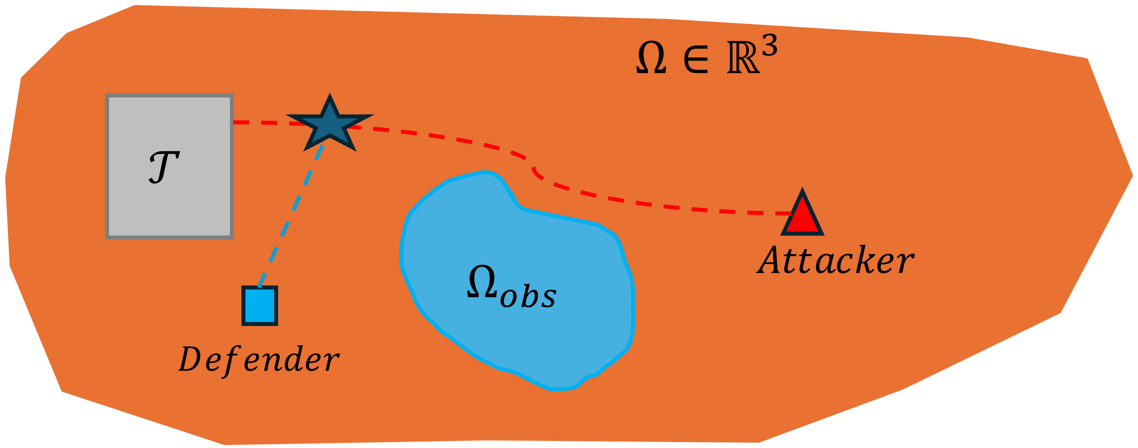
Reach-Avoid Differential game with Reachability Analysis for UAVs: A decomposition approach*

Minh Bui (1) Simon Monckton Mo Chen

(*): Published in the Journal of Guidance, Control and Dynamics (JGCD)
(1): Work done during internship at Defense Research & Development Canda (DRDC)

Motivations

Reach-Avoid Game

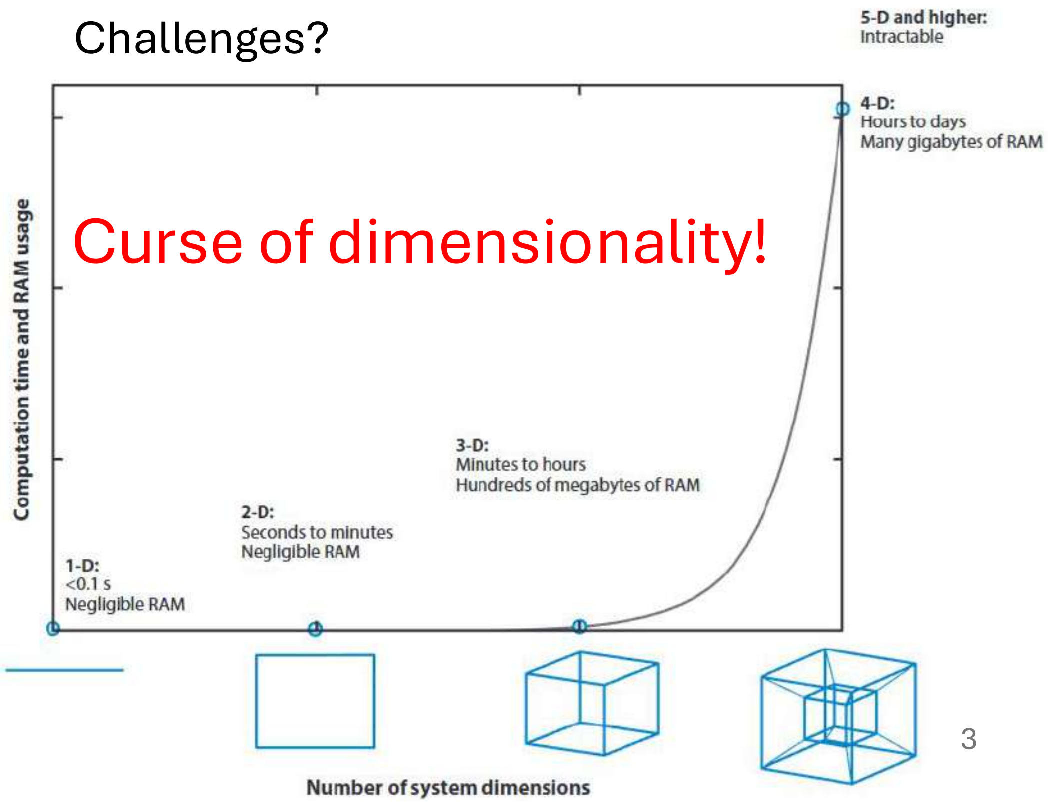


Defender:
- Avoid hitting Ω_{obs}
- Capture attacker

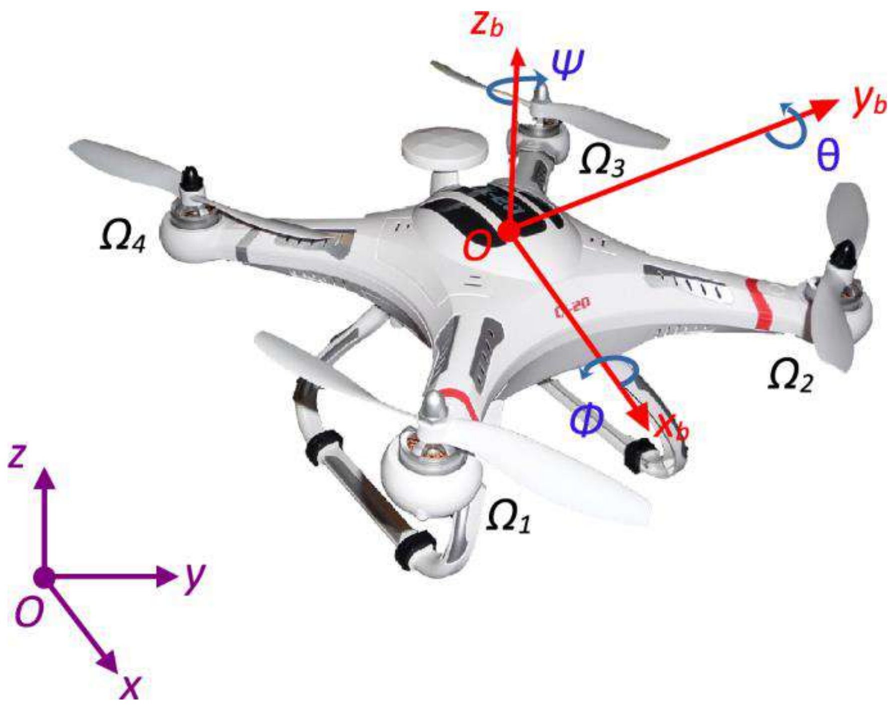
Defender:
- Avoid hitting Ω_{obs}
- Arrive $\mathcal{T}$ before captured

What do we want to know?
- Optimal control strategy for defender
- Optimal control strategy for attacker
- Guaranteed winning conditions for each player

Challenges?



Quadrotors



Basic Dynamics:

$$
\begin{bmatrix} \dot{x} \\ \dot{v}_x \\ \dot{y} \\ \dot{v}_y \\ \dot{z} \\ \dot{v}_z \\ \dot{\omega}_x^b \\ \dot{\omega}_y^b \\ \dot{\omega}_z^b \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}
=
\begin{bmatrix}
v_x \\
\frac{1}{m}\sum_{i=1}^4 T_i\,(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\
v_y \\
\frac{1}{m}\sum_{i=1}^4 T_i\,(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \\
v_z \\
\frac{1}{m}\sum_{i=1}^4 T_i\,(\cos\phi\cos\theta) - g \\
\frac{l}{2I_x}(T_1 - T_2 - T_3 + T_4) + \frac{1}{I_x}\omega_y^b\omega_z^b(I_y - I_z) \\
\frac{l}{2I_y}(-T_1 - T_2 + T_3 + T_4) + \frac{1}{I_y}\omega_x^b\omega_z^b(I_z - I_x) \\
\frac{l}{2I_z}\tau_{\text{drag}}^b + \frac{1}{I_z}\omega_x^b\omega_y^b(I_x - I_y) \\
\omega_x^b + \omega_y^b\sin\phi\tan\theta + \omega_z^b\cos\phi\tan\theta \\
\omega_y^b\cos\phi - \omega_z^b\sin\phi \\
-\omega_y^b\frac{\sin\phi}{\cos\theta} + \omega_z^b\frac{\cos\phi}{\cos\theta}
\end{bmatrix}
$$

X 2

24-dimensional system !
Intractable !!

SFU

National Defence / Défense nationale

Hierarchy Control : Dimensionality Reductions

High-level game modeling

Attacker

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} v_A^{x,C} \\ v_A^{y,C} \\ v_A^{z,C} \end{bmatrix},$$

Defender

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ k_x\left(v_D^{x,C} - v_D^x\right) \\ k_y\left(v_D^{y,C} - v_D^y\right) \\ k_z\left(v_D^{z,C} - v_D^z\right) \end{bmatrix}$$

3D + 6D = 9D

Velocity commands

~50Hz

Low-level system controller

ARDUPILOT Versatile, Trusted, Open

PX4 autopilot

$$\begin{bmatrix} \dot{x} \\ \dot{v_x} \\ \dot{y} \\ \dot{v_y} \\ \dot{z} \\ \dot{v_z} \\ \dot{\omega_x}^{\mathbf{b}} \\ \dot{\omega_y}^{\mathbf{b}} \\ \dot{\omega_z}^{\mathbf{b}} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v_x \\ \frac{1}{m}\sum_{i=1}^{4} T_i\left(cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi)\right) \\ v_y \\ \frac{1}{m}\sum_{i=1}^{4} T_i\left(cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi)\right) \\ v_z \\ \frac{1}{m}\sum_{i=1}^{4} T_i\left(cos(\phi)cos(\theta)\right) - g \\ \frac{l}{2I_x}(T_1 - T_2 - T_3 + T_4) + \frac{1}{I_x}\omega_y^{\mathbf{b}}\omega_z^{\mathbf{b}}(I_y - I_z) \\ \frac{l}{2I_y}(-T_1 - T_2 + T_3 + T_4) + \frac{1}{I_y}\omega_x^{\mathbf{b}}\omega_z^{\mathbf{b}}(I_z - I_x) \\ \frac{l}{2I_z}\tau_{\text{drag}}^{\mathbf{b}} + \frac{1}{I_z}\omega_x^{\mathbf{b}}\omega_y^{\mathbf{b}}(I_x - I_y) \\ \omega_x^{\mathbf{b}} + \omega_y^{\mathbf{b}}sin(\phi)tan(\theta) + \omega_z^{\mathbf{b}}cos(\phi)tan(\theta) \\ \omega_y^{\mathbf{b}}cos(\phi) - \omega_z^{\mathbf{b}}sin(\phi) \\ -\omega_y^{\mathbf{b}}\frac{sin(\phi)}{cos(\theta)} + \omega_z^{\mathbf{b}}\frac{cos(\phi)}{cos(\theta)} \end{bmatrix}$$

Throttles

~500 Hz

Slower attacker with no limits on accelerations

Faster defender but with limits on accelerations

Main focus: controlling the defender

Decomposition

$$\dot{\mathbf{x}}_D = \begin{bmatrix}\dot{\mathbf{p}}_D \\ \dot{\mathbf{v}}_D\end{bmatrix} = \begin{bmatrix}\dot{x}_D \\ \dot{y}_D \\ \dot{z}_D \\ \dot{v}_D^x \\ \dot{v}_D^y \\ \dot{v}_D^z\end{bmatrix} = \begin{bmatrix} v_D^x \\ v_D^y \\ v_D^z \\ k_x(v_D^{x,C} - v_D^x) \\ k_y(v_D^{y,C} - v_D^y) \\ k_z(v_D^{z,C} - v_D^z)\end{bmatrix}, \quad \dot{\mathbf{x}}_A = \dot{\mathbf{p}}_A = \begin{bmatrix}\dot{x}_A \\ \dot{y}_A \\ \dot{z}_A\end{bmatrix} = \begin{bmatrix} v_A^{x,C} \\ v_A^{y,C} \\ v_A^{z,C}\end{bmatrix}$$

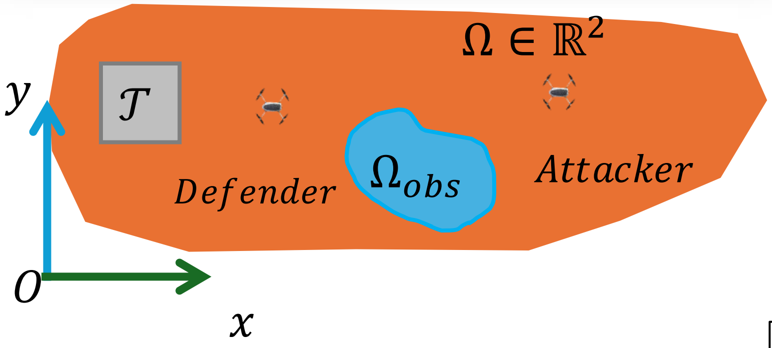
6D + 3D = 9D

National Defence / Défense nationale

Horizontal sub-game

$$\dot{\mathbf{x}}_D^h = \begin{bmatrix}\dot{x}_D \\ \dot{y}_D \\ \dot{v}_D^x \\ \dot{v}_D^y\end{bmatrix} = \begin{bmatrix} v_D^x \\ v_D^y \\ k_x(v_D^{x,C} - v_D^x) \\ k_y(v_D^{y,C} - v_D^y)\end{bmatrix}, \quad \dot{\mathbf{x}}_A^h = \begin{bmatrix}\dot{x}_A \\ \dot{y}_A\end{bmatrix} = \begin{bmatrix} v_A^{x,C} \\ v_A^{y,C}\end{bmatrix}$$

4D + 2D = 6D



Control bounds

$$\sqrt{(v_D^{x,C})^2 + (v_D^{y,C})^2} \leq \mathrm{U}_D^h$$
$$\sqrt{(v_A^{x,C})^2 + (v_A^{y,C})^2} \leq \mathrm{U}_A^h$$

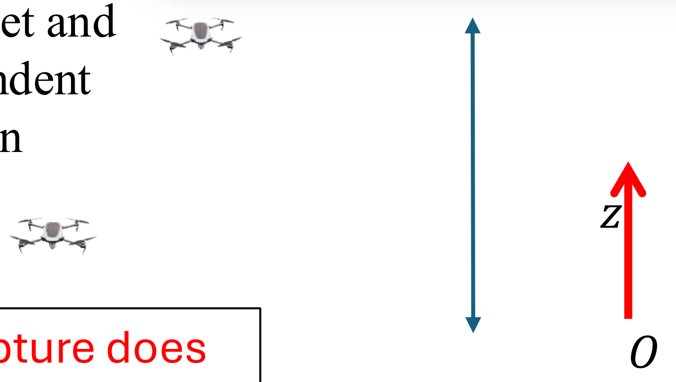
Assumptions: Target set and obstacle are only dependent on horizontal dimension

Not done yet: Horizontal capture does not necessarily happen at the same time as the vertical capture

Vertical sub-game

$$\dot{\mathbf{x}}_D^z = \begin{bmatrix}\dot{z}_D \\ \dot{v}_D^z\end{bmatrix} = \begin{bmatrix} v_D^z \\ k_z(v_D^{z,C} - v_D^z)\end{bmatrix}, \quad \dot{\mathbf{x}}_A^z = \begin{bmatrix}\dot{z}_A\end{bmatrix} = \begin{bmatrix} v_A^{z,C}\end{bmatrix}$$

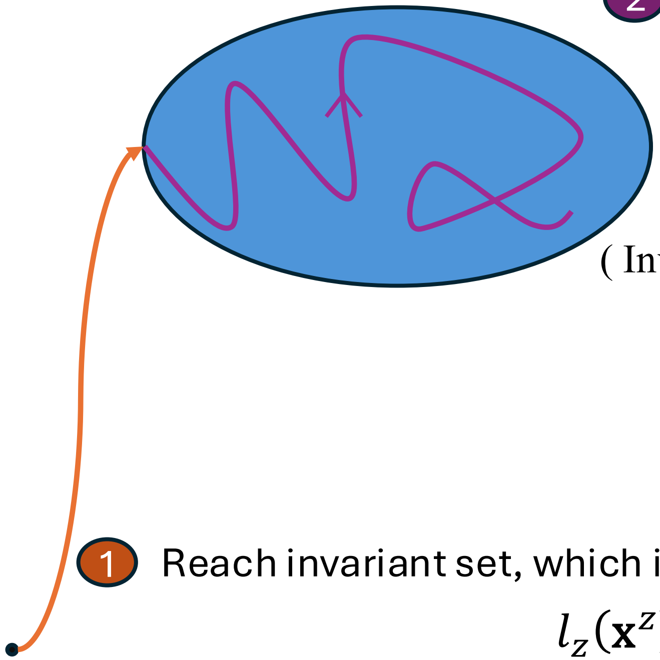
2D + 1D = 3D



$$\left|v_D^{z,C}\right| \leq \mathrm{U}_D^z$$
$$\left|v_A^{z,C}\right| \leq \mathrm{U}_A^z$$

Two-Staged Reach-Track Control

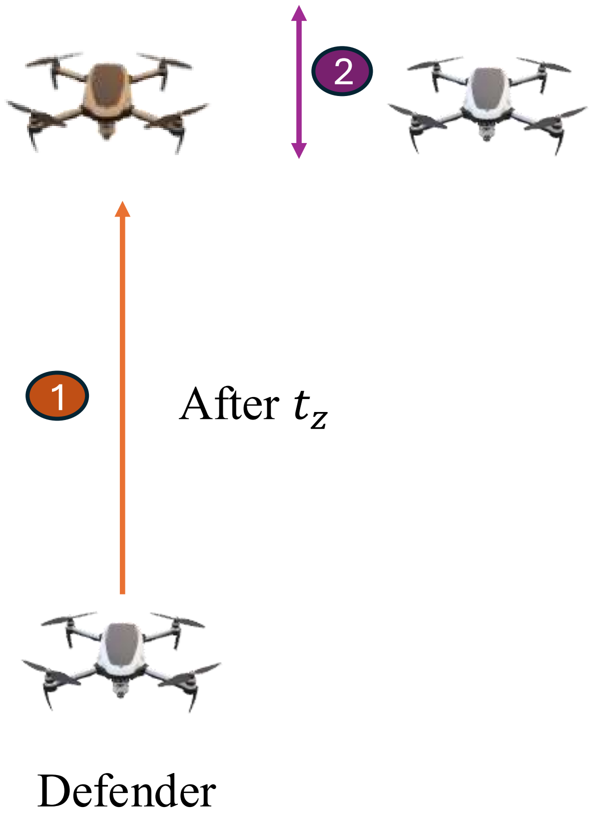
2 Track the attacker inside invariant set



(Invariant capture set)

1 Reach invariant set, which is the zero sub-level set of

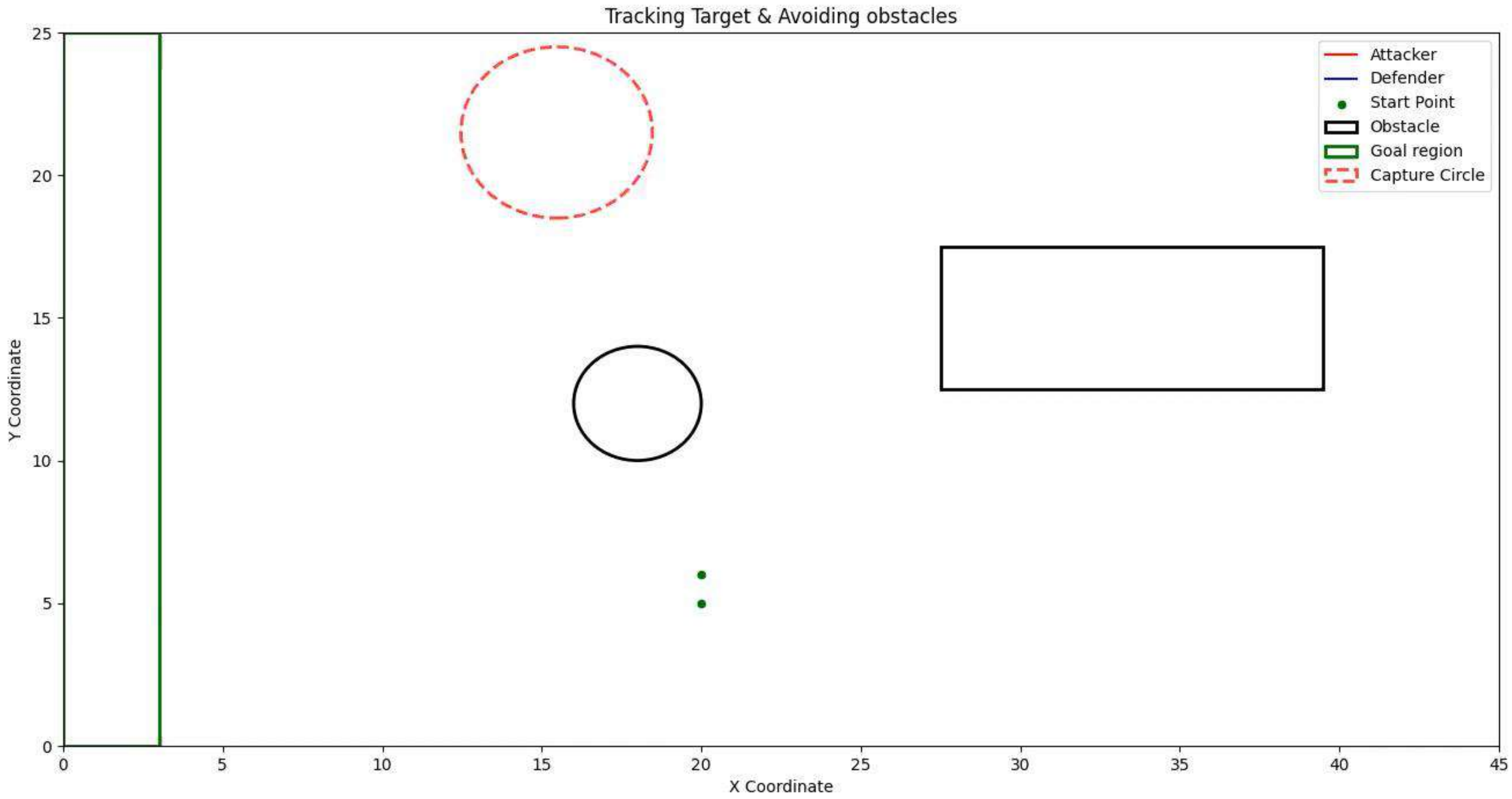
$$l_z(\mathbf{x}^z) = V_{z,\infty}\left(\mathbf{x}_z^{rel}\right) - d_z$$



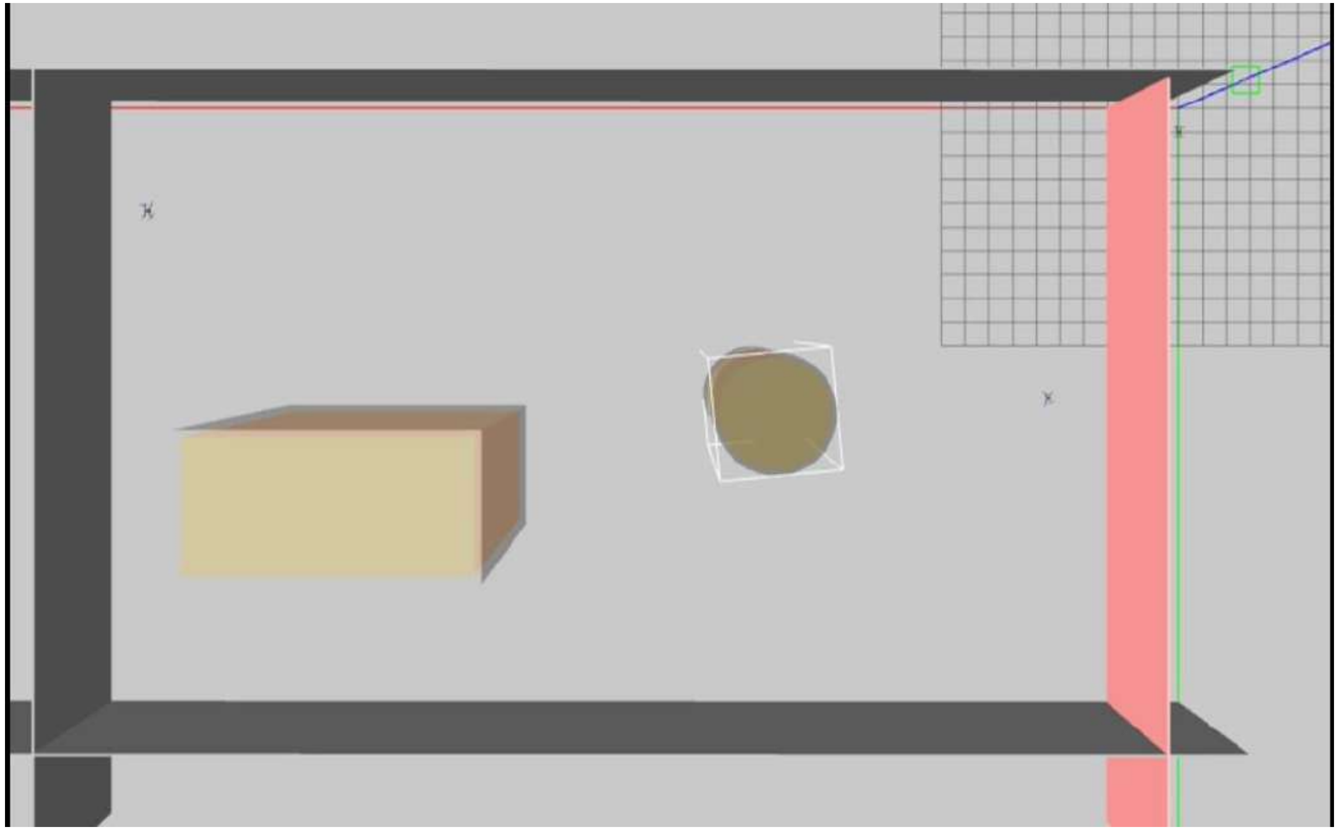
Solution: track as soon as the target is reached in that sub-game while waiting for capture in the other game.

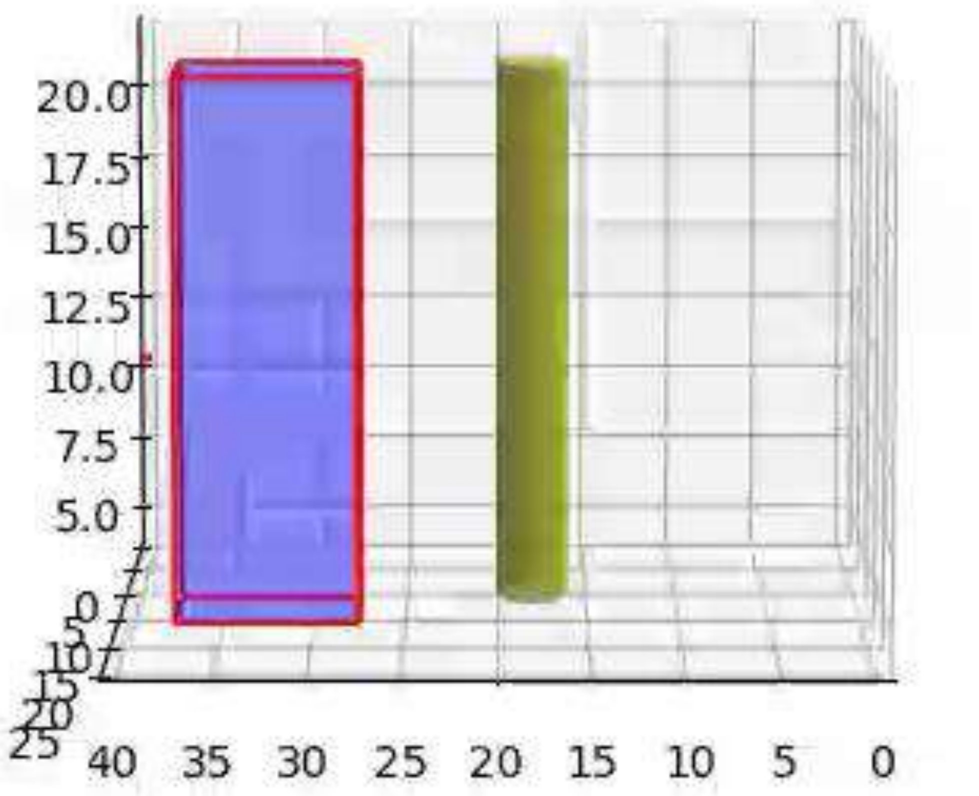
Tracking example

Tracking while avoid obstacles



Experiment in Gazebo





More details in our paper:

Thank you for your listening !

Questions?

Email contact: minh_bui_3@sfu.ca