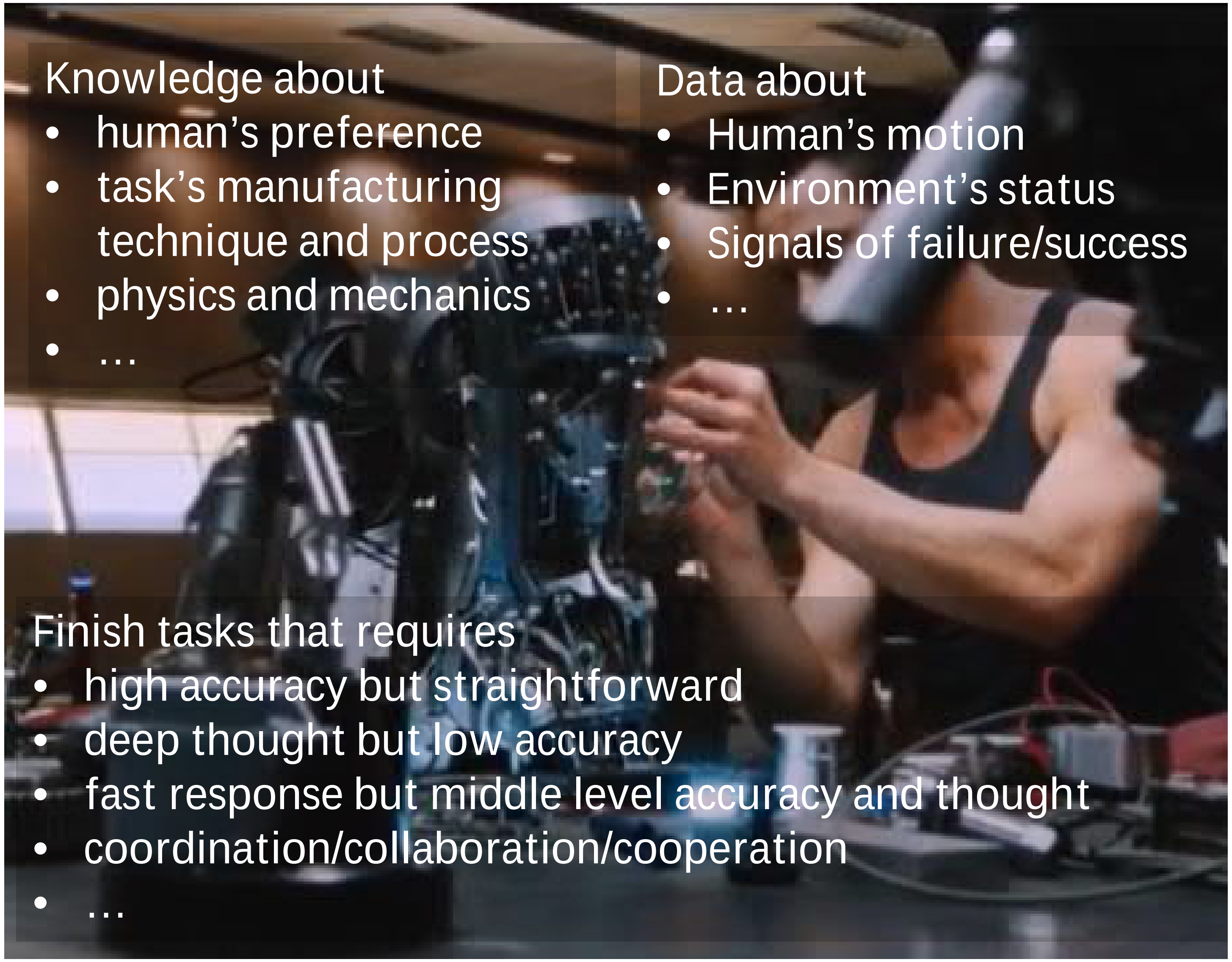
Autonomous robotic systems by combining control and learning

Jia Pan

Department of Computer Science

University of Hong Kong

What ideal robotic systems should look like



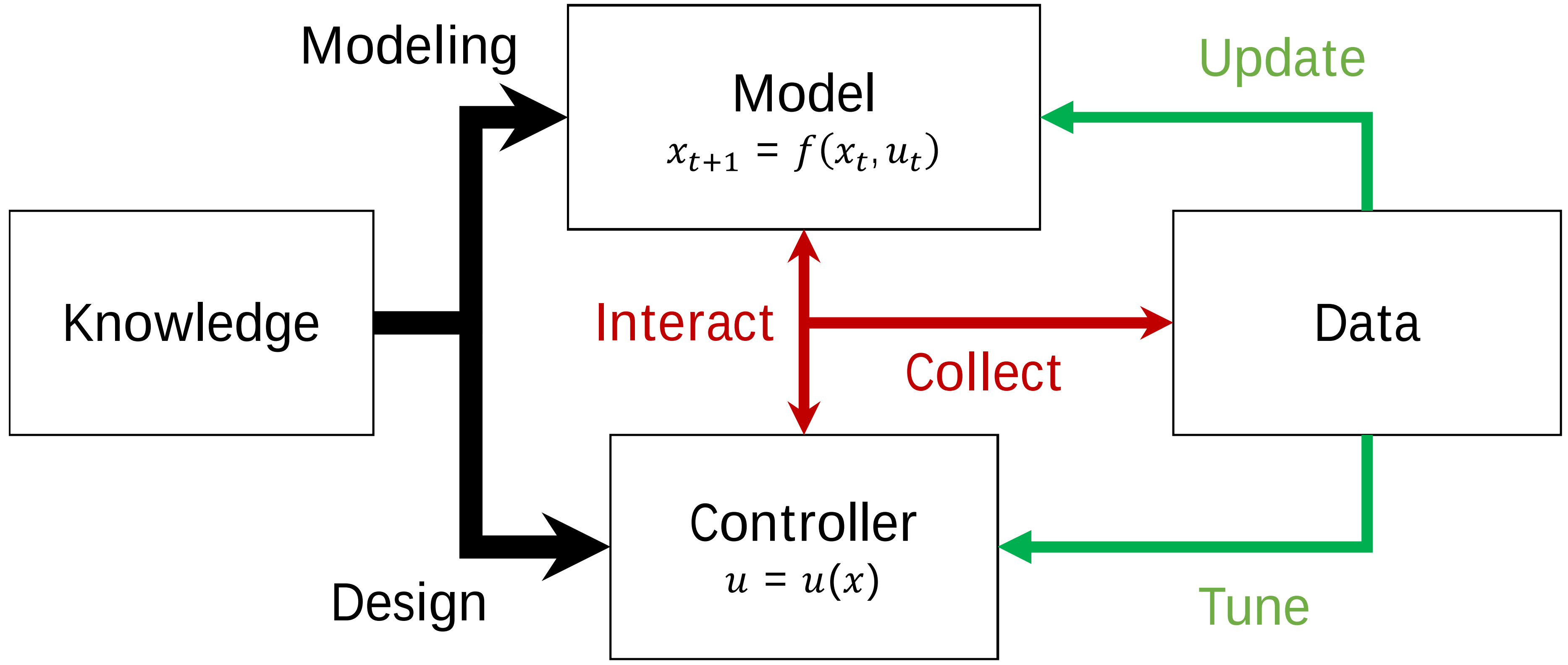
Knowledge about
- human's preference
- task's manufacturing technique and process
- physics and mechanics
- …

Data about
- Human's motion
- Environment's status
- Signals of failure/success
- …

Finish tasks that requires
- high accuracy but straightforward
- deep thought but low accuracy
- fast response but middle level accuracy and thought
- coordination/collaboration/cooperation
- …

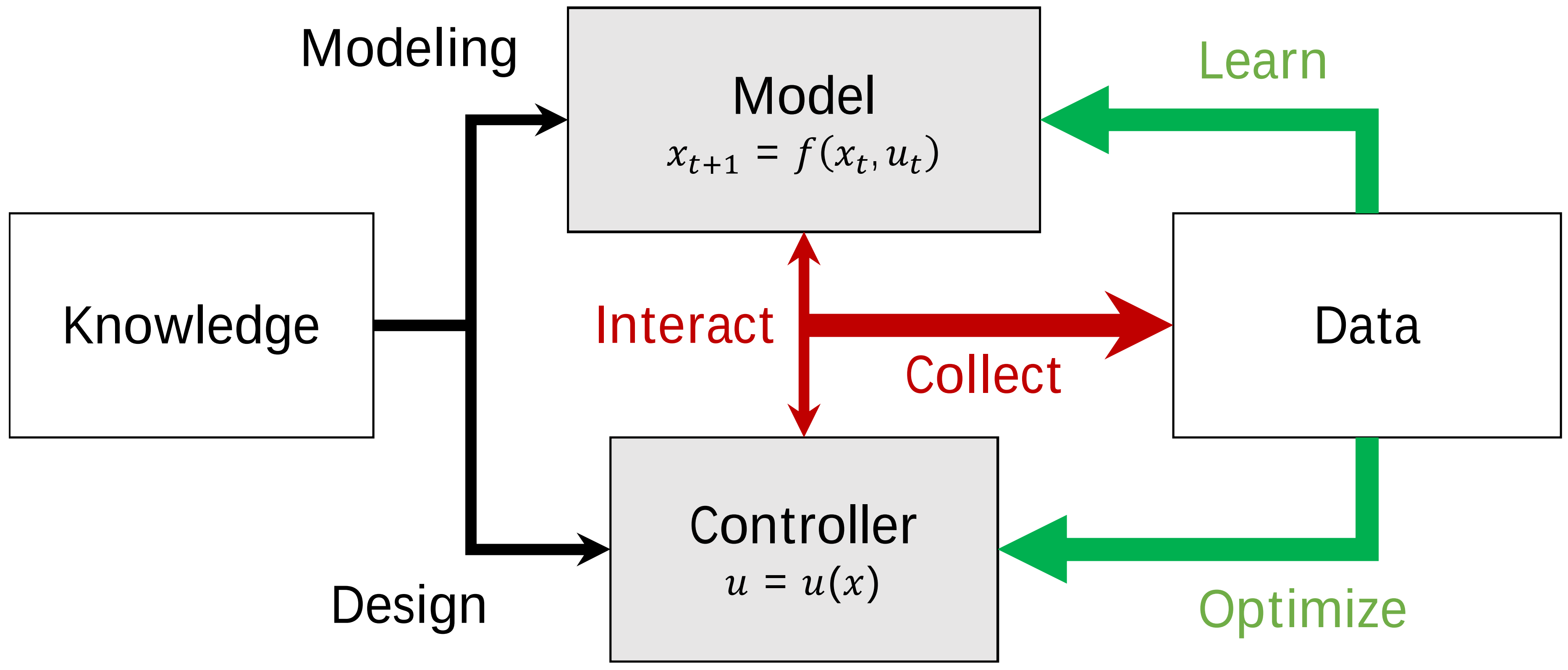
Control theory

- The study of how to use past knowledge and some data to enhance the future operation of a dynamical system



Reinforcement learning

- The study of how to use past data and some knowledge to enhance the future operation of a dynamical system



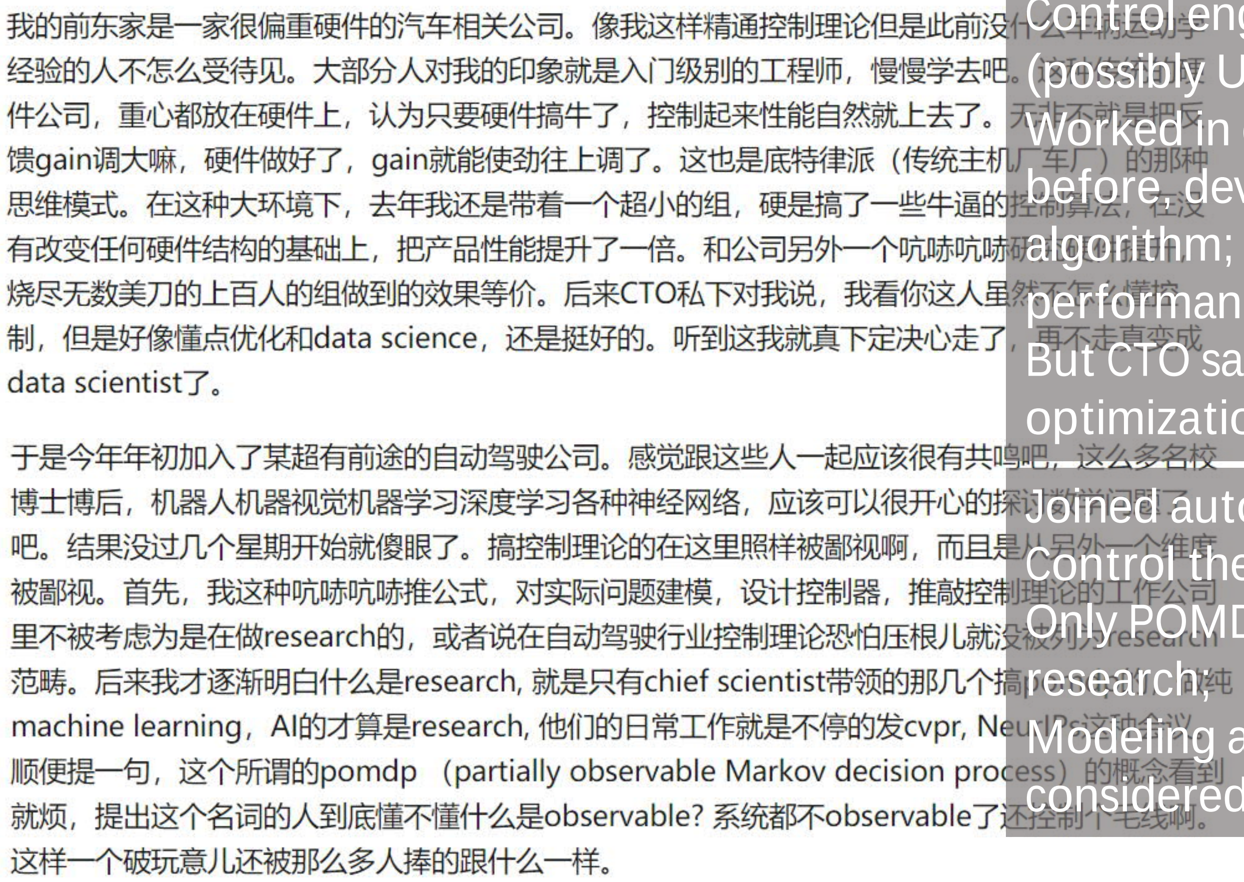
But almost the same thing

Are learning scientists and control scientists know each other well?

How are they thinking about each other?

Complains from control engineers

- 2o19-Dec-18 on Zhihu (Chinese version of quora): 1.8K likes and 294 comments

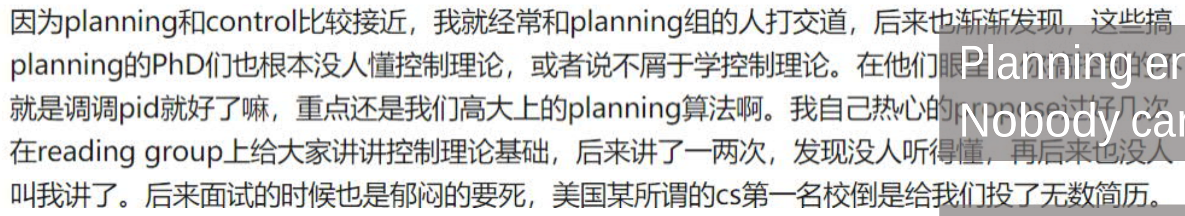


我的前东家是一家很偏重硬件的汽车相关公司。像我这样精通控制理论但是此前没什么车辆运动学经验的人不怎么受待见。大部分人对我的印象就是入门级别的工程师，慢慢学去吧。这种传统的硬件公司，重心都放在硬件上，认为只要硬件搞牛了，控制起来性能自然就上去了。无非不就是把反馈gain调大嘛，硬件做好了，gain就能使劲往上调了。这也是底特律派（传统主机厂车厂）的那种思维模式。在这种大环境下，去年我还是带着一个超小的组，硬是搞了一些牛逼的控制算法，在没有改变任何硬件结构的基础上，把产品性能提升了一倍。和公司另外一个吭哧吭哧研发硬件提升，烧尽无数美刀的上百人的组做到的效果等价。后来CTO私下对我说，我看你这人虽然不怎么懂控制，但是好像懂点优化和data science，还是挺好的。听到这我就真下定决心走了，再不走真变成data scientist了。

于是今年年初加入了某超有前途的自动驾驶公司。感觉跟这些人一起应该很有共鸣吧，这么多名校博士博后，机器人机器视觉机器学习深度学习各种神经网络，应该可以很开心的探讨数学问题了吧。结果没过几个星期开始就傻眼了。搞控制理论的在这里照样被鄙视啊，而且是从另外一个维度被鄙视。首先，我这种吭哧吭哧推公式，对实际问题建模，设计控制器，推敲控制理论的工作公司里不被考虑为是在做research的，或者说在自动驾驶行业控制理论恐怕压根儿就没被列为research范畴。后来我才逐渐明白什么是research, 就是只有chief scientist带领的那几个搞perception的纯machine learning，AI的才算是research, 他们的日常工作就是不停的发cvpr, NeurIPS这种会议。顺便提一句，这个所谓的pomdp （partially observable Markov decision process）的概念看到就烦，提出这个名词的人到底懂不懂什么是observable? 系统都不observable了还控制个毛线啊。这样一个破玩意儿还被那么多人捧的跟什么一样。

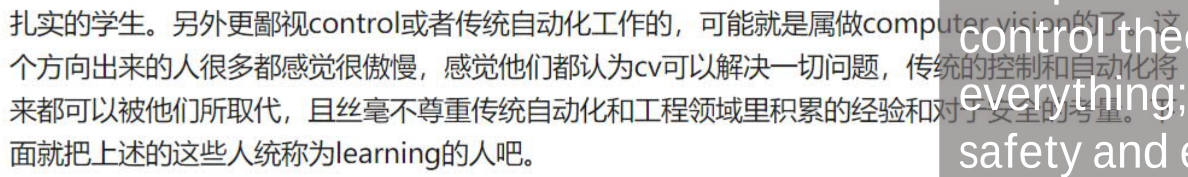
Control engineer, get control Ph.D. in US (possibly Umich);
Worked in car hardware company before, developed advanced control algorithm; greatly improved performance;
But CTO said: you know some optimization and data science ☹

Joined autonomous driving startup
Control theory is despised there
Only POMDP, vision, and learning are research;
Modeling and controller design are not considered as research ☹



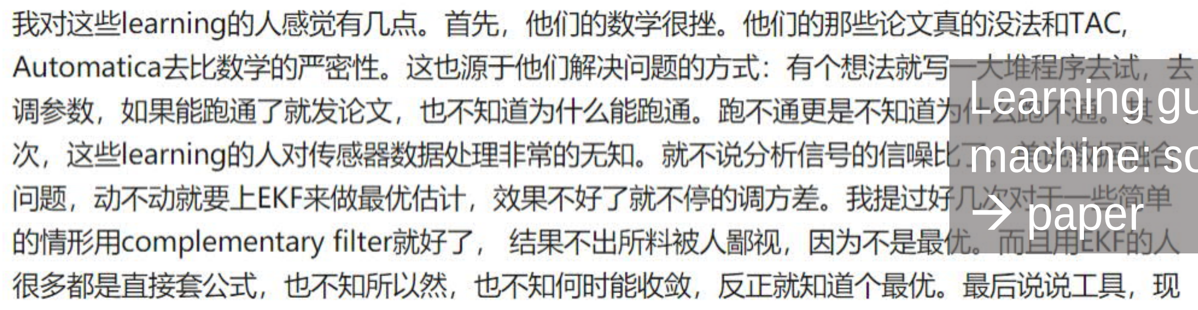
因为planning和control比较接近，我就经常和planning组的人打交道，后来也渐渐发现，这些搞planning的PhD们也根本没人懂控制理论，或者说不屑于学控制理论。在他们眼里，你搞控制的不就是调调pid就好了嘛，重点还是我们高大上的planning算法啊。我自己热心的propose过好几次在reading group上给大家讲讲控制理论基础，后来讲了一两次，发现没人听得懂，再后来也没人叫我讲了。后来面试的时候也是郁闷的要死，美国某所谓的cs第一名校倒是给我们投了无数简历。

Planning engineers also despise control
Nobody cares about control theory



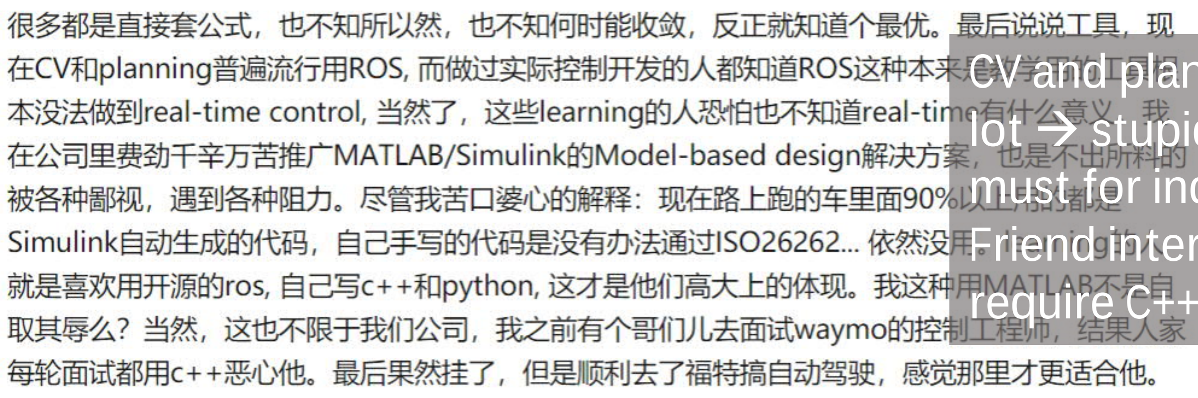
扎实的学生。另外更鄙视control或者传统自动化工作的，可能就是属做computer vision的了。这个方向出来的人很多都感觉很傲慢，感觉他们都认为cv可以解决一切问题，传统的控制和自动化将来都可以被他们所取代，且丝毫不尊重传统自动化和工程领域里积累的经验和对于安全的考量。下面就把上述的这些人统称为learning的人吧。

Computer vision community despises control theory even more: CV can solve everything; arrogant; no respect to safety and experience in control



我对这些learning的人感觉有几点。首先，他们的数学很挫。他们的那些论文真的没法和TAC, Automatica去比数学的严密性。这也源于他们解决问题的方式：有个想法就写一大堆程序去试，去调参数，如果能跑通了就发论文，也不知道为什么能跑通。跑不通更是不知道为什么跑不通。其次，这些learning的人对传感器数据处理非常的无知。就不说分析信号的信噪比了，单说数据融合问题，动不动就要上EKF来做最优估计，效果不好了就不停的调方差。我提过好几次对于一些简单的情形用complementary filter就好了， 结果不出所料被人鄙视，因为不是最优。而且用EKF的人很多都是直接套公式，也不知所以然，也不知何时能收敛，反正就知道个最优。最后说说工具，现

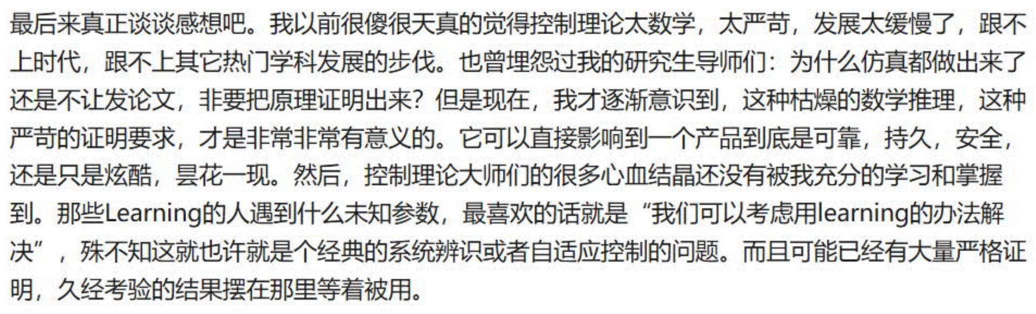
Learning guy are parameter tuning machine: some idea → tune parameter → paper



很多都是直接套公式，也不知所以然，也不知何时能收敛，反正就知道个最优。最后说说工具，现
在CV和planning普遍流行用ROS, 而做过实际控制开发的人都知道ROS这种本来是教学用的工具根本没法做到real-time control, 当然了，这些learning的人恐怕也不知道real-time有什么意义。我在公司里费劲千辛万苦推广MATLAB/Simulink的Model-based design解决方案，也是不出所料的被各种鄙视，遇到各种阻力。尽管我苦口婆心的解释：现在路上跑的车里面90%以上用的都是Simulink自动生成的代码，自己手写的代码是没有办法通过ISO26262... 依然没用。learning的人就是喜欢用开源的ros, 自己写c++和python, 这才是他们高大上的体现。我这种用MATLAB不是自取其辱么？当然，这也不限于我们公司，我之前有个哥们儿去面试waymo的控制工程师，结果人家每轮面试都用c++恶心他。最后果然挂了，但是顺利去了福特搞自动驾驶，感觉那里才更适合他。

CV and planning community use ROS a lot → stupid → MATLAB/Simulink is a must for industry quality
Friend interviewed in Waymo: also require C++ and Python, not Matlab

Complains from control engineers



最后来真正谈谈感想吧。我以前很傻很天真的觉得控制理论太数学，太严苛，发展太缓慢了，跟不上时代，跟不上其它热门学科发展的步伐。也曾埋怨过我的研究生导师们：为什么仿真都做出来了还是不让发论文，非要把原理证明出来？但是现在，我才逐渐意识到，这种枯燥的数学推理，这种严苛的证明要求，才是非常非常有意义的。它可以直接影响到一个产品到底是可靠，持久，安全，还是只是炫酷，昙花一现。然后，控制理论大师们的很多心血结晶还没有被我充分的学习和掌握到。那些Learning的人遇到什么未知参数，最喜欢的话就是“我们可以考虑用learning的办法解决”，殊不知这就也许就是个经典的系统辨识或者自适应控制的问题。而且可能已经有大量严格证明，久经考验的结果摆在那里等着被用。

Control is the king!
Control is rigorous, with mathematics guarantee.
Much better than learning

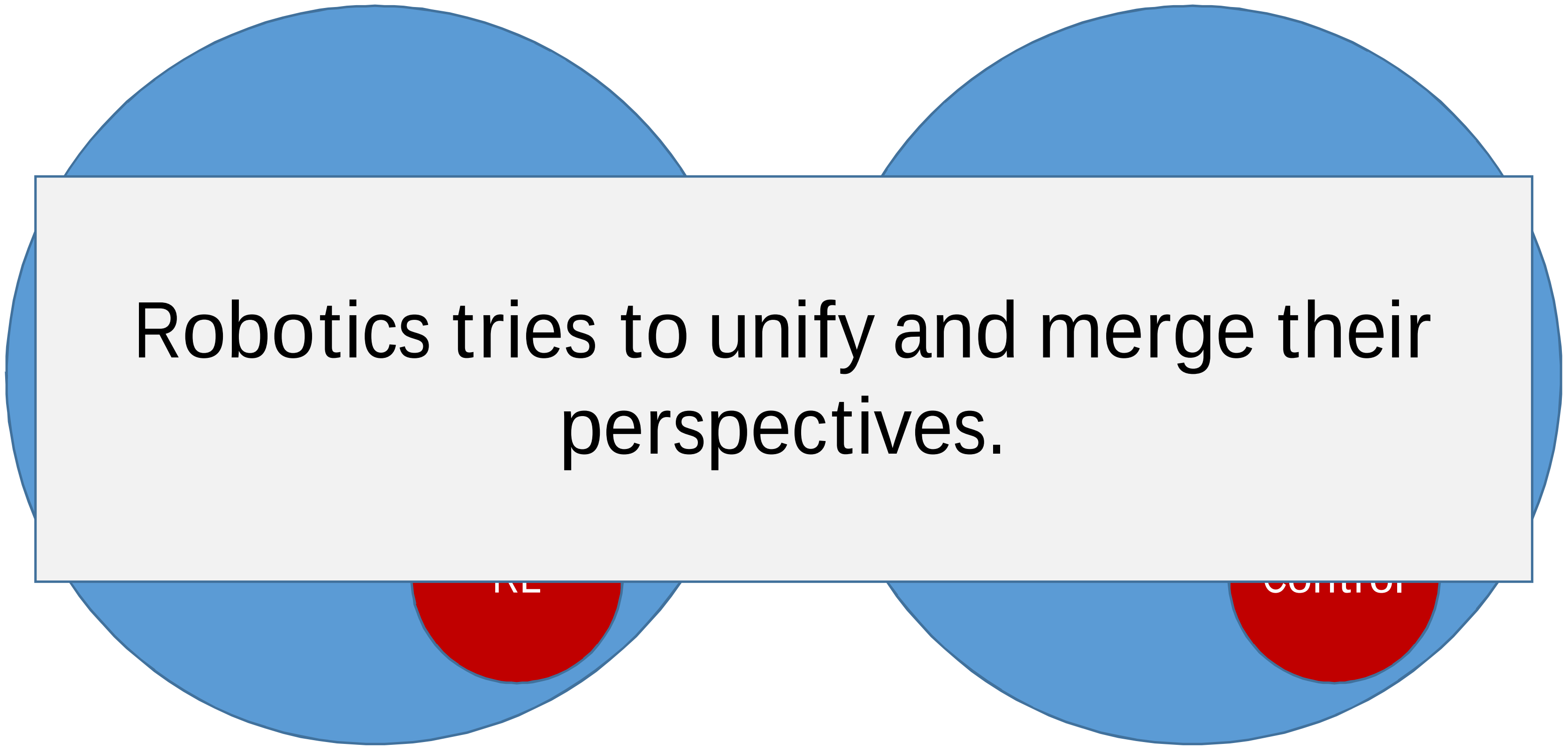
- https://www.zhihu.com/answer/939129746

Reality

Learning scientists and control scientists don’t know each other.

They look down upon each other

Disciplinary biases



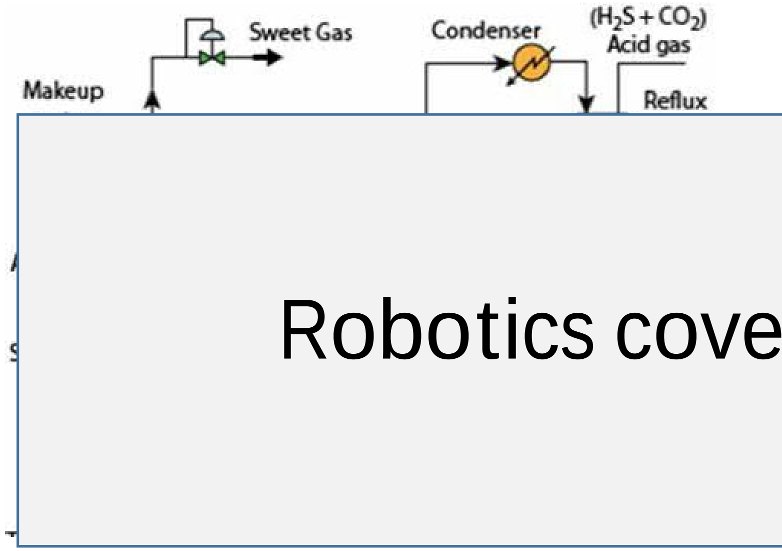
Robotics tries to unify and merge their perspectives.

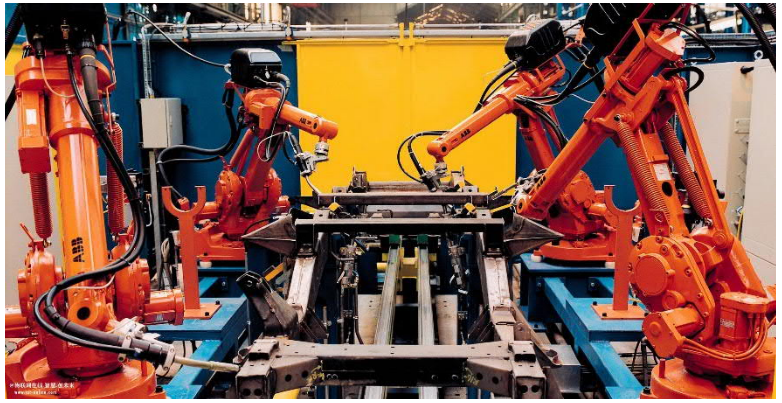
Continuous/discrete/hybrid
Model → action

Mostly discrete
Data → action

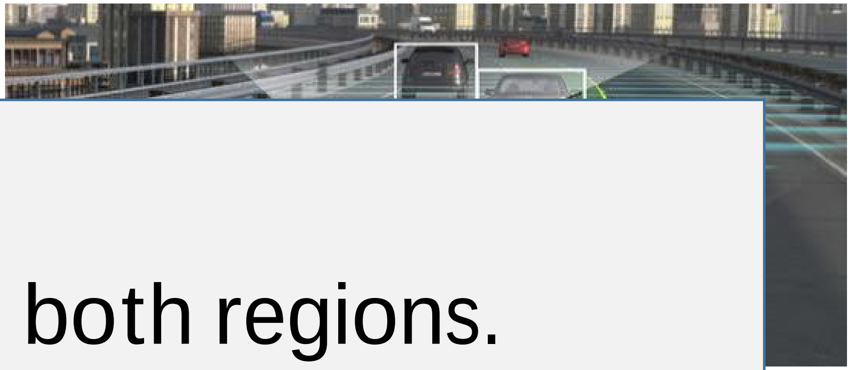
Task biases

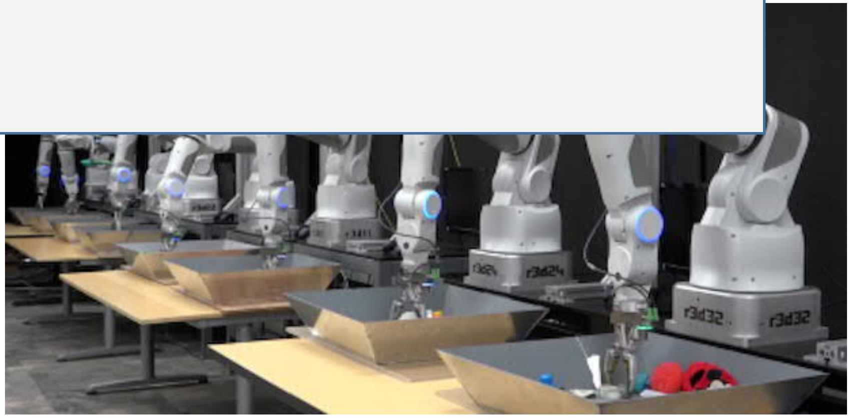
Control





Reinforcement learning





Robotics covers both regions.

Criterion biases

Control

- System guarantee

Modern robotics want both.

- Oscillation
- Convergence rate

- Low-level tasks

Reinforcement learning

- Intelligence

ng

py

ls

- High-level tasks

Why bother to combine?

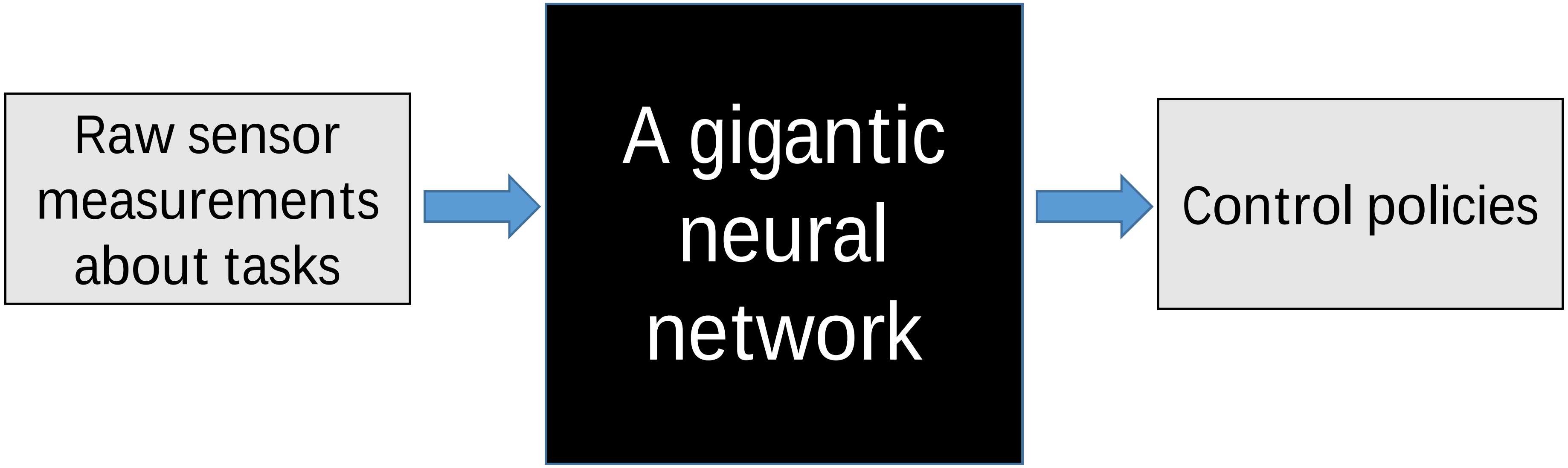
Pure control

- Modeling and design: time-consuming manual procedure; still an art
- Difficult to leverage rich sensory data (e.g. image or video)
- Huge gap between theory and applications, due to unrealistic assumptions necessary for math simplicity

Pure learning

- Low sample efficiency → large training set
- Convergence and hyper-parameter tuning: still an art
- Low dynamic performance
- Mostly limited in non-industrial applications

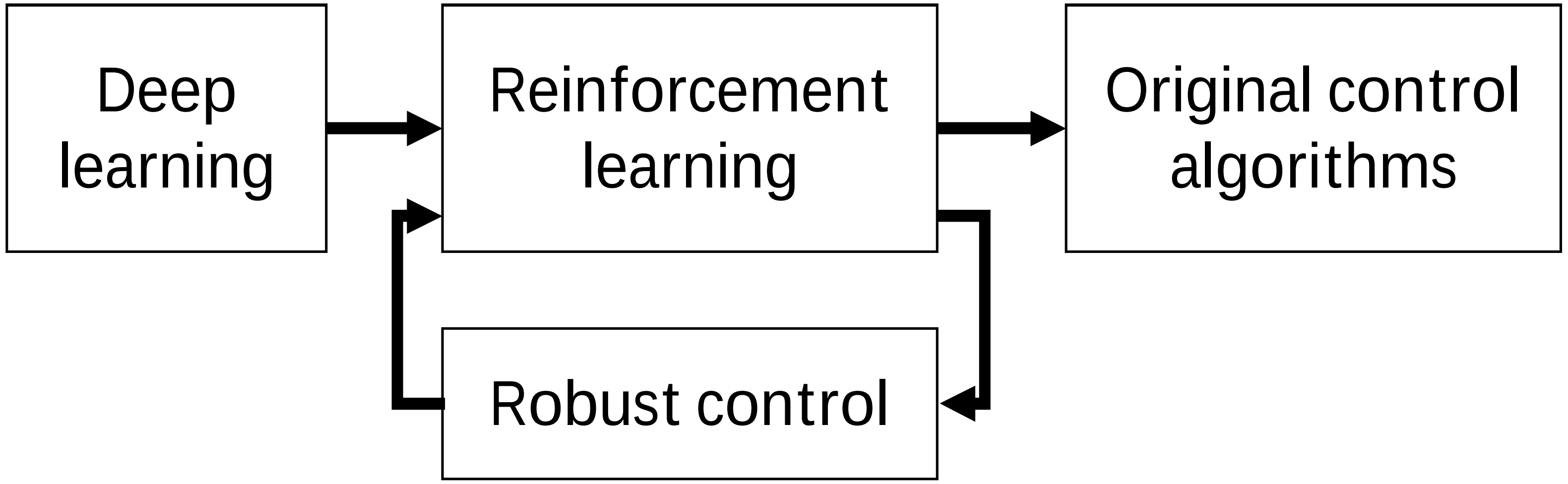
Reinforcement learning is limited


Raw sensor measurements about tasks
A gigantic neural network
Control policies

- Difficult to transfer from simulation to real world
- Requires huge number of training data / experience
- Discard most structural knowledge about tasks
- Limited to toy tasks:
  - e.g., flexible grasping is useless without the subsequent accurate and fast manipulation

How to combine?

- Common practice: add patches to fix problems



- Divide-and-conquer principle: control & learning as blackbox
- Simple combination; NOT always a good solution

How to combine – need deeper thinking

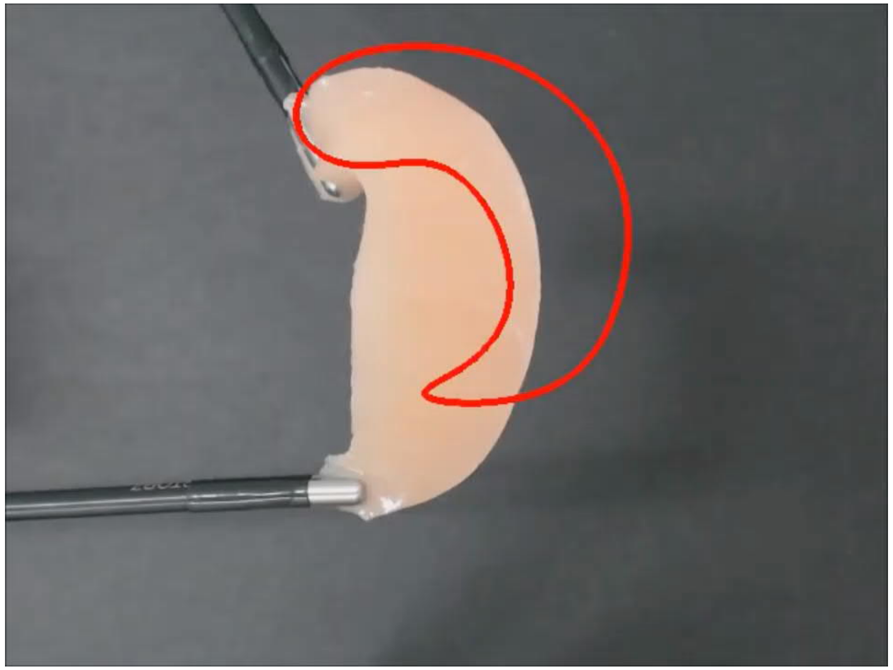
- Understand the core problem to be solved for your task
- Understand reinforcement learning and control's capability, e.g.,
  - RL: good at trade-off in complex situations
  - Control: good at fast response in simple situations
- Let reinforcement learning and control work on parts most suitable for them
  - Accomplish your task requirement; no more no loss

Let examples speak out

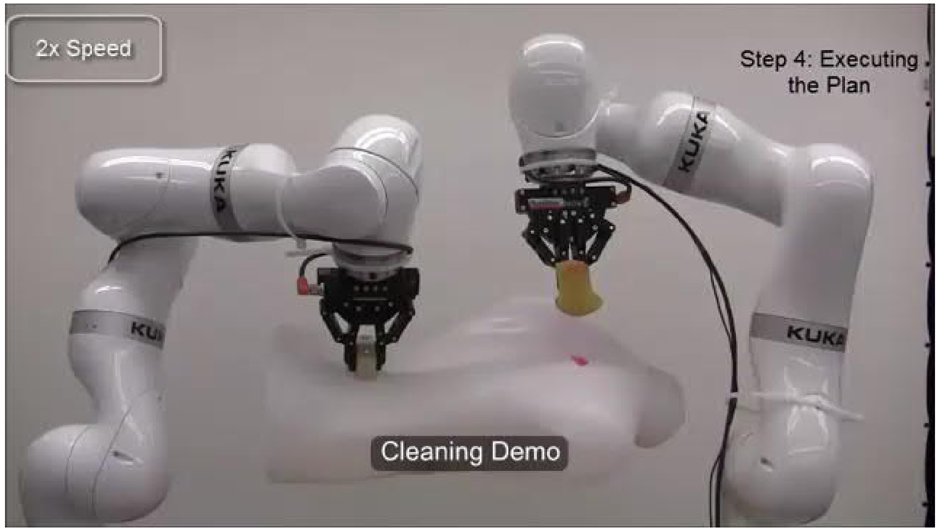
- Autonomous navigation in dense crowds

Let examples speak out

- Deformable object manipulation
  - Accurately change the shape or configuration of a deformable object



by Navarro-Alarcon and YH Liu,
Chinese University of Hong Kong



by . D. Langsfeld, A. M. Kabir, K. N. Kaipa, and S. K. Gupta,
Maryland Robotics Center

Autonomous navigation in dense crowds

Many important applications

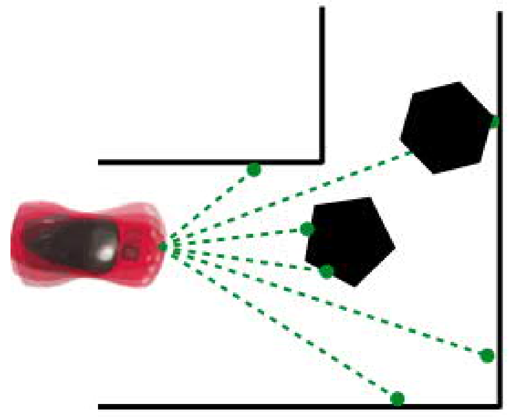
- Low-speed autonomous driving
  - Valent parking
  - Package delivery system
- Service robot
  - Hotel service
  - Elderly assistance
  - Crowd surveillance

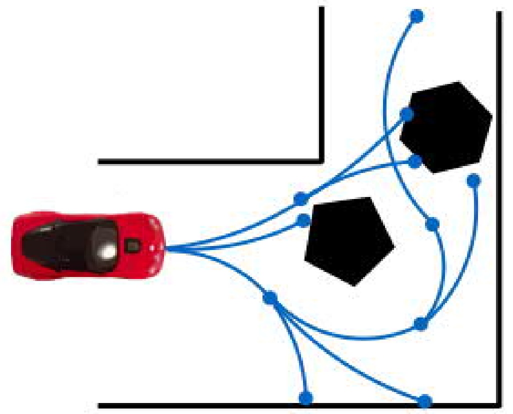
State of the arts

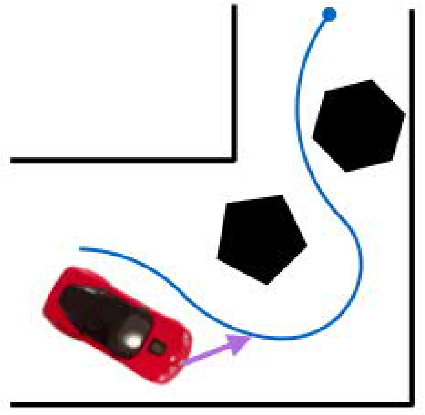
Common solutions: divide-and-conquer

- Based on the well-known solution to autonomous deriving
- Simultaneous Localization and Mapping (SLAM)

Build a map and localize the robot using SLAM → Motion planning: avoid collisions with static obstacles → Feedback control follows the collision-free trajectory

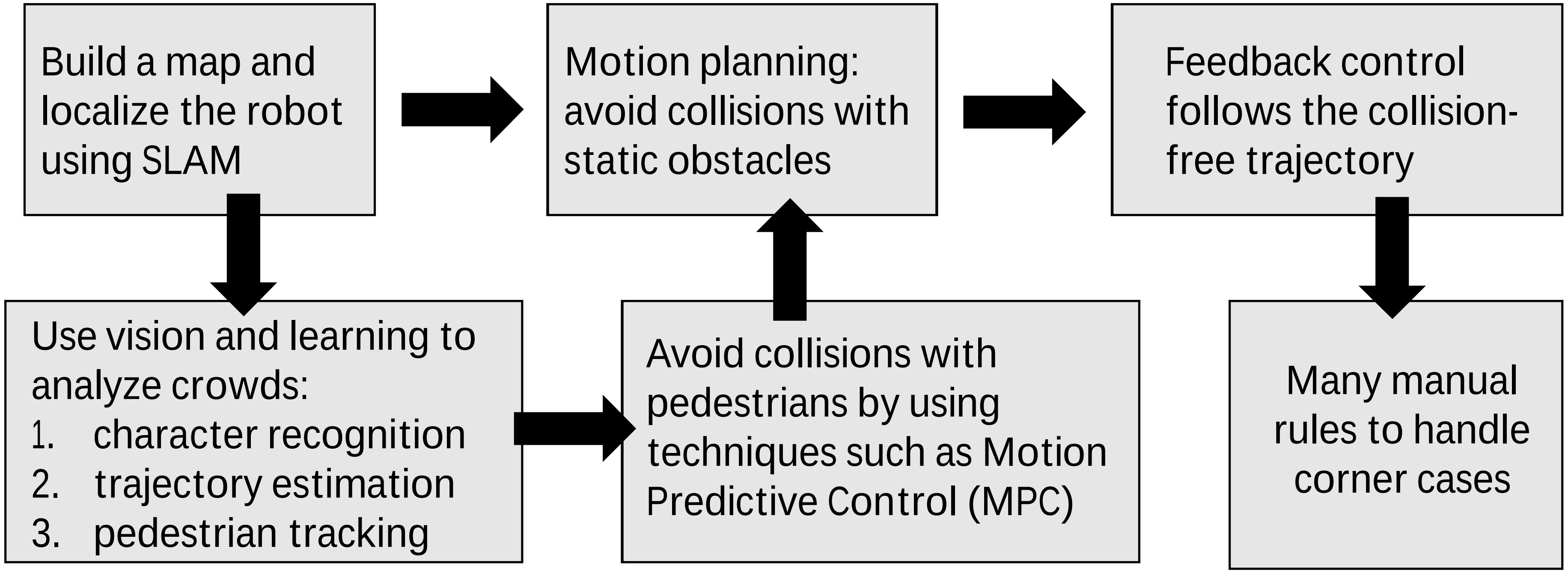




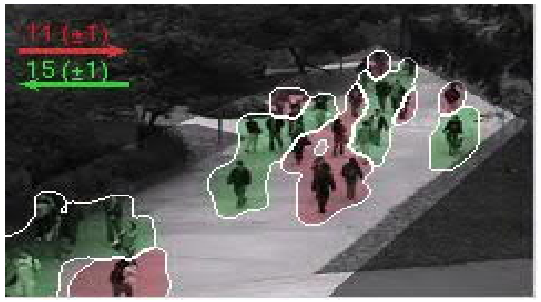


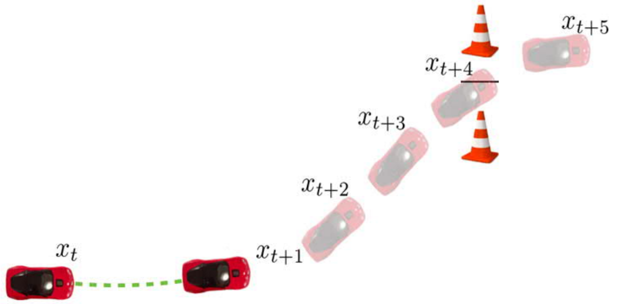
Common solutions: divide-and-conquer

- Then add patches for handling moving pedestrians

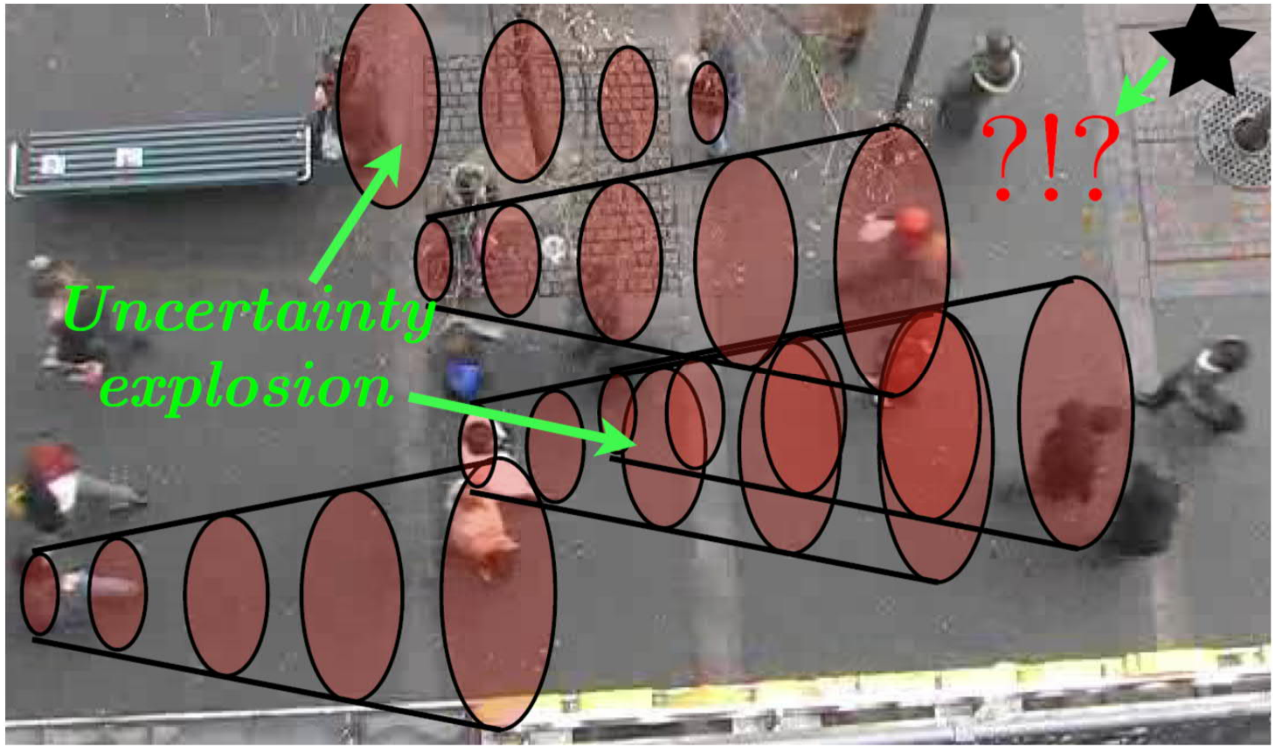


Build a map and localize the robot using SLAM
Motion planning: avoid collisions with static obstacles
Feedback control follows the collision-free trajectory
Use vision and learning to analyze crowds:
1. character recognition
2. trajectory estimation
3. pedestrian tracking
Avoid collisions with pedestrians by using techniques such as Motion Predictive Control (MPC)
Many manual rules to handle corner cases



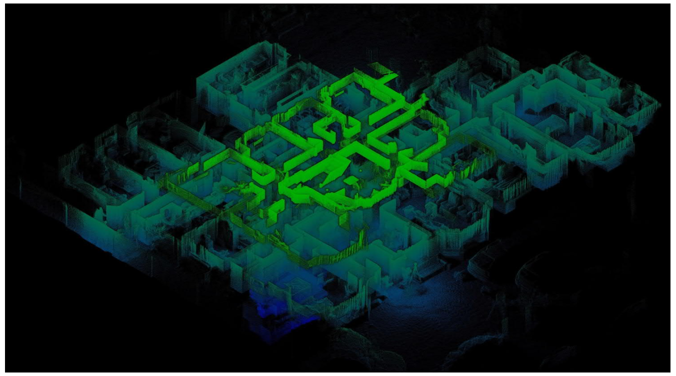


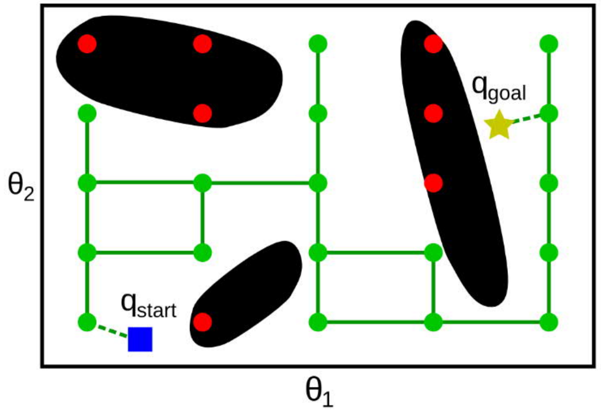
Performance: robot gets stuck



- Each component is not perfect and you need to leave some margin for accumulated uncertainty
- But then uncertainty explosion blocks all moves

Divide-and-conquer issues (I)





Connections of perfect components

$\neq$

High-quality navigation system in dense crowds

Divide-and-conquer issues (II)

- Each component may be solving a much more difficult problem than the task

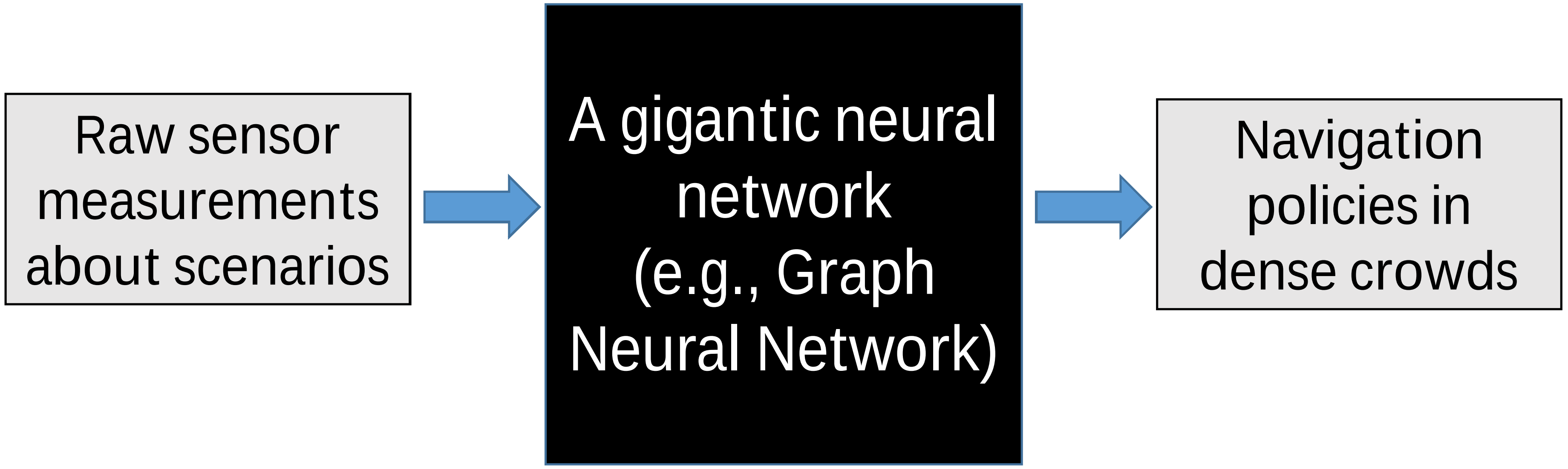
	Pipeline	Bottleneck requirement
Divide-and-conquer	1. Construct map 2. Plan trajectory in the map 3. Avoid collision with moving objects	Accurate robot localization & map: global knowledge about scenes
Better solution ?	Directly solve navigation in dense crowds	Collision avoidance: local information is sufficient, no need for localization or map Moving to a goal: a rough global localization and a rough map; no need for map if the scene topology is simple

Divide-and-conquer issues (III)

- Difficulty of an individual component may be due to ignorance of interaction among components

	Tracking in dense crowds	Localization in dense crowds
Challenges	Occlusion; Re-id	Lost of localization or close-loop features
If having perfect dense-crowd collision avoidance	Always track the people with fewer occlusion or re-id troubles	Always recover lost features by active exploration using collision avoidance

End-to-end RL solutions?


Raw sensor measurements about scenarios
A gigantic neural network (e.g., Graph Neural Network)
Navigation policies in dense crowds

- Difficult to transfer from simulation to real world
- Requires huge number of training data / experience
- Discard most structural knowledge in divide-and-conquer

Re-investigate our task:

What is the core capability required in “navigation in dense crowd”?

Let’s get some idea from human

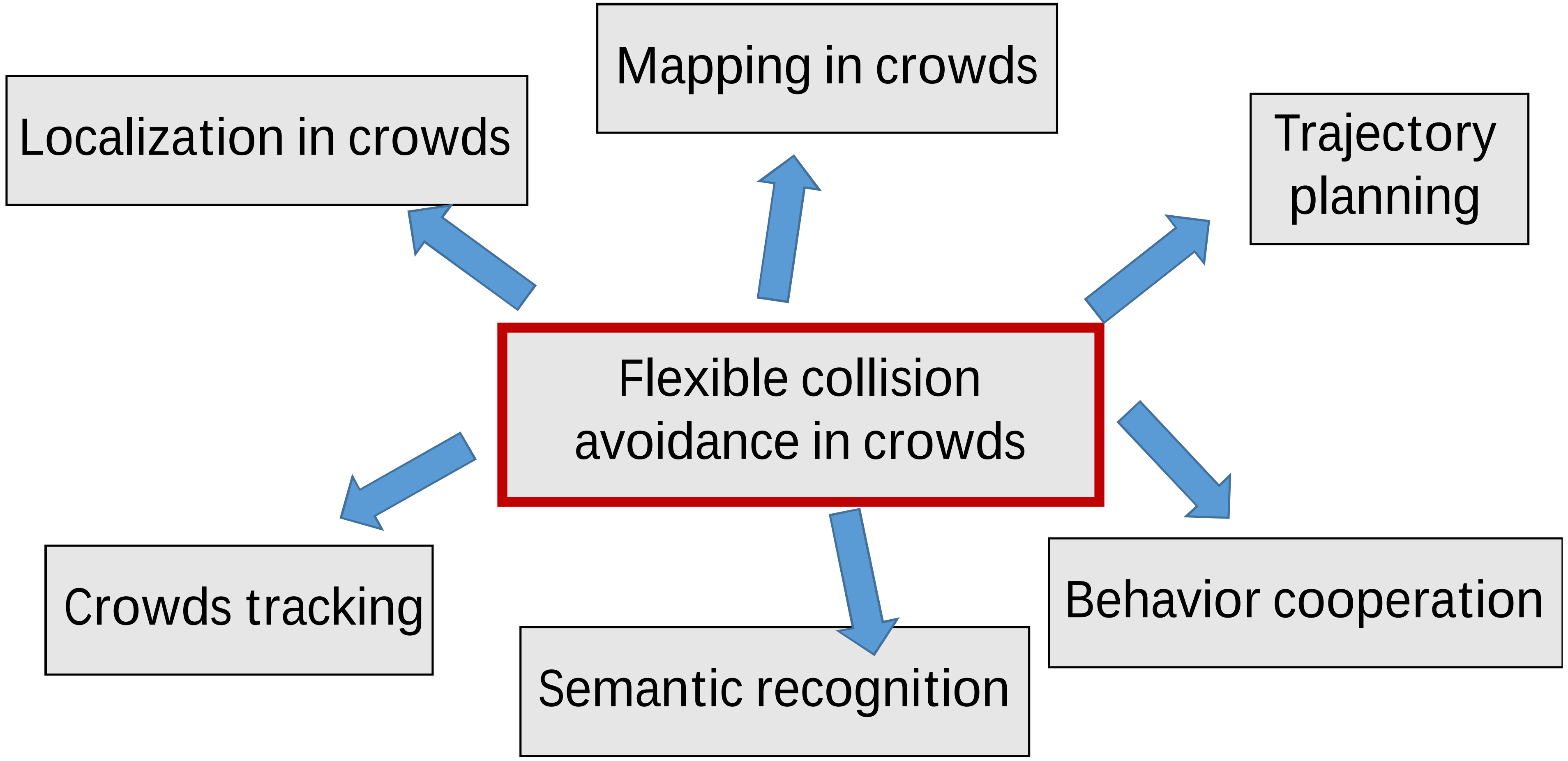
What is human's core skill for navigation in dense crowds

The cooperated collision avoidance
without central guidance or communication

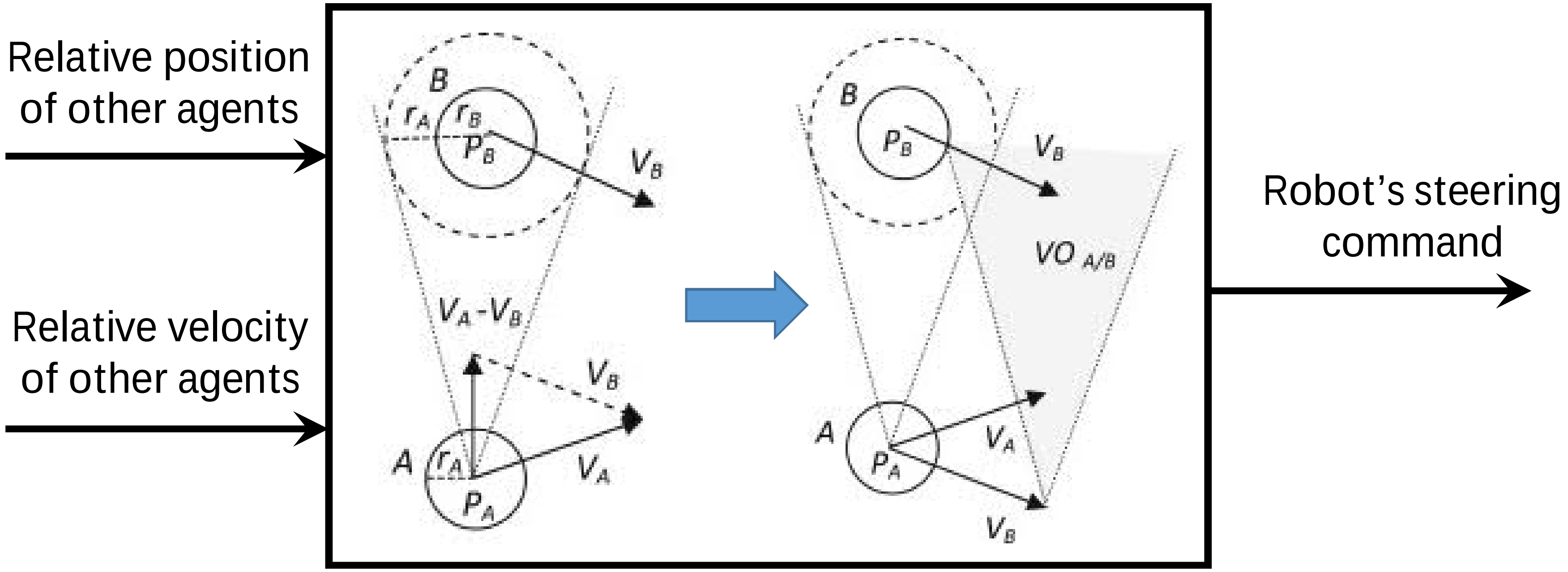
What is human's core skill for navigation in dense crowds

The cooperated collision avoidance
without central guidance or communication

Lets first solve this core task

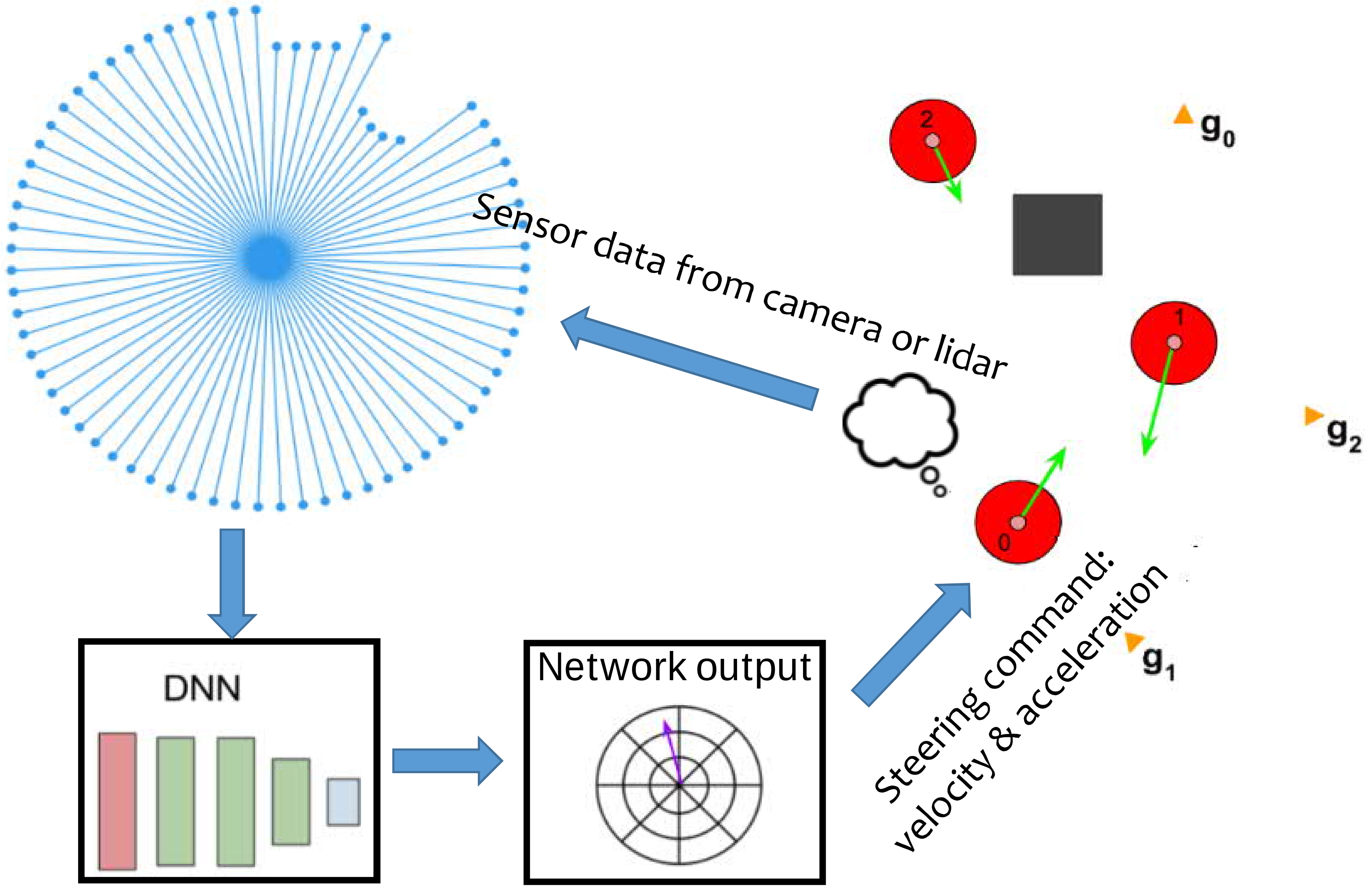


Human-like decentralized collision avoidance



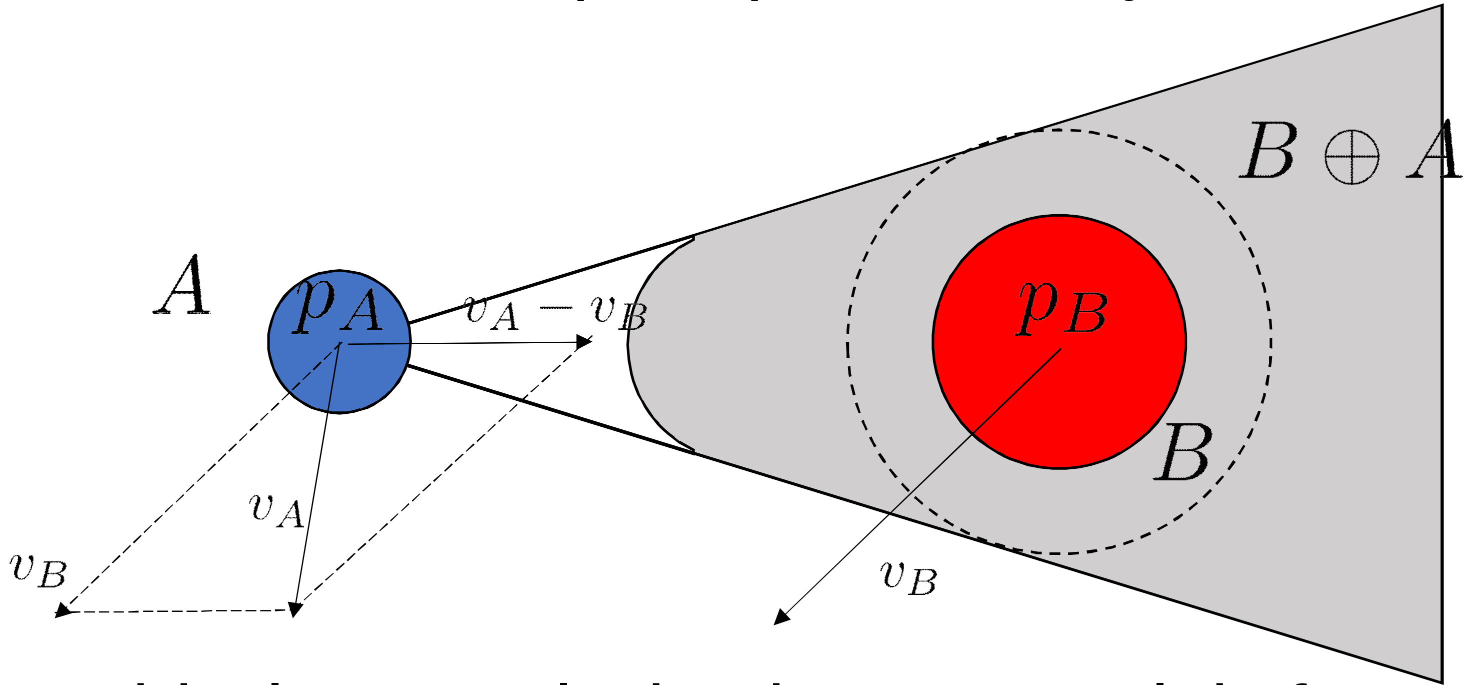
- Prefer to avoid traditional difficulties like robust recognition and position/velocity estimation

Use raw sensor data


Sensor data from camera or lidar
DNN
Network output
Steering command: velocity & acceleration
0
1
2
g_0
g_1
g_2

How to find this controller

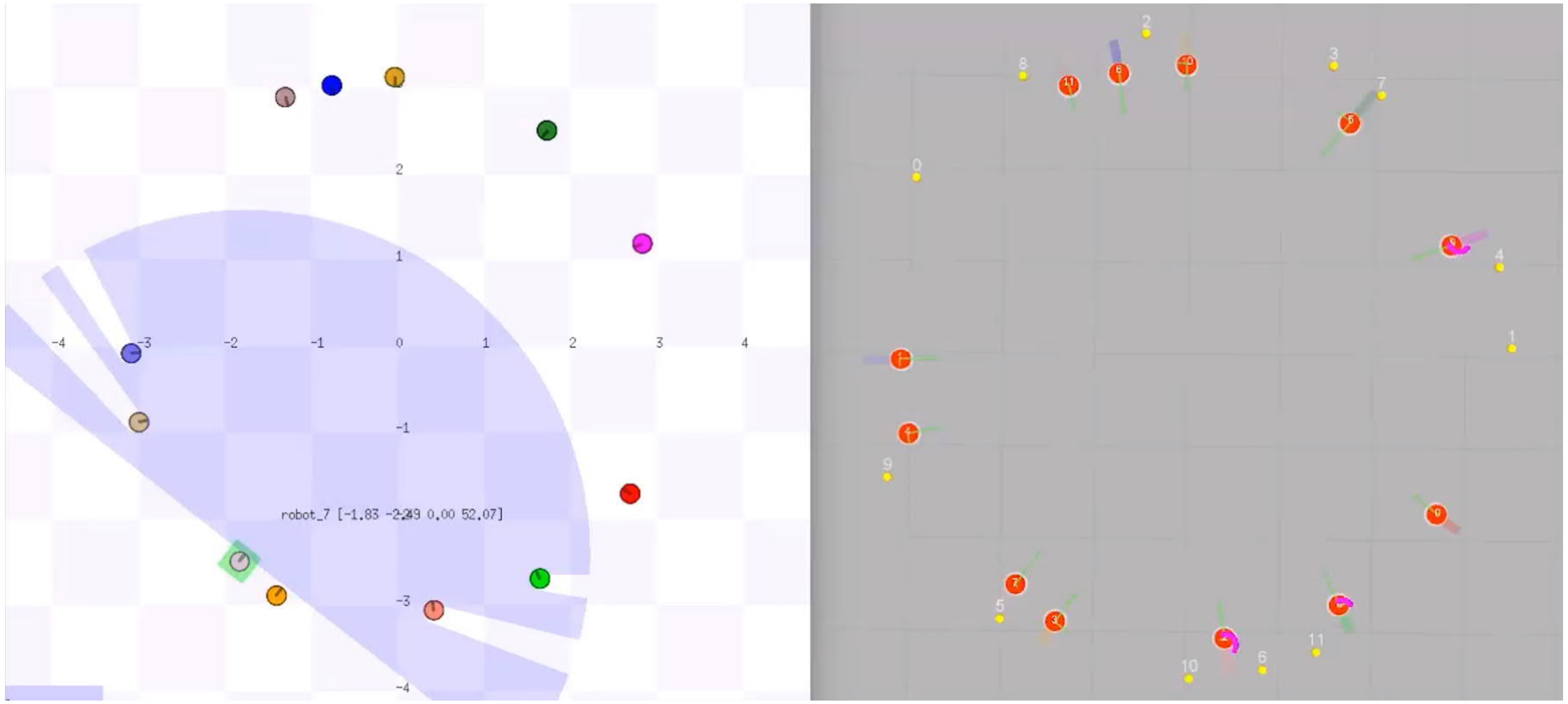
- One way is to build a collision avoidance model based on the first principle – velocity obstacle



- Intuitively, any velocity that can result in future collision in a window $[0, t]$ should be discarded.

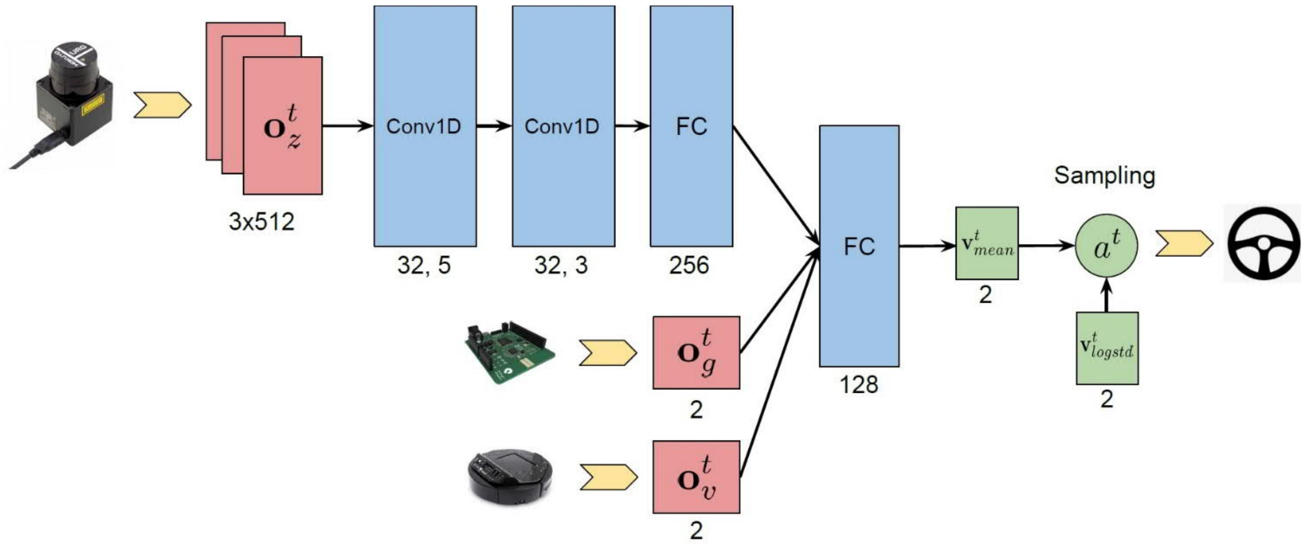
How to find this controller?

- But learning is more appropriate
  - Rich sensor + complex trade-off between safety and efficiency – first principle is limited and has many parameters to be tuned
  - Rough localization is sufficient for learning-based policy



Learn controller using RL

- Neural network controller

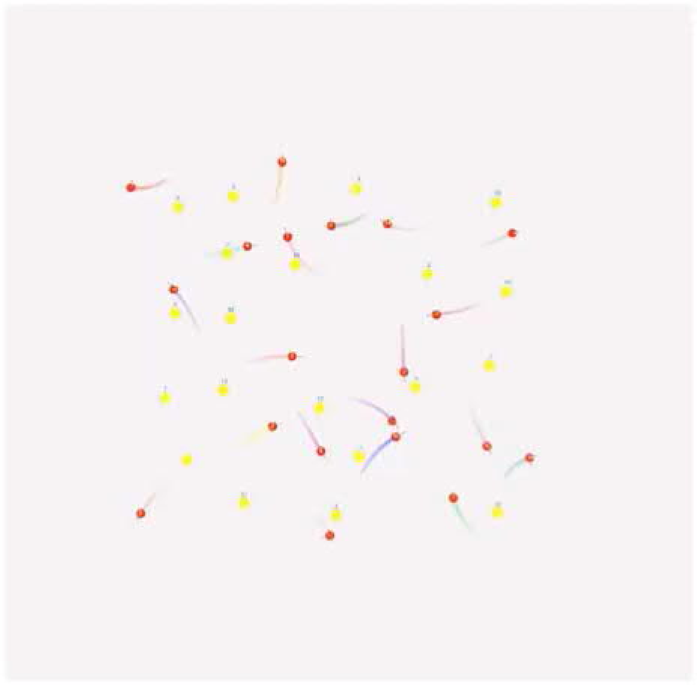


- Optimize the entire crowd for:
  - Minimizing collision cost
  - Minimizing the time to reach goal
  - Maximizing trajectory smoothness

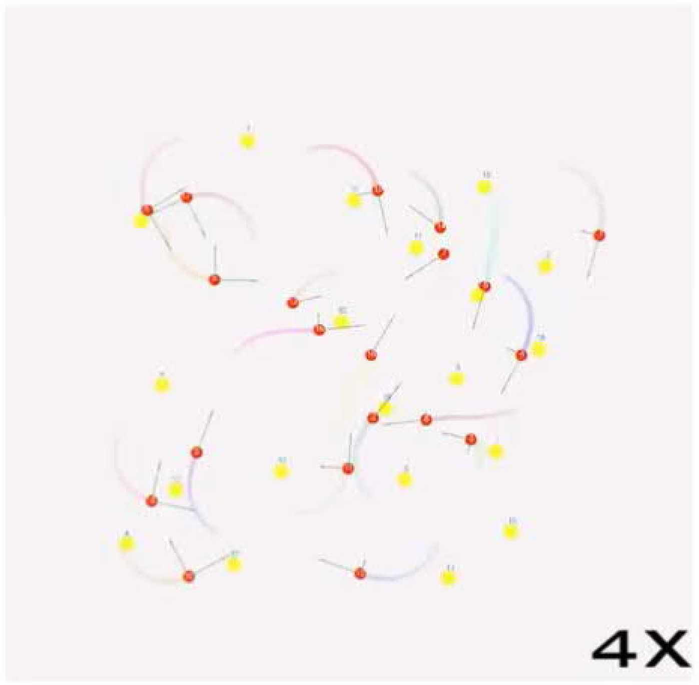
$\Bigg\}$ encourage smooth cooperation

Training step 1

- “Baby” controller trained on simple scenarios
- Random configuration for start and goal (yellow)



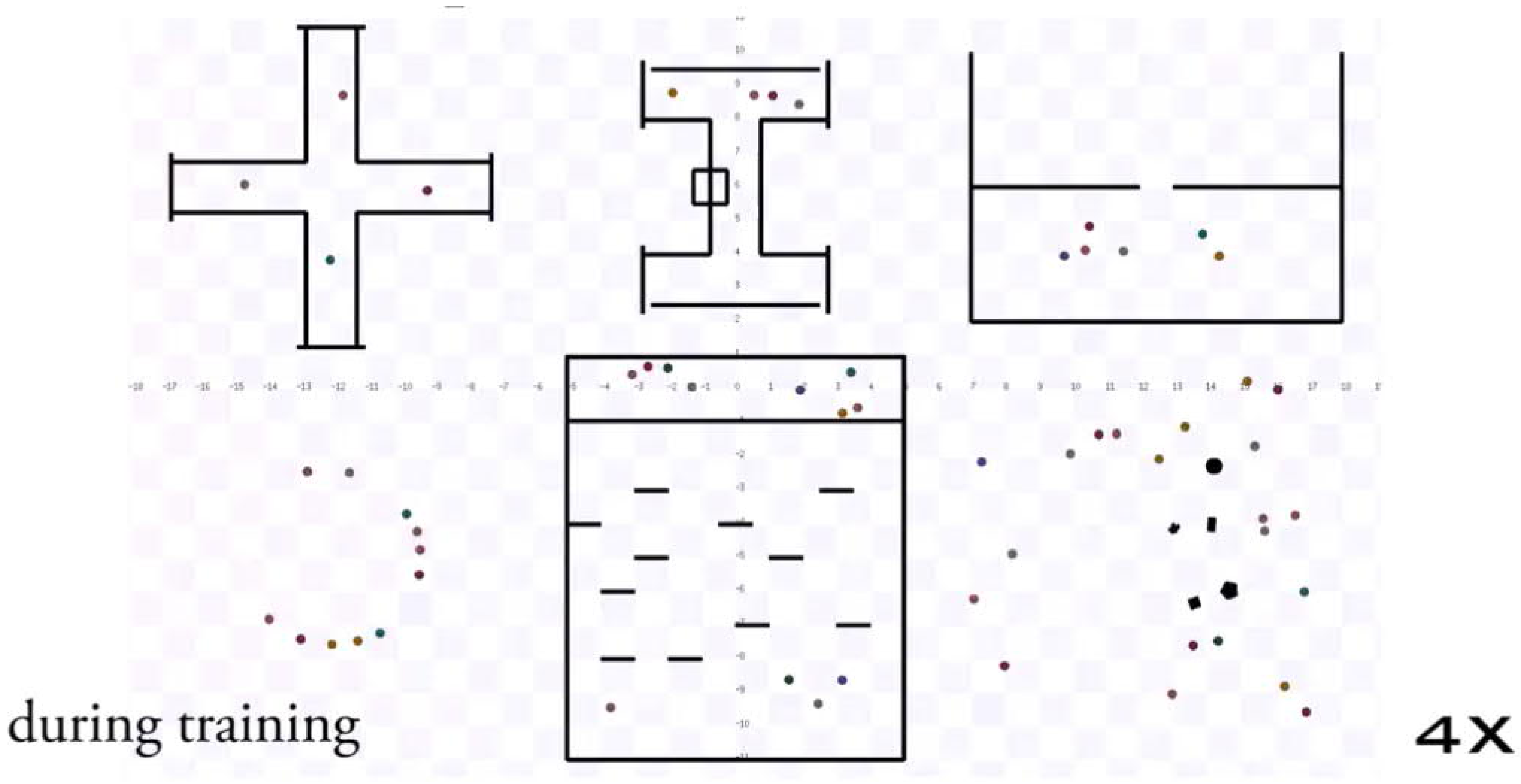
During training



Training finished

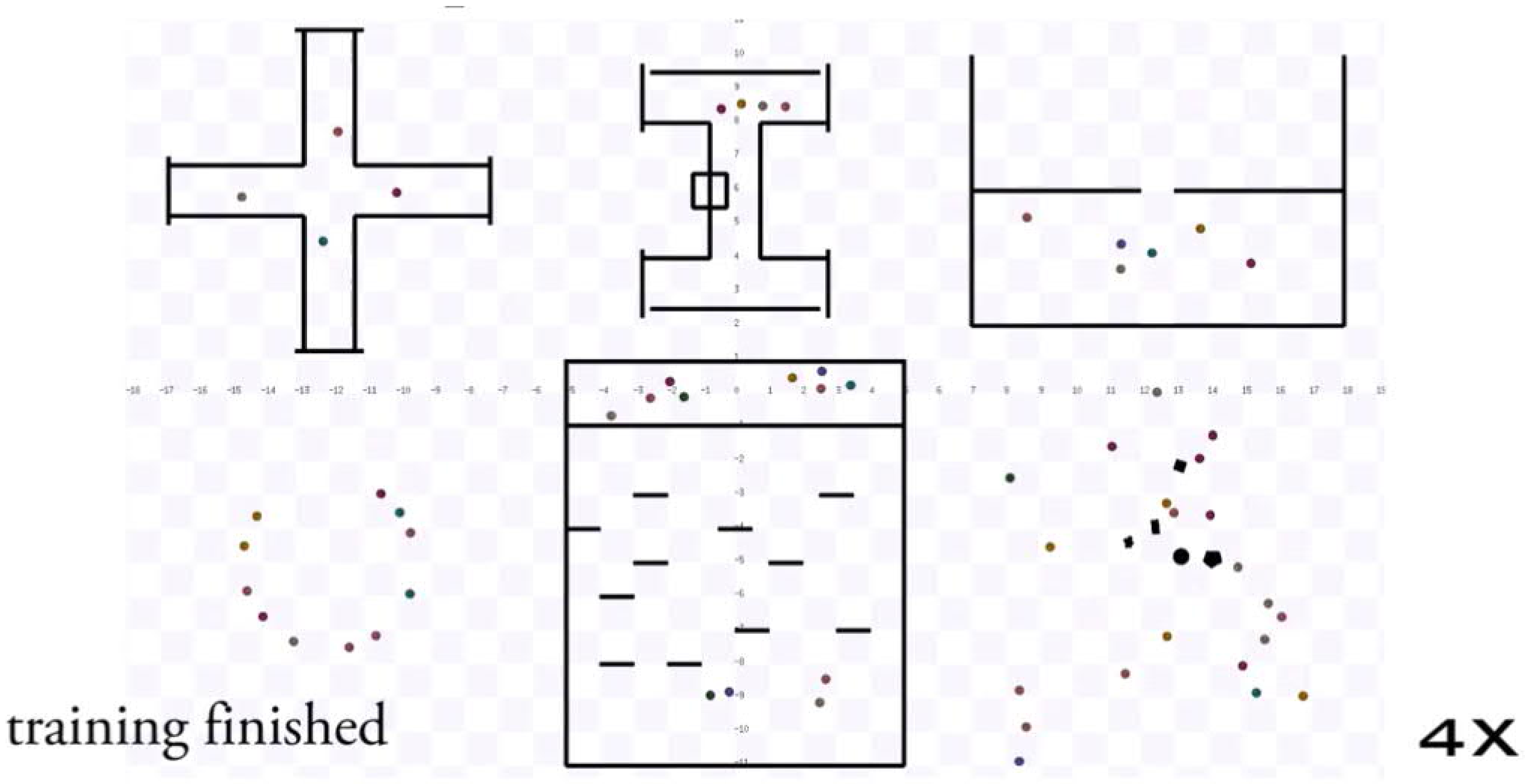
Training step 2

- Controller gets “mature” after being trained on multiple challenging scenarios

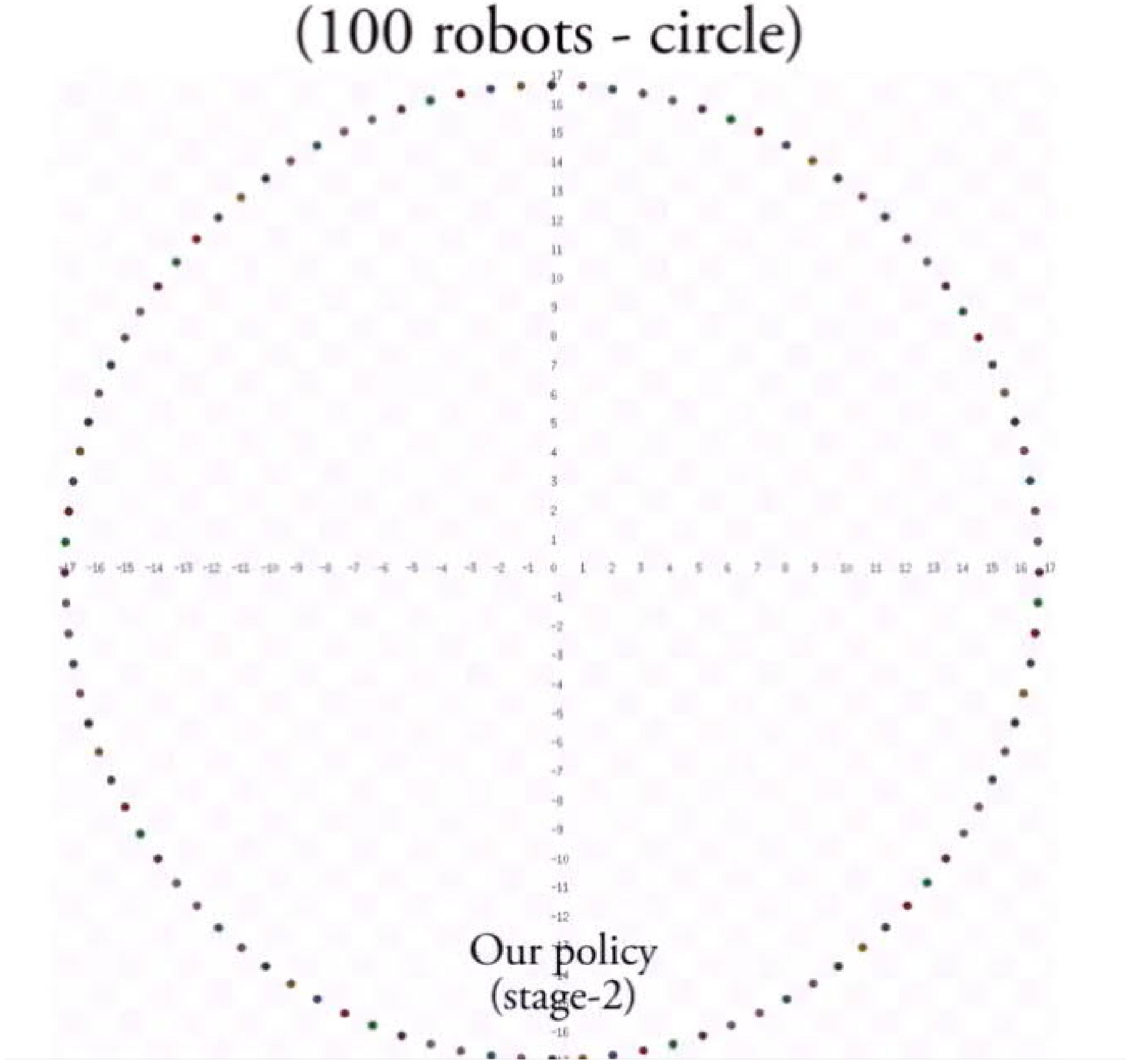


Training step 2

- Controller gets “mature” after being trained on multiple challenging scenarios



Test in a challenging benchmark

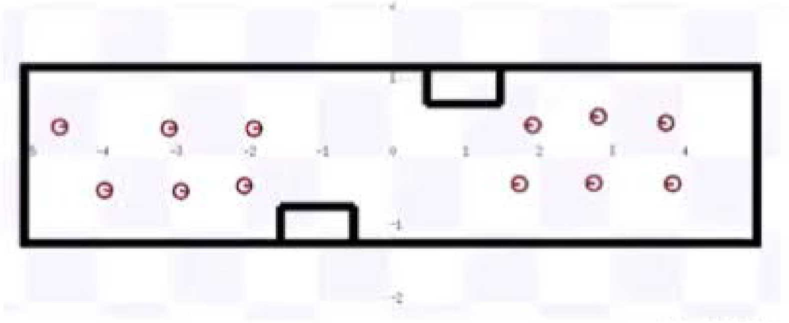
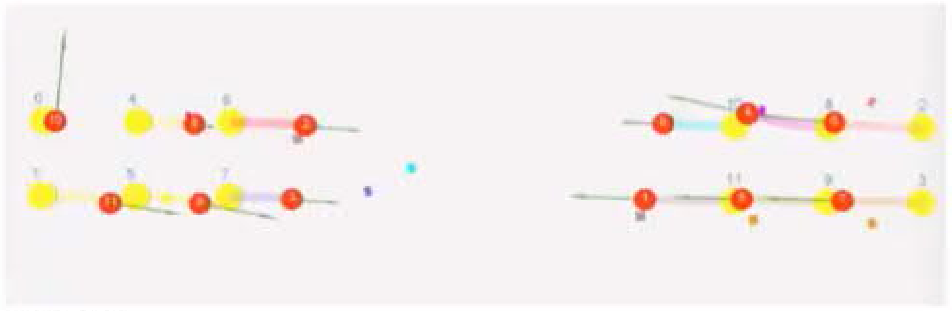


Each robot moves from a position on the circle to its dual position on the circle.

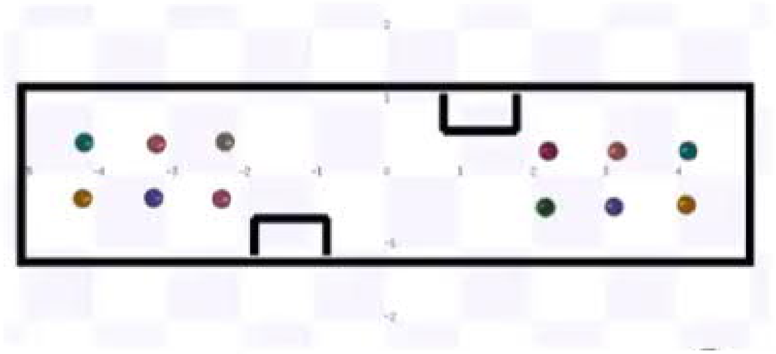
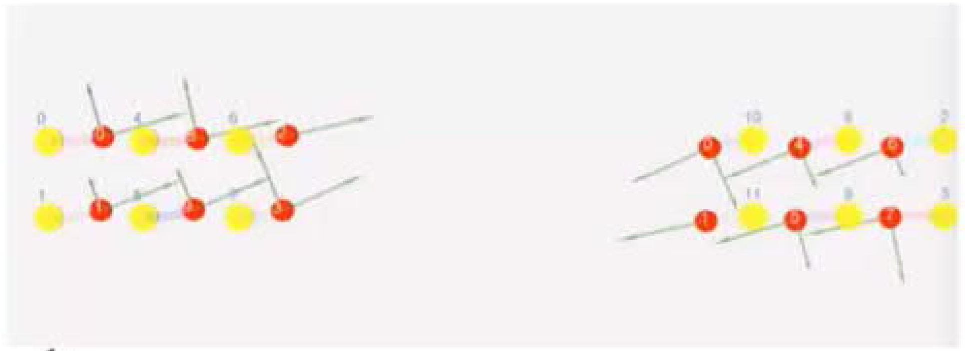
The robots behave cooperatively to resolve the congestion caused by partial observation to the environment.

Unknown map

Effective navigation is possible without map,
if the topology is simple

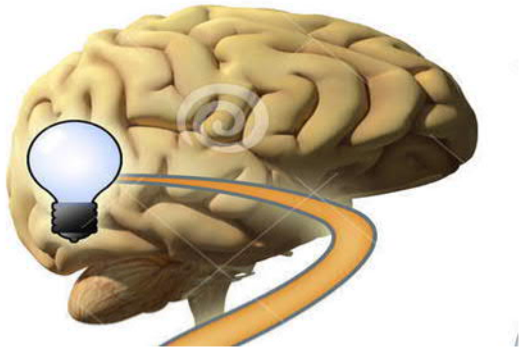



Traditional methods based on
collision avoidance rules

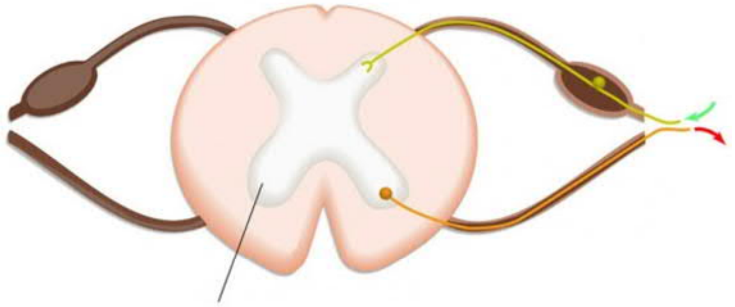



Our reinforcement learned policy

Pure learning is NOT enough

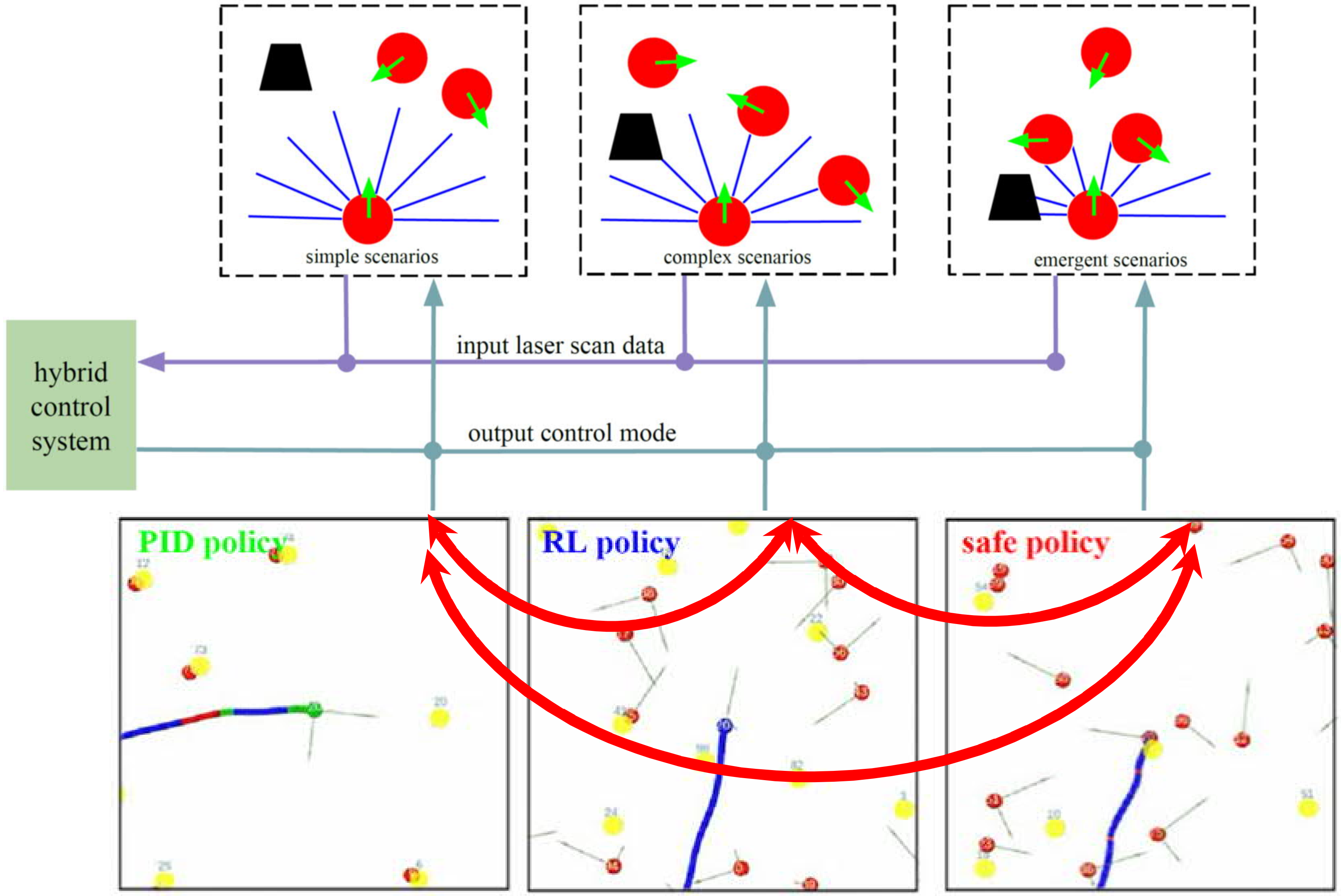


- Learning based policy
  - Make complex trade-off between navigation safety and efficiency
  - Difficult to respond in time and optimally for extreme cases



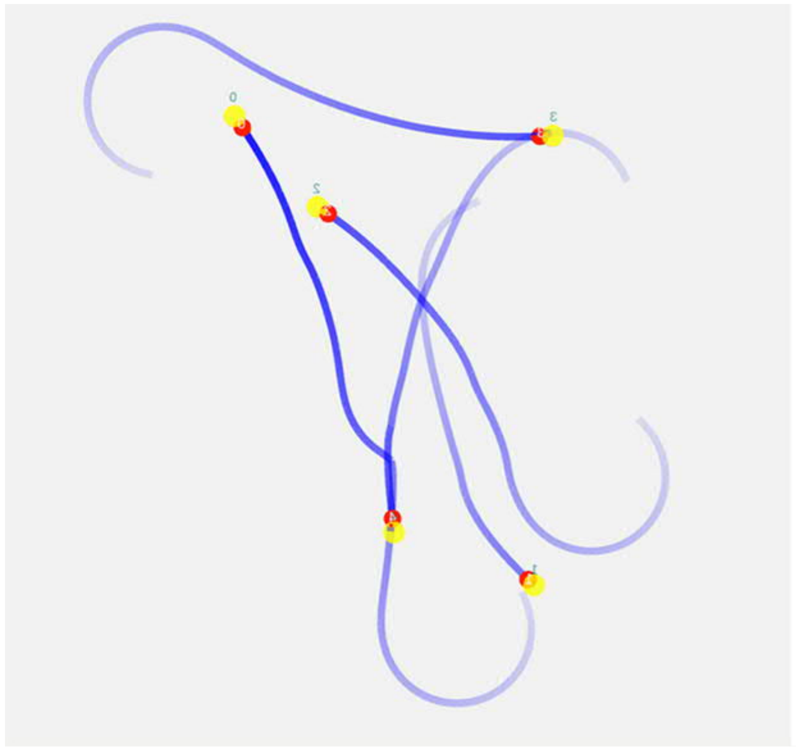
- Low-level control based policy
  - Aggressive optimal control when the moving obstacles are of small-number, distant, and slow
  - Conservative safe policy when the moving obstacles are of large number, close, and fast

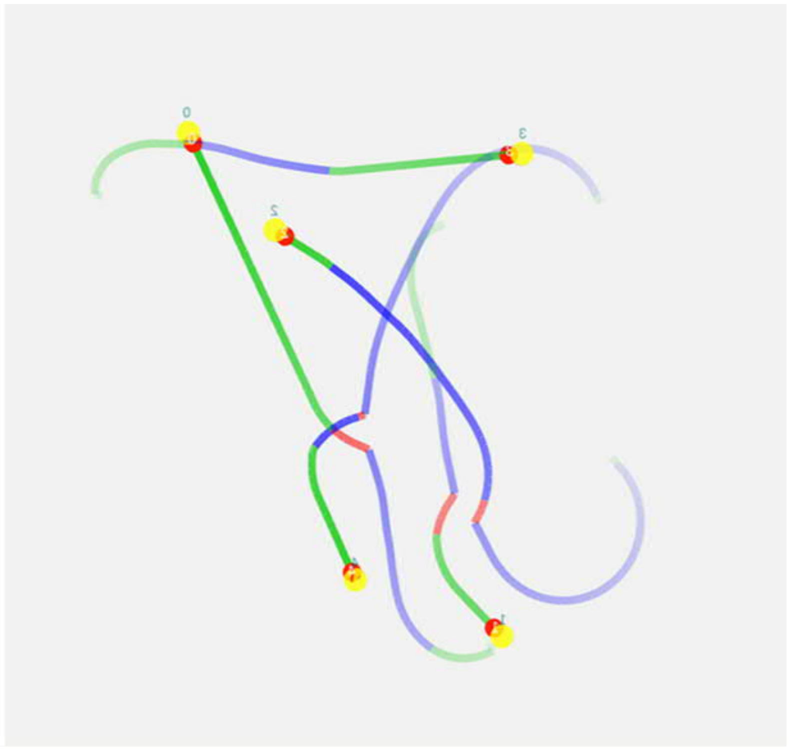
Hybrid control



Benefit of hybrid control

Shorter trajectory, smaller curvature, and lower collision probability





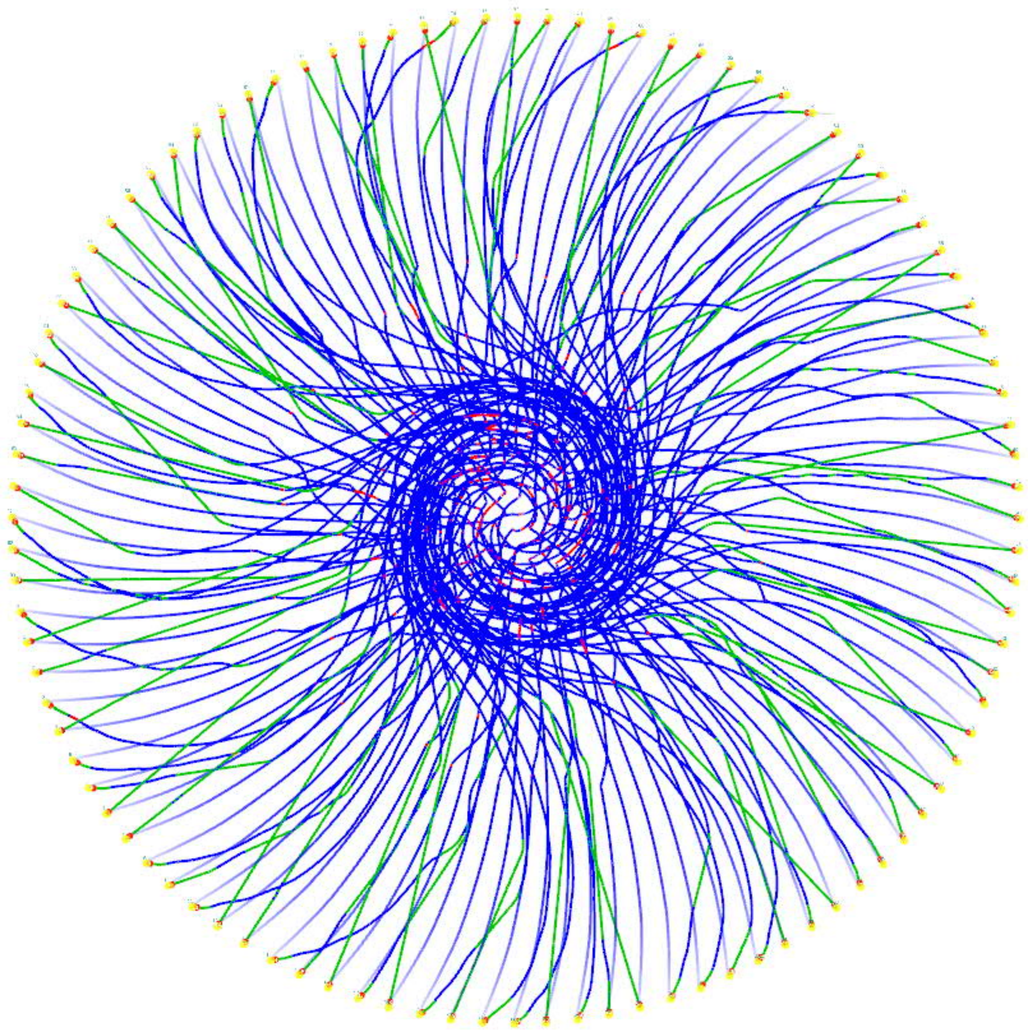
RL

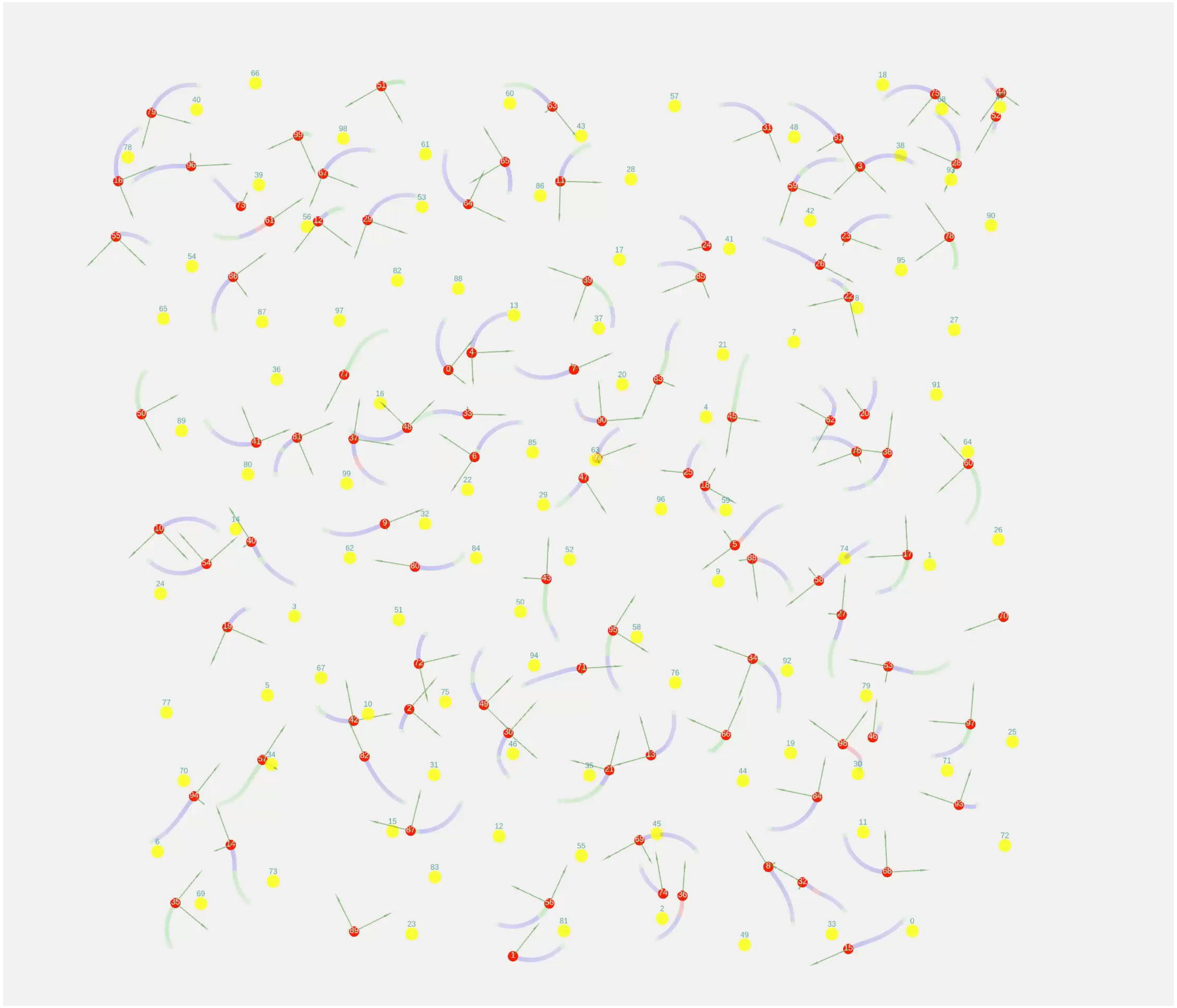
Hybrid-RL

PID control RL control Emergent control

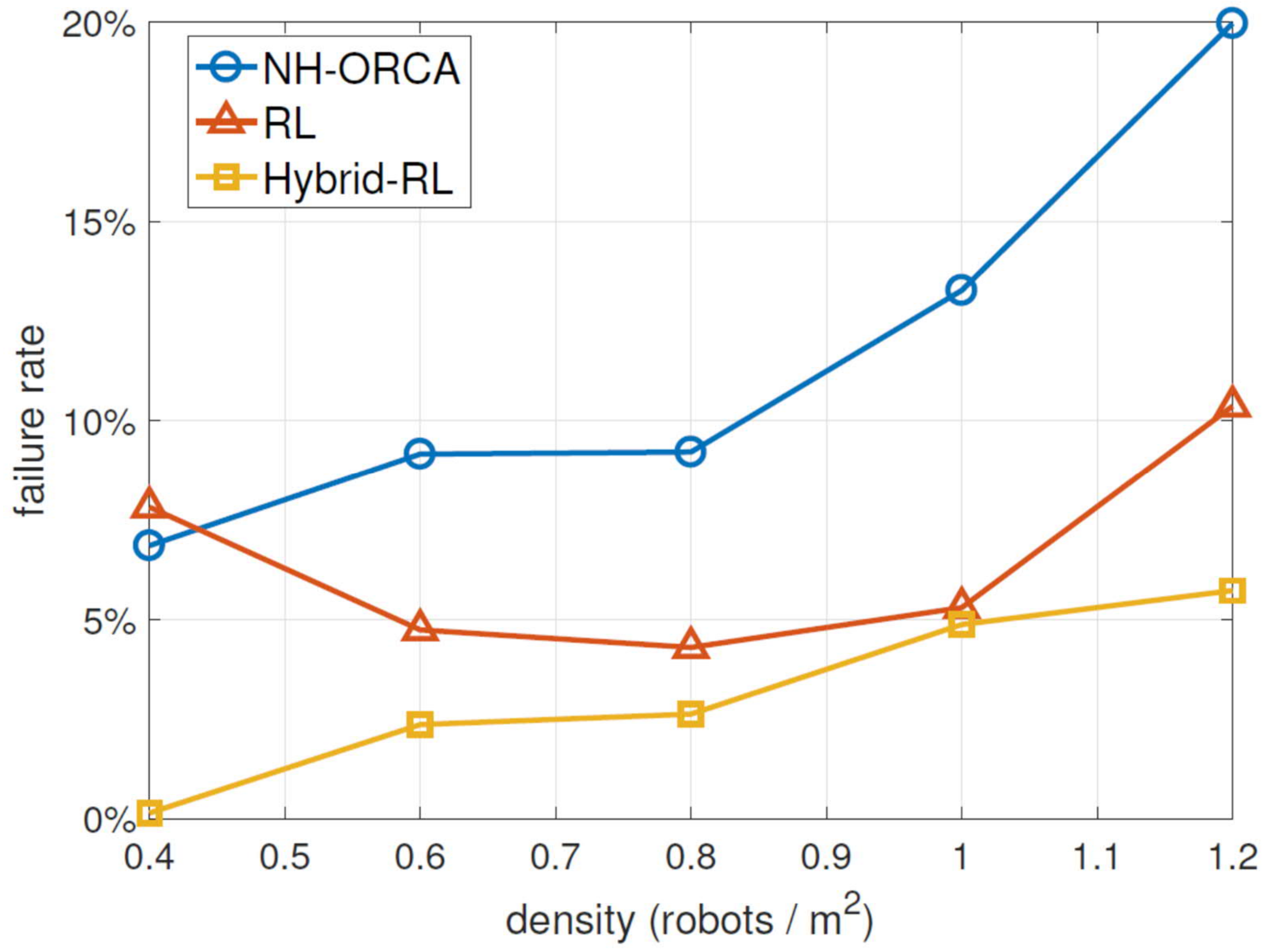
Multi-level swarm behavior

- Close to center
  - mainly safe policy (in red)
- Close to periphery
  - Mainly PID policy (in green)
- Between center and periphery
  - Mainly RL policy (in blue)



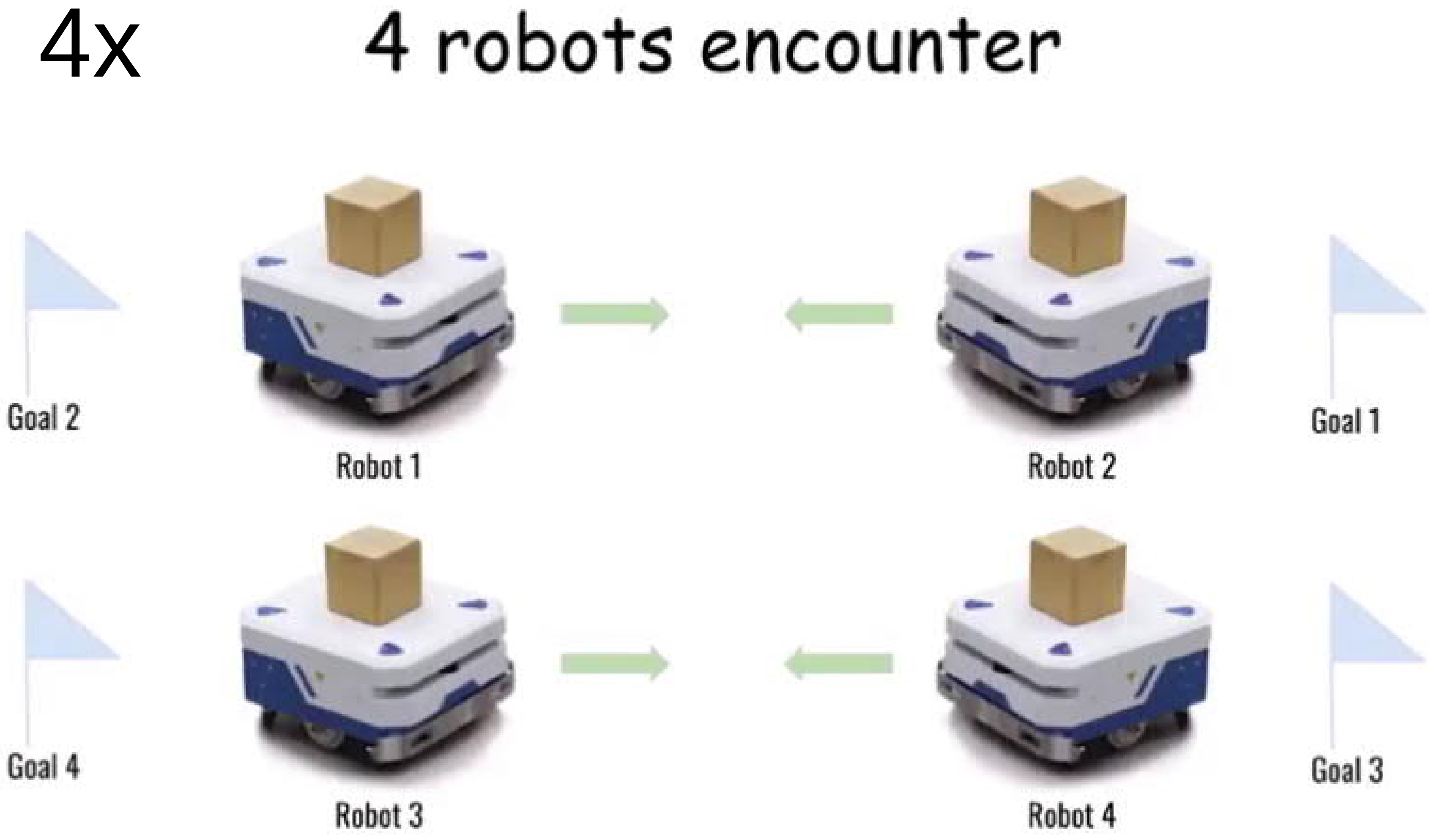


Performance under different density

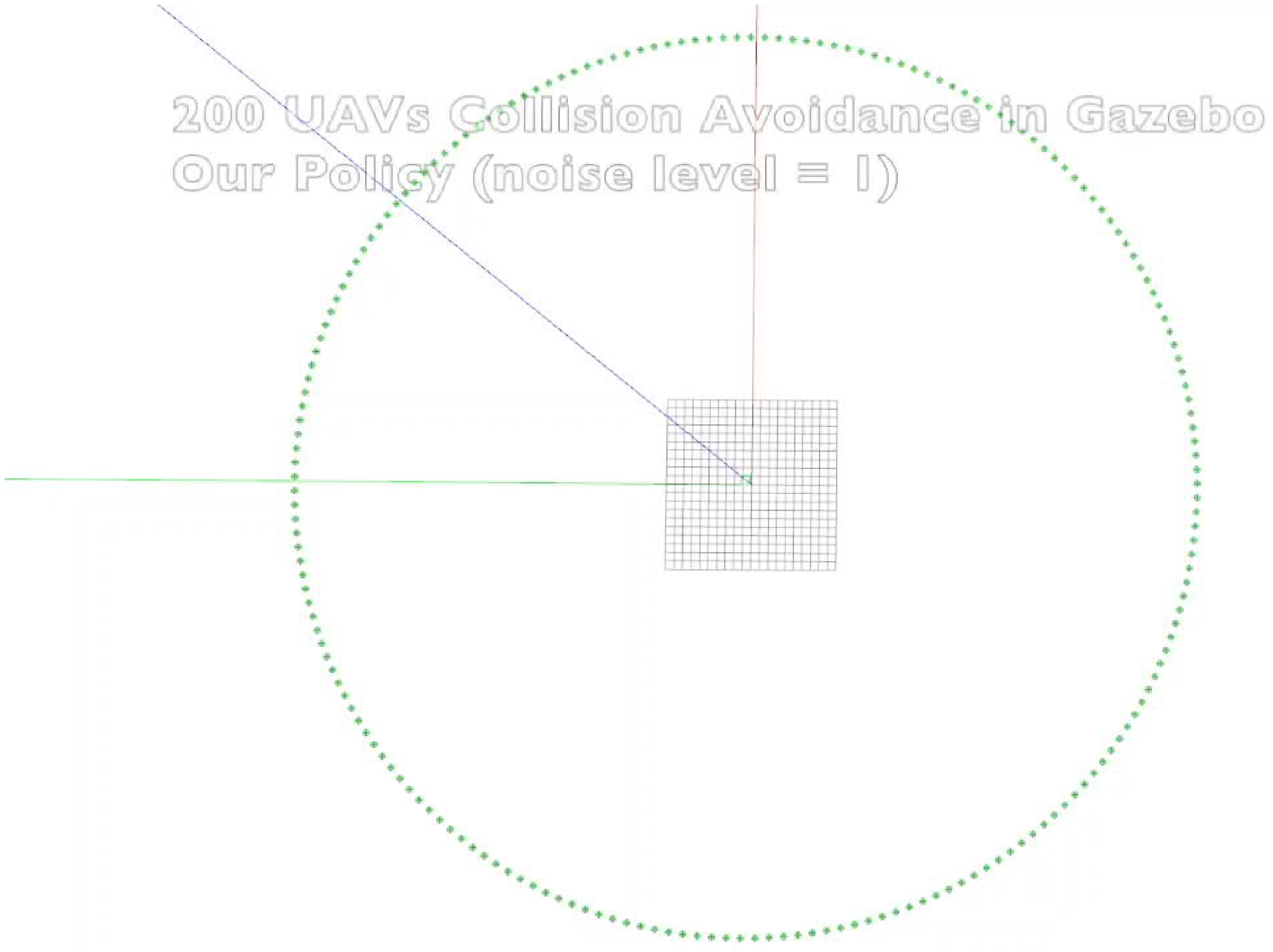


Real robot demo: UWB for rough global localization

4x



Extension in 3D: multiple drone collision avoidance


200 UAVs Collision Avoidance in Gazebo
Our Policy (noise level = 1)

Robot has learned:

Avoid collision with moving obstacles in a flexible way

Cooperate with moving obstacles in a clever way

Track a target in human crowd

- Let’s first see how the robot’s performance when only considering human as dumb moving objects

- The guider has a Ultra-Wideband (UWB) label in his hand

- The robot tries to follow the location of the label

Indoor test

Limitation?

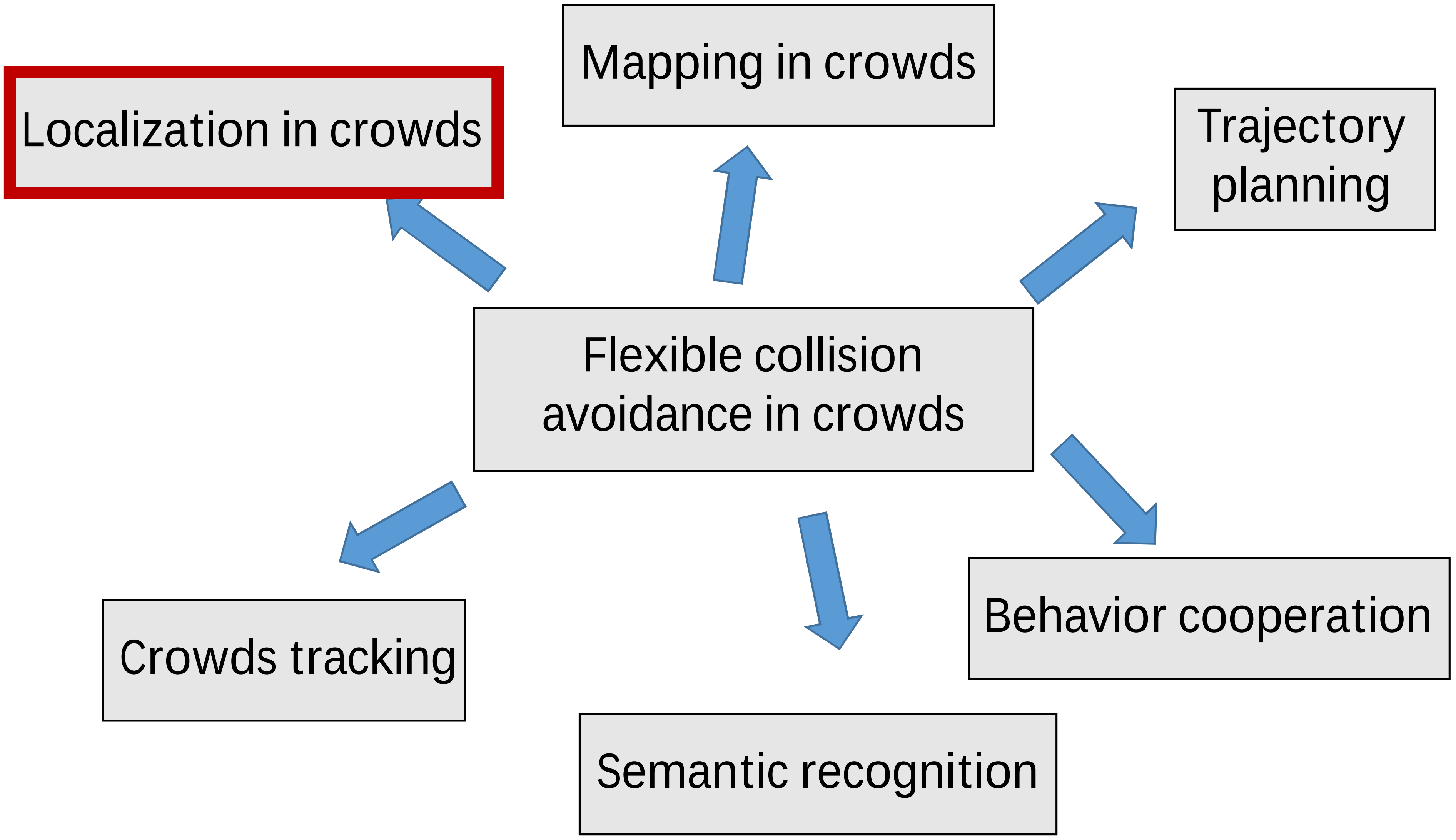
- Does not consider human's preference
- The robot is overly responsible for collision avoidance

Outdoor test

Escape from the surround

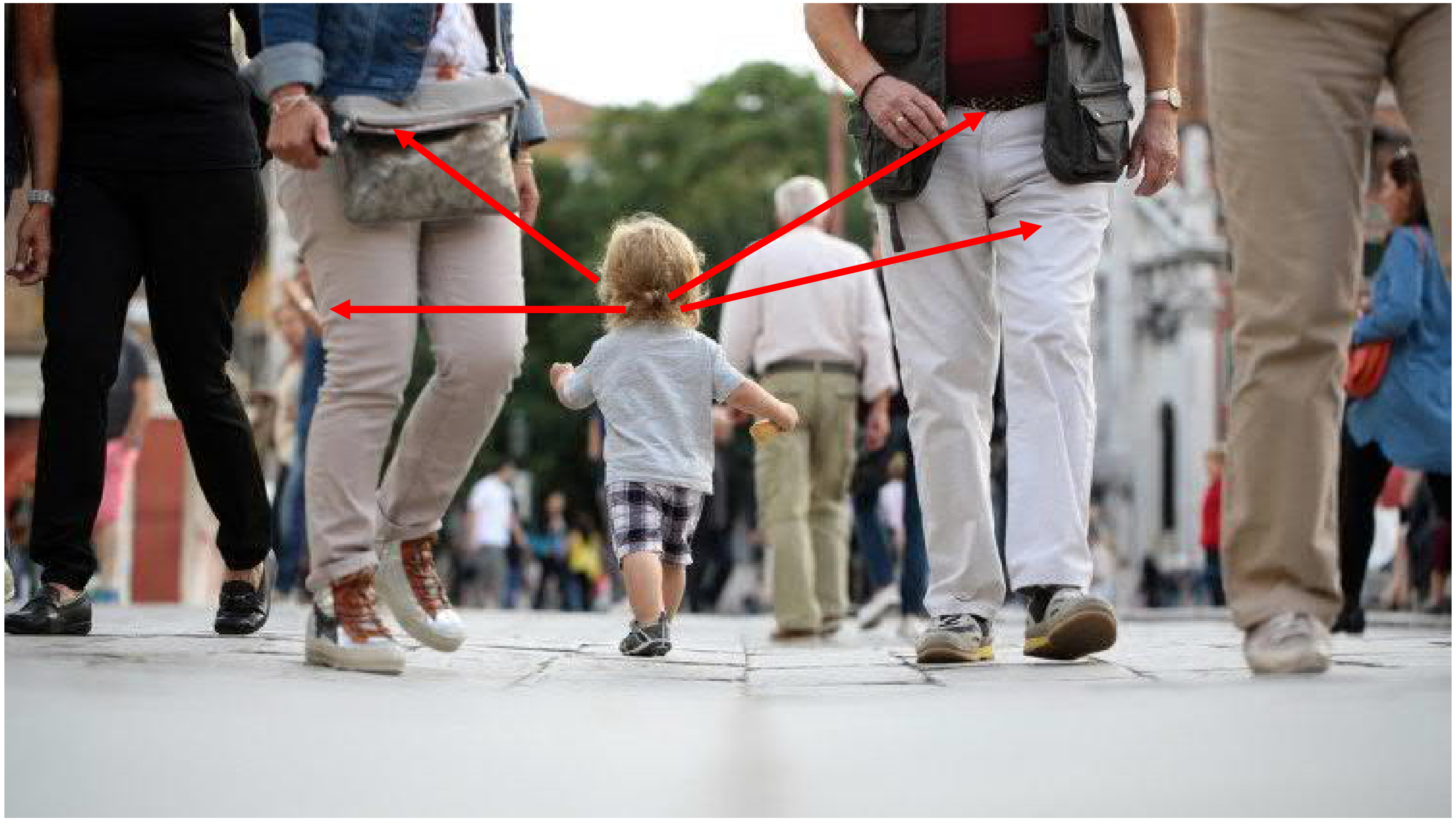
Test on a legged robot

Collision avoidance helps challenges in other components



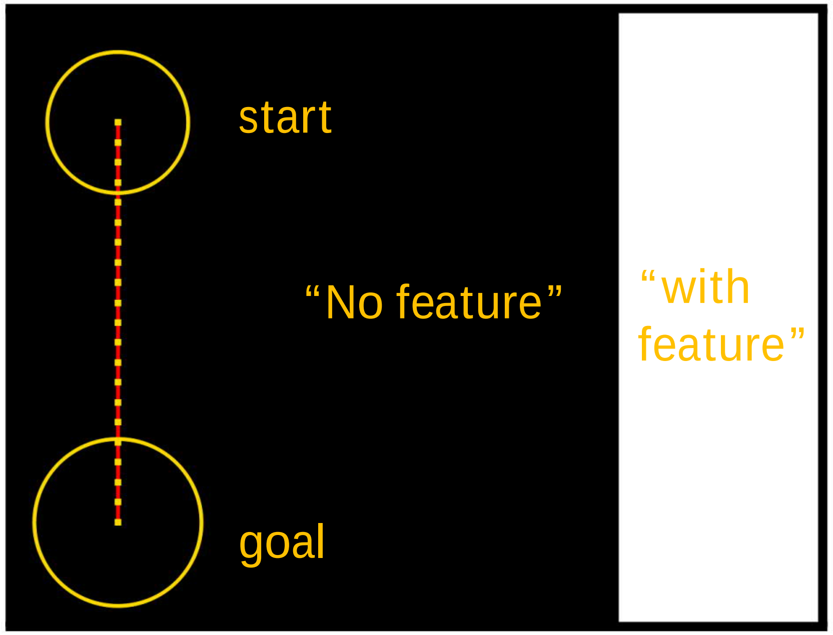
Traditional localization in dense crowds

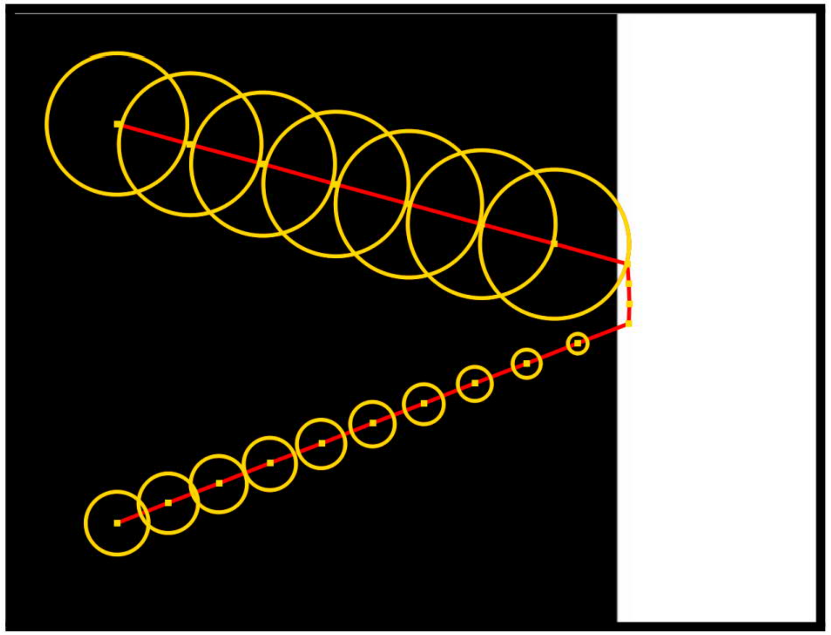
- Need to overcome the static feature lost difficulty



Theoretical solutions

- Gather information by moving to regions with rich localization features, and then continue





Chicken-and-egg difficulty

- But how to execute such “information gain” movement in a dense crowd?

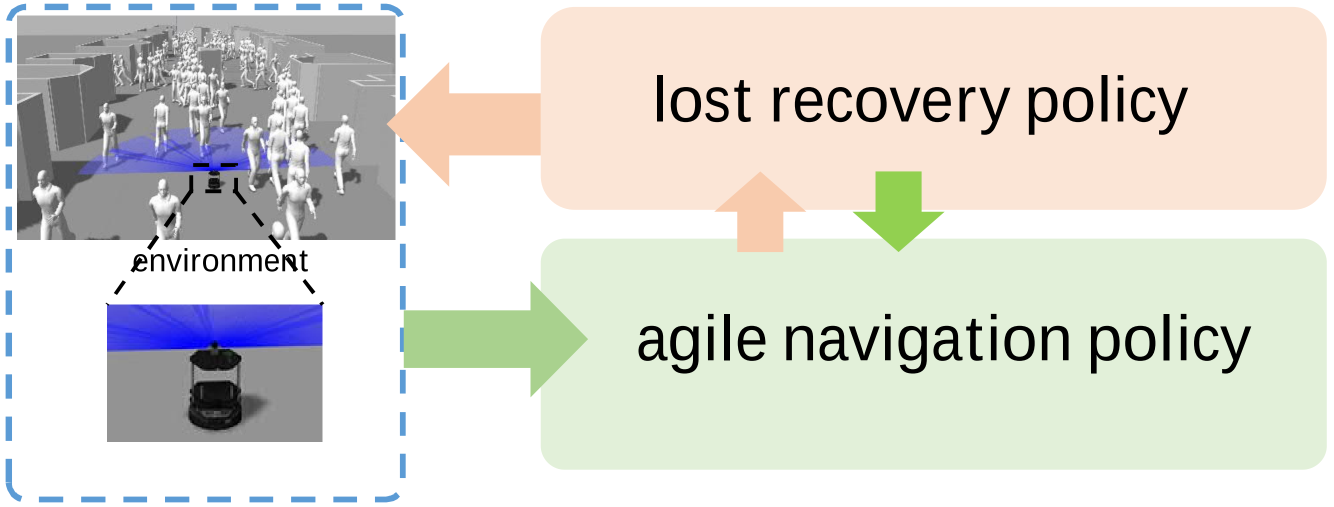
Lost localization and at most has a rough localization from inertial odometry

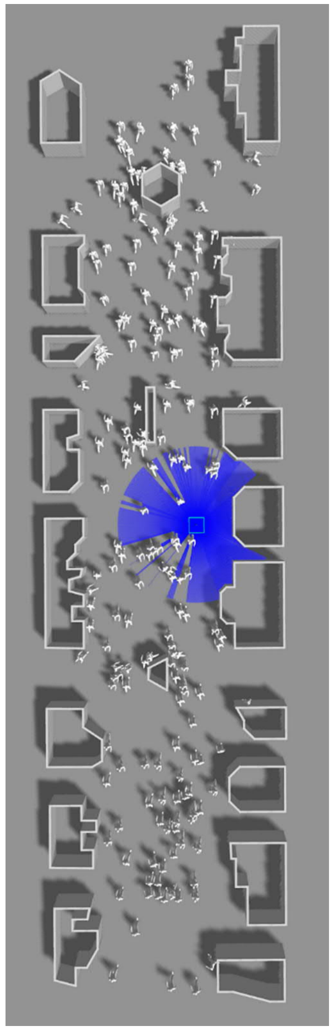
Cannot navigation through crowds to the region with “rich feature for localization”

Cannot navigation through crowds to the region with “rich feature for localization”

Cannot recover localization

Our solution



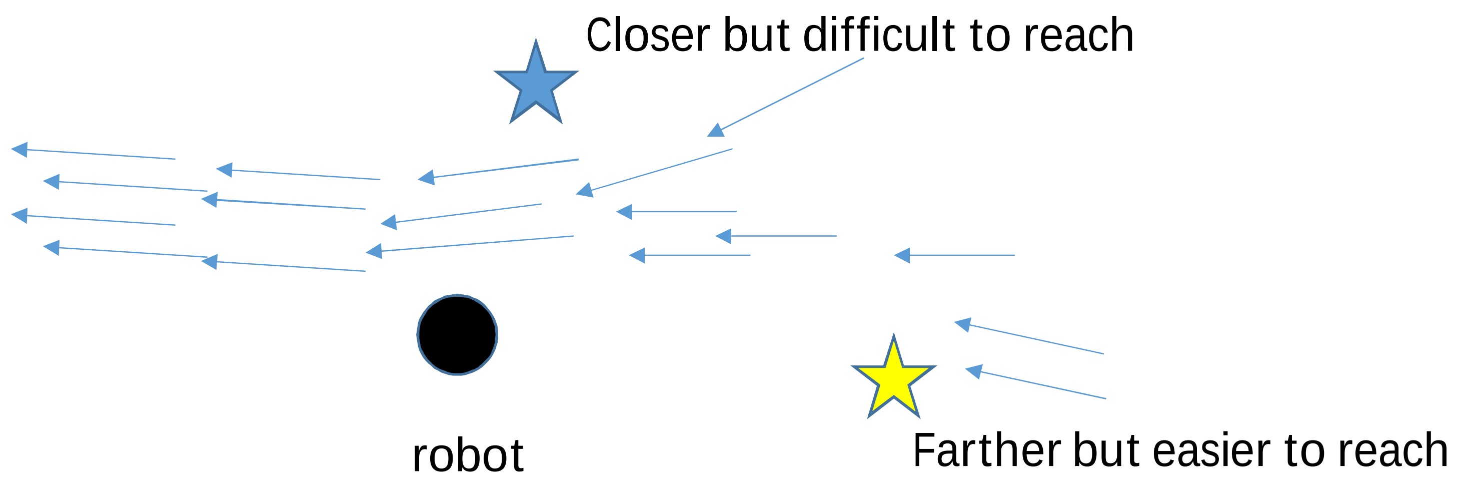


- Our collision avoidance does not rely on localization
- Our "moving to the goal" can safely use a rough localization to indicate where the feature-rich region is

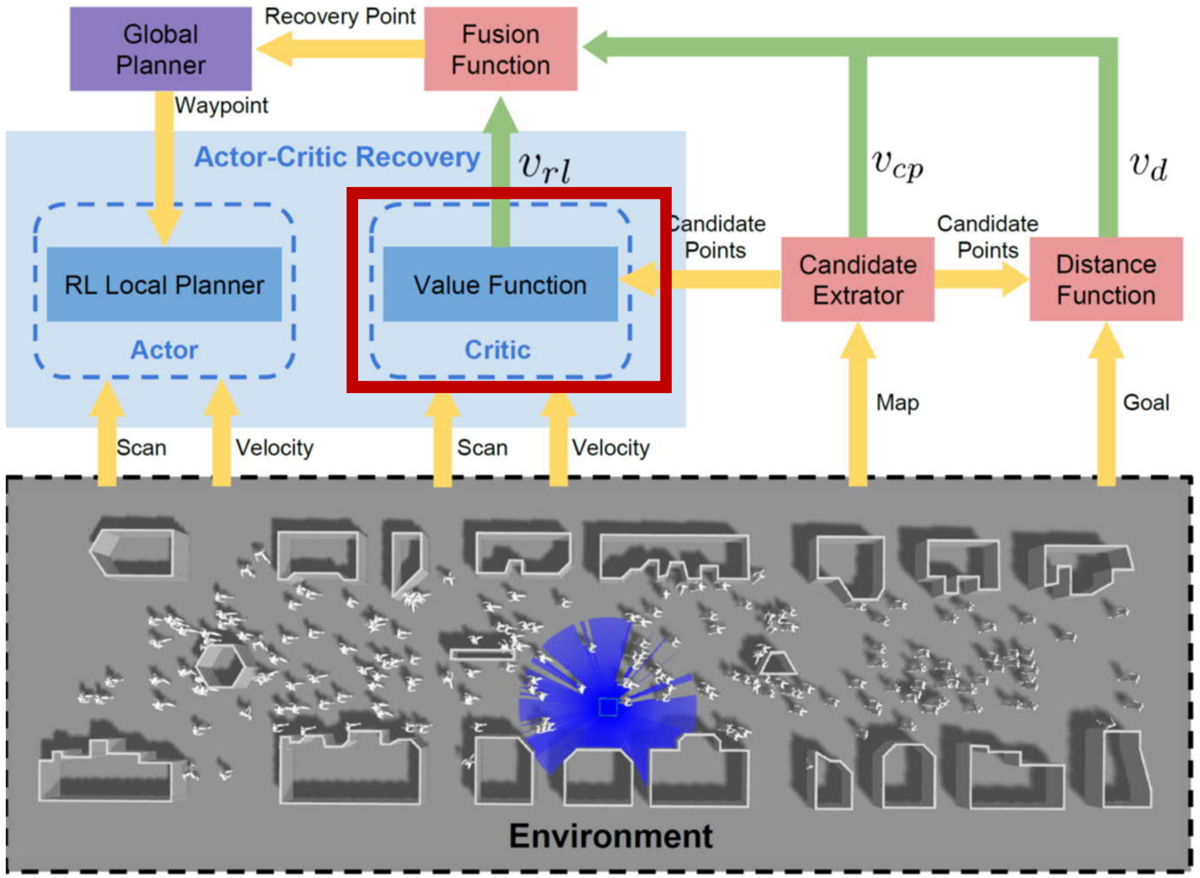
Chicken-and-egg problem solved

Localization recovery policy

- Choose a region in the map as a temporary goal for observing features and recover localization
- Trade-off different regions according to:
  - The distance to the robot
  - The richness of features
  - How difficult for the robot to pass through the crowd flow and reach the region



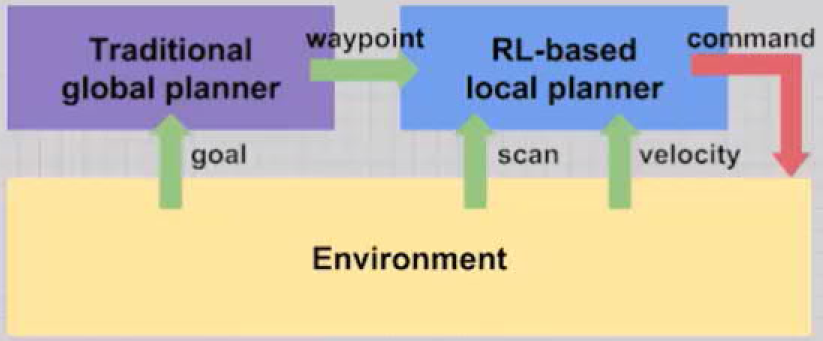
Actor-Critic recovery

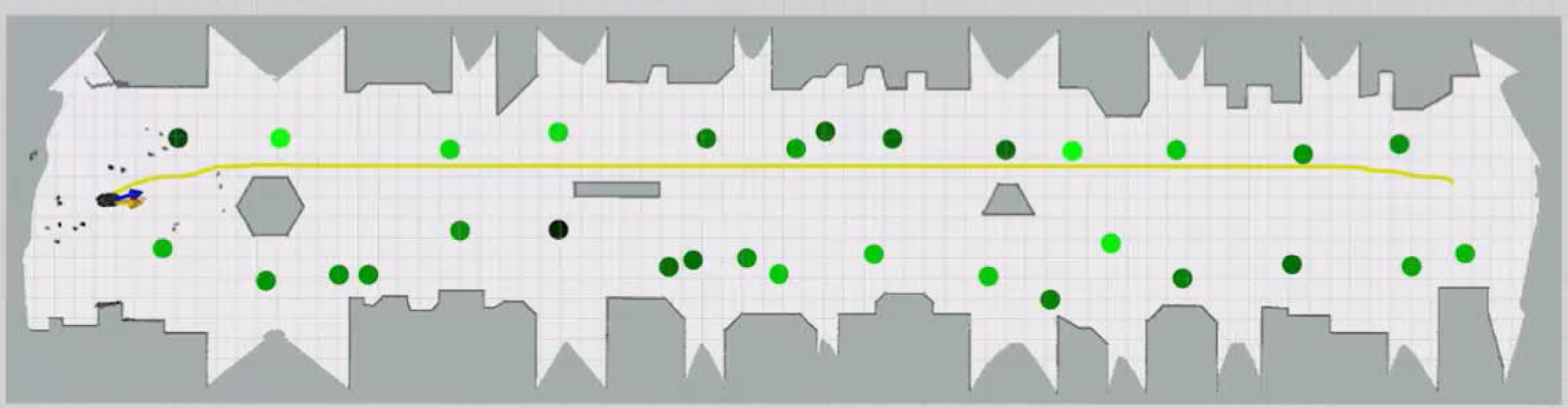


The crowd-related criterion can be computed using the value function, a by-product of collision avoidance policy training

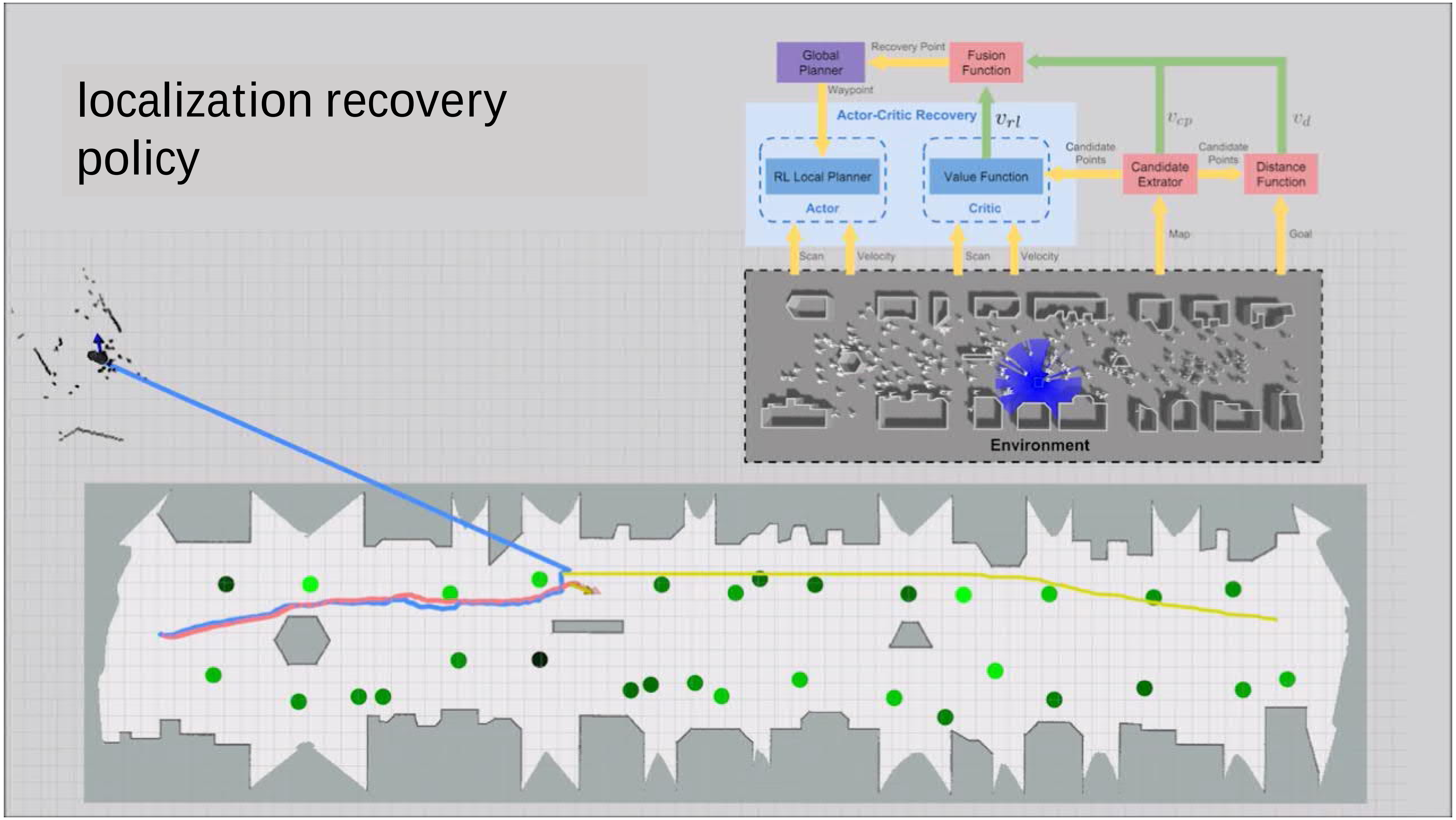
Adaptively update policy according to the crowd flow status

agile navigation policy

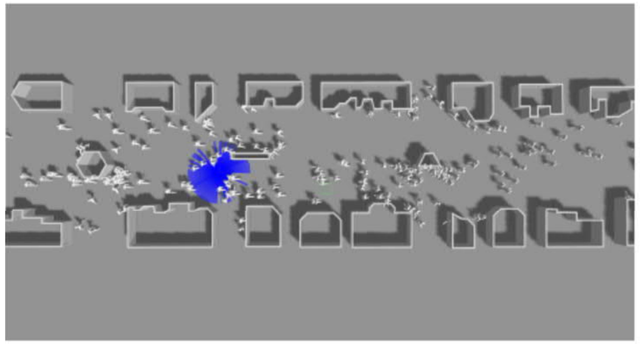


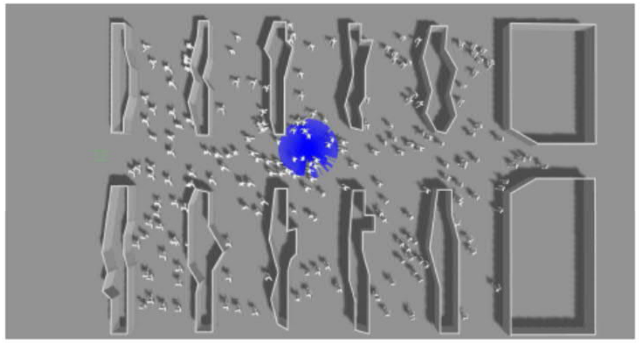


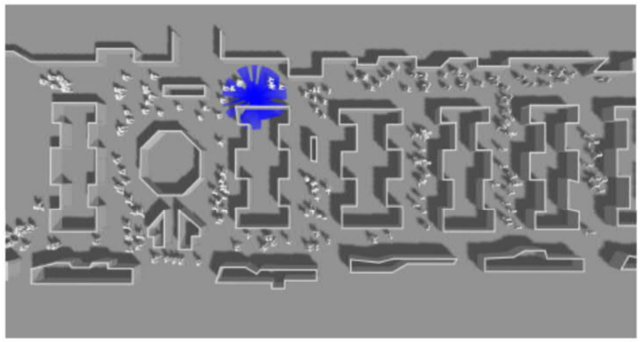
localization recovery policy

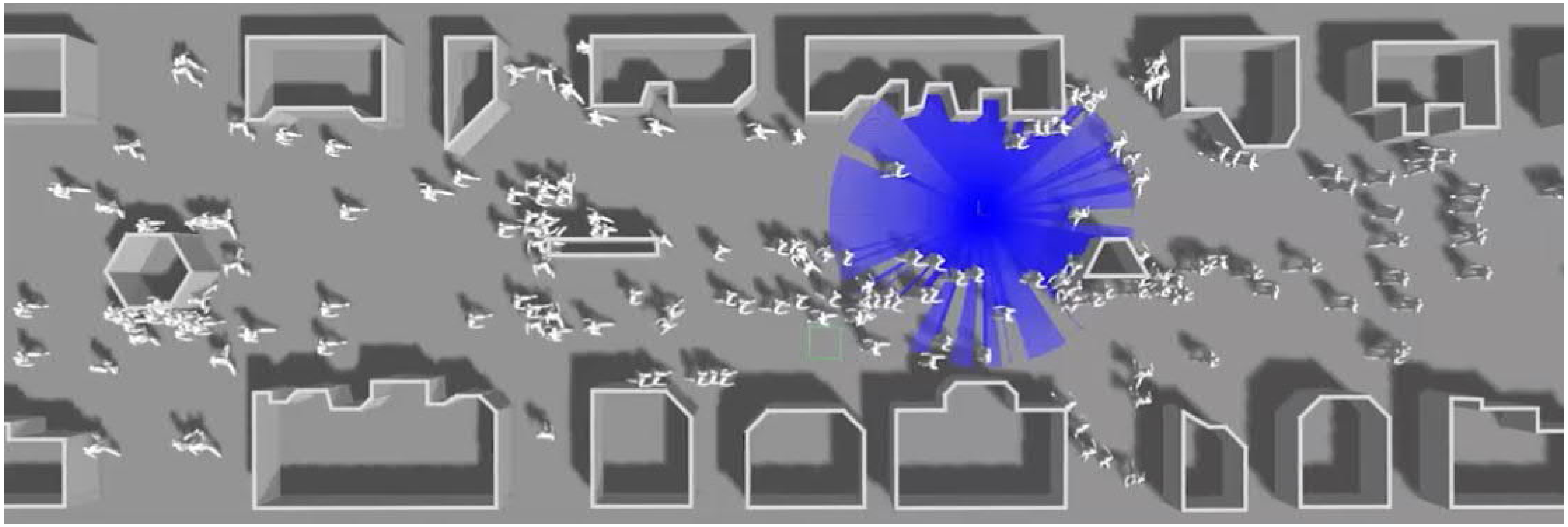


Simulation benchmarks for quantitative comparison

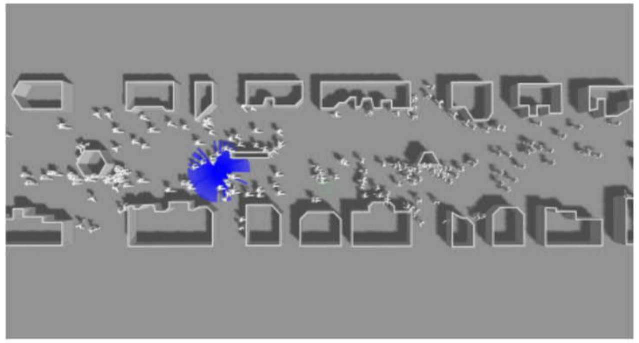

(a) corridor environment

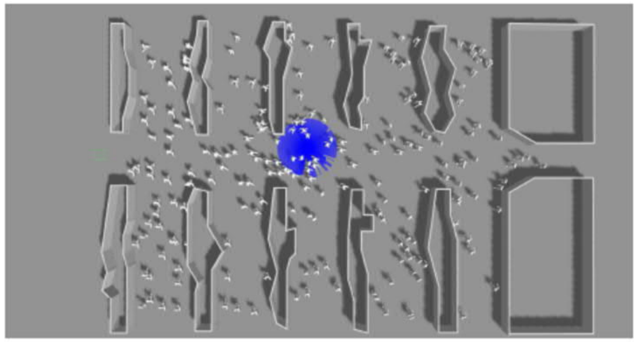

(b) supermarket environment

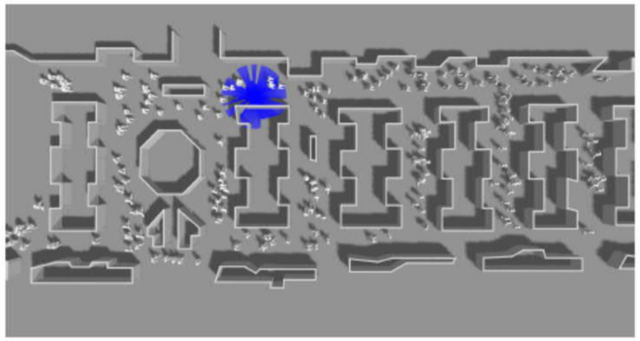

(c) airport environment



Simulation benchmarks for quantitative comparison

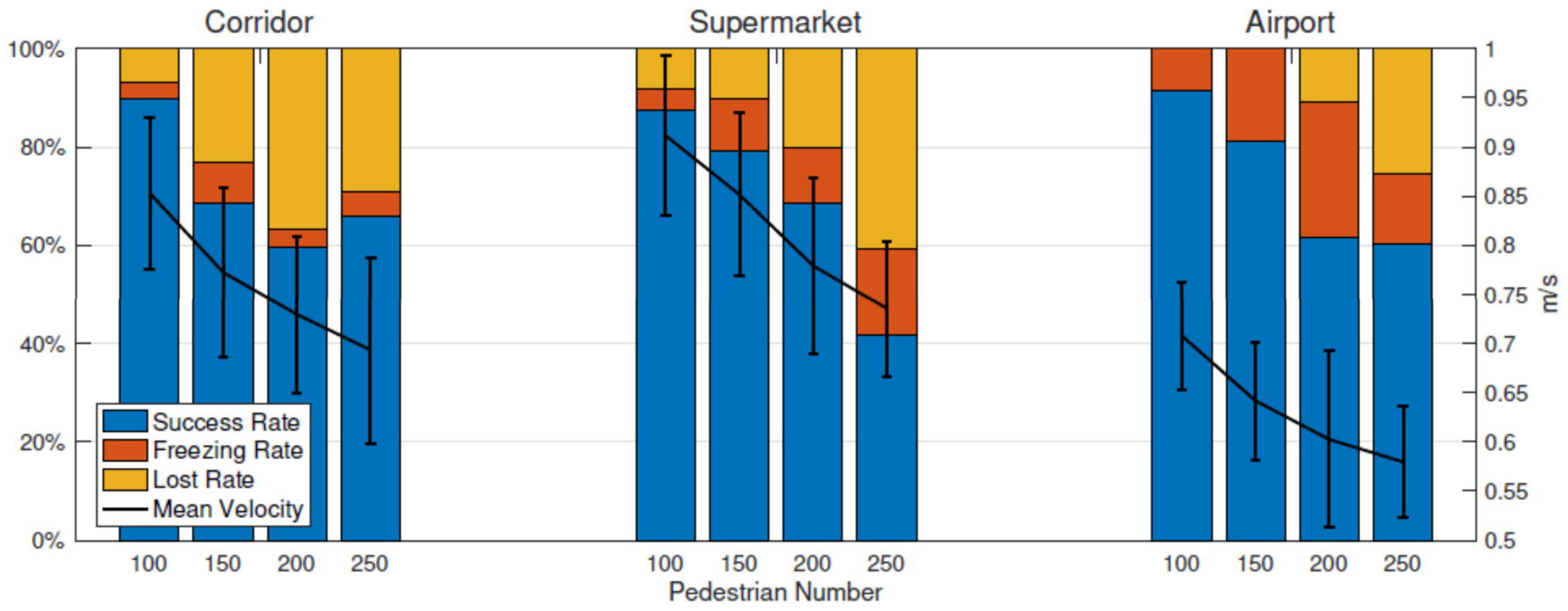

(a) corridor environment


(b) supermarket environment


(c) airport environment

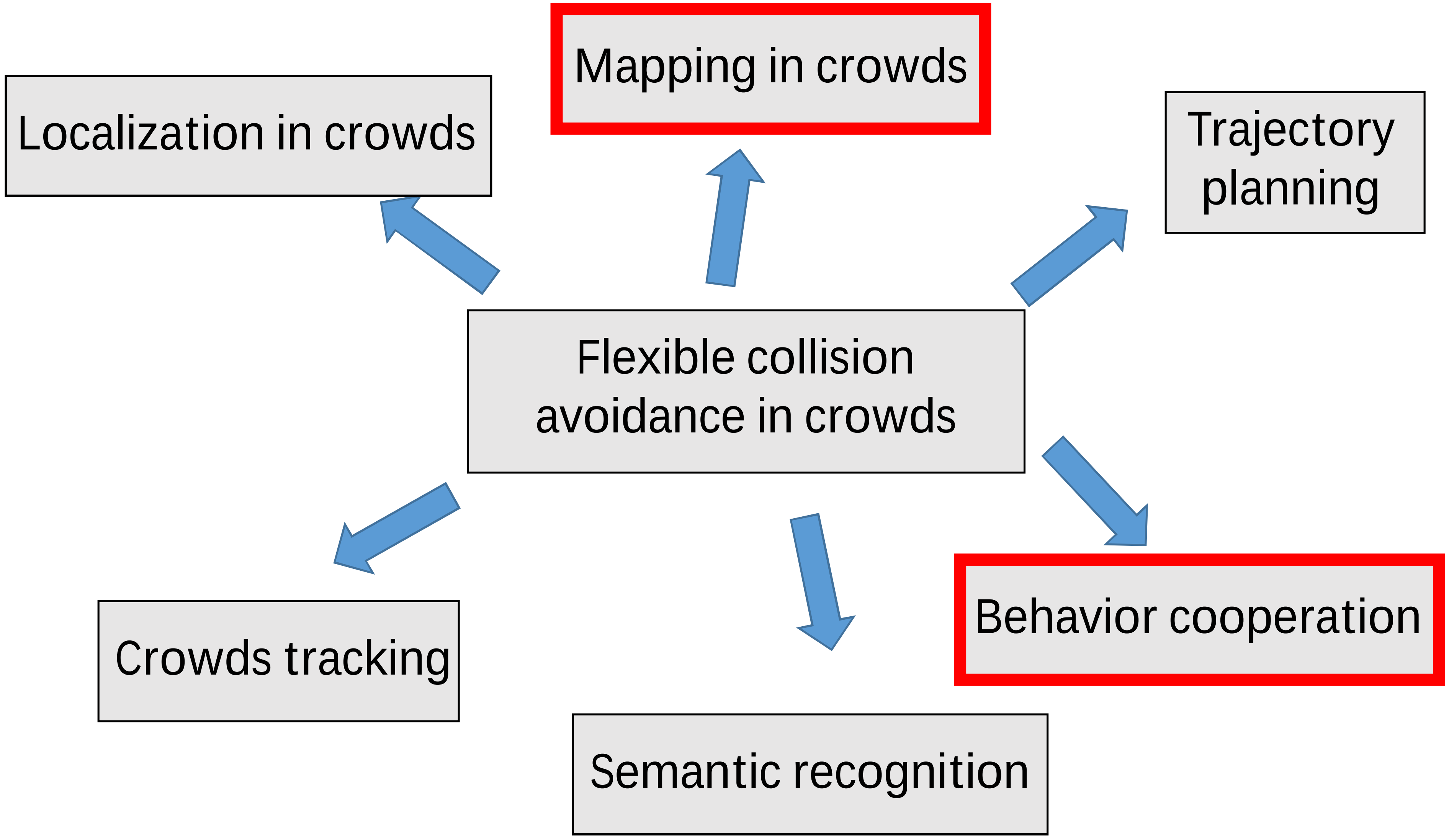
Methods	Corridor Scenario				SuperMarket Scenario				Airport Scenario			
	Lost	Frozen	Success	Velocity	Lost	Frozen	Success	Velocity	Lost	Frozen	Success	Velocity
baseline	32%	65%	3%	0.52 (0.074)	18%	82%	0%	0	16%	84%	0%	0
RL	63%	17%	20%	0.76 (0.065)	86%	10%	4%	**0.81 (0.095)**	71%	5%	24%	**0.68 (0.051)**
RL-CP	44%	6%	50%	**0.81 (0.070)**	28%	8%	64%	0.64 (0.038)	33/%	31%	36%	0.51 (0.049)
$(RL)^2$	36%	3%	**61%**	0.73 (0.079)	20%	12%	**68%**	0.78 (0.089)	10%	28%	**62%**	0.60 (0.090)

Performance under different density



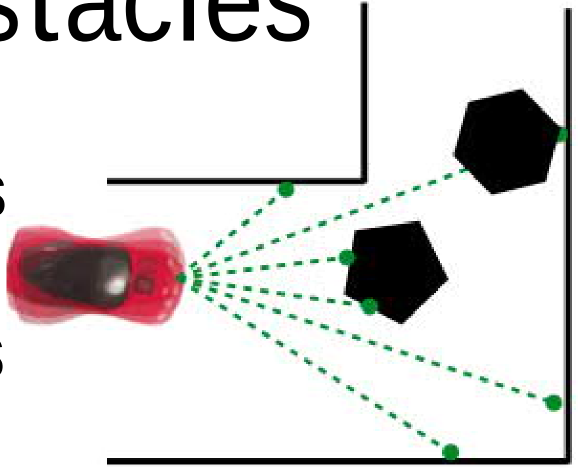
Real-world
localization recovery demo

Collision avoidance helps challenges in other components

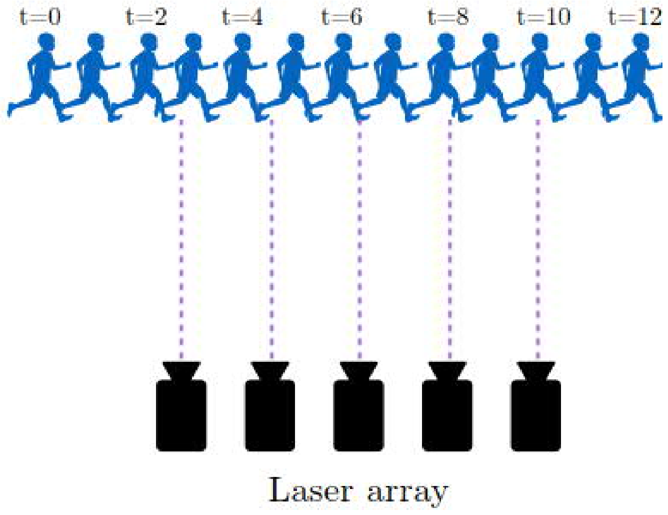


Curse of dynamic obstacles

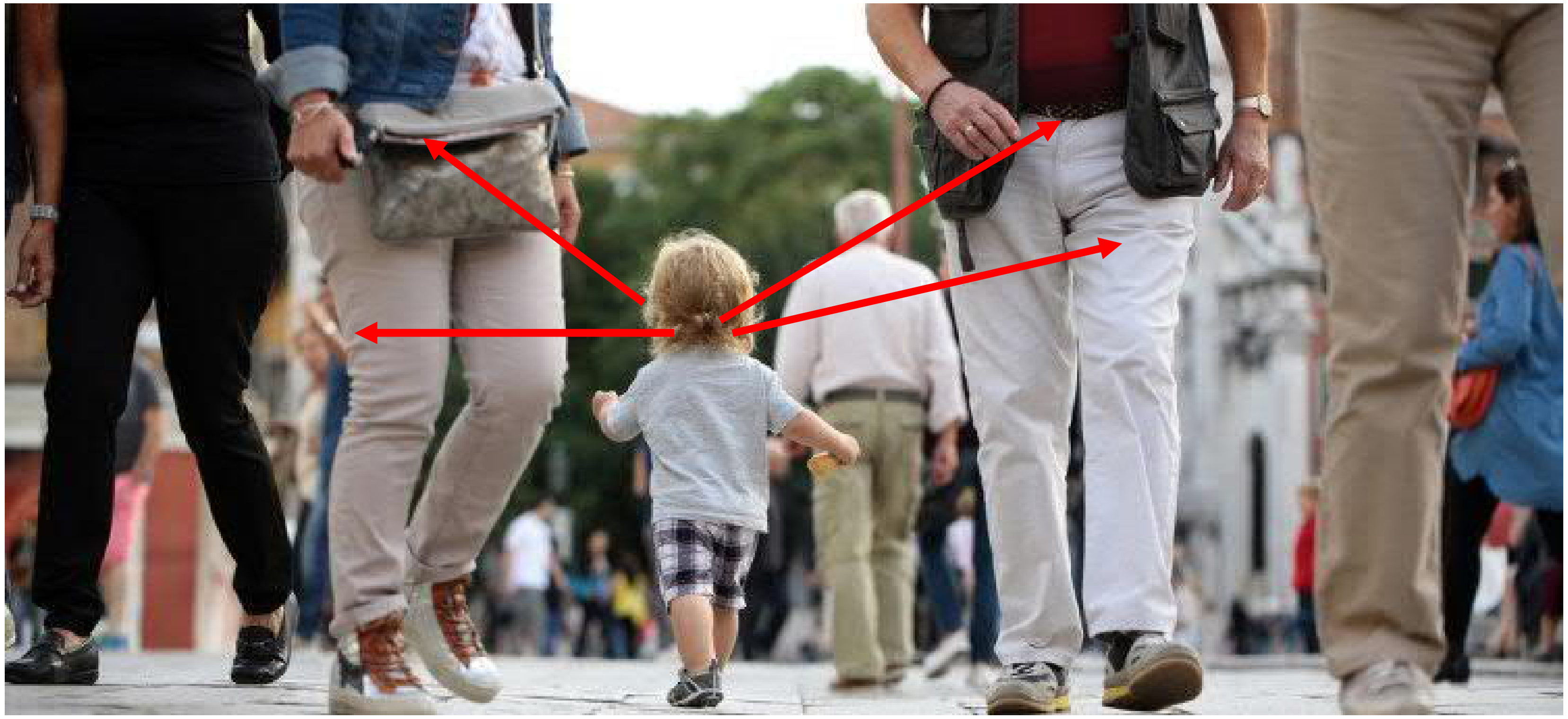
- Traditional mapping algorithm relies the matching features of the same static object at different time points



- In crowd scenario, mapping algorithms will be confused by dynamic obstacles:
  - Mistaking dynamics obstacles for static obstacles
  - Mistaking features of one dynamic obstacle for features of one static obstacle observed before



Mapping in crowds?



- Common solutions: detect moving obstacles and remove them from sensor measurements
- But such detection is difficult to be robust

Our solution

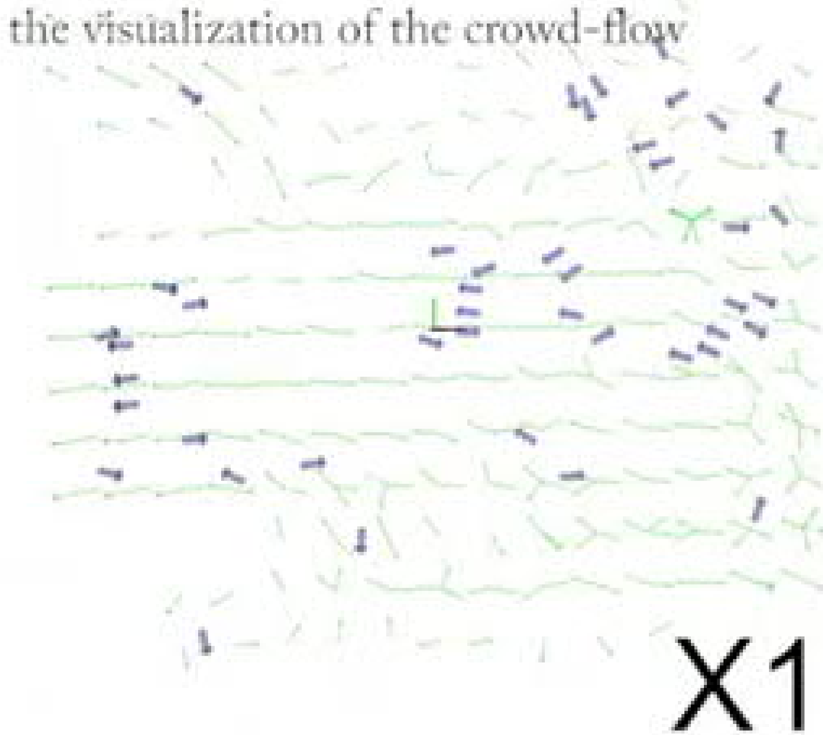
Rather than considering moving pedestrians as troubles for mapping

We would consider them as useful information source for mapping

How is this possible?

Intuition

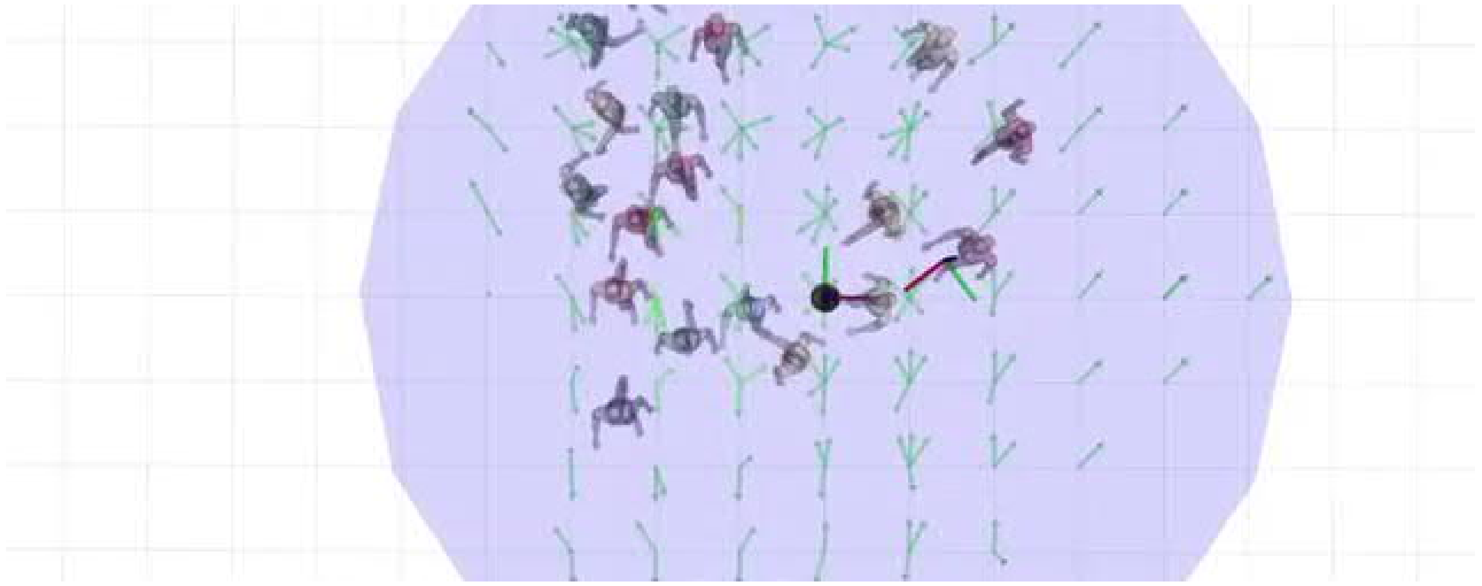
- Crowd vector field indicates the position of static obstacles



- Even more: information about social preference about moving directions

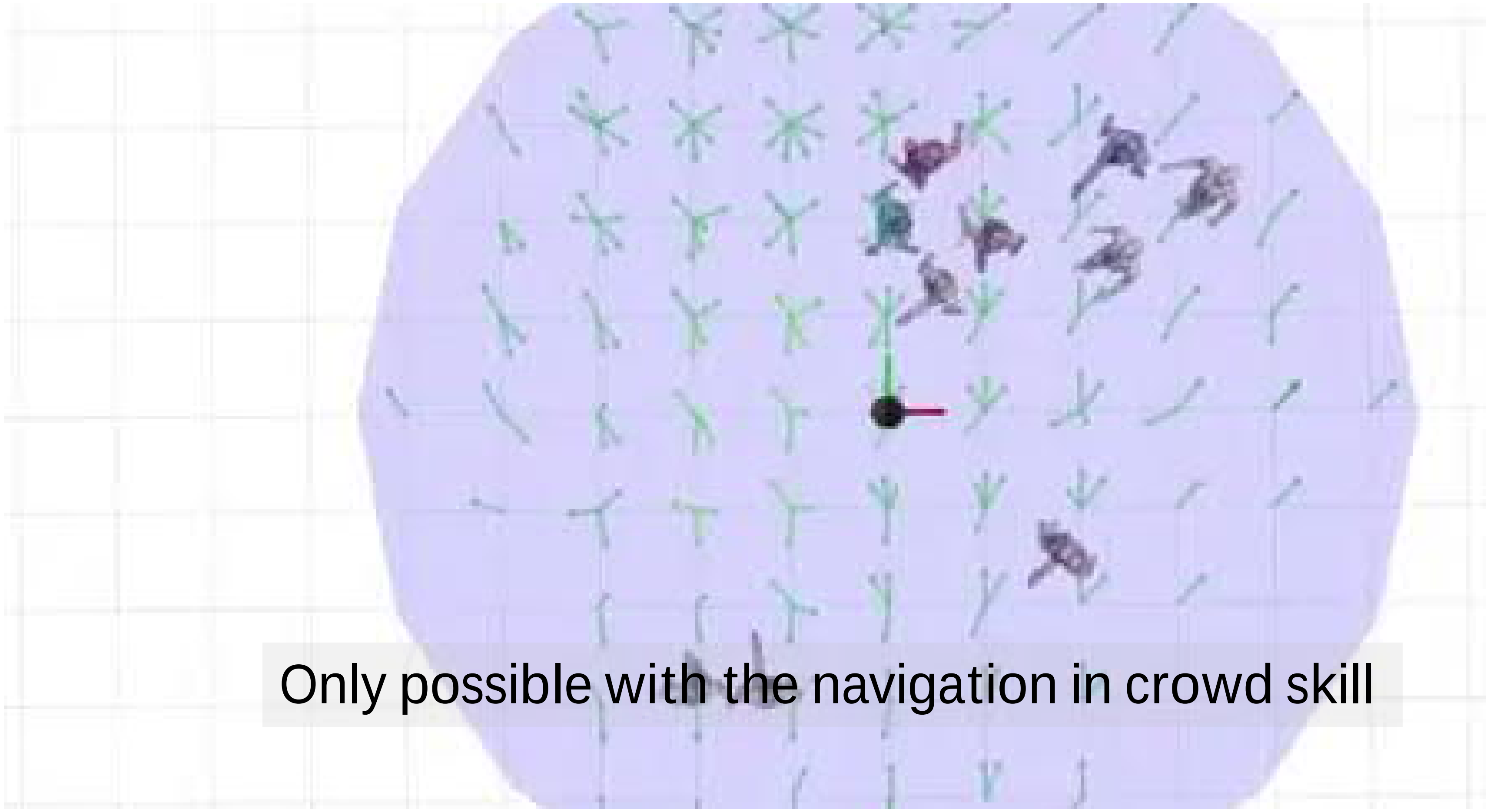
Mapping based on crowd flow

- The local crowd flow provides information about the moving resistance in all directions
- As the only sensor measurements in mapping
  - I.e., the robot has no knowledge about static obstacles and only use the crowd for mapping



Mapping based on crowd flow

- Then the sensor measurements are integrated using the occupancy-based mapping

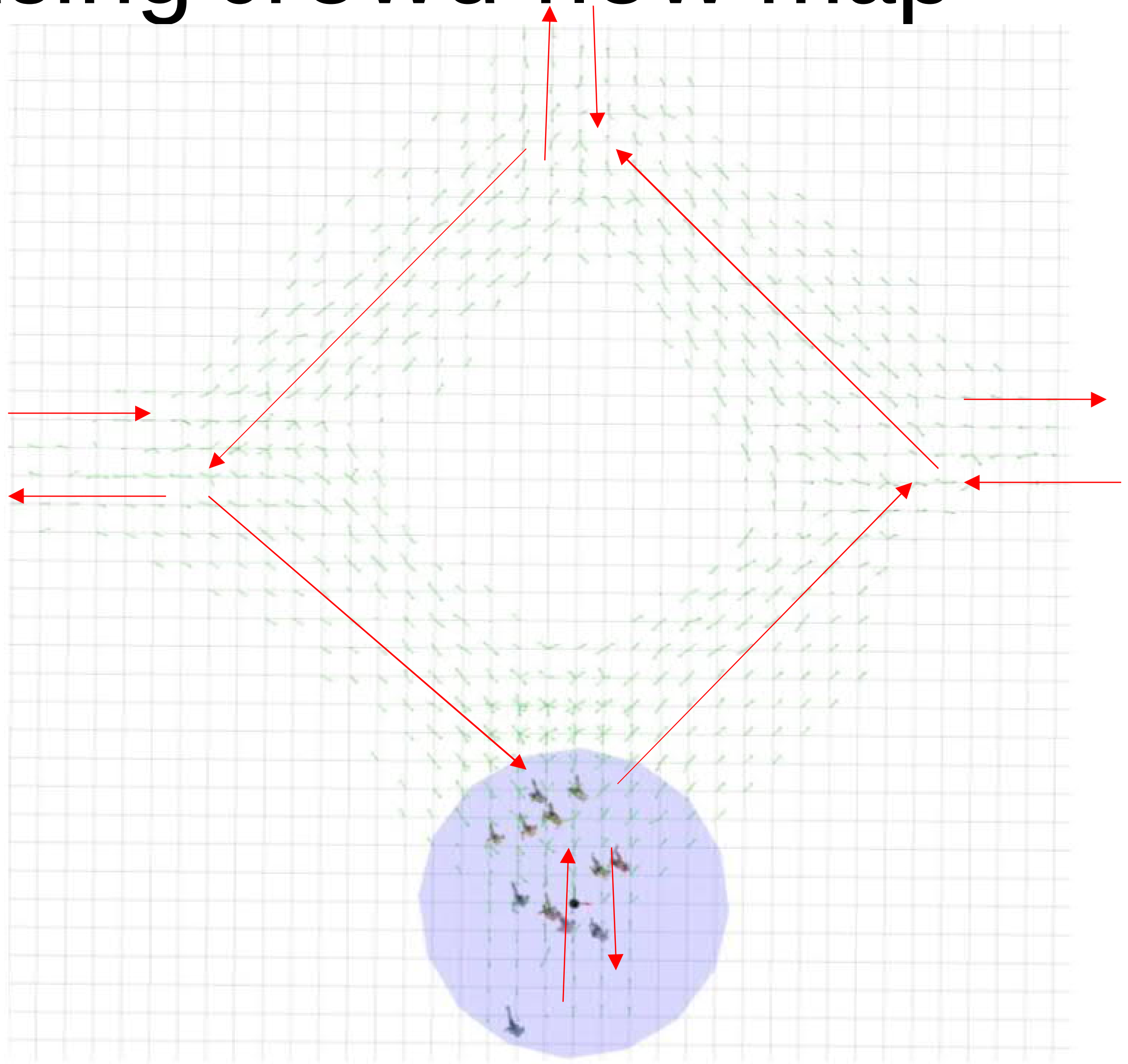


Only possible with the navigation in crowd skill

Navigation using crowd-flow map

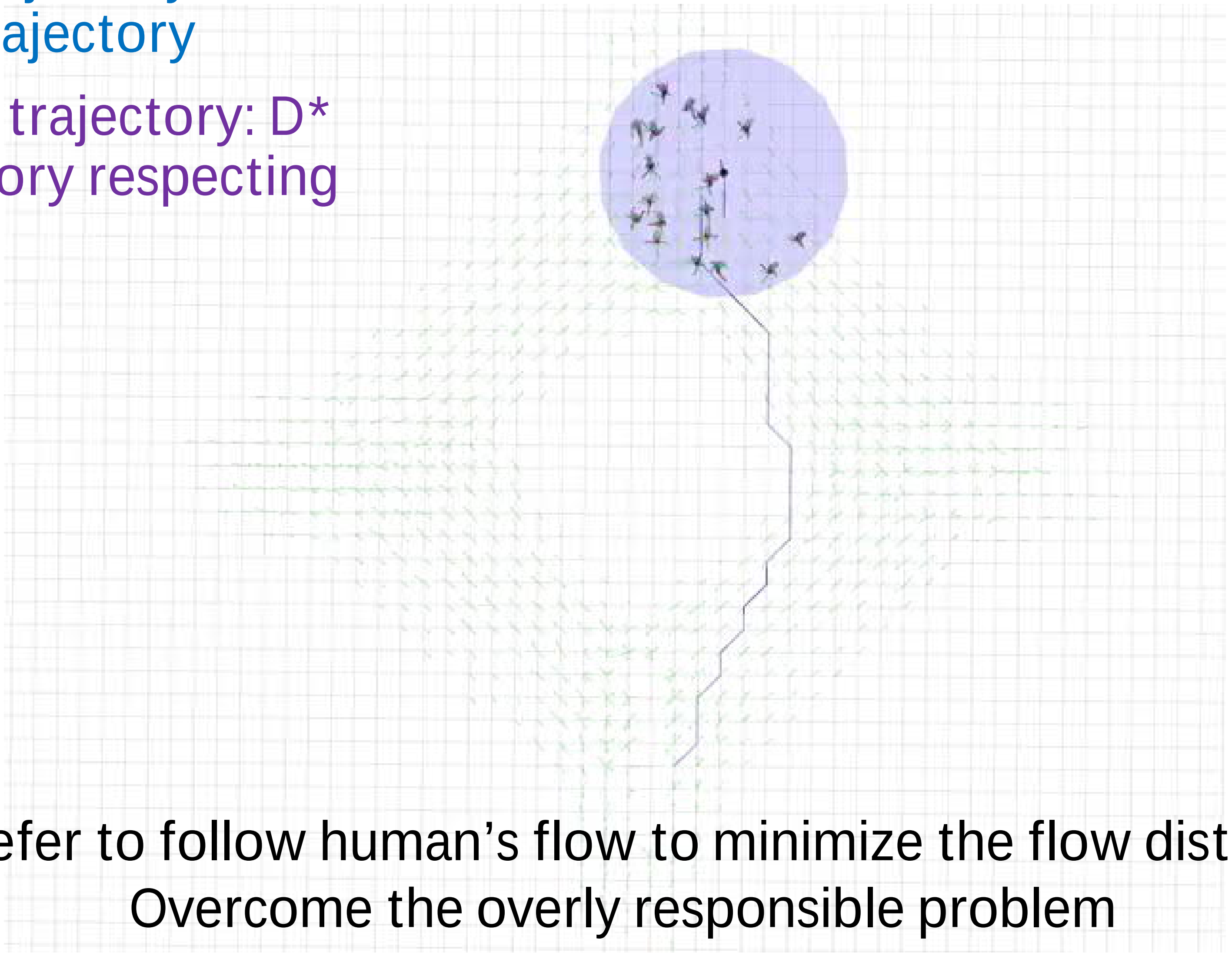
- Search algorithms (like D^*) can use the map to plan a social-compliable trajectory

- Then use RL-based collision avoidance to local adjust



Navigation using crowd-flow map

- Blue trajectory: min-path trajectory
- Purple trajectory: D* trajectory respecting flow



Prefer to follow human's flow to minimize the flow disturb
Overcome the overly responsible problem

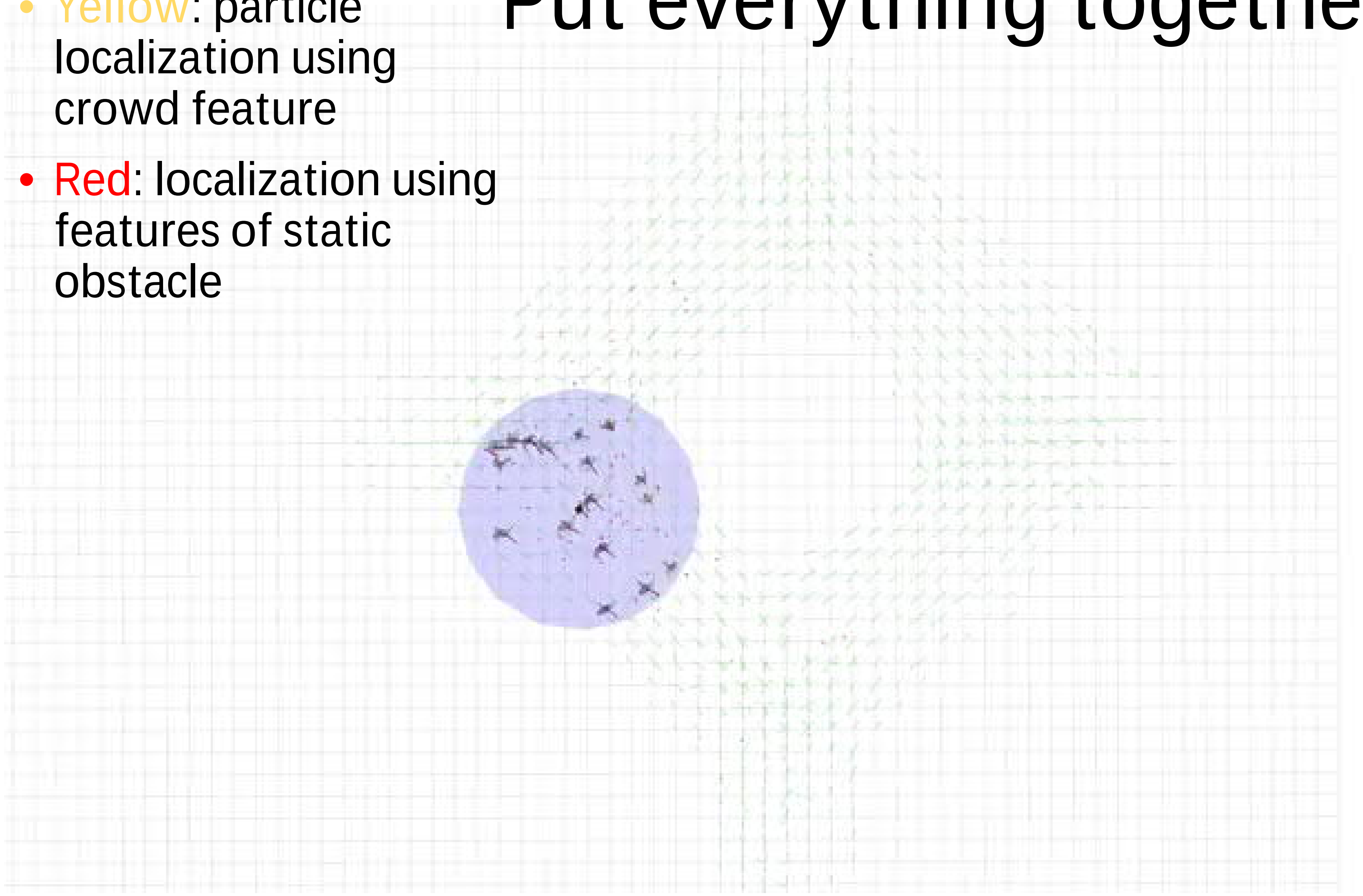
Can we do mapping and localization using crowd data only?
(i.e. replacing the SLAM using static objects?)

YES!

Using the crowd vector field as features to obtain rough localization

Put everything together

- Green: ground truth
- Yellow: particle localization using crowd feature
- Red: localization using features of static obstacle



Summary of part I

- Start with collision avoidance as the core task, we obtain much better navigation policy in dense crowds than state-of-the-arts
- Let learning focus on complex trade-off in collision avoidance, and use traditional control and robotics to handle other parts.
- Further solve a set of well-known robotics challenges
  - Localization recovery in crowd
  - Mapping in crowd
  - Localization in crowd
  - Human-friendly navigation

Deformable object manipulation

Motivation

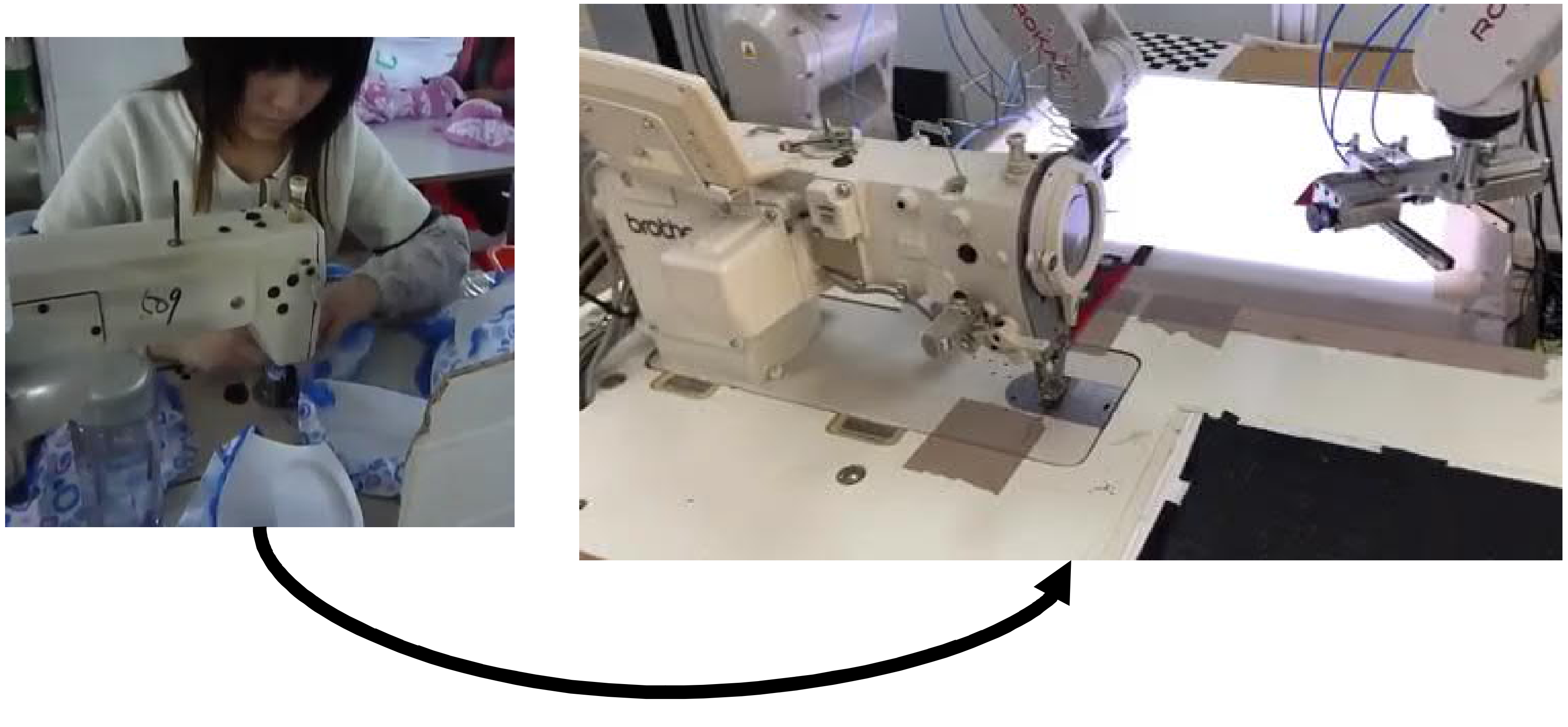
- Observed in a shoe factory in Anta
- Insole fabricated in 2 steps: assembly & sewing

Cloth piece assembly by human worker

Industrial sewing of assembled result

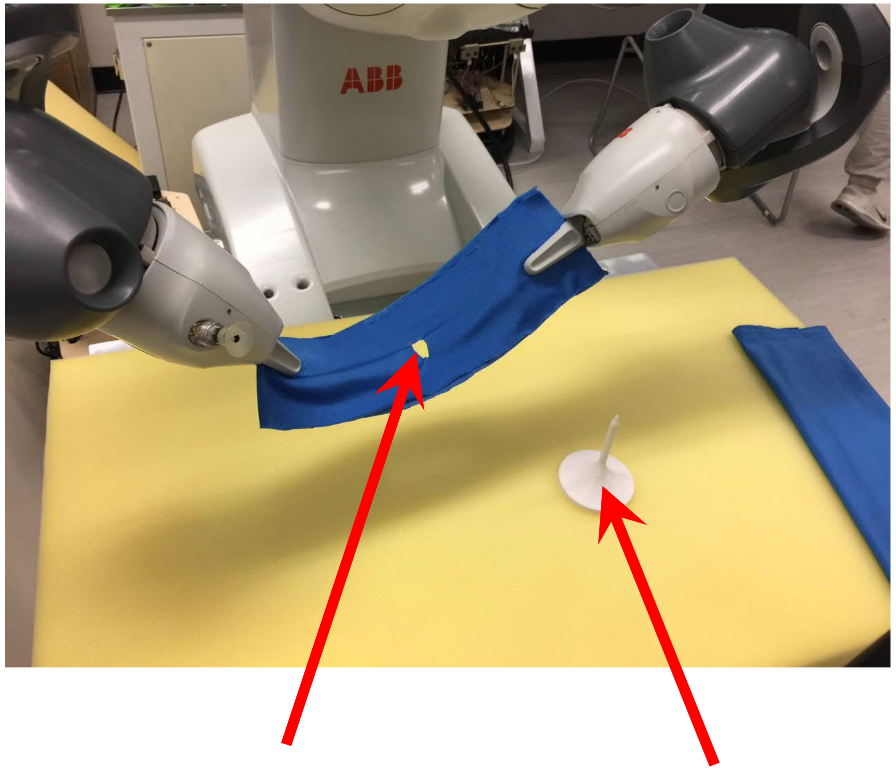
More challenging task: bra suturing

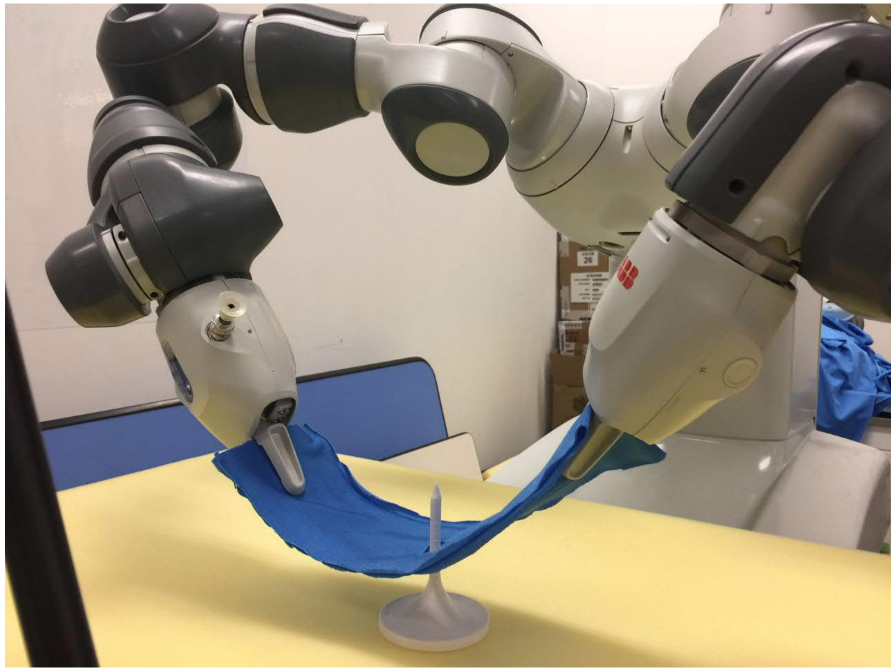
- 2.5D shape
- Geometric interface between soft/hard materials



Simplified problem

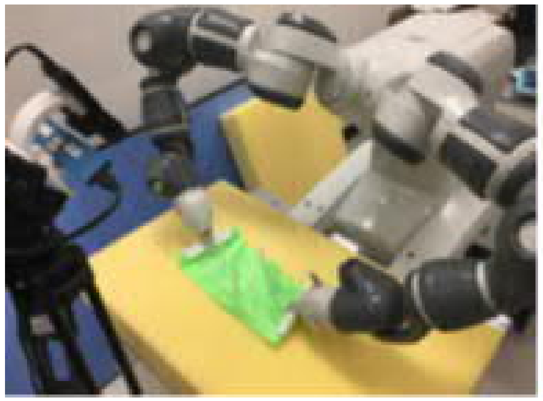
- Autonomous robotic cloth assembly

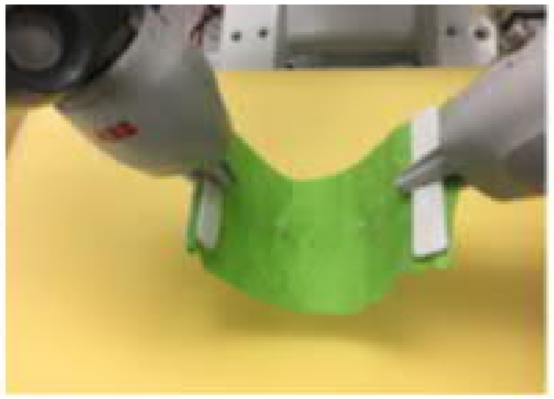


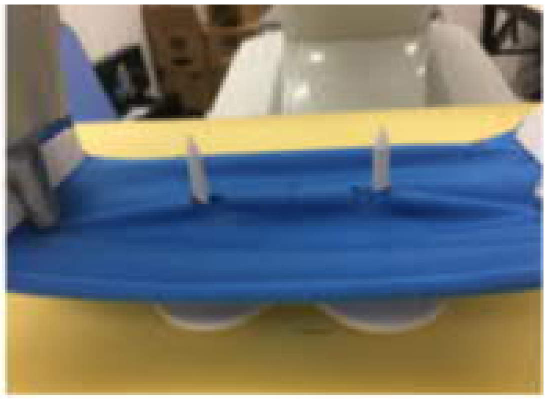


Still a non-trivial task

- The controller must generalize to
  - Cloth pieces with different types of materials
  - Cloth pieces with different numbers of holes
  - Holes with different tolerance


hard fabric, 4 holes


hard fabric, 2 holes


soft fabric, 2 holes, large tolerance where stretch is necessary

Learning based controller seems to be appropriate?

- Require complicated high-level policy

- Difficult to build mathematical model for the complicated physical interaction between cloth and environments

- Possible solutions:
  - Reinforcement learning
  - Learning from demonstration
  - Or combinations

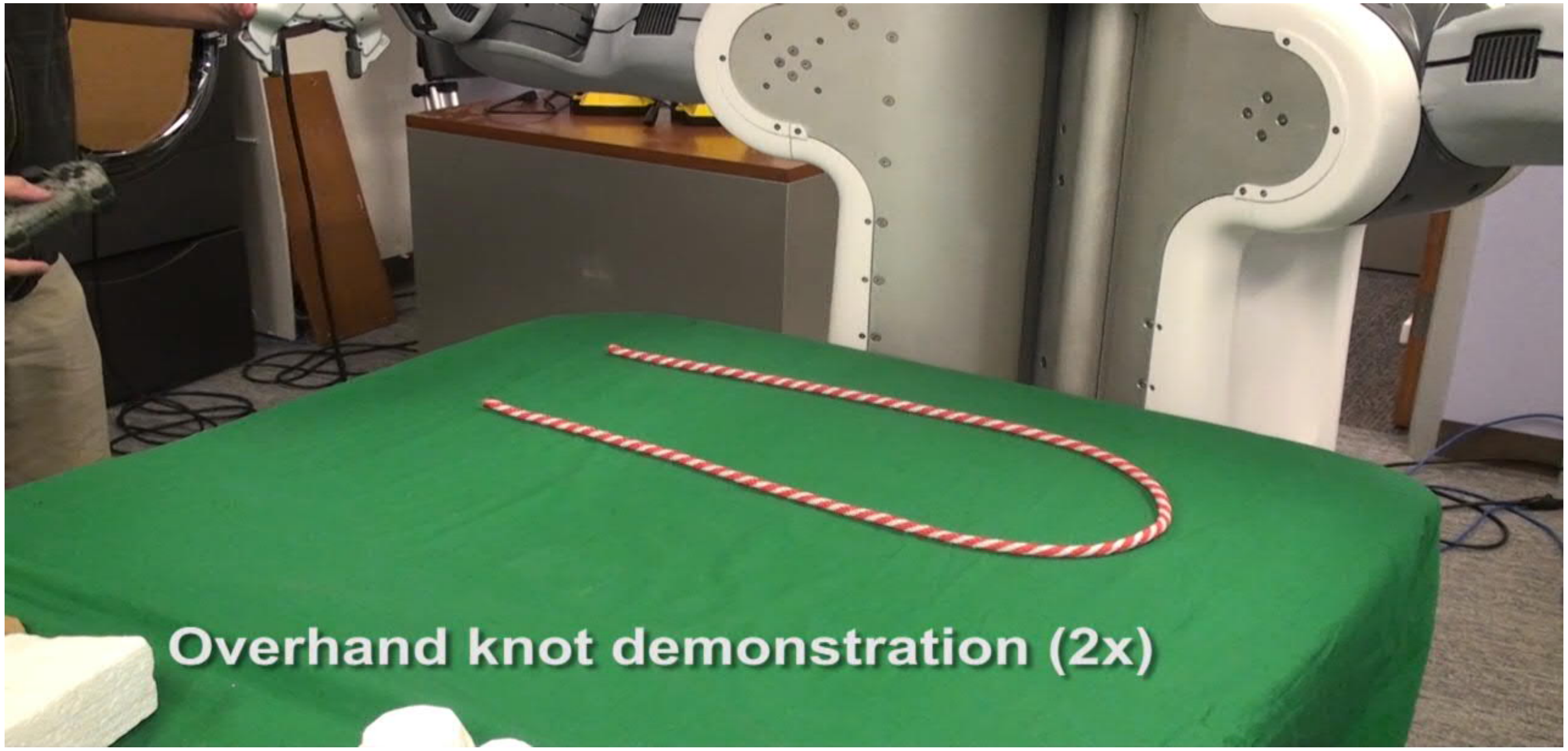
Learning from demonstration

- Operators provide demonstrations/examples about how to achieve a task (intuitively)
- Robots learn and generalize from examples
- Examples can be provided by teleoperation, motion capture, VR/AR, or kinesthetic teaching

demonstrations by kinesthetic teaching

automated knot tying

LfD knot tying



Combine LfD, RL and DL

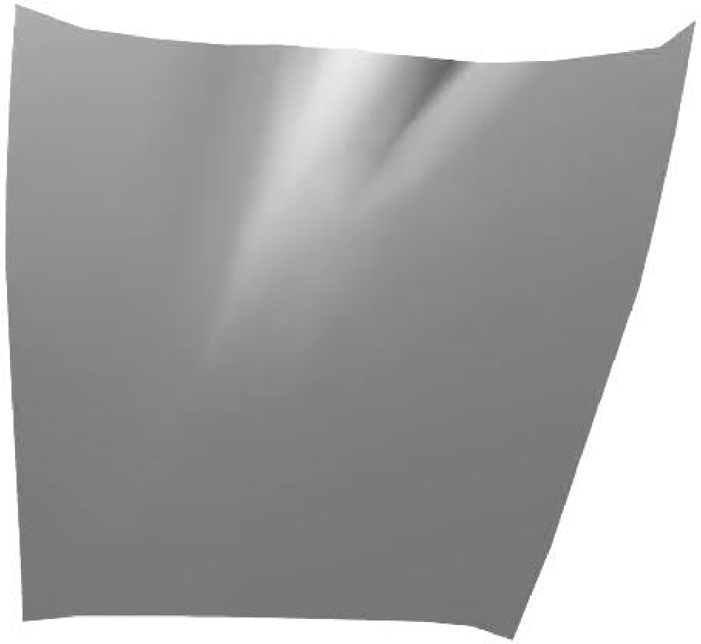
- Demonstration can include image, video, and tactile sensing measurement

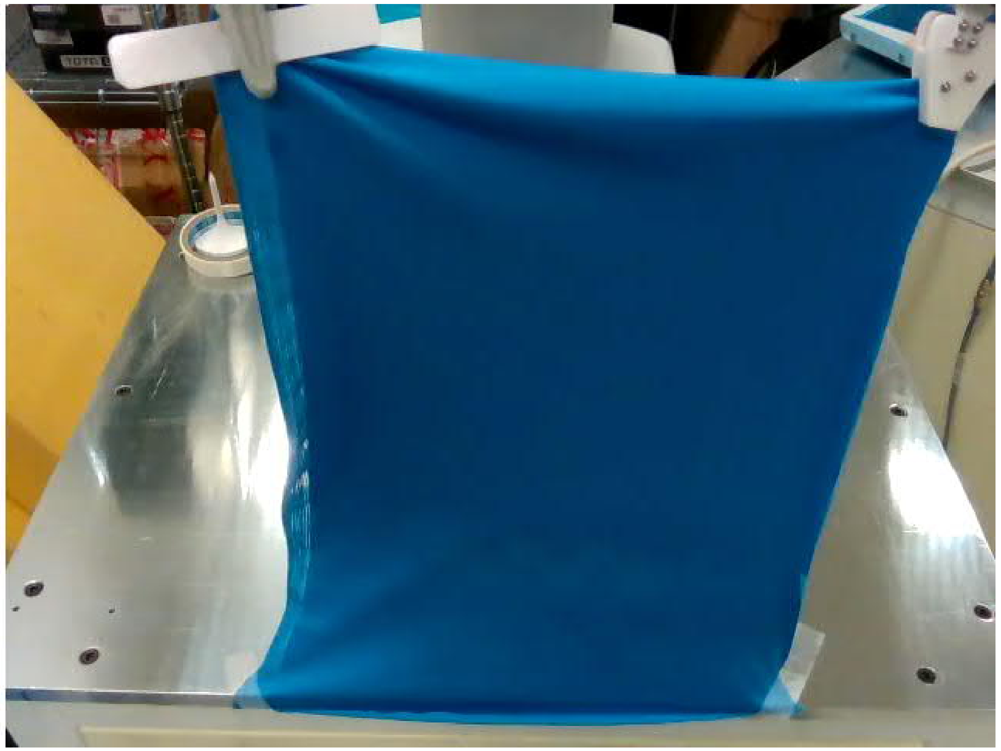
But fails on assembly task

Knot tying	Cloth assembly
Requires flexible multi-step policy	Requires both high accuracy and high flexibility
Knot physics is simpler, and thus possible to generate simulation consistent with real world	Complex interaction between cloth, fixture, and gripper, high-quality simulation is difficult

Simulator: Achilles' heel for RL

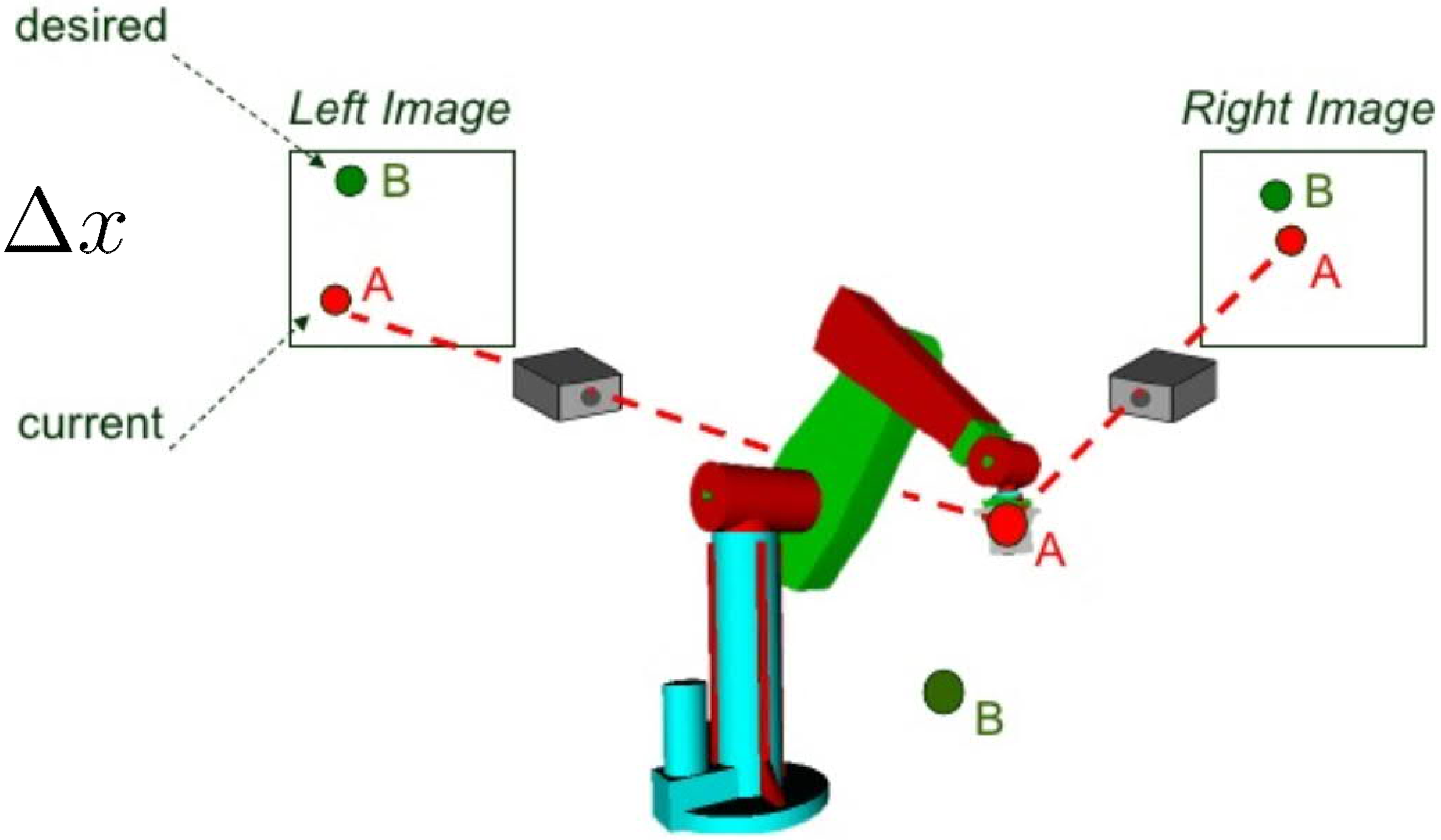
- Realistic simulation of deformable objects is extremely difficult (and also expensive)
- Even after careful optimization of physics parameters



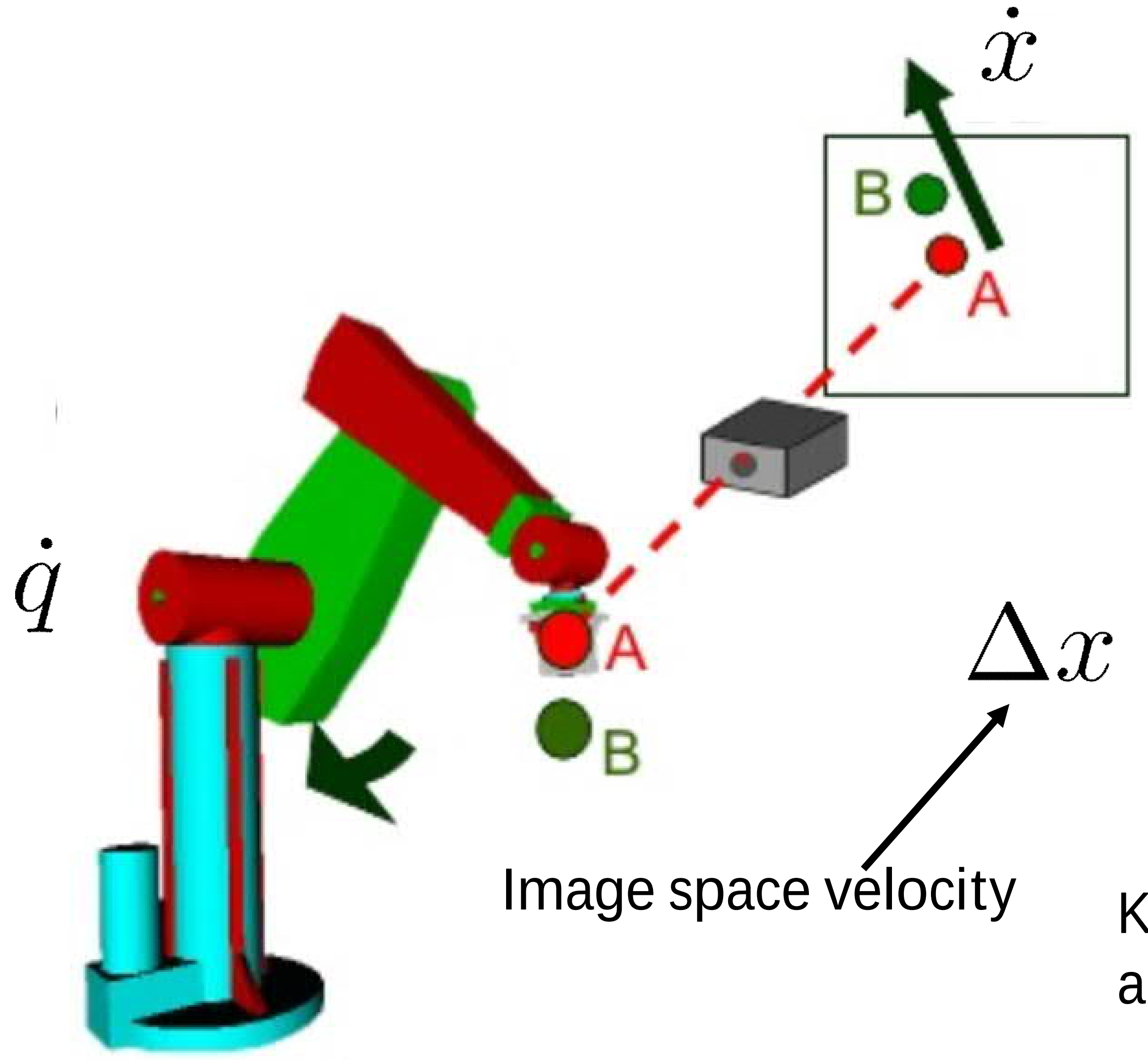


Recap traditional control: visual-servo control

Δx



Recap traditional control: visual-servo control



Joint space velocity

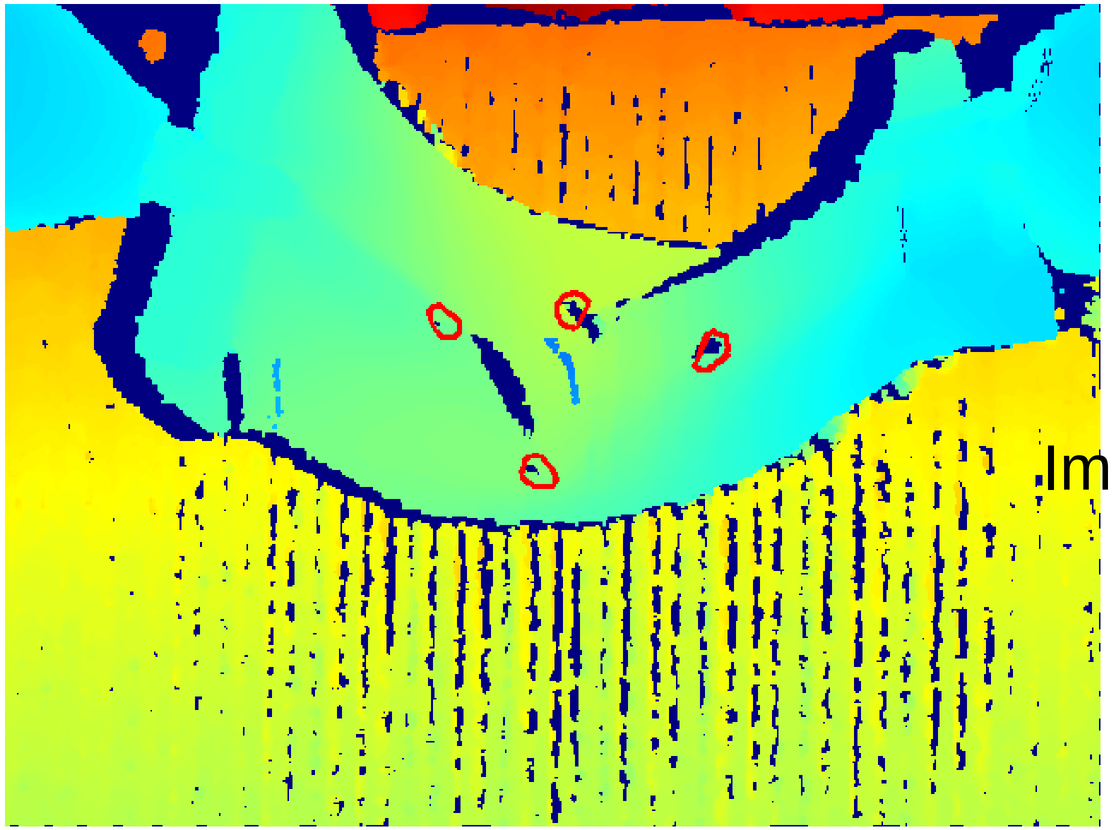
$$\Delta x = J \Delta q$$

Image space velocity

Kinematics Jacobian: analytically computed

Visual-servo control for deformable objects

- We can do similar thing for deformable object



Joint space velocity

$$\Delta x = J \Delta q$$

Image space velocity

How the joint influences the cloth status:
unknown,
time-varying,
material-varying

Visual-servo control for deformable objects

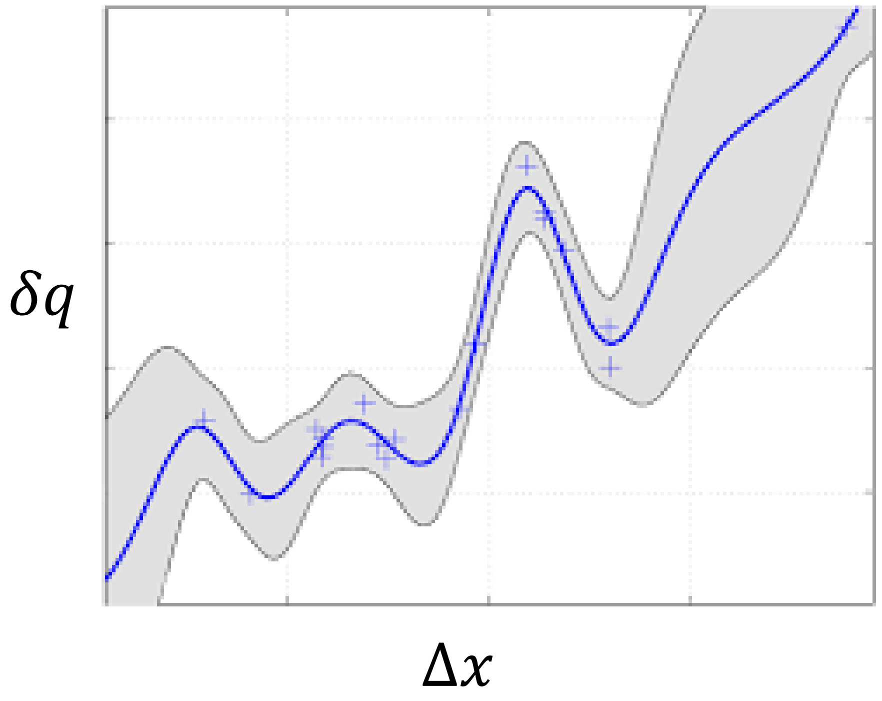
- Traditional methods use linear functions to model the J matrix

- We are using nonlinear approximators, including
  - Gaussian process
  - Neural network

- Dynamically update the J function during the manipulation to best describe the relation between Δx and Δq in the near future

Nonlinear visual-servo controller using Gaussian processes

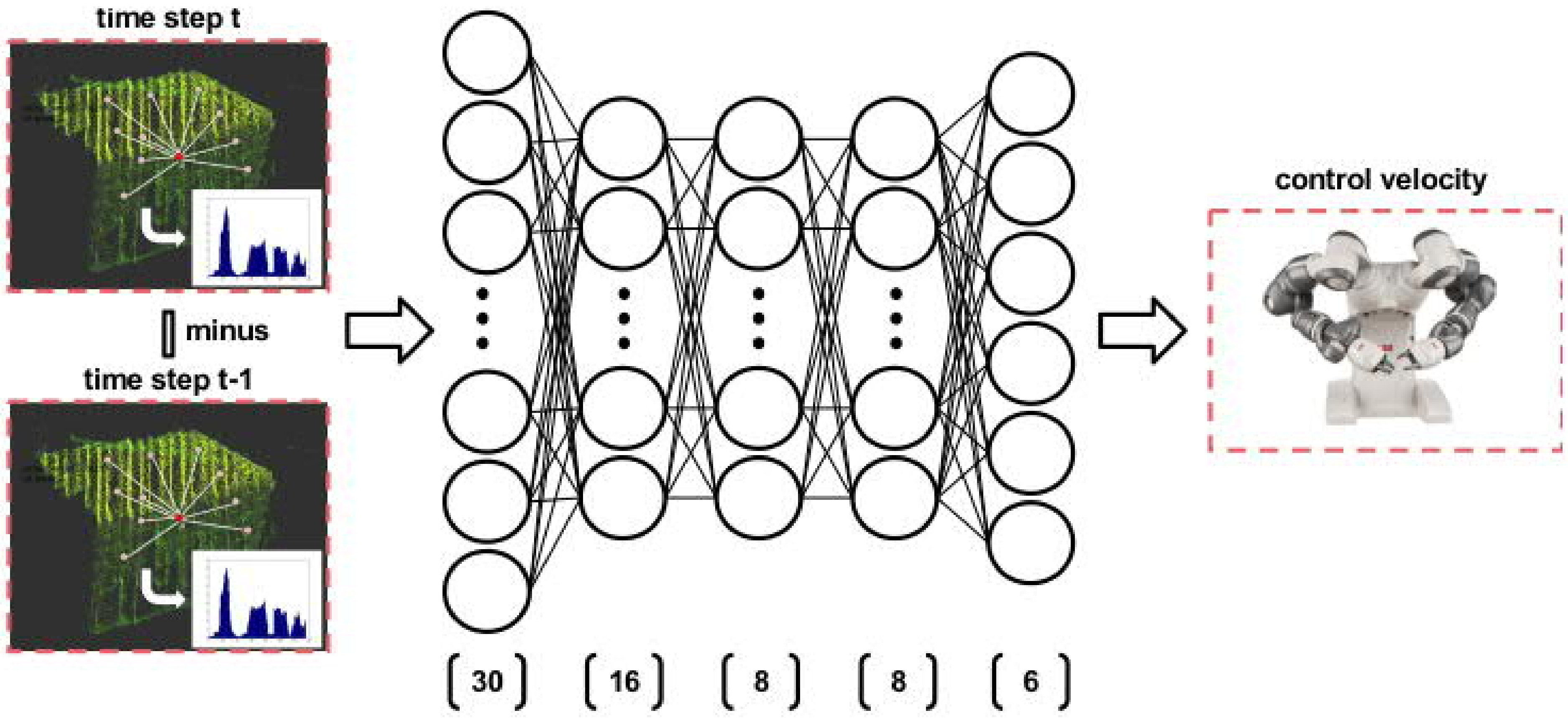
- $J \sim GP(m(\delta x), k(\delta x, \delta x'))$
- Initialize GP using random perturbation of the cloth piece around the initial configuration
- Use the data collected by the robot during the manipulation process to gradually update GP



$$\mu_t(\delta x) = m(\delta x) + k^T(\delta X_t, \delta x)$$
$$[K(\delta X_t, \delta X_t) + \sigma_n^2 I]^{-1}(\delta q_t - m(\delta X_t))$$

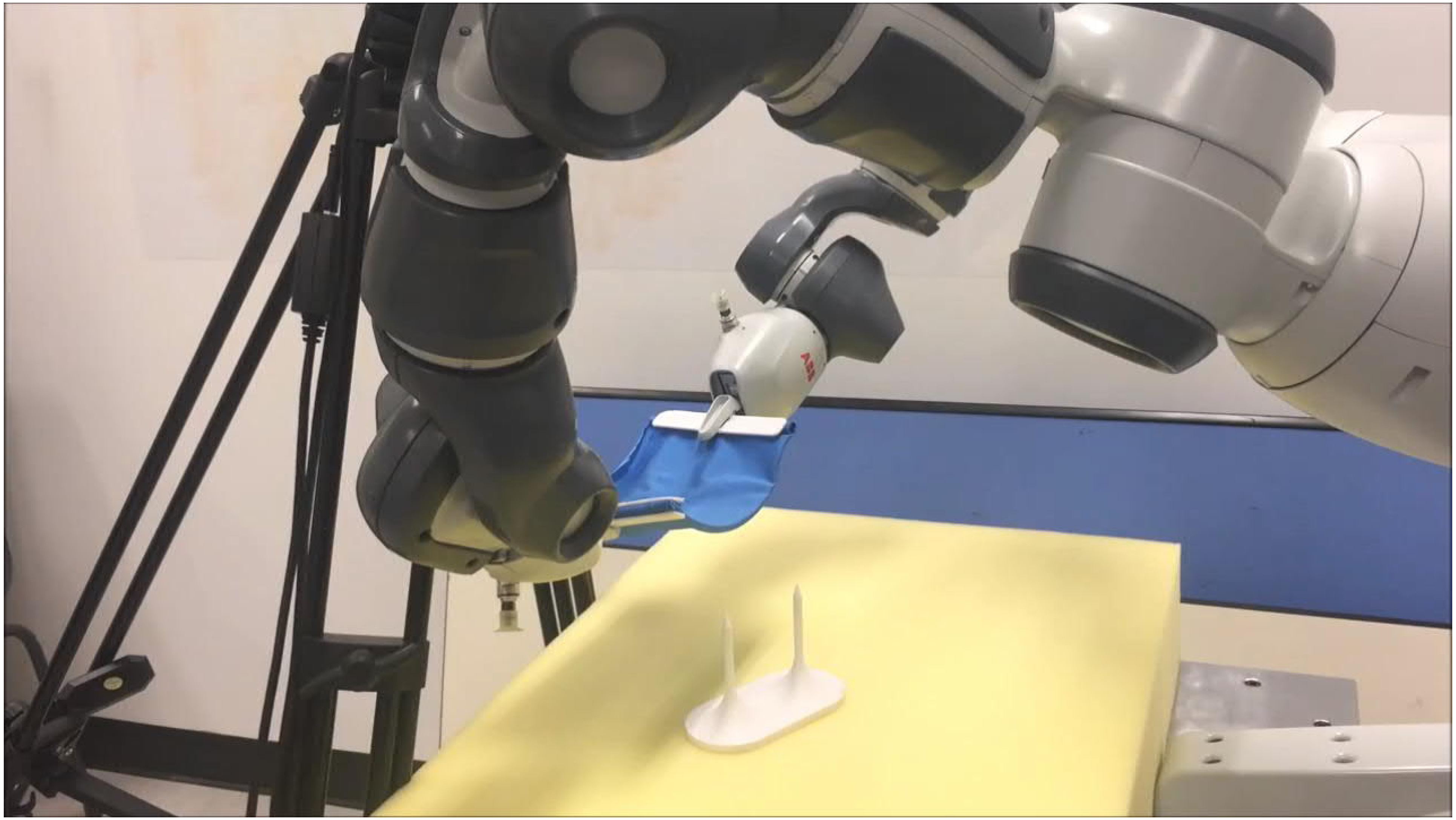
$$\sigma_t^2(\delta x) = k(\delta x, \delta x) - k^T(\delta X_t, \delta x)$$
$$[K(\delta X_t, \delta X_t) + \sigma_n^2 I]^{-1} k(\delta X_t, \delta x)$$

More general controller using deep neural network

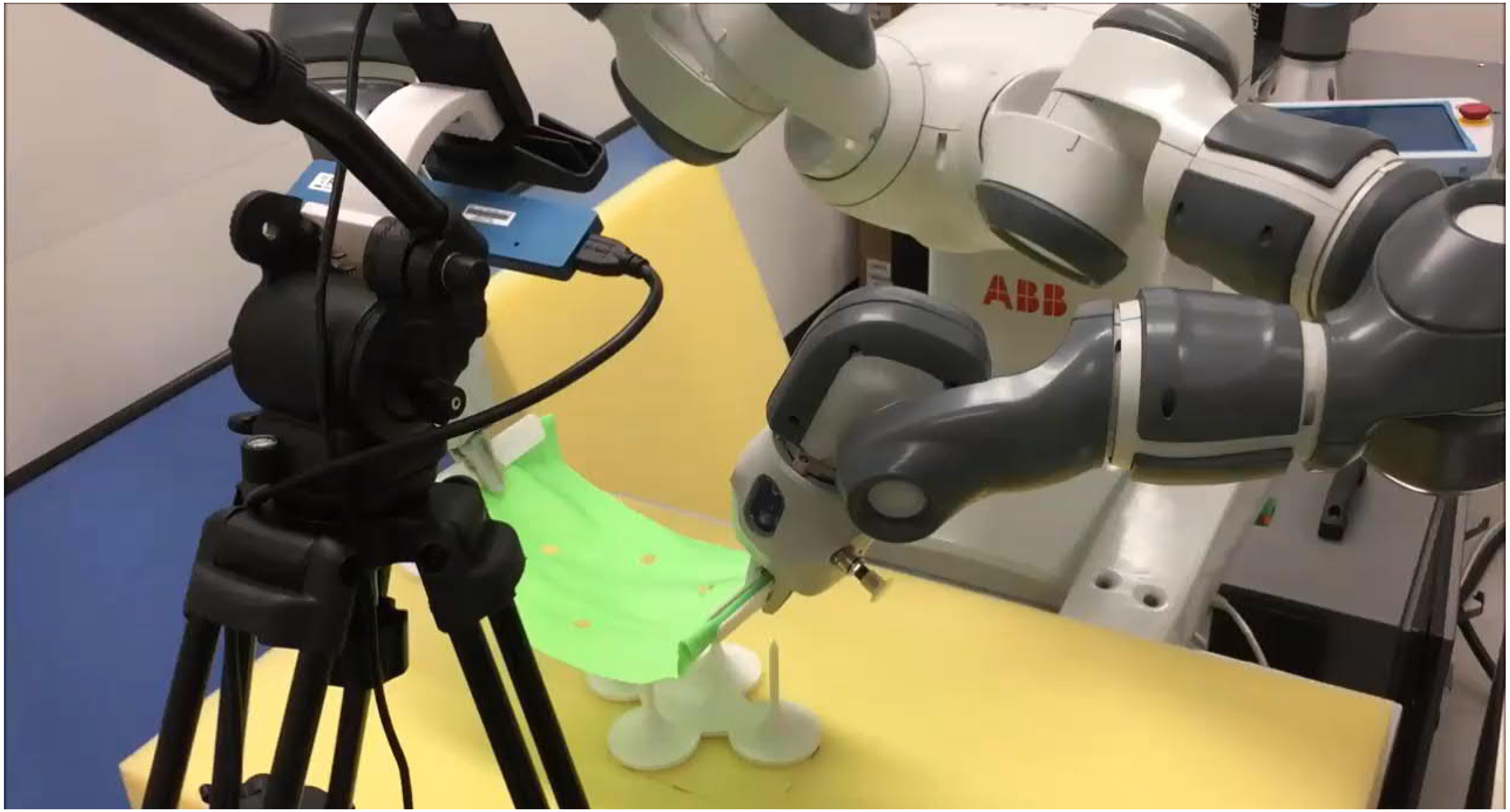


Besides fixture holes, we can also use raw image as controller input

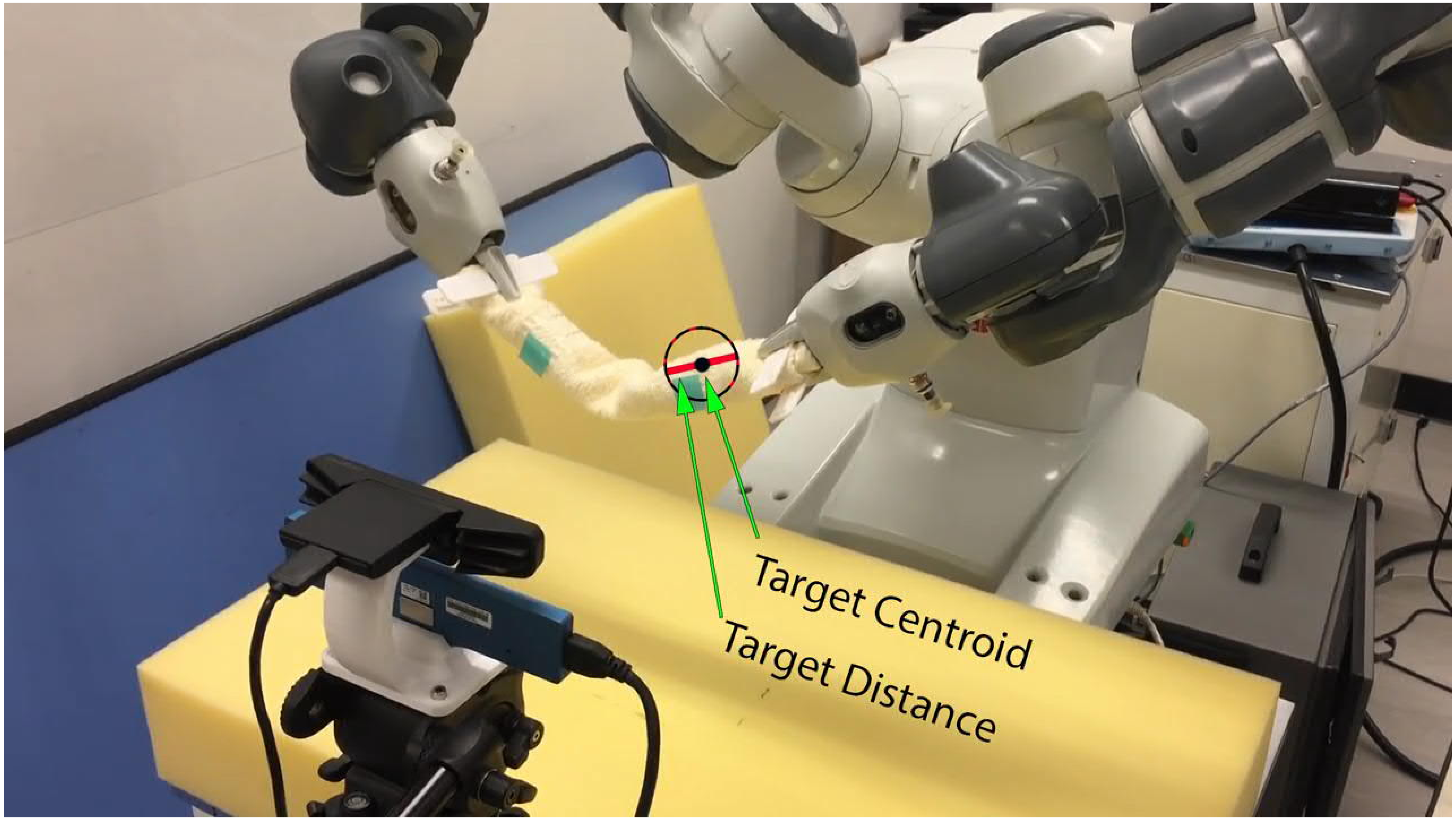
Peg-in hole assembly for soft fabric:
2 holes/2 pins



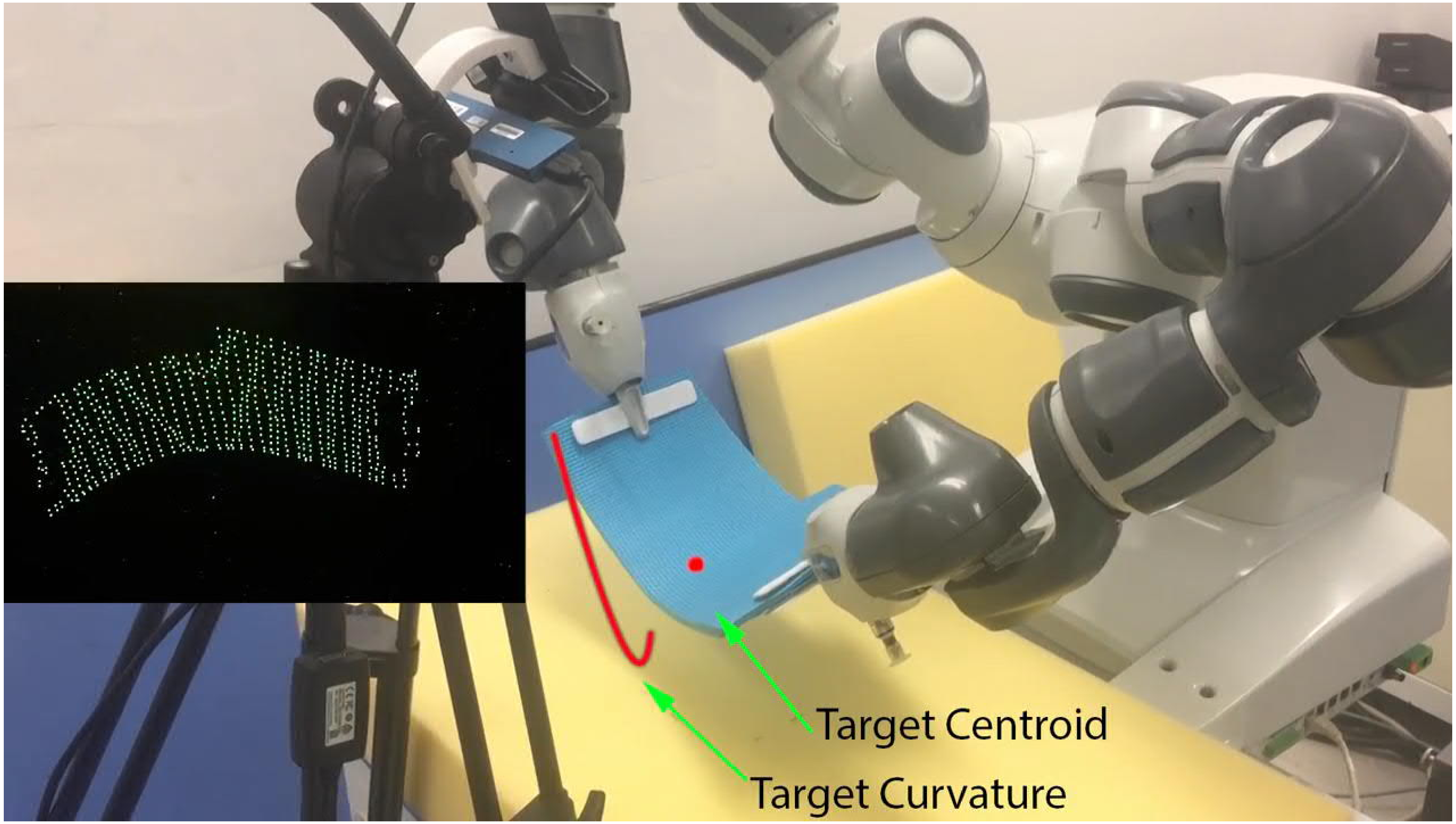
Peg-in hole assembly for hard fabric:
4 holes/4 pins



Rolled towel bending:
point feedback features

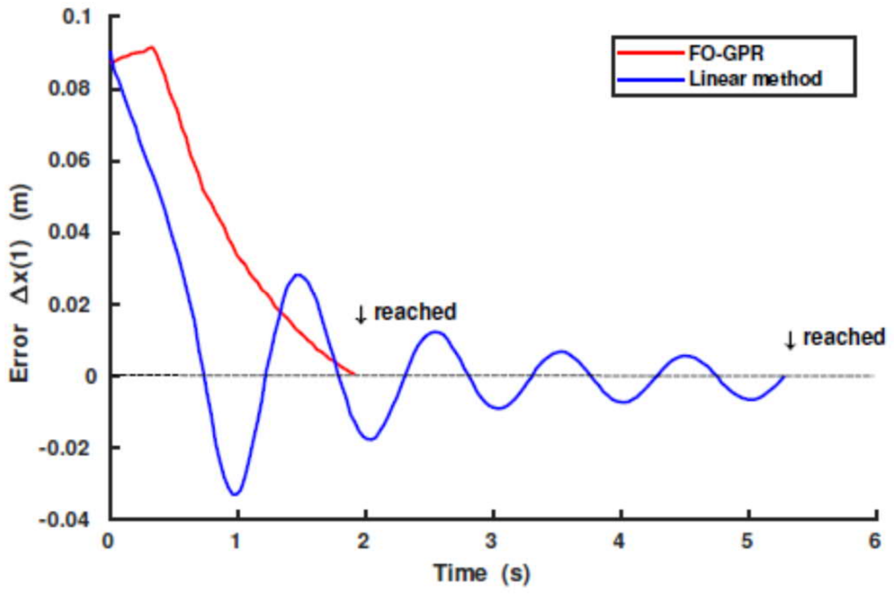


Plastic sheet bending:
curve feedback features

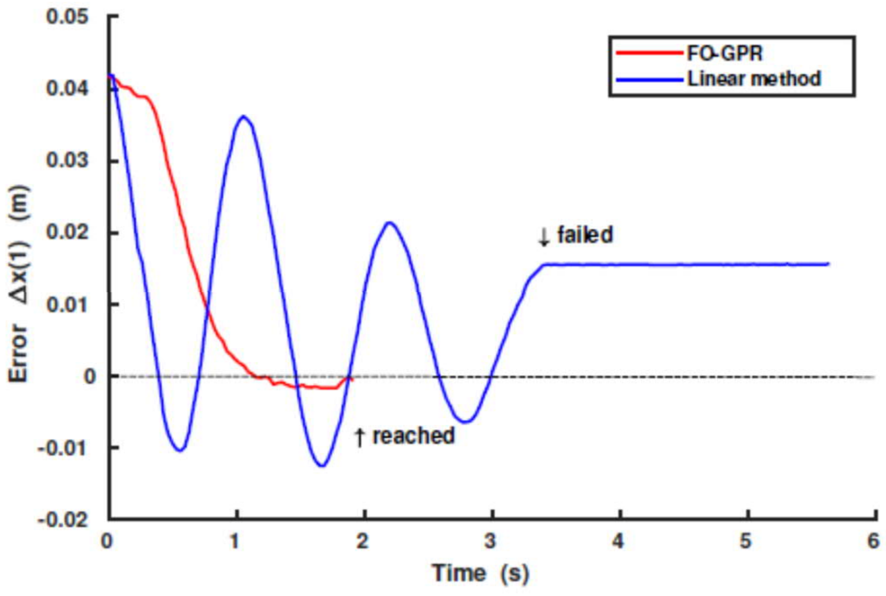


Better performance than linear controller

- The linear controller is also updated online
- GP controller converges faster and more robust
- Success rate >95%; >80% with human perturbation

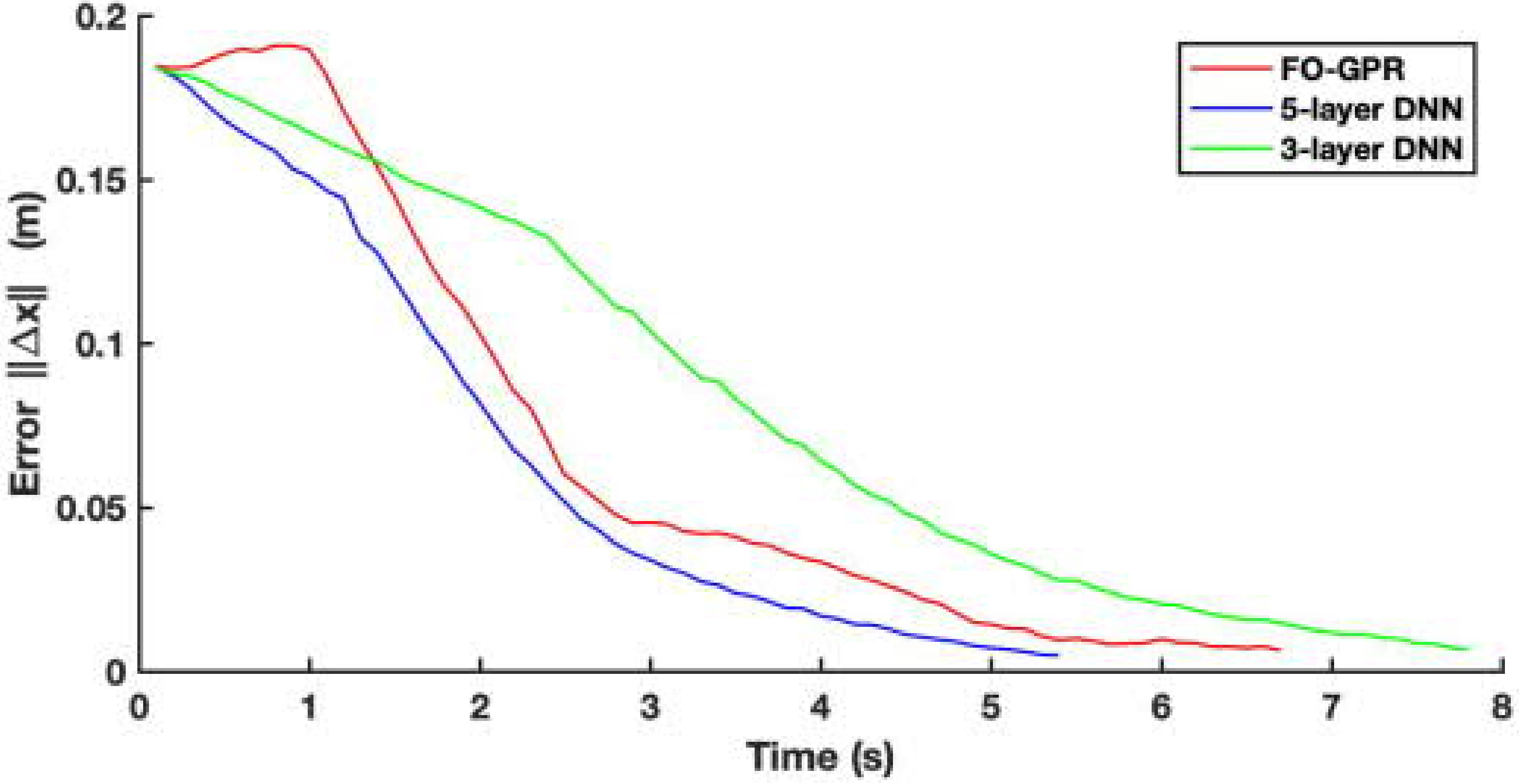


on rolled towel task



on peg-in-hole task

Neural network controller further improves



Summary of part II

- Learning is appropriate for modeling the complex interaction physics between deformable object and the environment

- But the overall pipeline is still more suitable for traditional visual-servo feedback controller due to the required high accuracy

- Pure reinforcement learning does not fit in here

Conclusion

- Combine control and learning – always an art

1. Inspect your problem/task carefully
2. Let control or learning handle the subtasks most appropriate for themselves
3. It usually is important to figure out which is the core subtask and start from there

Thank you