pr[Y_\infty^{RRT} > c^*] = 1.$$



Why is RRT not asymptotically optimal?

- 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. Why?
- Heuristics such as running the RRT multiple times, running multiple trees concurrently etc., work better than the standard RRT, but also result in almost-sure sub-optimality.
- A careful rethinking of the RRT algorithm is required for (asymptotic) optimality.



RRT in action [Frazzoli]

- Talos, the MIT entry to the 2007 DARPA Urban Challenge, relied on an “RRT-like” algorithm for real-time motion planning and control.
- The devil is in the details: provisions needed for, e.g.,
 - Real-time, on-line planning for a safety-critical vehicle with substantial momentum.
 - Uncertain, dynamic environment with limited/faulty sensors.
- Main innovations [Kuwata, et al. '09]
 - Closed-loop planning: plan reference trajectories for a closed-loop model of the vehicle under a stabilizing feedback
 - Safety invariance: Always maintain the ability to stop safely within the sensing region.
 - Lazy evaluation: the actual trajectory may deviate from the planned one, need to efficiently re-check the tree for feasibility.
- The RRT-based P+C system performed flawlessly throughout the race.
- <https://journals.sagepub.com/doi/abs/10.1177/0278364911406761>



Limitations

The MIT DARPA Urban Challenge code, as well as other incremental sampling methods, suffer from the following limitations:

- No characterization of the quality (e.g., “cost”) of the trajectories returned by the algorithm.
- Keep running the RRT even after the first solution has been obtained, for as long as possible (given the real-time constraints), hoping to find a better path than that already available.
- No systematic method for imposing temporal/logical constraints, such as, e.g., the rules of the road, complicated mission objectives, ethical/deontic code.
- In the DARPA Urban Challenge, all logics for, e.g., intersection handling, had to be hand-coded, at a huge cost in terms of debugging effort/reliability of the code.



Rapidly Exploring Random Graphs (possibly cyclic)

```
V ← {xinit}; E ← ∅;  
for i = 1, . . . , N do  
  xrand ← SampleFree;  
  xnearest ← Nearest(G = (V, E), xrand);  
  xnew ← Steer(xnearest, xrand);  
  if ObstacleFree(xnearest, xnew) then  
    Xnear ← Near(G = (V, E), xnew, min{γRRG(log(card V)/ card V)1/d, η});  
    V ← V ∪ {xnew}; E ← E ∪ {(xnearest, xnew), (xnew, xnearest)};  
    foreach xnear ∈ Xnear do  
      if CollisionFree(xnear, xnew) then E ← E ∪ {(xnear, xnew), (xnew, xnear)}  
return G = (V, E);
```

At each iteration, the RRG tries to connect the new sample to all vertices in a ball of radius r_n centered at it. (Or just default to the nearest one if such ball is empty.)



Theorems [Proofs not required for exam]

- **Probabilistic completeness.** Since $V_n^{RRG} = V_n^{RRT}$, for all n RRG has the same completeness properties as RRT, i.e.,

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

- **Asymptotic optimality.** If the **Near** procedure returns all nodes in V within a ball of volume $Vol = \frac{\gamma \log n}{n}$, $\gamma > 2^d \left(1 + \frac{1}{d}\right)$, 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.,

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



Final thoughts on RRG

- What is the additional computational load?
 - $O(\log n)$ extra calls to ObstacleFree compared to RRT
- Key idea in RRG/RRT*:
 - 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.



Summary and future directions

- State-of-the-art algorithms such as RRT converge to a NON-optimal solution almost-surely
- new algorithms (RRG and the RRT*), which almost-surely converge to optimal solutions while incurring no significant cost overhead
- Bibliographical reference: S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. Int. Journal of Robotics Research, 2011. TAlso available at <http://arxiv.org/abs/1105.1186>.
- research directions:
 - Optimal motion planning with temporal/logic constraints
 - Anytime solution of differential games
 - Stochastic optimal motion planning (process + sensor noise)
 - Multi-agent problems.



Algorithm	Prob. Completeness	Asymptotic Optimality	Complexity
sPRM	Yes	Yes	$O(N)$
k-nearest sPRM	No	No	$O(\log N)$
RRT	Yes	No	$O(\log N)$
PRM*	Yes	Yes	$O(\log N)$
k-nearest PRM*	Yes	Yes	$O(\log N)$
RRG	Yes	Yes	$O(\log N)$
k-nearest RRG	Yes	Yes	$O(\log N)$
RRT*	Yes	Yes	$O(\log N)$
k-nearest RRT*	Yes	Yes	$O(\log N)$

