Chair for Cognitive Systems
Department of Electrical and Computer Engineering
Technische Universität München

Trajectory Planning using Bi-RRT Algorithm

Humanoid Robocup Advanced Lab

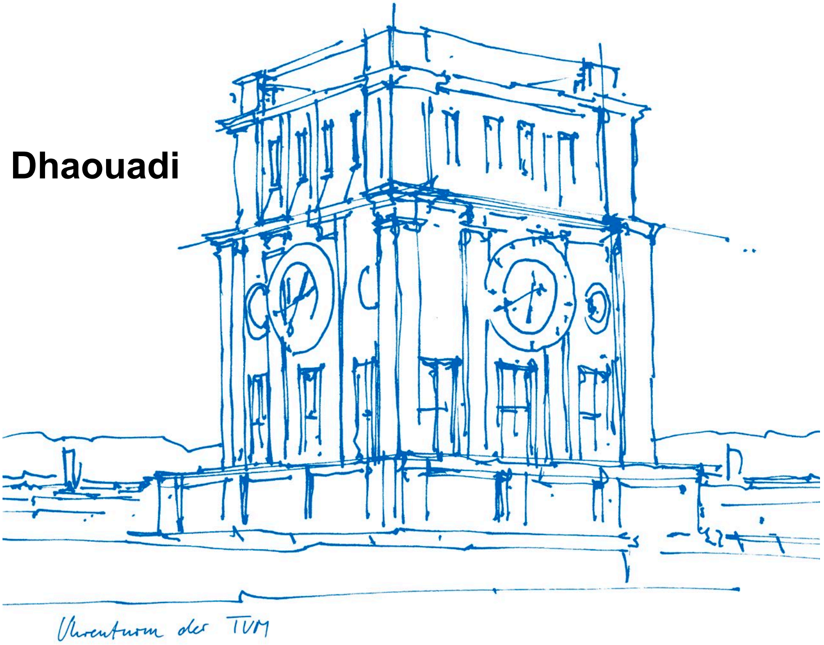
B.Sc. Ahmed Mzid – B.Sc. Helmi Abidi – B.Sc. Oussema Dhaouadi

Supervisor : M.Sc. Quentin Leboutet

Chair for Cognitve Systems

Prof. Dr. Gordon Cheng

Munich, 23. July 2019



Motivation: Problem Statement

- Motion Planning: core functionality

- Old implementation based on aggregation of displacement vectors

- Poor performance in obstacle avoidance
- easily predictable deterministic behavior

- This motivates the necessity of a more robust approach

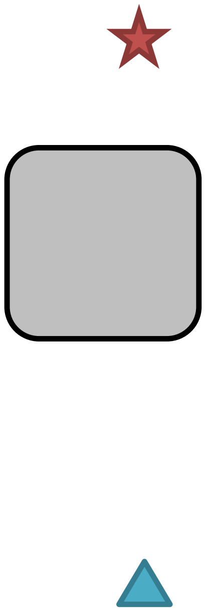
Motivation: Solution Proposal

- **RRT**: **R**apidly **R**andom exploring **T**ree

- Collision-free trajectory: more efficient environment exploration

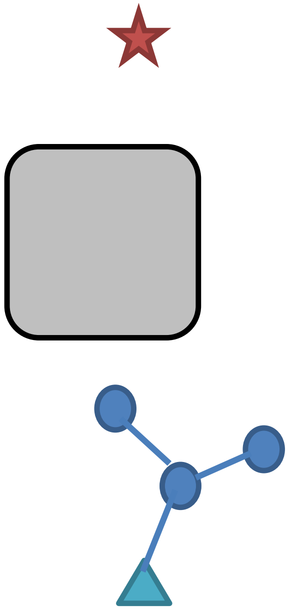
- Randomness: Behavior harder to predict

RRT* Algorithm



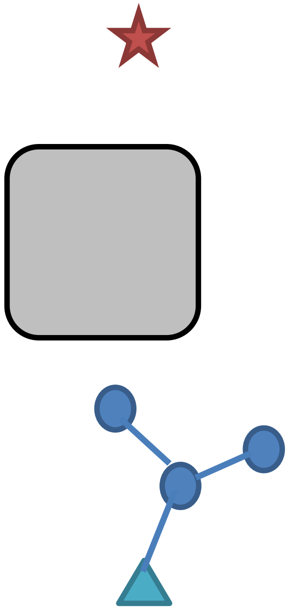
First iteration

RRT* Algorithm

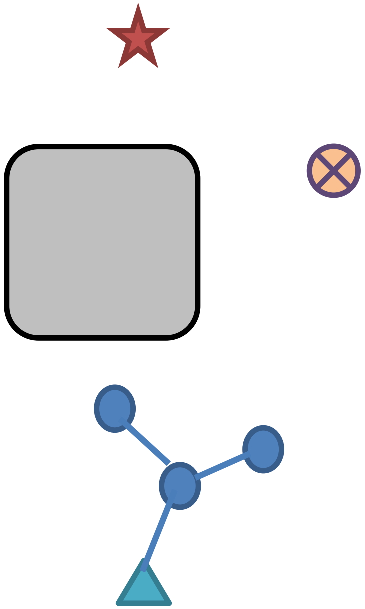


Iteration begin

RRT* Algorithm

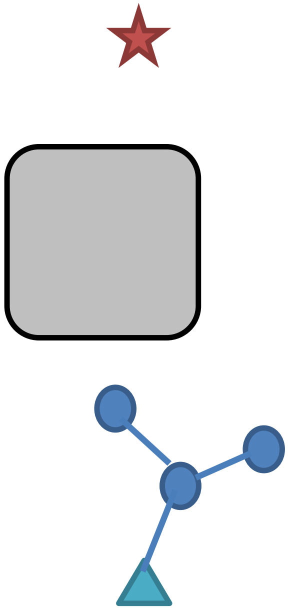


Iteration begin

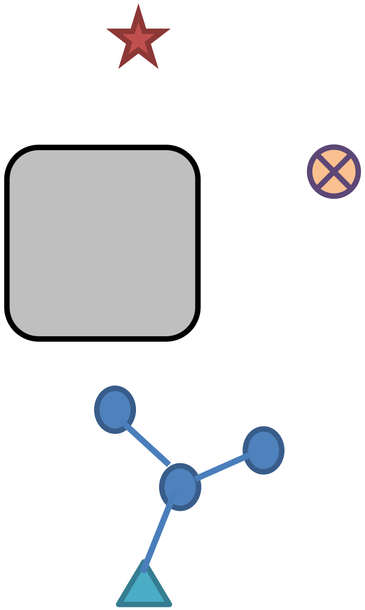


Sample
random point

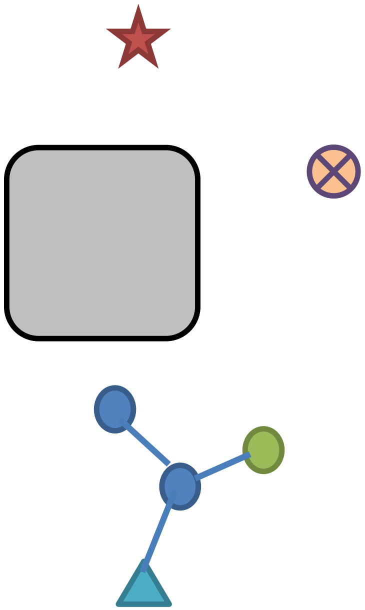
RRT* Algorithm



Iteration begin

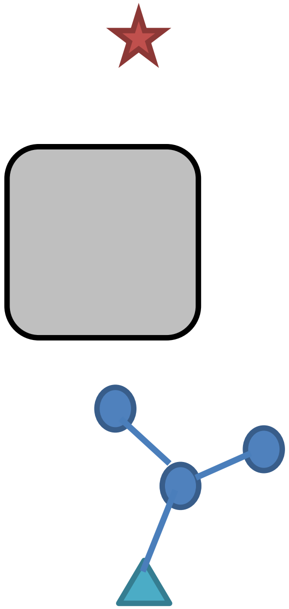


Sample random point

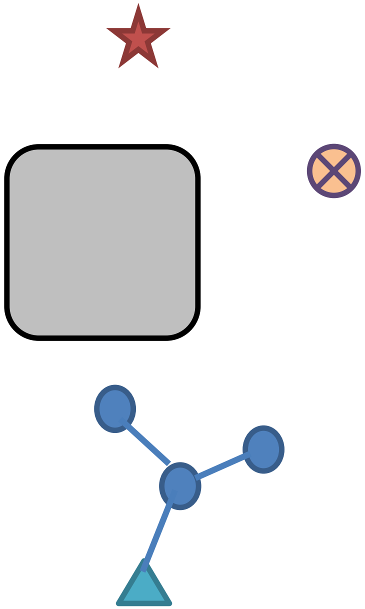


Find closest node

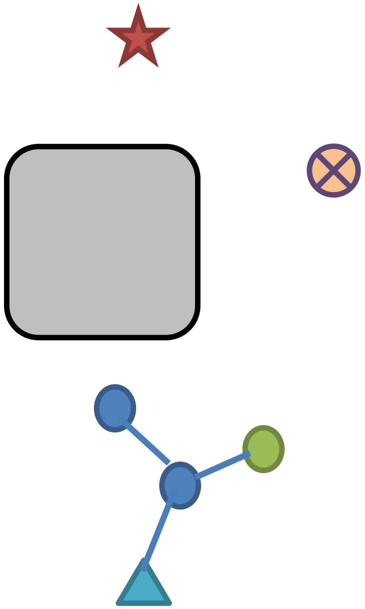
RRT* Algorithm



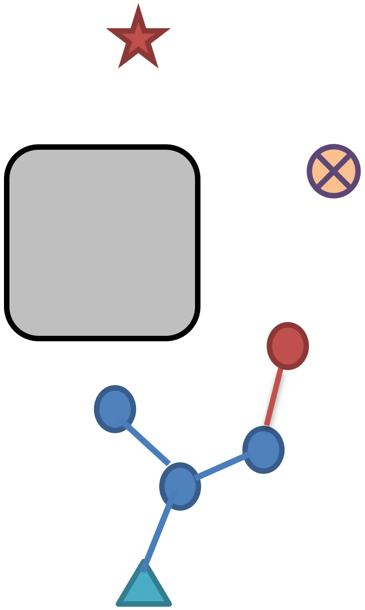
Iteration begin



Sample random point

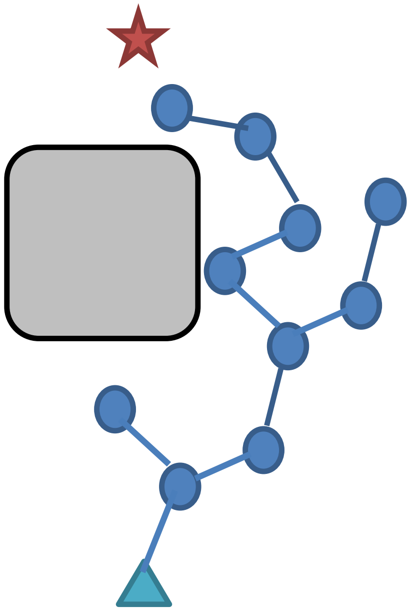


Find closest node



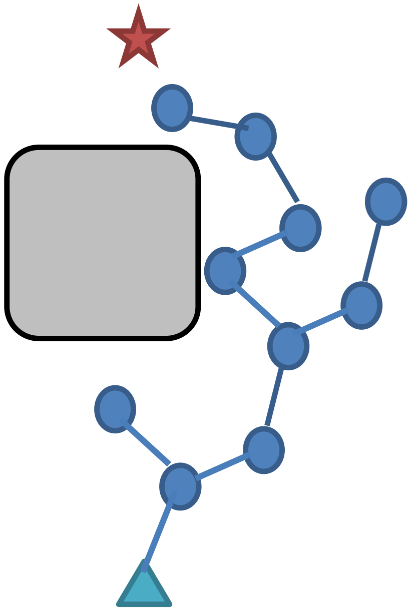
Do a small step in the sampled direction and update parent

RRT* Algorithm

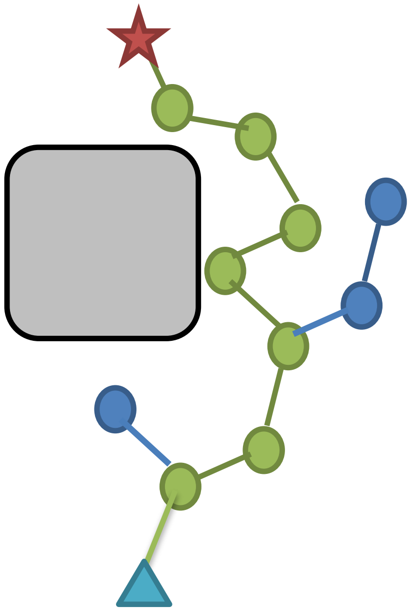


Reiterate the previous
process until goal reached

RRT* Algorithm



Reiterate the previous process until goal reached



Compute the final optimal path

Bi-RRT* Algorithm

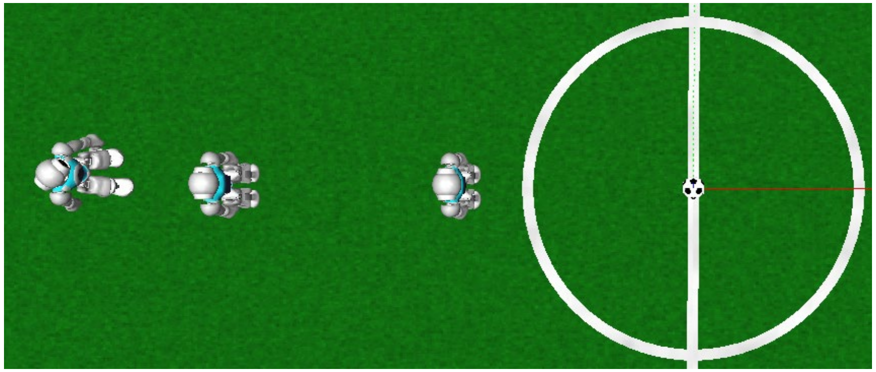
Bi-RRT* Algorithm:

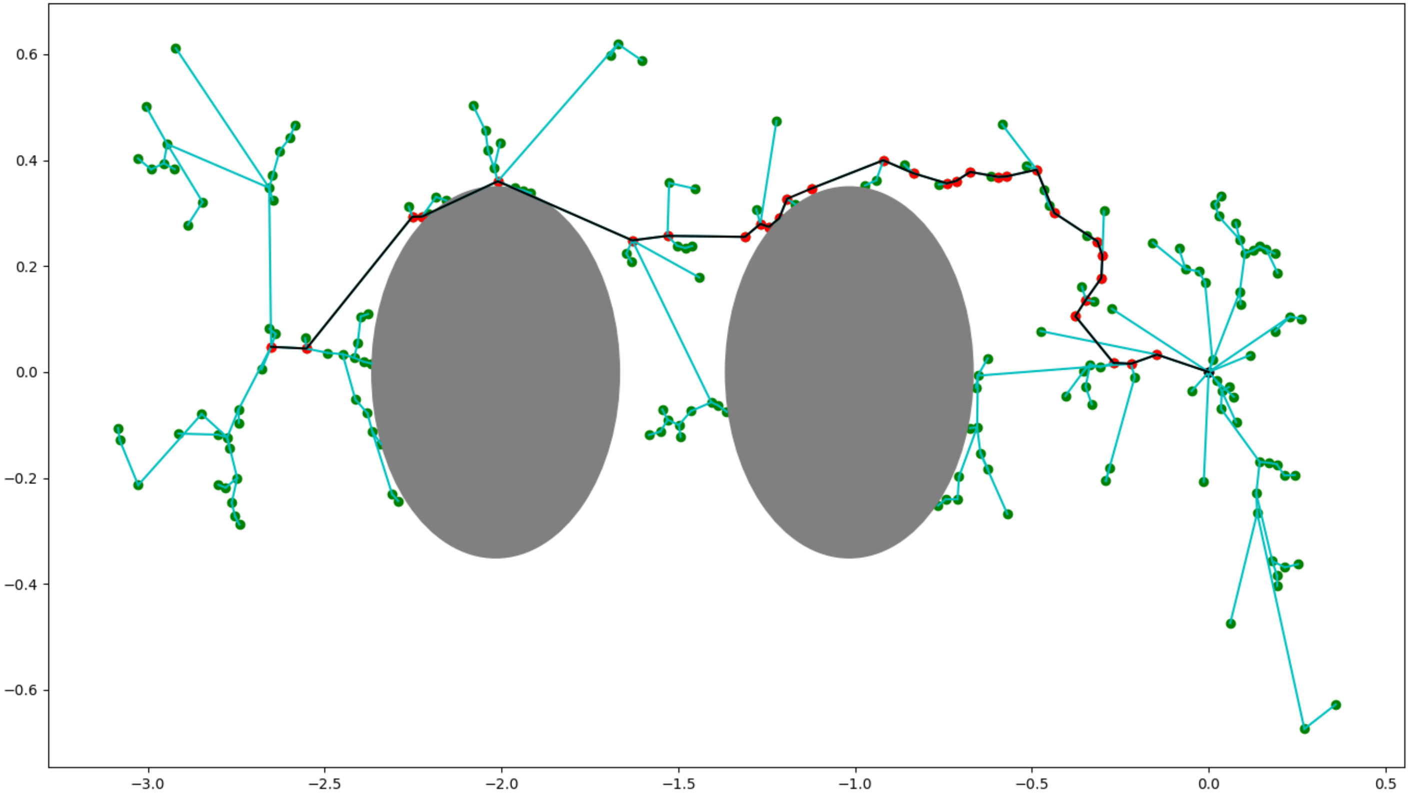
Perform the previous algorithm (RRT*) in two opposite directions:
- Build an exploring tree starting from the robot position
- Build a second exploring tree starting from the target position

Merge both trees once two nodes are close enough

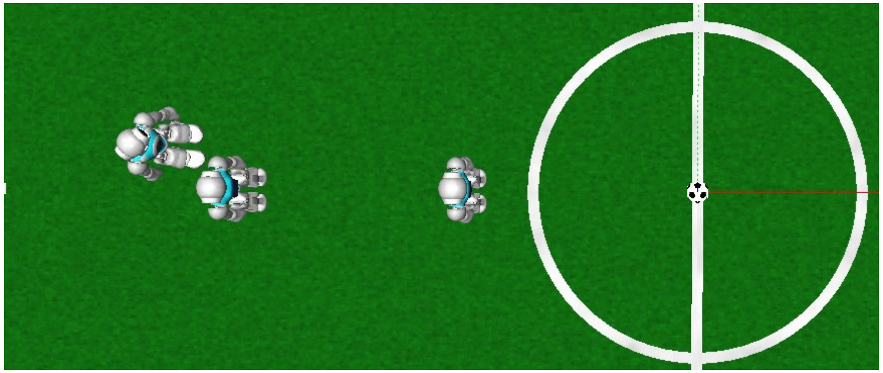
Implementation based on:
- Adaptive step size
- Adaptive neighborhood radius for parent assignment
- Adaptive sampling area

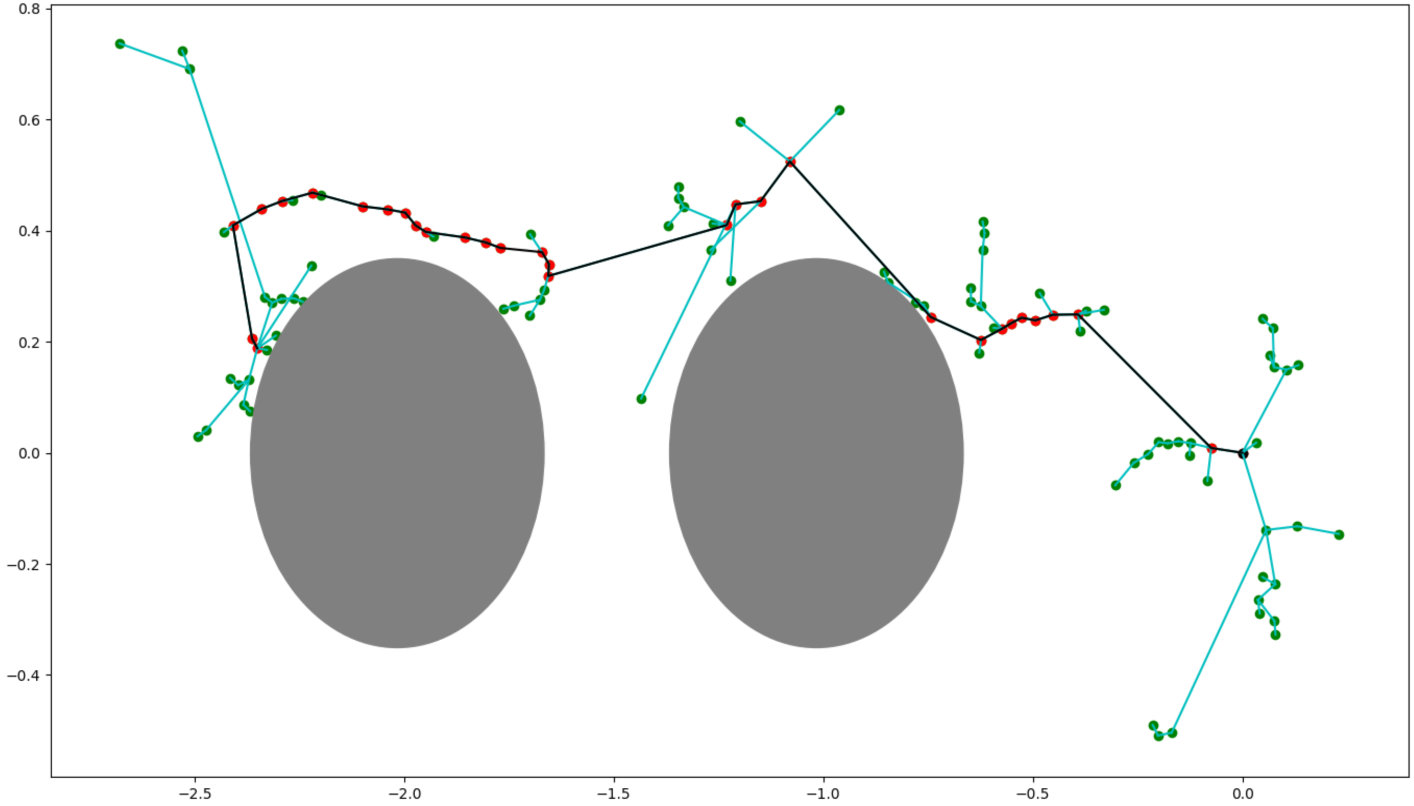
Bi-RRT* Algorithm





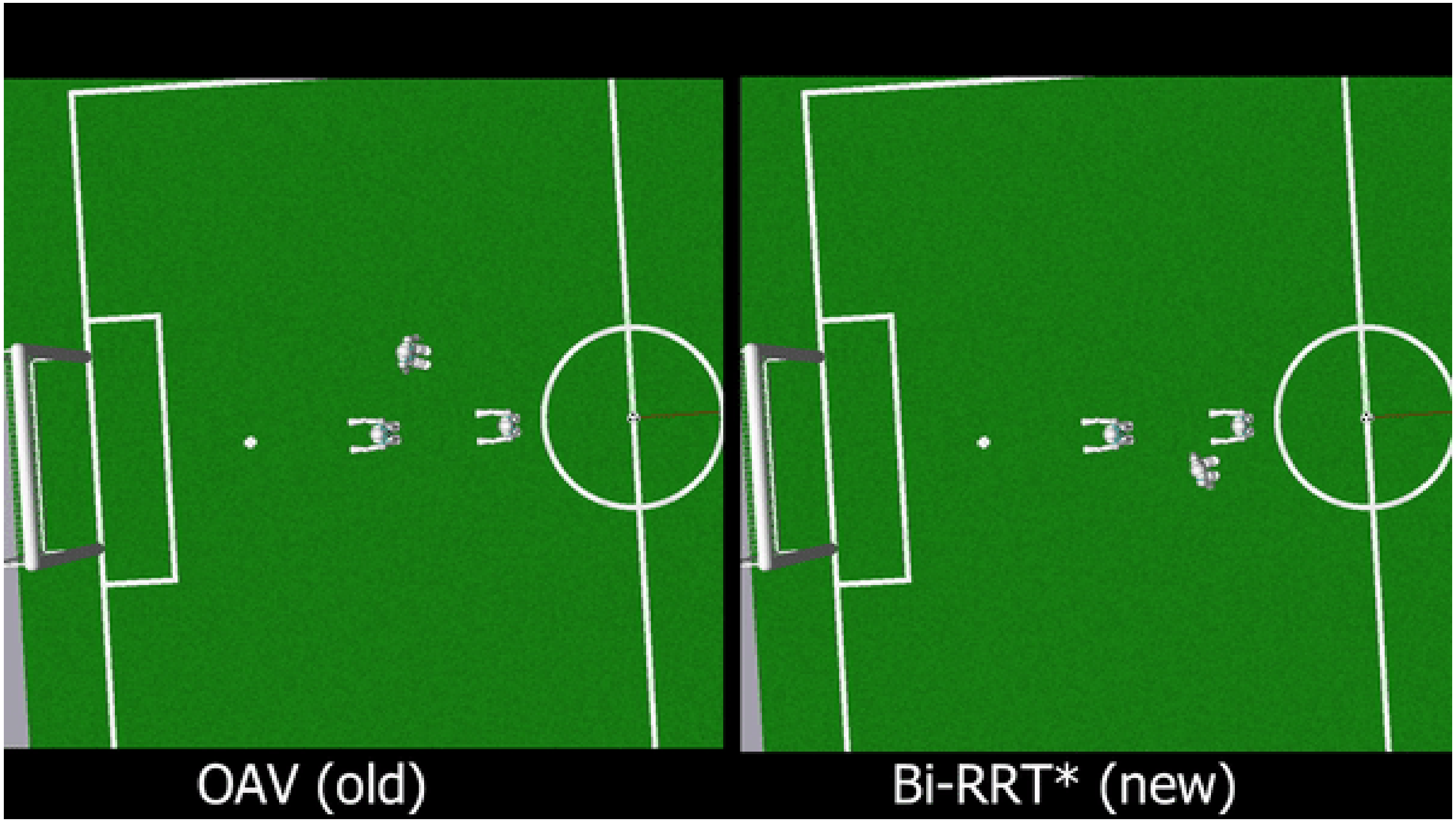
Bi-RRT* Algorithm





Simulated Scenarios
Obstacle Avoidance Vector vs. Bi-RRT*

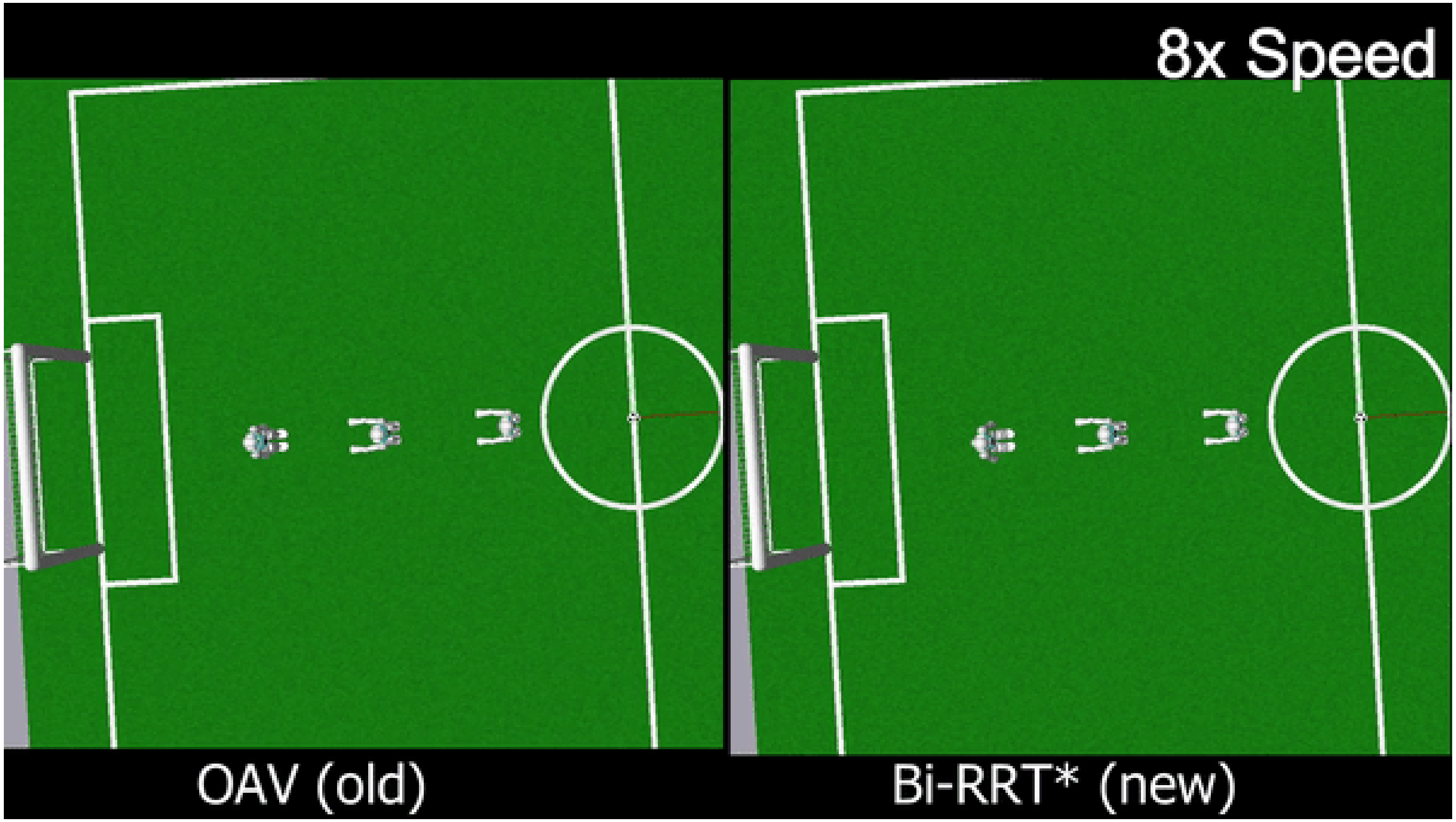
Static obstacles: video OAV vs Bi-RRT



Video link

Simulated Scenarios
Obstacle Avoidance Vector vs. Bi-RRT*

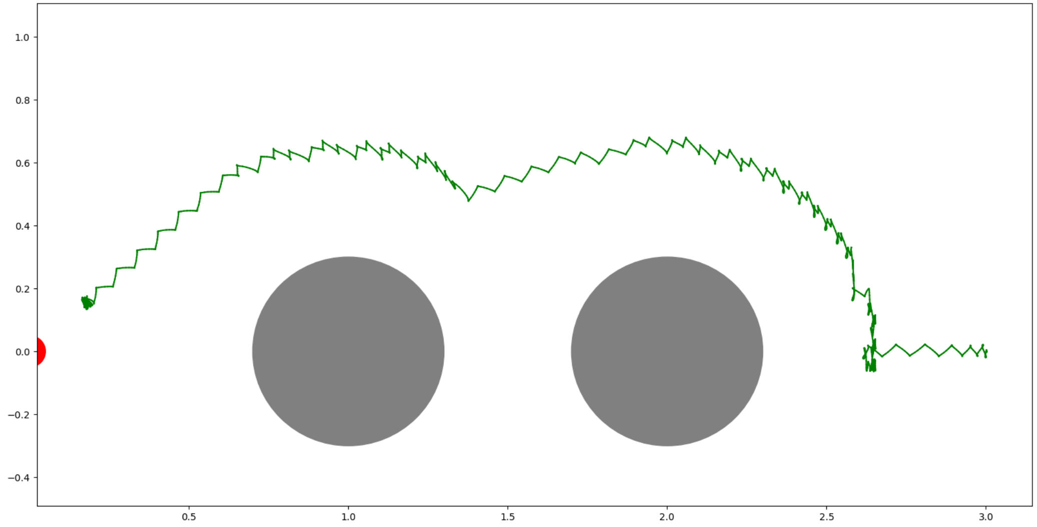
Static obstacles: video OAV vs Bi-RRT



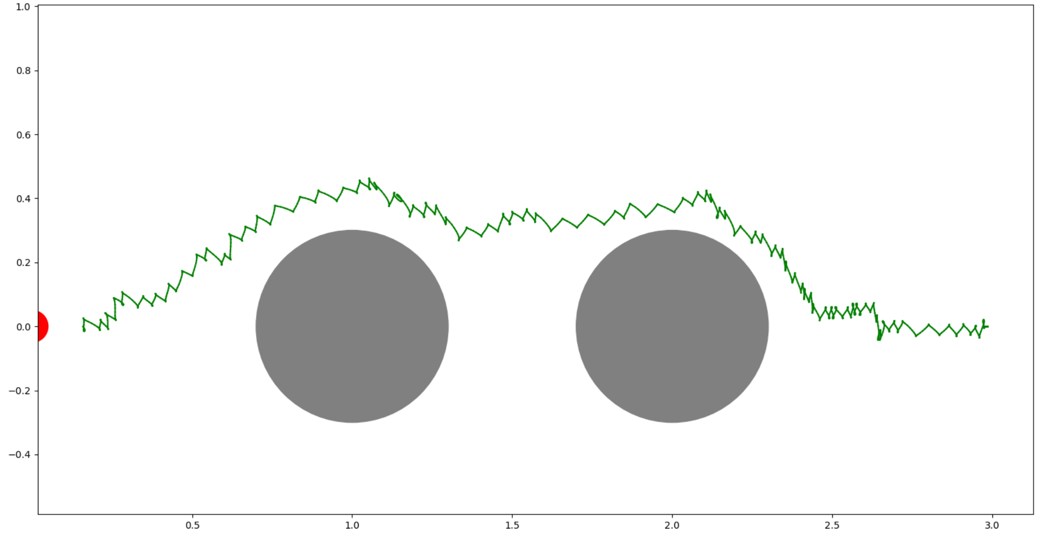
Video link

Simulated Scenarios

Final path taken by the robot by the old implementation (OAV)



Final path taken by the robot by the new implementation (RRT)



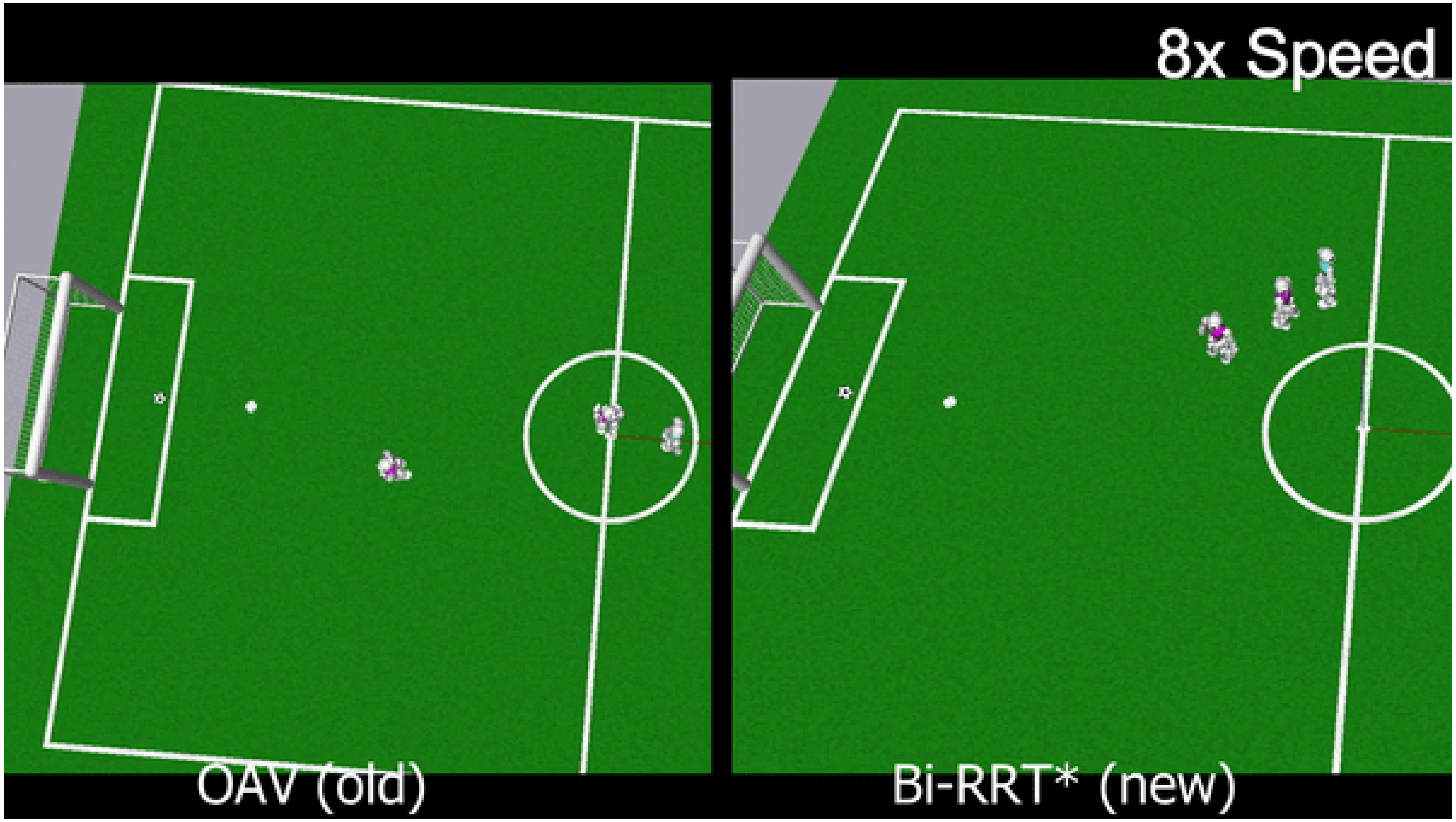
Simulated Scenarios

Obstacle Avoidance Vector vs. Bi-RRT*

	Obstacle Avoidance Vector	Bi-RRT*
Time from starting position to target	142.335 s	113.716 s
Number of steps from starting position to target	2604 steps	2448 steps

Simulated Scenarios
Obstacle Avoidance Vector vs. Bi-RRT*

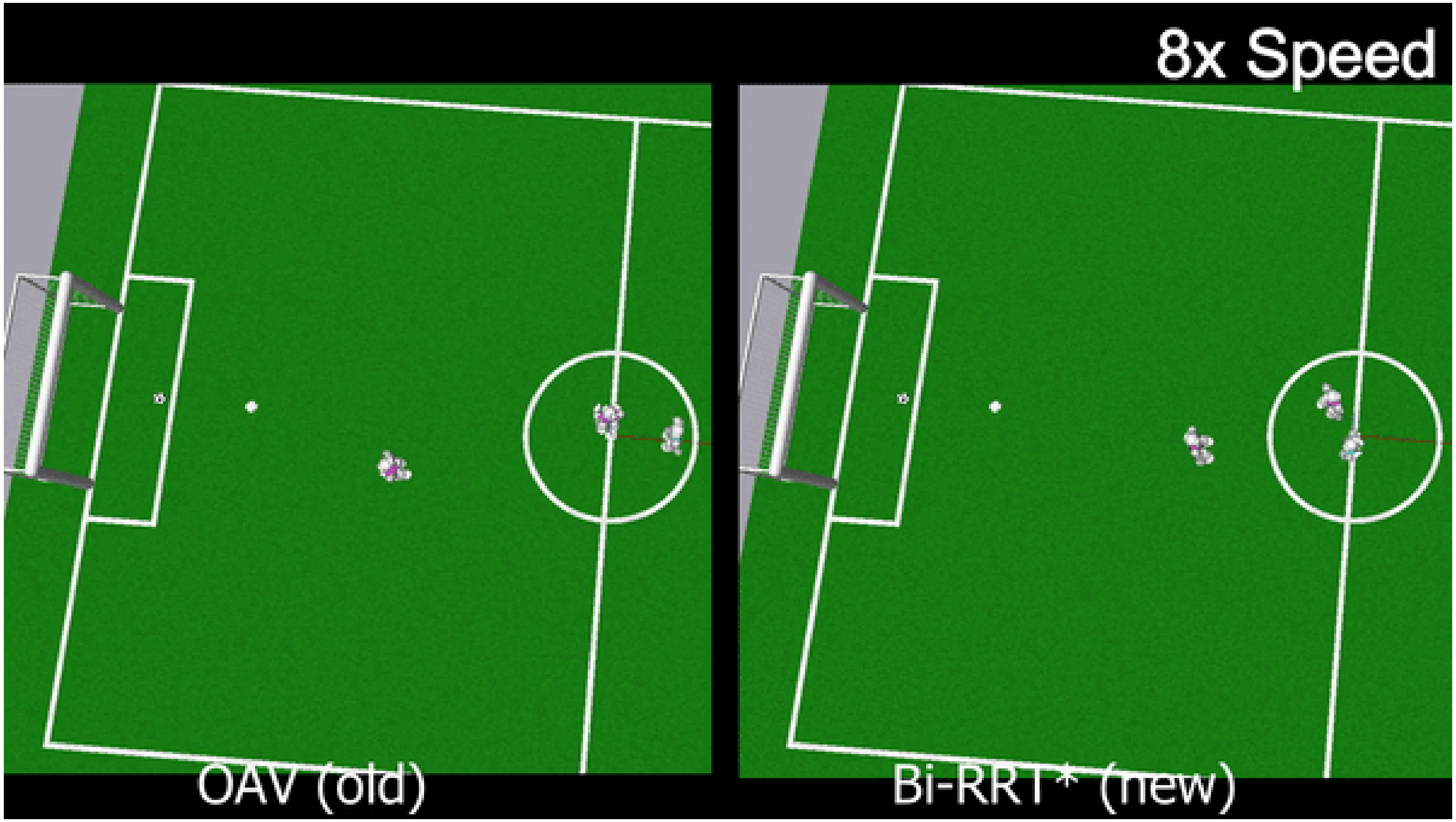
Dynamic obstacles: attack mode OAV vs Bi-RRT



Video link

Simulated Scenarios
Obstacle Avoidance Vector vs. Bi-RRT*

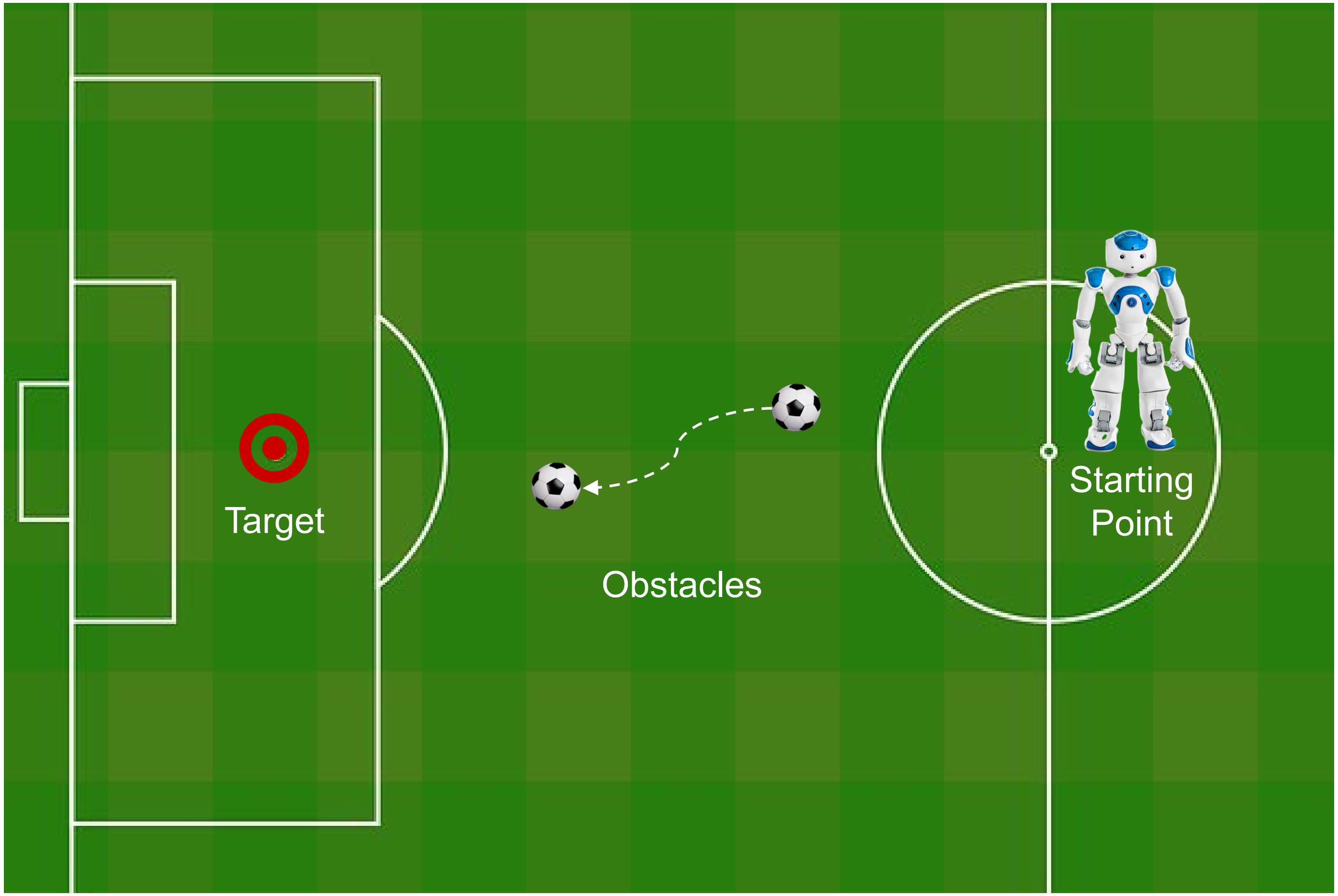
Dynamic obstacles: defense mode OAV vs Bi-RRT



Video link

Real Robot deployment

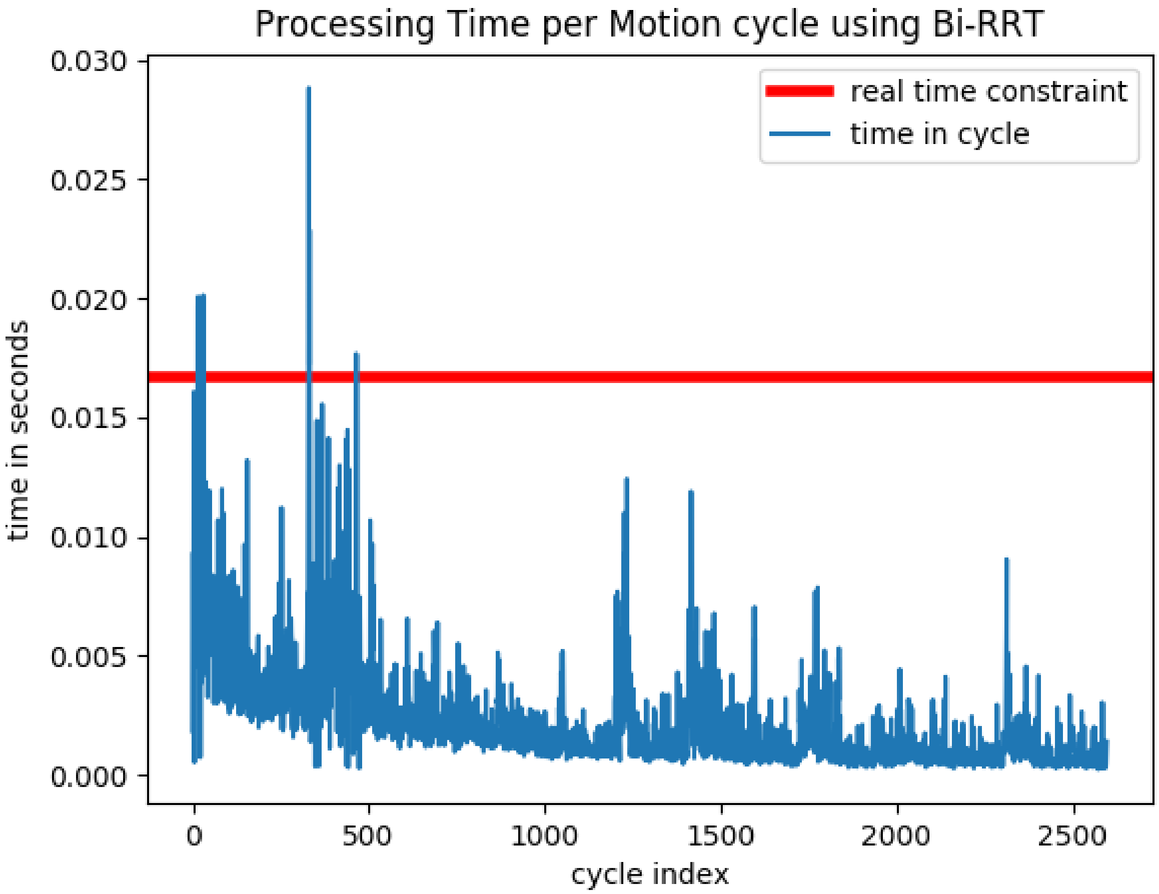
Experiment Setup:



Real Robot deployment

Computational Cost:

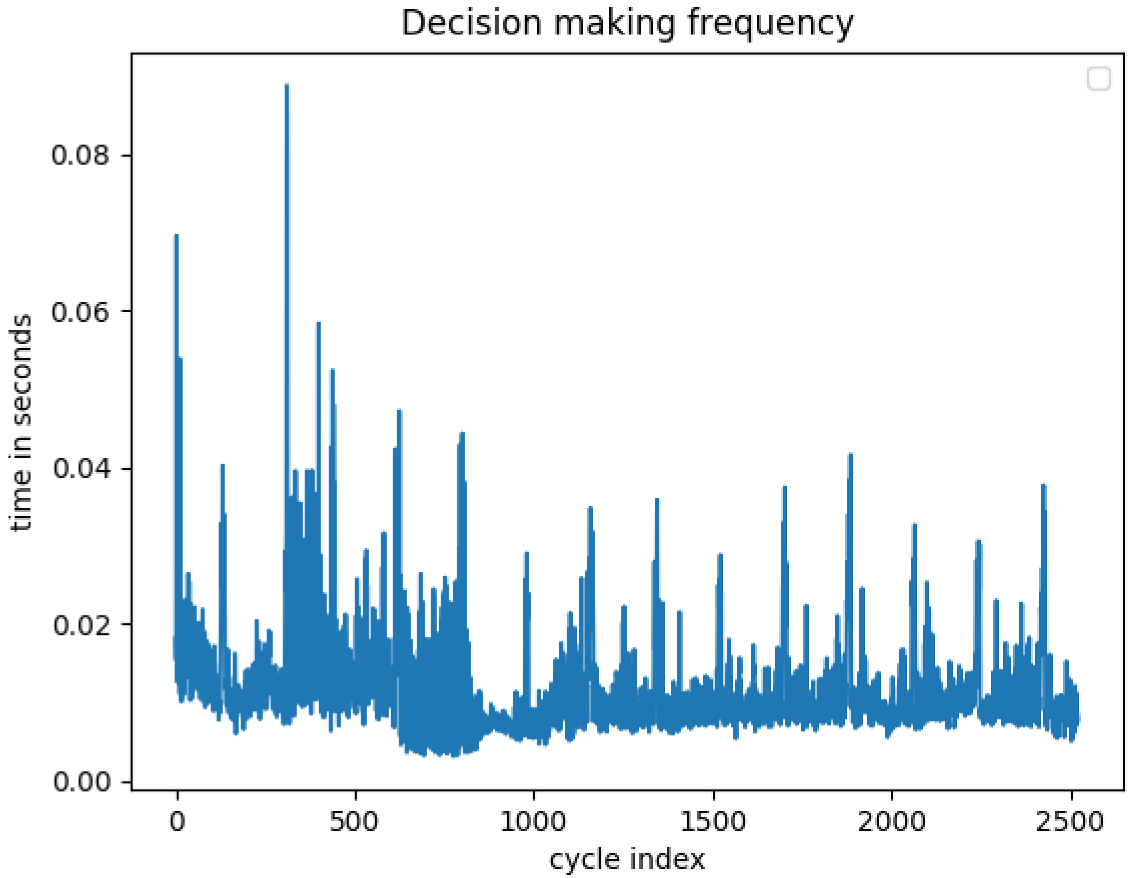
Average processing time per cycle is equal to **1.924 ms**



Real Robot deployment

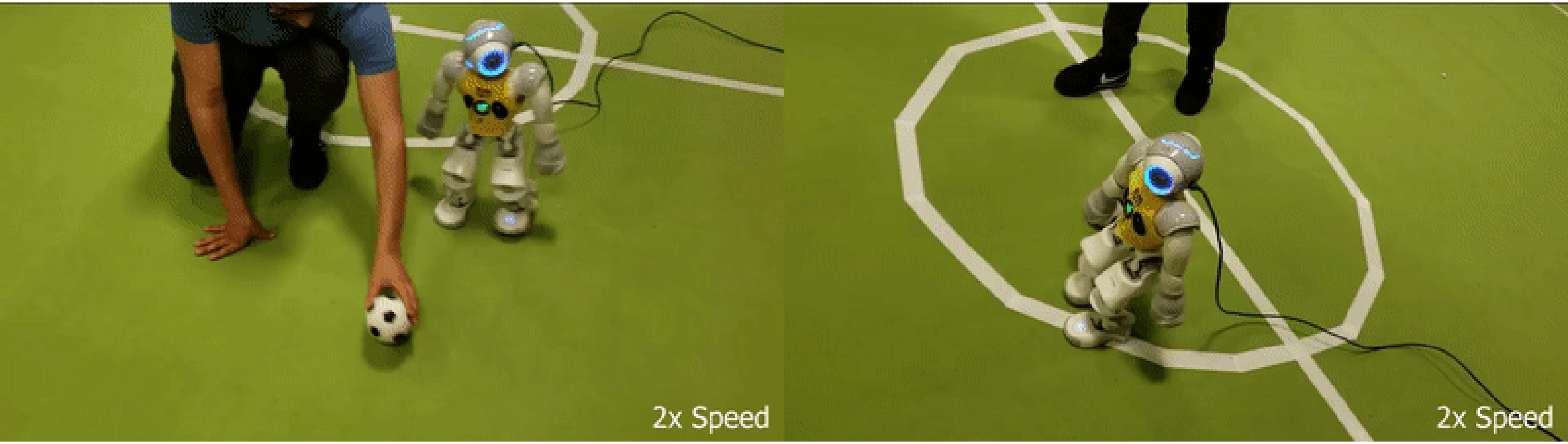
Computational Cost:

In average, a new decision is made every **13.153 ms**



Real Robot deployment

Video



Video link

Conclusion & Future Work

Conclusion:

- Bi-RRT: better obstacle avoidance than old algorithm
- Real-time constraints preserved

Future Work:

- Further parameter tuning
- Fusion of Bi-RRT with B-Splines for more smoothness

- Obstacle Detection
- Odoemtry
- Strategy and Player roles