Learned Collision Avoidance Policy for Distributed Multi-Robot Navigation in Warehouses

Jia Pan

City University of Hong Kong

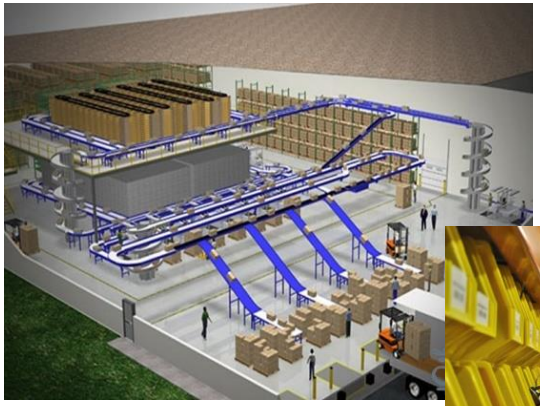
香港城市大學
City University of Hong Kong

dorabot


Problems we try to solve

Different choices for warehouse automation

Number of Items Per Goods Type

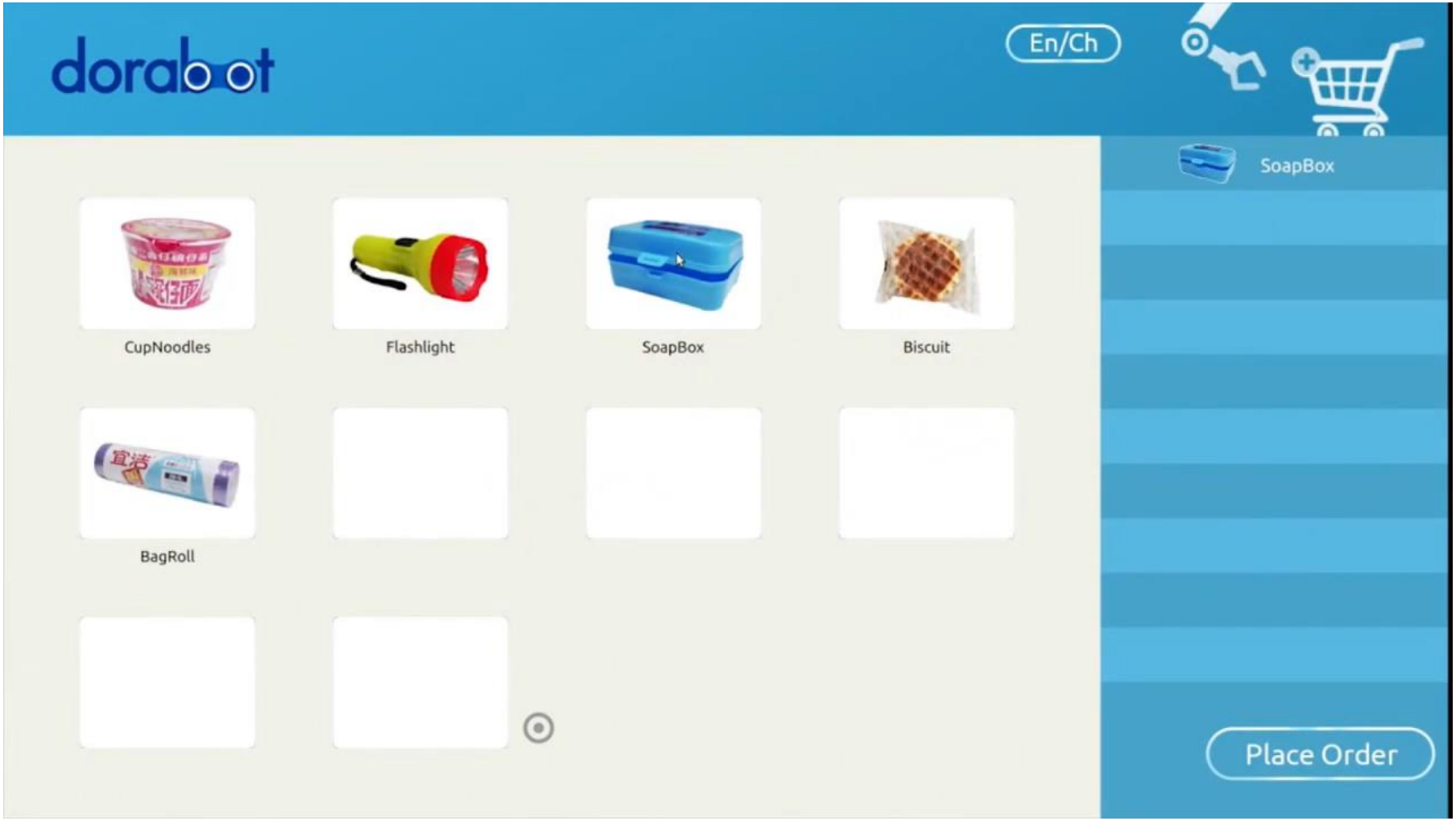


Number of Goods Types

Autonomous agents

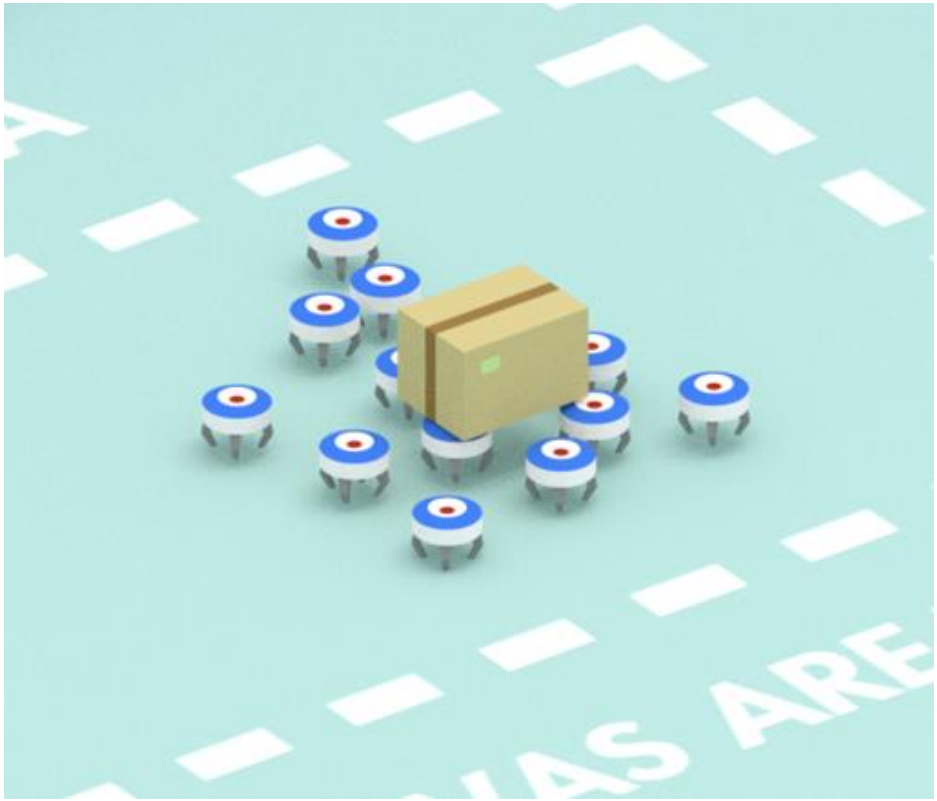
Autonomous agents

Overview


doraboot
En/Ch
SoapBox
CupNoodles
Flashlight
SoapBox
Biscuit
BagRoll
Place Order

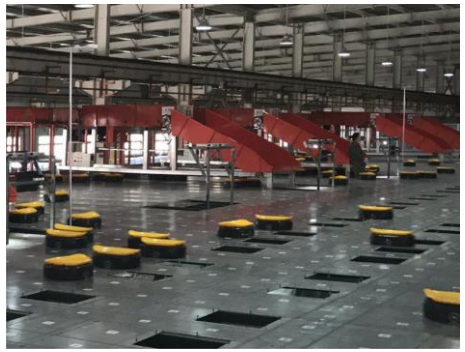
Multi-agent navigation challenge

- How to enable a **large number** agents **efficiently** work **together** in a warehouse?



Centralized solutions

- Using barcode tracks
- Centralized or distributed task / goal assignments
- Mostly centralized path planning for all agents



Kiva

Similar systems: Geek+, HKVISION, etc.

Significant changes to the scenario ☹

Not energy efficient ☹

A centralized scheduler ☹

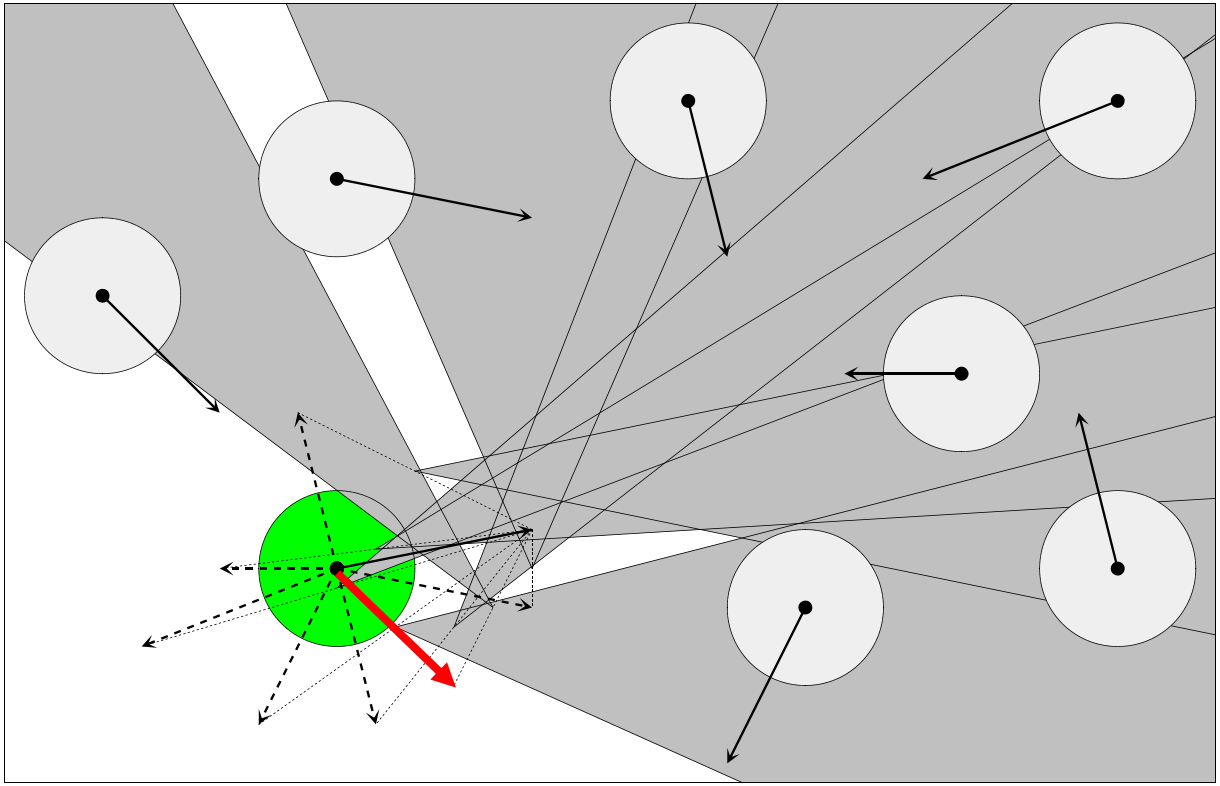
Not space efficient ☹

Robot alone; no human co-workers ☹

Optimal navigation plans ☺

Distributed solutions

- Many methods are based on reciprocal velocity obstacles (RVO) [Jur van den Berg, 2008]



Inputs:

1. An agent's velocity and preferred velocity (aiming at the goal)
2. All neighboring agents' velocities and positions

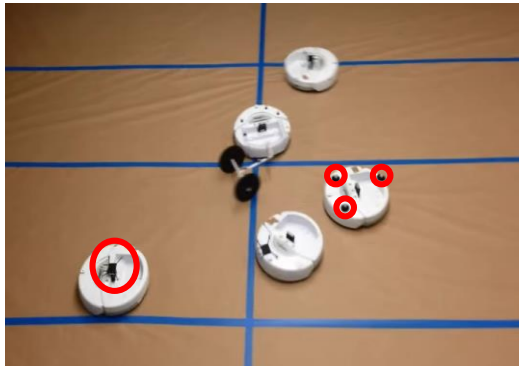
Output:

The agent's new velocity, outside the union of RVOs and closest to the preferred velocity

Distributed solutions X

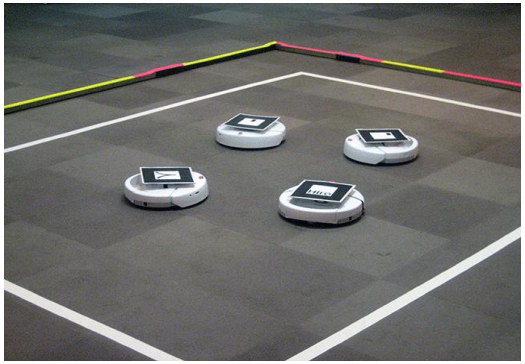
CALU: Collision Avoidance with Localization Uncertainty
Claes et al. 2012

Need inter-robot communication ☹



Generalized Reciprocal Collision Avoidance
Daman Bareiss and Jur van den Berg, 2015

Need overhead motion capture system ☹



The Hybrid Reciprocal Velocity Obstacle
Jamie Snape et al. 2011

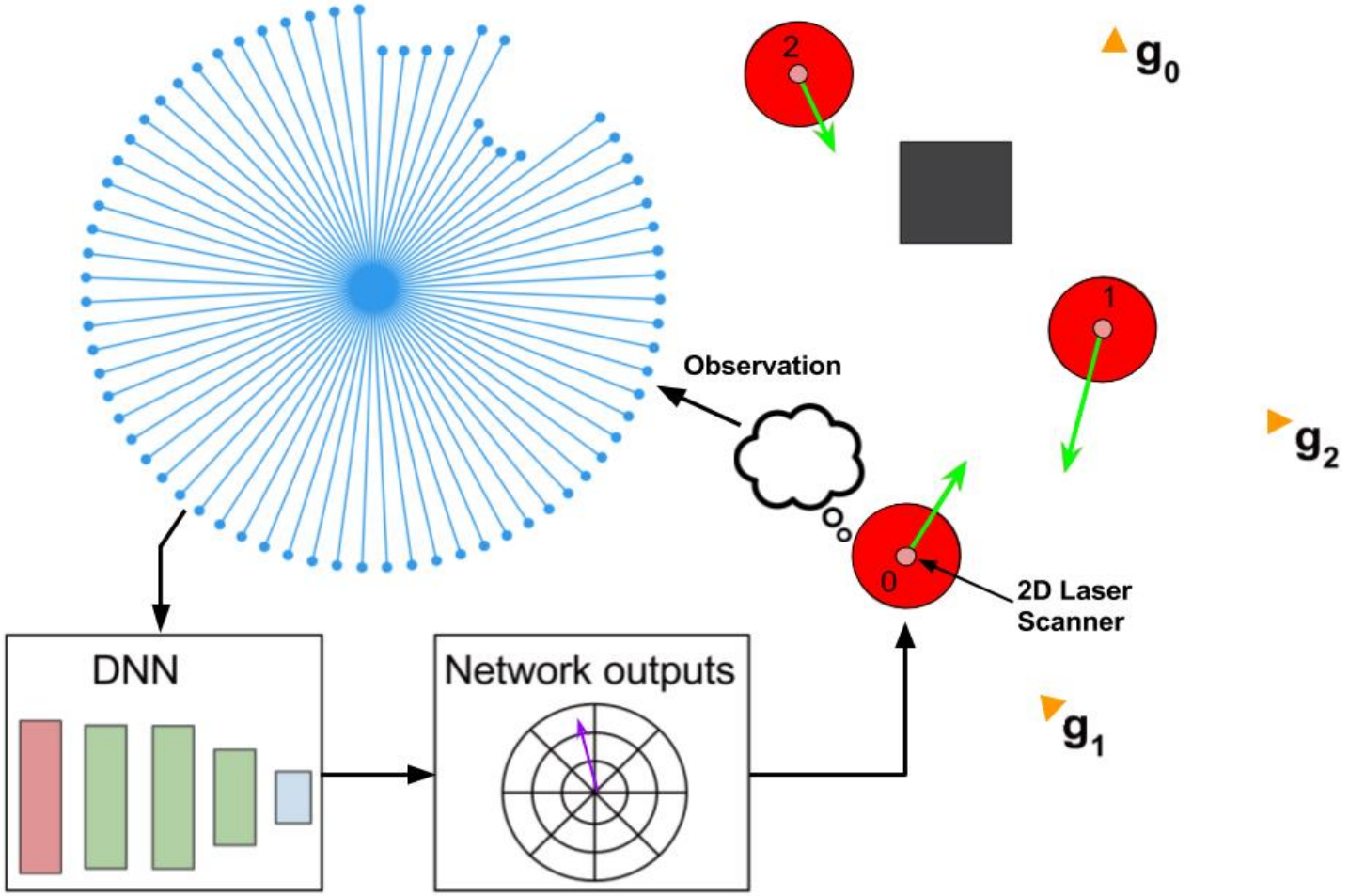
Need overhead motion capture system & AR markers ☹

Other problems of RVO

- Parameter tuning is rather difficult ☹☹☹
  - Different parameters → different behaviors
  - Many parameters; no clear guidance about tuning
- Agent-level planning/navigation algorithm ☹☹☹
  - First need the pipeline of segmentation, recognition and tracking to identify nearby agents or moving obstacles
  - Difficult to be robust
- Suboptimal policies due to partial information ☹
- No way to fine-tune the navigation policy for a specific scenario or application ☹☹☹

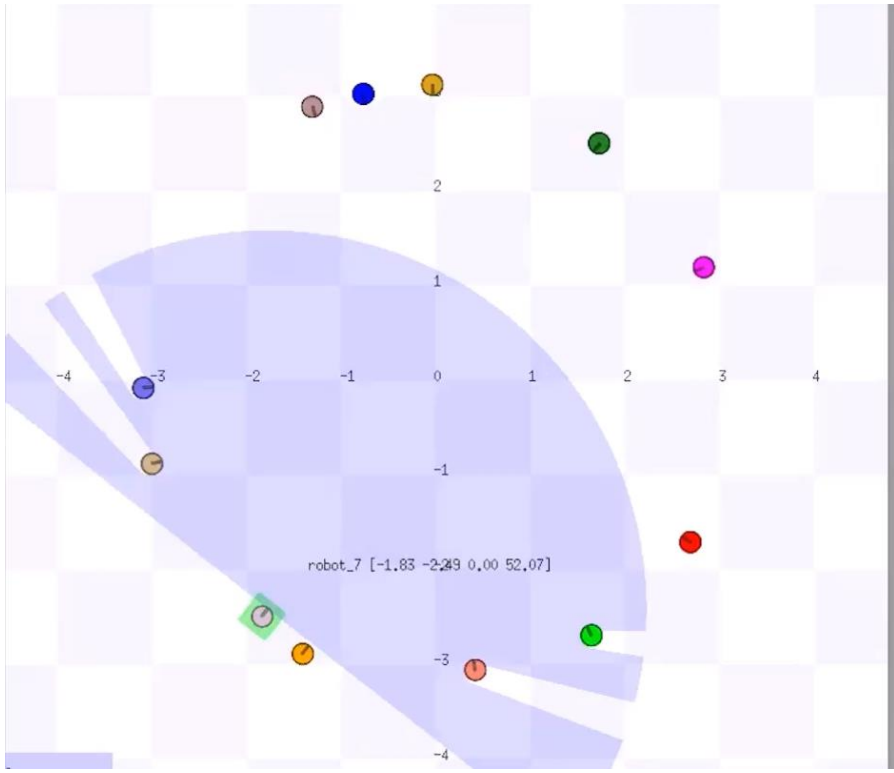
Completely distributed navigation system “seems” to be possible

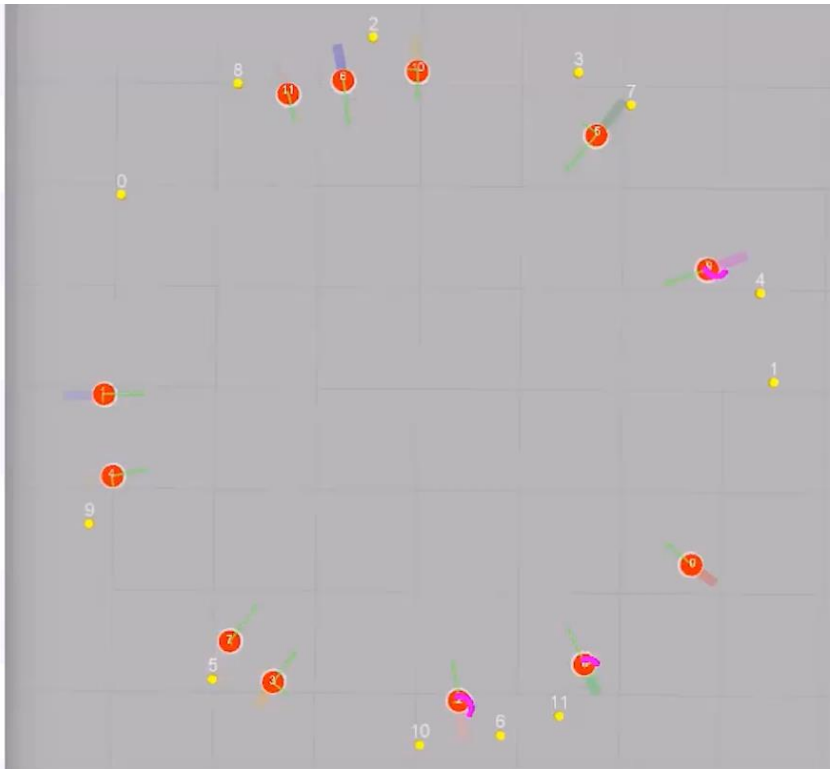
Robots navigate as humans



Decision making based on sensor measurements

- Using raw LIDAR/camera data; avoid troubles for segmentation, recognition, and tracking

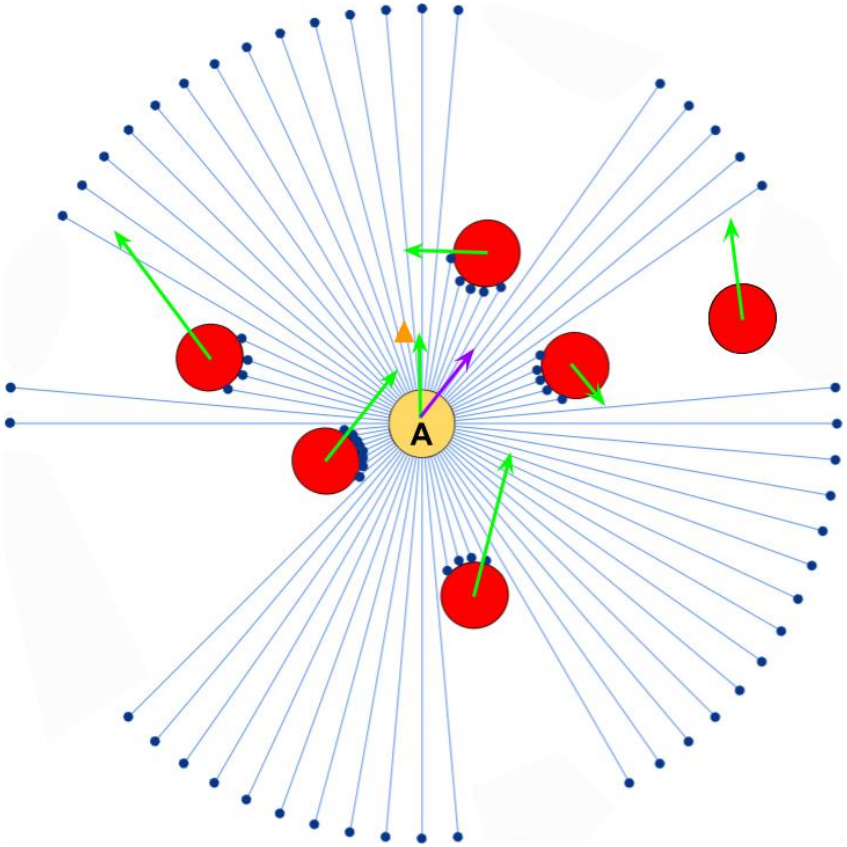




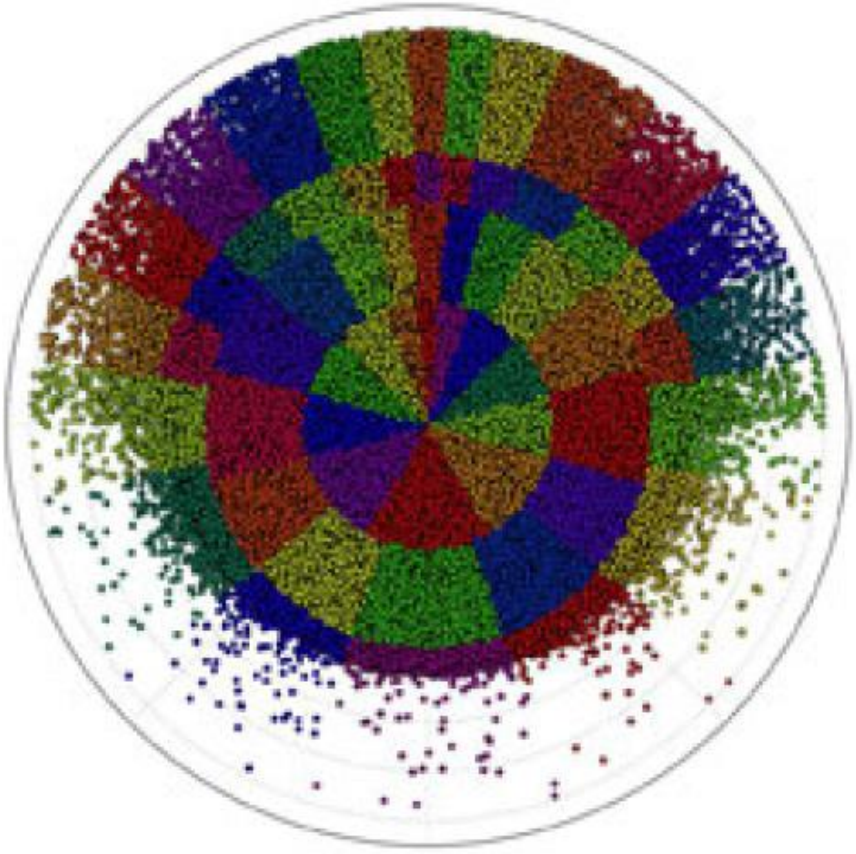
Training data

- Given an agent
  - Randomly sample its velocity and preferred velocity
  - Randomly sample a set of agents in its neighborhood
  - Velocities and goals of these agents are also randomly sampled
- Get the LIDAR data (with additional noise added)
- Select the agent's new velocity by running RVO simulator for one time step
  - Try a set of different RVO parameters
  - Remove RVO failure cases
  - Augment data using symmetries

Training data



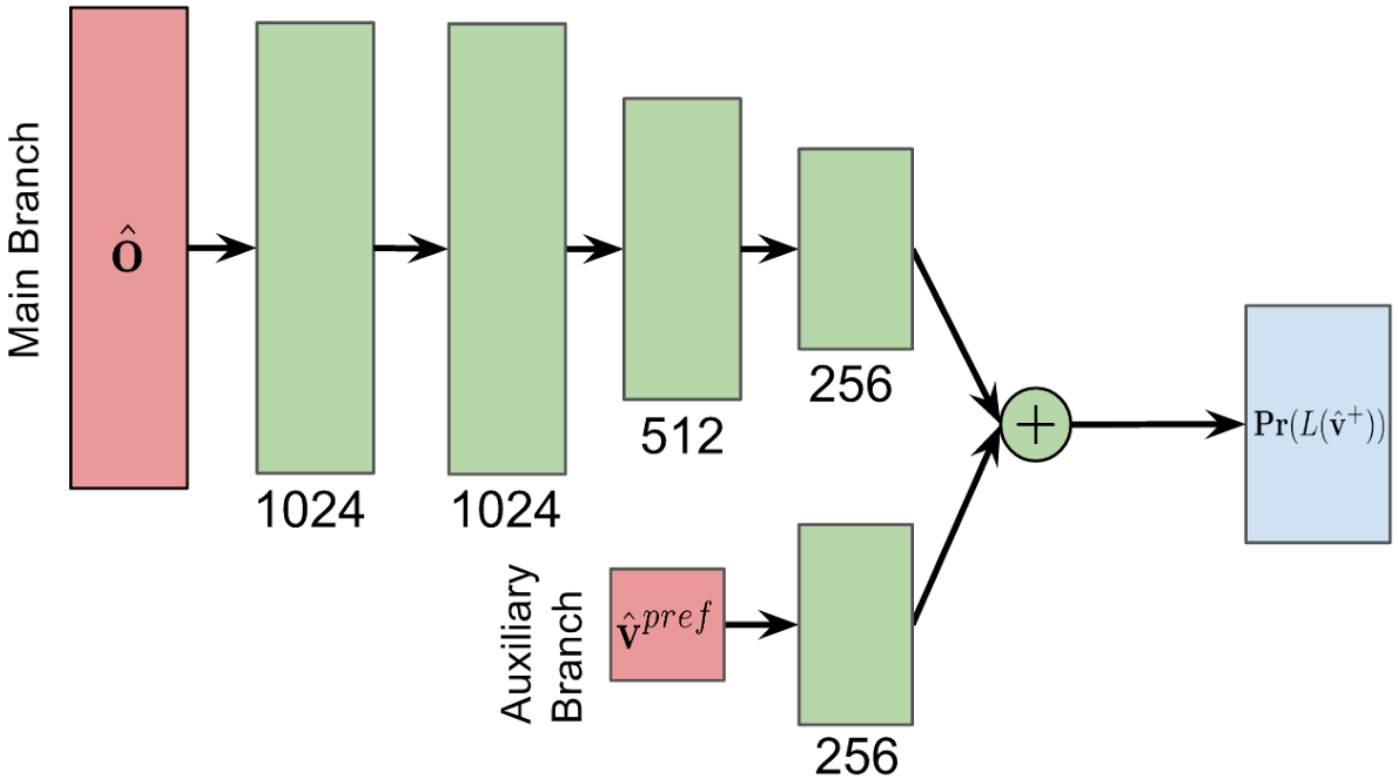
A sampled data and the RVO result



Quantization setting for velocities

Supervised learning

- Given sensor measurements & preferred velocities
- Output a reactive velocity for collision avoidance



Differences with RVO?

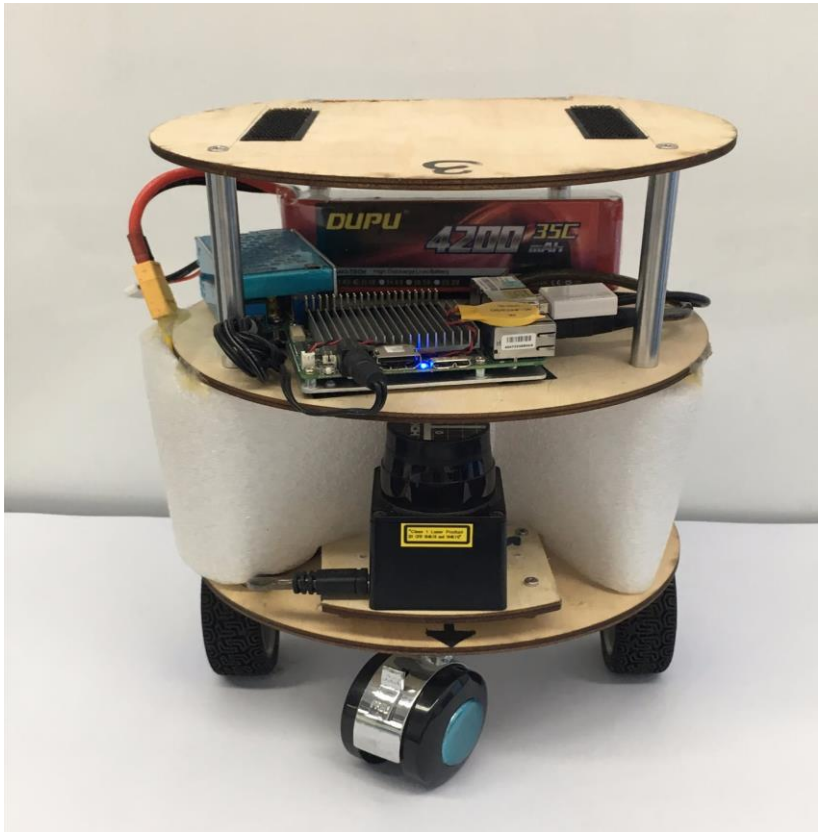
- Avoid the parameter tuning difficulty in RVO
  - Fuse RVO policies with different parameters
  - Data clearance removes bad behaviors of each policy
  - Combine different RVO policies' strengths
- From agent-level to sensor-level planning
  - The input to the controller is the sensor data
  - No need for complex vision pipelines
  - Easier to transfer to different robots and environments with obstacles
- Also benefit from the RVO's (correct) local collision avoidance heuristic

Experiments

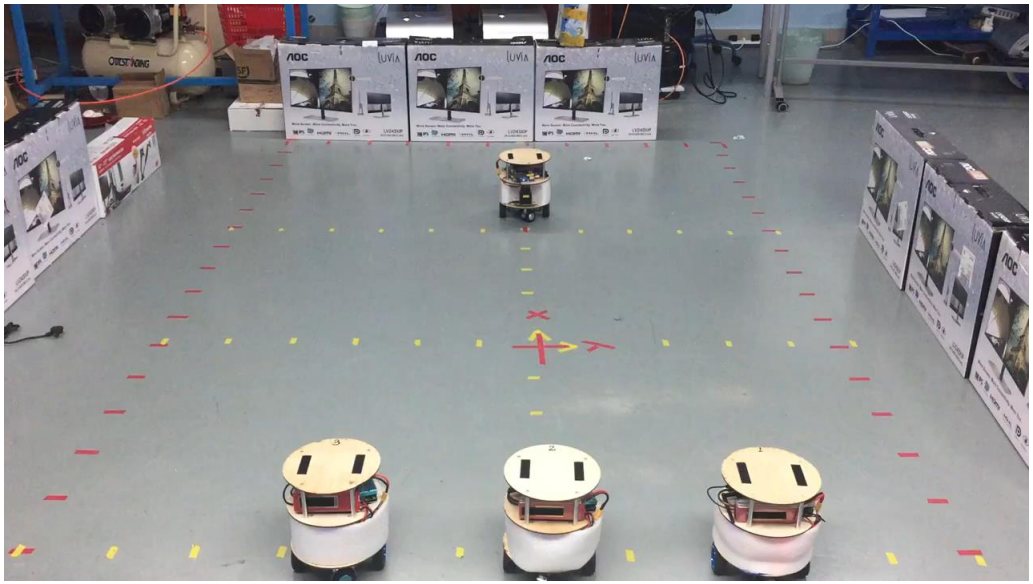
- Fully distributed and parallel system
- Non-holonomic robot dynamics
- Training data:
  - Scenarios without any static obstacles
  - Robots are of the same size
  - Only in simulation
- Test data
  - Scenarios with obstacles with different shapes
  - Robots with different size
  - Both simulation and real robots

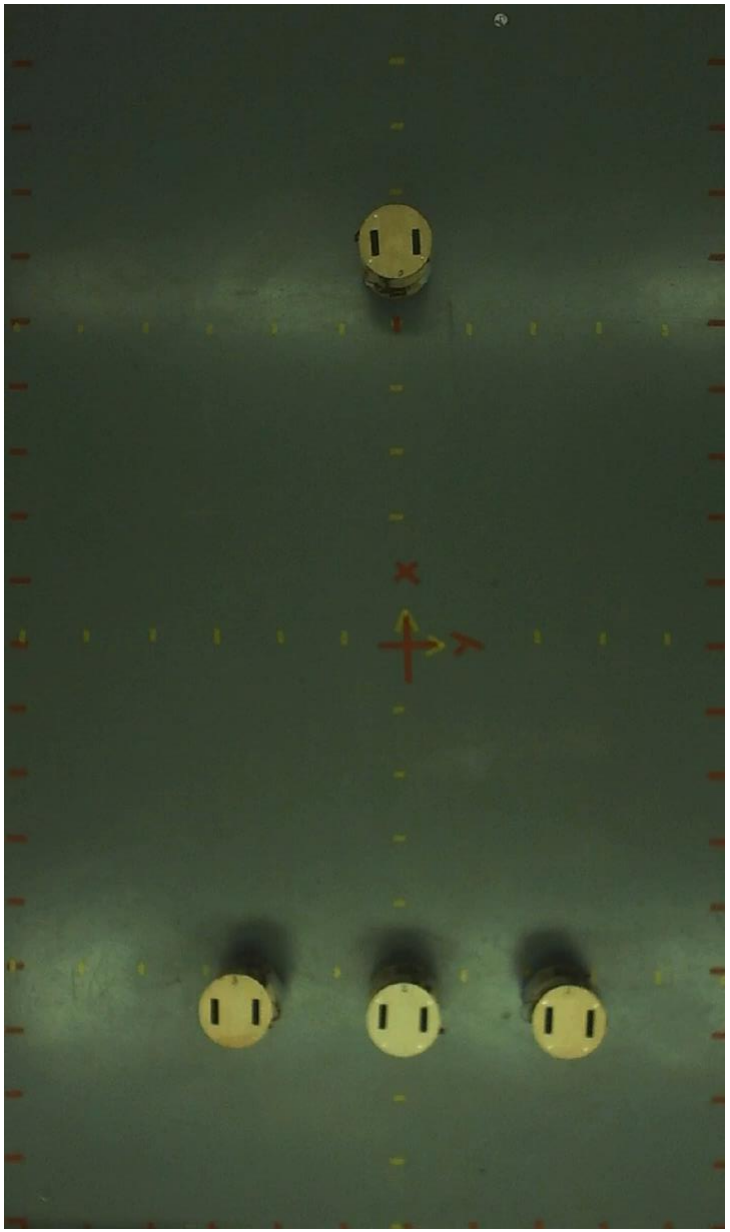
Experiment platform

- Differential driven robots
- 180-degree 2D laser scanner
- Each robot observes other robots via on-board 2D laser scanner.

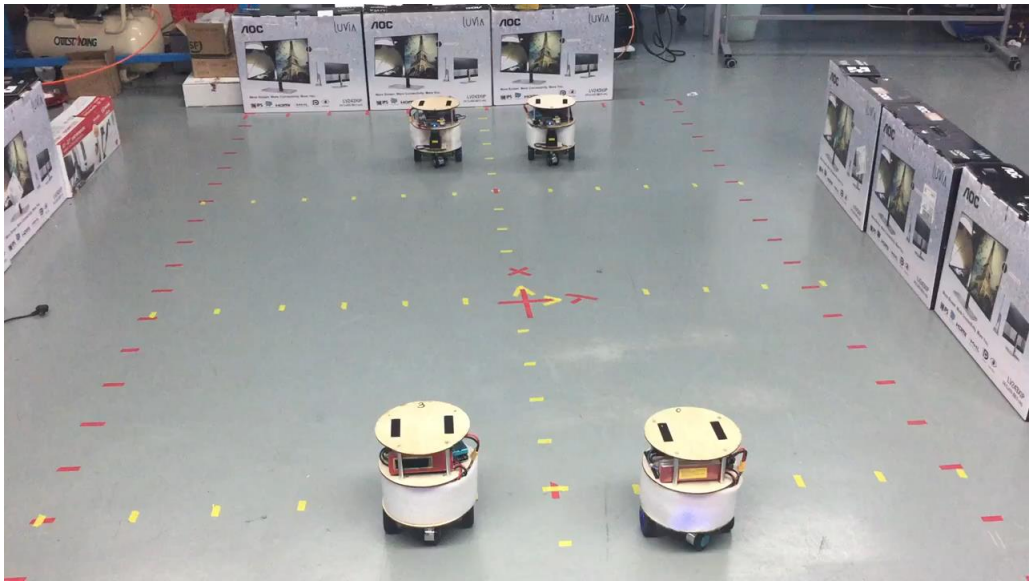


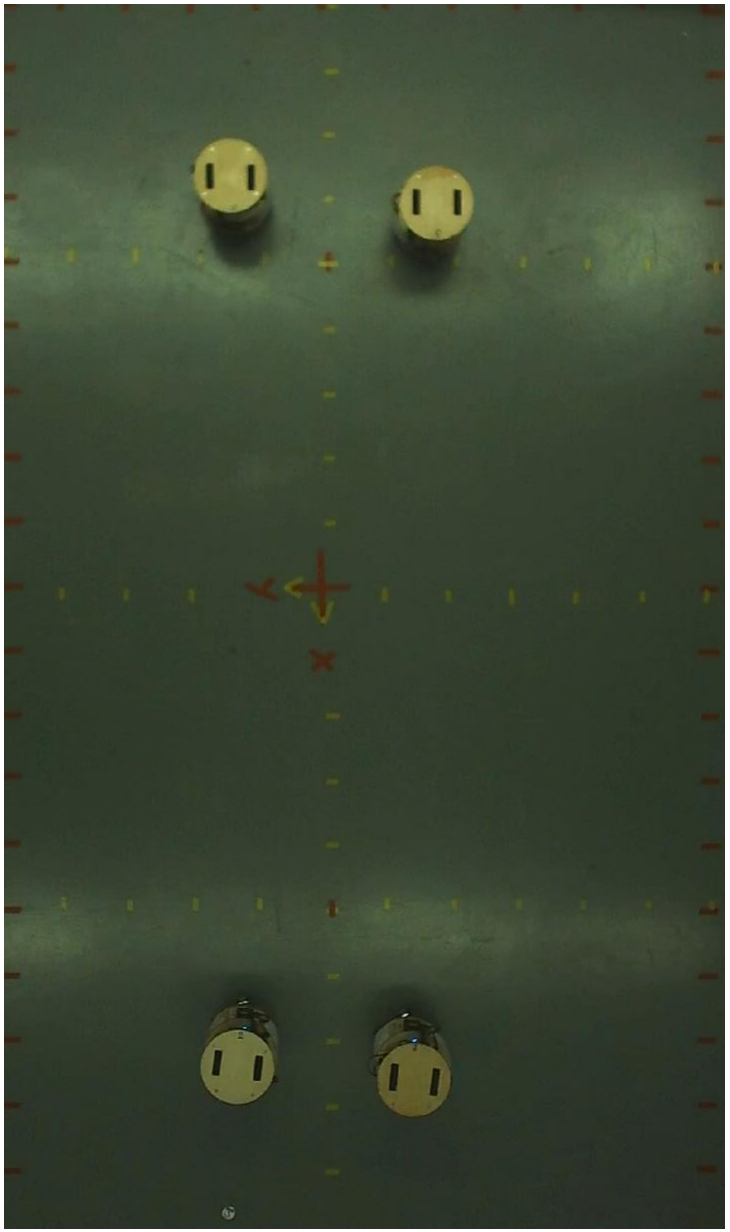
Real robots (1vs3)



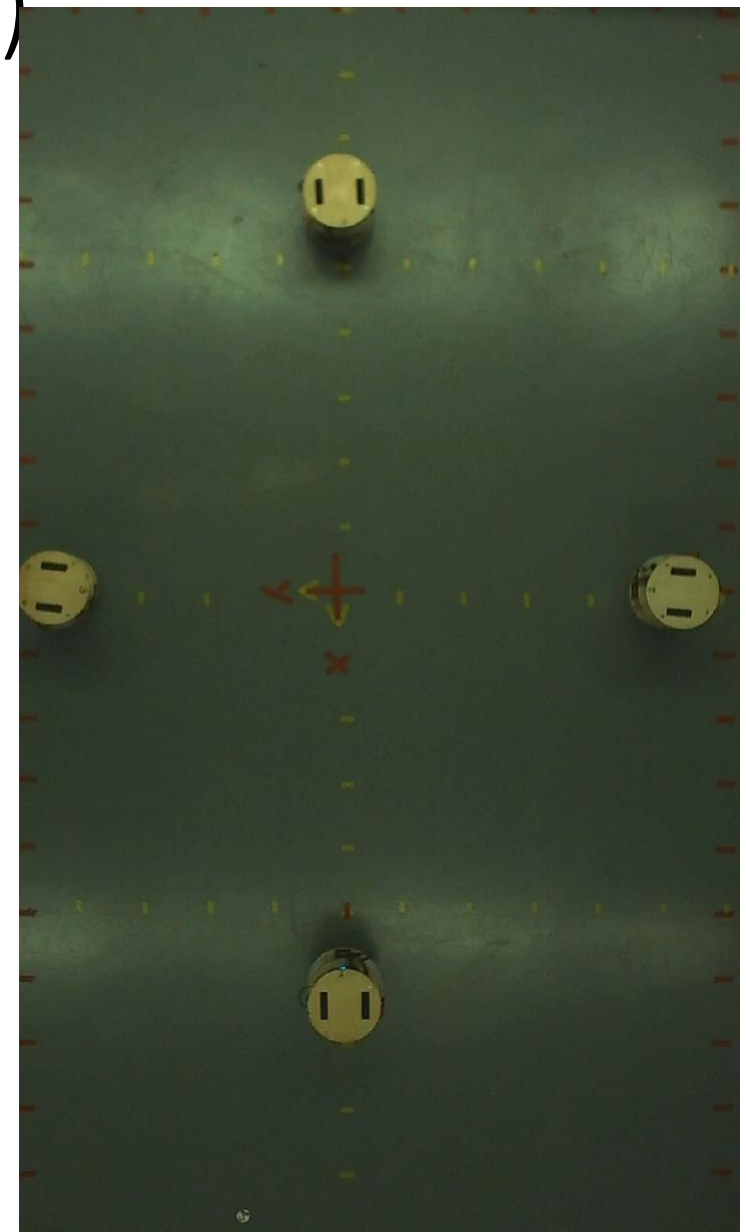


Real robots (2vs2)

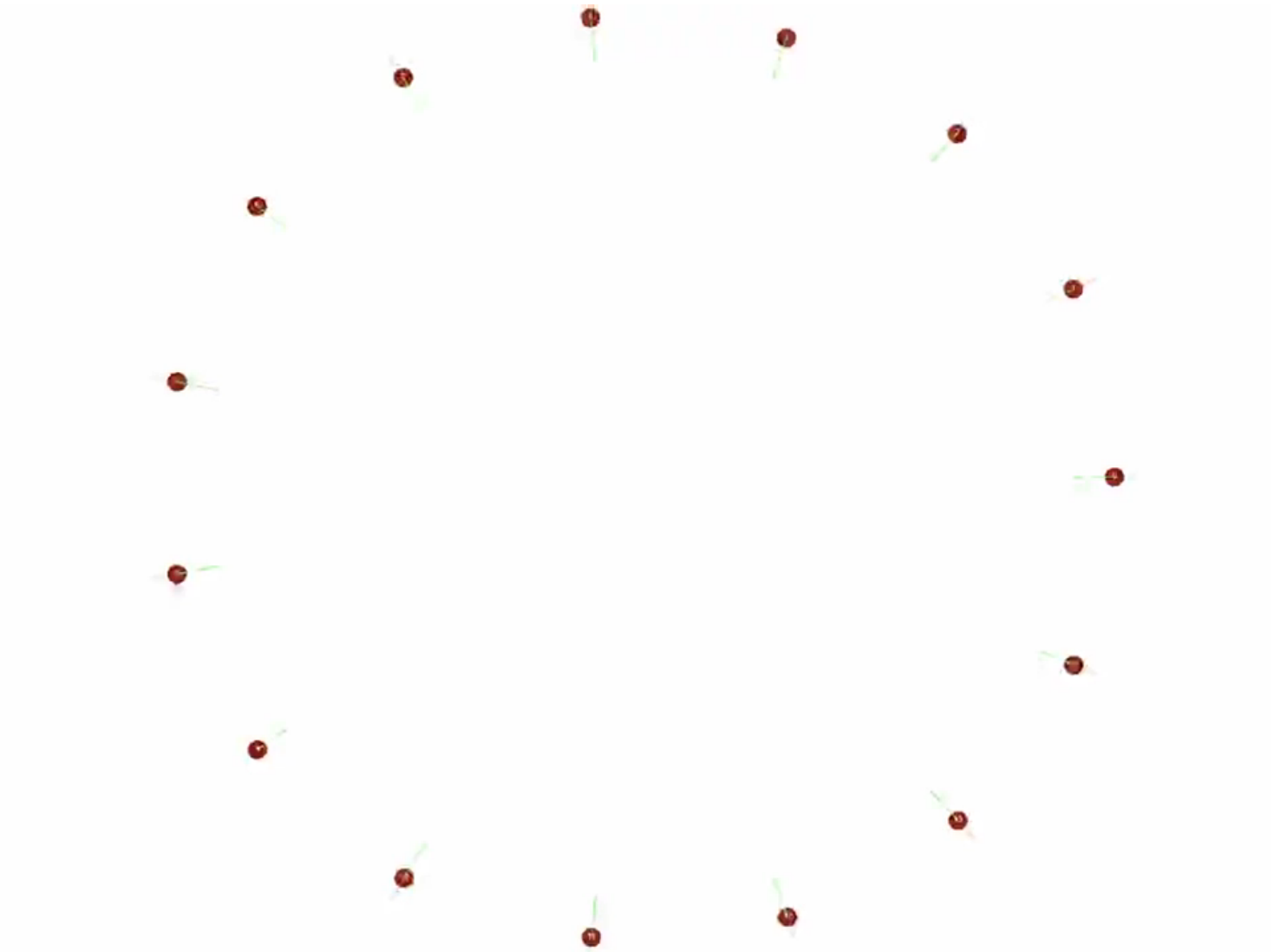




Real robots (4 circles)



Simulated robots

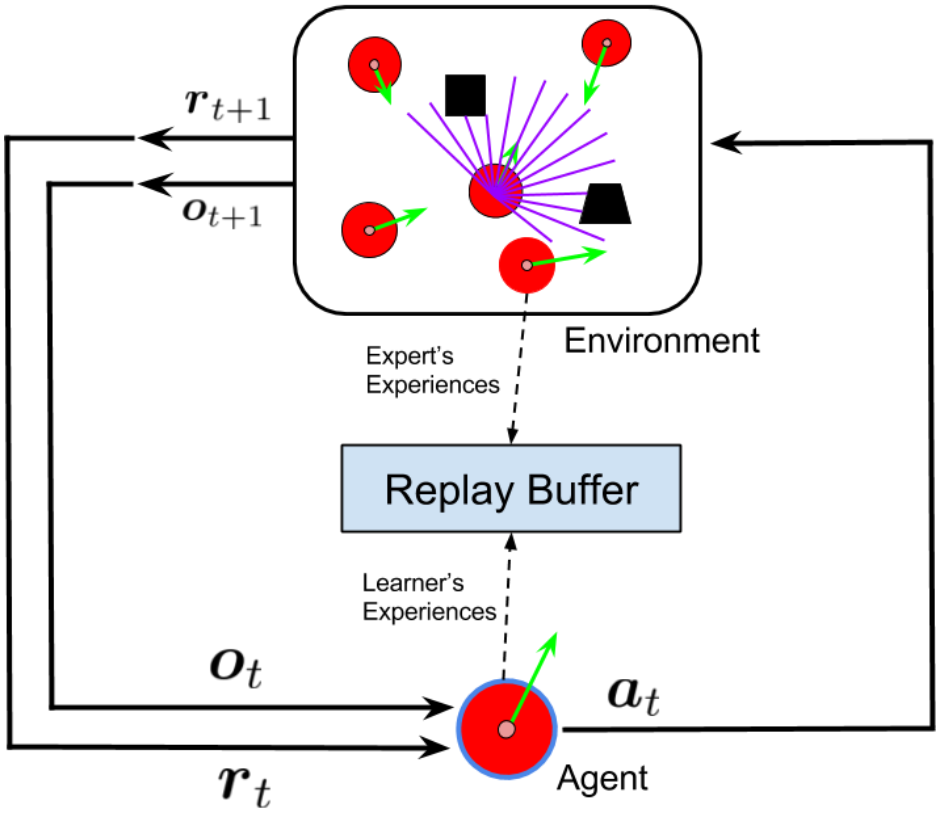


Pros and Cons

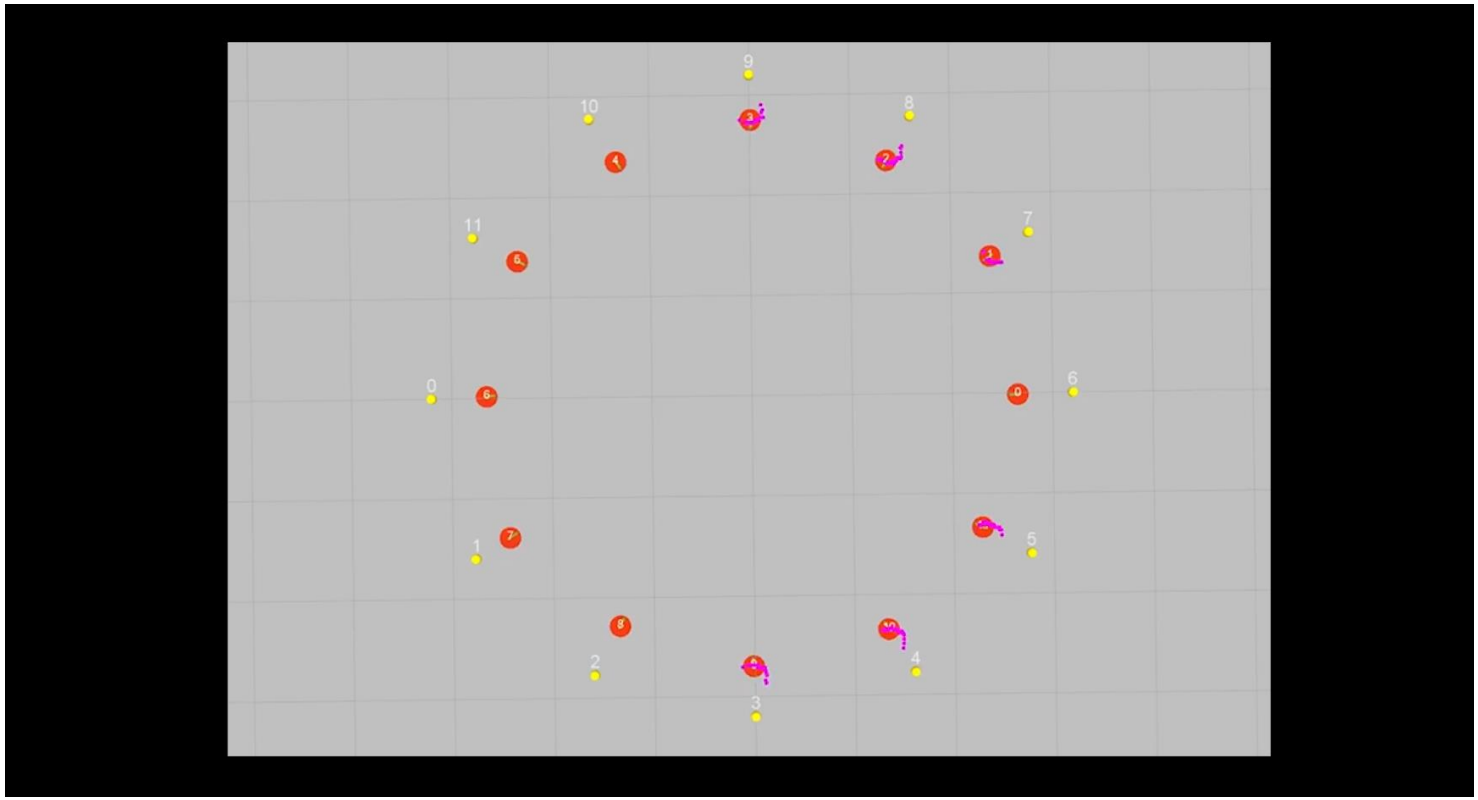
- Compared with RVO variants
  - No tedious parameter tuning ☺
  - Better generalization for static obstacles, robots with different shapes, and robots with complex dynamics ☺
- Compared with centralized methods
  - Better use of free spaces ☺
  - Better scalability ☺
  - But suboptimal ☹
- Solution: use reinforcement learning to improve trajectory quality for a given scenario/task

Optimize the navigation policy using reinforcement learning

- Given some initial policy π_0
- For k-th iteration
  - Select one agent s
  - All other agents use policy π_{k-1}
  - Agent s updates policy from π_{k-1} to π_k using DDPG [Lillicrap 2015]
  - Repeat until convergence

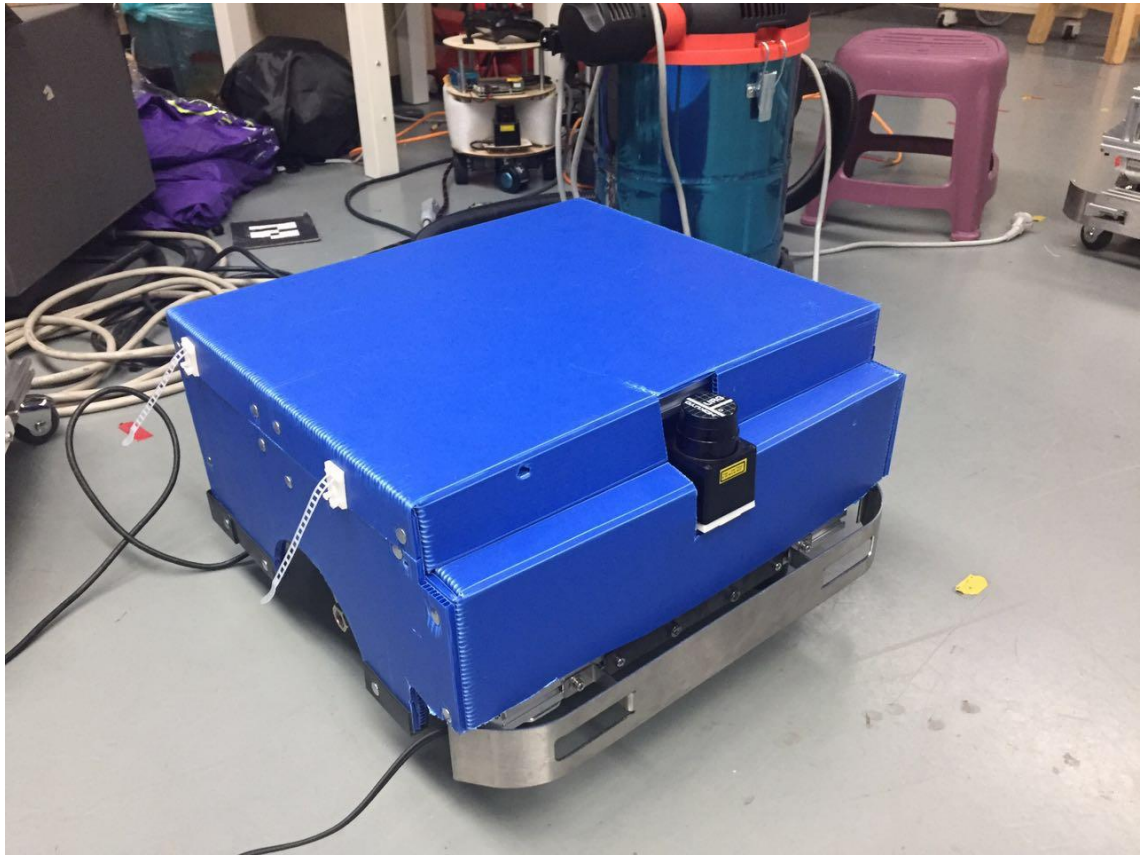


More natural and faster movements



Next step: robot platforms

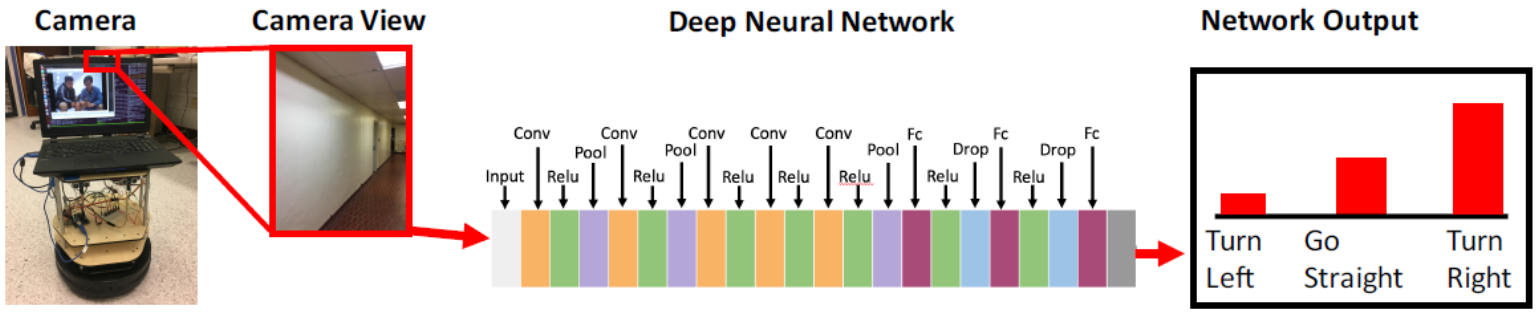
- 10 to 20



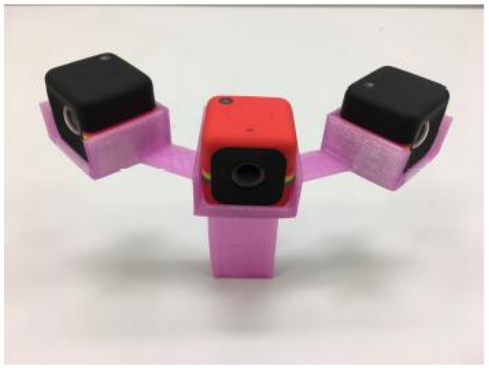
Next step: add human coworkers

- More difficult than multi-robot navigation or robot collision avoidance in static environments
- Human-human collision avoidance policy
- Human-robot & robot-human collision avoidance policy
- Human’s policies are different with robots and have higher variance

Navigation through human crowds



(a) Polaroid Cube+, motion cameras for video recording.



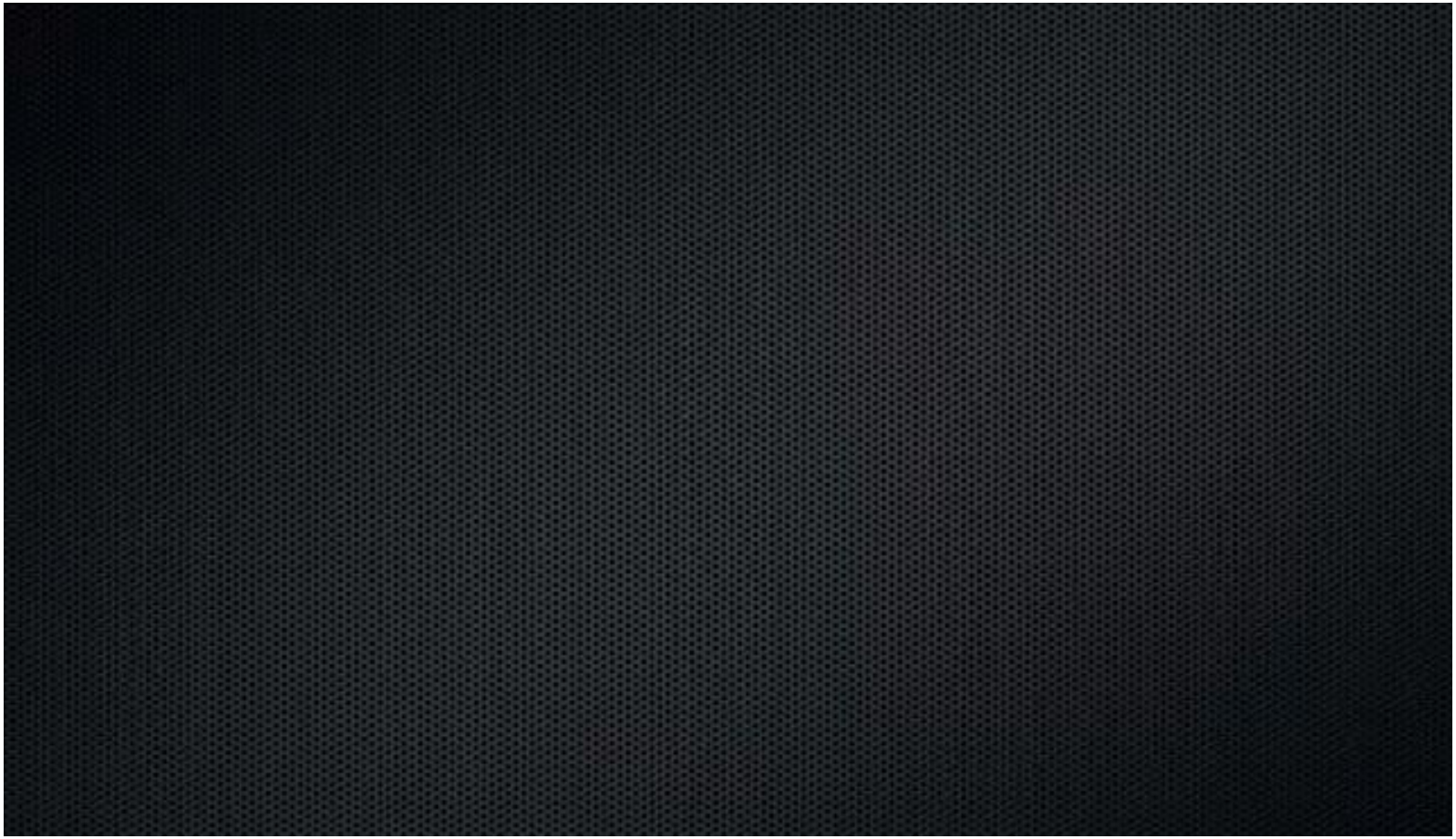
(b) Three cameras mounted on a 3D-printed holder

(c) DJI OSMO is used as a mechanical stabilizer

Data collection similar to [Giusti 2016]

Navigation through human crowds (preliminary result)

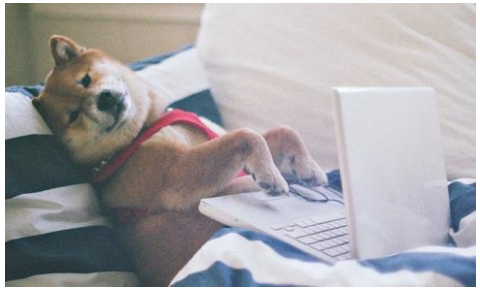
Fast Collision Avoidance



Conclusions

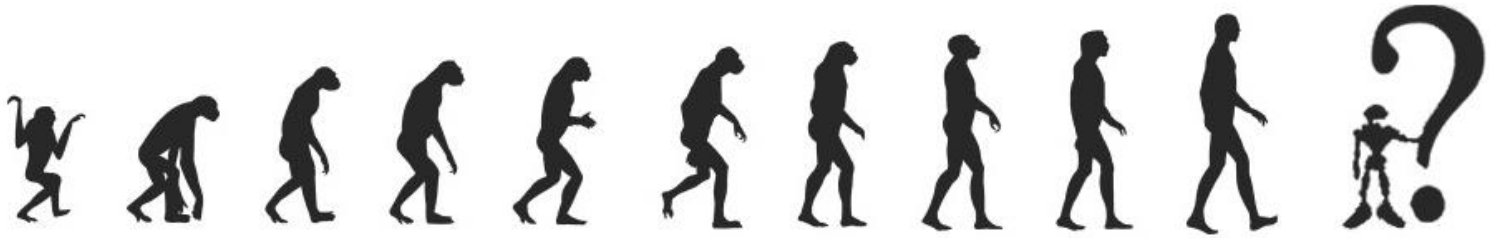
- Multi-robot navigation control for warehouses
- Distributed instead of centralized policy
- Sensor-level instead of agent-level planning
- Reinforcement learning to improve optimality
- Eventual goal: minimize the quality gap between distributed planning and centralized planning

Students wanted



- Perception for robotic manipulation
  - Tracking & reconstruction for manipulation
  - Bonus: background on slam / vision
- Advanced manipulation policies
  - Trajectory/task planning, deep reinforcement learning
  - Bonus: background on optimal control / deep learning
- Design of manipulation devices
  - Task-specific design &optimization of grippers & fixtures
  - Bonus: sensor fabrication / control / learning

Thanks



Recognition and tracking

- Would be useful for improving the distributed navigation policy for heterogonous robots or human-robot teaming
- Navigation policy can be used to help improve the vision task quality
- Vision task can be used to further improve navigation performance