Pursuit-Evasion Strategies for Multi-Agent Autonomous Robots

Journées de rencontre de la Graduate Initiative EIF (02/07/2026)
Ahmad HABLY (Lyon 1 Université, LAGEPP) and Oussama ERROUJI

Ce travail a bénéficié d'une aide de l'État gérée par l'Agence Nationale de la Recherche au titre de France 2030 portant la référence ANR-21-SFRI-0001

Motivation

R&D expenses on control/ navigation of mobile robots are increasing

Mobile robots are more and more present in our life and they become more and more intelligent

Farming activities

Transportation/industrial activites

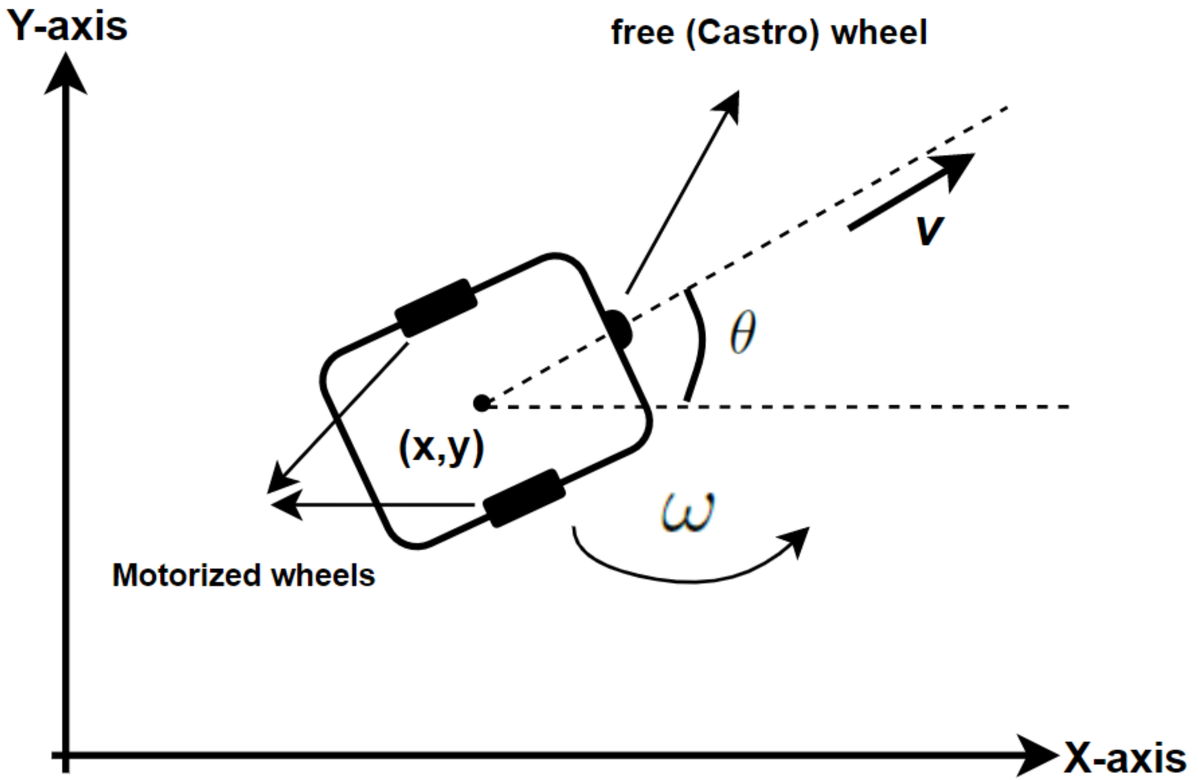
Surveillance and security

Autonomous racing

Robot Modeling

$$\begin{cases} \dot{x}(t) &= v(t)\cos\theta(t) \\ \dot{y}(t) &= v(t)\sin\theta(t) \\ \dot{\theta}(t) &= \omega(t) \end{cases}$$

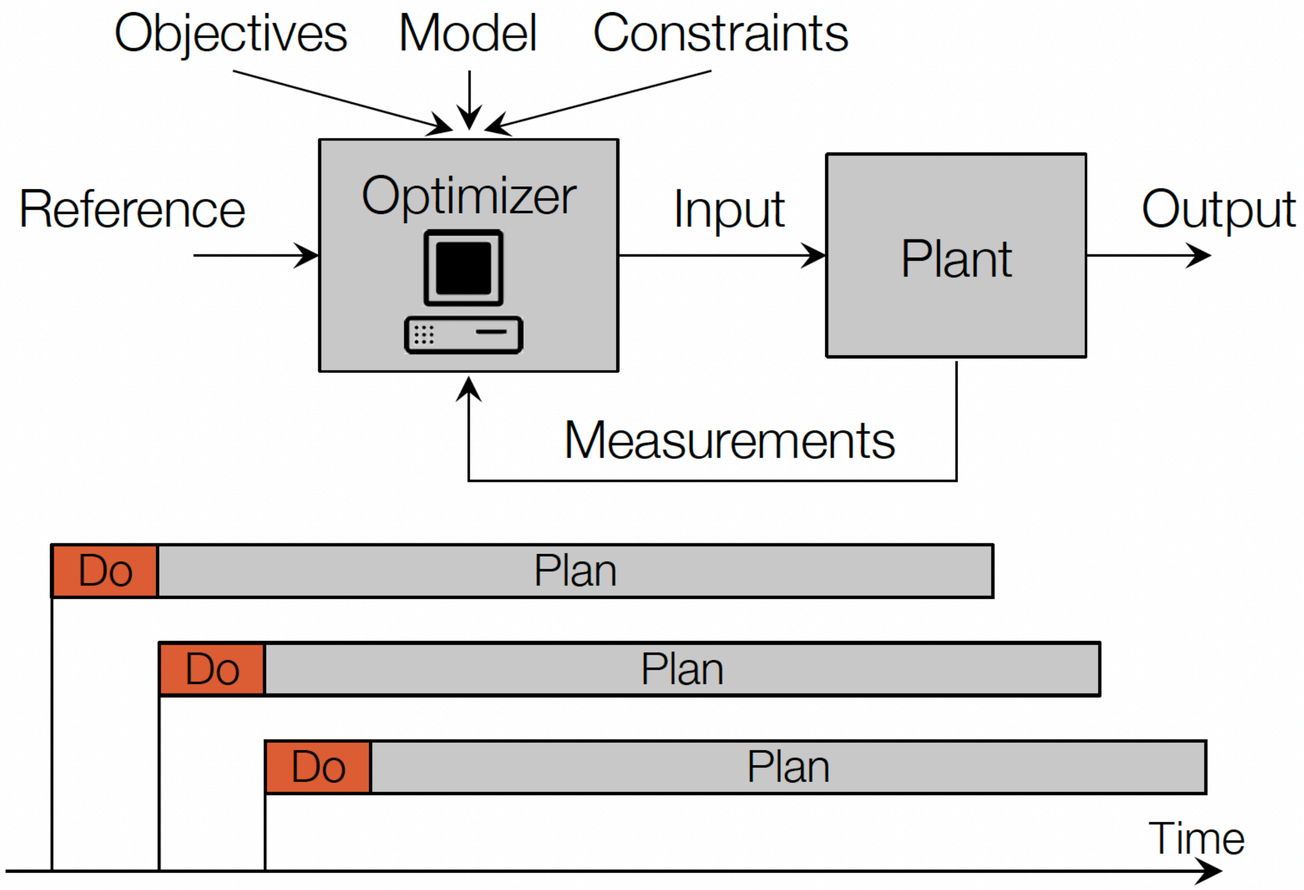
The linearized model is not controllable, because the rank of the controllability matrix is less than the number of states



Model predictive control (MPC)

MPC design consists in applying the action that maximizes the achievement of some objective based on a dynamic model and the current available measurements.

Online optimization technique, can handle non-linear dynamics, MIMO systems, inputs and states constraints.



Problem formulation

OCP to nonlinear programming problem using multiple shooting

$$J_N(\mathbf{x_0}, \mathbf{u}) = \sum_{k=0}^{N-1} V(\mathbf{x}(k), \mathbf{u}(k)) + W(\mathbf{x}(N))$$

Running cost

$$V(\mathbf{x}(k), \mathbf{u}(k)) = ||\mathbf{x}(k) - \mathbf{x}^r(k)||^2_Q + ||\mathbf{u}(k) - \mathbf{u}^r(k)||^2_R$$

Terminal cost

$$W(\mathbf{x}(N)) = ||\mathbf{x}(N) - \mathbf{x}^r(N)||^2_P$$

Constraints

$$\begin{aligned}
\mathbf{x}(0) &= \mathbf{x_0}, \\
\mathbf{x}(k+1) &= \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)); \quad k \in \{0, 1, ...N-1\}, \\
||\mathbf{x}(k) - \mathbf{x}_{obs}(k)|| &\geq r_{rob} + r_{obs} \\
\mathbf{x}_{min} \leq \mathbf{x}(k) &\leq \mathbf{x}_{max} \quad k \in \{1, 2, ...N\}, \\
\mathbf{u}_{min} \leq \mathbf{u}(k) &\leq \mathbf{u}_{max} \quad k \in \{0, 1, ...N-1\}
\end{aligned}$$

$$\sqrt{(x_{rob} - x_{obs})^2 + (y_{rob} - y_{obs})^2} \geq (r_{obs} + r_{rob})$$

MPC Results

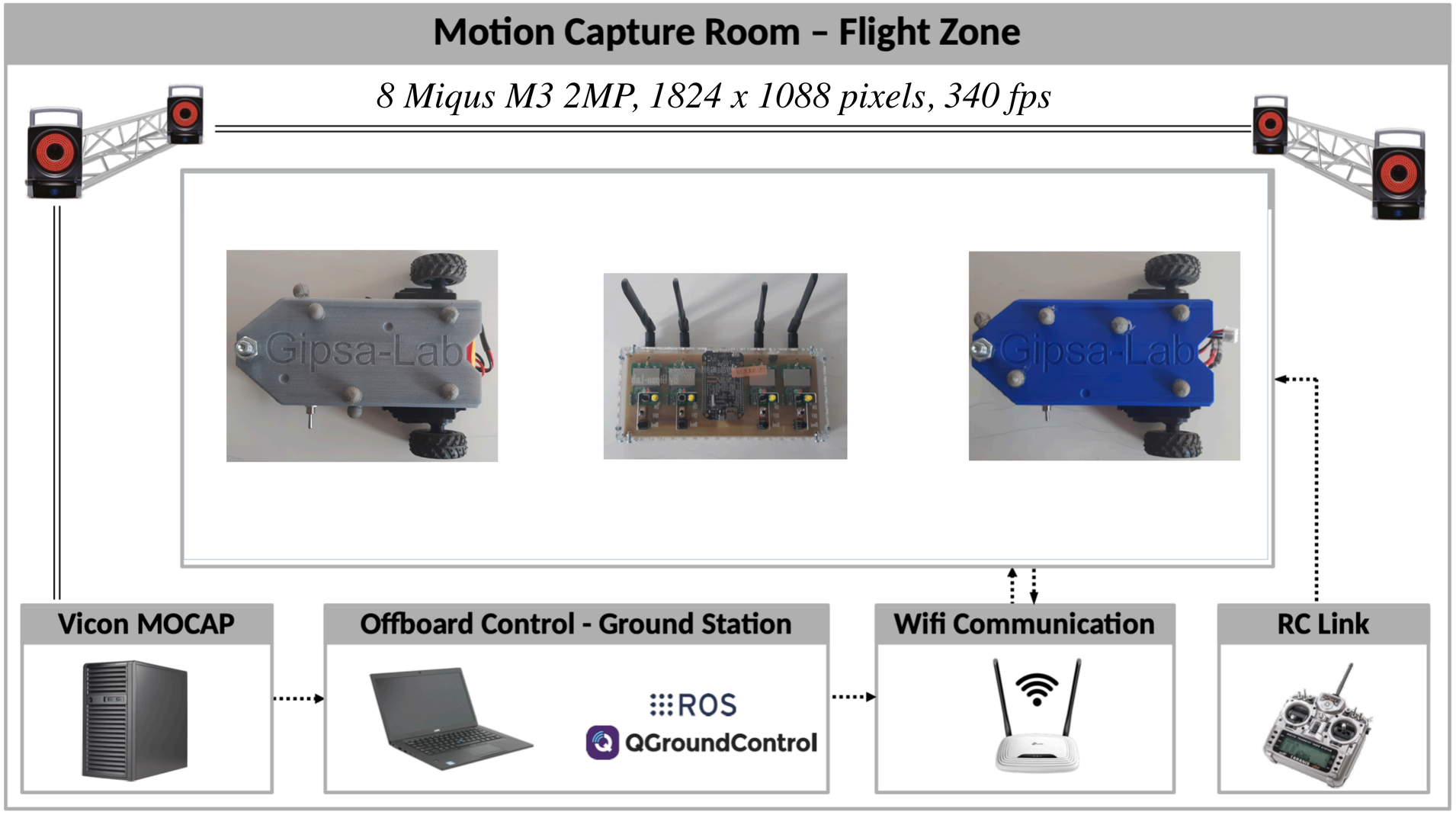
Experimental platform

$$\omega_r = (2v + \omega R_{rob})/2r$$
$$\omega_l = (2v - \omega R_{rob})/2r$$

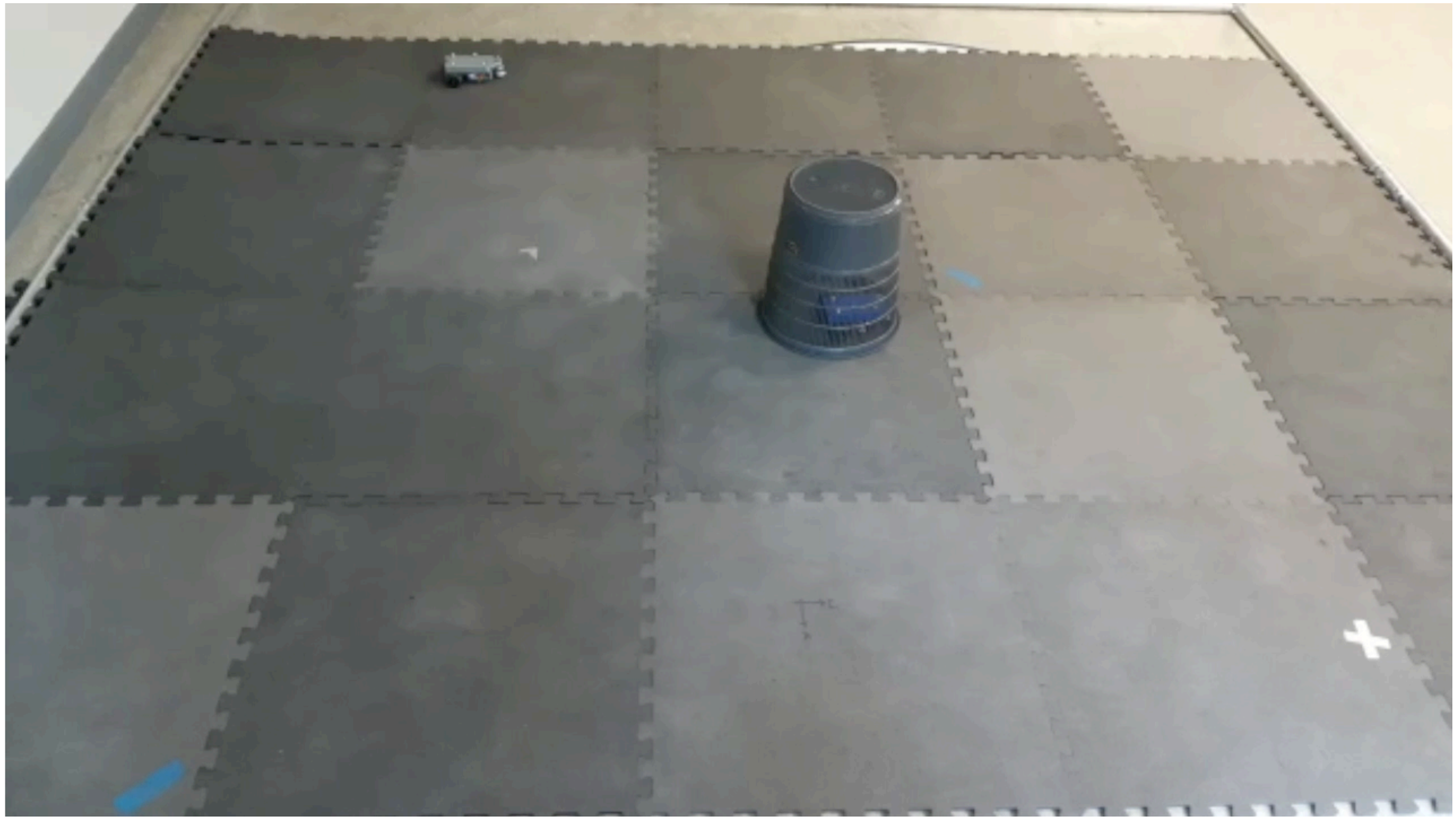
CasADi
Matlab™
Simulink Real-Time™
(IPOPT)

$$v_{max} = 0.065 m/s$$
$$v_{min} = -0.065 m/s$$
$$\omega_{max} = 0.05 rad/s$$
$$\omega_{min} = -0.05 rad/s$$

$$N = 10$$
$$T_S = 0.1 s$$

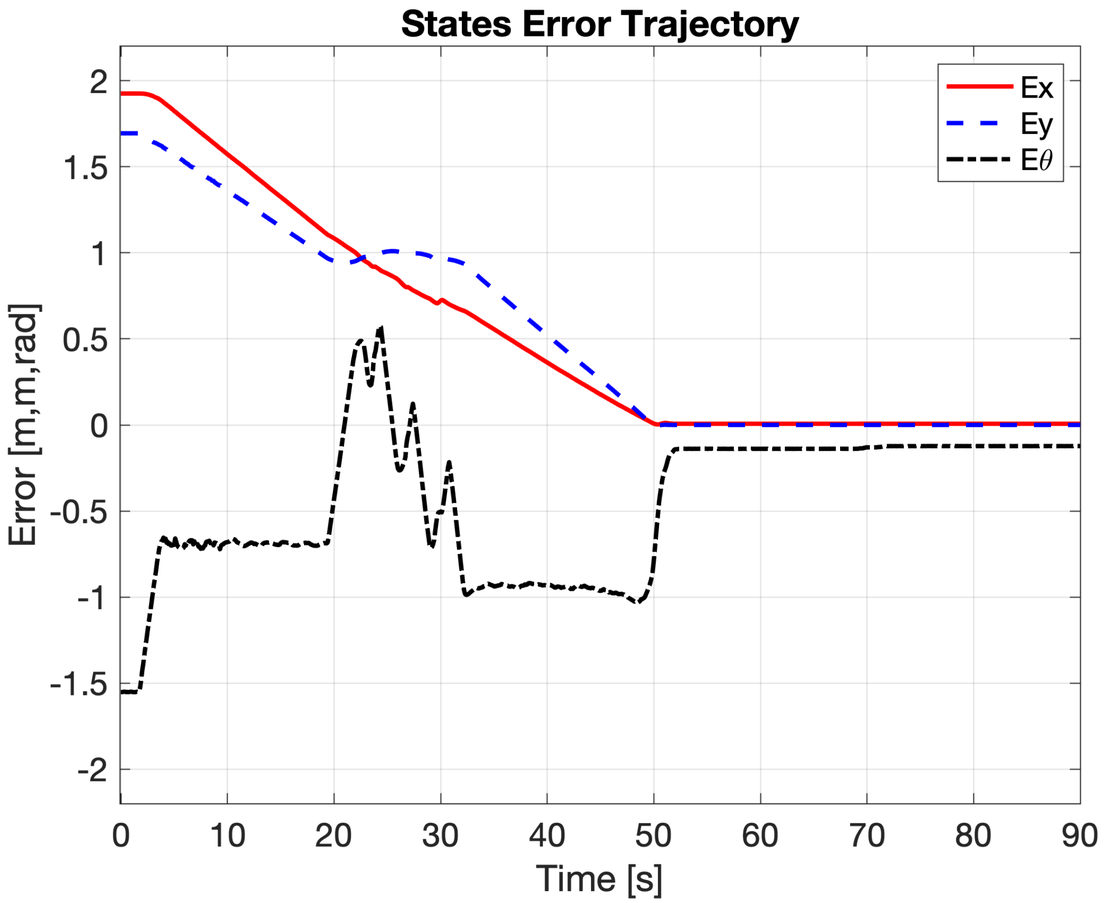


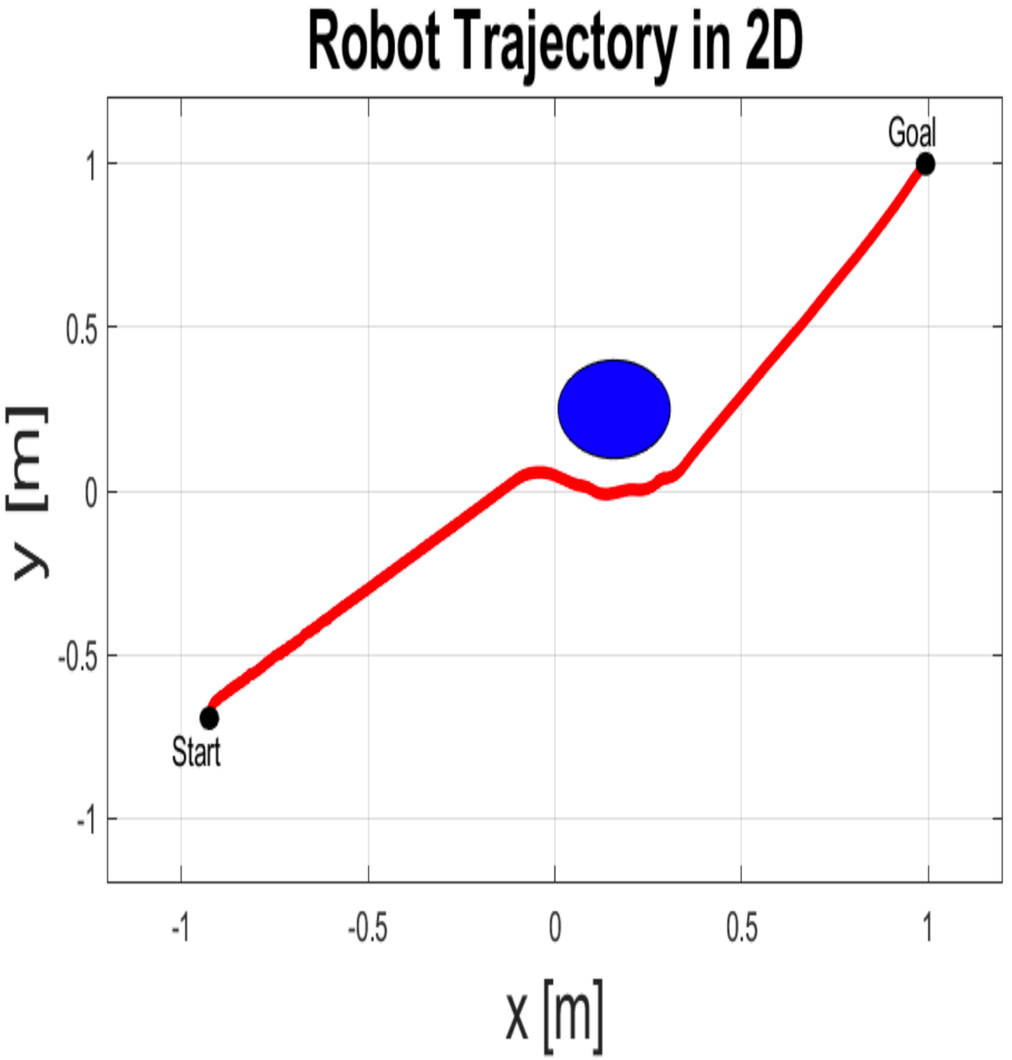
Point stabilization with static obstacles avoidance



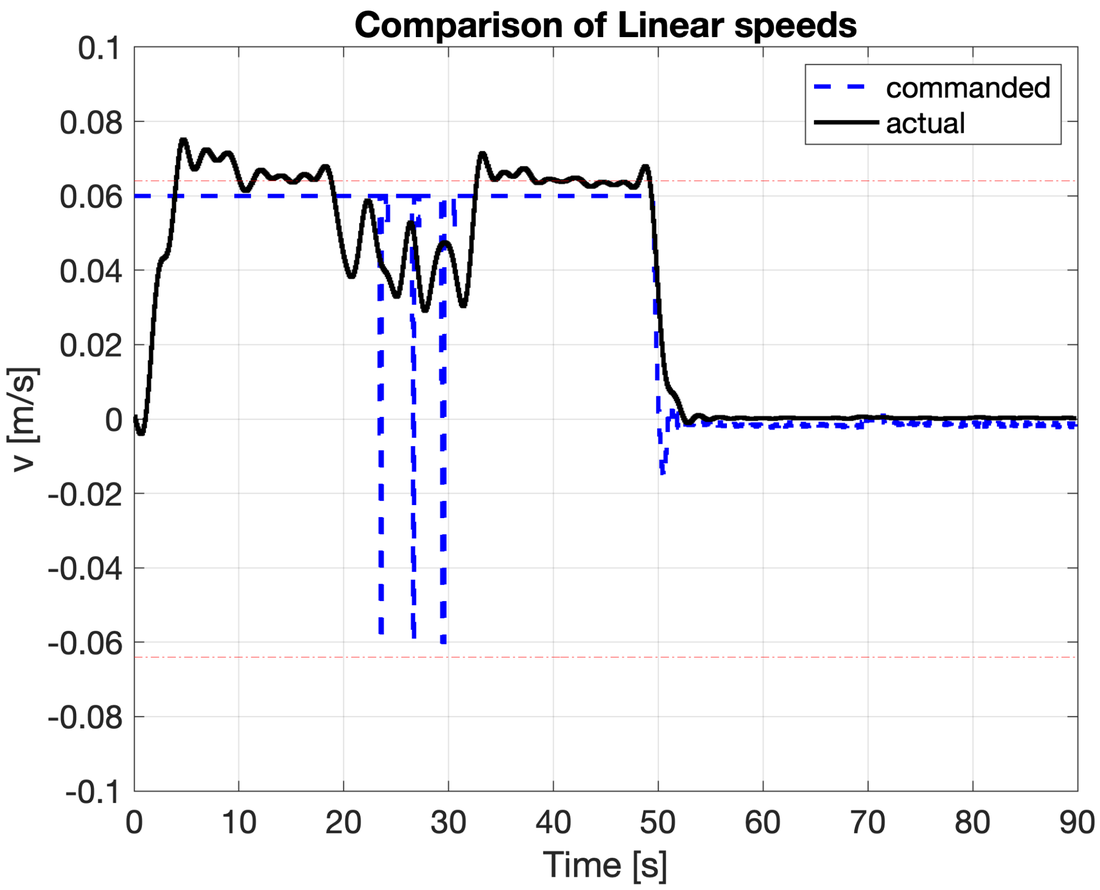
MPC Results

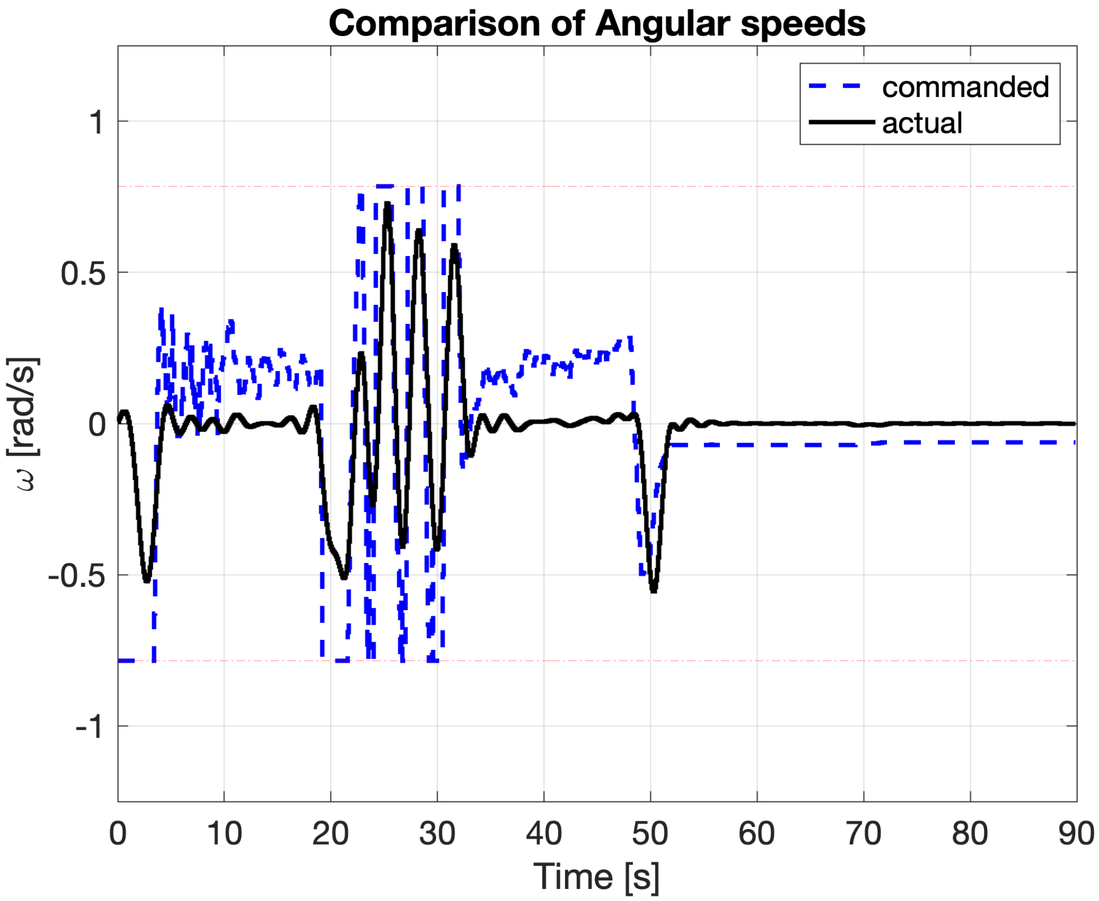
Point stabilization with static obstacles avoidance



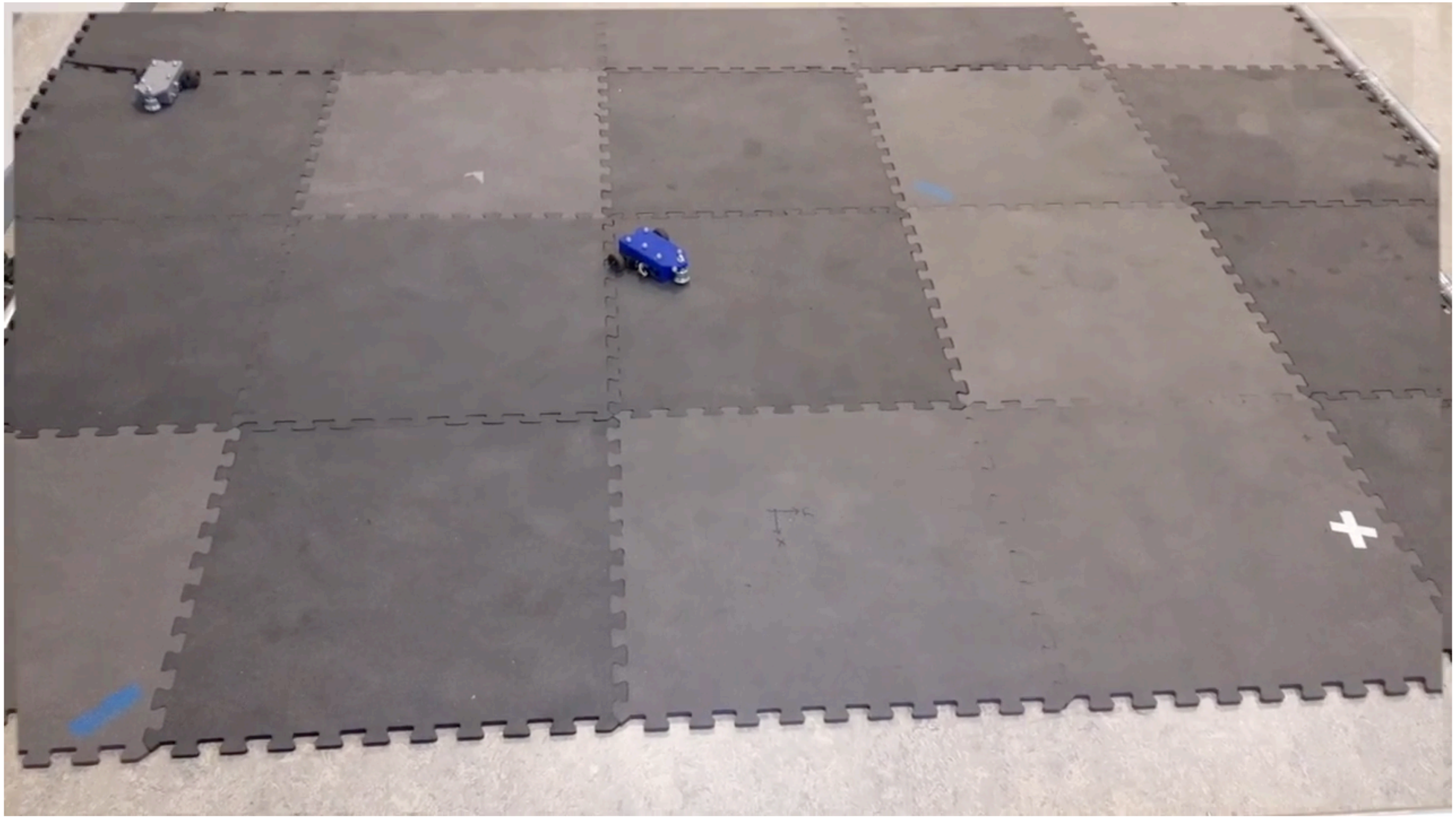


Point stabilization with static obstacles avoidance

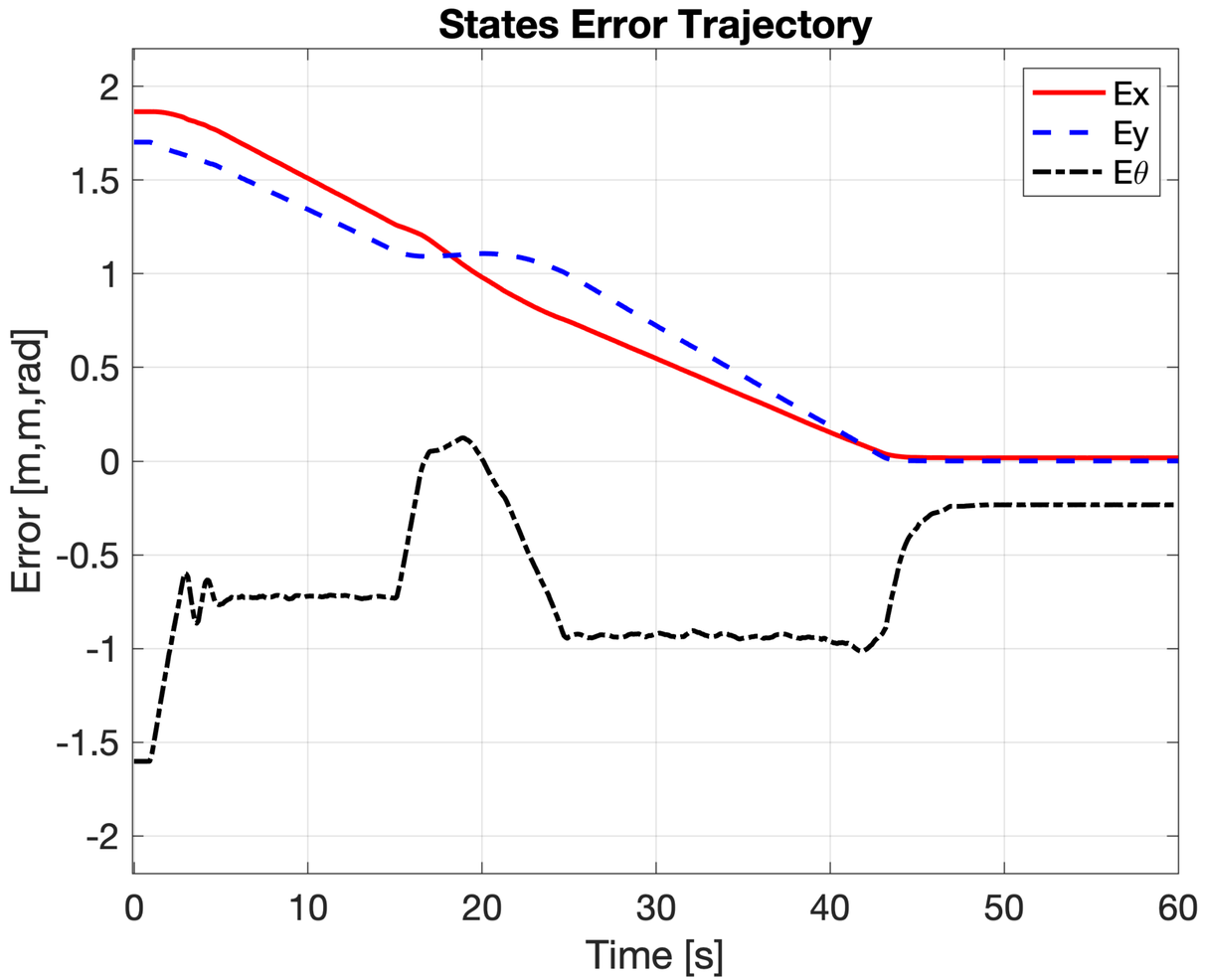


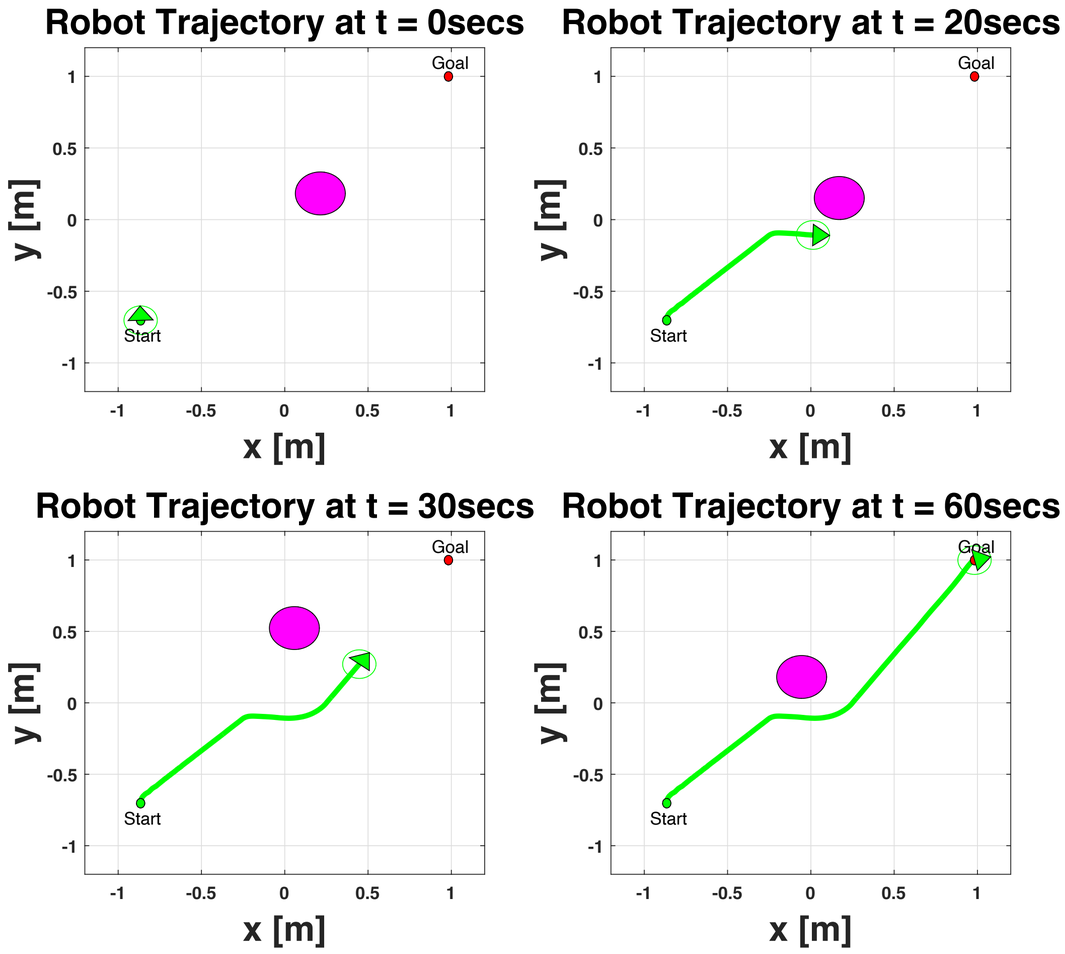


Point stabilization with dynamic obstacles avoidance



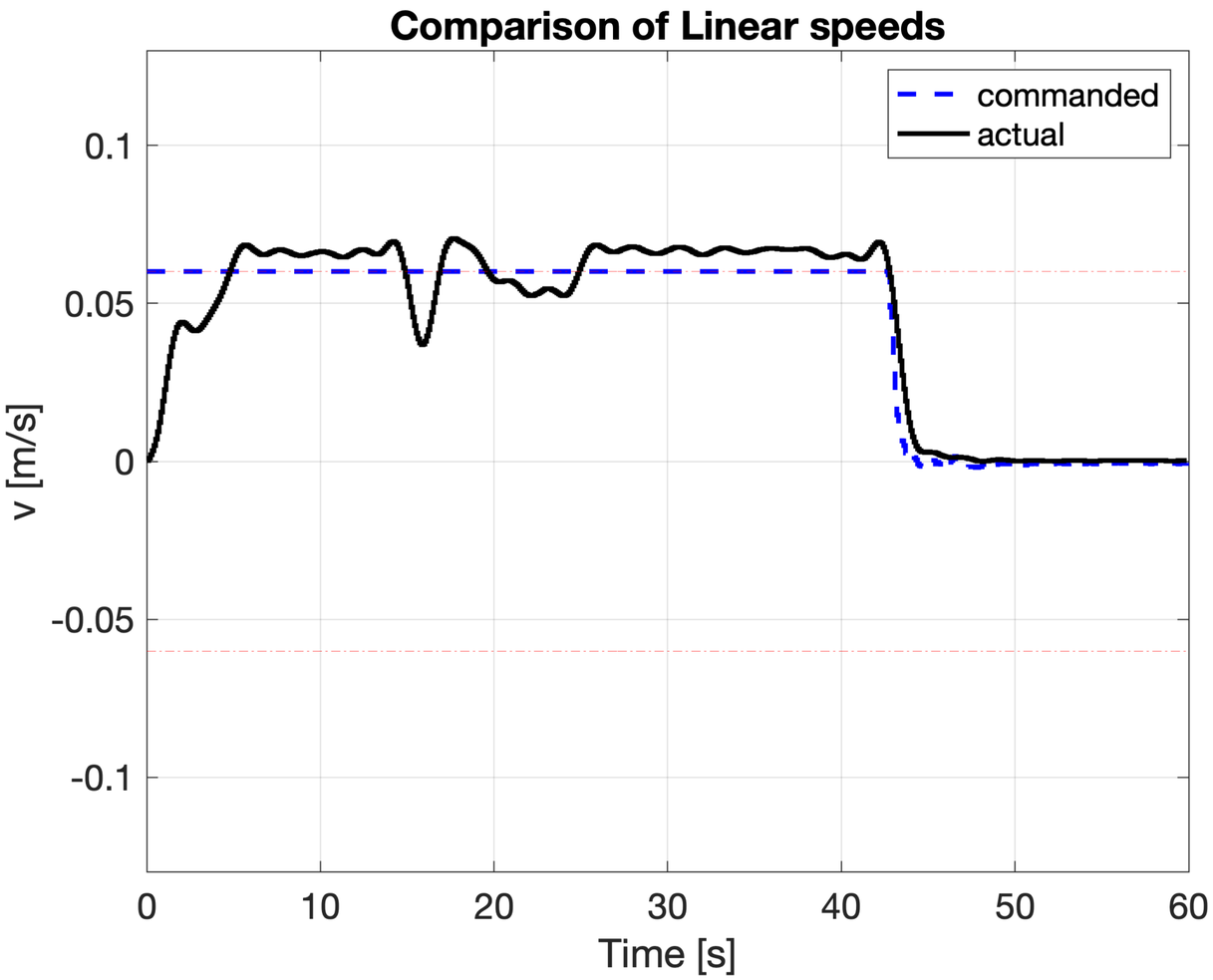
Point stabilization with dynamic obstacles avoidance

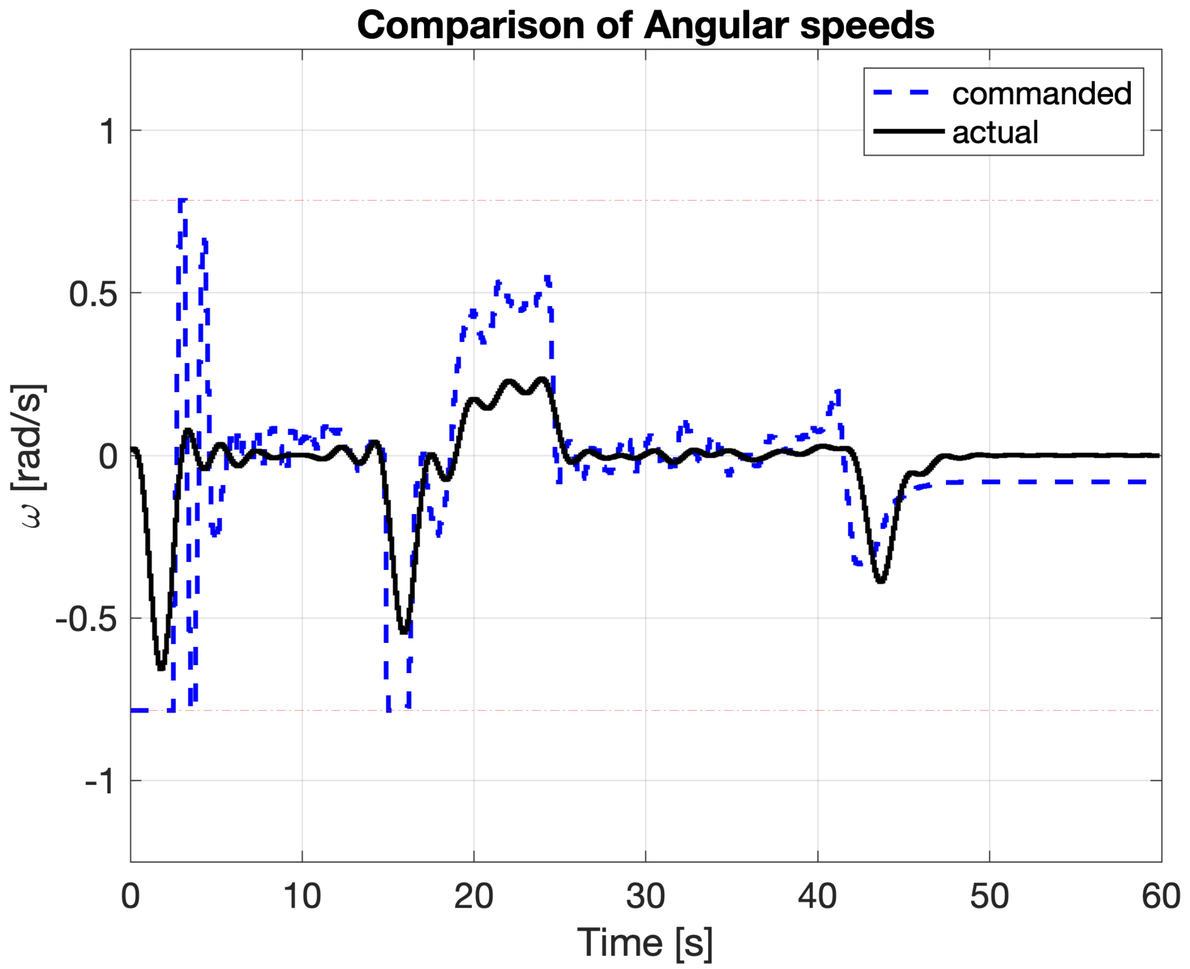




MPC Results

Point stabilization with dynamic obstacles avoidance





Pursuit-evasion games

Extending the point-stabilization problem to dynamic target controlled with conflicting objective: **Pursuit-evasion game**

Players:

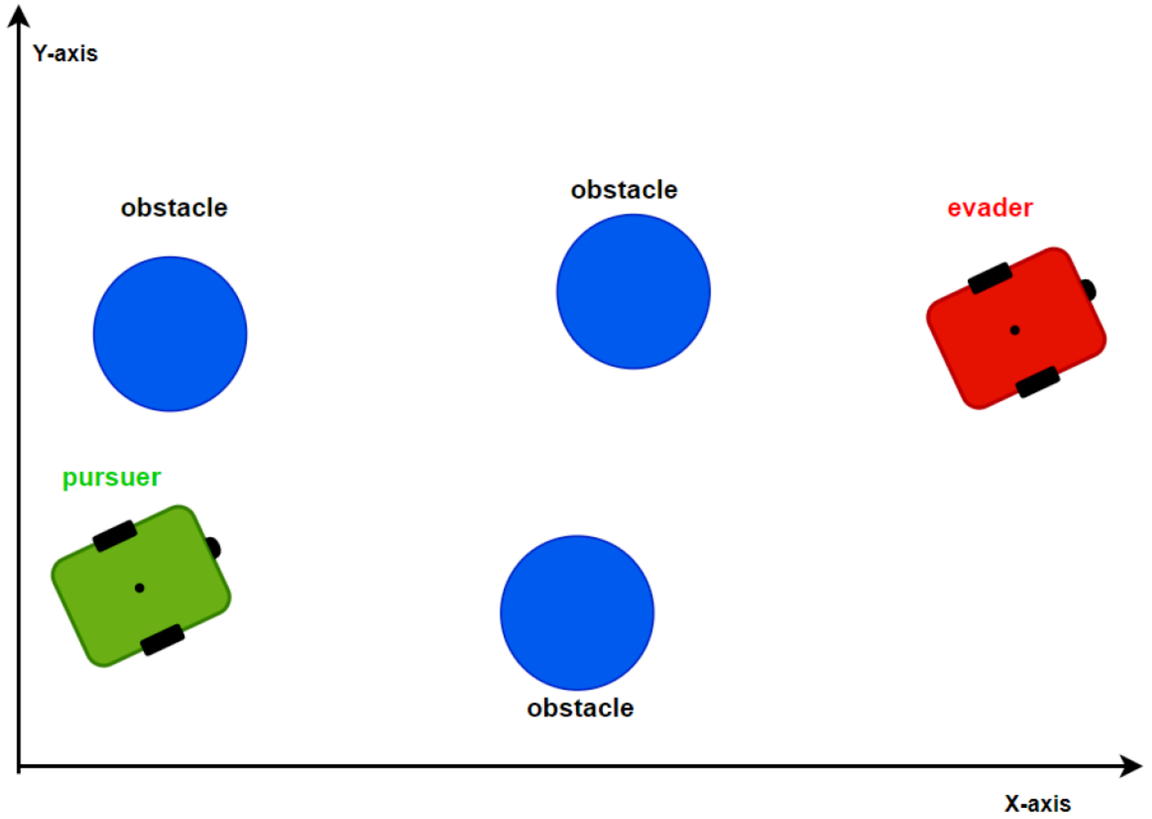
- Pursuer: to catch the evader.
- Evader: to escape from the pursuer.

Performance $\rightarrow$ capture time.

Capture condition: $R.D \leq l$?

$$R.D = \sqrt{(x_p - x_e)^2 + (y_p - y_e)^2}$$

$$l = R_p + R_e \ .$$



Pursuit-evasion games

Pursuer solves minimization problem and Evader solves maximization problem

pursuer

$$\min_{\mathbf{u}_p} J = ||\mathbf{x}_p(N) - \mathbf{x}_e||^2_{Q_N} + \sum_{k=0}^{N-1} ||\mathbf{x}_p(k) - \mathbf{x}_e||^2_Q + ||\mathbf{u}_p(k)||^2_R$$

subject to:

$$\begin{cases} \mathbf{x}_p(k+1) = f_p(\mathbf{x}_p(k), \mathbf{u}_p(k)), \quad k = 0, 1, ..N-1 \\ \sqrt{(x_p(k) - x_{obs}(k))^2 + (y_p(k) - y_{obs}(k))^2} \geq (R_{obs} + R_p) \\ \mathbf{x}_{p_{min}} \leq \mathbf{x}_p(k) \leq \mathbf{x}_{p_{max}}; \quad \mathbf{u}_{p_{min}} \leq \mathbf{u}_p(k) \leq \mathbf{u}_{p_{max}} \end{cases}$$

$$\max_{\mathbf{u}_e} J = ||\mathbf{x}_e(N) - \mathbf{x}_p||^2_{Q_N} + \sum_{k=0}^{N-1} ||\mathbf{x}_e(k) - \mathbf{x}_p||^2_Q + ||\mathbf{u}_e(k)||^2_R$$

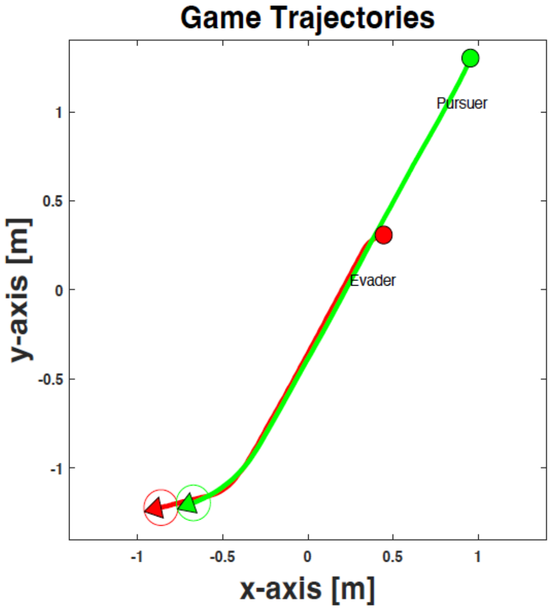
subject to:

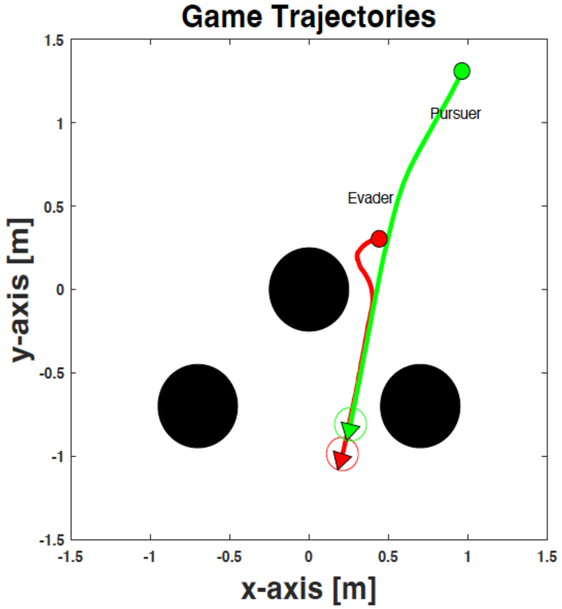
$$\begin{cases} \mathbf{x}_e(k+1) = f_e(\mathbf{x}_e(k), \mathbf{u}_e(k)), \quad k = 0, 1, ..N-1 \\ \sqrt{(x_e(k) - x_{obs}(k))^2 + (y_e(k) - y_{obs}(k))^2} \geq (R_{obs} + R_e) \\ \mathbf{x}_{e_{min}} \leq \mathbf{x}_e(k) \leq \mathbf{x}_{e_{max}}; \quad \mathbf{u}_{e_{min}} \leq \mathbf{u}_e(k) \leq \mathbf{u}_{e_{max}} \end{cases}$$

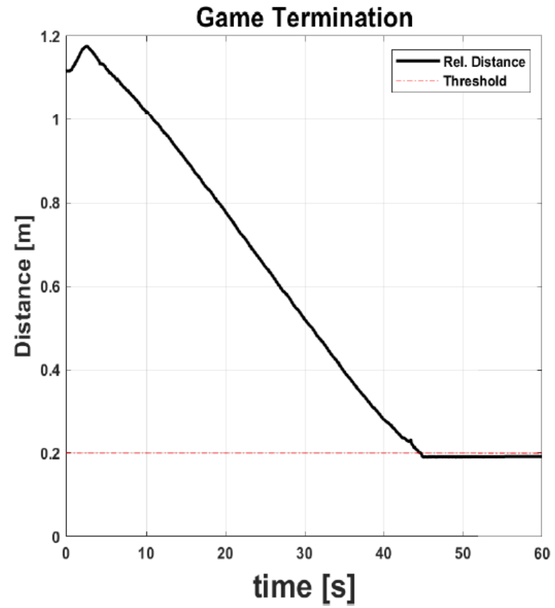
evader

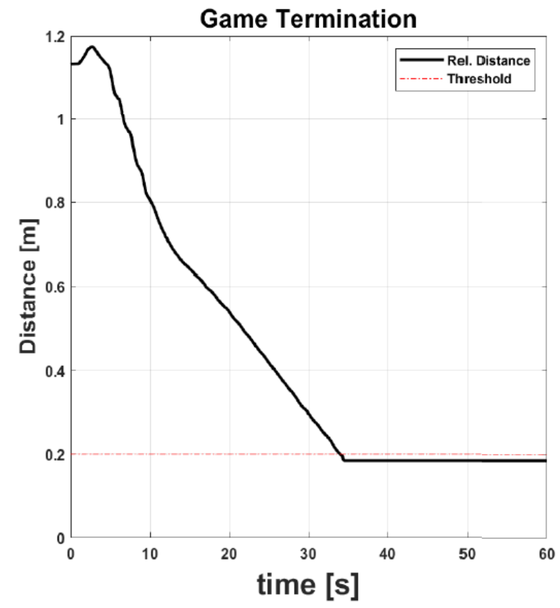
Pursuit-evasion games

Experimental results with and without obstacles





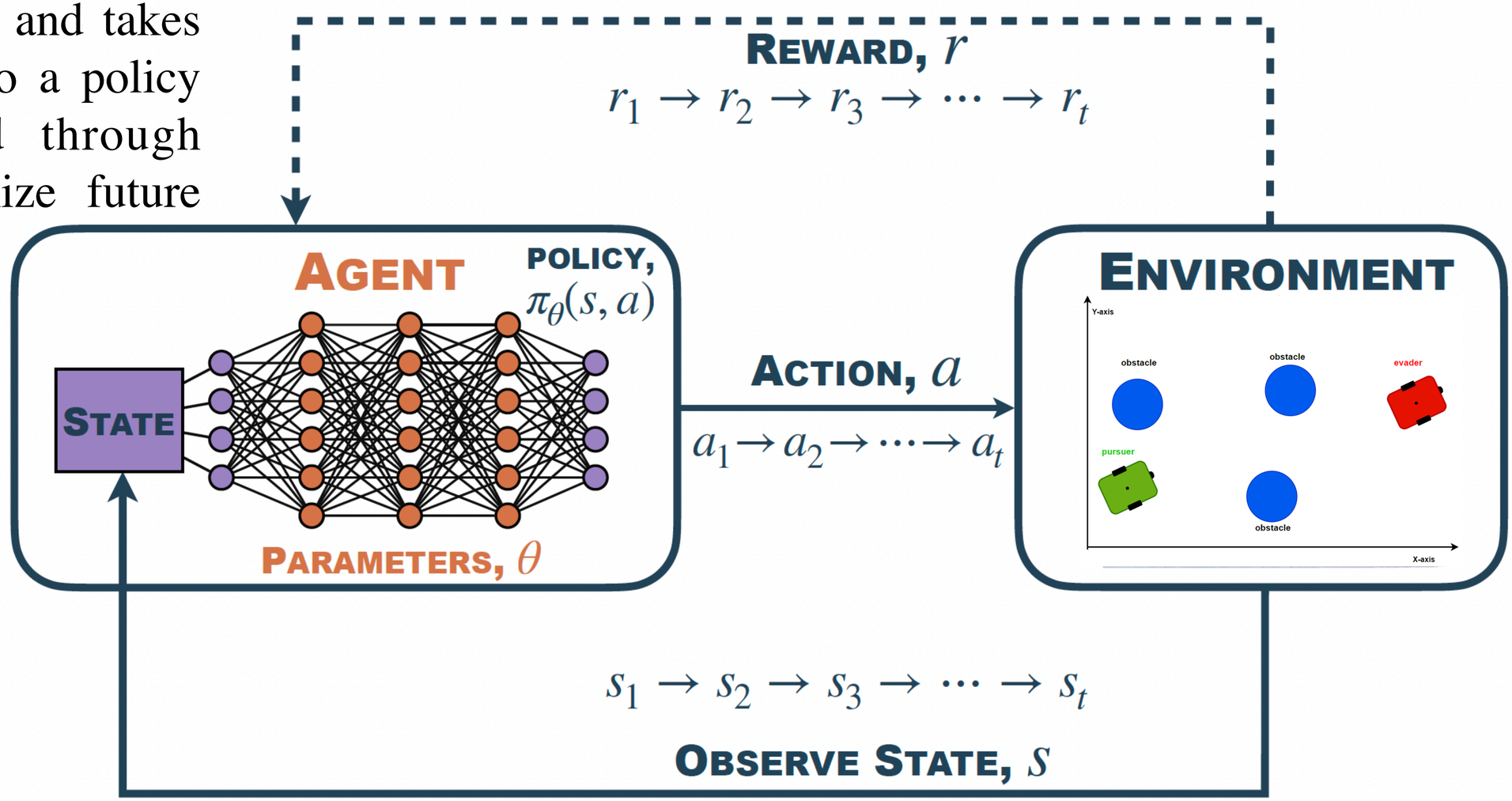




Another approach

Reinforcement Learning

Agent senses its environmental state and takes actions according to a policy that is optimized through learning to maximize future rewards



Another approach

Twin delayed deep deterministic network

Every training episode has a maximum number of steps, that if reached, the evader wins the game. The pursuer wins only when it catches the evader before the end of the episode.

- Consider only the last 200 episodes
- The number of steps must be above 400, equivalent to 40 seconds
- Take the agent with the highest score
- Additional check of the validity of the episode

Algorithm 1: TD3

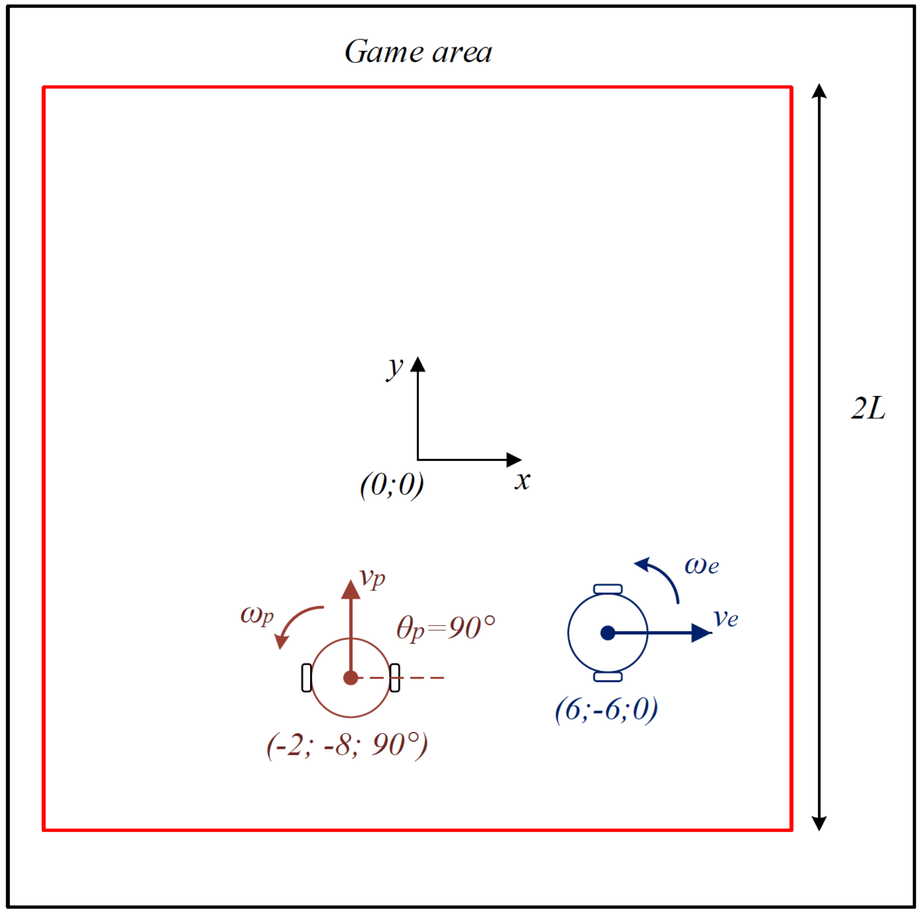
Initialize pursuer networks π_ϕ, $\mathcal{Q}_{\theta_1}$ and $\mathcal{Q}_2$ with random parameters ϕ, θ_1, θ_2
Initialize pursuer target networks $\phi' \leftarrow \phi$, $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$
Initialize pursuer Replay Buffer B
for t =1 **to** T **do**
Select action with exploration noise $a \sim \pi_{\phi_p} + \epsilon$,
$\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
Execute actions and observe rewards r and new state s'
Store transition tuple (s, a, r, s') in B
Sample mini batch of N transitions (s, a, r, s') from B
$\tilde{a} \leftarrow \pi'_{\phi'}(s') + \epsilon_p$, where $\epsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
$y \leftarrow r + \gamma \, min_{i=1,2} \, \mathcal{Q}'_{\theta_i'}(s', \tilde{a})$
Update critics $\theta_i \leftarrow argmin_{\theta_i} N^{-1} \sum (y - \mathcal{Q}_{\theta_i}(s, a))^2$
if t mod d **then**
Update ϕ by deterministic policy gradient:
$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a \mathcal{Q}_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
Update target Networks
$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
end if
end for

Some notations

Every training episode has a maximum number of steps, that if reached, the evader wins the game. The pursuer wins only when it catches the evader before the end of the episode.

- At each step k we measure the Euclidian distance D between the centers of the two robots

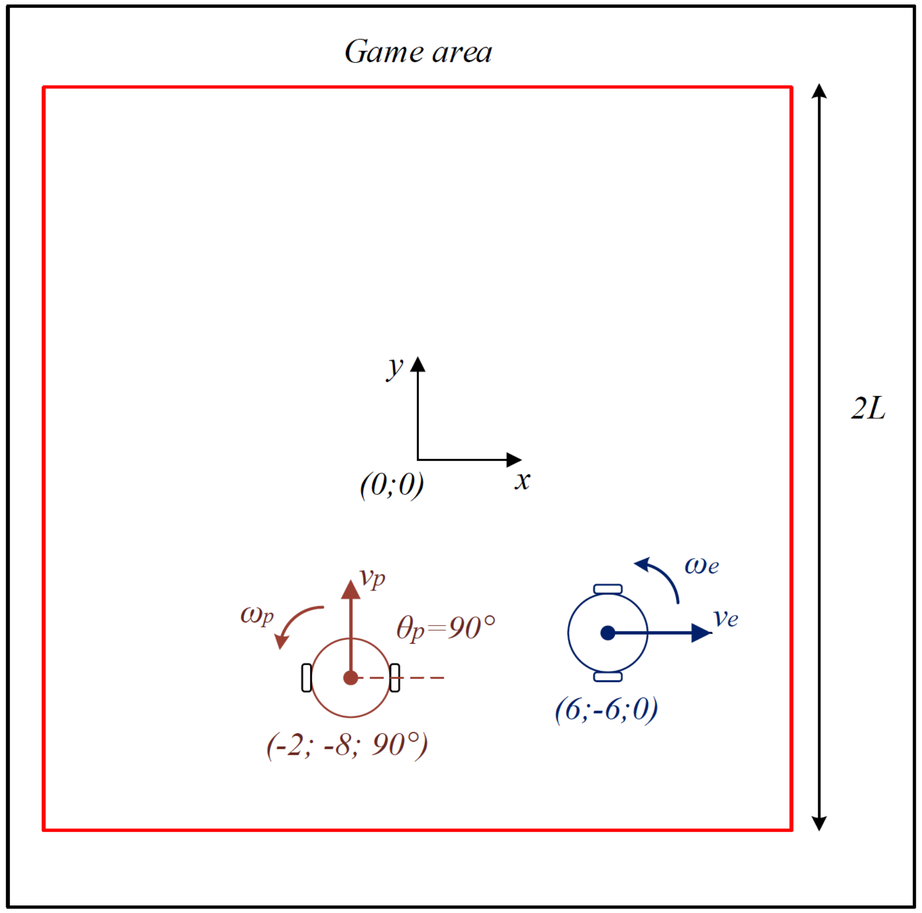
- Both agents have the same maximum steering velocity ωmax , meaning that no robot have an agility advantage



Observations, actions and transitions

Normalized observation that serves as input to the neural net and saved in the replay buffer

$$s^{(k)} = \left(\frac{x_p^{(k)}}{L}, \frac{y_p^{(k)}}{L}, \frac{\theta_p^{(k)} mod\ 2\pi}{2\pi}, \frac{x_e^{(k)}}{L}, \frac{y_e^{(k)}}{L}, \frac{\theta_e^{(k)} mod\ 2\pi}{2\pi}\right)$$



Actions:

- Linear velocities remain constant
- [-1, 1], which is multiplied by the maximum steering velocity ωmax

Transitions:

$$\begin{cases} x_i^{(k+1)} = x_i^{(k)} + T_s . v_i^{(k)} . cos\theta_i^{(k)} \\ y_i^{(k+1)} = y_i^{(k)} + T_s . v_i^{(k)} . sin\theta_i^{(k)} \\ \theta_i^{(k+1)} = \theta_i^{(k)} + T_s . \omega_i^{(k)} \end{cases}$$

Reward function

Step reward

$$r_1 = \Delta D * g = (D_k - D_{k+1}) * g$$

The pursuer receives the reward $r1$, which, when positive, means that the pursuer is closing the gap. On the other hand, the evader receives a reward of -$r1$, which, when positive, means that the evader is distancing the pursuer

Time penalty

$$r_3 = -10$$

Failure penalty

$$r_4 = \begin{cases} -2000, & if \ |x_i| \geq L - 2R_{rob} \ or \ |y_i| \geq L - 2R_{rob} \\ -10, & if \ |x_i| \geq L - 2R_{rob} - b \ or \ |y_i| \geq L - 2R_{rob} - b \\ 0, & if \ |x_i| < L - 2R_{rob} - b \ and \ |y_i| < L - 2R_{rob} - b \end{cases}$$

Outcome reward

$$r_2 = \begin{cases} +1000, & if \ D \leq D_{capture} \\ -1000, & if \ step = 1000 \\ 0, & else \end{cases}$$

Final reward

$$r_p = r_1 + r_2 + r_3 + r_{4_p}$$

$$r_e = -r_1 - r_2 - r_3 + r_{4_e}$$

Parameters

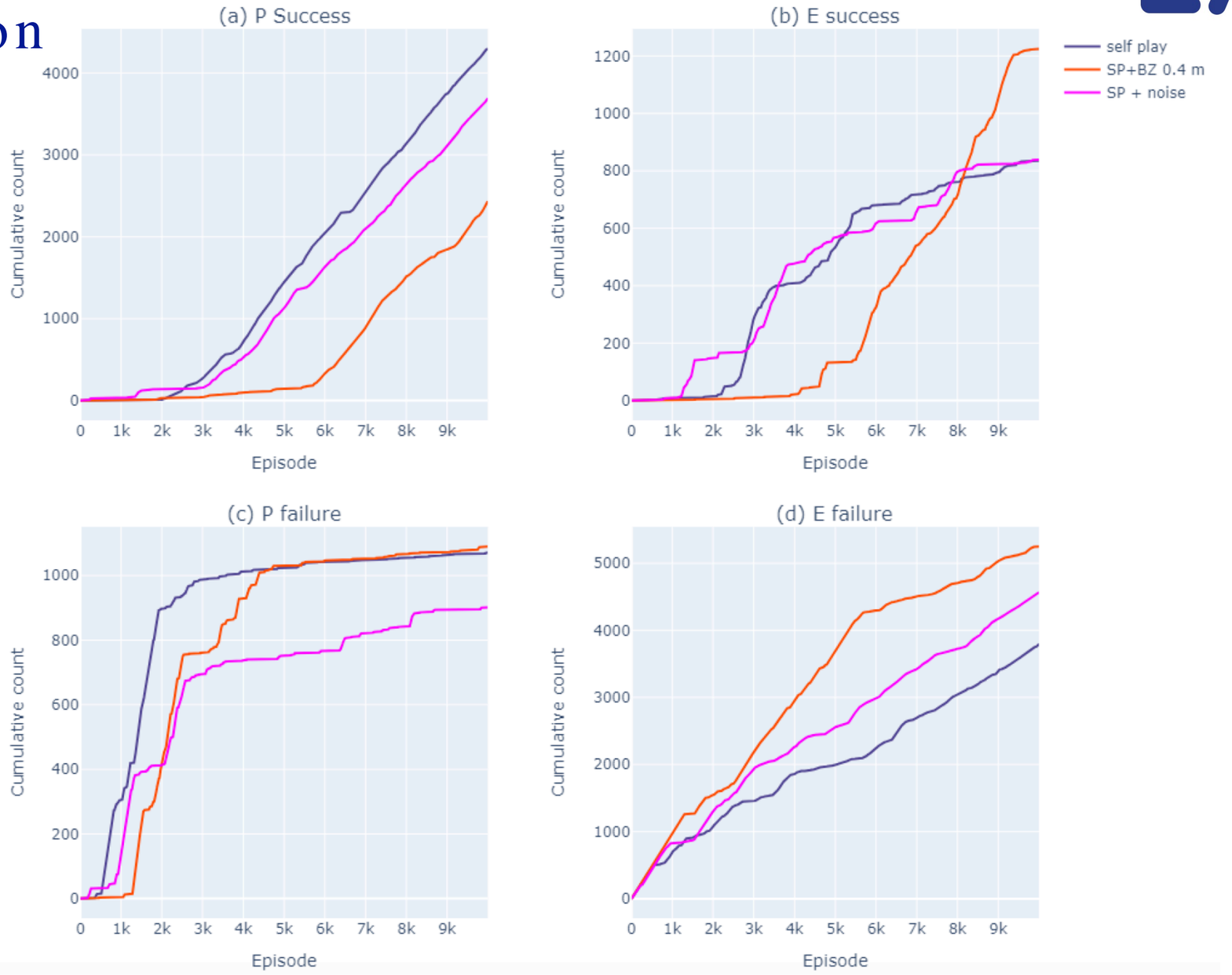
Parameter	Value	Description
State dimension	6	
Action dimension	1	
M	10000	Training episodes
P	1000	Steps per episode
W	10000	Warmup steps
X_p	(-2,-8, π/2)	Pursuer initial conditions
X_e	(6,-6, 0)	Evader initial conditions
L	10 m	Game area length/2
b	0.4	Buffer zone width
ω_{max}	π/3 (rad/s)	Maximum steering velocity
v_p	0.5 m/s	Pursuer maximum velocity
v_e	0.48 m/s	Evader maximum velocity
Rrob	0.1 m	Robot radius
T_s	0.1	Sampling time
D capture	0.3 m	Capturing distance
g	1000	Reward scaling factor
noise	μ = 0, σ =0.1	Random noise of 3rd scenario

Parameter	Value
Discount factor γ	0.99
Soft update factor τ	0.005
Target policy noise σ	0.2
Policy noise clipping c	0.5
Actor update delay d	2
Actor layers sizes	[256, 256, 1]
Actor activation	[Relu, Relu, tanh]
Critic layers sizes	[256, 256, 1]
Critic activation	[Relu, Relu, None]

Learning rate for actor	0.0003
Learning rate for critic	0.0003
Optimizer	Adam
Replay memory size B	1000000
Minibatch size	256

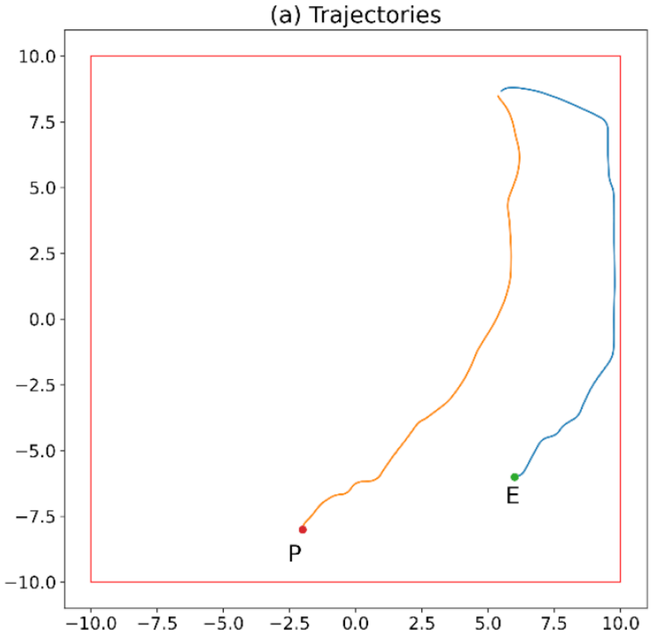
Training

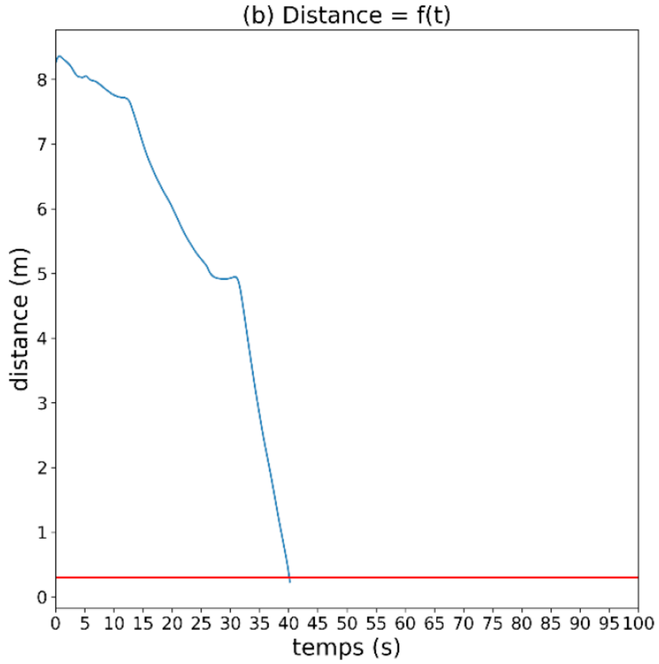
Evolution



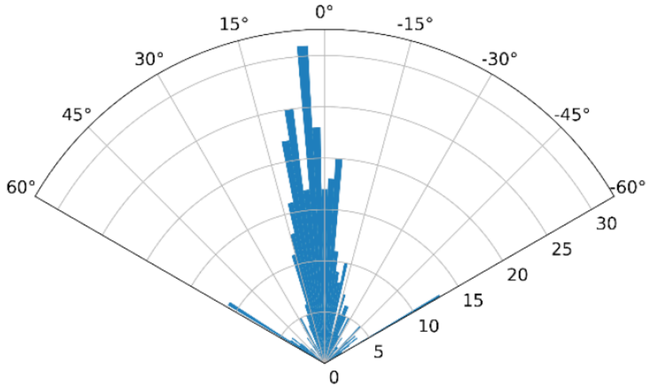
Self play

Pursuer does not follow the exact path of the evader, but rather follows a trajectory that traps the evader into the corner in order to close the gap

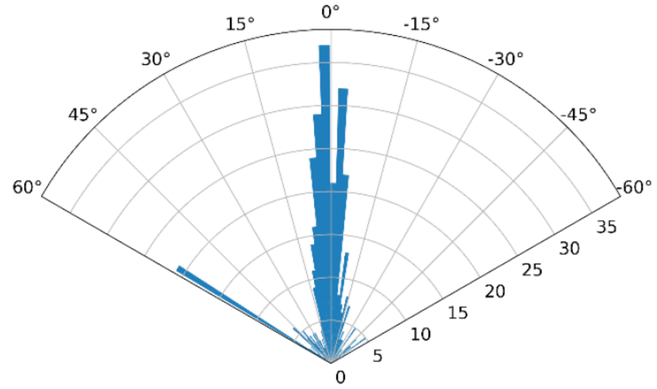




(c) Histogramme des actions de p

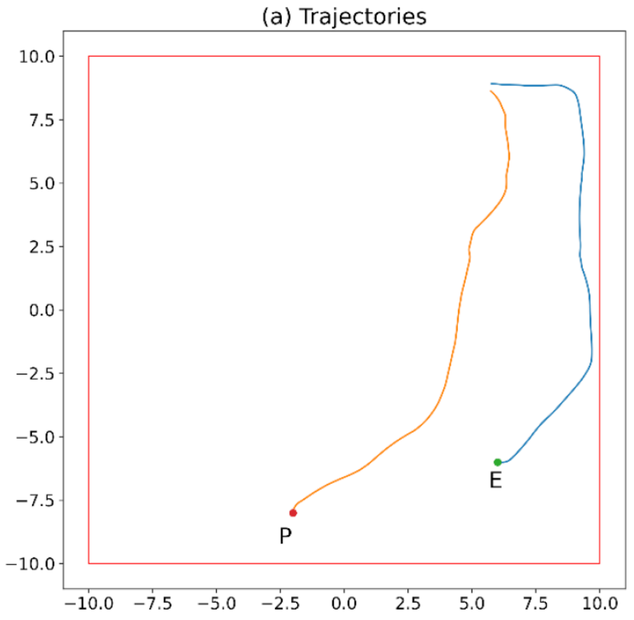


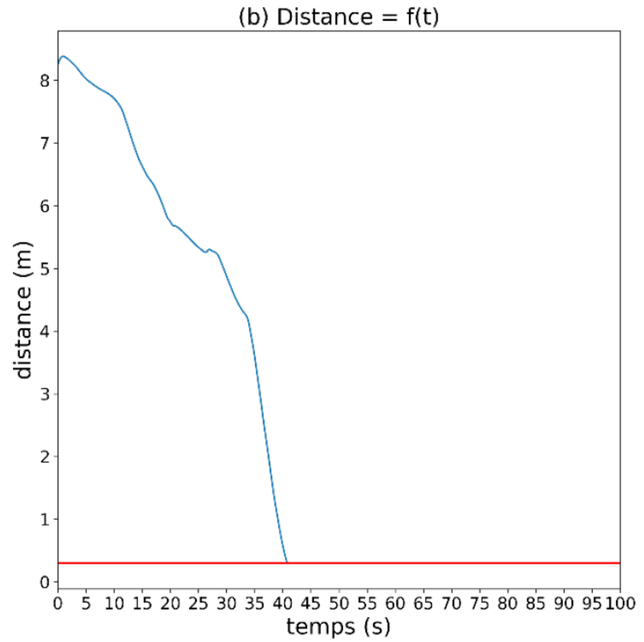
(d) Histogramme des actions de e



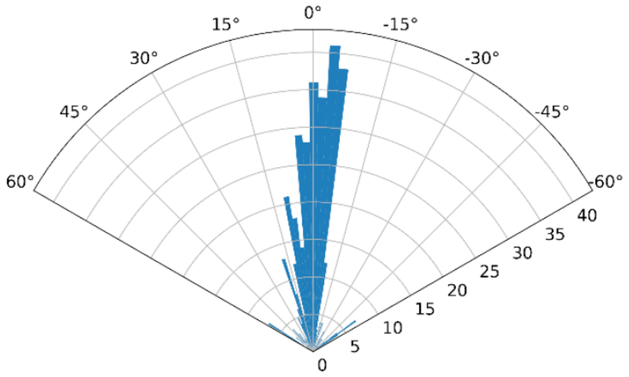
Self-play scenario with a buffer zone

Same behavior but both agents have trajectories slightly skewed to the center of the game area.

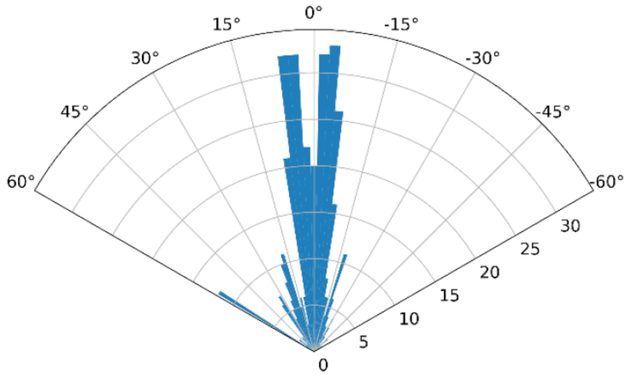




(c) Histogramme des actions de p



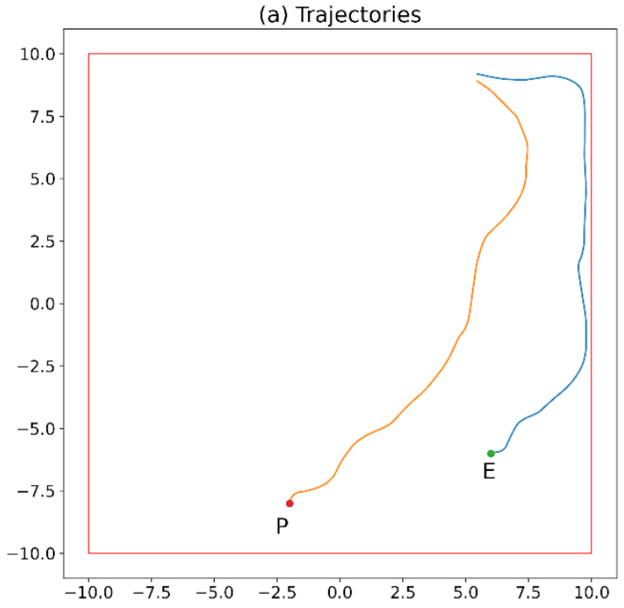
(d) Histogramme des actions de e

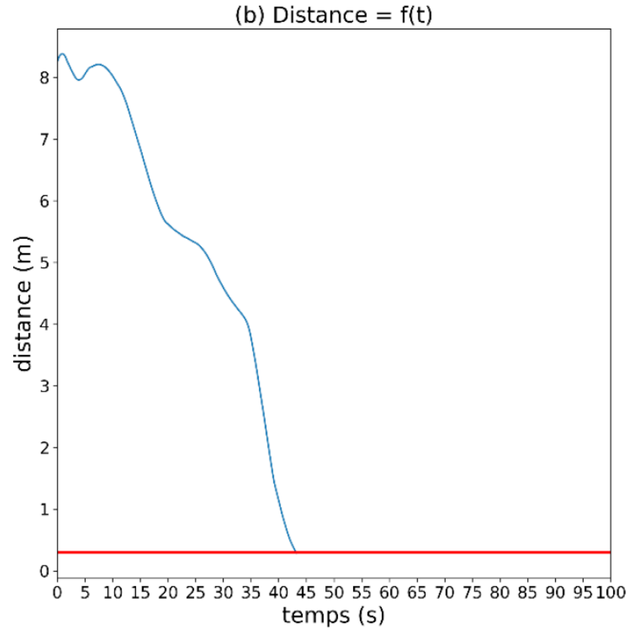


Self-play scenario with noise

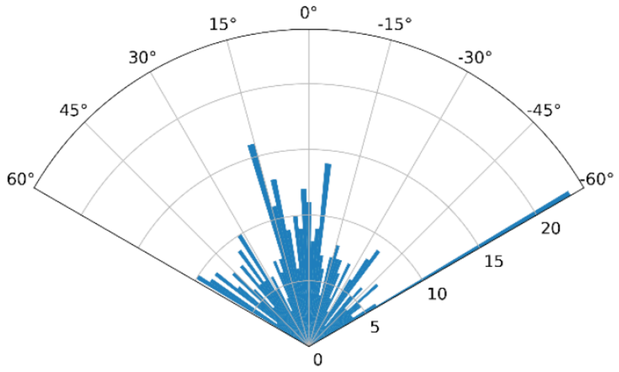
Histograms are more dispersed while more extreme actions are taken by both agents

Capture time is slightly higher than the self-play capture time

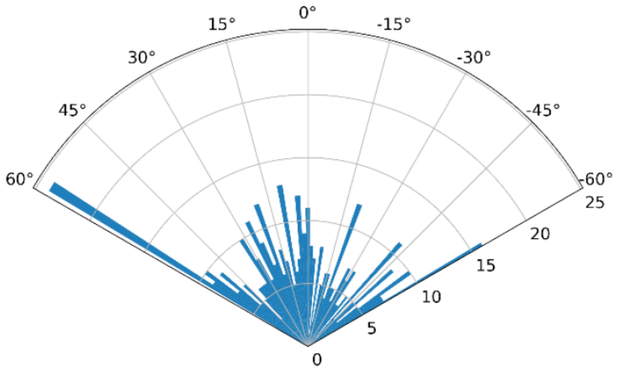




(c) Histogramme des actions de p

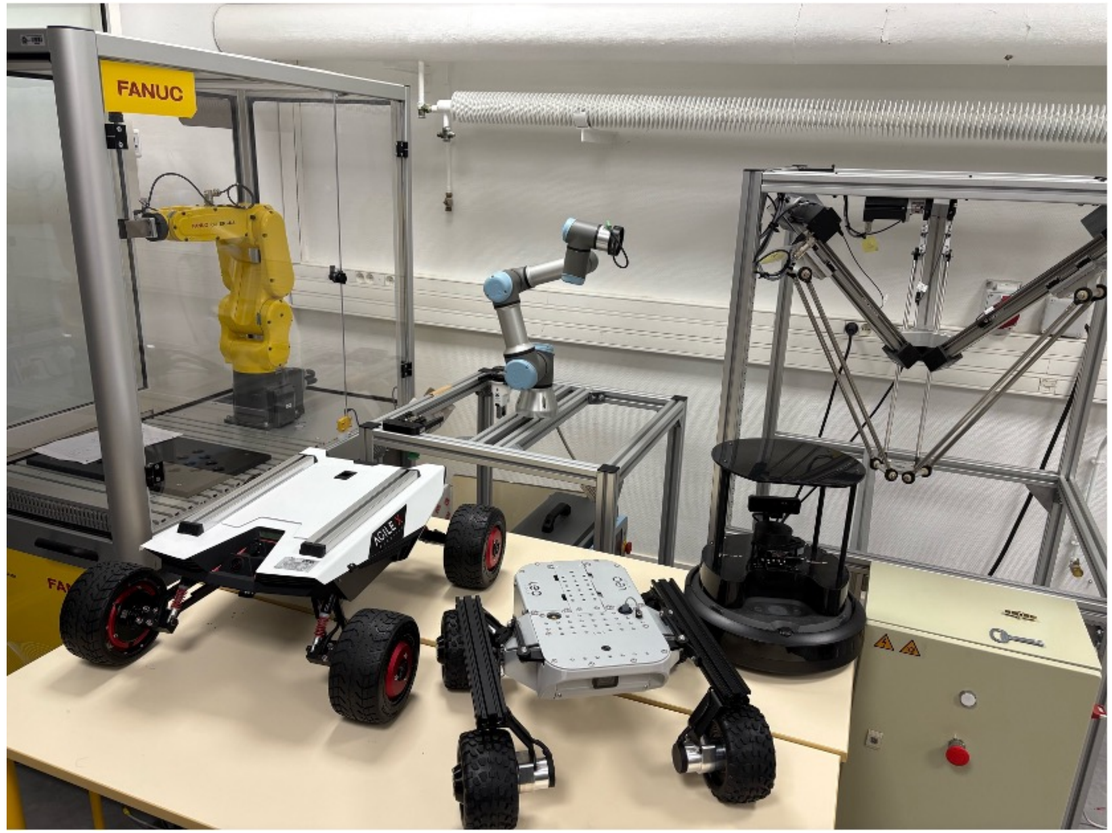


(d) Histogramme des actions de e



Outlooks

- Enhancing computational efficiency
- Use the approach developed in the internship of Sander Miller Murphy
  - using LIDAR with no motion capture system
  - Safety with CBF
- Apply to other mobile robots like Turtlebot4
- Incorporating multi-agent interactions



Thank you …

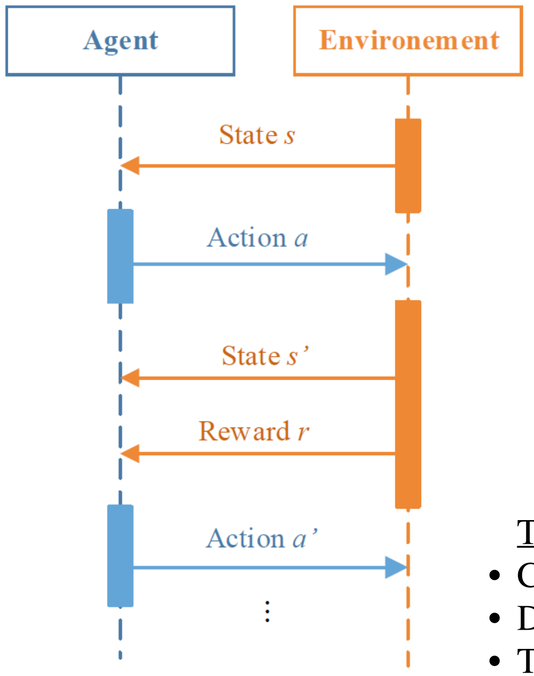
Questions…

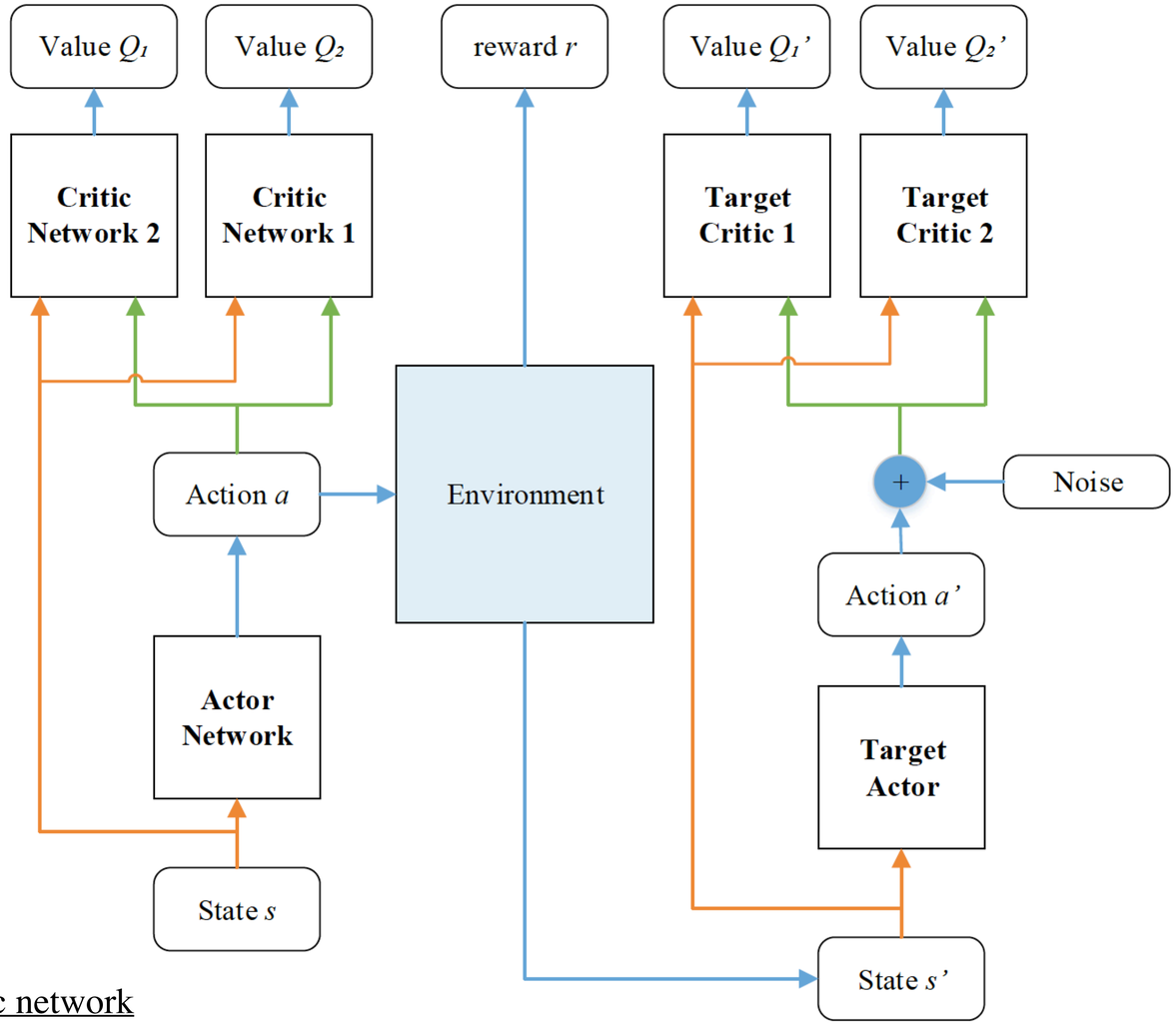
Another approach

Reinforcement Learning

Markov decision process (MDP) framework to define the interaction between a learning agent and the environment

$$Q_n(s,a) = Q_{n-1}(s,a) + \alpha\left(r + \gamma \max_{a'} Q_{n-1}(s',a') - Q_{n-1}(s,a)\right)$$





Twin delayed deep deterministic network

- Clipped double Q-learning to reduce value overestimation
- Delay the update of the actor and target networks by performing it every d steps.
- Target Policy Smoothing Regularization, by adding noise to the input of the target critic network, in order to better estimate the Q value.