Mobile Robot Systems

Lecture 9: Multi-Robot Navigation and Path Planning

Dr. Amanda Prorok

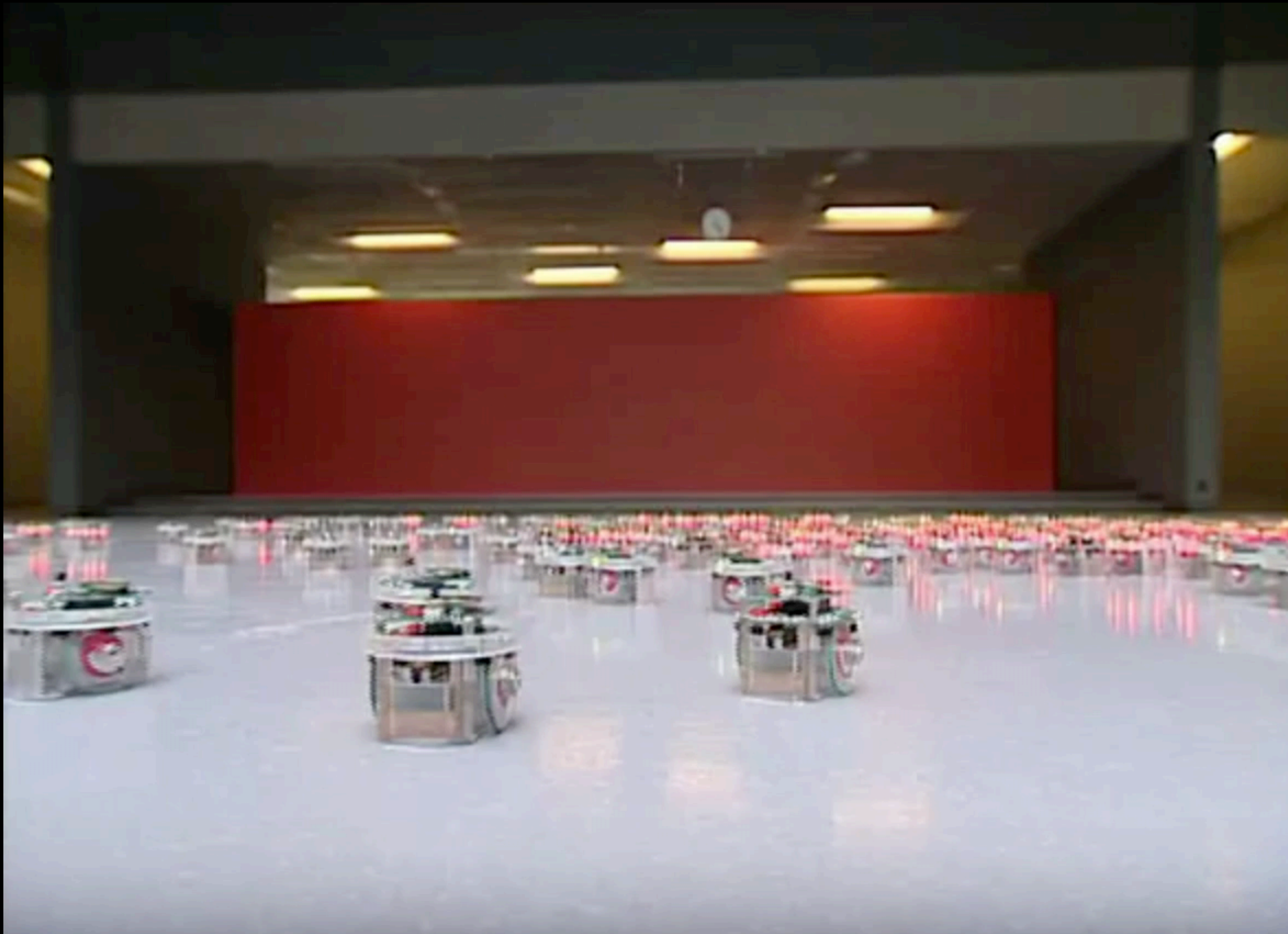
asp45@cam.ac.uk

www.proroklab.org

UNIVERSITY OF CAMBRIDGE

In this Lecture

- Taxonomy of MR path planning problems
- MR path planning methods:
  - Discrete
  - Continuous
- Concurrent assignment and path planning

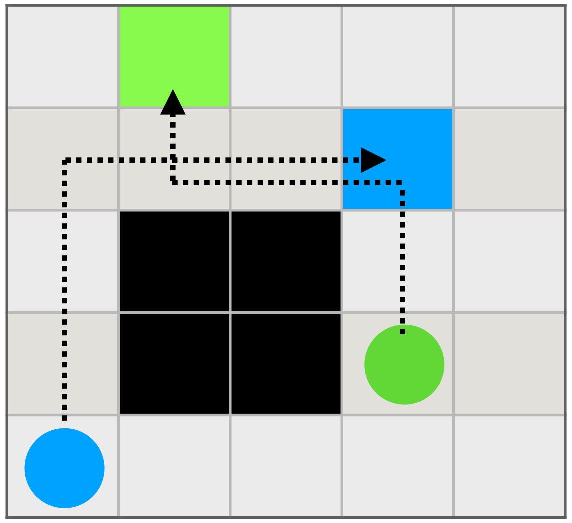


Taxonomy of Multi-Robot Path Planning Problems

- Domain: continuous vs. discrete
  - **Continuous**: planning time-parameterized trajectories in metric space.
  - **Discrete**: planning on graphs, or regular grids
- Goal assignment: labeled vs. unlabeled
  - **Labeled**: each robot has a predetermined goal destination
  - **Unlabeled**: all goals must be reached, but assignment is not predetermined
- Problem representation: coupled vs. decoupled
  - **Coupled**: represent the joint state of all robots in the system
  - **Decoupled**: each robot's state represented independently
- Planning: reactive vs. deliberative
  - **Reactive**: dynamic obstacle avoidance; plan as you go (cf. **decentralized**)
  - **Deliberative**: planning for optimality (cf. **centralized, coupled**)
- Computation: centralized vs. decentralized

Multi-Agent Path Planning

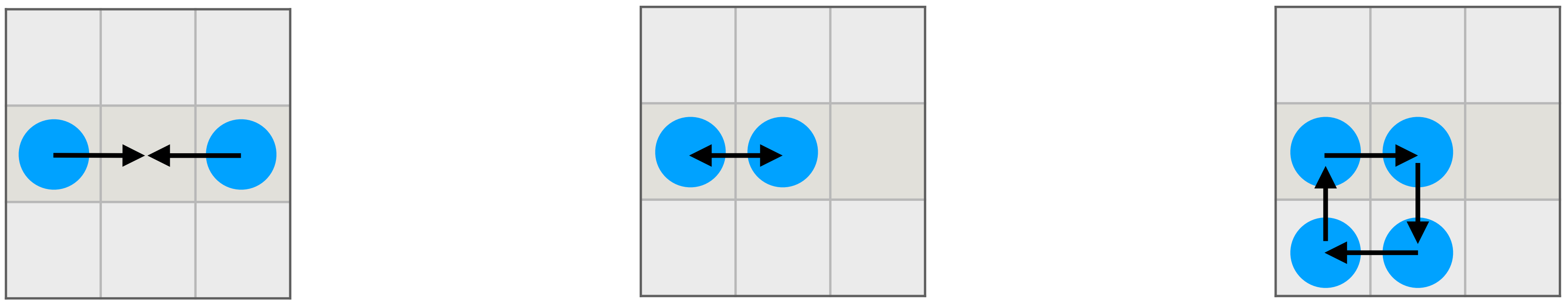
- Multi-robot path planning $\longrightarrow$ multi-agent path planning:
  - discretized environment (grids or planar graphs)
  - point robots (holonomic, no motion constraints)



- The problem:
  - Given: a number of agents at start locations with predefined goal locations, and a known environment
  - Task: find **collision-free paths** for the agents from their start to their goal locations that optimize some objective
- Generally, we assumed a **labeled** problem.
- Classical application domain: automated warehouses (e.g., Amazon)

Multi-Agent Path Planning

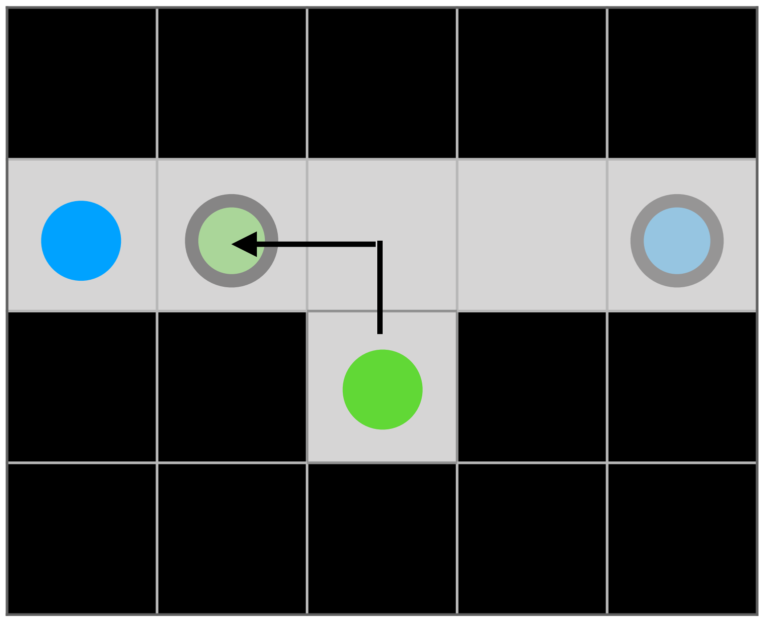
- Allowed motion: North, East, South, West
- Collisions:



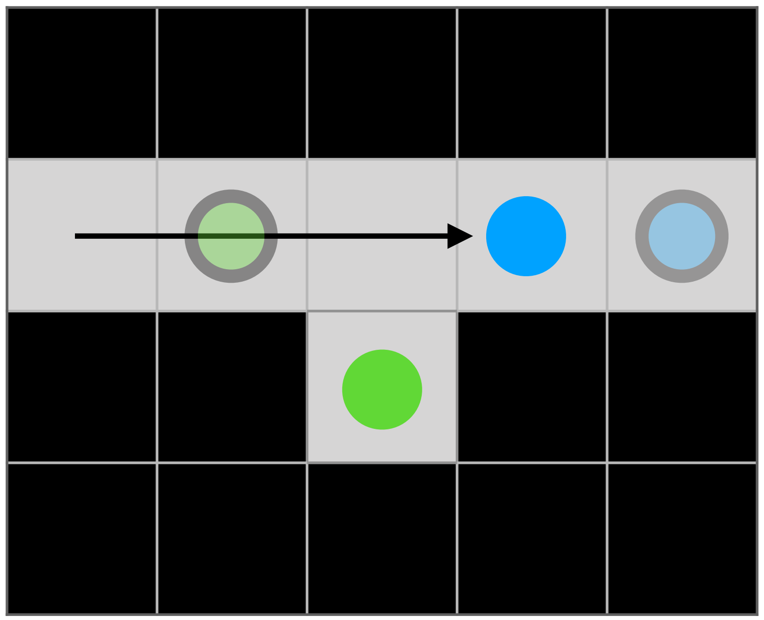
vertex-collision edge-collision no collision

- Performance metrics
  - **Makespan**: time of last robot's arrival time
  - **Flowtime**: sum of arrival times, over all robots

Coupled vs Decoupled Path Planning



Potential deadlock



Completeness achieved.

- Coupled planning provides completeness.
- Decoupled path planning is not complete, in general.

Coupled Path Planning

Coupled formulation:

Robot i has configuration space: $\mathscr{C}_i$

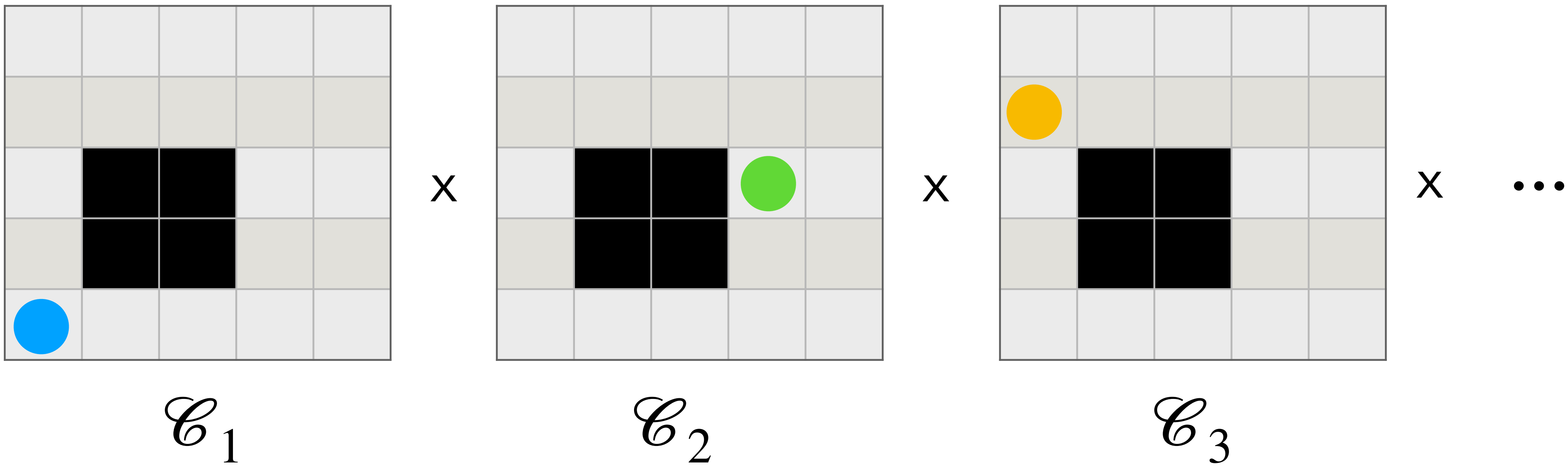
The joint state space is given by the Cartesian product:

$$X = \mathscr{C}_1 \times \mathscr{C}_2 \times \ldots \times \mathscr{C}_n$$

The dimensionality grows **linearly** w.r.t. the number of robots. Complete algorithms (such as A*) require time that is at least **exponential** w.r.t. the search space dimension!

Coupled Path Planning

Coupled formulation for N robots and M cells in grid-world:



For M possible states in each configuration space, we have M^N states in the coupled system.

E.g., worst case complexity for A*: $O(|E|) \approx O(|V|) = O(M^N)$

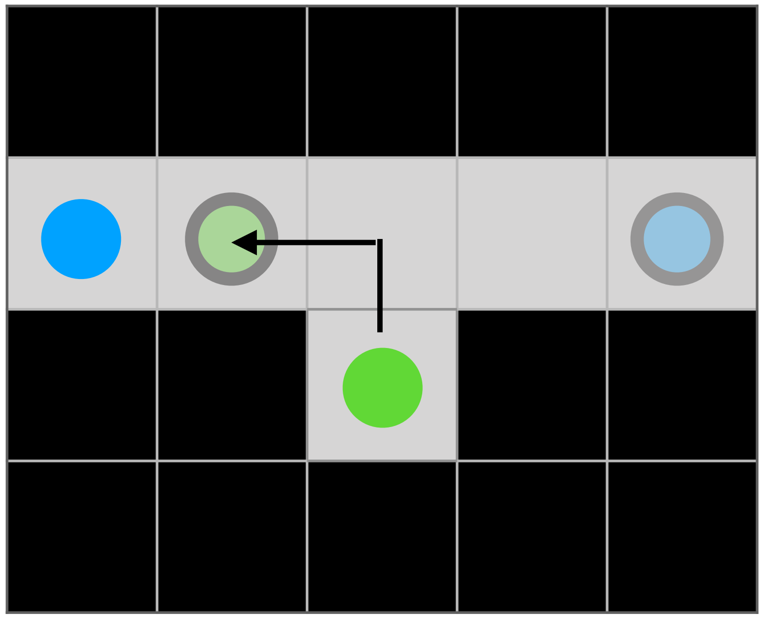
Exponential complexity in the number of robots!

* if graph is sparse

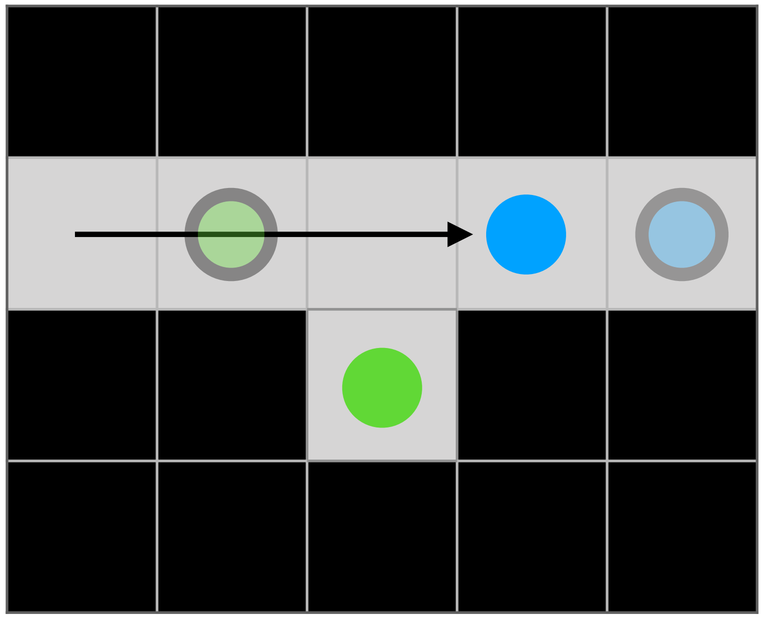
Coupled Path Planning

- Hardness: **NP-hard to solve optimally** for makespan or flowtime minimization [Yu and LaValle; 2013]
- It is impossible to minimize both objectives simultaneously (Pareto)
- But: coupled method provides **completeness** and **optimality**
  - Lots of attention devoted to this field
  - Development of approximate solutions (see literature by Sven Koenig; Howie Choset; Maxim Likhachev)

Coupled vs Decoupled Path Planning



Potential deadlock

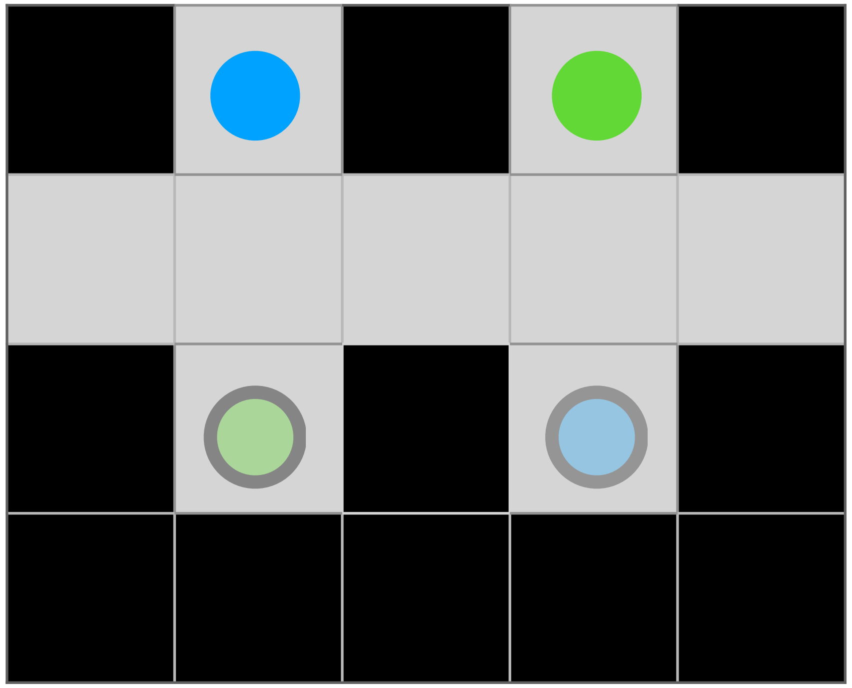


Completeness achieved.

- Decoupled path planning is not complete, in general.
- But: in ***well-formed*** environments, prioritized **decoupled** planning is complete!
  - Well-formed environment: goals are distributed in such a way that any robot standing on a goal cannot completely prevent other robots from moving between any other two goals.

[Cap, Novak, Klaeiner, Selecky; 2015]

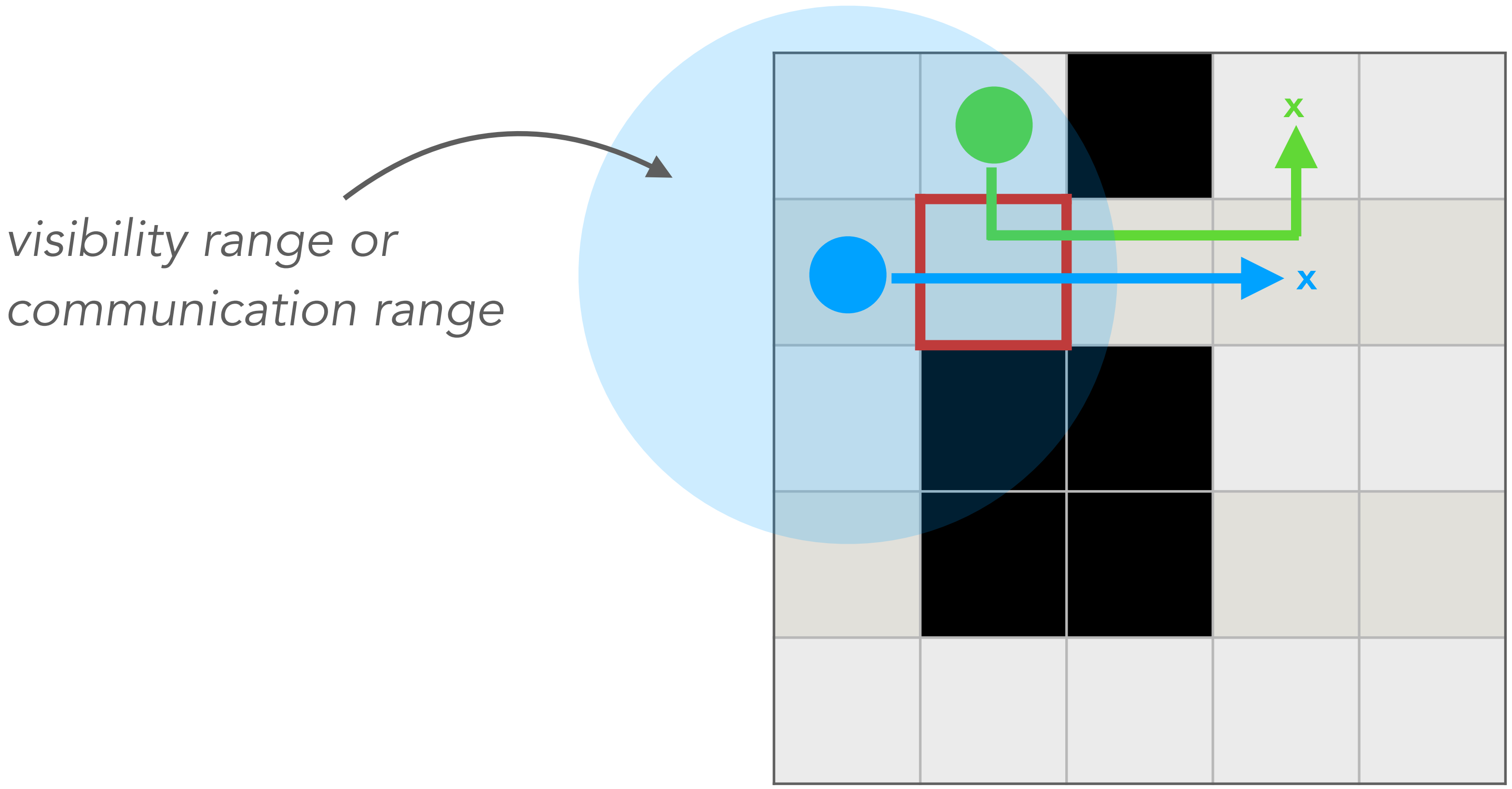
Decoupled Path Planning



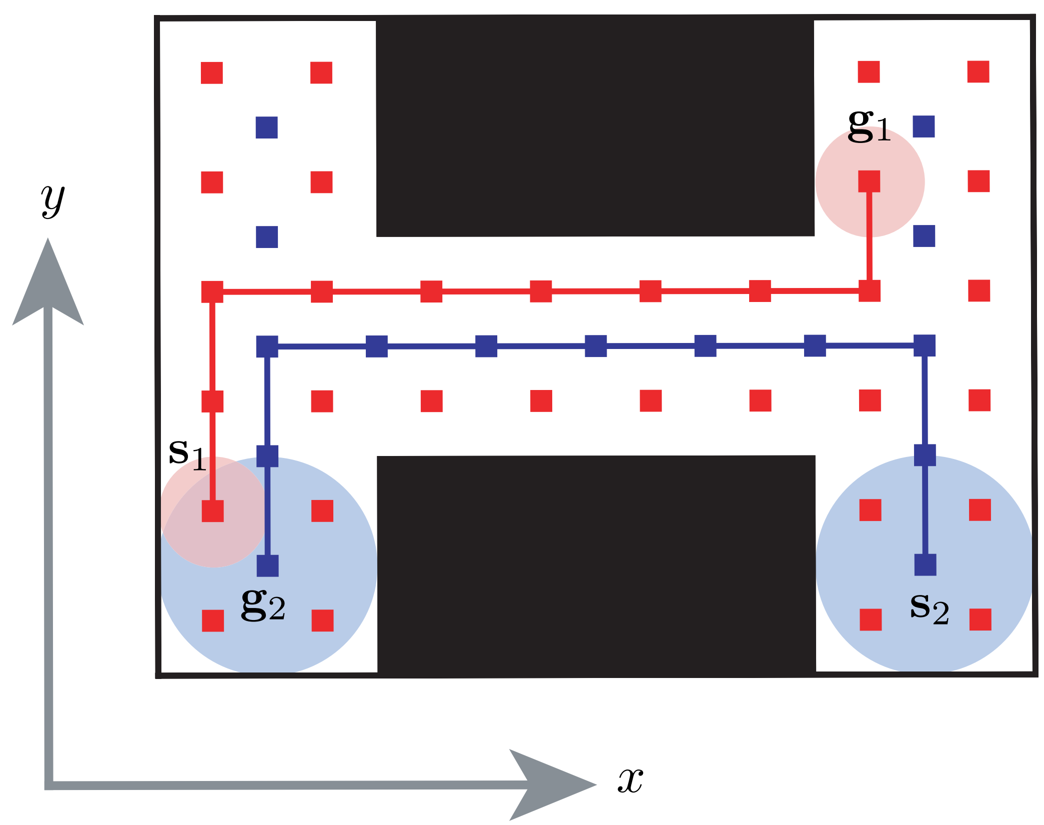
- Well-formed environment:
  - There must exist a path between any two endpoints.
  - That path must have with at least *R*-clearance with respect to static obstacles and at least *2R*-clearance to any other endpoint.
  - A robot is always able to find a collision-free trajectory to its goal by waiting for other robots to reach their goals, and then following a path around those occupied goals (any prioritization works!).

Decoupled Path Planning

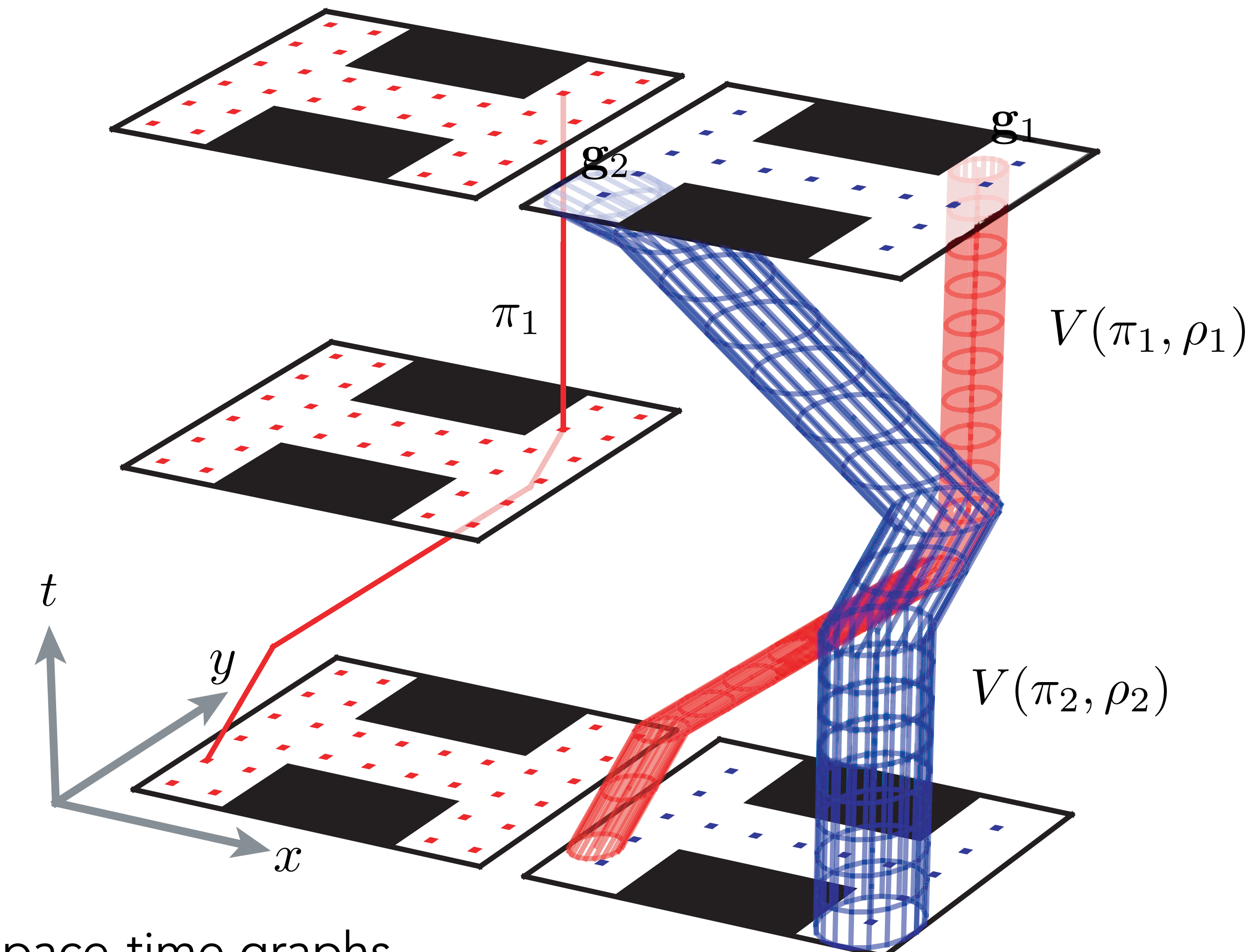
- De-coupling the problem:
  - Each robot plans in its own space-time
  - Robots negotiate path plans as conflicts arise
  - De-confliction can be online (dynamic) or offline (a-priori)



Decoupled, Prioritized Path Planning



Ideal trajectories for 2 robots



Space-time graphs

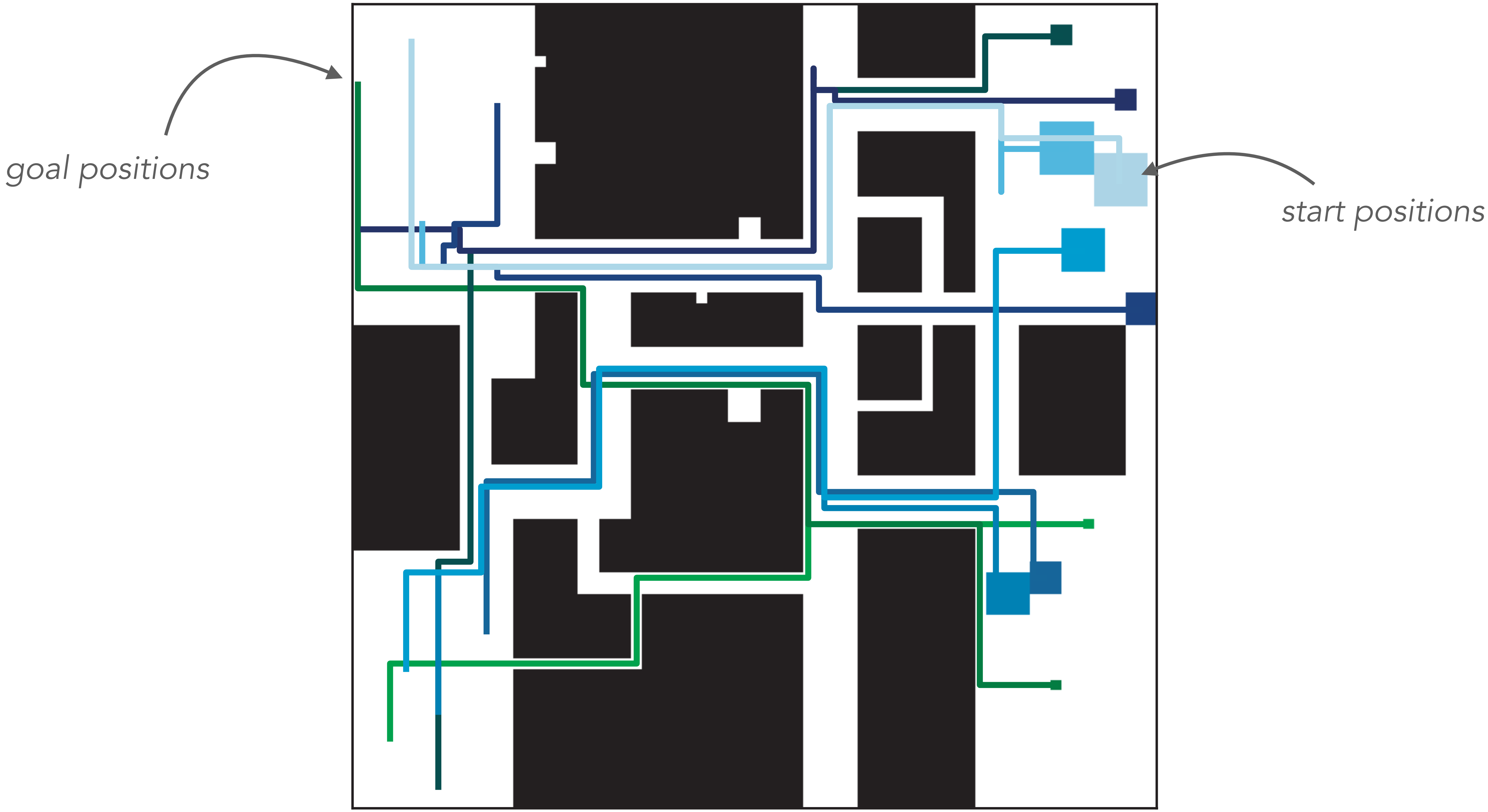
The red robot is prioritized and plans a space-time path that is optimal.
The blue robot plans a path that does not collide with the red robot's path.

[Wu, Bhattacharya, Prorok]

Decoupled, Prioritized Path Planning

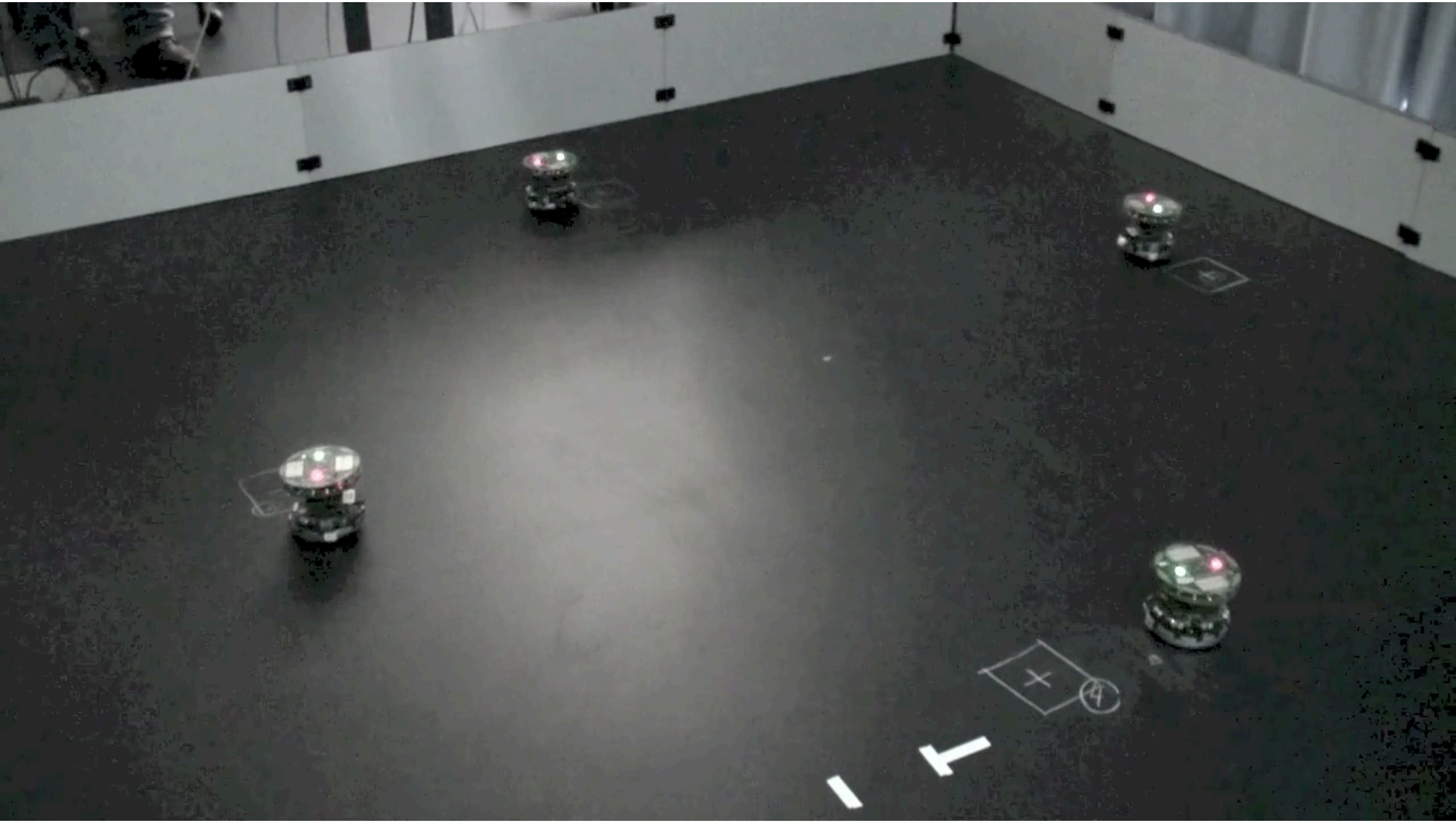
- Key question: How to prioritize robots?
- Online, exhaustive method:
  - Evaluate all $N!$ options (where N is robots within communication or visibility neighborhood) [Azarm, Schmidt; 1997]
- Existing **prioritization heuristics** (online and offline):
  - <u>Ideal path length</u>: Robots with longer ideal path length have higher priority. [Van den Berg et al.]
  - <u>Planning time</u>: Robots that take longer to plan their paths get higher priority. [Velagapudi, Sycara, Scerri; 2010]
  - <u>Workspace clutter</u>: Robots with more clutter in local vicinity have higher priority. [Clark, Bretl, Rock; 2002]
  - <u>Path prospects</u>: Robots with fewer path options have higher priority [Wu, Bhattacharya, Prorok; 2019]

Decoupled, Prioritized Path Planning



Example of a multi-agent system where agents have heterogeneous sizes.
Agents with fewer path prospects are prioritized.

The Continuous Domain

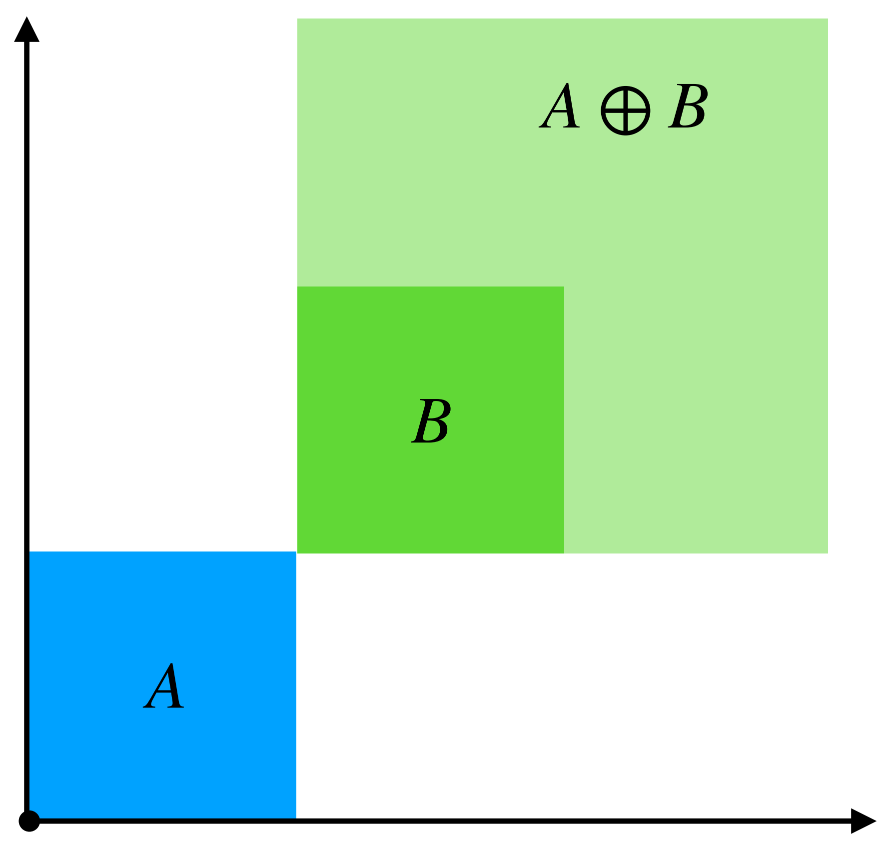


*movie credit: Gowal, Martinoli

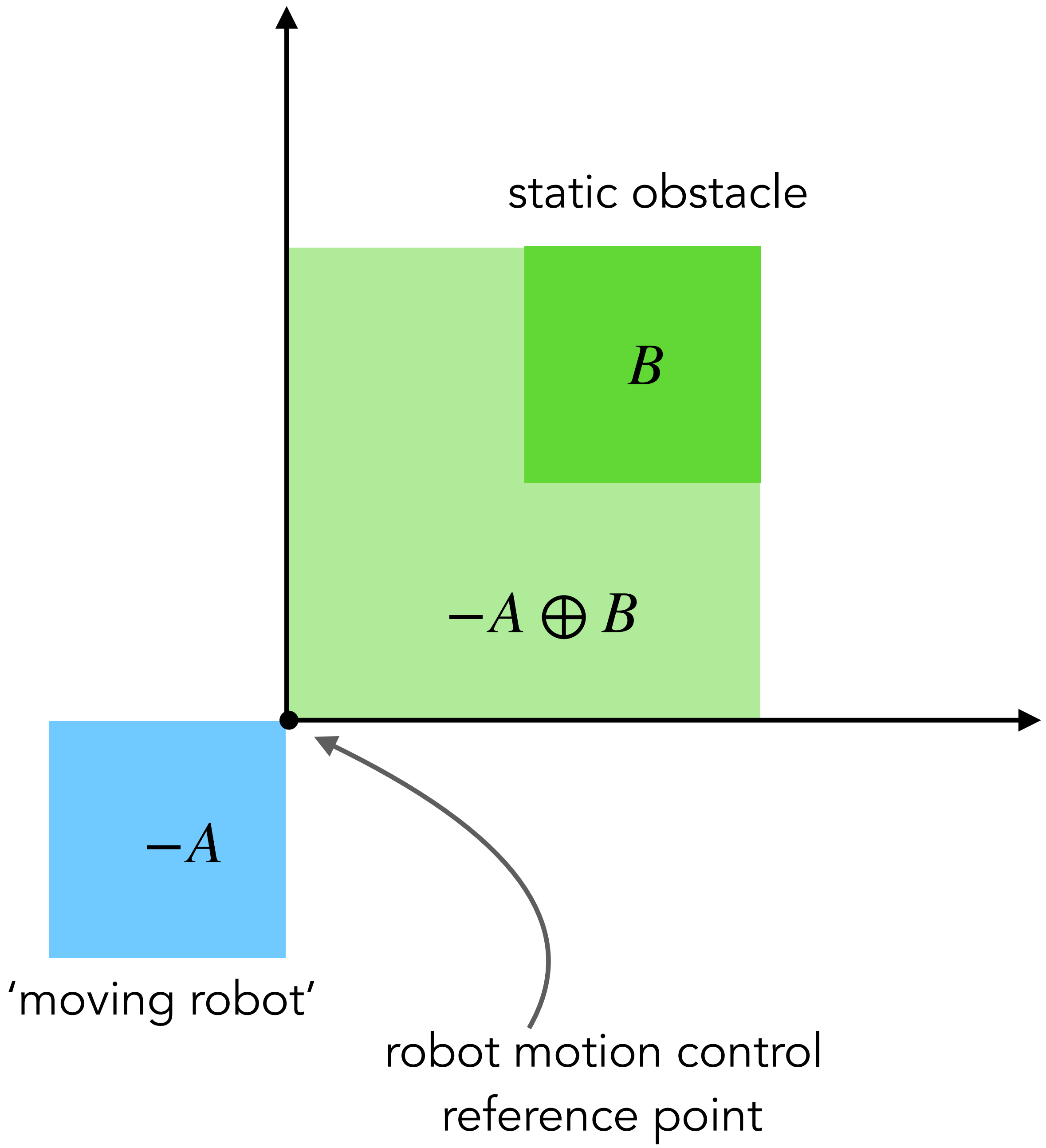
Minkowski Sum (Reminder)

- In geometry, the Minkowski sum (also known as dilation) of two sets of position vectors $\boldsymbol{A}$ and $\boldsymbol{B}$ in Euclidean space is formed by adding each vector in $\boldsymbol{A}$ to each vector in $\boldsymbol{B}$, i.e., the set:

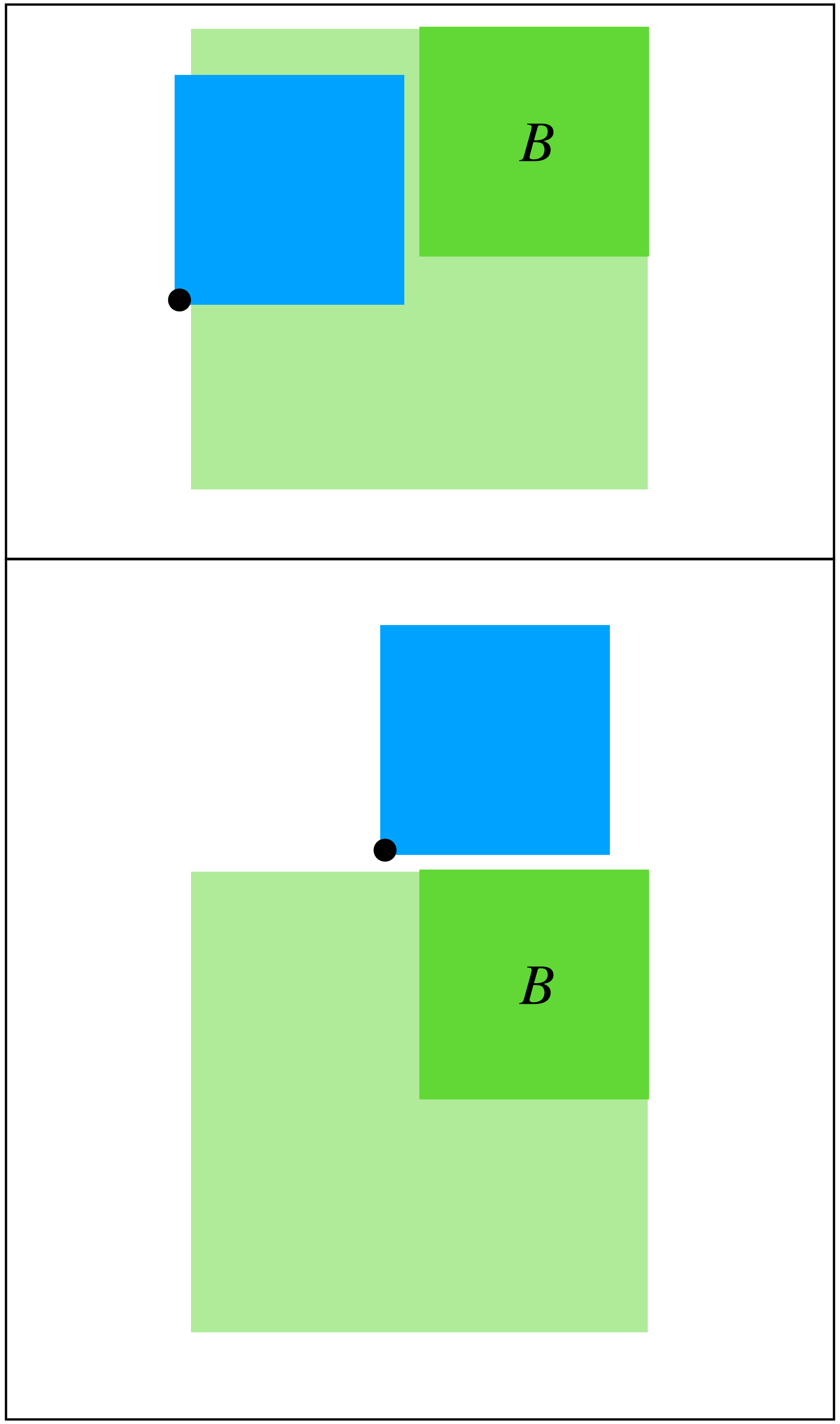
$$A \oplus B = \{\mathbf{a} + \mathbf{b} \,|\, \mathbf{a} \in A, \mathbf{b} \in B\}$$



Minkowski Sum (Reminder)

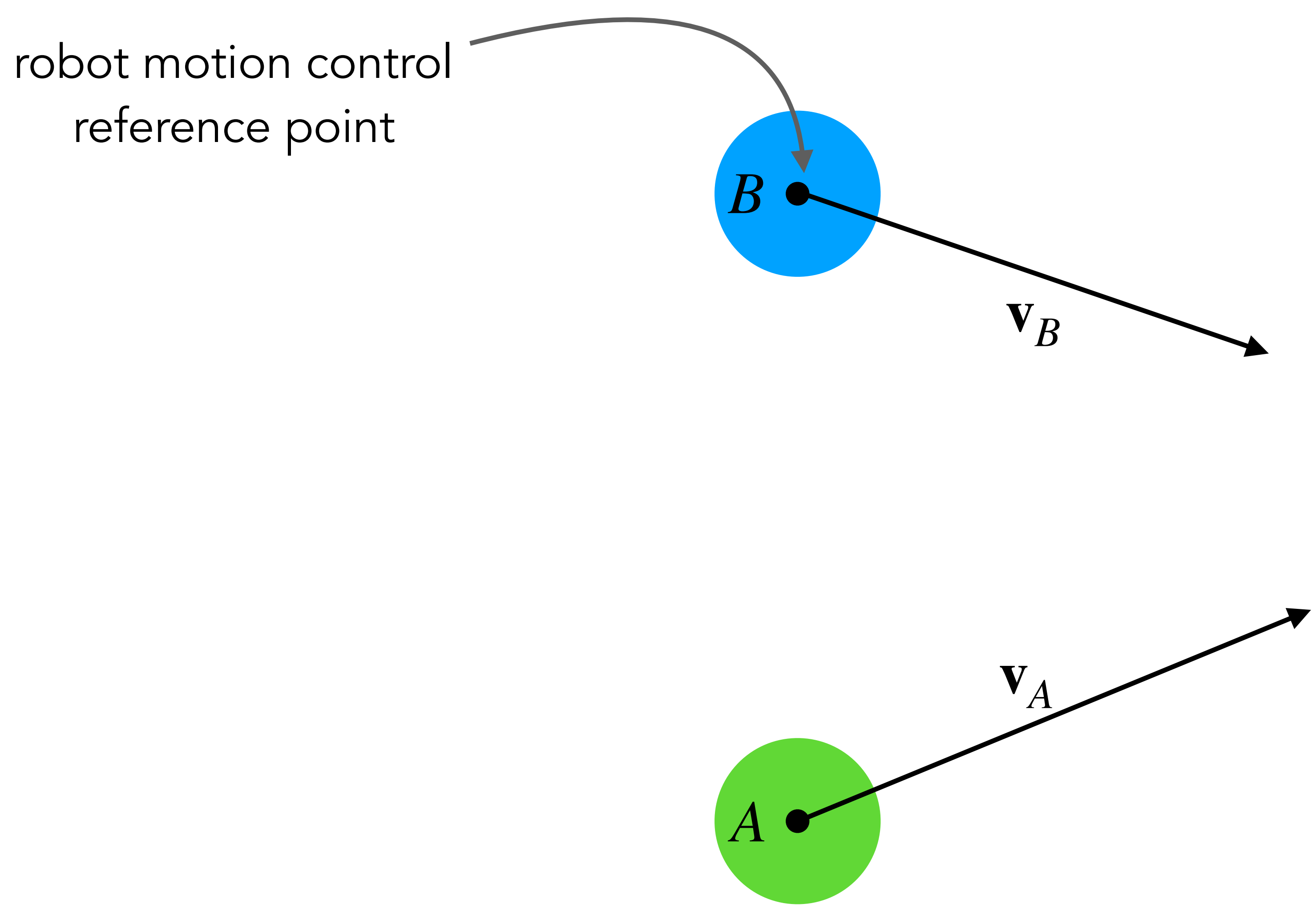


As long as reference point stays outside dilated area, there will be no collisions.



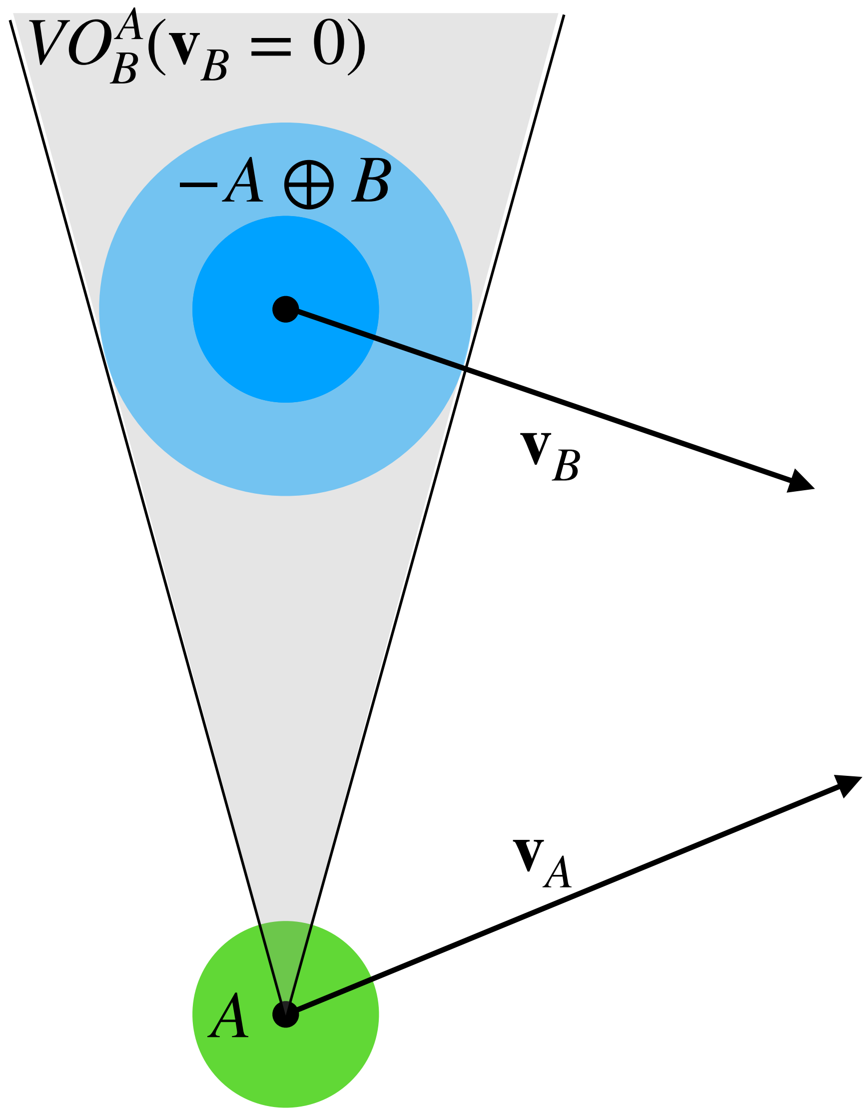
Velocity Obstacle Method

[Fiorini, Shiller; 1998]



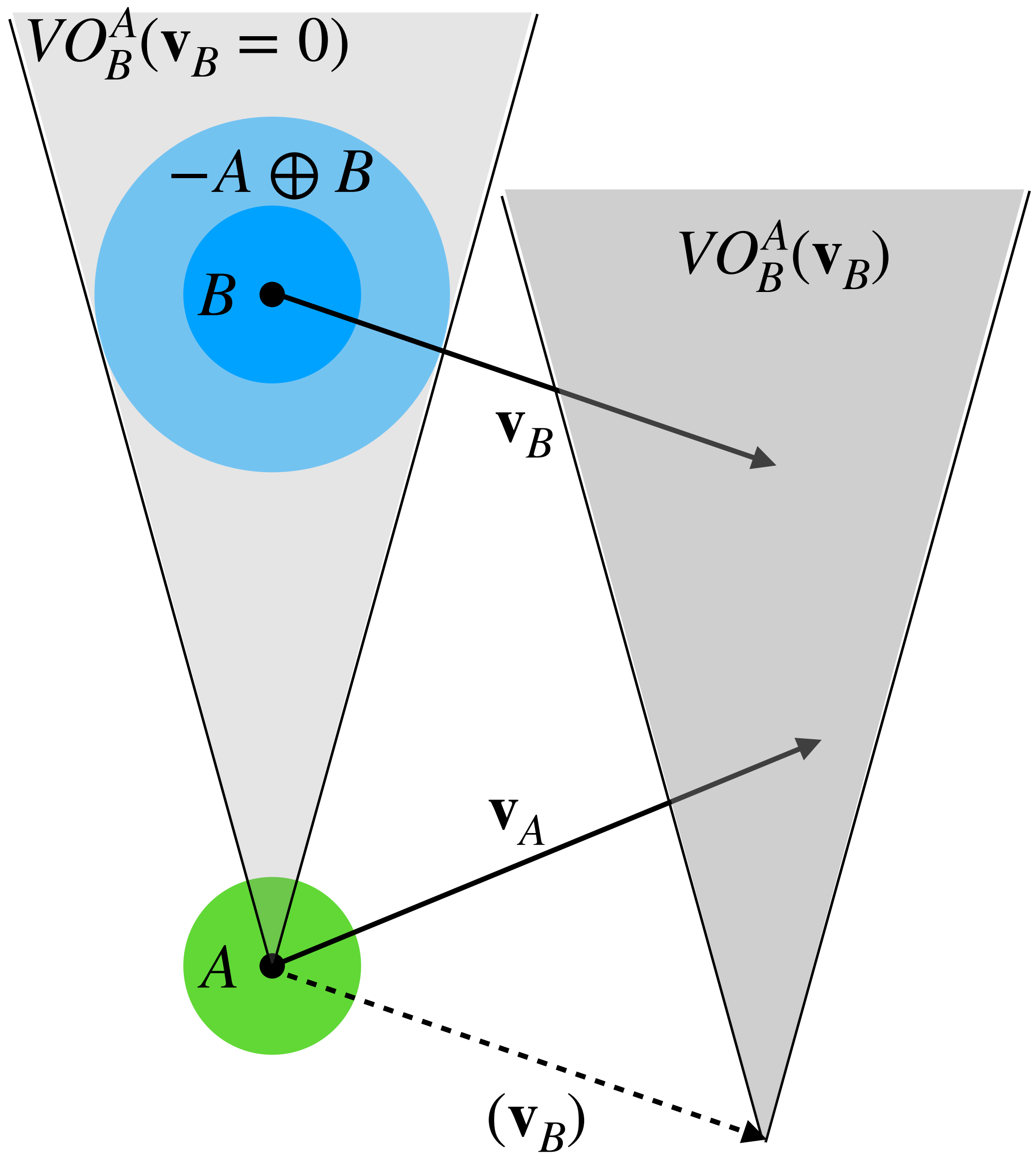
Two robots, A and B, translating in space. Will they collide?

Velocity Obstacle Method



Two robots, A and B, translating in space. Will they collide?
Step 1: inflate robot B by area of robot A.

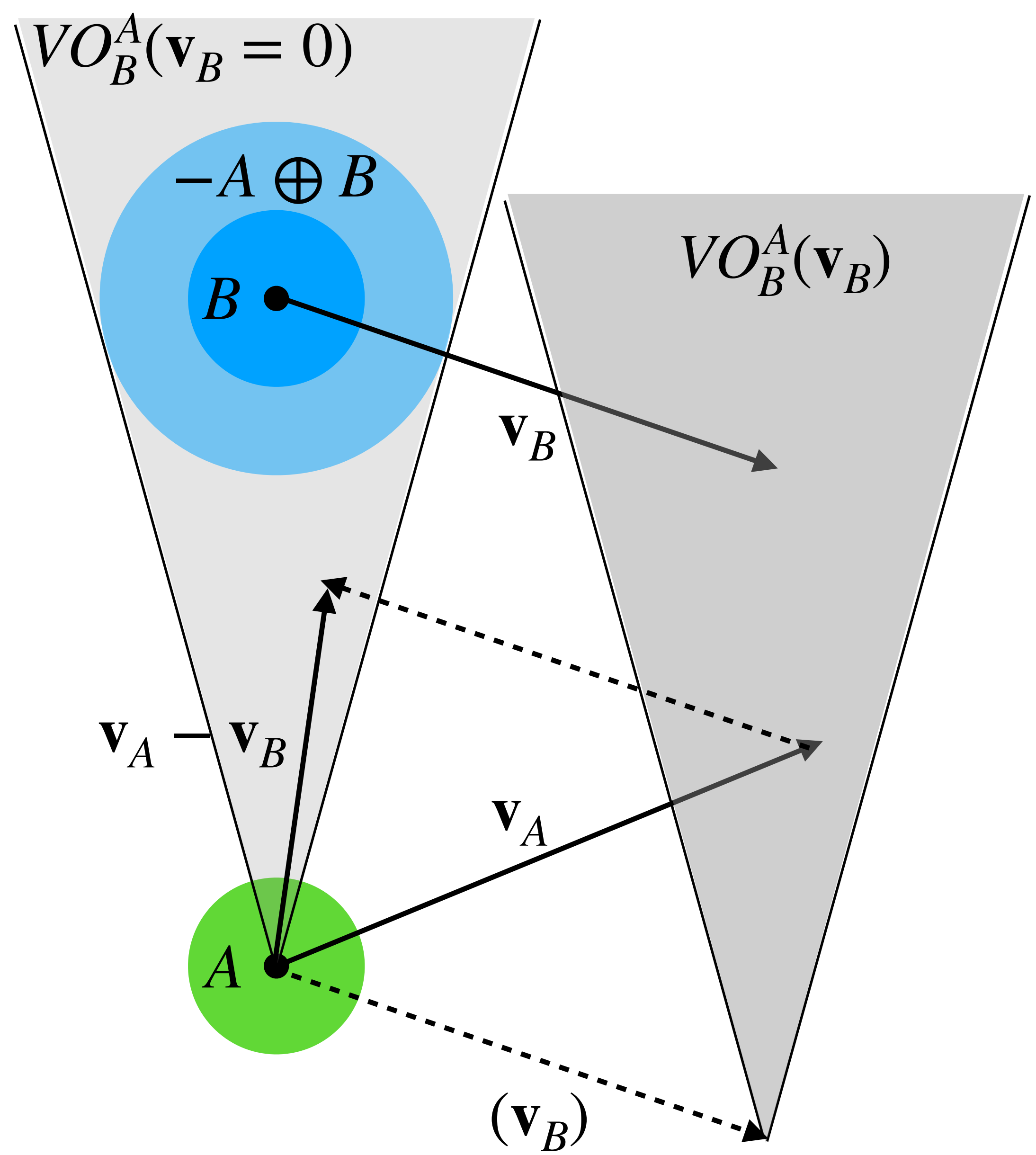
Velocity Obstacle Method



Step 2: determine whether $\boldsymbol{v}_A$ lies in the velocity obstacle of B to A

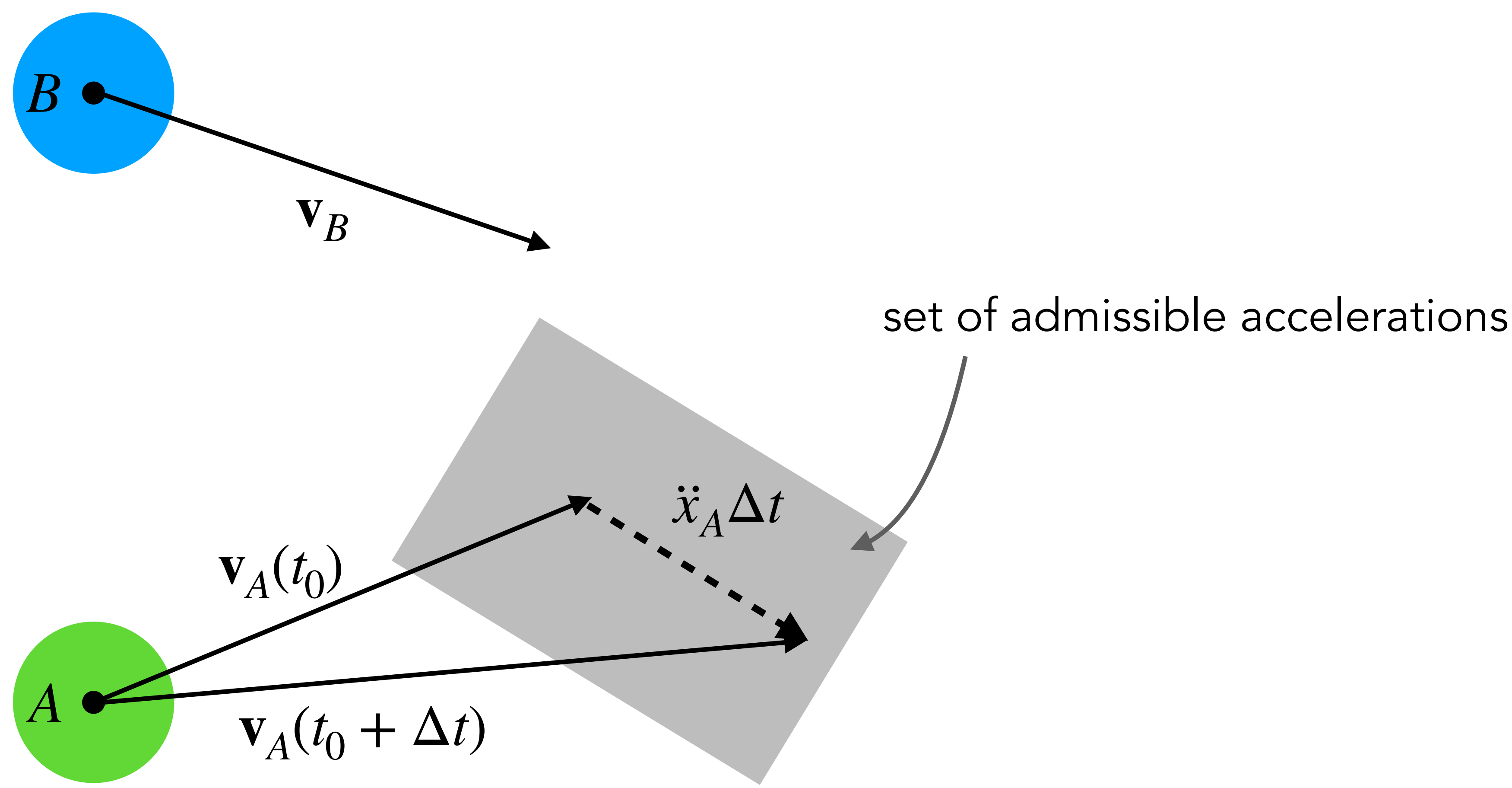
If $\boldsymbol{v}_A$ is outside the VO, then the robots will never collide.

Velocity Obstacle Method



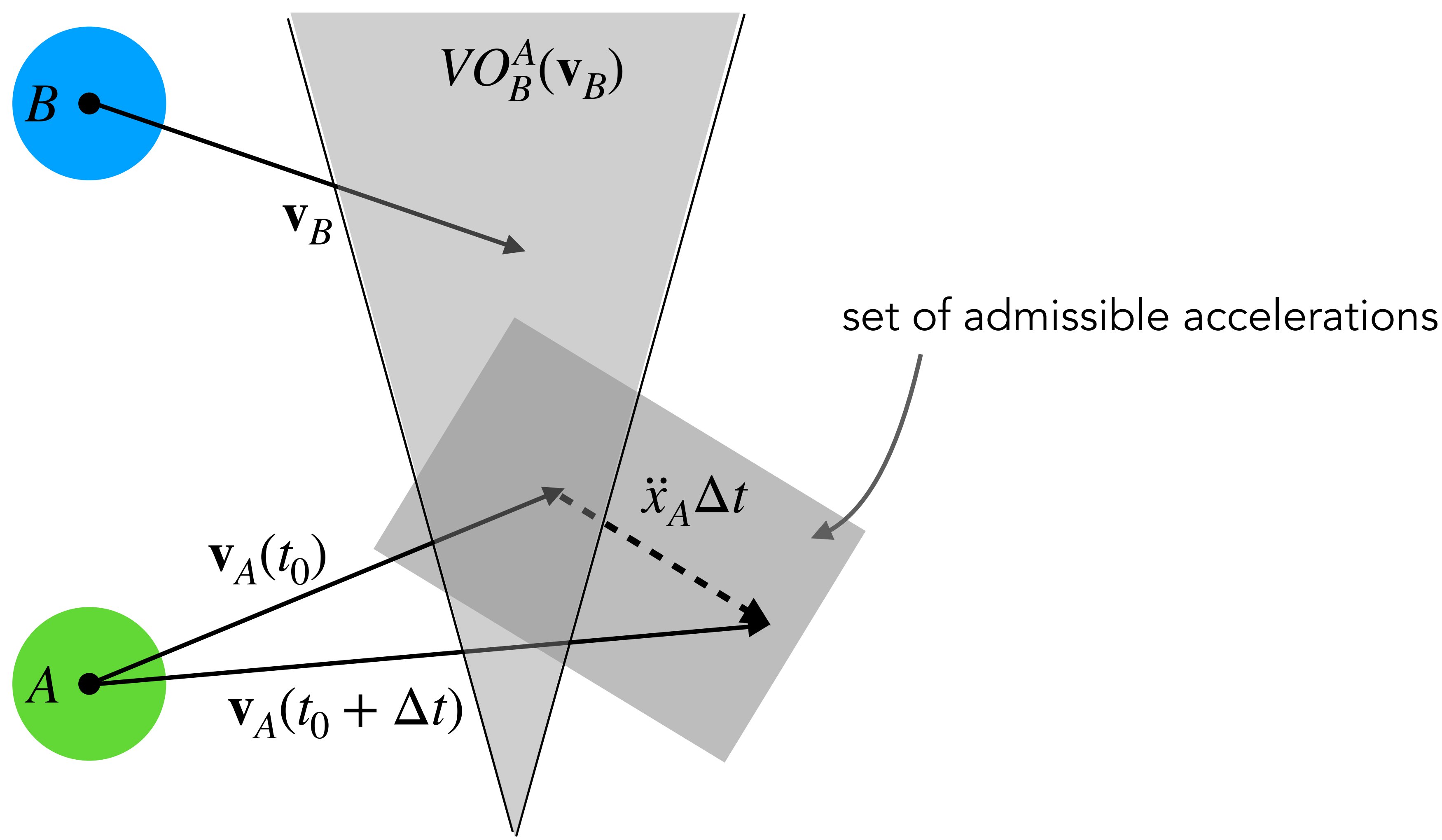
Equivalence: $\boldsymbol{v}_A$ lies in the velocity obstacle of B to A $\longrightarrow$ the relative velocity $\boldsymbol{v}_A$ - $\boldsymbol{v}_B$ lies in the velocity obstacle of B to A, assuming B does not move.

Velocity Obstacle Method



Compute set of admissible accelerations for robot A.

Velocity Obstacle Method



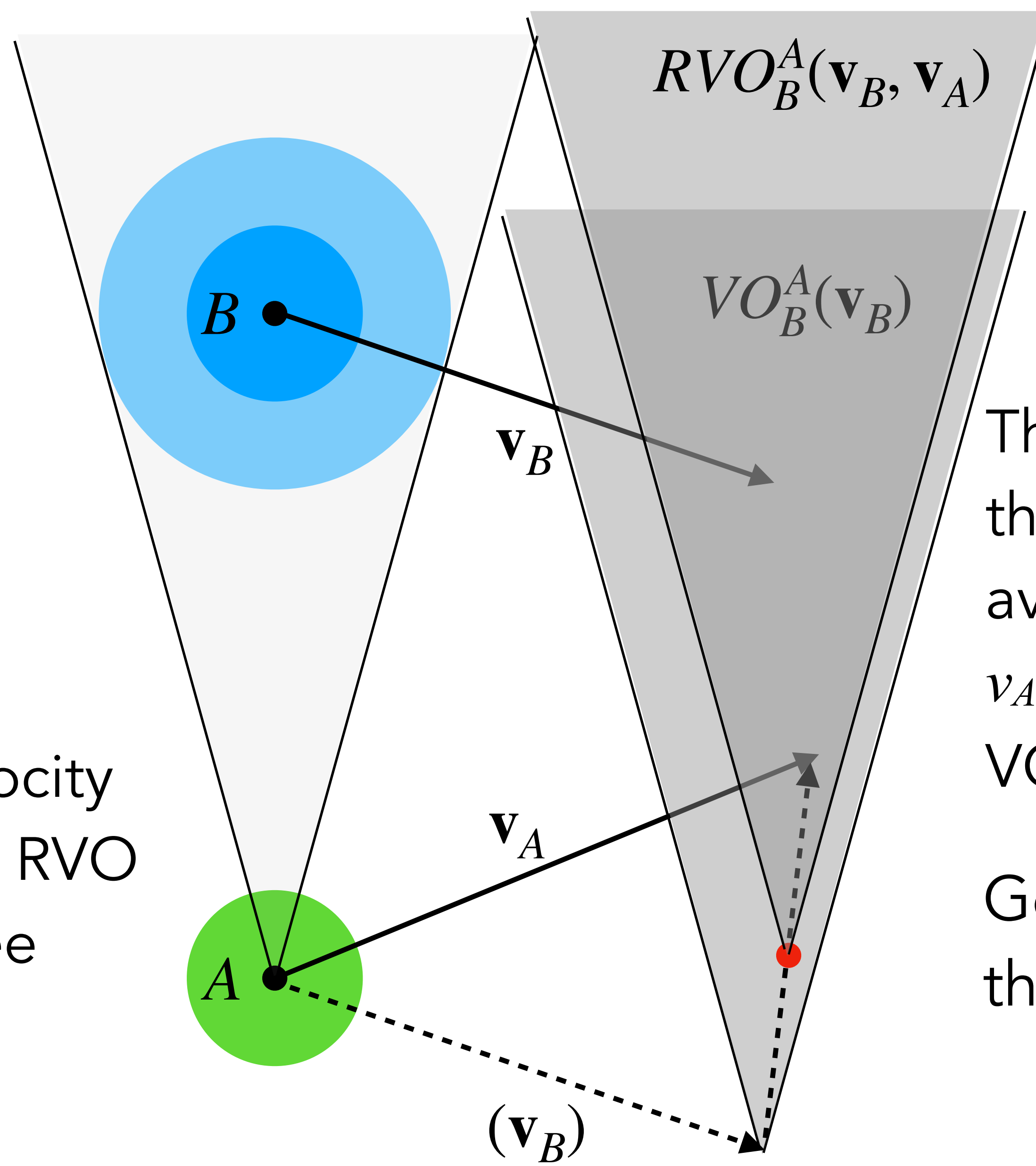
Check that new velocity is outside VO.

Velocity Obstacle Method

- Assumptions:
  - Robots share their current (noise-free) position and velocity
  - Robots truthfully execute reported velocities
- Complications:
  - **Oscillations**! Scenario: Robots with current velocities $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$ currently lie in each others VOs. **Both** robots select new $\boldsymbol{v}'_A$ and $\boldsymbol{v}'_B$ such that new velocities lie outside respective VOs. In new situation, the old velocities $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$ lie outside VOs. If $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$ are **preferable** (e.g., they lie on direct path to goal), they will be chosen again, hence, leading to oscillations.
  - Solution: See *reciprocal velocity obstacle* method.

Reciprocal Velocity Obstacle Method

Idea: Choose a new velocity that is the average of its current velocity and a velocity that lies outside the other agent's velocity obstacle. [Van den Berg, Lin, Manocha; 2008]



The RVO of B to A contains all the velocities of A that are the average of the current velocity v_A and a velocity inside the VO of B to A.

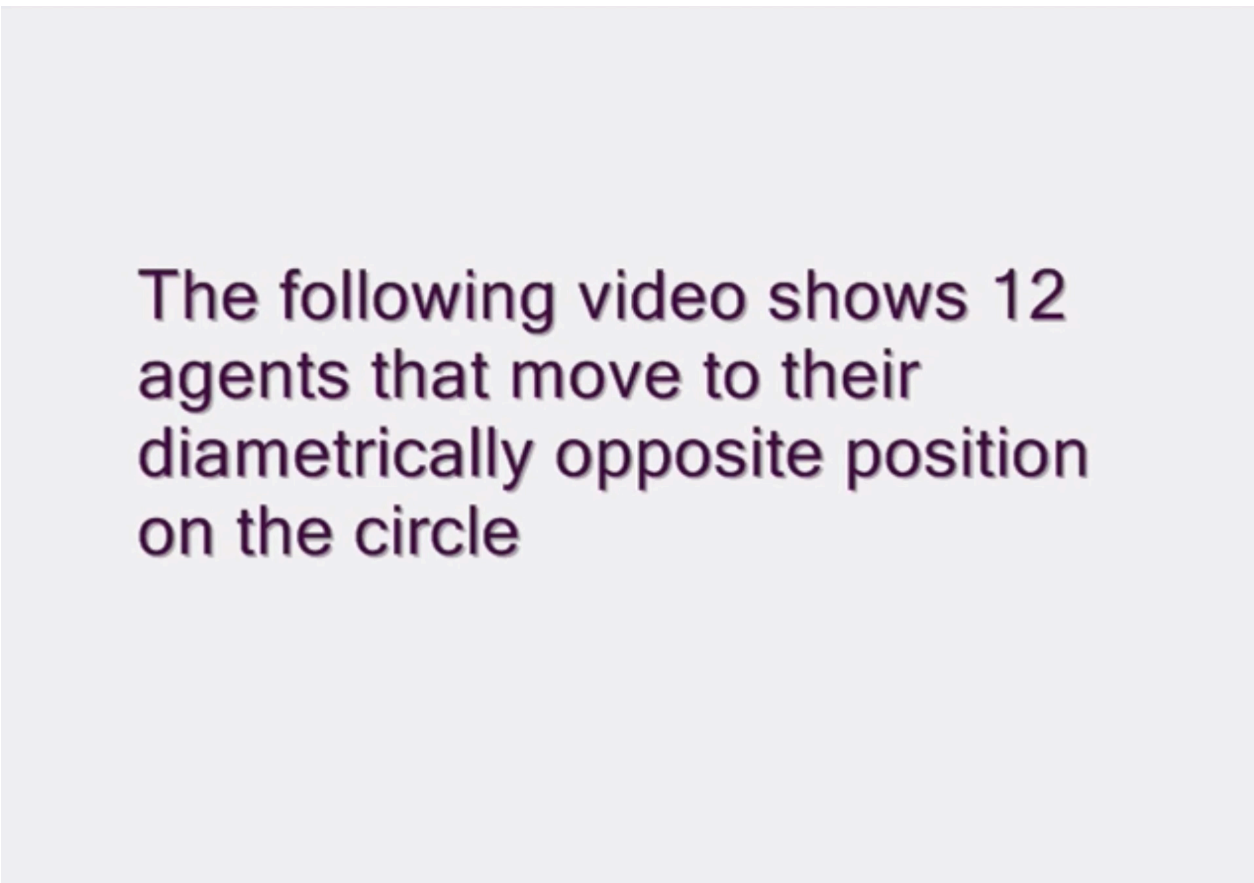
Choosing the closest velocity outside the other agent's RVO guarantees oscillation-free navigation.

Geometric interpretation: the apex of the RVO lies at:

$$\frac{\mathbf{v}_A + \mathbf{v}_B}{2}$$

The old velocity of A is inside the new RVO of B to A, given the new velocities.

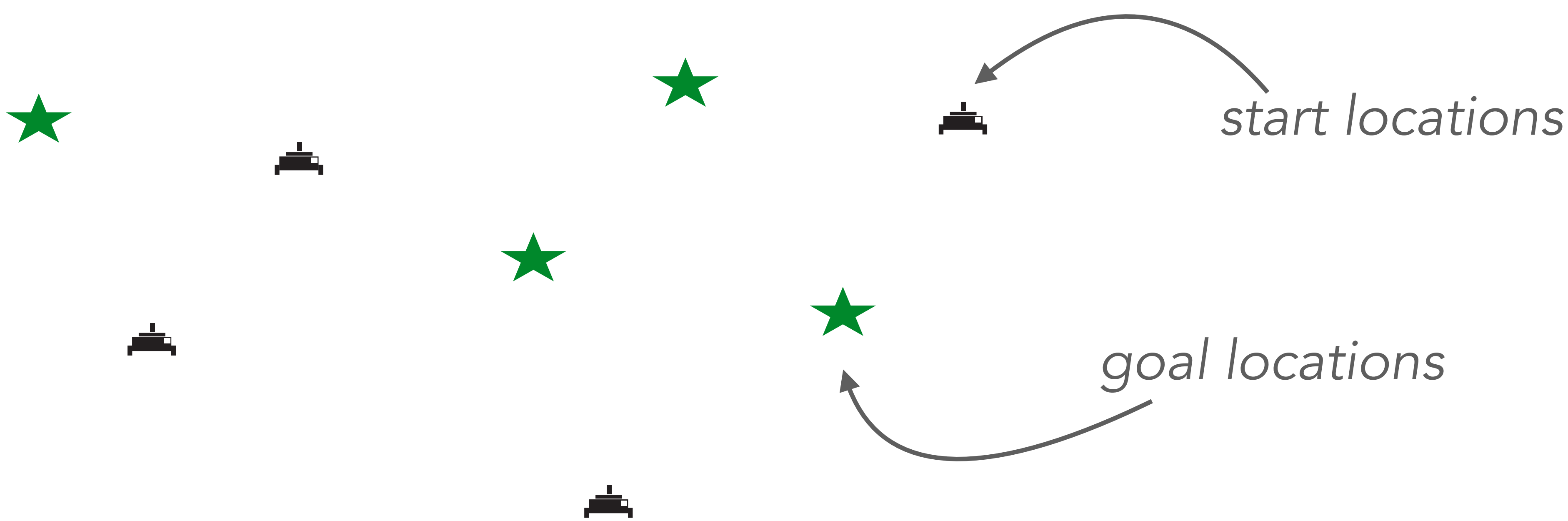
Reciprocal Velocity Obstacle Method



[D. Manocha et al.]

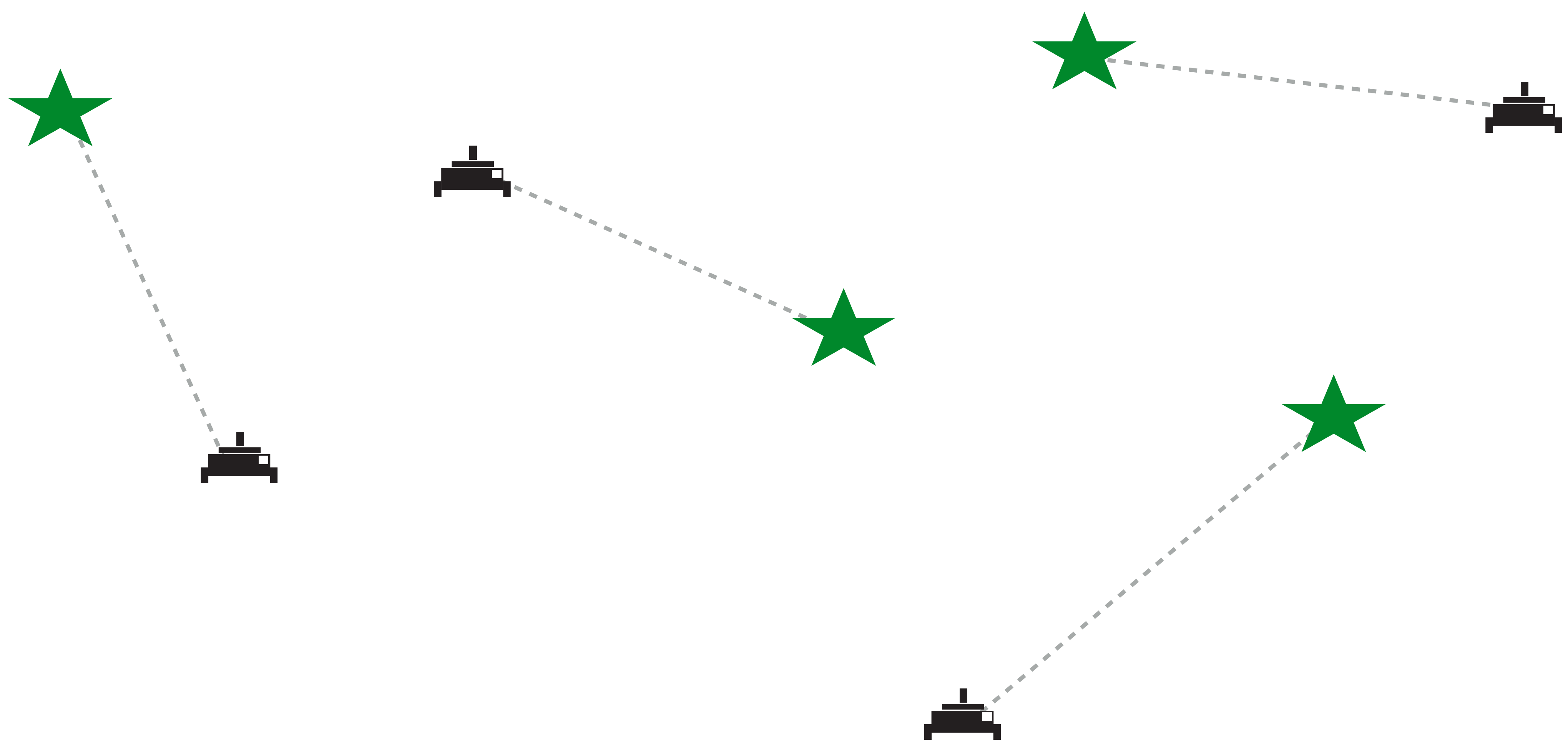
Concurrent Assignment and Planning of Trajectories

- New problem formulation:
  - N robots need to reach N goal locations as efficiently as possible: we want to find the **assignment** as well as **generate the trajectories,** simultaneously.
  - Un-labeled problem (any robot may go to any goal)
  - Robots must have collision-free trajectories
- Assumptions:
  - Robots have a **minimum separation distance** at start / goal locations
  - Robots are **holonomic** and arrive **simultaneously** at goals



Concurrent Assignment and Planning of Trajectories

Given start and goal locations, find assignments **AND** trajectories that are optimal and collision-free



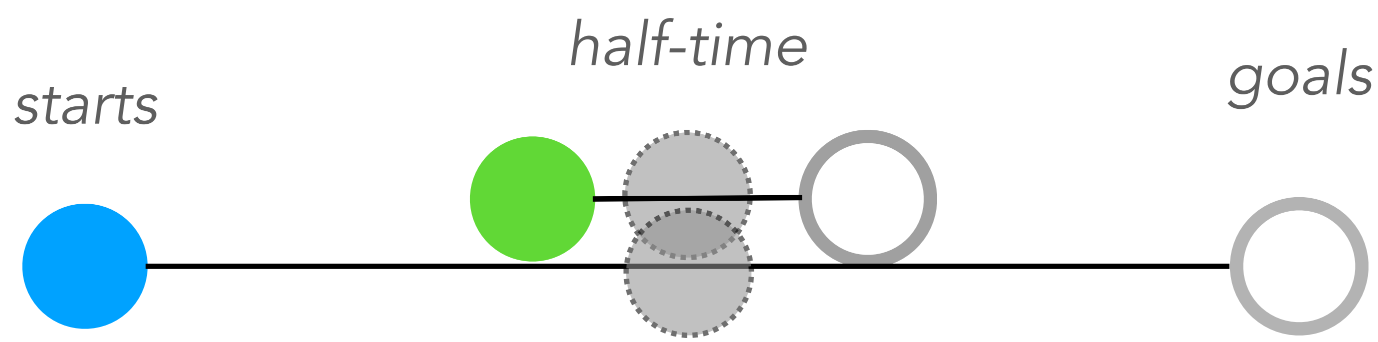
Concurrent Assignment and Planning of Trajectories

Given start and goal locations, find assignments **AND** trajectories that are optimal and collision-free

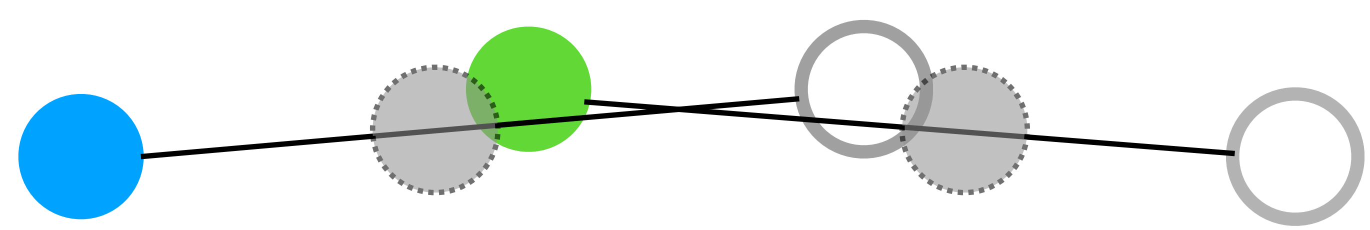
Concurrent Assignment and Planning of Trajectories

What is the **optimization objective**?

Sum of distances:



Sum of distances squared:



[Turpin et al.; IJRR 2013]

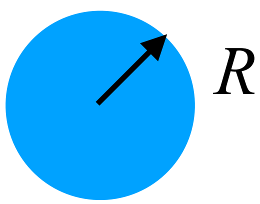
Concurrent Assignment and Planning of Trajectories

Objective:

$$\underset{\phi,\gamma(t)}{\text{minimize}} \quad \sum_{i=1}^{N} \int_{t_0}^{t_f} \dot{\mathbf{x}}_i(t)^{\mathrm{T}} \dot{\mathbf{x}}_i(t)\,dt$$

Key result:

If separation distance between any start and goal location is $\Delta > 2\sqrt{2}R$ we can guarantee collision-free trajectories.

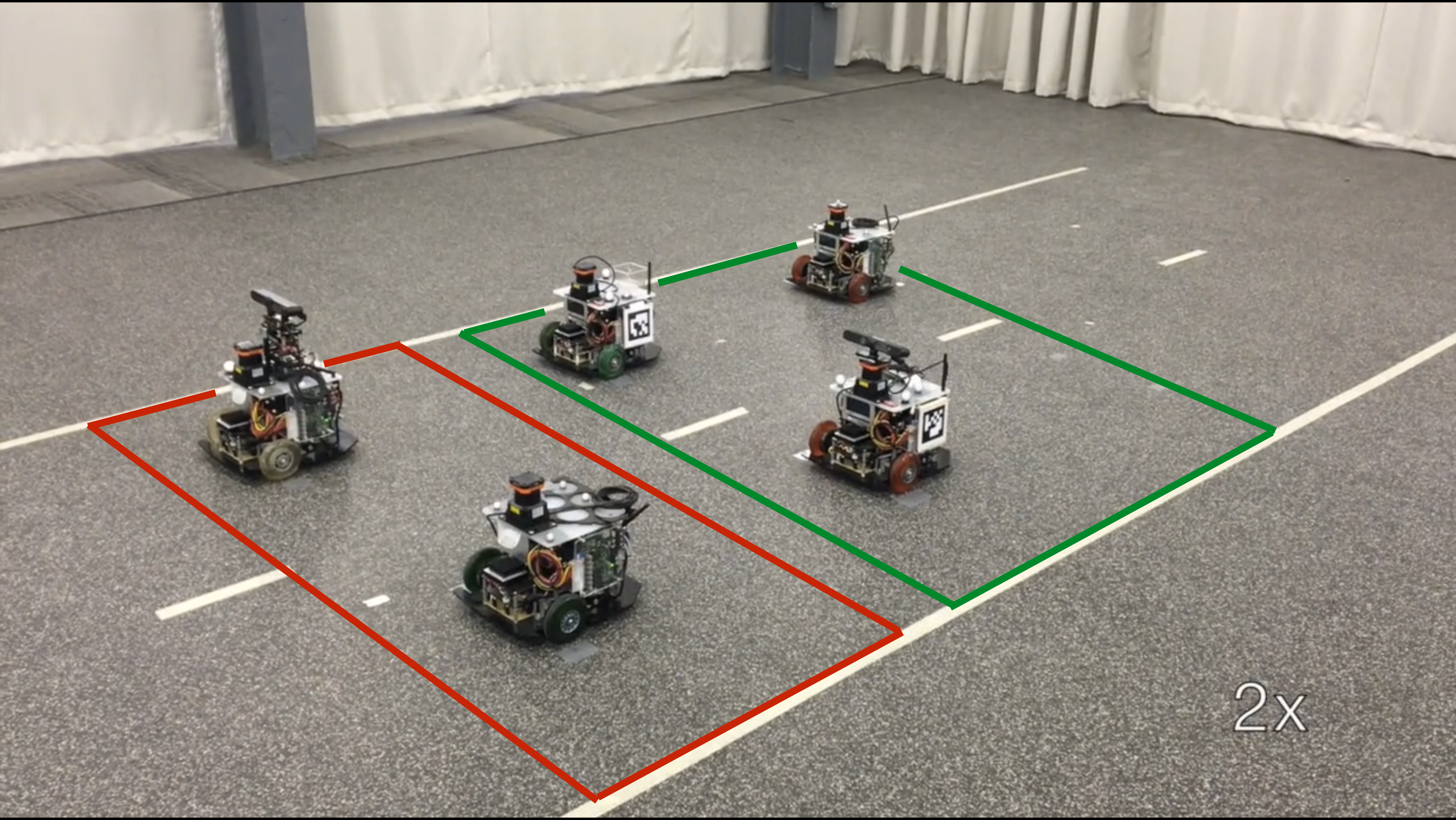


Solve assignment:

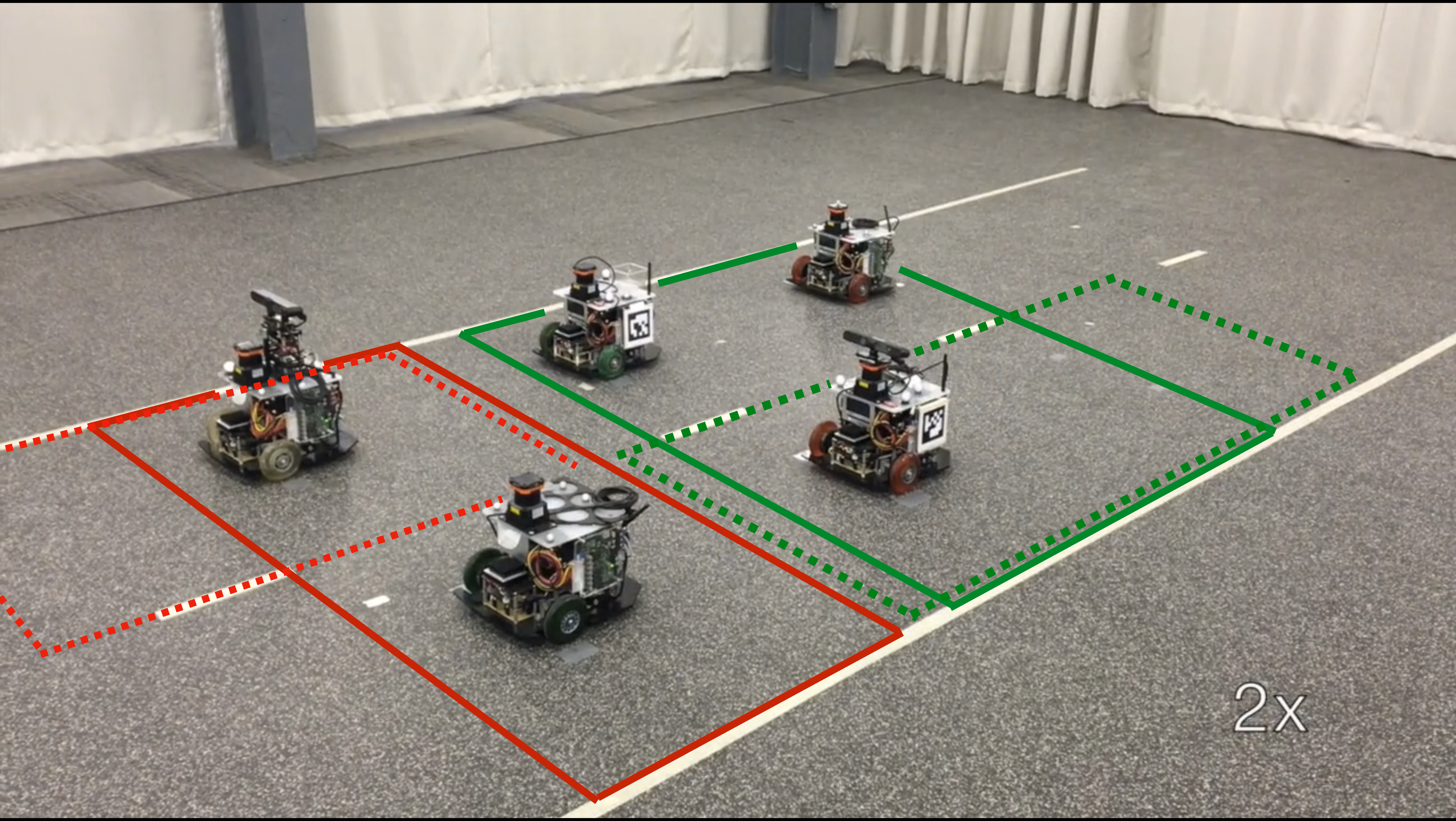
$$\phi^{\star} = \underset{\phi}{\text{argmin}} \quad \sum_{i=1}^{N} \sum_{j=1}^{N} \phi_{i,j} D_{i,j}$$

cost: distance squared

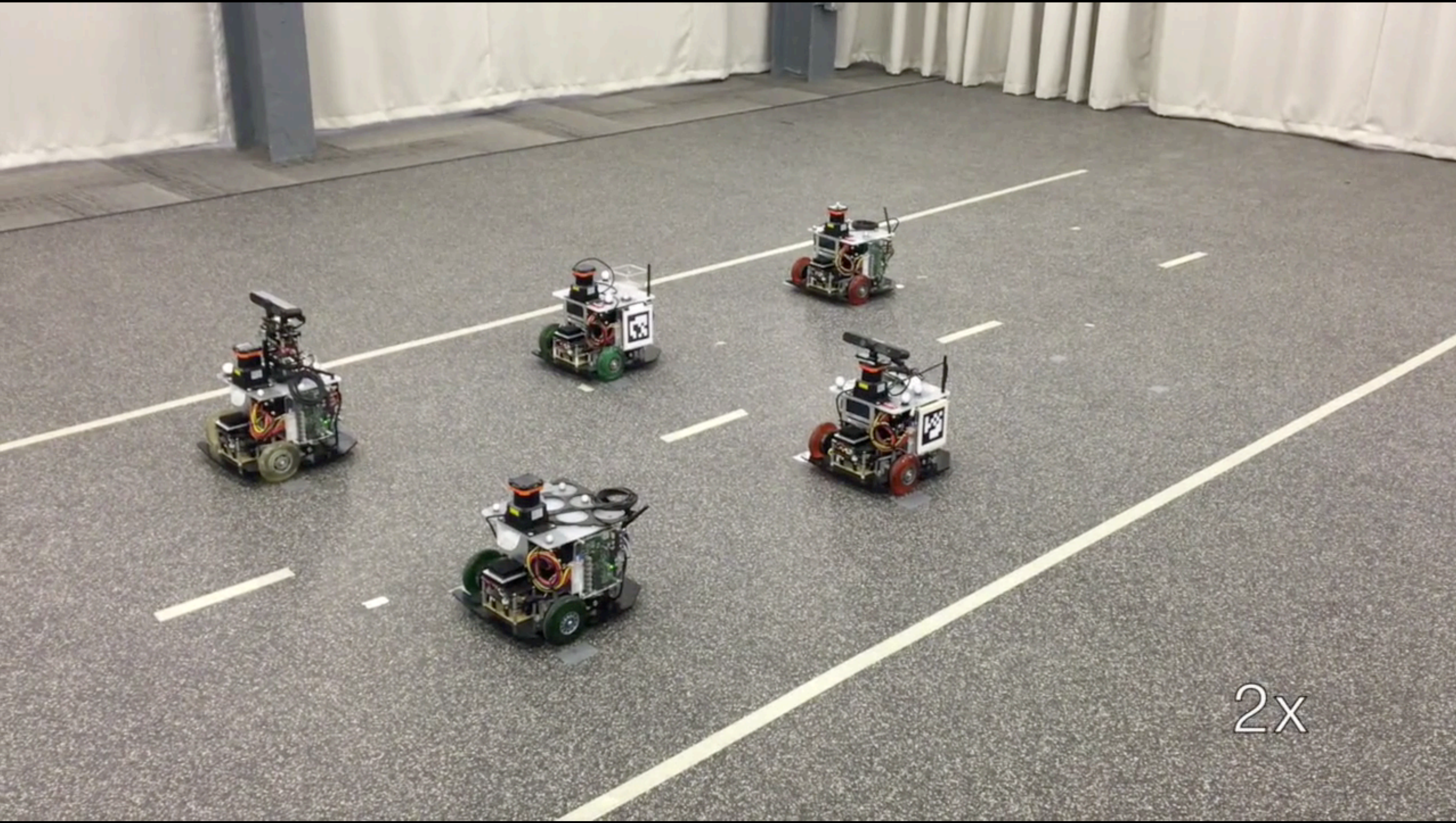
[Turpin et al.; IJRR 2013]



[Whitzer, Kennedy, Prorok, Kumar; 2016]



[Whitzer, Kennedy, Prorok, Kumar; 2016]



[Whitzer, Kennedy, Prorok, Kumar; 2016]

Further Reading

Fundamental planning concepts:

- Some of the planning concepts in Steven LaValle’s book.

Seminal papers:

- P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles”; 1998
- J. van den Berg, M. Lin, D. Manocha; “Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation”; 2008
- J. Van Den Berg, M. Overmars. "Prioritized motion planning for multiple robots." 2005

More recent papers:

- M. Turpin, N. Michael and V. Kumar; “CAPT: Concurrent assignment and planning of trajectories for multiple robots”; IJRR 2013
- M. Čáp, P. Novák, A. Kleiner, M. Selecký; “Prioritized Planning Algorithms for Trajectory; “Coordination of Multiple Mobile Robots”; 2015