Efficient Collision Detection in Sampling-Based Path Planning via Candidate Obstacle Filtering by Sorting Axis-Aligned Boundaries

Yechun Ruan

12233219

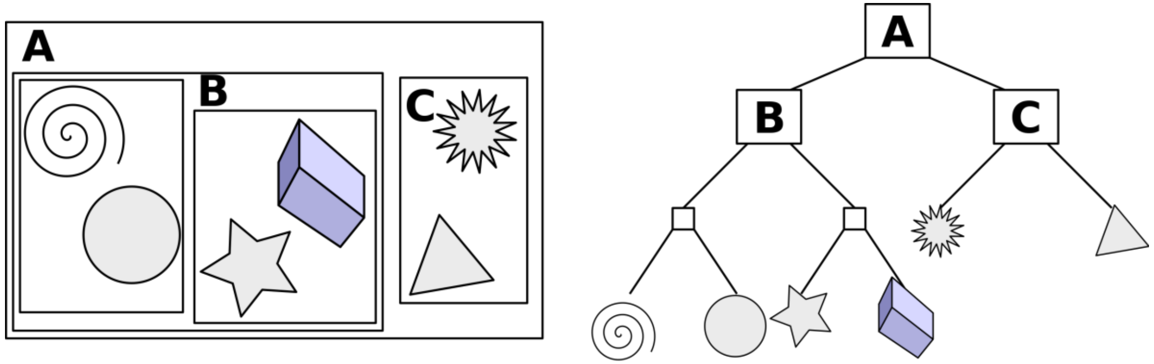
2023/05/04

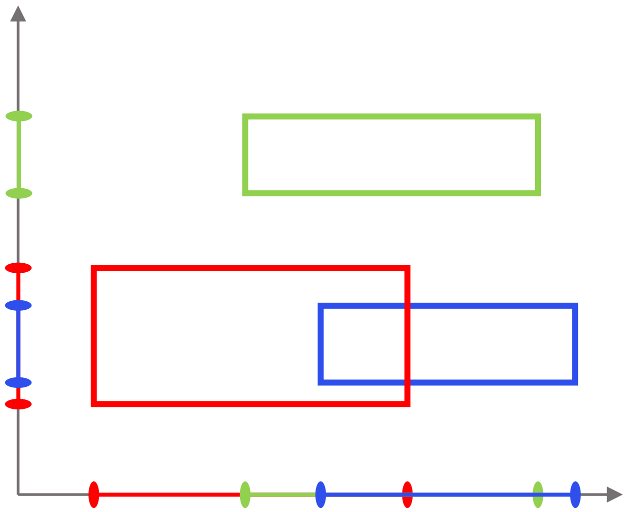
Introduction

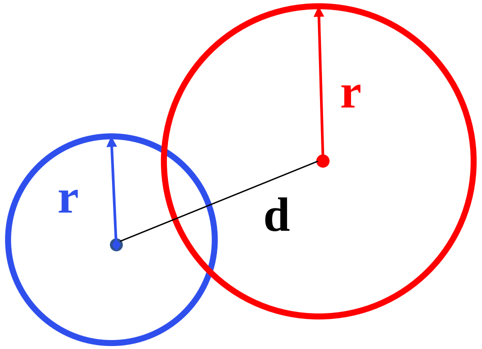
Collision detection time =
single execution time × number of executions

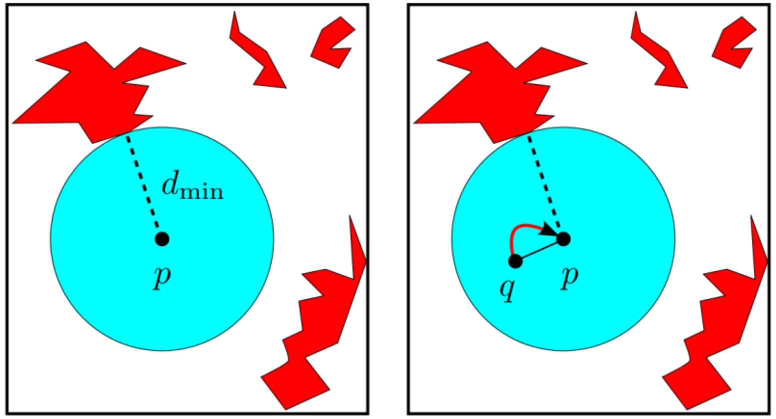
Obstacle Detection in sampling-based path planning

- Axis Aligned Bounding Box
- Circle
- Oriented Bounding Box
- Separating Axis Theorem
- Gilbert–Johnson–Keerthi









https://en.wikipedia.org/wiki/Bounding_volume_hierarchy

Joshua Bialkowski, Michael Otte, Sertac Karaman, and Emilio Frazzoli. Efficient collision checking in sampling-based motion planning via safety certificates. The International Journal of Robotics Research, 35(7):767–796, 2016

Method

Research Question: How to avoid needless collision detection efficiently

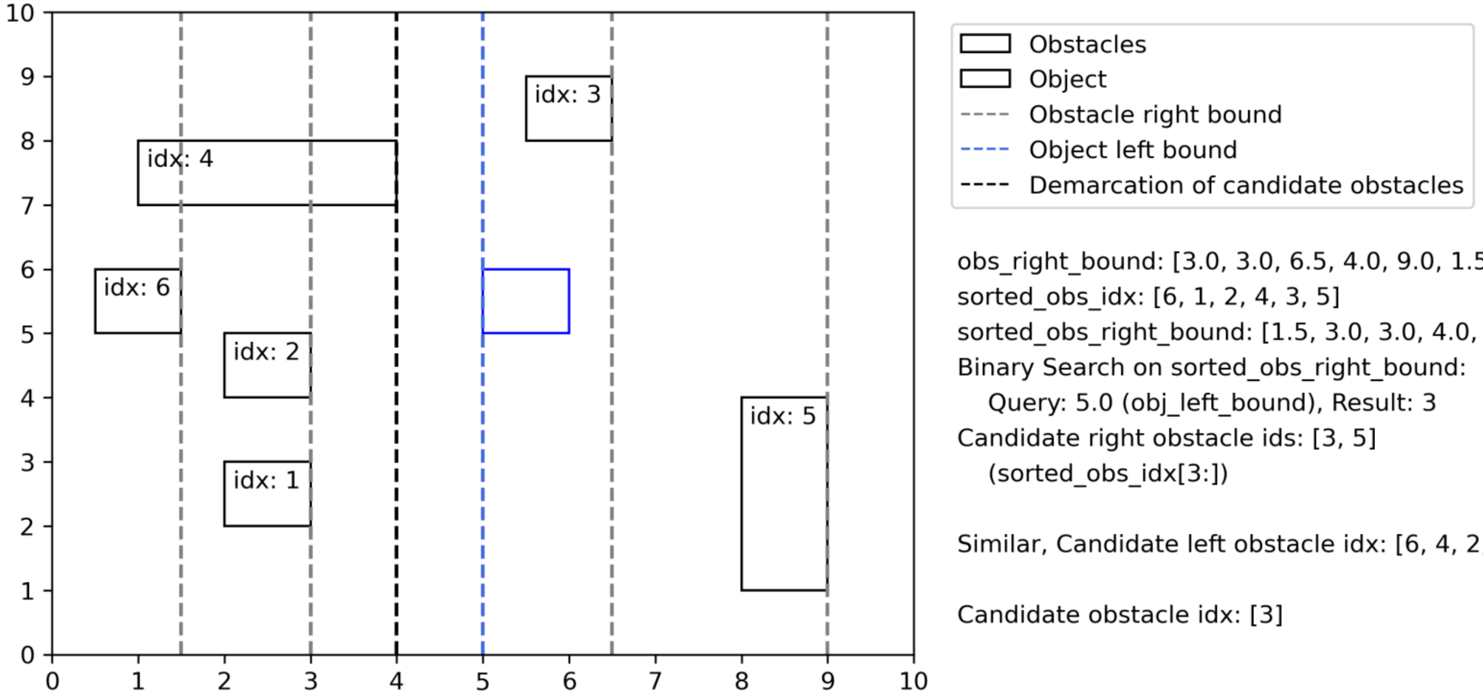
Observation: An obstacle whose right boundary is on the left side of the left boundary of the object must not collide with the object

Key Techniques:

- Sorted Axis-Aligned Boundaries
- Binary Search

Moreover:

- High-dimensional Space
- Dynamic Environment



```
obs_right_bound: [3.0, 3.0, 6.5, 4.0, 9.0, 1.5]
sorted_obs_idx: [6, 1, 2, 4, 3, 5]
sorted_obs_right_bound: [1.5, 3.0, 3.0, 4.0, 6.5, 9.0]
Binary Search on sorted_obs_right_bound:
    Query: 5.0 (obj_left_bound), Result: 3
Candidate right obstacle ids: [3, 5]
    (sorted_obs_idx[3:])

Similar, Candidate left obstacle idx: [6, 4, 2, 1, 3]

Candidate obstacle idx: [3]
```

Results

RRT: collision detection with Brute Force
RRT_COF: collision detection with Candidate Obstacle Filtering

The functions *get_obstacles_about_segment* and *get_obstacles_about_node* are called inside the function *is_collision_free*

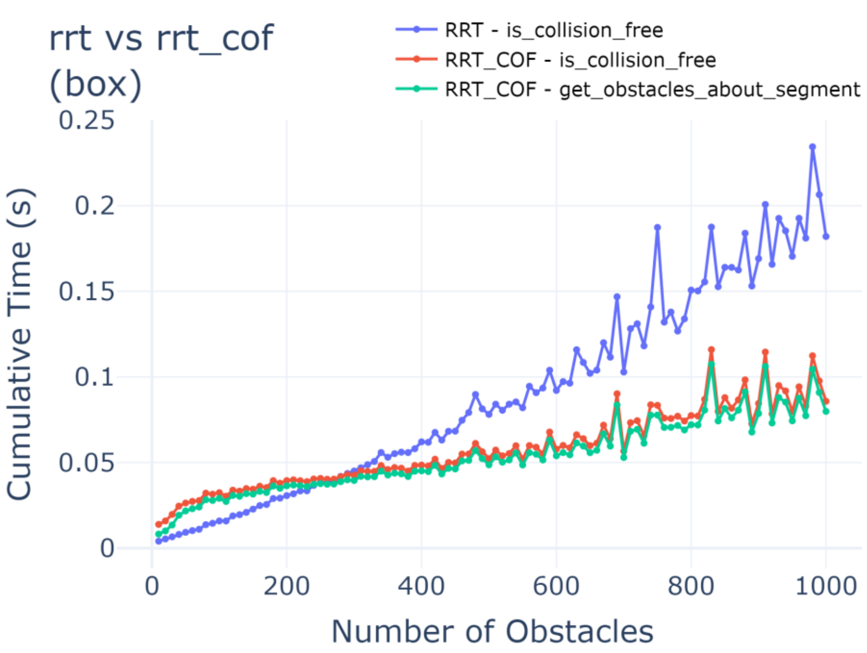
Experiments in the **random world**

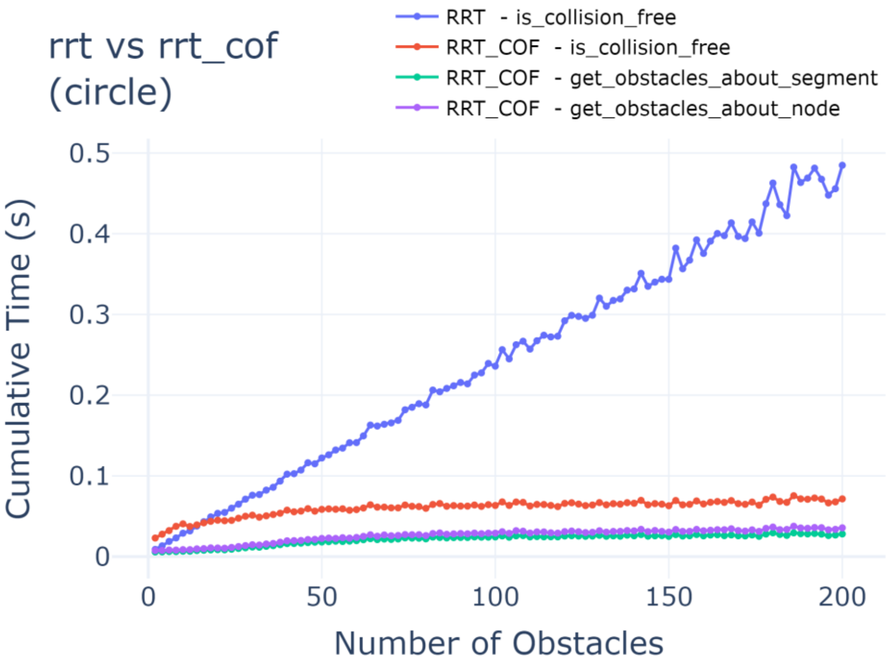
map config

- $play\_area = (0, 600, 0, 600)$
- $rnd\_area = (10, 590, 10, 490)$
- $box\_obstacle\_constraint = (min\_width, min\_height, max\_width, max\_height) = (2, 6, 2, 6)$
- $circle\_obstacle\_constraint = (min\_radius, max\_radius) = (2, 6)$
- $map\_cnt\_per\_obstacle\_num = 10$
- $num\_box\_obstacles = [10, 20, ..., 1000]$
- $num\_circle\_obstacles = [2, 4, ..., 200]$

robot config

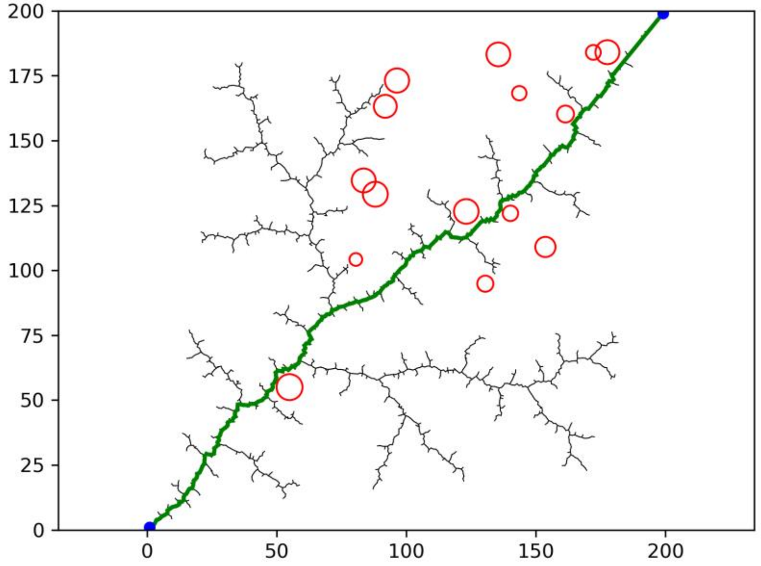
- $start = (1, 1)$
- $goal = (599, 599)$
- $step\_length = 3$
- $robot\_radius = 1$

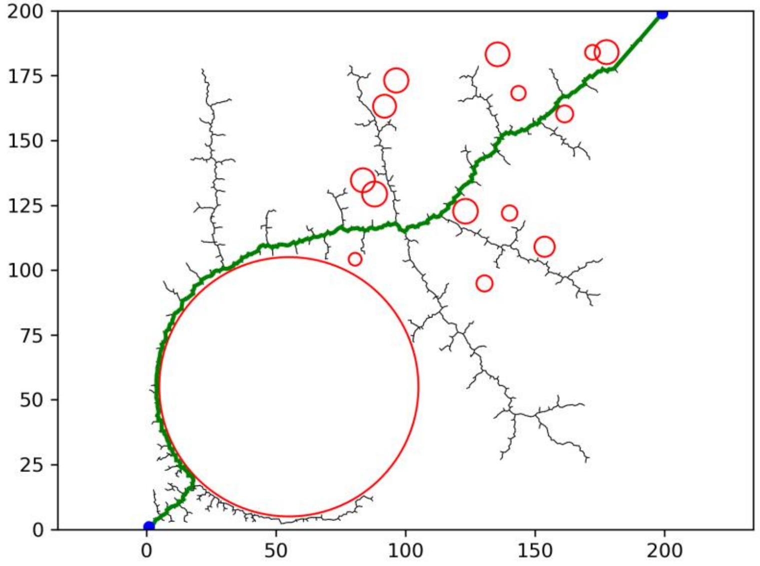




Results

Experiments in the **simple map**





number of iterations 2041

time consumption of function *is_collision_free*:

RRT 0.046 s, RRT_COF 0.039 s

speedup **15.2%**

number of iterations 8446

time consumption of function *is_collision_free*:

RRT 0.127 s, RRT_COF 0.091 s

speedup **28.3%**

Conclusions

Candidate Obstacle Filtering

- based on **sorted Axis-Aligned Boundaries** and **Binary Search**
- optimizes the number of obstacles to be detected from O(n) to O(1)
- the performance improvement is proportional to
    - obstacle complexity (individual obstacle collision detection time)
    - the number of obstacles
    - Number of obstacle detection executions (number of iterations)