Motion Planning for Autonomous Vehicles

Michal Cap

Autonomous Car Architecture

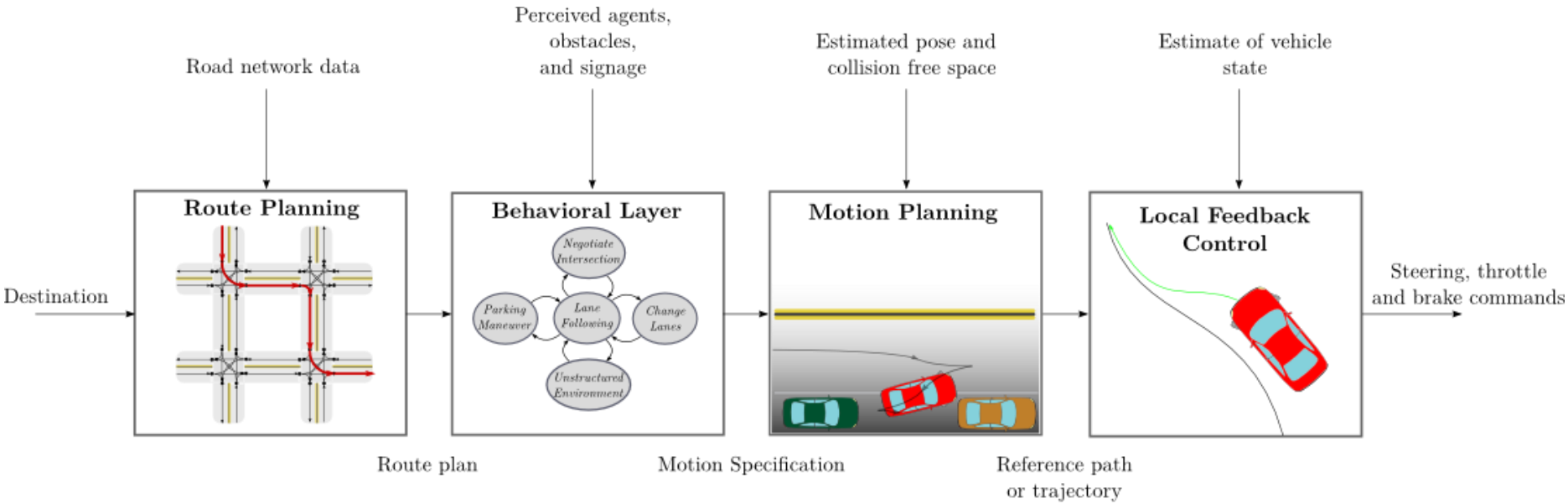


Figure from Brian Paden

Autonomous Car Architecture

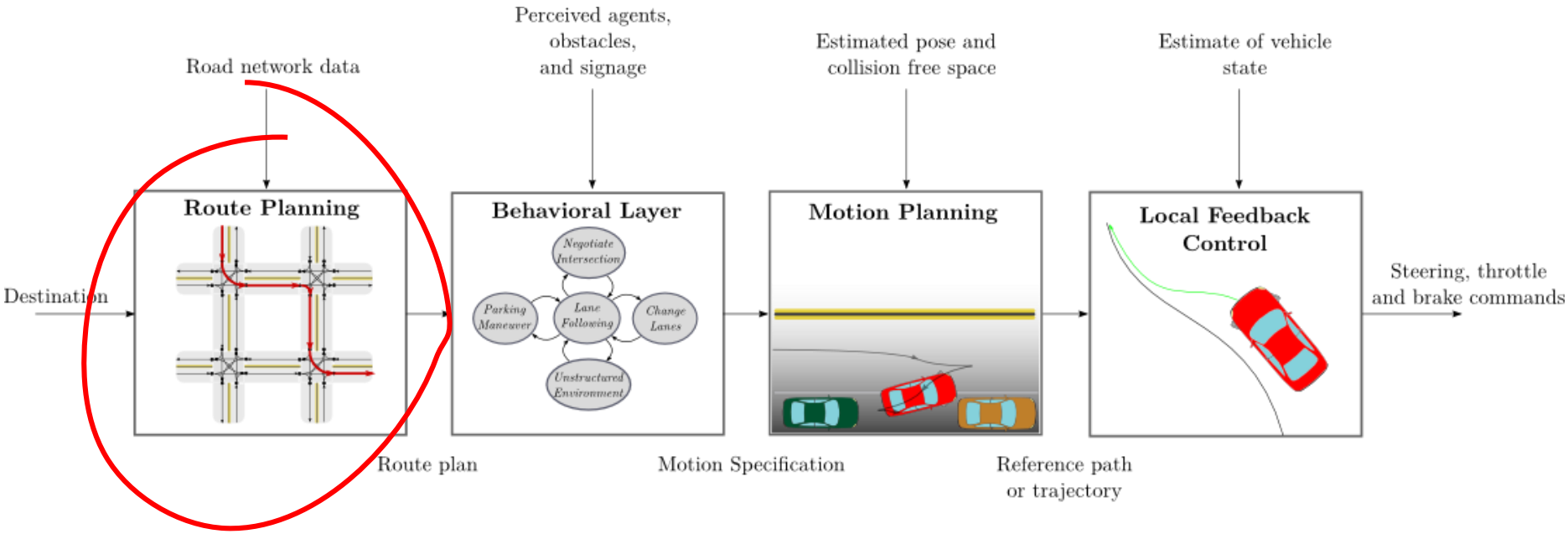


Figure from Brian Paden

Autonomous Car Architecture

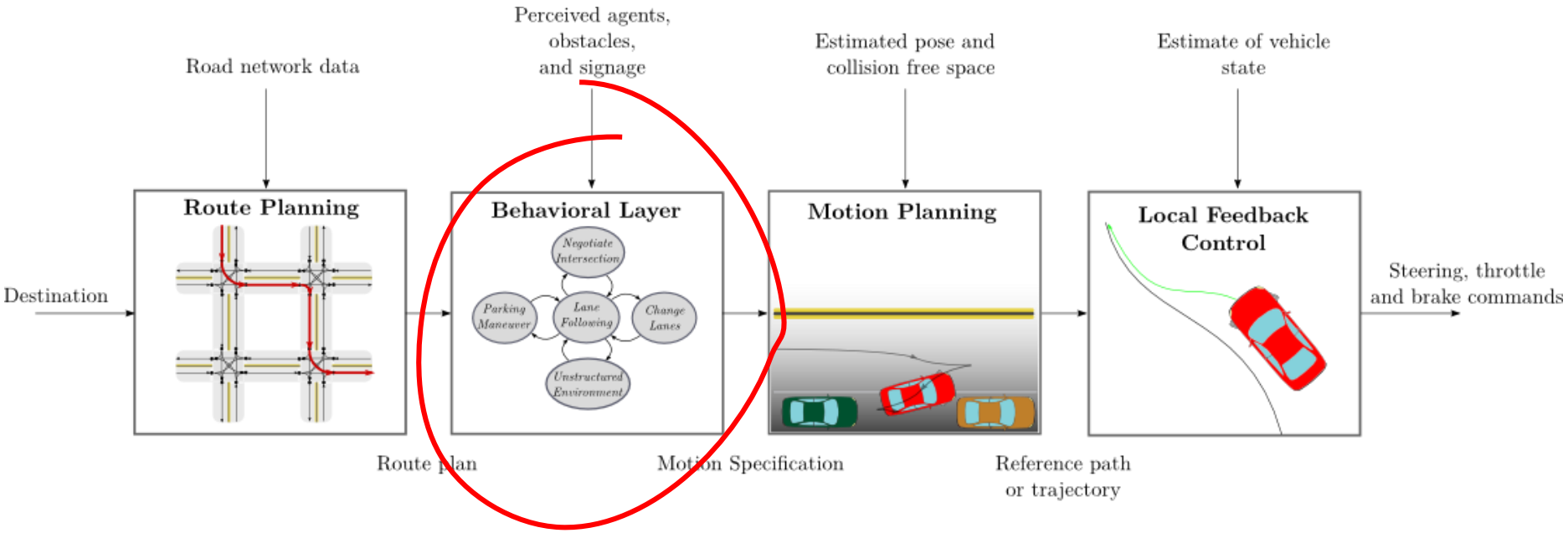


Figure from Brian Paden

Autonomous Car Architecture

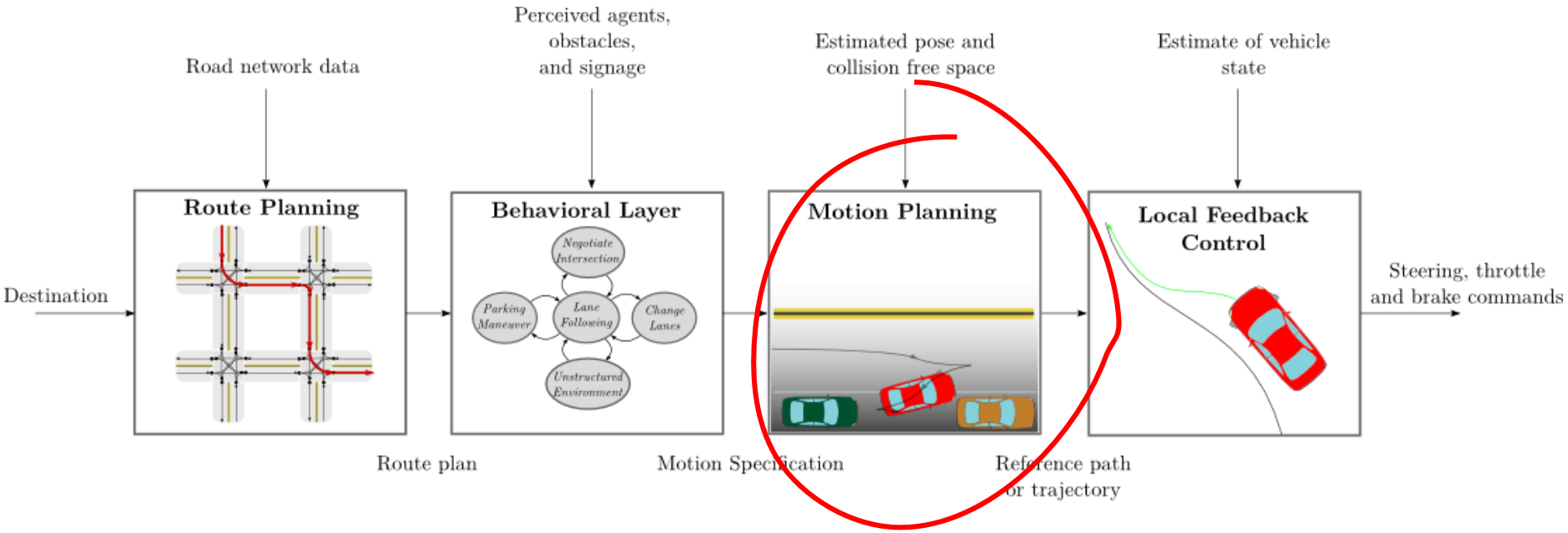


Figure from Brian Paden

Autonomous Car Architecture

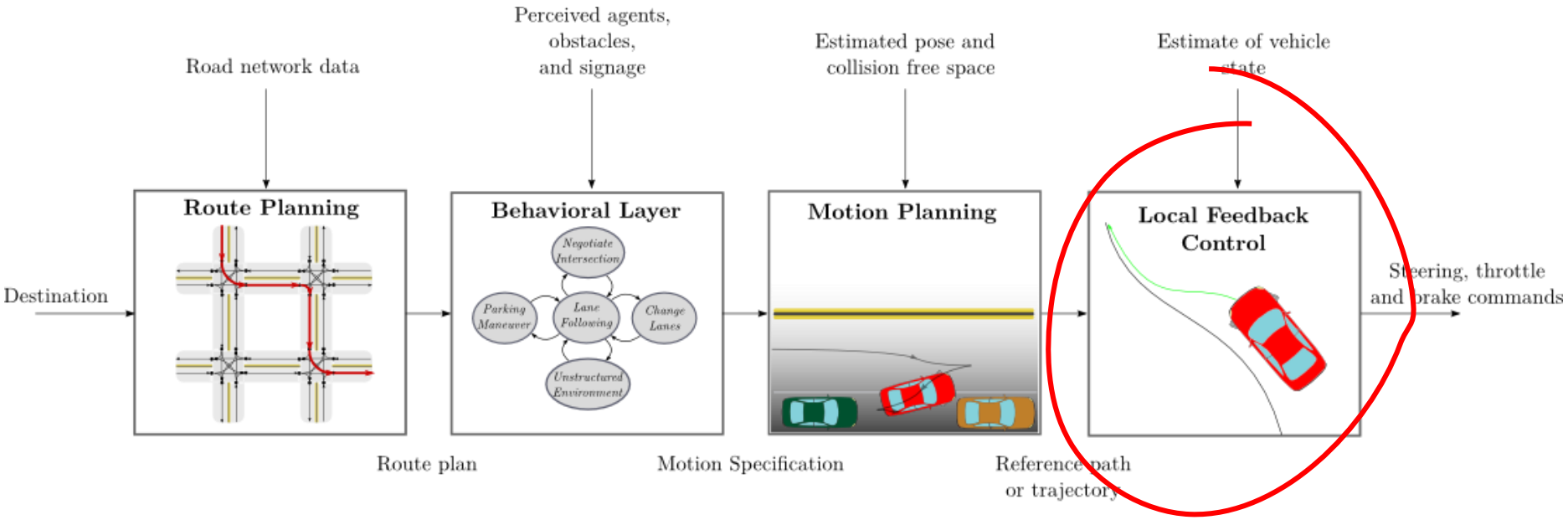


Figure from Brian Paden

Motivation

Motivation

Driving in highly structured environments;

Video by Ricco831

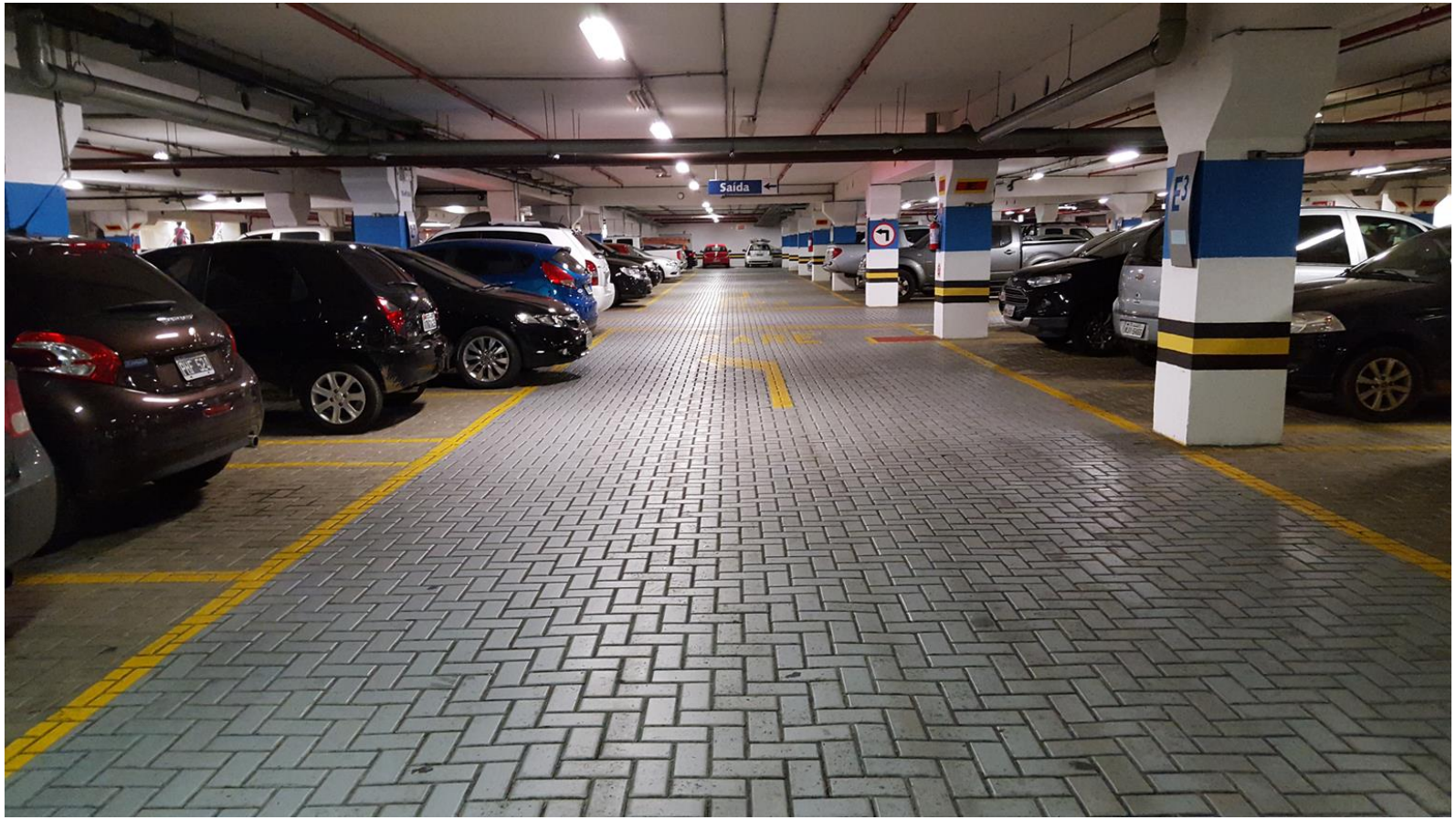
Motivation

Driving in urban environments:

Video by Mobileye

Motivation

Driving in unstructured cluttered environments:



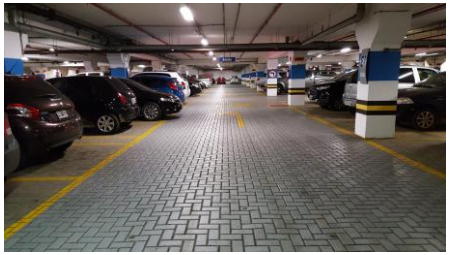
Picture by Rachmaninoff

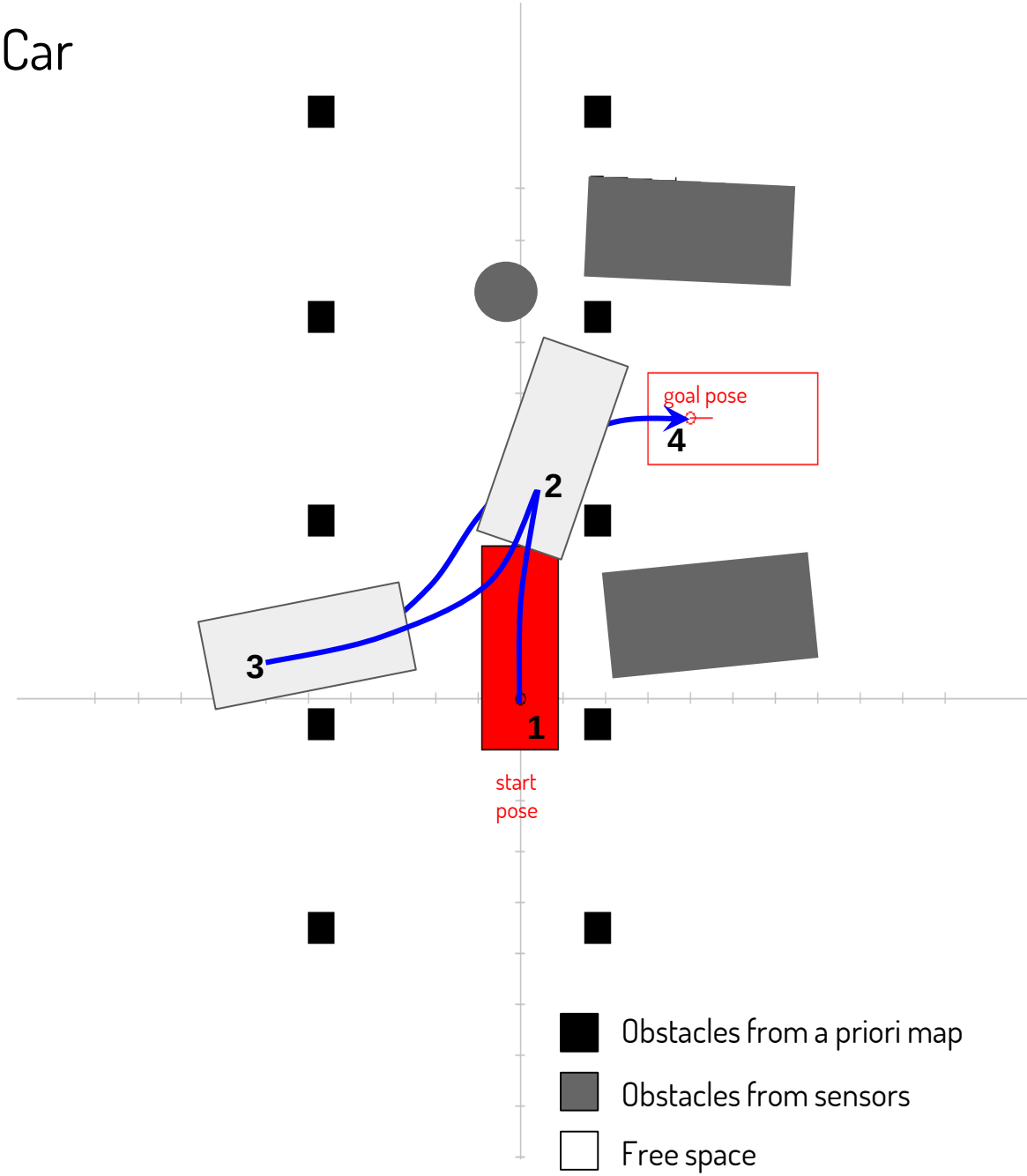
Problem Informally

Motion Planning:

The problem of finding a collision-free motion for a robot from given start pose to given destination pose.

Planning for a Conventional Car





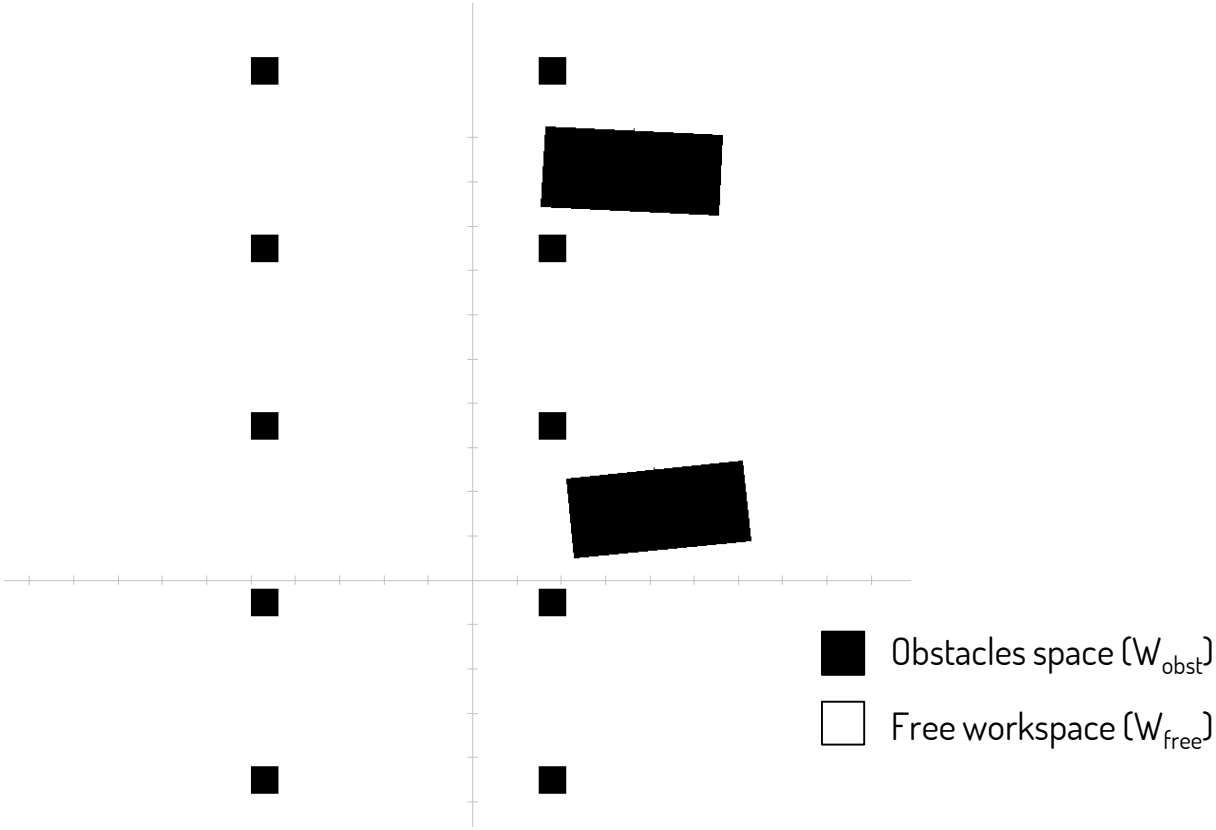
Constraints on path:

- Starts at current position
- Ends at goal position
- Robot does not collide with obstacles
- Respect limited turning-radius

Formalization

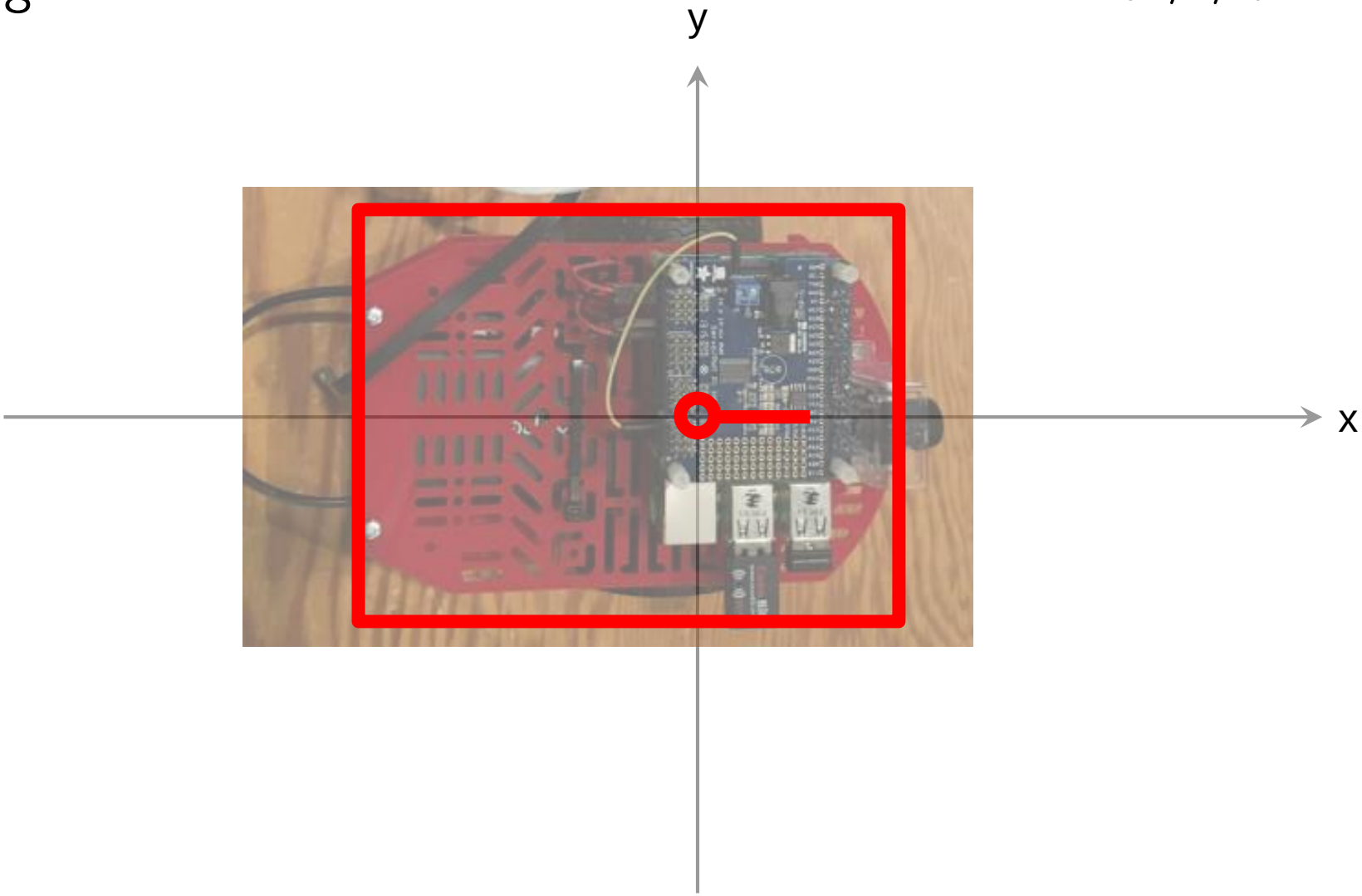
Workspace: (x,y)

$W \subset R^2$



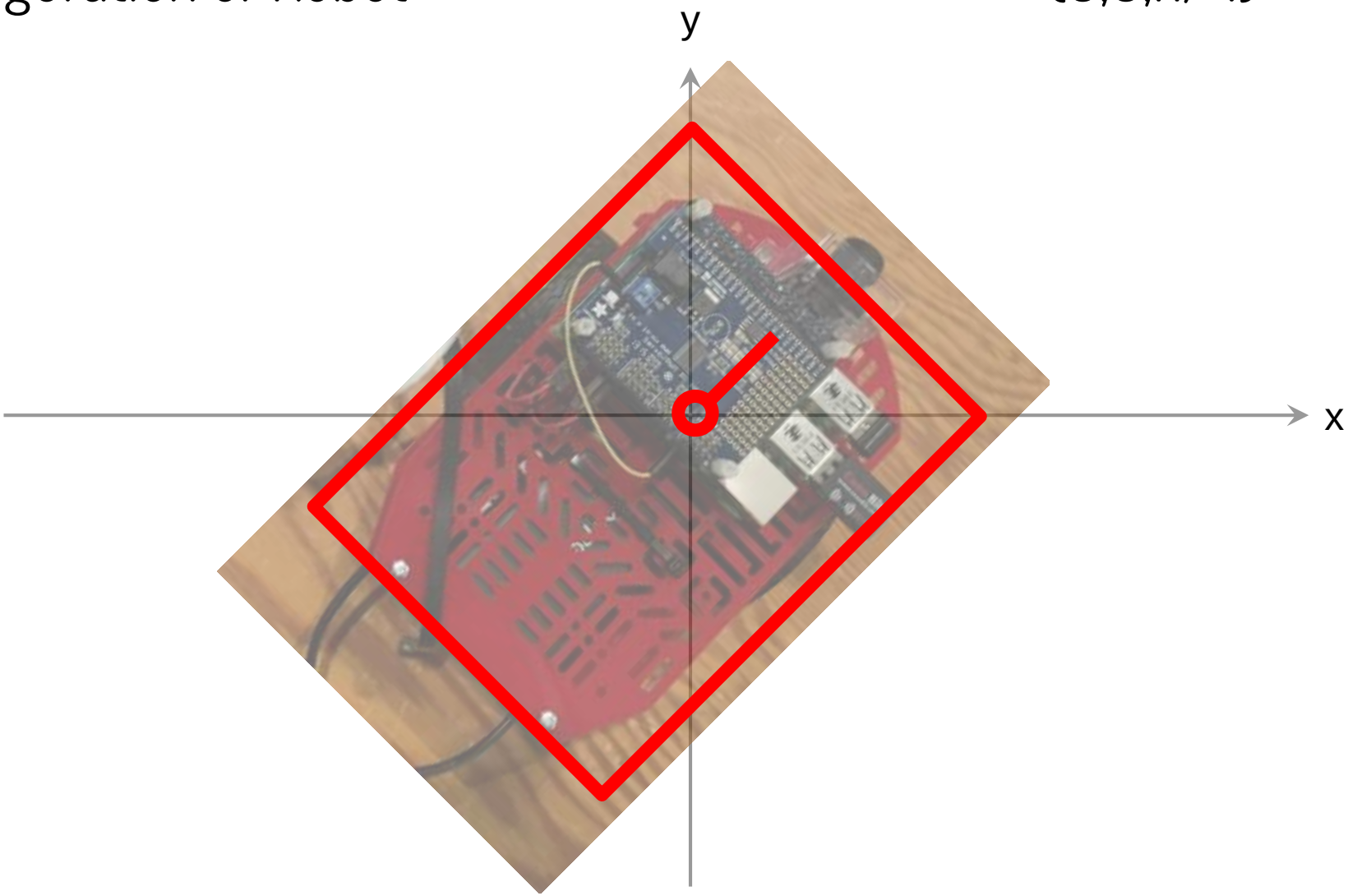
Configuration of Robot

(0,0,0)



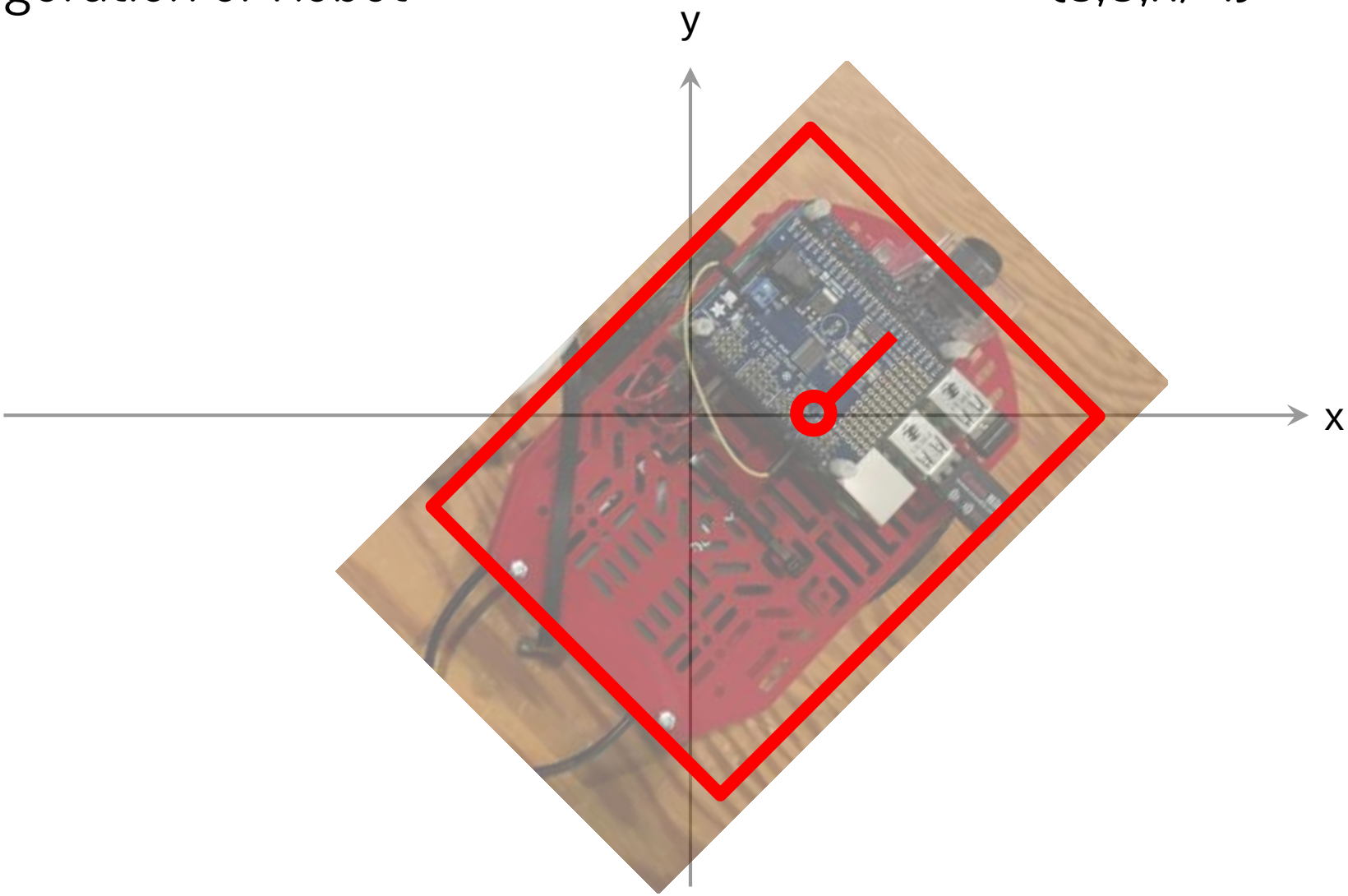
Configuration of Robot

(0,0,π/4)



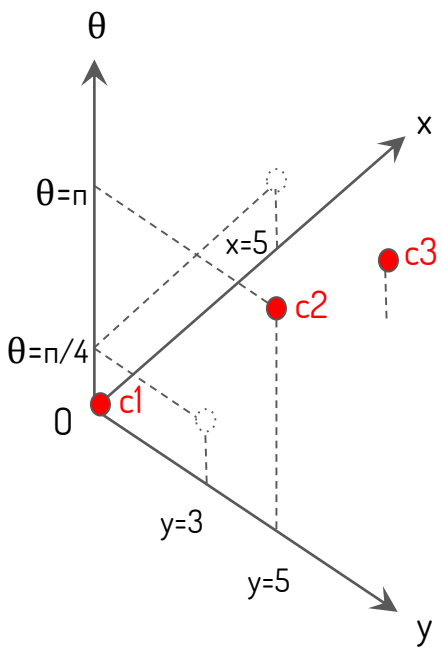
Configuration of Robot

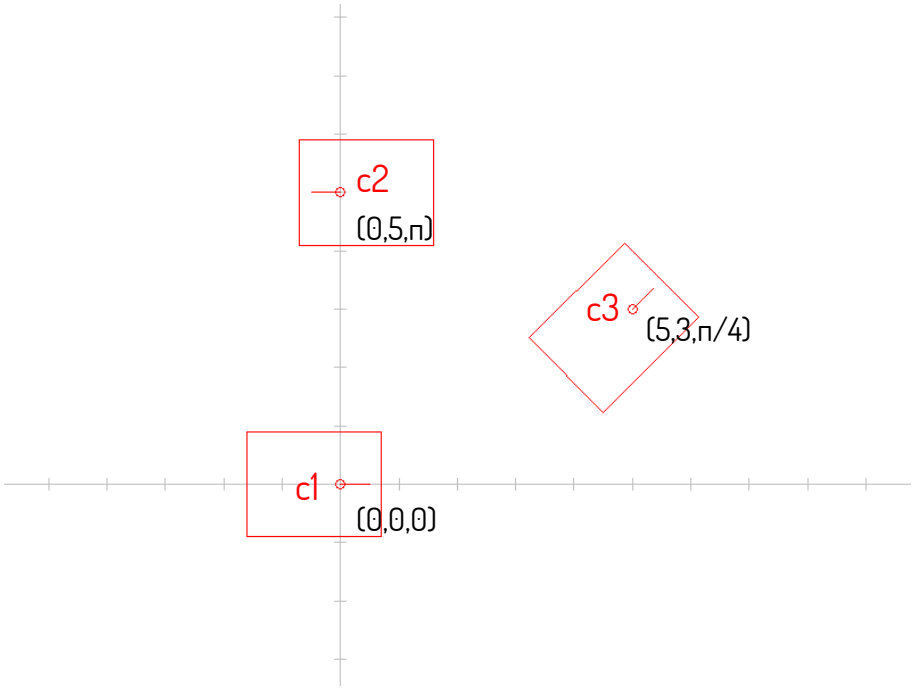
(5,0,π/4)



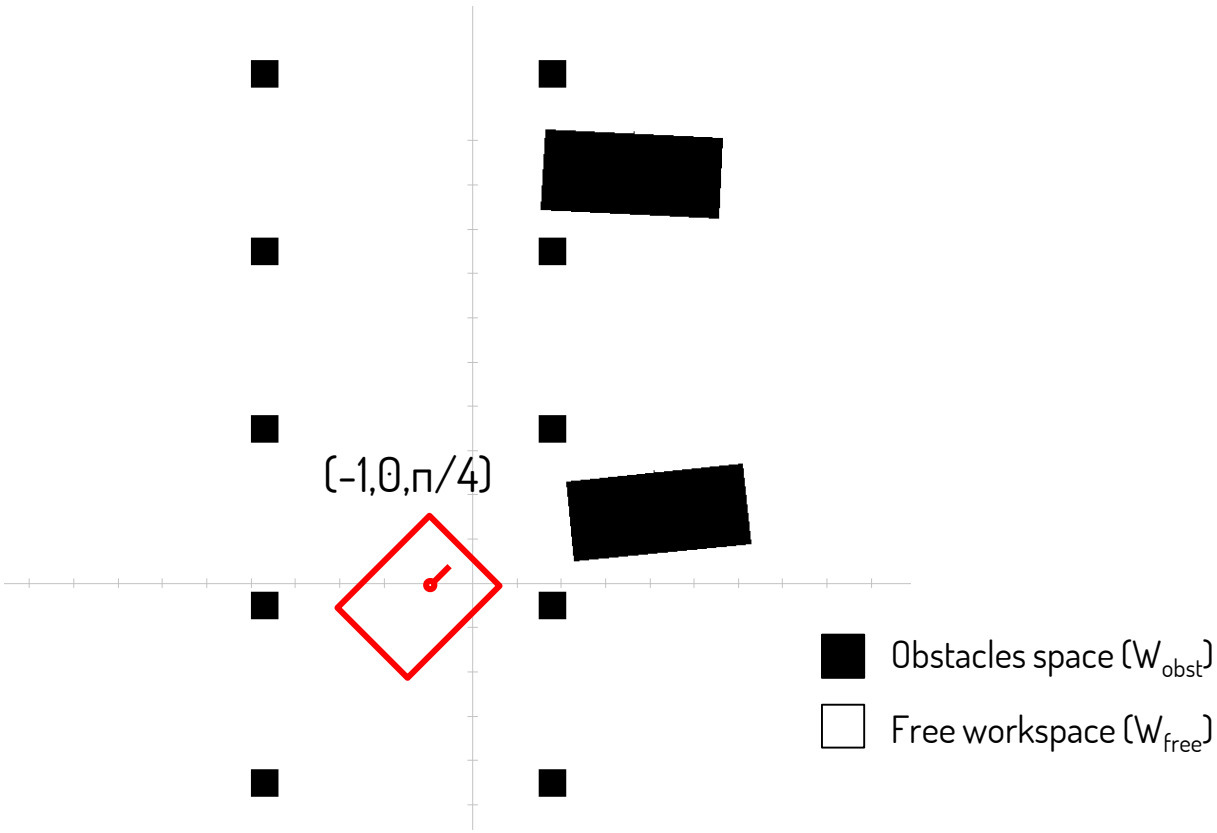
Configuration Space: (x,y,θ)

$$X \subset R^2 \times [-\pi,+\pi]$$

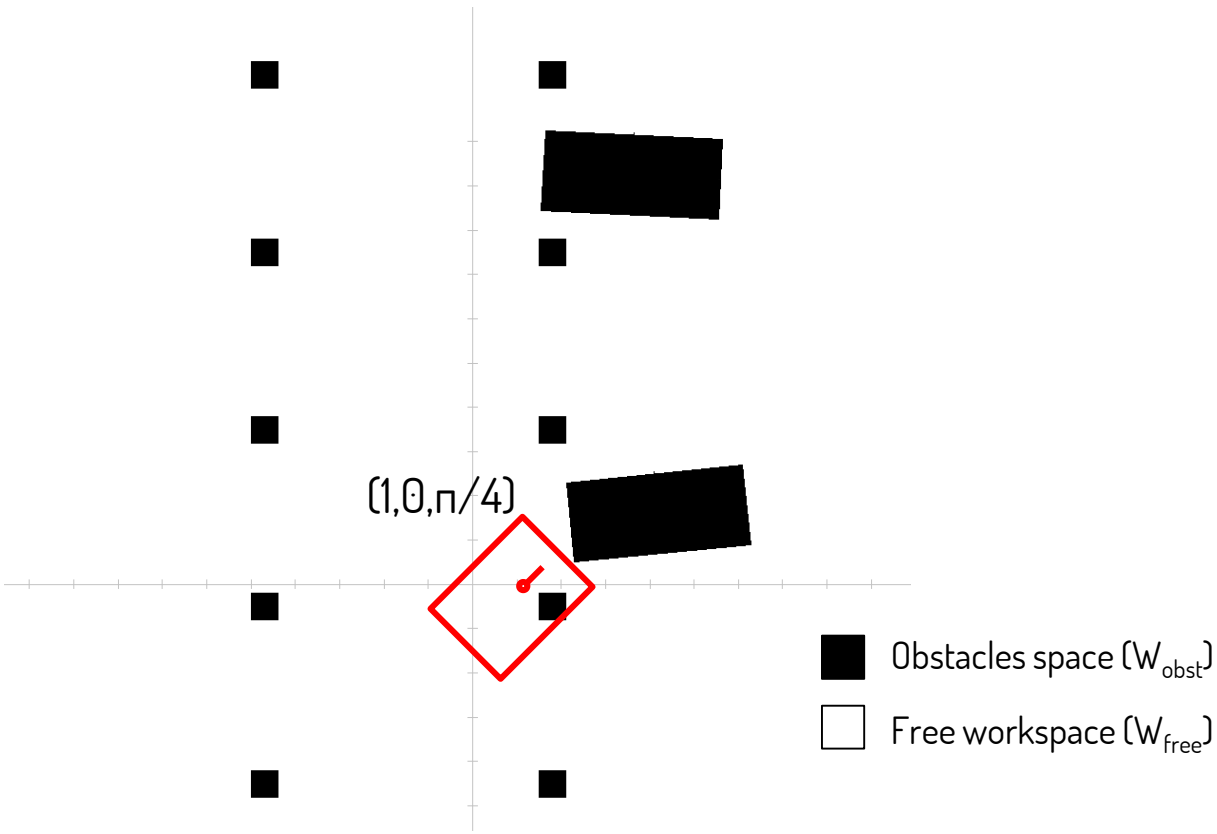




Free Configuration:



Collision Configuration:



Free Configuration Space

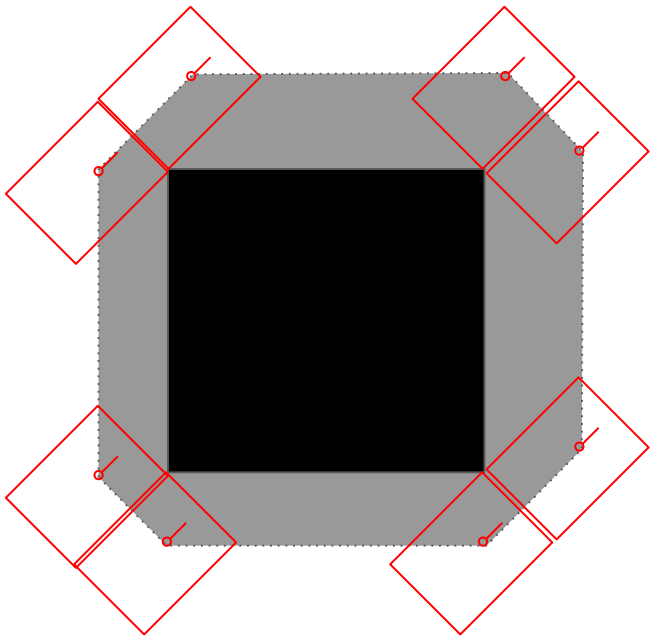
Let R(x) be the region occupied by robot at configuration $x \in X$.

The set of all collision-free configurations is

$$X_{free} = \{x : x \in X \text{ and } R(x) \subset W_{free}\}$$

Free Configuration Space

Slice of X_{free} for $\theta=\pi/4$:



Obstacle space (W_{obst})

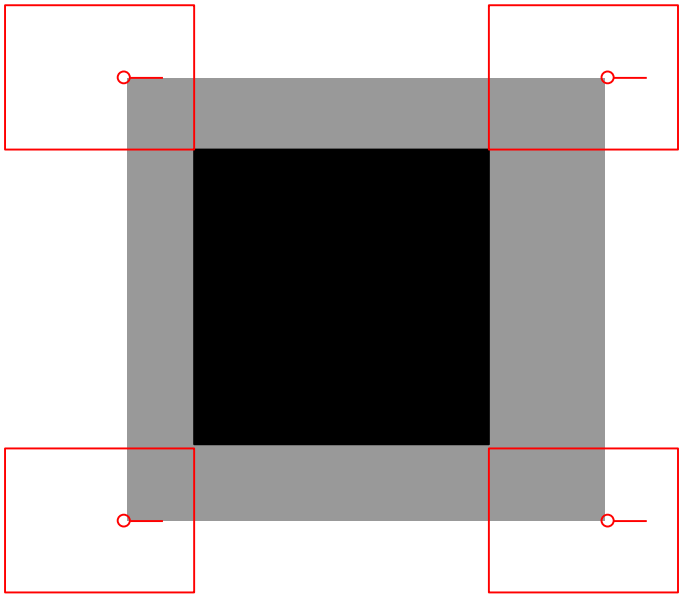
Collision configurations

Free configuration space (C_{free})

Free Configuration Space -- Illustration

Free Configuration Space

$\theta = 0$

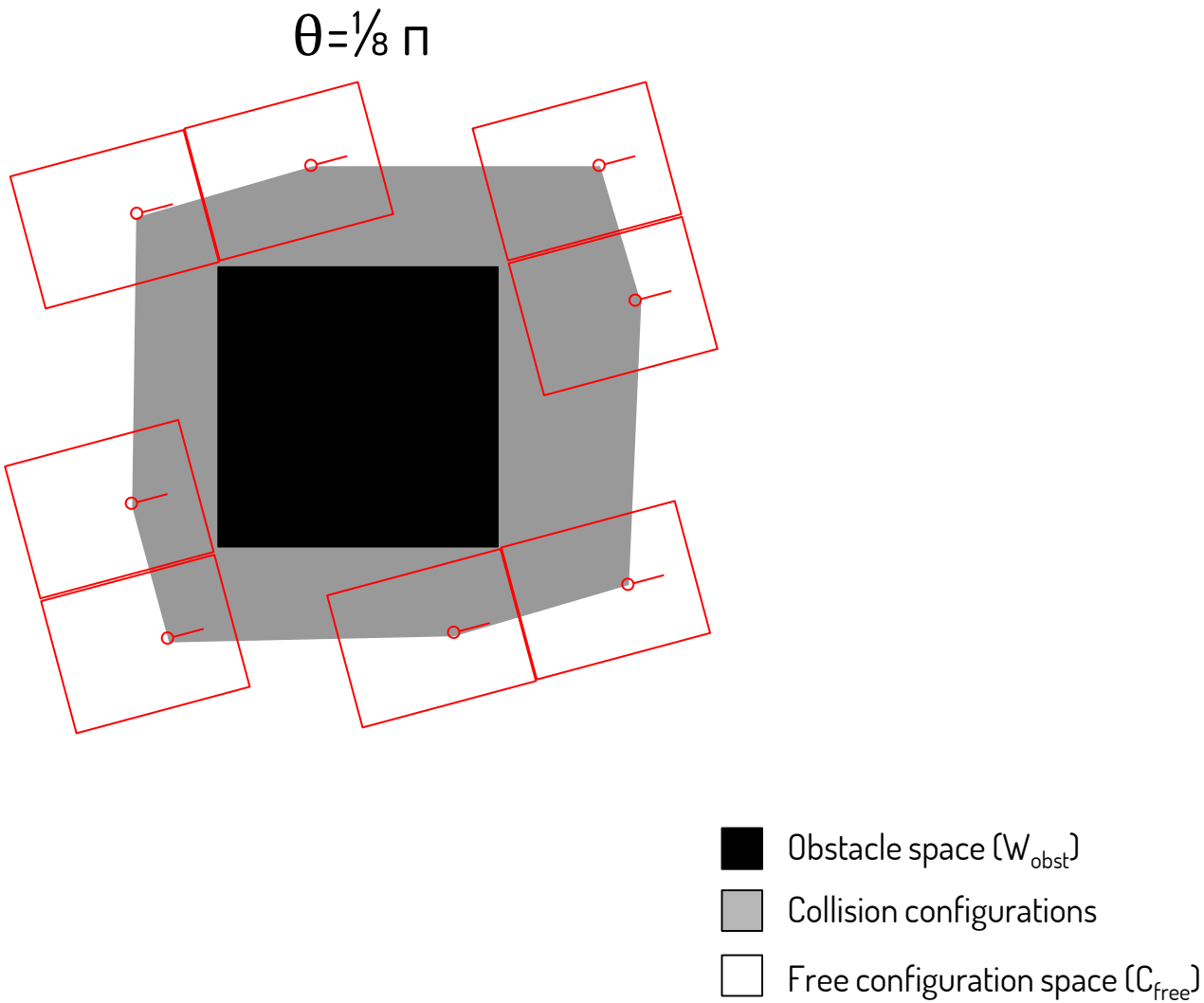


Obstacle space (W_{obst})

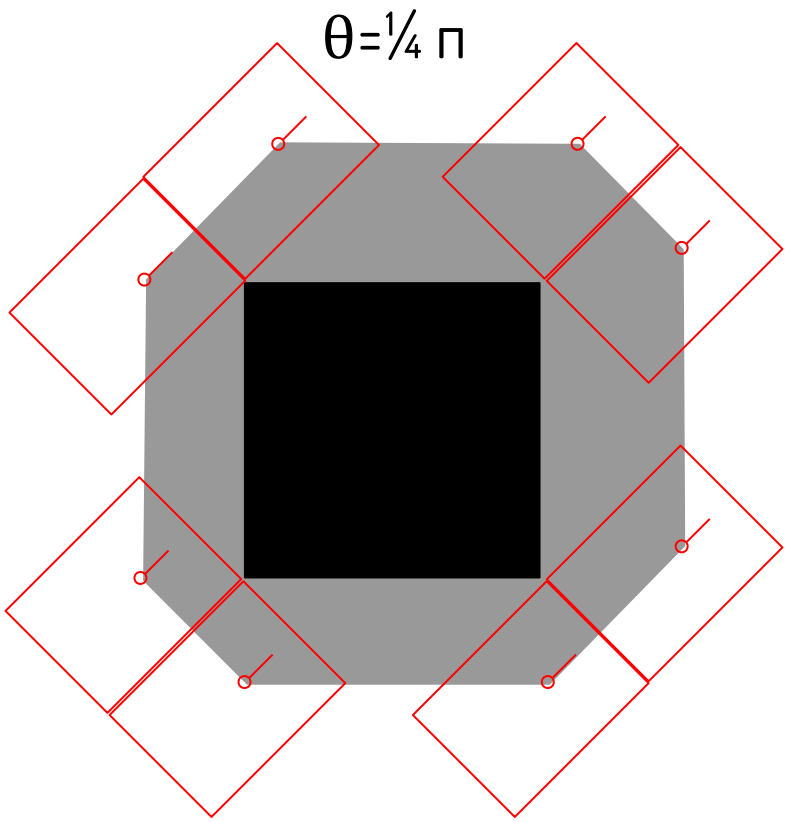
Collision configurations

Free configuration space (C_{free})

Free Configuration Space



Free Configuration Space

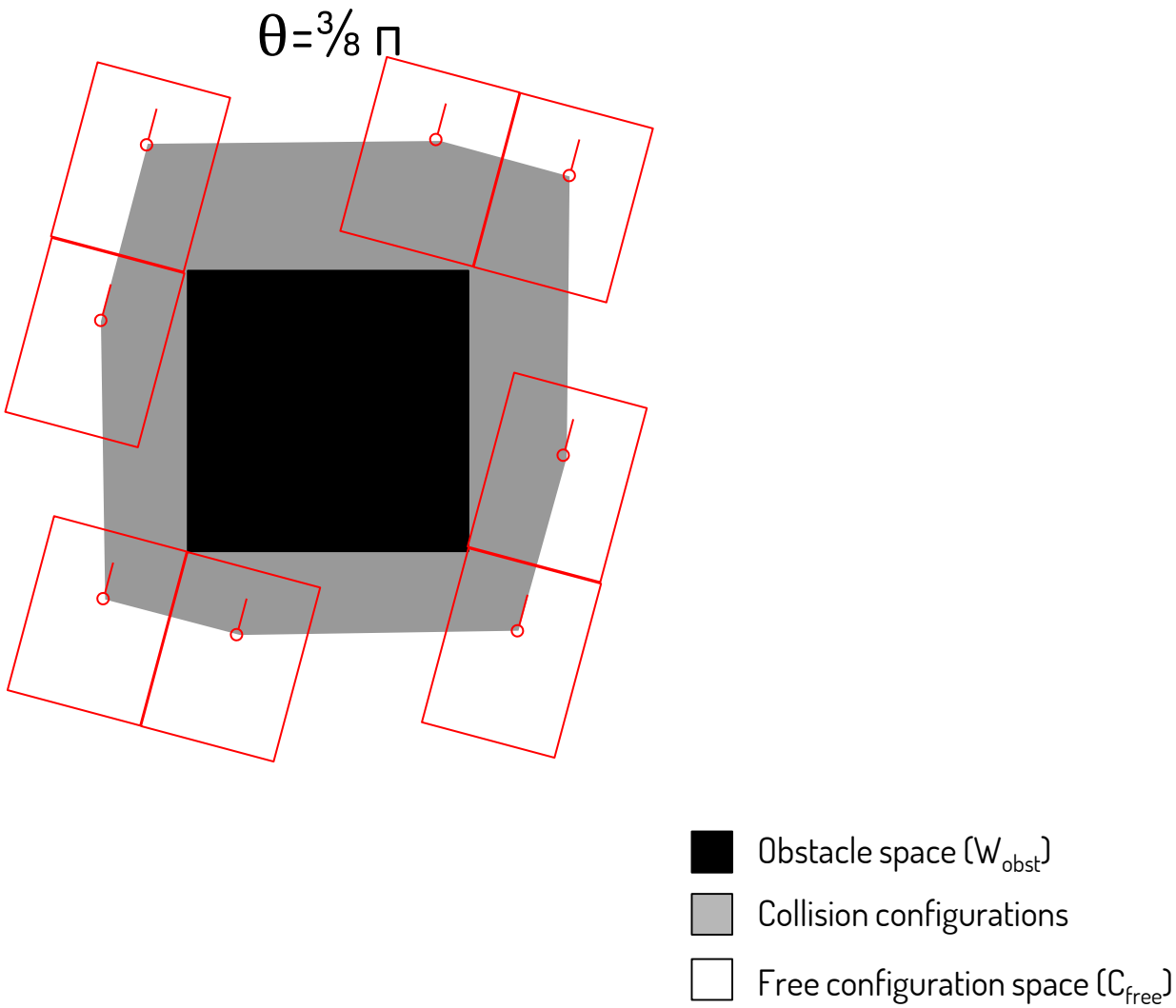


Obstacle space (W_{obst})

Collision configurations

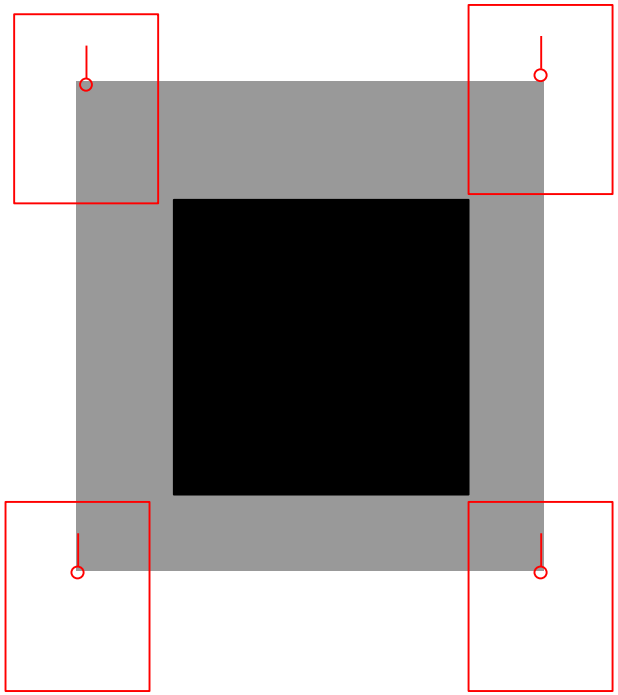
Free configuration space (C_{free})

Free Configuration Space



Free Configuration Space

$\theta = \frac{1}{2}\pi$



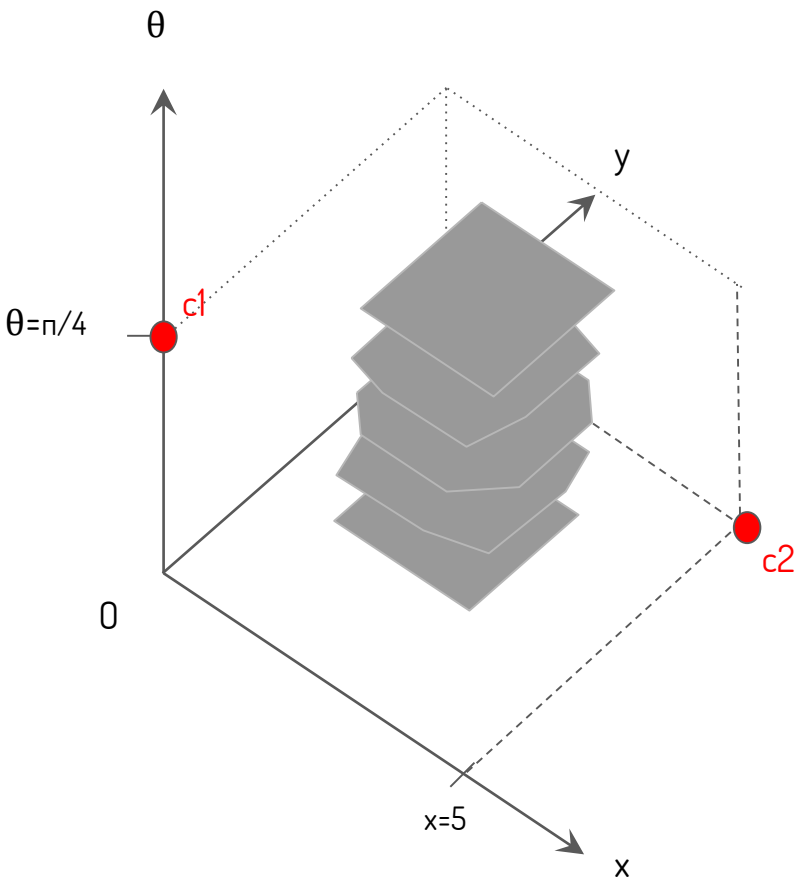
Obstacle space (W_{obst})

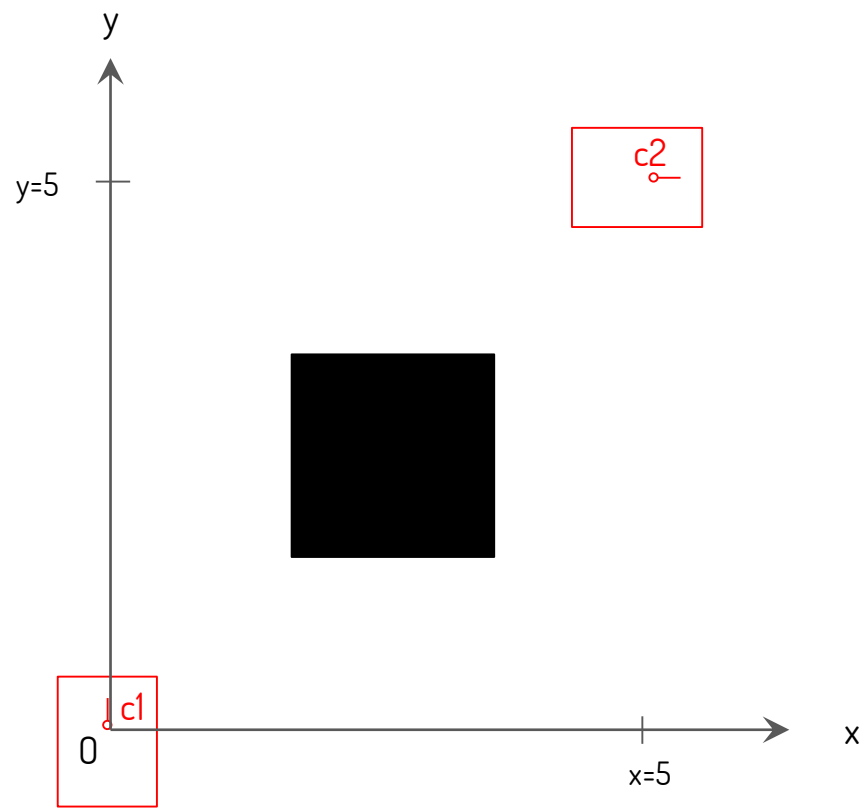
Collision configurations

Free configuration space (C_{free})

Path in Configuration Space

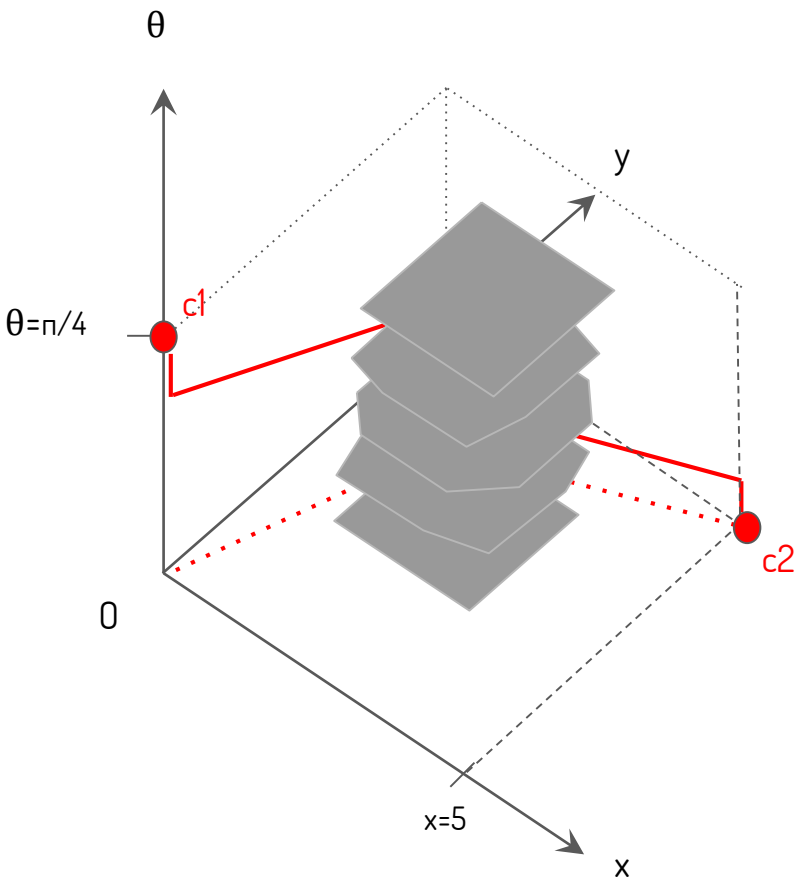
$\sigma(\alpha): [0,1] \rightarrow X$

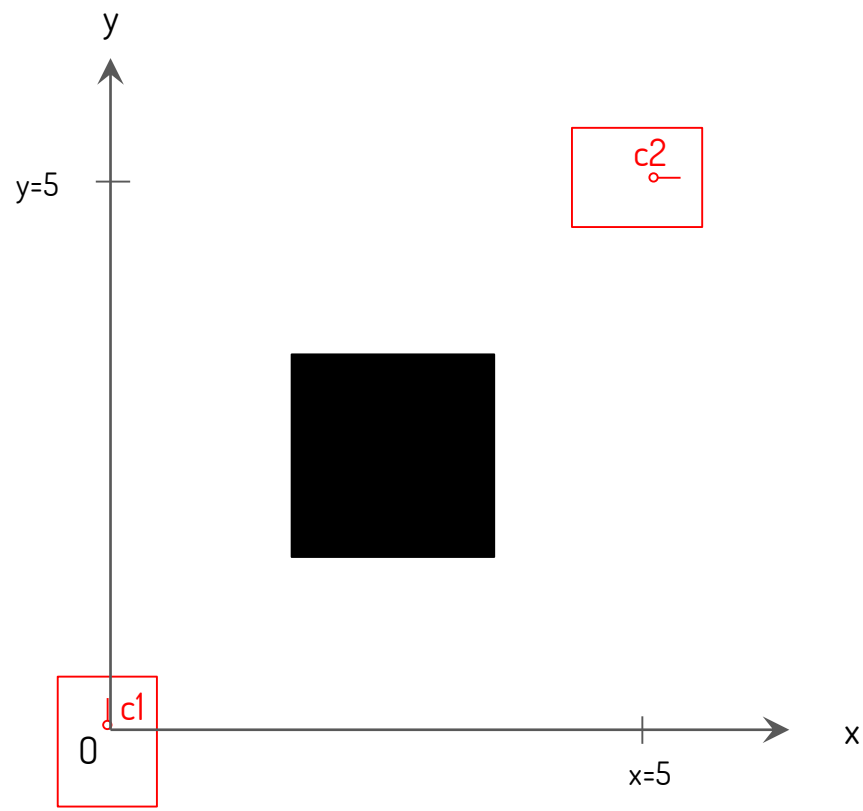




Path in Configuration Space

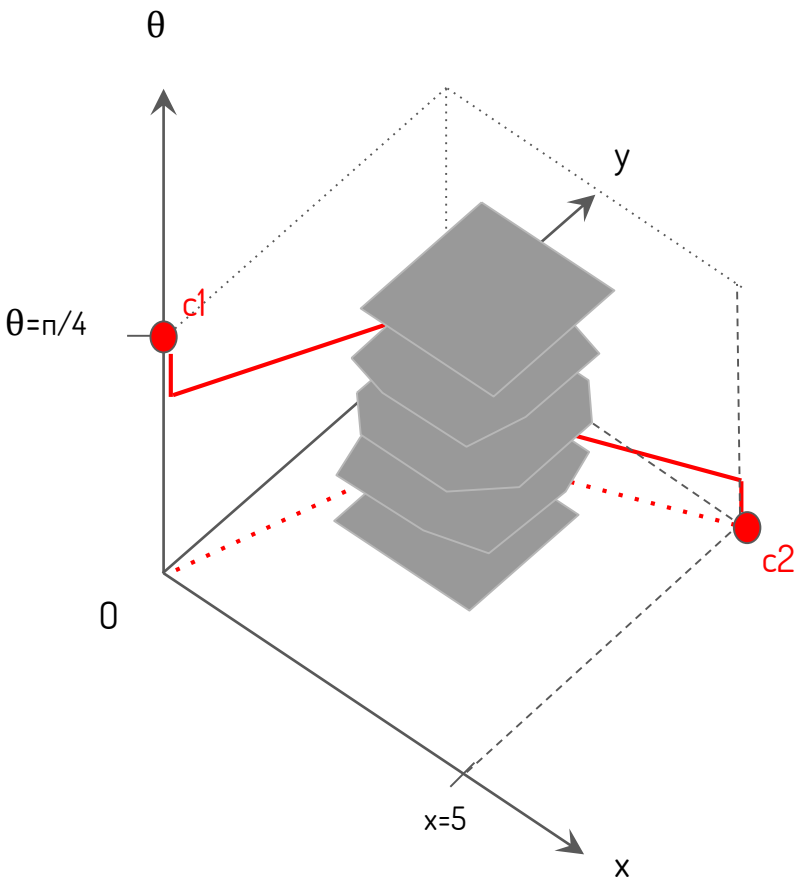
$\sigma(\alpha): [0,1] \rightarrow X$

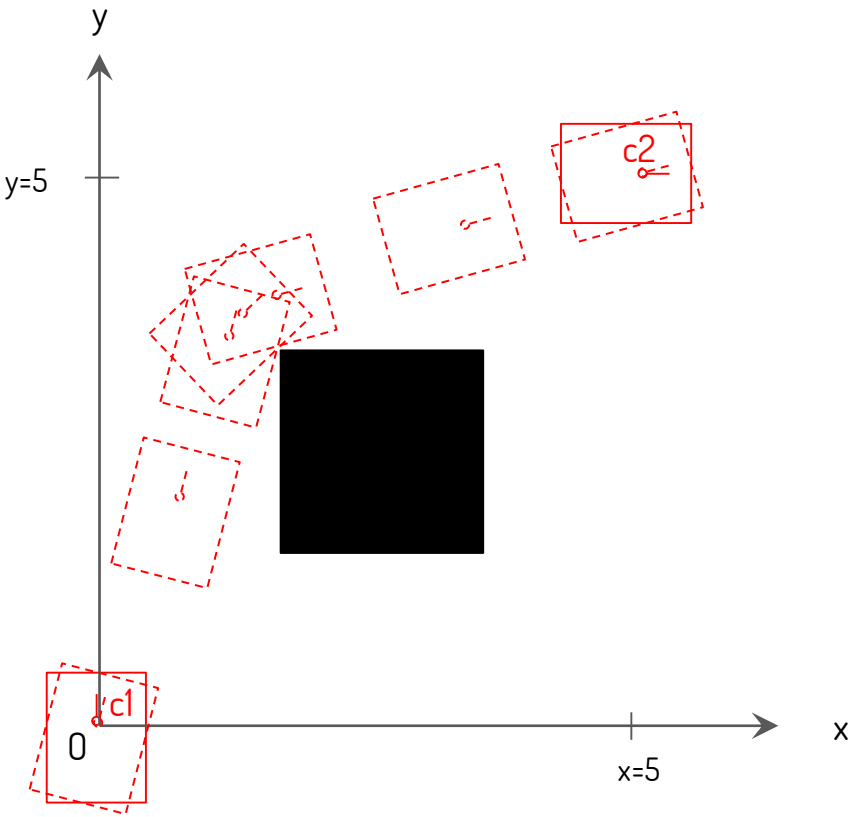




Path in Configuration Space

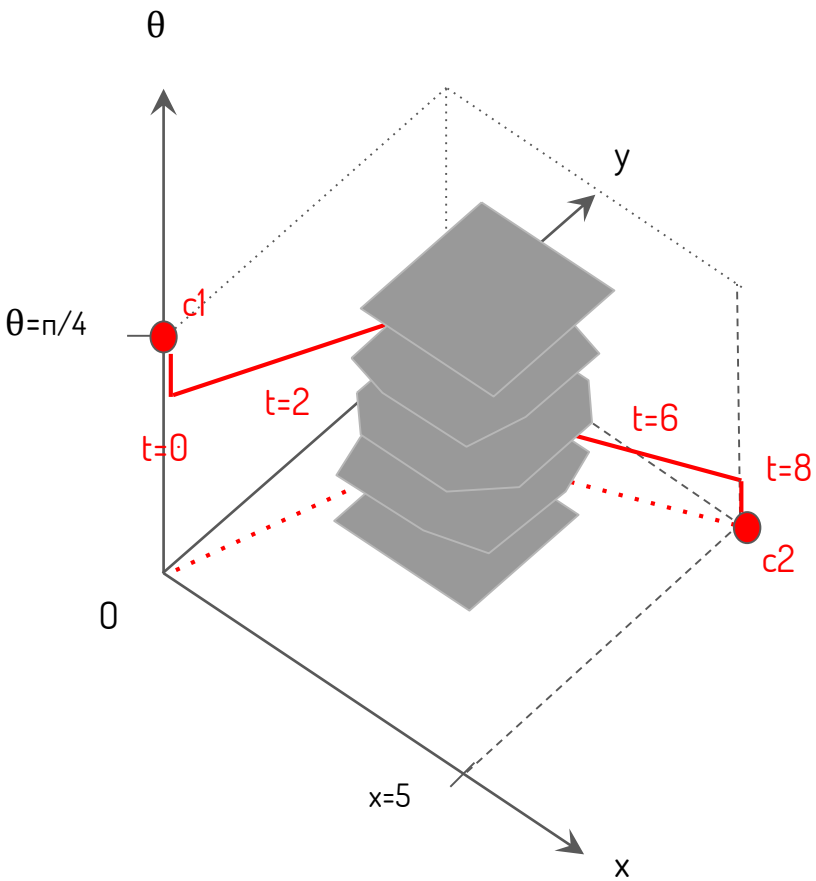
$\sigma(\alpha)$: $[0,1] \rightarrow X$

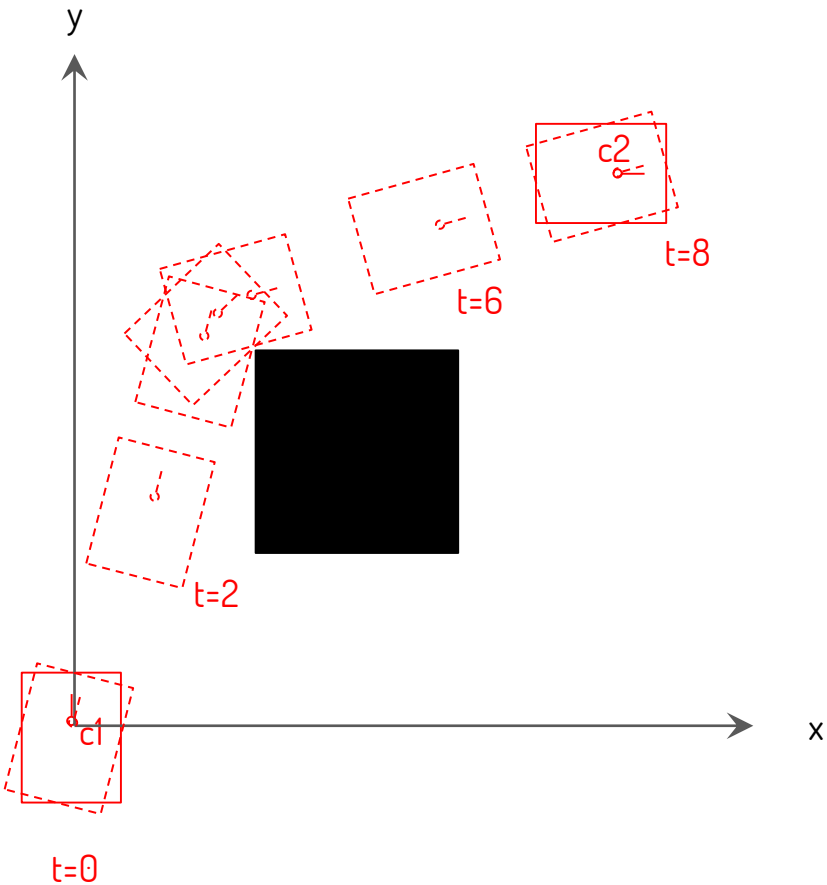




Trajectory in Configuration Space

$\pi(t): [0,T] \rightarrow X$





Path Planning Problem Formulation

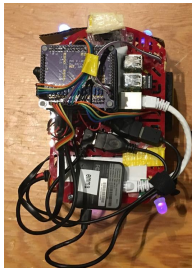
Can be formulated as an optimization problem over all paths in configuration space:

$$\arg\min \; J(\boldsymbol{\sigma}) \text{ subject to}$$
$$\boldsymbol{\sigma}(0) = x_{init}$$
$$\boldsymbol{\sigma}(1) \in X_{goal}$$
$$\boldsymbol{\sigma}(\alpha) \in X_{free} \qquad \forall \alpha \in [0,1]$$

- $\boldsymbol{\sigma}(\alpha)$ is a continuous function $[0,1] \rightarrow X$
- $J(\boldsymbol{\sigma})$ is a cost functional
- x_{init} is the initial configuration of the robot
- X_{goal} is the set of goal configurations
- X_{free} is the free configuration space

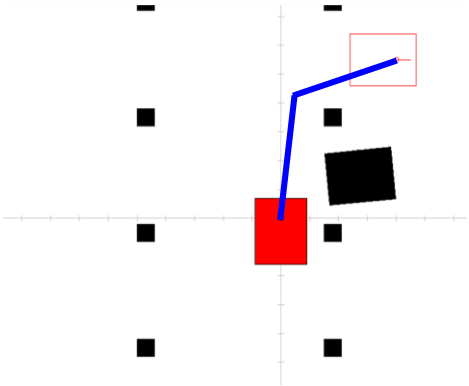
Holonomic System

Holonomic system:
(no differential constraints)



arg min $J(\boldsymbol{\sigma})$ subject to

$$\boldsymbol{\sigma}(0) = x_{init}$$
$$\boldsymbol{\sigma}(1) = X_{goal}$$
$$\boldsymbol{\sigma}(\alpha) = X_{free} \quad \forall \alpha \in [0,1]$$



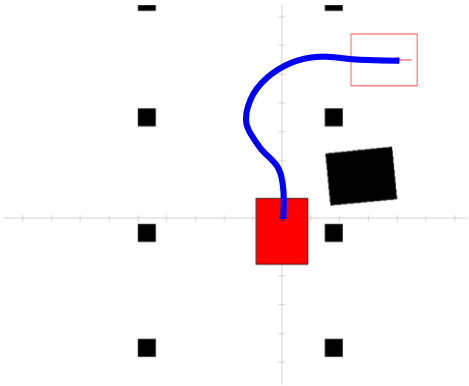
Nonholonomic System

Nonholonomic system:
(differential constraints)

arg min $J(\boldsymbol{\sigma})$ subject to

$$\boldsymbol{\sigma}(0) = x_{init}$$
$$\boldsymbol{\sigma}(1) = X_{goal}$$
$$\boldsymbol{\sigma}(\alpha) = X_{free} \quad \forall \alpha \in [0,1]$$
$$D(\boldsymbol{\sigma}(\alpha), \boldsymbol{\sigma}'(\alpha), \boldsymbol{\sigma}''(\alpha), \ldots) \quad \forall \alpha \in [0,1]$$

E.g., bound on path curvature k can be enforced as $|\boldsymbol{\sigma}'(\alpha)\ \boldsymbol{\sigma}''(\alpha)|/|\boldsymbol{\sigma}'(\alpha)|^3 < k$.



Complexity of Path Planning

- Path planning “Piano Movers problem” is PSPACE-hard [Reif ‘79]
- Complete (non-optimal) algorithms for exist [Canny ‘88] but have running time exponential in dimension of configuration space.

Solution Techniques for Path Planning Problem

- Variational Methods
- Graph-search Methods
  - Cell decomposition
  - Visibility graph
  - Sampling-based roadmap construction
  - Tree of motion primitives
- Incremental Search Methods
  - RRT: Rapidly-exploring Random Trees
  - RRT*: Optimal Rapidly-exploring Random Trees

Properties of Path Planning Methods

An algorithm is

- **Complete:** if it finds valid path or detect non-existence of thereof in finite time.
- **Optimal:** if it find optimal path in finite time.

- **Anytime:** if it can be terminated at any point of the execution, but the path quality improves with computation time

- **Probabilistically Complete:** if the probability that the algorithm finds valid solution goes to 1 with running time.
- **Asymptotic Optimal:** if it returns a sequence of solutions converging to an optimal solution.

Trajectory Planning Problem Formulation

Useful for

- Dynamic constraints
- Dynamic obstacles

Can be formulated as optimization in the space of trajectories over time interval [0,T] in configuration space:

$$
\begin{aligned}
\arg\min \; & J(\pi) \text{ subject to} \\
& \pi(0) = x_{init} \\
& \pi(T) \in X_{goal} \\
& \pi(t) \in X_{free}(t) \qquad \forall t \in [0,T] \\
& D(\pi(t), \pi'(t), \pi''(t), \ldots) \qquad \forall t \in [0,T]
\end{aligned}
$$

Solution Techniques for Trajectory Planning Problem

- Variational Methods
- Convert to Path Planning in Space-Time:

trajectory planning in (x,y,θ)

=>

path planning in (x,y,θ,t) + diff. constraints

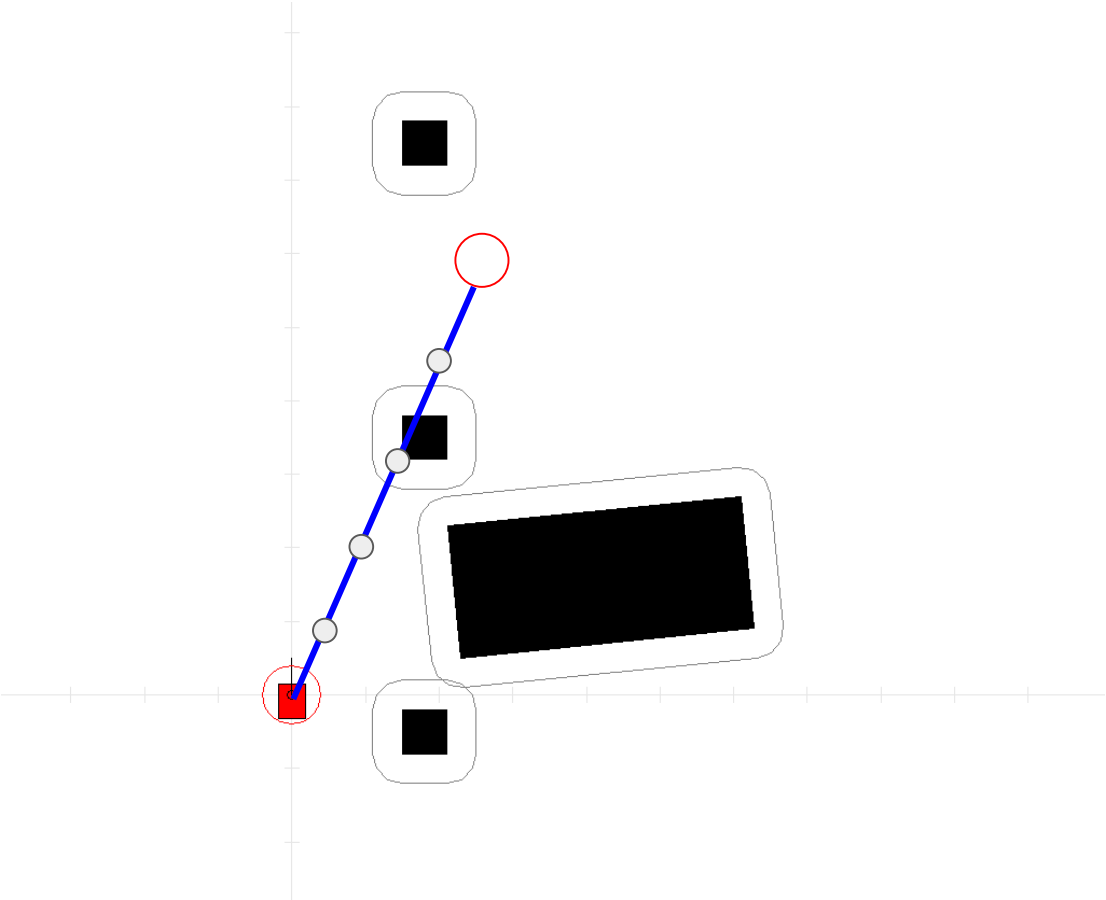
Solution Techniques for Path Planning Problem

- Variational Methods
- Graph-search Methods
  - Cell decomposition
  - Visibility graph
  - Sampling-based roadmap construction
  - Tree of motion primitives
- Incremental Search Methods
  - RRT: Rapidly-exploring Random Trees
  - RRT*: Optimal Rapidly-exploring Random Trees

Variational Techniques

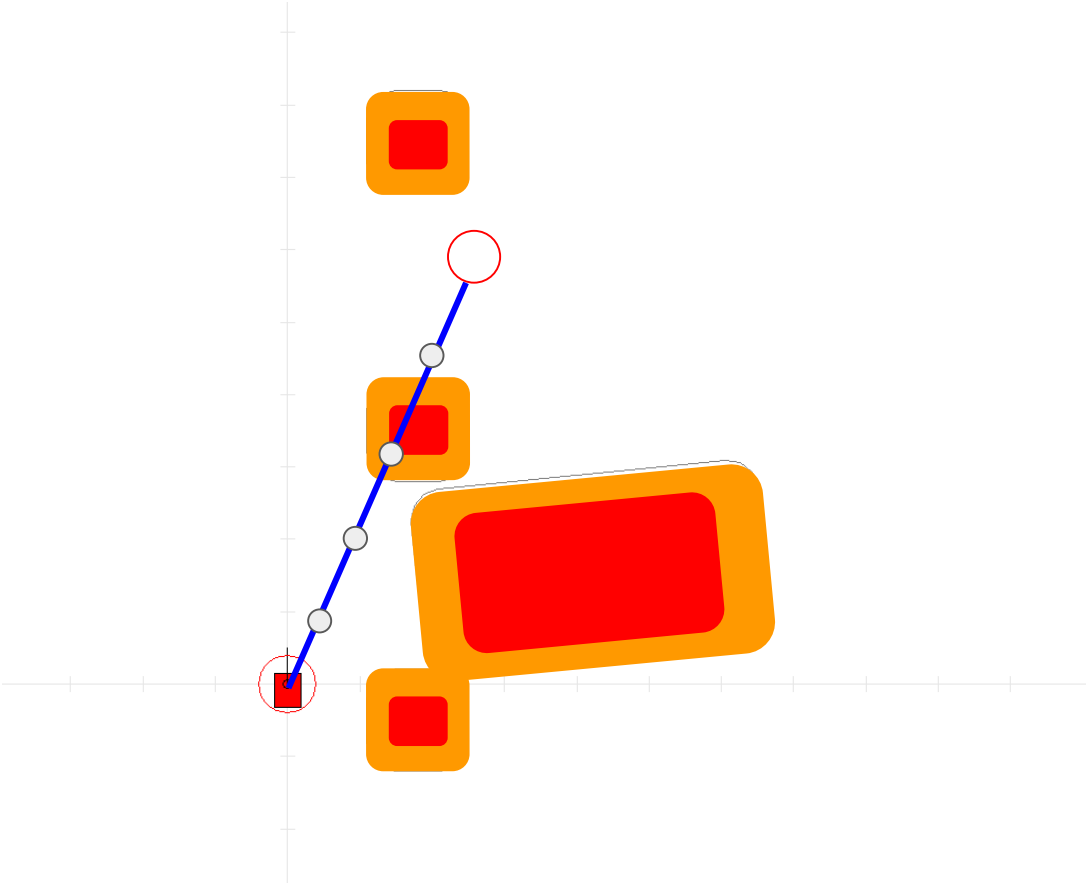
Aka Trajectory optimization, Optimal control, etc.

- Non-linear optimization
- Path is represented as a spline
- Position of control points is optimized



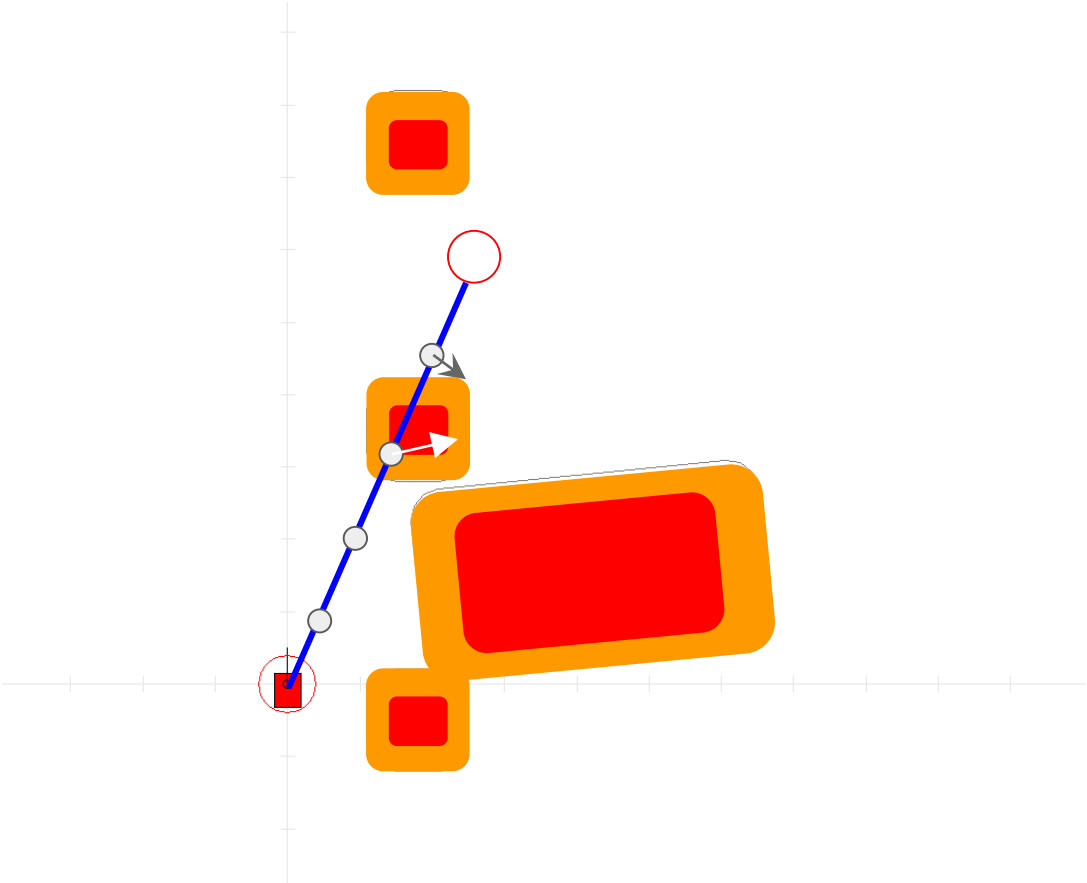
Variational Techniques

Obstacles are modelled as high-cost regions



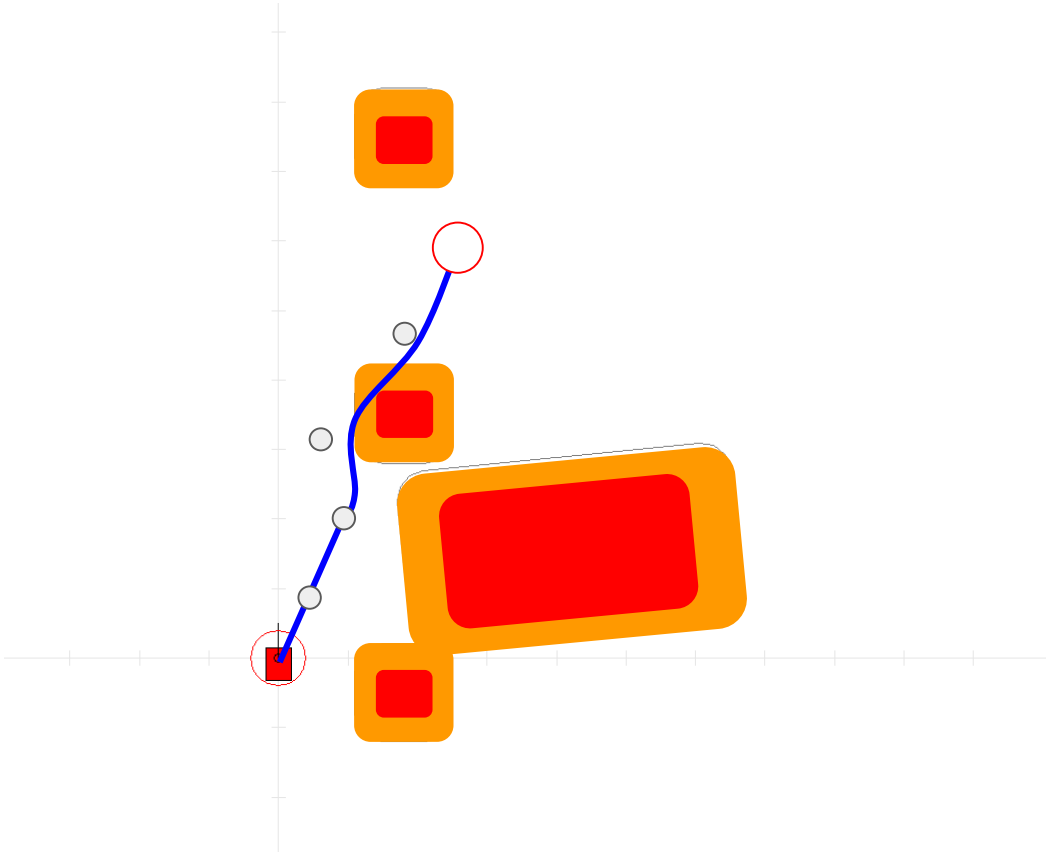
Variational Techniques

Find gradient



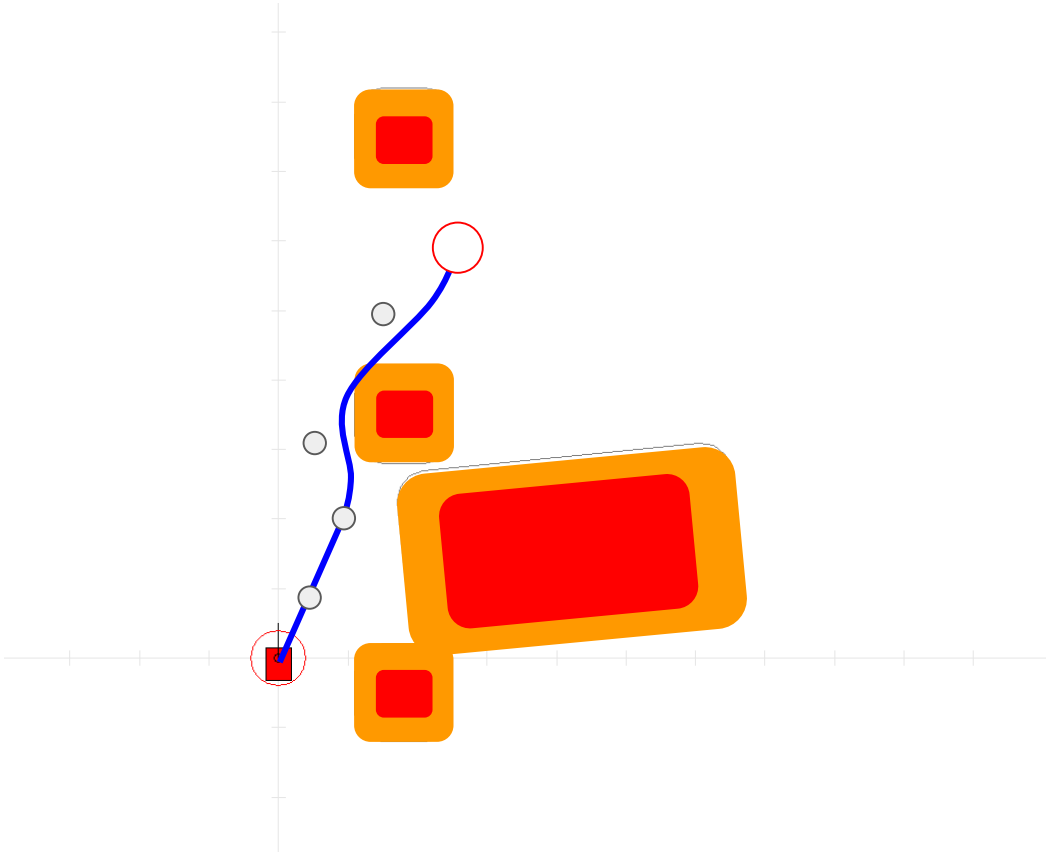
Variational Techniques

Move the control points in the direction of negative gradient

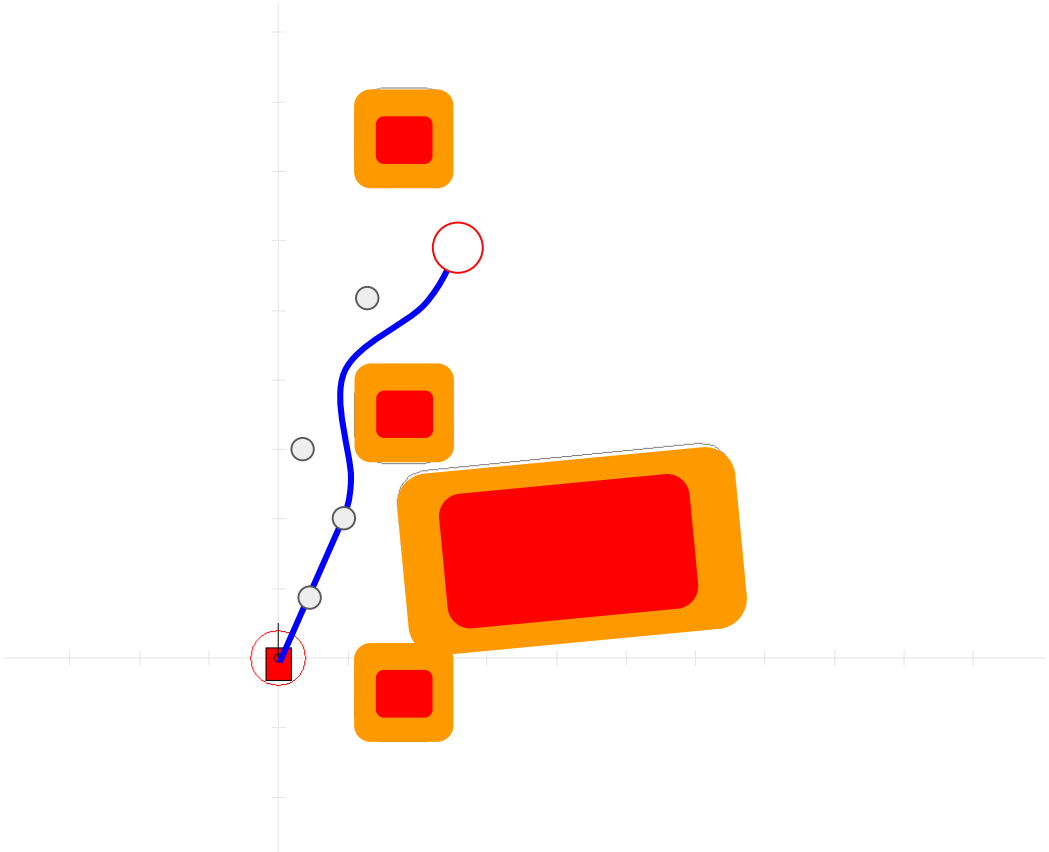


Variational Techniques

Repeat until convergence



Variational Techniques



Variational Techniques

Pros:

- Efficient
- Widely applicable

Cons:

- Only local convergence
    - Incomplete
    - Locally optimal

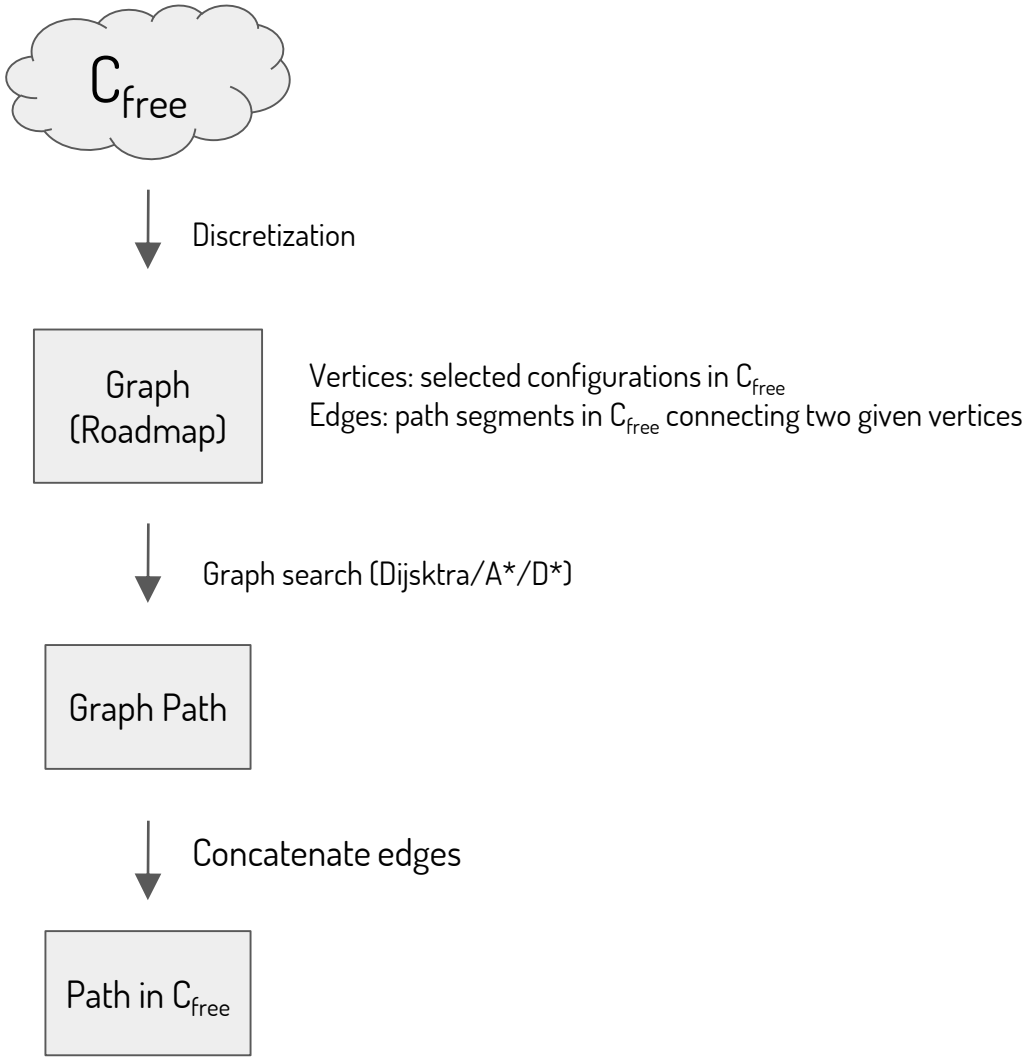
Notes:

- Used in for local path optimization within CMU's car 'Boss'.

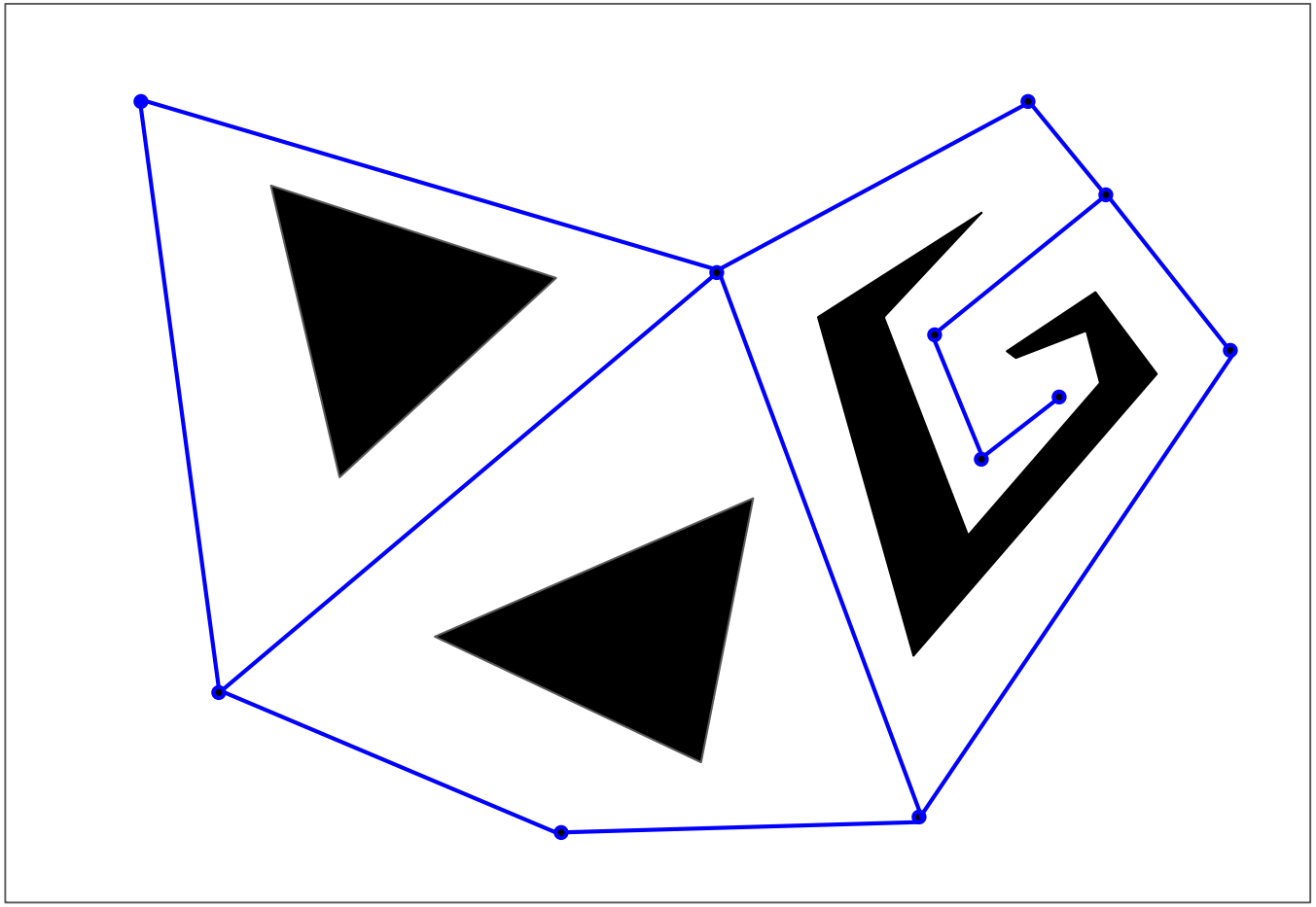
Solution Techniques for Path Planning Problem

- Variational Methods
- <u>Graph-search Methods</u>
  - Cell decomposition
  - Visibility graph
  - Sampling-based roadmap construction
  - Tree of motion primitives
- Incremental Search Methods
  - RRT: Rapidly-exploring Random Trees
  - RRT*: Optimal Rapidly-exploring Random Trees

Graph-based Methods



Roadmap

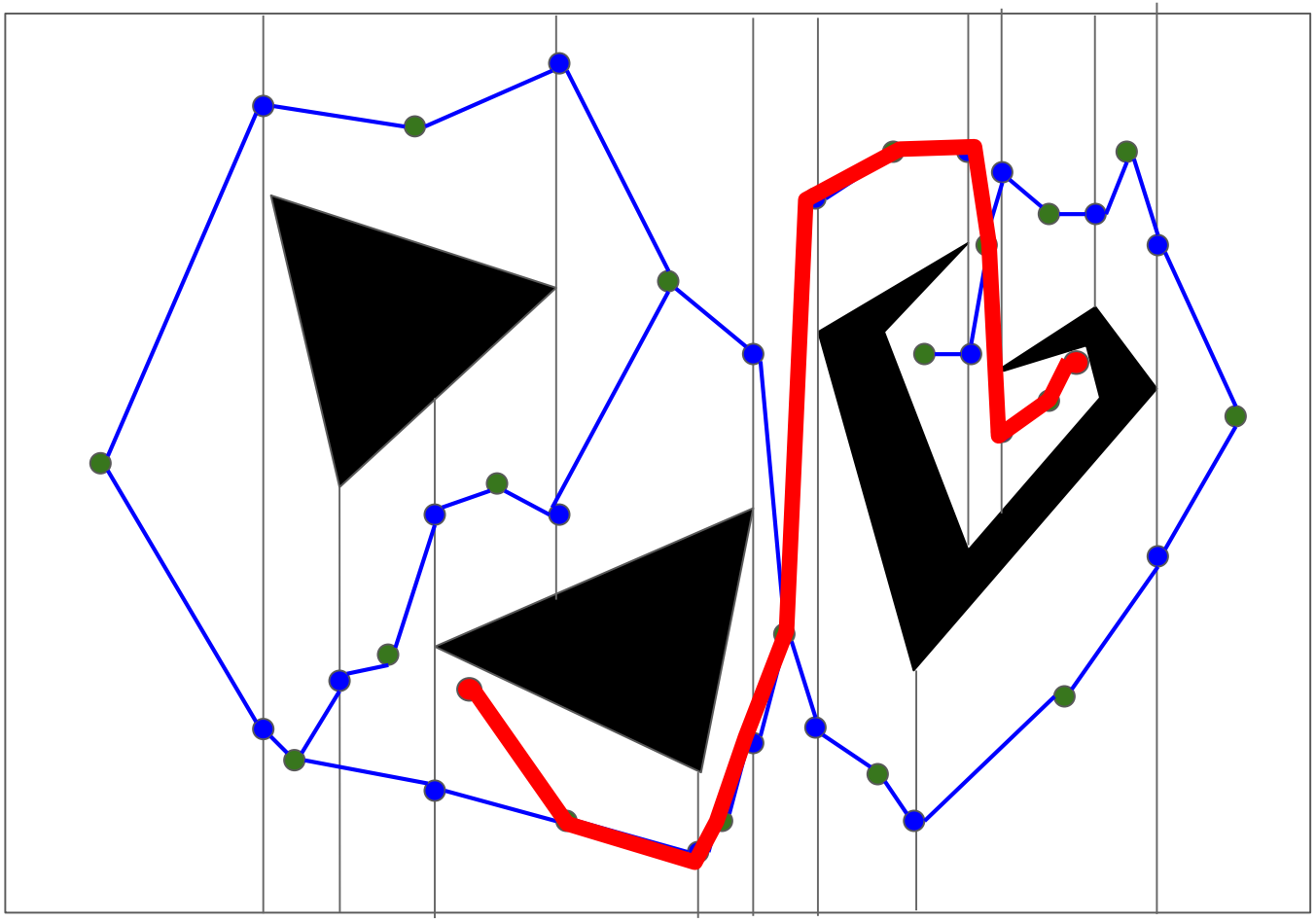


Solution Techniques for Path Planning Problem

- Variational Methods
- Graph-search Methods
  - <u>Cell decomposition</u>
  - Visibility graph
  - Sampling-based roadmap construction
  - Tree of motion primitives
- Incremental Search Methods
  - RRT: Rapidly-exploring Random Trees
  - RRT*: Optimal Rapidly-exploring Random Trees

Cell Decomposition

- A method for building roadmaps in 2d polygonal environments



Cell Decomposition

Pros:

- Complete
- Generalizes to higher dimensions and beyond polygonal models

Cons:

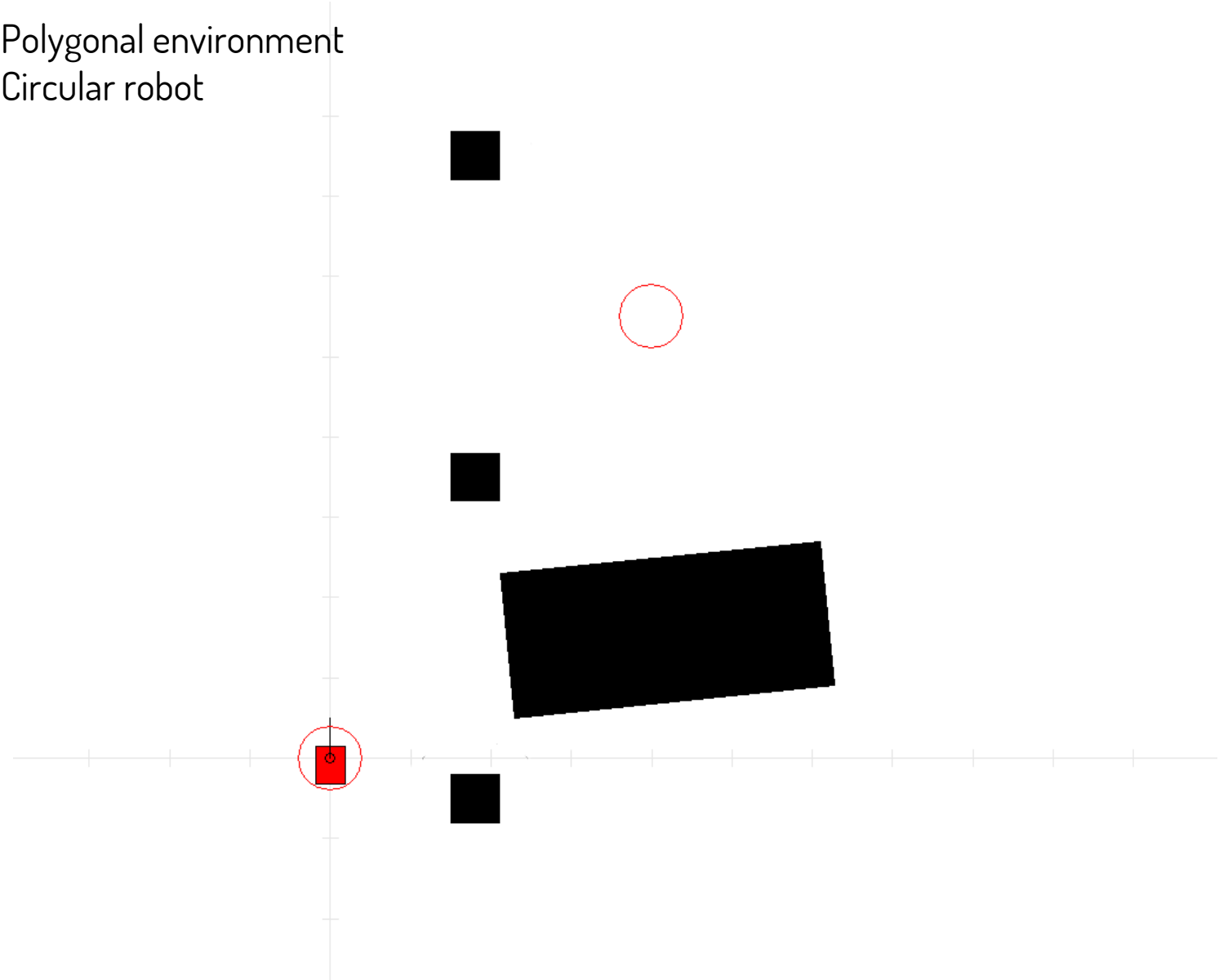
- Only holonomic systems
- Suboptimal

Solution Techniques for Path Planning Problem

- Variational Methods
- Graph-search Methods
  - Cell decomposition
  - <u>Visibility graph</u>
  - Sampling-based roadmap construction
  - Tree of motion primitives
- Incremental Search Methods
  - RRT: Rapidly-exploring Random Trees
  - RRT*: Optimal Rapidly-exploring Random Trees

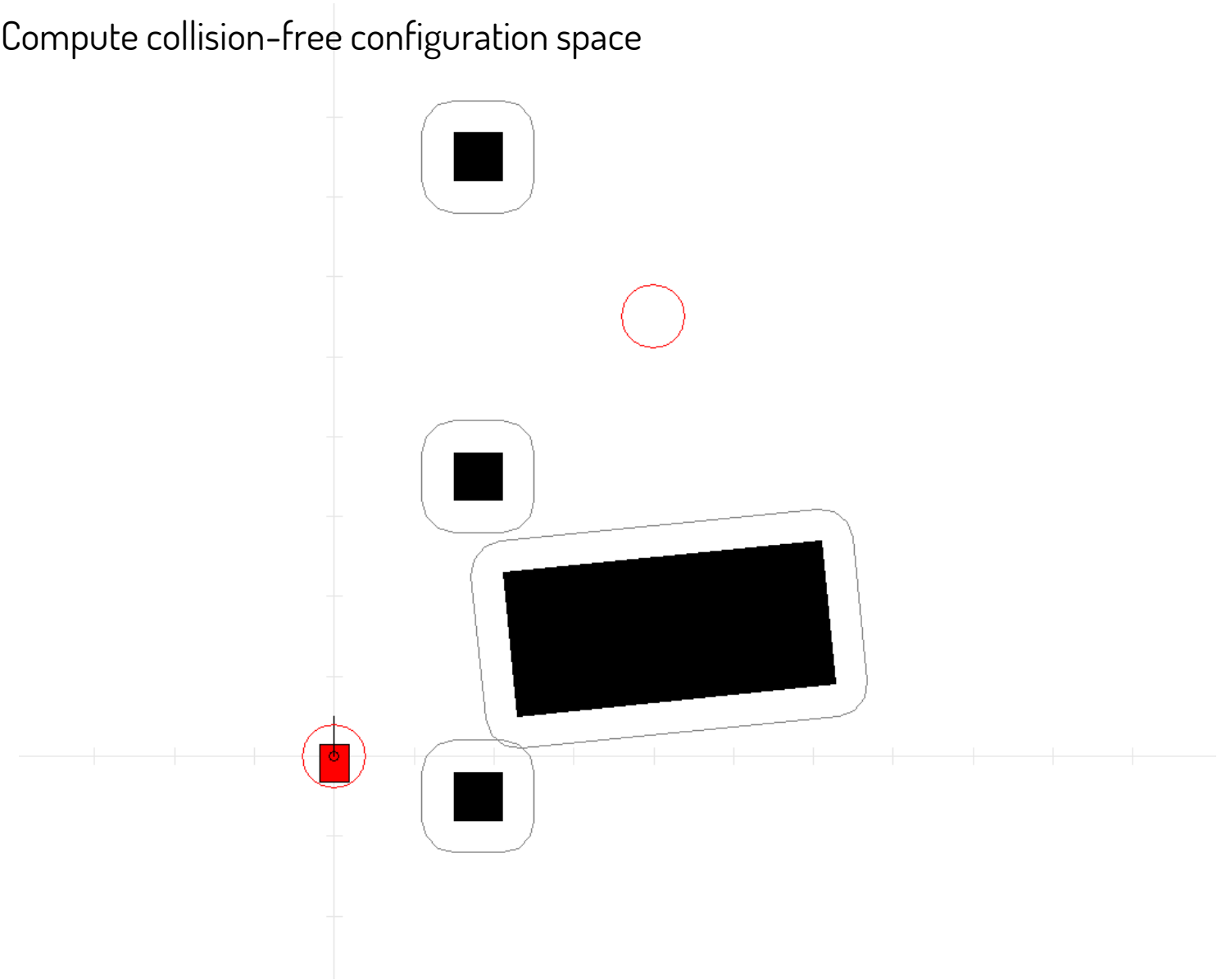
Visibility Graph

- Polygonal environment
- Circular robot



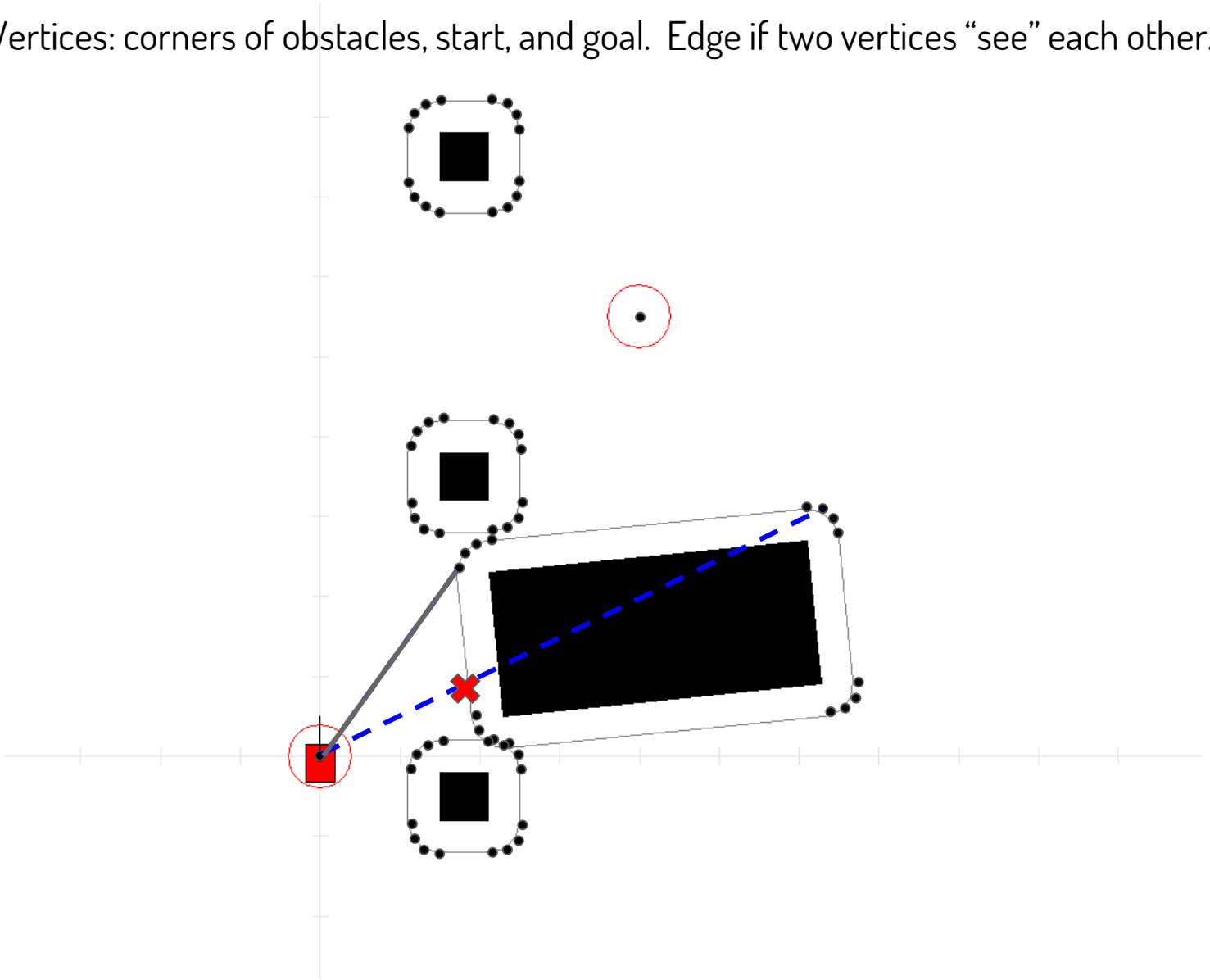
Visibility Graph

- Compute collision-free configuration space



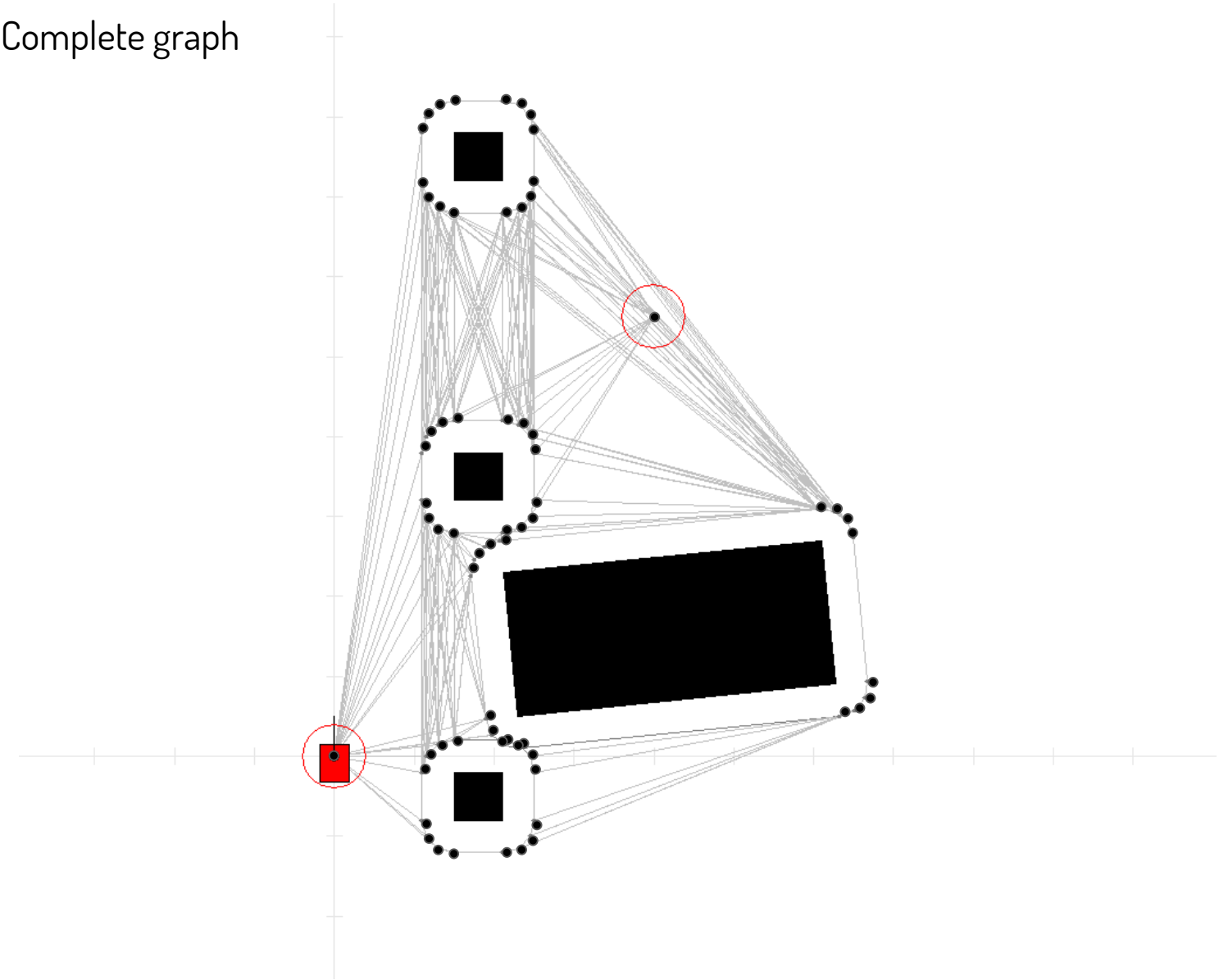
Visibility Graph

- Vertices: corners of obstacles, start, and goal. Edge if two vertices “see” each other.



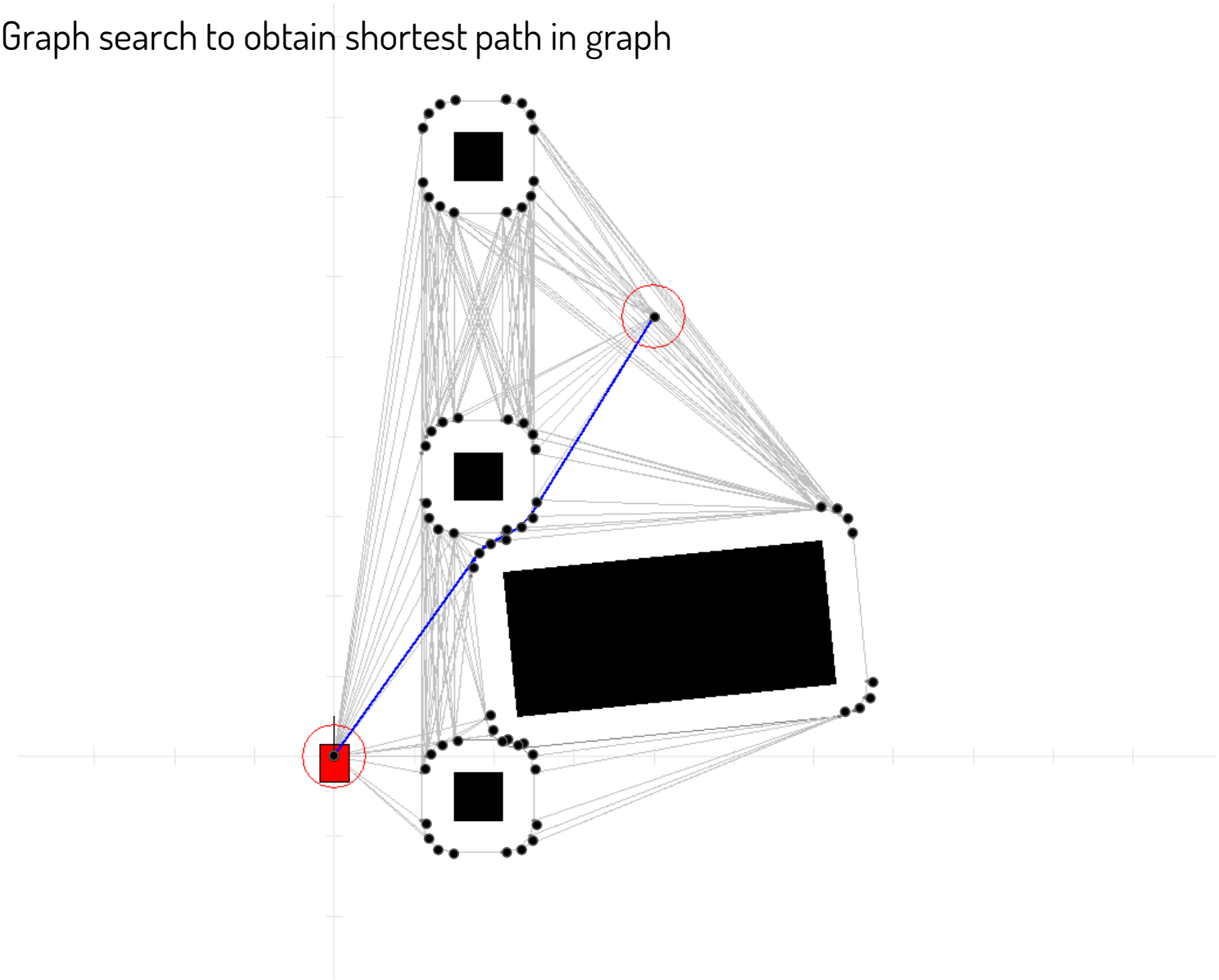
Visibility Graph

- Complete graph



Visibility Graph

- Graph search to obtain shortest path in graph



Visibility Graph

Pros:

- Efficient: $O(n^2)$
- Exact optimal

Cons:

- Optimality guarantee only for 2-d environments and circular robot
- Only for holonomic systems and polygonal models

Solution Techniques for Path Planning Problem

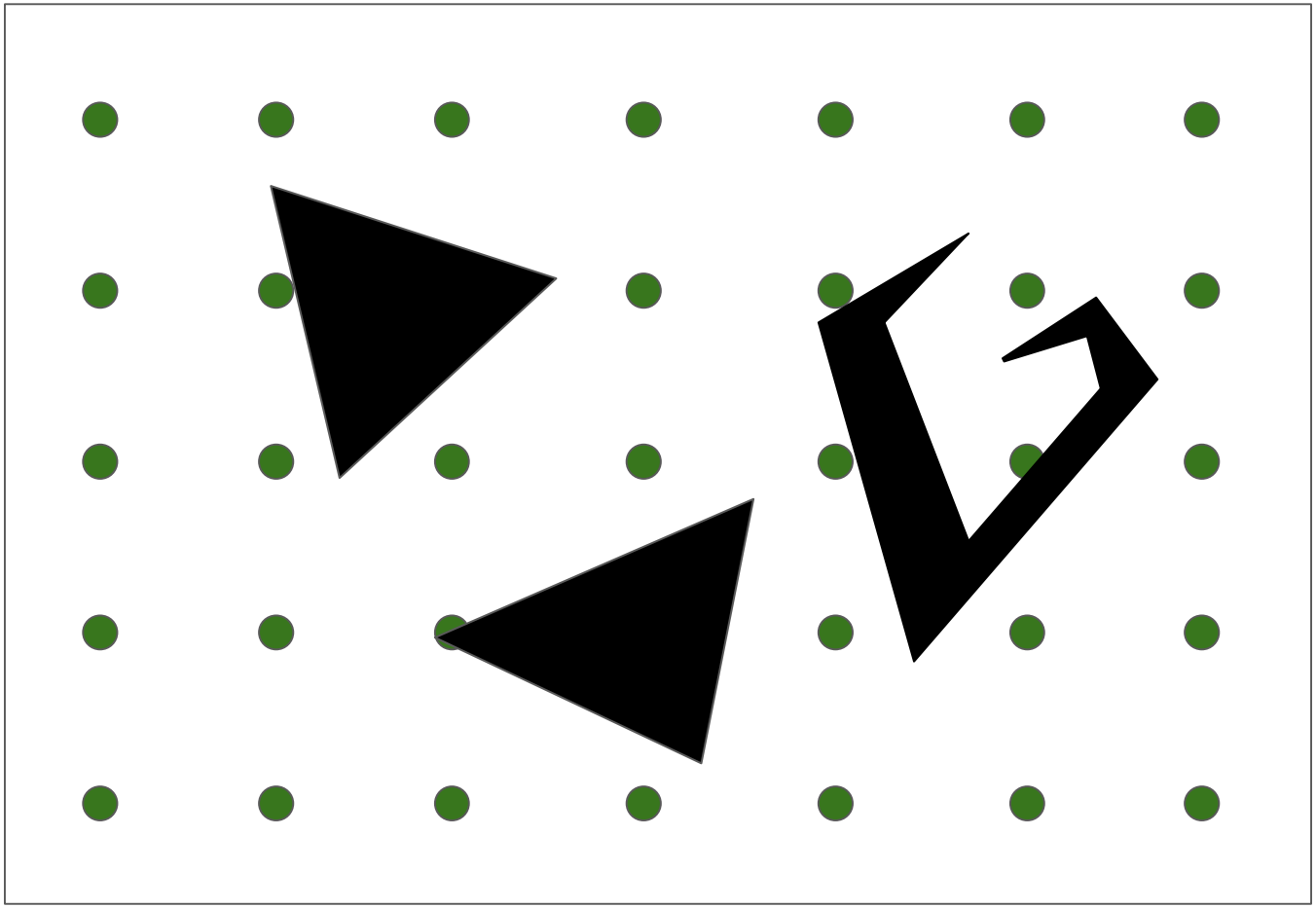
- Variational Methods
- Graph-search Methods
  - Cell decomposition
  - Visibility graph
  - <u>Sampling-based roadmap construction</u>
  - Tree of motion primitives
- Incremental Search Methods
  - RRT: Rapidly-exploring Random Trees
  - RRT*: Optimal Rapidly-exploring Random Trees

Sampling-based Roadmap Construction

Generate roadmap by:

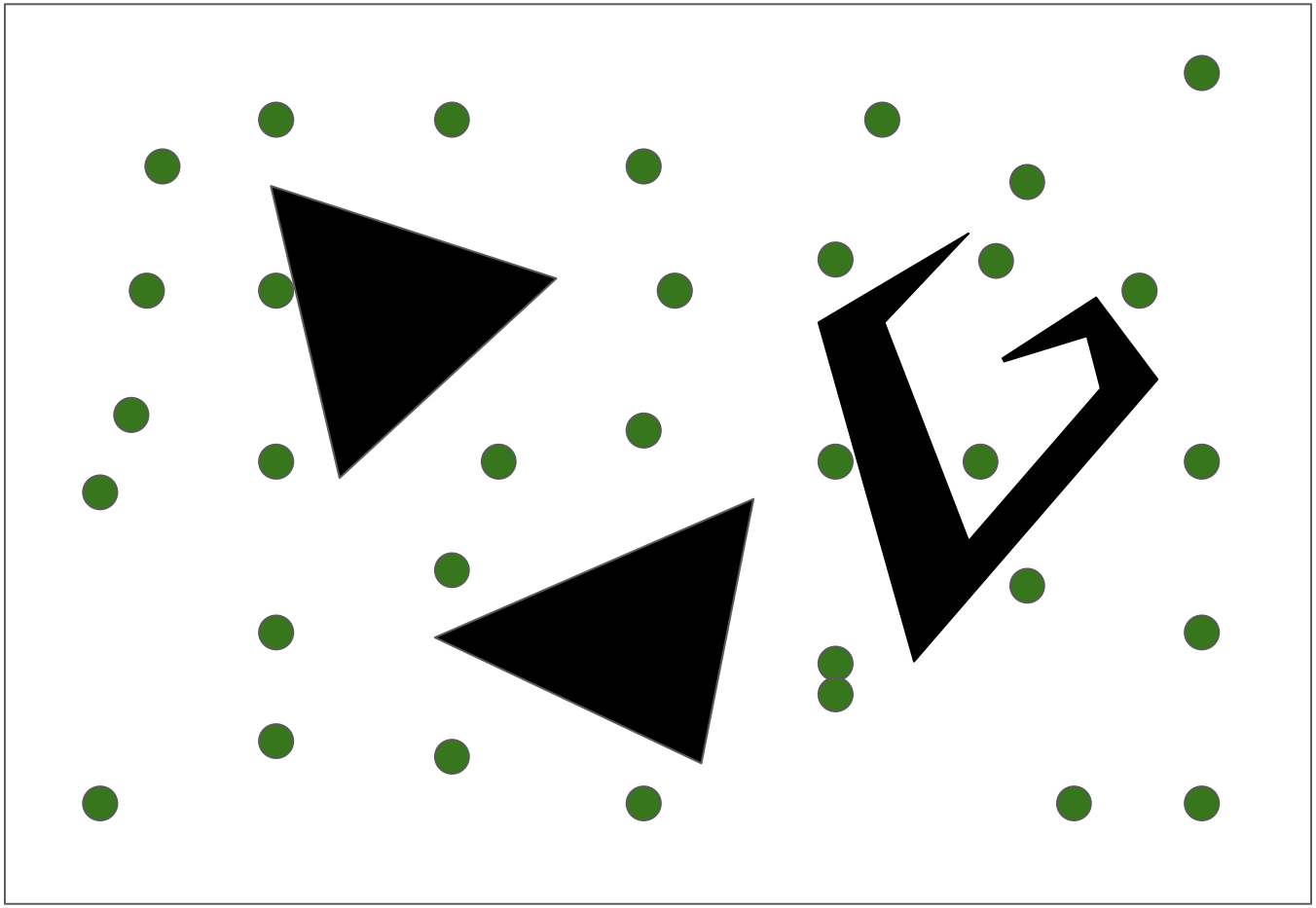
- 1. Sampling the free configuration space
  - Deterministically
  - Randomly
- 2. Connecting nearby samples
  - All neighbors closer than distance r
  - K-nearest neighbors

Sampling-based Roadmap Construction



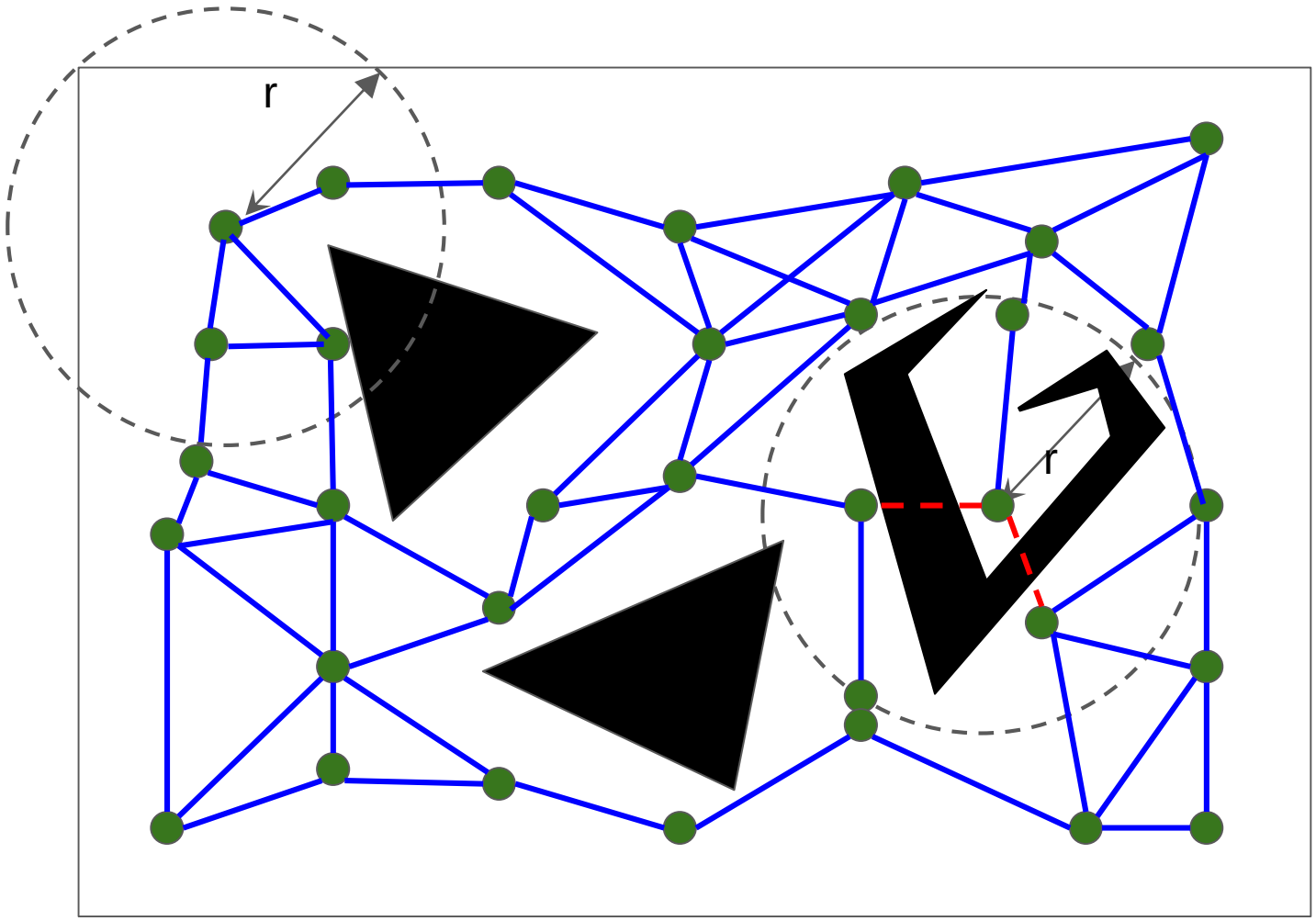
Deterministic Sampling (Sukharev Grid)

Sampling-based Roadmap Construction



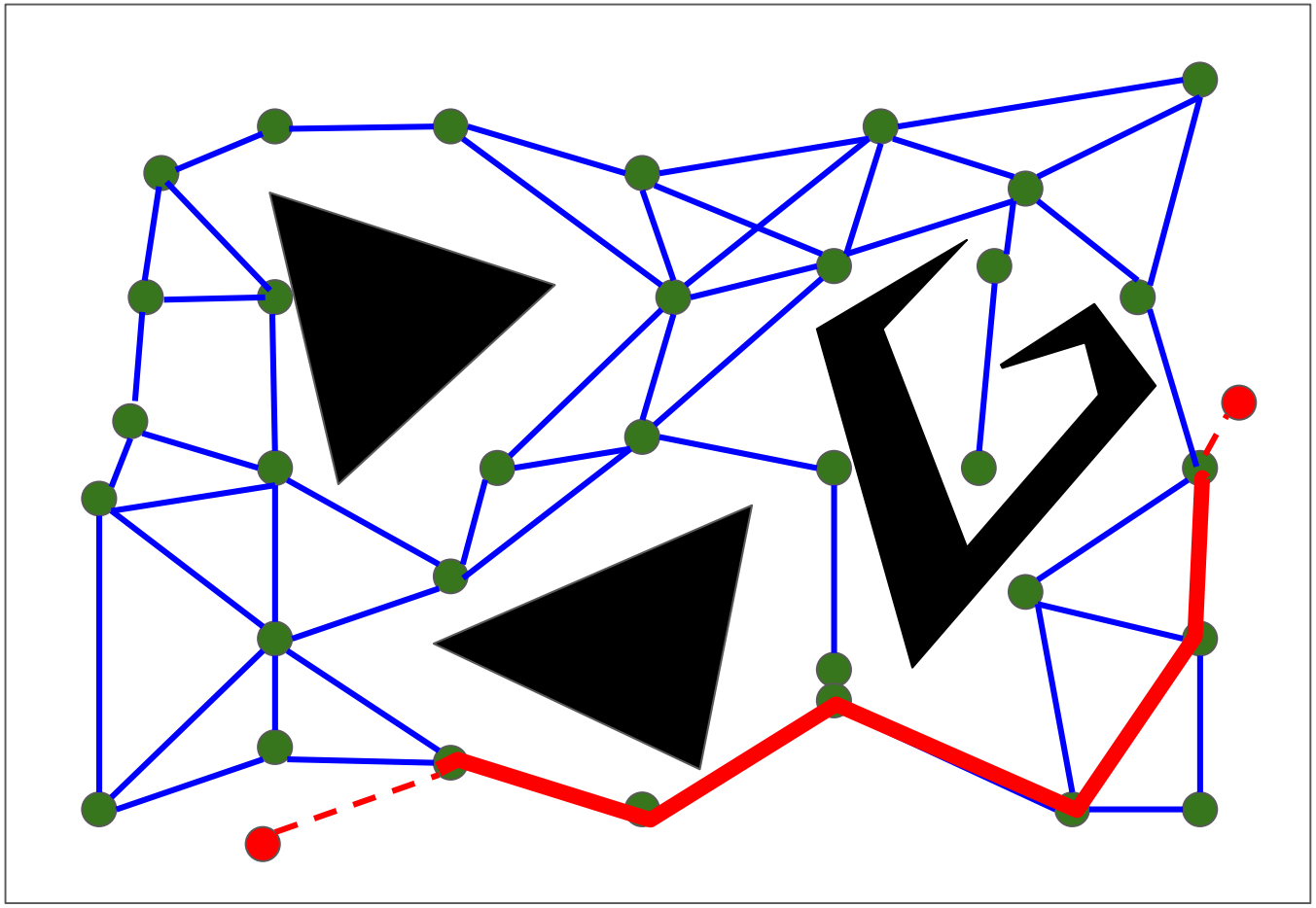
Probabilistic Roadmap

Sampling-based Roadmap Construction



Connecting Samples

Sampling-based Roadmap Construction



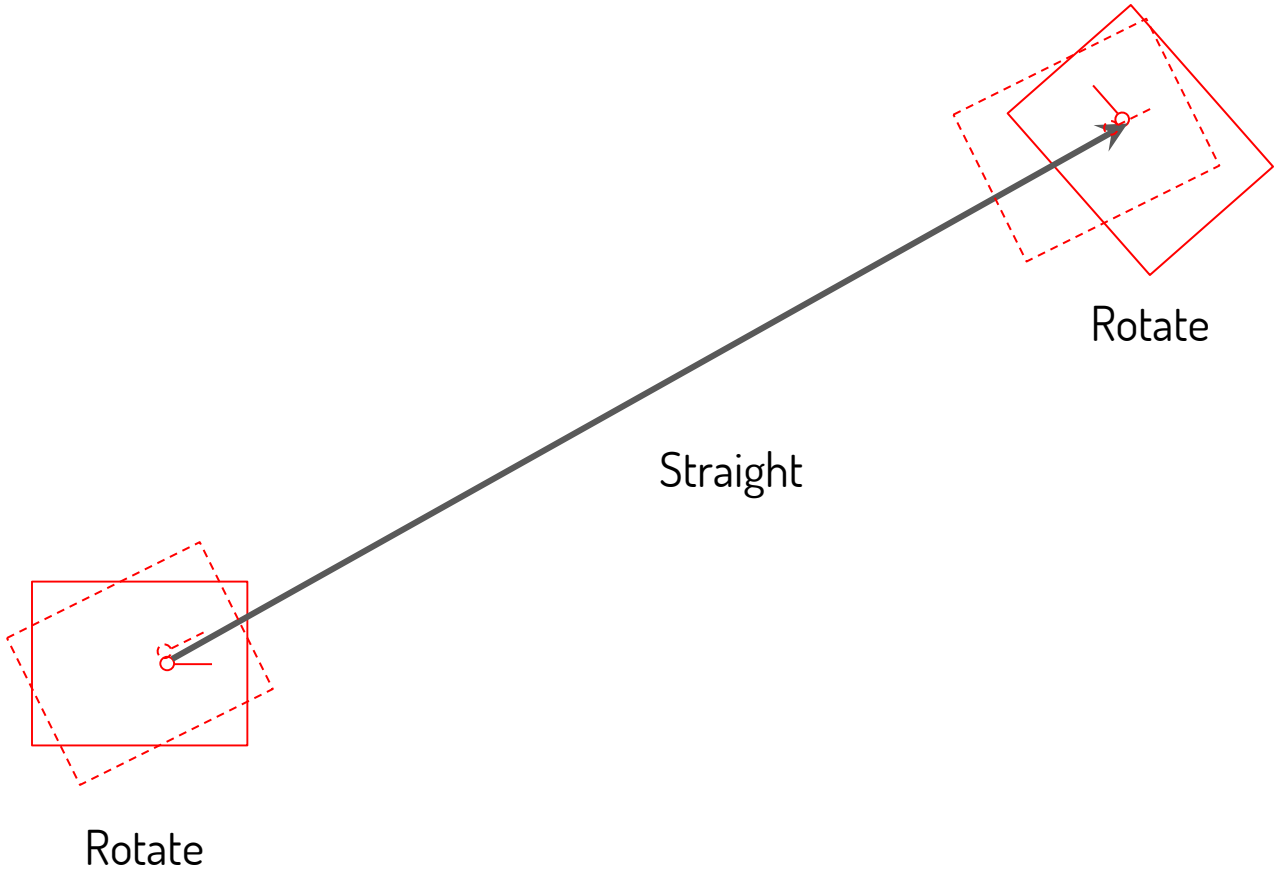
Finding Path

Steering: Connecting the Samples

Sampling-based methods rely on steering function.

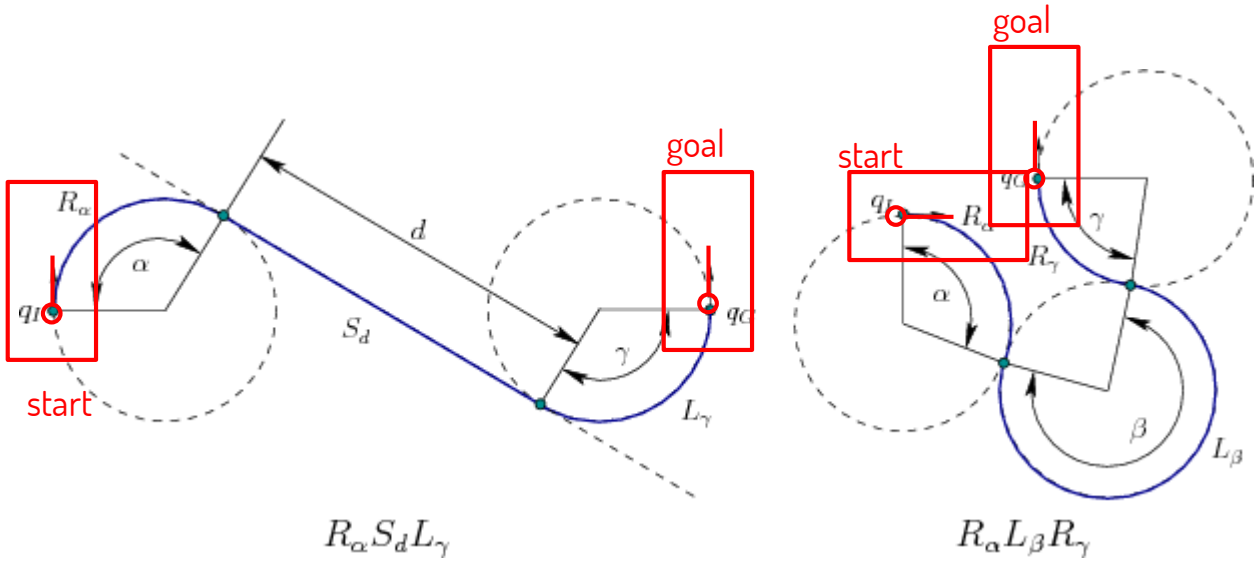
- Steer(**x**,**y**) returns a feasible path segment between configuration **x** and **y**.
- Steering function respects kinematic and dynamic constraints, but **does not consider obstacles.**
- Often obtained by simulating a dynamic model of the vehicle.

Steering for Duckiebot



Dubins Path: Steering for vehicle moving forward

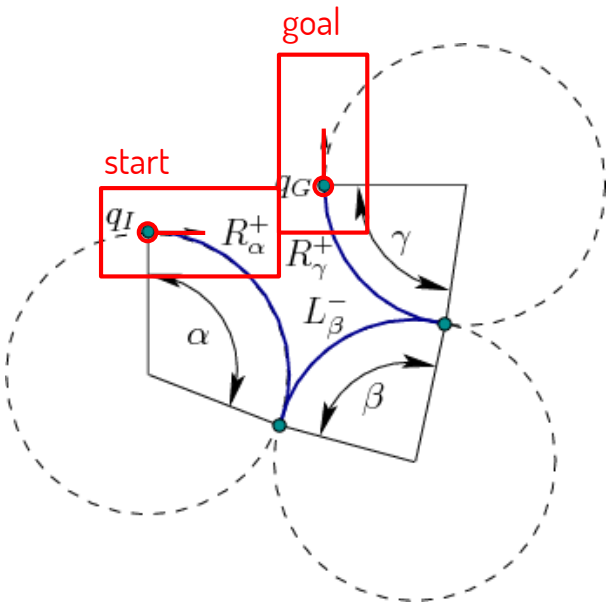
- Car that moves only forward.
- The shortest path for can be computed analytically
- It consist of three path segments: sharpest-possible turn left (L), right (R) or straight (S).
- Total six templates: {LRL, RLR, LSL, LSR, RSL, RSR}



Picture by Steven LaValle.

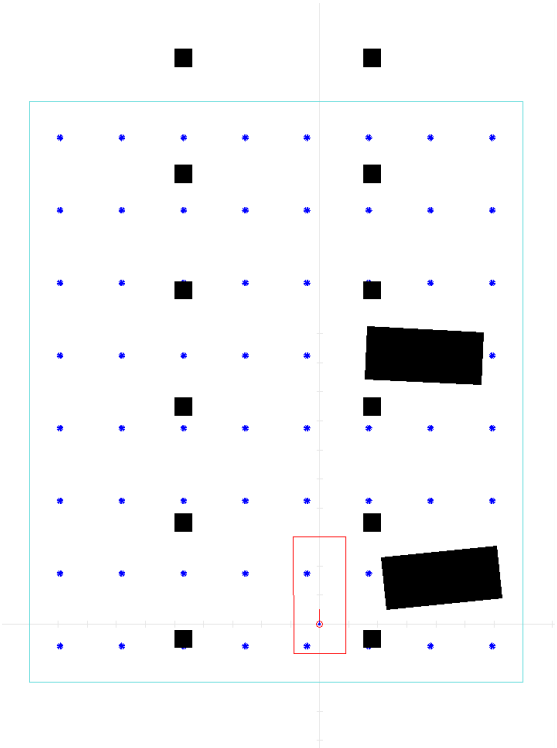
Reeds-Shepp Path: Steering for Car Moving Forwards and Backwards.

- Car that can move both forward and backwards.
- Up to 5 segments: {R+,R-, L+, L-, S+, S-}.
- 46 templates.

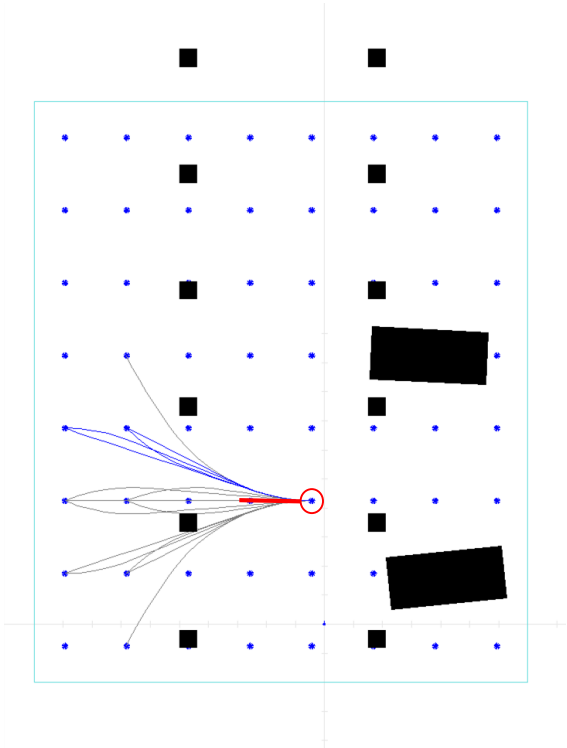


Picture by Steven LaValle.

Sampling-based Roadmap with Dubins Path

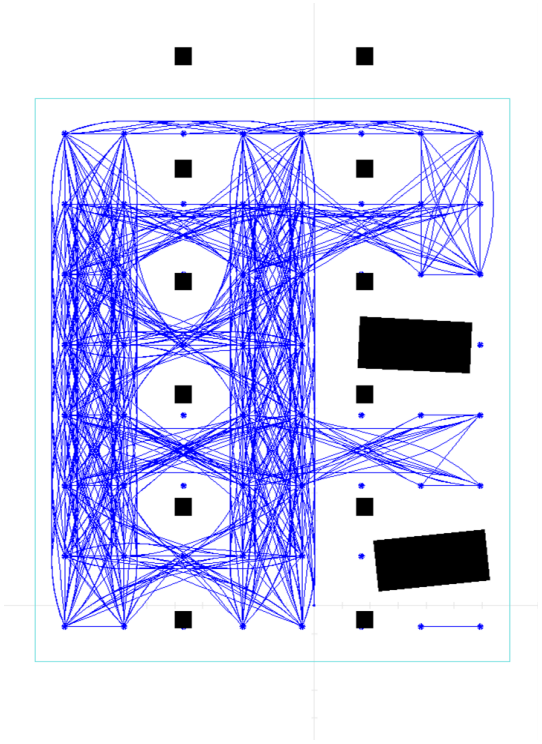


Sample the configuration space.
Here 8 x 8 x 16 = 1024 regular samples.

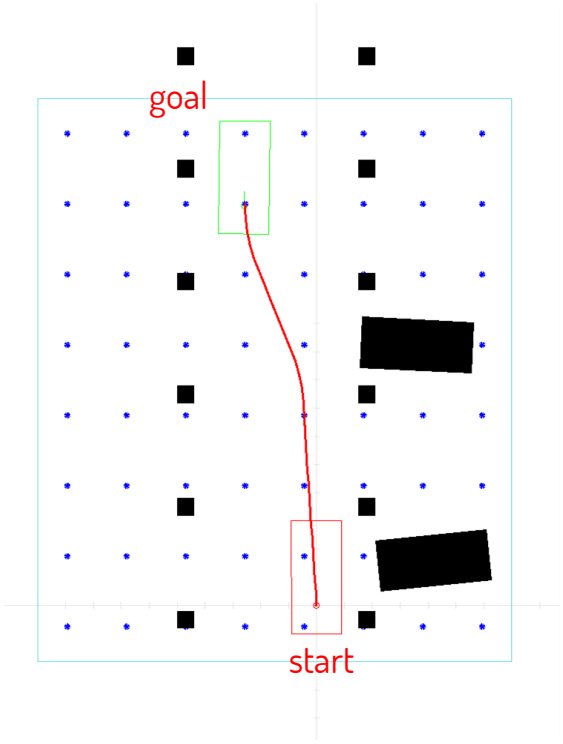


For each sampled configuration, connect neighbors closer than 6m in Dubins distance.
Blue are collision-free connections.

Sampling-based Roadmap Construction



Resulting roadmap



Graph searched using Dijkstra/A*, we obtain a feasible path for the vehicle.

Choosing Connection Radius

- How to choose connection radius?
  - too small: roadmap will be disconnected
  - Too large: to many computationally intensive

- PRM* [Karaman 2011]: for asymptotic optimality, chose connection radius as a function of number of samples of graph:

$$r = \gamma \sqrt[d]{(\log n)/n}$$

  - O(log n) connections attempted at each iteration
  - Maintains asymptotic optimality with O(n log n) complexity

Sampling-based Roadmap Construction

Pros:

- Handle differential constraints
- Model agnostic
- Multiquery
- PRM/PRM* - asymptotic optimality guarantee

Cons:

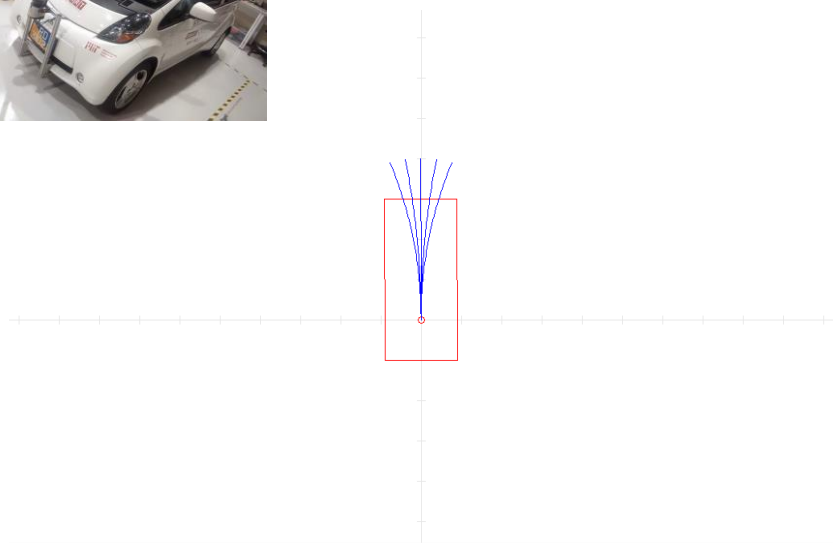
- Completeness and optimality achieved only up to discretization resolution
- Need exact steering

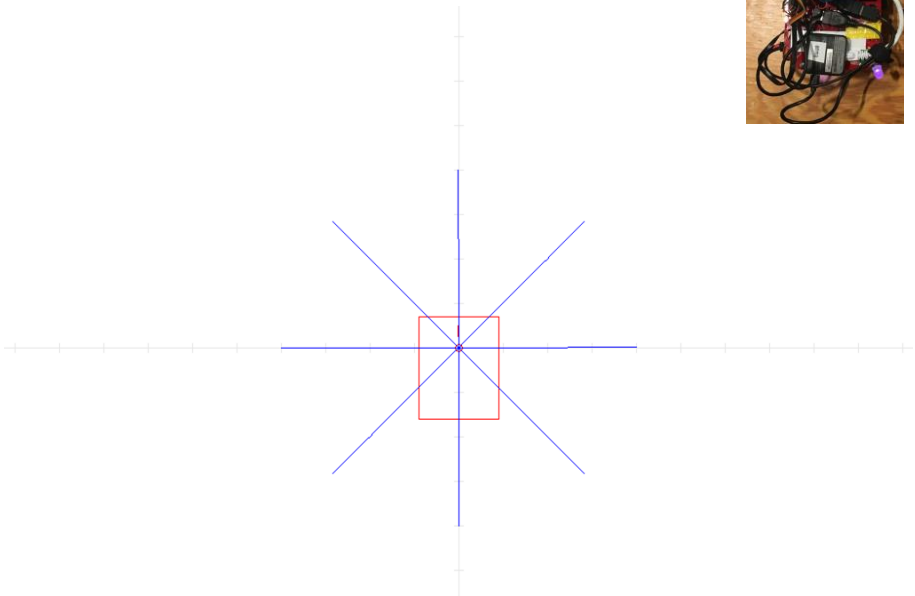
Solution Techniques for Path Planning Problem

- Variational Methods
- Graph-search Methods
  - Cell decomposition
  - Visibility graph
  - Sampling-based roadmap construction
  - Tree of motion primitives
- Incremental Search Methods
  - RRT: Rapidly-exploring Random Trees
  - RRT*: Optimal Rapidly-exploring Random Trees

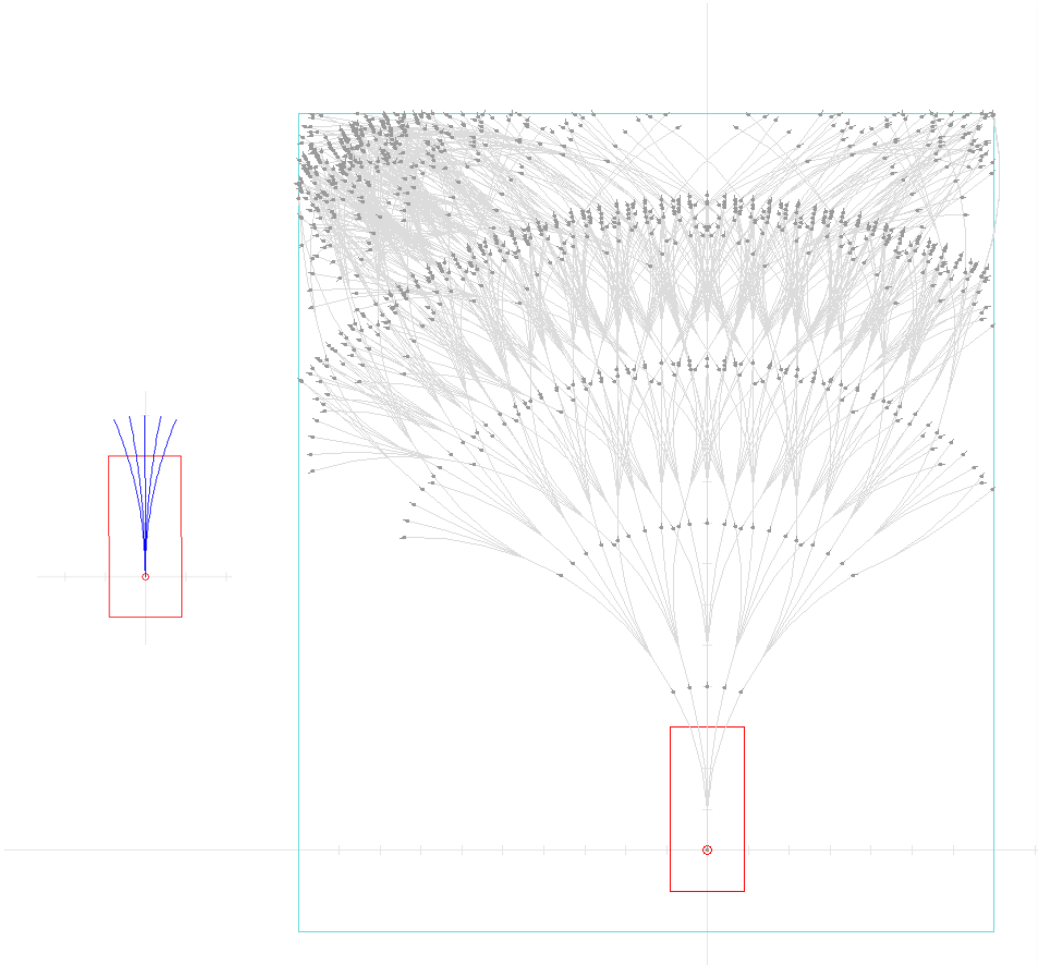
Motion Primitives

A discrete set of of maneuvers that the vehicle can execute from each configuration:

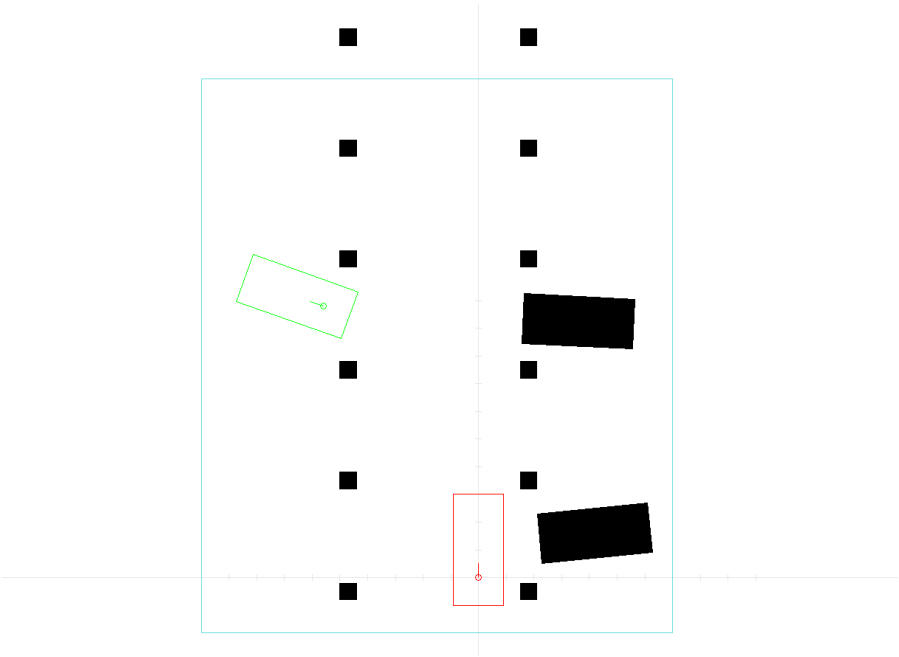




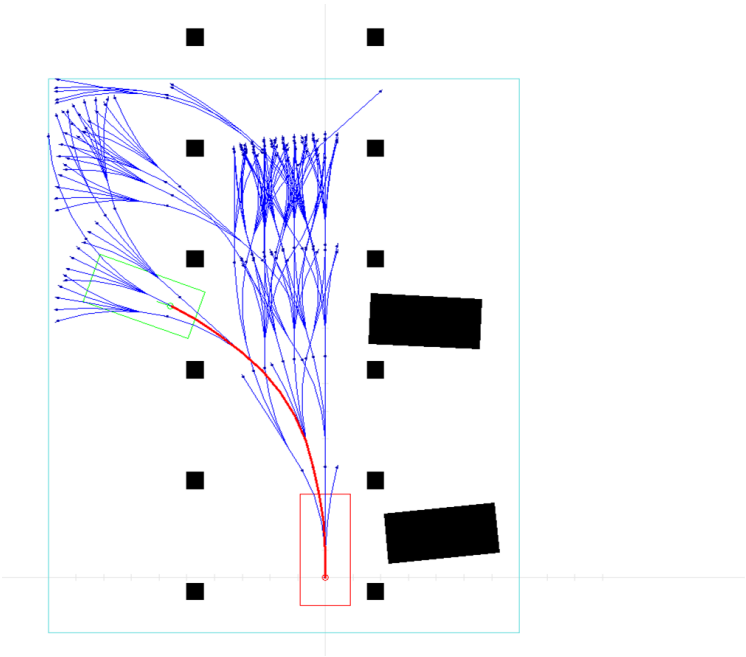
Recursive Application of Motion Primitives



Recursive Application of Motion Primitives



Start expanding motion primitives from current configuration.

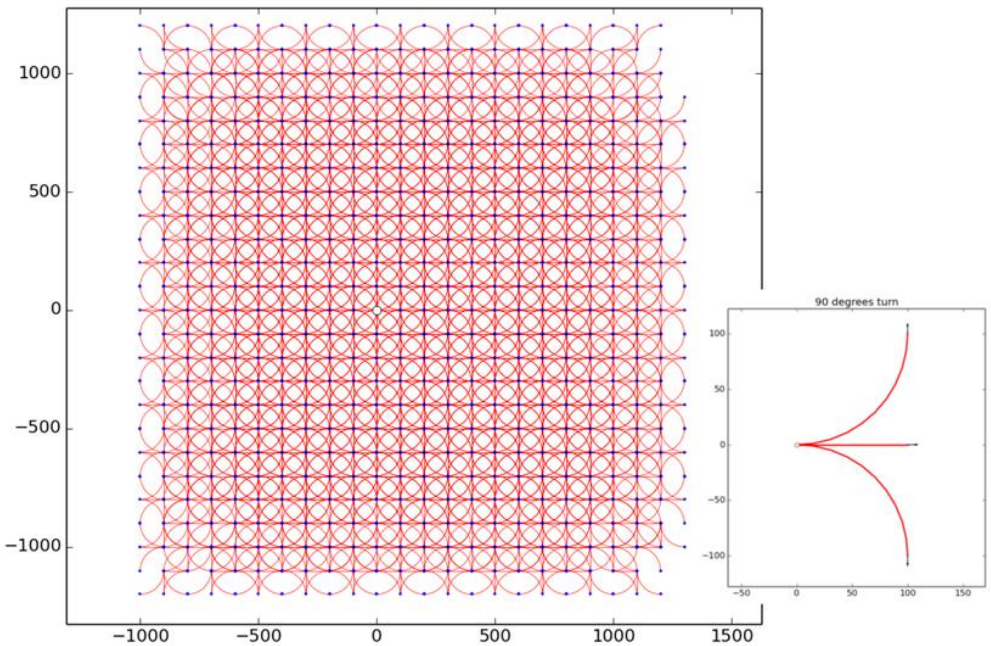


Use Dijkstra/A* to find the shortest path to the desired region in the tree.

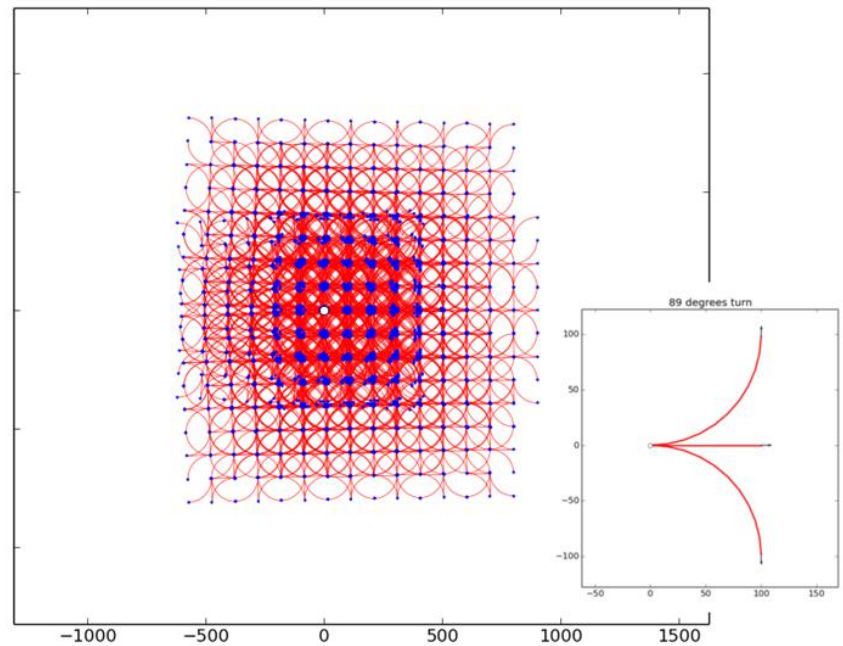
- Can be computed “lazily” during A* search.

Lattice generating motion primitives

- Some motion primitives generate regular lattice.



90-deg turns generate lattice



89-deg turns do not generate lattice.

Motion Primitives

Pros:

- No need for exact steering function
- Can handle differential constraints
- Model agnostic

Cons:

- Completeness and optimality achieved only up to discretization resolution
- Single-query

Notes:

- Used in CMU's Boss and Stanford's Junior during DARPA Urban Challenge

Solution Techniques for Path Planning Problem

- Variational Methods
- Graph-search Methods
  - Cell decomposition
  - Visibility graph
  - Sampling-based roadmap construction
  - Tree of motion primitives
- <u>Incremental Search Methods</u>
  - RRT: Rapidly-exploring Random Trees
  - RRT*: Optimal Rapidly-exploring Random Trees

Incremental Search

- Graph-based methods plan on a fixed resolution.
  - => Path might be suboptimal
  - => They may fail to find solution

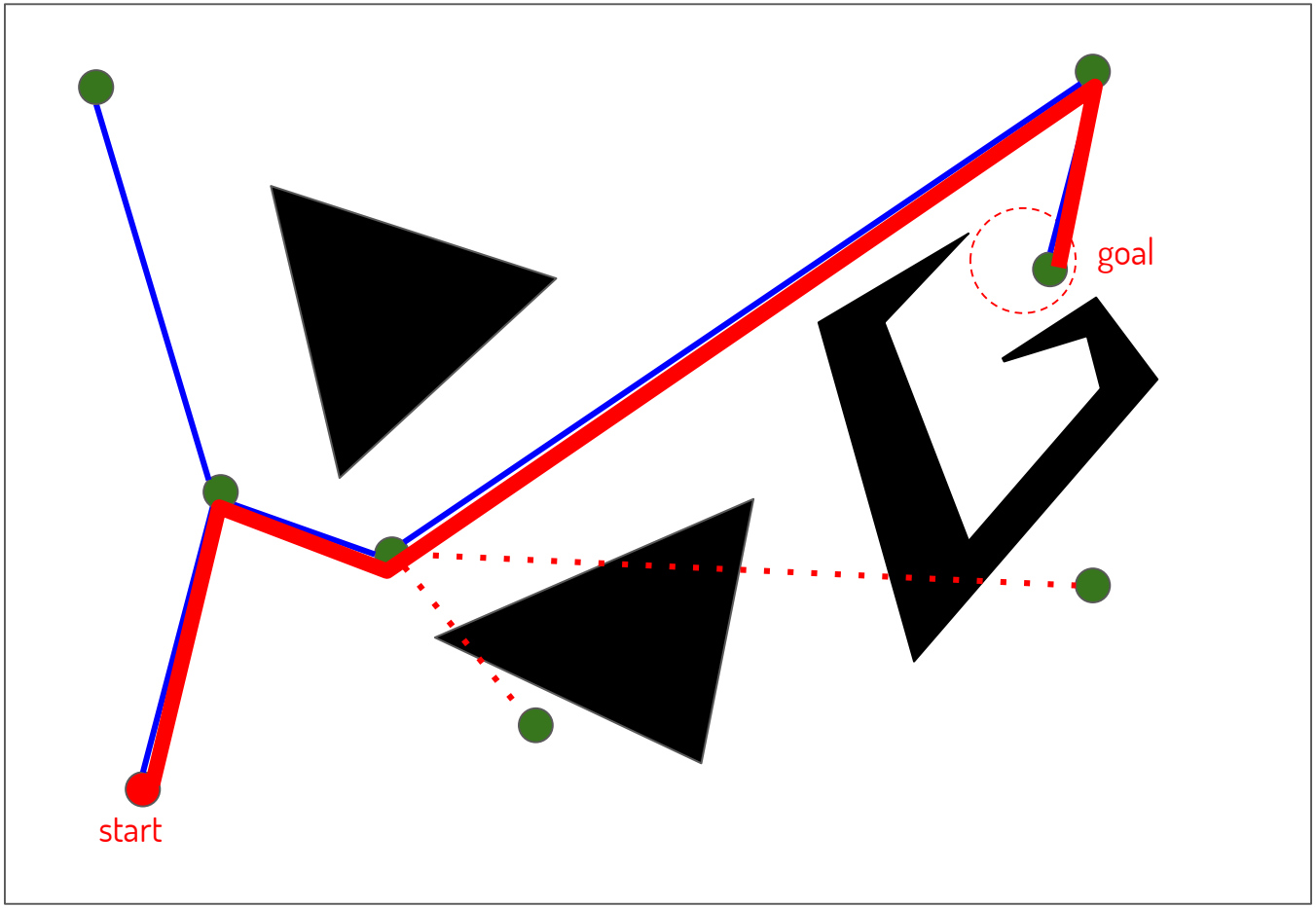
- Main idea:
  - Incrementally grow a tree rooted at initial configuration to explore the reachable region of the configuration space.
  - Once first branch reaches goal region, return the branch as the first solution.
  - Keep reporting the shortest branch found so far

- Anytime

Solution Techniques for Path Planning Problem

- Variational Methods
- Graph-search Methods
  - Cell decomposition
  - Visibility graph
  - Sampling-based roadmap construction
  - Tree of motion primitives
- Incremental Search Methods
  - RRT: Rapidly-exploring Random Trees
  - RRT*: Optimal Rapidly-exploring Random Trees

Rapidly-exploring Random Tree (RRT)



Rapidly-exploring Random Tree (RRT)

Pros:

- Anytime
- Handles differential constraints
- Does not need exact steering
- Probabilistic completeness guarantee (shown for some variants of the algorithm)
- Demonstrated good performance in high-dimensional systems

Cons:

- Suboptimal
- Single-query

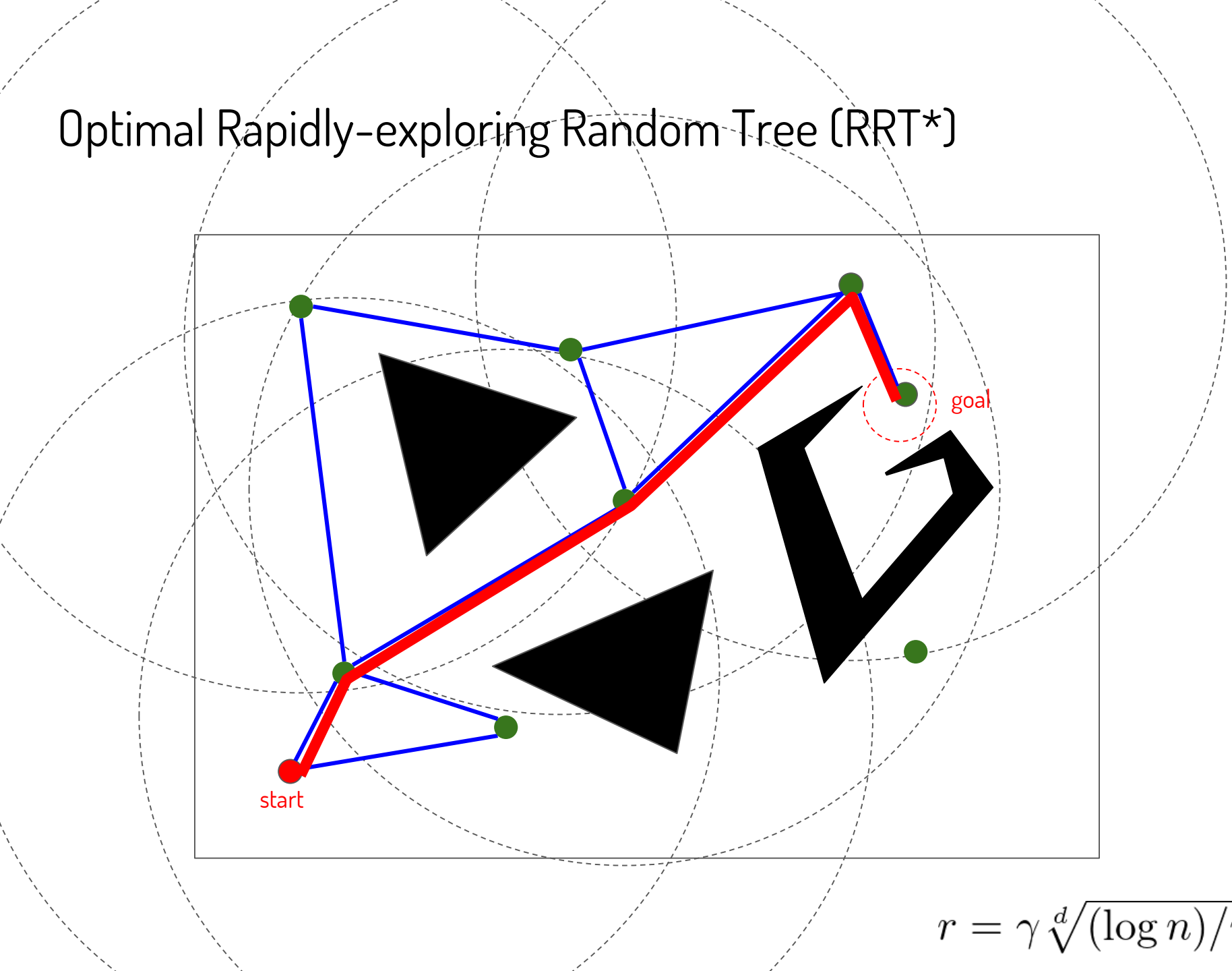
Notes:

- Used in MIT Talos Urban Challenge Vehicle

Solution Techniques for Path Planning Problem

- Variational Methods
- Graph-search Methods
  - Cell decomposition
  - Visibility graph
  - Sampling-based roadmap construction
  - Tree of motion primitives
- Incremental Search Methods
  - RRT: Rapidly-exploring Random Trees
  - RRT*: Optimal Rapidly-exploring Random Trees

Optimal Rapidly-exploring Random Tree (RRT*)



$$r = \gamma \sqrt[d]{(\log n)/n}$$

Optimal Rapidly-exploring Random Tree (RRT*)

Pros:

- Anytime
- Asymptotic optimality/Probabilistic completeness guarantee
- Can handle differential constraints

Cons:

- Requires exact steering
- Single-query

Summary

- Motion planning is needed in complex driving situations
- Path Planning vs. Trajectory Planning
- Different solution approaches
  - Variational
  - Graph-based
  - Incremental