A Comparison Study of Various Trajectory Optimization Techniques for Multi-Agent Air Hockey

Xiaoyi Cai
Aeronautics and Astronautics Dpt.
Massachusetts Institute of Technology
Cambridge, USA
xyc@mit.edu

Andrea Tagliabue
Aeronautics and Astronautics Dpt.
Massachusetts Institute of Technology
Cambridge, USA
atagliab@mit.edu

Andrew Torgesen
Aeronautics and Astronautics Dpt.
Massachusetts Institute of Technology
Cambridge, USA
torgesen@mit.edu

***Abstract*—A simulation environment implementing stochastic dynamics and event-based collision detection is created to provide a sandbox for comparing and showcasing the effectiveness of several trajectory optimization techniques in facilitating team performance in an adversarial game of multi-agent air hockey. Among the techniques being compared or showcased are a family of open-loop trajectory optimizations featuring linear, nonlinear, and hybrid solvers. In addition to the open-loop strategies, both decentralized and centralized model predictive control (MPC) formulations are presented and tested in the air hockey environment. The effect of utilizing control barrier functions (CBFs) for centralized collision avoidance is also explored. In comparing the performance of the various control schemes, a regularized strategy is used for each team to avoid confounding the effects of strategy choice and control method. Comparison results suggest that there are inherent tradeoffs between scoring ability and robust obstacle avoidance in dynamic, adversarial environments. Additionally, control barrier functions provide a promising convex method for ensuring obstacle avoidance from a centralized planner.**

I. Supplementary Material

All code used for this project can be found at https://github.com/goromal/robo-game-sim. The repository contains a directory called "videos" with select recordings of individual tournament games. A video presentation of our results is available https://youtu.be/3z3LBnkS10M.

II. Introduction

The control of underactuated systems is facilitated by a wide variety of techniques ranging from dynamic programming to ad hoc nonlinear controllers. In the special case of linear underactuated systems, large state spaces can be stabilized by the globally optimal, analytical LQR state space controller that minimizes a quadratic cost function derived from a combination of state and input error. However, when continuous and discrete input and path constraints are imposed on the control formulation, LQR is unable to take them into account, running the risk of resulting in infeasible trajectories. Thus, trajectory optimization transcription techniques must be used in conjunction with mathematical program solvers to obtain a context-specific optimal control strategy. When the imposed constraints are continuous and linear, the solver can exploit the convexity of the mathematical program and obtain a globally optimal trajectory very quickly. The addition of hybrid dynamics, such as those involving collisions, and non-convex constraints necessitate the use of nonlinear and mixed-integer solvers that produce only locally optimal solutions after longer solver runtimes.

Considering the inherent tradeoffs between runtime, optimality, and constraint satisfaction that exists between various trajectory optimization techniques, the overarching goal of this project was to explore those tradeoffs in a dynamic environment with non-convex constraints, multiple agents, and a control objective pertaining to an underactuated system with hybrid dynamics. A game of multi-agent air hockey seemed like a natural fit to provide a sandbox for testing these concepts. The goal, or control objective, of the game is to maneuver the puck into the opponent's goal as many times as possible while preventing the converse to whatever extent possible. Because only the agents on each team are controllable through input commands, the agent-puck system is inherently both an underactuated (autonomous) system and a hybrid system that evolves through phases defined by elastic collisions. Path constraints imposed by the arena limits and saturation on the allowable agent inputs further contributes to the underactuated classification of the multi-agent system.

Besides satisfying the various dynamic and constraint characteristics necessary for evaluating different trajectory optimization formulations, the adversarial nature of the air hockey game provides a natural way to directly compare techniques with easy-to-understand performance indicators. Further, the continuous dynamics are linear and simple enough to not render the prospect of large-scale testing and comparison intractable.

The structure of this project report is as follows: Section III discusses related work that inspired or assisted the implementation of the algorithms in this project. Sections V-VII detail the methods used for the simulation environment as well as trajectory optimization and obstacle avoidance techniques. Results are discussed in Section VIII, and Section IX gives

the conclusion with some final insights.

III. Related Work

The original impetus for this project came from investigating the RoboCup international competition–particularly the model-based winning strategy of the 2000 Cornell team, as outlined in [1]. Due to the complex nature of the game, it is natural to decompose the coordination of the robots into high-level strategic planning and low-level motion planning that enables the former. In [1], the control system is decomposed into three parts, namely the Strategy module, the Trajectory Generation module, and the Local Control module. The Strategy module is a finite state automaton that decides the goals that robots should carry out during the game. These goals are fulfilled by specific plays, such as Offense, Defense or Pass play. The Trajectory Generation module takes the desired final position, velocity and time to target information from the Strategy module to generate feasible trajectories that respect safety, time requirement and energy constraint. At the lowest level, the these desired wheel velocities are achieved by the he Local Control module through local feedback loops. The baseline methods used in this project are most closely related to this formulation.

MPC [2] (and its distributed variant DMPC [3]) is a common framework employed for for dynamic coordination of multiple agents. The work in [4], for example, uses DMPC coordinate 25 UAVs performing a point-to-point transition while avoiding each other, while [5] uses a non-linear MPC for collision avoidance between two aerial vehicles. Hybrid variants such as the one outlined in [6] have shown to be promising for tasks such as controlling an underactuated puck through collisions or utilizing arena walls.

Works such as [7] have made use of control barrier functions (CBF) [8] to minimally perturb independently derived trajectories for centralized obstacle avoidance in a multi-agent setting with a convex optimization problem. This approach is attractive due to its convexity and ability to guarantee obstacle avoidance. However, there are certainly questions about how such minimal perturbations to individual agent trajectories serve to derail the original trajectory path objectives in a dynamic environment.

Methods for event-based collision detection are outlined in large-scale simulation environment architectures like [9], where it is necessary to enforce multibody and environmental collisions in an efficient and methodical manner. This project aims to implement similar collision detection algorithms to maximize the fidelity of the air hockey simulation, which depends largely on collision dynamics to achieve control and game objectives.

IV. Notations

Several parameters common to all control schemes include the desired final state $\boldsymbol{x}_{\text{des}}$, arena height and width bounds $\boldsymbol{p}_{\text{max}} = [h, w]$, $\boldsymbol{p}_{\text{min}} = -\boldsymbol{p}_{\text{max}}$, and input saturation limit $\boldsymbol{u}_{\text{max}}, \boldsymbol{u}_{\text{min}}$. We denote the discrete time-step with k, the number of time-steps with N. Puck and player have, respectively, radius r and R, and mass m_{puck} and m_{player}, and a time-constant of τ_{player} and τ_{puck}. We denote all the vectors in bold and the matrices in capital. For convenience, the notation is summarized in Table I. Other notations are introduced later when necessary.

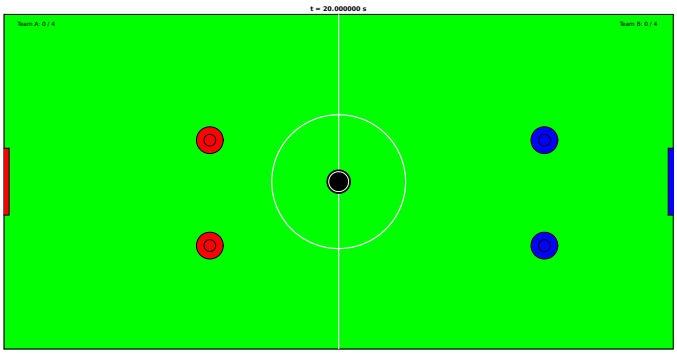

Fig. 1: Multi-agent air hockey simulation environment. The geometry of the puck, the agents, the goals, and the arena is depicted to scale. Each agent is governed by its own set of linear dynamics, and event-based collision detection enforces elastic collisions between agents, other agents, and the walls of the arena.

Symbol	Meaning	Domain
$\boldsymbol{x}_i$	Position and velocity of player i	$\mathbb{R}^4$
$\boldsymbol{x}_{\text{puck}}$	Position and velocity of puck	$\mathbb{R}^4$
$\boldsymbol{p}_i$	Position of player i	$\mathbb{R}^2$
$\boldsymbol{v}_i$	Velocity of player i	$\mathbb{R}^2$
$\boldsymbol{p}_{\text{puck}}$	Position of puck	$\mathbb{R}^2$
$\boldsymbol{v}_{\text{puck}}$	Velocity of puck	$\mathbb{R}^2$
τ_{puck}	Time constant of puck	$\mathbb{R}$
τ_{player}	Time constant of player	$\mathbb{R}$
r	Radius of the puck	$\mathbb{R}$
R	Radius of a player	$\mathbb{R}$
N	Number of time-steps	$\mathbb{N}$
m_{player}	Mass of player i	$\mathbb{R}$
m_{puck}	Mass of the puck	$\mathbb{R}$

TABLE I: Symbols used for this work

V. Simulation Environment

To provide a self-contained interface for enforcing the continuous dynamics, collisions, and rules associated with the multi-agent air hockey game, a simulation environment was written in C++ with Python bindings. Fig. 1 depicts the geometry of the simulation environment. The simulator provides an interface where agent inputs can be given by an external program and the simulator returns the evolved state of every player and puck after a time step of Δt.

The version of air hockey enforced by the simulator differs from a traditional air hockey game in that there is significantly more damping, the effective damping coefficient is allowed to be different for the puck than for the players, and random noise is added to the input for each player. The linear dynamics governing player movement can be expressed as:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{1}{\tau_{\text{player}}} & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_{\text{player}}} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{\tau_{\text{player}}} & 0 \\ 0 & \frac{1}{\tau_{\text{player}}} \end{bmatrix} \boldsymbol{u} + \boldsymbol{\nu} \quad (1)$$

where $\boldsymbol{x} = \begin{bmatrix} p_x & p_y & v_x & v_y \end{bmatrix}^T$ consists the position and velocity of the player in the plane, τ_{player} is a player-specific parameter related to rise time, $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \in \mathbb{R}^4$ is Gaussian noise acting on each velocity component, and $\boldsymbol{u} = \begin{bmatrix} u_x & u_y \end{bmatrix}^T$ can be conceptualized either as a desired input velocity to motors with first-order dynamics or as input forces to a damped mechanical system. Similarly, the dynamics for the puck (which has no internal input) are:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{1}{\tau_{\text{puck}}} & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_{\text{puck}}} \end{bmatrix} \boldsymbol{x} \quad (2)$$

Equations (1) and (2) are integrated forward in time using fourth-order Runge-Kutta integration. The scoring rules are identical to normal air hockey, and the puck's position is reset to the arena origin each time it enters a goal. Inter-body collisions and collisions with the arena walls are enforced using a version of event-based collision detection, which aims to predict when collisions will occur and preemptively handle them rather than simply check for collisions that have already occurred and backtrack to handle them after the fact.

Though the governing dynamics of each agent in the game are linear, the resulting trajectories are nonlinear in shape, making the prospect of analytically solving for future collision times non-trivial. Thus, at each time step, the time interval is divided further into a dense grid of predicted states for each moving body. Each node in the grid is populated using Runge-Kutta integration, and linear interpolation is used between each grid point to allow for a pseudo-analytical calculation of the future collision times. A priority queue is then used to handle each collision in the order of imminence, and new imminent collisions are added to the queue as they arise. The full collision detection and handling strategy is outlined by Algorithm 1.

For collisions between players, the simulator logs a "damage coefficient" proportional to the squared induced velocity change imposed by one player on the other. A matrix of these damage coefficients is maintained throughout the game, providing a measure of how well each player is able to avoid colliding with each other player in the arena. At the end of each game, the Python interface to the C++ simulator allows the user to log the results of each game for subsequent visualization and analysis before resetting the simulator for another game. The flexible interface for the simulator allows for the quick development and testing of numerous control strategies using Drake's Python interface classes.

Algorithm 1 Event-Based Collision Handling

Input: $p_i(0)$, $v_i(0)$, u_i Initial position, velocity, input for each body i

Output: $p_i(\Delta t)$, $v_i(\Delta t)$ Resulting position/velocity at end of simulation window

Initialize $t_{\text{grid}} = \{t_k\}$ with 50 time points between $t(0)$ and $t(0) + \Delta t$
Initialize $x_{\text{grid},i} = \{x_{k,i}\}$ with each $x_{k,i}$ calculated from integrating Equation STATE **Initialize** $k_{\text{base}} = 0$ 1 or 2 with RK4
Initialize $\mathcal{P} = \varnothing$ Priority queue for sorting collisions by calculated time
do
  if $\mathcal{P} \neq \varnothing$ **then**
    (t_p, p, i_1, i_2) = remove top time-stamped collision element involving i_1, i_2 (i_2 only if multi-body collision) from $\mathcal{P}$
    Simulate collision p and propagate effects to x_{k,i_1} and x_{k,i_2} for all $k > t_p$
    Set k_{base} to greatest $k < t_p$
    $\mathcal{C} = \{i_1, i_2\}$
  else
    $\mathcal{C} = \{i\}$ for all i
  end if
  for c in $\mathcal{C}$ **do**
    for $x_{k,c} \in x_{\text{grid,c}},\ k \geq k_{\text{base}}$ **do**
      Use inter-grid linear interpolation to find pseudo-analytical time to collision for each possible collision involving c
      Add time-stamped collision to $\mathcal{P}$ if time to collision $< \Delta t$
    end for
  end for
while $\mathcal{P} \neq \varnothing$

VI. Trajectory Optimization Methods

To avoid confounding the comparison between different control strategies, each team follows an identical, relatively simple strategy, where the current play is either an *offensive* or *defensive* play and each team has a player primarily concerned with offense and defense. The team-level strategy can be described by the logic trees depicted in Fig. 2.

The end result of the strategy is that at each time step, each player is given a desired state $\boldsymbol{x}_{\text{des}}$ to arrive at as quickly as possible, and the following sections detail the various implemented methods used to control each player to achieve an often constantly evolving $\boldsymbol{x}_{\text{des}}$ in a dynamic environment with arena constraints and moving obstacles.

A. Baseline Program

The baseline approach is loosely inspired by early work on the Robot-cup challenge [1], which served as a benchmark and the starting point for the more advanced algorithms presented in the next sections. In the baseline strategy, each player

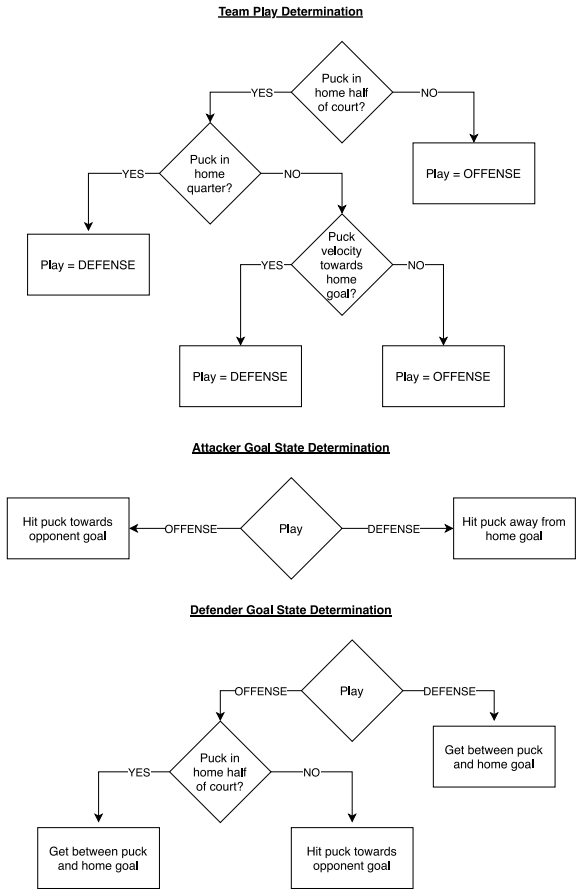

Fig. 2: The common logic dictating the strategy of each team.

has three actions available: *kick*, *defend*, and *defensive_kick*. Direct Collocation for the system in (1) is used to compute minimum trajectory for these actions, where the terminal time step is denoted as N. The baseline strategy is open-loop, i.e., new trajectories are planned only after full execution. During trajectory planning, the puck is assumed to be static for simplicity.

a) Kick action: The player generates a trajectory from its current position to hit the puck towards the goal. We achieve a directional kicking at the final time step N by specifying the terminal velocity for player i as

$$\boldsymbol{v}_{\text{des},i} = \boldsymbol{v}_i[N] = v_{\text{kick}} \frac{\boldsymbol{p}_{\text{goal}} - \boldsymbol{p}_{\text{puck}}}{||\boldsymbol{p}_{\text{goal}} - \boldsymbol{p}_{\text{puck}}||}$$

where $v_{\text{kick}} > 0$ determines the aggressiveness of the kick. The desired final position to kick is the position of the puck shifted along the kicking direction by the sum of the radius of the puck and of the player

$$\boldsymbol{p}_{\text{des},i} = \boldsymbol{p}_i[N] = \boldsymbol{p}_{\text{puck}} - (r + R) \frac{\boldsymbol{v}_i[N]}{||\boldsymbol{v}_i[N]||} \tag{3}$$

Further, at all knot points k of the planned trajectory, the player has to remain in the arena (defined by $\boldsymbol{p}_{\min}, \boldsymbol{p}_{\max} \in \mathbb{R}^2$)

$$\boldsymbol{p}_{\min} \leq \boldsymbol{p}_i[k] \leq \boldsymbol{p}_{\max} \tag{4}$$

and to respect the actuation limits

$$-\boldsymbol{u}_{\min} \leq \boldsymbol{u}_i[k] \leq \boldsymbol{u}_{\max}. \tag{5}$$

To achieve minimum-time behavior, we define the cost function to be

$$\boldsymbol{J}_{\text{kick},i} = T + (\boldsymbol{x}_{\text{des},i} - \boldsymbol{x}_i[N])^T \boldsymbol{Q} (\boldsymbol{x}_{\text{des},i} - \boldsymbol{x}_i[N]) \tag{6}$$

where $\boldsymbol{Q} \in \mathbb{R}^{4\times4}$ is a positive definite matrix.

b) Defend action: The player i tries to stay between the home goal and the puck by planning a minimum trajectory that goes to

$$\boldsymbol{p}_{\text{des},i} = \boldsymbol{p}_{\text{home-goal}} + (\boldsymbol{p}_{\text{puck}}[k] - \boldsymbol{p}_{\text{home-goal}})/2.$$

c) Defensive kick action: The formulation is similar to the *kick action*, but the player i tries to kick the puck in a direction towards the adversary's goal, but along a trajectory that does not immediately intersect any obstacle (any other player). We set the desired final velocity to be

$$\boldsymbol{v}_{\text{des},i} = \begin{bmatrix} -\text{sign}(p_{\text{goal},x}) \\ \text{sign}(p_{\text{opponent(i,1),y}} + p_{\text{opponent(i,2),y}}) \end{bmatrix} \tag{7}$$

where $-\text{sign}(\boldsymbol{p}_{\text{goal},x})$ denotes the x-direction of the adversary's, while $p_{\text{opponent(i,1),y}}$ and $p_{\text{opponent(i,2),y}}$ denote the y-coordinated of the two opponents of player i. while the final position is obtained according to Equation (3).

d) Other actions and what did not work: During the development phase of the baseline strategy we tested multiple different approaches which eventually did not make it in the final version of the strategy. These approaches included a fixed-time kicking mode, and a mixed-integer formulation for avoidance of the opponents. While the first formulation did not provide to be very useful (since there is little gain in moving at a non-maximum velocity and acceleration), the second strategy did not prove to be very practical, since the Branch-and-Bound solver usually took too long to complete (more than 20 minutes for a single iteration, using 80 integers variables).

B. Decentralized MPC

The decentralized model predictive control (DMPC) control strategy has each individual agent formulating and solving its own constrained nonlinear optimization program at each time step. The nonlinear program is built on top of a direct collocation scheme for (1) with a fixed number of time steps N but variable time step length. The cost function for player i is the same as (6) which encourages the robot to move to desired terminal state as quickly as possible.

Similar to the baseline, the player has to remain in the arena and respect the actuation limits by enforcing the constraints (4) and (5). In addition, collision avoidance constraint

$$(\boldsymbol{p}_i[k] - \boldsymbol{o}_j)^T (\boldsymbol{p}_i[k] - \boldsymbol{o}_j) \geq 4R^2 \tag{8}$$

for all knot points k and obstacles $\boldsymbol{o}_j$. The program is subsequently solved by SNOPT, and only the first time step of the input solution is extracted and returned to the simulator. The entirety of the state and input solutions $\boldsymbol{x_i}[k], \boldsymbol{u_i}[k]$ is

stored to be used as the initial guess for the next optimization, which greatly decreases the runtime of subsequent solutions.

The practice of only executing $\boldsymbol{u_i}[0]$ after each solution, a characteristic of MPC methods, gives DMPC a feedback control quality that provides adaptability in the dynamic air hockey environment.

C. Centralized Non-Linear MPC

The centralized non-linear MPC strategy is based on the strategy proposed in Part VI-B, with the difference that the optimization problem is formulated and solved jointly for the two players. This implies that the state of program contains the position of both the players $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. Thanks to this formulation, we can enforce that the trajectories generated by the two players belonging to the same team should not be in collision with each other. Reciprocal collision avoidance is achieved by introducing the following quadratic constraint

$$(\boldsymbol{p}_1[k] - \boldsymbol{p}_2[k])^T(\boldsymbol{p}_1[k] - \boldsymbol{p}_2[k]) \geq 4R^2 \tag{9}$$

for every knot point k. The additional constraint does not fundamentally change the formulation of the problem and, as for the DMPC case, the non-linear program is solved using SNOPT.

D. MIQP-like Non-Linear Program for Centralized Planning

In this formulation we try to plan jointly a dynamic trajectory for the puck and a player, without having to pre-specify the impact time, position or velocity of the player with the puck. The objective of this formulation is to evaluate a more "holistic" approach to planning and trajectory generation, by letting the solver figure out every aspect of the sequence of actions that the player has to follow to score. The approach, unfortunately, did not work as expected and is not included in our results, but its formulation is here presented for completeness.

As an initial step, we focus our formulation for the joint planning of the trajectory of one player and the puck. We set the cost function to simply represent the high-level objective of having the puck in the goal (by minimizing its distance from the goal)

$$J_{\text{joint}} = \sum_{k=1}^{N} (\boldsymbol{p}_{\text{goal}}[k] - \boldsymbol{p}_{\text{puck}}[k])^T \boldsymbol{Q}_1 (\boldsymbol{p}_{\text{goal}}[k] - \boldsymbol{p}_{\text{puck}}[k]) \tag{10}$$

and we introduce an additional small penalty on the distance between the agent and the puck

$$J_{\text{joint}}^{\text{optional}} = \sum_{k=1}^{N} (\boldsymbol{p}_{\text{puck}} - \boldsymbol{p}_i[k])^T \boldsymbol{Q}_2 (\boldsymbol{p}_{\text{puck}} - \boldsymbol{p}_i[k]) \tag{11}$$

to aid the solver in finding a feasible solution. $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2 \in \mathbb{R}^{2\times 2}$ are positive definite matrices, tuning parameter of the approach.

Following [10] we choose a mixed-integer-like approach to the problem, since it allows the flexibility to let the solver chose impact time - or no impact at all - which can be convenient in the case we are trying to control multiple agents (ideally the solver will even choose which agent should hit the puck). To this end, we introduce two new variables. First, we define a binary variable $t_{\text{kick},i}[k] \in \{0, 1\}$ for $k = 1, \ldots, N$, which is 1 if player i is in contact with the puck at time k, and 0 otherwise. Second, we introduce the contact force between the puck and player i as $\boldsymbol{\lambda}_i[k] \in \mathbb{R}^2$, with $k = 1, \ldots, N$. We use the guard/reset map approach to switch in between the two possible modes for player i: in contact with the puck ($\boldsymbol{\lambda}_i \neq \boldsymbol{0}$) or not ($\boldsymbol{\lambda}_i = \boldsymbol{0}$). The guard corresponds to the squared distance between player and puck:

$$\Phi[k] = (\boldsymbol{p}_i[k] - \boldsymbol{p}_{\text{puck}}[k])^T (\boldsymbol{p}_1[k] - \boldsymbol{p}_{\text{puck}}[k]) - (r + R)^2 \tag{12}$$

which triggers the change between contact/not contact mode of $t_{\text{kick},i}[k]$ via a constrain formulated using the big-M notation:

$$\Phi[k] \leq M(1 - t_{\text{kick},i}[k]) \tag{13}$$

where $M \in \mathbb{R}$ is a large, positive constant. The value of the contact force $\boldsymbol{\lambda}_i[k]$ is enforced to be $\boldsymbol{0}$ when player and puck are not in contact via the big-M notation, according to the constraint

$$-M(1 - t_{\text{kick},i}[k]) \leq \boldsymbol{\lambda}_i[k] \leq M(1 - t_{\text{kick},i}[k]) \tag{14}$$

for every $k = 1, \ldots, N$. We assume the collision between puck and player to be perfectly elastic. By applying the principle of conservation of the kinetic energy, we obtain the reset map

$$[\boldsymbol{x}_i^+[k], \boldsymbol{x}_{\text{puck}}^+[k]] = \boldsymbol{\Delta}(\boldsymbol{x}_i^-[k], \boldsymbol{x}_{\text{puck}}^-[k]), \tag{15}$$

defined as

$$\begin{cases} \boldsymbol{p}_i^+[k] = \boldsymbol{p}_i^-[k], \\ \boldsymbol{v}_i^+[k] = \boldsymbol{v}_i^-[k] \\ \boldsymbol{p}_{\text{puck}}^+[k] = \boldsymbol{p}_{\text{puck}}^-[k], \\ \boldsymbol{v}_{\text{puck}}^+[k] = \boldsymbol{v}_i^-[k] \end{cases}$$

where, for simplicity, we have assumed that the mass of the player $>>$ than the mass of the puck so that, after the impact, the puck will acquire the same velocity as the player, while the velocity of the player will not be affected. The reset map is enforced at impact time via the big-M notation

$$-M(1 - t_{\text{kick},i}[k]) \leq \boldsymbol{\Delta}(\boldsymbol{x}_i^-, \boldsymbol{x}_{\text{puck}}^-) \leq M(1 - t_{\text{kick},i}[k]) \tag{16}$$

We apply the contact force lambda as an additional acceleration term to the dynamics of the two bodies by introducing a new input in their respective discrete-time state space representation (discretized using forward Euler with the fixed sampling time h), obtaining, for player i

$$\boldsymbol{x}_i[k+1] = \boldsymbol{A}_i \boldsymbol{x}_i[k] + \boldsymbol{B}_i \boldsymbol{u}_i[k] + \boldsymbol{E}_i \boldsymbol{\lambda}_i[k] \tag{17}$$

$$\boldsymbol{E}_i = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ h/m_{\text{player}} & 0 \\ 0 & h/m_{\text{player}} \end{bmatrix} \tag{18}$$

and, for the puck

$$\boldsymbol{x}_{\text{puck}}[k+1] = \boldsymbol{A}_{\text{puck}}\boldsymbol{x}[k] - \boldsymbol{E}_{\text{puck}}\boldsymbol{\lambda}_i[k] \tag{19}$$

$$\boldsymbol{E}_{\text{puck}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ h/m_{\text{puck}} & 0 \\ 0 & h/m_{\text{puck}} \end{bmatrix} \tag{20}$$

where the definition of $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{A}_{\text{puck}}$ can be obtained by discretizing equation eq. (1) and eq. (2). We observe that the problem here formulated is a Mixed-Integer Quadratic Program (quadratic due to the presence of the constraint Φ on the distance between puck and player). We solve it using SNOP, defining $t_{\text{kick},i}[k]$ as a continuous variable having only values 0 or 1, thanks to the additional smooth constraint

$$t_{\text{kick},i}[k](1 - t_{\text{kick},i}[k]) = 0 \tag{21}$$

As before, we additionally limit the maximum control input for each player using the constraint defined in (5), and we constraint the player and the puck to be inside the arena, following what done in (4).

E. Bounce Kick

Inspired by the bouncing basketball example in the textbook, we have also devised a special trick for a player to score points by bouncing the puck off of the arena wall towards the opponents' goal. To simplify the planning process, we have taken a decoupled approach, where the initial velocity of the puck is first calculated, and then the player will plan a timed trajectory that enforces this desired velocity for the puck through an elastic collision constraint. In the following discussion, we assume that the puck is initially stationary, collisions are elastic, and τ_{puck} is large enough such that the puck can slide for longer distance.

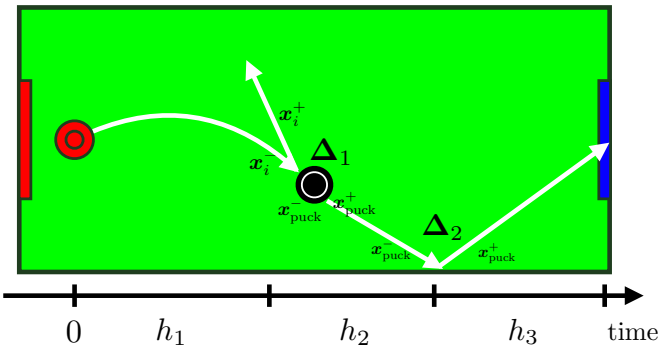

Fig. 3: Illustration of the bounce kick trajectory planning, with reset maps, player and puck states, and the time periods between reset maps.

The entire maneuver is separated into three time periods, as visualized in Fig. 3, where $[0, h_1]$ (provided by the user) is when the player moves to hit the puck, $[h_1, h_1 + h_2]$ is when the puck moves towards the designated wall, and $[h_1 + h_2, h_1 + h_2 + h_3]$ is when the puck bounces off of the wall and reaches the goal. As a result, we define two reset maps $\boldsymbol{\Delta}_1$ and $\boldsymbol{\Delta}_2$. The first reset map captures the elastic collisions between the player and the puck, where

$$[\boldsymbol{x}_i^+, \boldsymbol{x}_{\text{puck}}^+] = \boldsymbol{\Delta}_1(\boldsymbol{x}_i^-, \boldsymbol{x}_{\text{puck}}^-), \tag{22}$$

such that

$$\begin{cases} \boldsymbol{p}_i^+ = \boldsymbol{p}_i^-, \\ \boldsymbol{v}_i^+ = \boldsymbol{v}_i^- - \frac{2m_{\text{puck}}}{m_{\text{player}}+m_{\text{puck}}} \frac{\langle \boldsymbol{v}_i^- - \boldsymbol{v}_{\text{puck}}^-, \boldsymbol{p}_i^- - \boldsymbol{p}_{\text{puck}}^- \rangle}{\|\boldsymbol{p}_i^- - \boldsymbol{p}_{\text{puck}}^-\|^2} (\boldsymbol{p}_i^- - \boldsymbol{p}_{\text{puck}}^-), \\ \boldsymbol{p}_{\text{puck}}^+ = \boldsymbol{p}_{\text{puck}}^-, \\ \boldsymbol{v}_{\text{puck}}^+ = \boldsymbol{v}_{\text{puck}}^- - \frac{2m_i}{m_{\text{puck}}+m_{\text{player}}} \frac{\langle \boldsymbol{v}_{\text{puck}}^- - \boldsymbol{v}_i^-, \boldsymbol{p}_{\text{puck}}^- - \boldsymbol{p}_i^- \rangle}{\|\boldsymbol{p}_{\text{puck}}^- - \boldsymbol{p}_i^-\|^2} (\boldsymbol{p}_{\text{puck}}^- - \boldsymbol{p}_i^-). \end{cases}$$

The second reset map is required for the puck and upper or lower wall collision, which has the form:

$$\boldsymbol{x}_{\text{puck}}^+ = \boldsymbol{\Delta}_2(\boldsymbol{x}_{\text{puck}}^-) = [p_{x,\text{puck}}^-, p_{y,\text{puck}}^-, v_{x,\text{puck}}^-, -v_{y,\text{puck}}^-]. \tag{23}$$

Note that the reset maps (22) and (23) are only activated when the player and the puck, or the puck and the wall are in contact.

Next, we notice that the state evolution of the puck can be solved in closed form after colliding with the player. During $[h_1, h_2]$ and $[h_2, h_3]$, we can integrate the dynamics of the puck and get:

$$\begin{aligned} \boldsymbol{p}_{\text{puck}}(t_0 + h) &= \boldsymbol{p}_{\text{puck}}(t_0) + \left(1 - \exp\left(\frac{-h}{\tau_{\text{puck}}}\right)\right) \tau_{\text{puck}} \boldsymbol{v}_{\text{puck}}(t_0), \\ \boldsymbol{v}_{\text{puck}}(t_0 + h) &= \boldsymbol{v}_{\text{puck}}(t_0) \exp\left(\frac{-h}{\tau_{\text{puck}}}\right), \end{aligned} \tag{24}$$

where t_0 is either h_1 or $h_1 + h_2$.

With the reset maps and the closed solutions for the puck state, we now show the strategy on a high level.

- **Puck bouncing off of wall:** The decision variables are h_2, h_3 and the initial velocity of the puck $\boldsymbol{v}_{\text{init}}$ such that the puck can hit the wall after h_2 seconds and enters the goal after another h_3 seconds. These variables can be solved by formulating the following through Drake:

$$\begin{aligned} \min_{h_2, h_3, \boldsymbol{v}_{\text{init}}} \quad & 0 \\ \text{s.t.} \quad & t = h_1: \text{ initial condition of the puck,} \\ & t = h_1 + h_2: \text{ puck is at the designated wall,} \\ & \quad \text{based on the closed form solution (24),} \\ & t = h_1 + h_2: \text{ reset map in (23),} \\ & t = h_1 + h_2 + h_3: \text{ the puck is in the goal} \\ & \quad \text{based on the closed form solution (24),} \\ & h_2 + h_3 \leq \text{designated time.} \end{aligned} \tag{25}$$

- **Player hits the puck:** Given the desired initial velocity for the puck $\boldsymbol{v}_{\text{init}}$, solving for a trajectory for the player during time $[0, h_1]$ becomes easy. Through a direct transcription program that encodes the dynamics of the player, we want to solve for control inputs that drive the player to hit the puck at terminal time h_1 (user specified), while enforcing the desired initial velocity for the puck $\boldsymbol{v}_{\text{init}}$ through the reset map in (22).

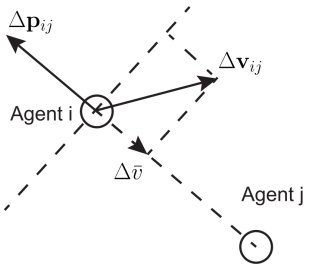

Fig. 4: Relative position and velocity between agent i and j, and the normal component $\Delta \hat{v}$ that contributes to the collision of two agents. Image source: [11].

The decoupled approach taken to achieve bounce kick works well, but the optimization framework in Drake can afford jointly solving for the trajectories for the player and the puck, by encoding the problem as a mixed integer program.

VII. Centralized Collision Avoidance

Control Barrier Functions (CBFs) [8] are Lyapunov-like functions that can be used to ensure safety of dynamical systems, which can be framed as the forward invariance property of a subset of the state space, i.e., if a system starts in the safe set, it remains in the safe set for all future time. Prior works have applied CBFs for systems with single integrator [7] or double integrator dynamics [11], but not for the damped double integrator dynamics considered in our report.

Due to the limited space, the definition for CBF and the theorem that provide guarantees for the forward invariance of safe sets are included in Appendix A. Next, we derive the barrier function for our problem and formulate an optimization problem that minimally changes the nominal control inputs of robots in order to guarantee collision avoidance.

A. Problem Formulation

In this section, we formulate control barrier functions and corresponding constraints that are used to guarantee inter-robot collision avoidance in the air hockey scenario. Following the notations in [11], we let $\mathcal{M} = \{1, 2, \ldots, M\}$ be the set of M robots. Rewriting the each player's dynamics (1) in a more compact form, we get

$$\begin{bmatrix} \dot{\boldsymbol{p}}_i \\ \dot{\boldsymbol{v}}_i \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{I} \\ 0 & \frac{-1}{\tau_{\text{player}}}\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_i \\ \boldsymbol{v}_i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{\tau_{\text{player}}}\boldsymbol{I} \end{bmatrix} \boldsymbol{u}_i, \quad \forall i \in \mathcal{M}$$

where $\boldsymbol{I}$ is an identity matrix, $\boldsymbol{p}_i,\ \boldsymbol{v}_i \in \mathbb{R}^2$ are the planar position and velocity of the i-th robot, and $\boldsymbol{u}_i \in \mathbb{R}^2$ is the corresponding control input. As shown in Fig. 4, we denote relative position and velocity between robots i and j as

$$\Delta \boldsymbol{p}_{ij} = \boldsymbol{p}_i - \boldsymbol{p}_j \quad \text{and} \quad \Delta \boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j,$$

as well as the normal component of the relative velocity as

$$\Delta \bar{v} = \frac{\Delta \boldsymbol{p}_{ij}^T}{\|\Delta \boldsymbol{p}_{ij}\|} \Delta \boldsymbol{v}_{ij} \in \mathbb{R},$$

which changes relative distance between robots i and j and may lead to collisions.

Consider the scenario where two robots are moving towards each other ($\Delta \bar{v} < 0$), and they collaboratively decelerate in the relative directions with maximum relative commanded acceleration:

$$\Delta a_{\max} = 2u_{\max} \in \mathbb{R}.$$

To formulate the safety constraint, we need to know he minimum braking time T_b for them to reduce $\Delta \bar{v}$ to 0. First, we can write the dynamics for $\Delta \bar{v}$ as

$$\Delta \dot{\bar{v}} = \frac{-1}{\tau_{\text{player}}}(\Delta \bar{v} - \Delta a_{\max}),$$

and the evolution of $\Delta \bar{v}$ follows

$$\Delta \bar{v}(t) = \Delta a_{\max} + (\Delta \bar{v}(0) - \Delta a_{\max}) \exp(\frac{-t}{\tau_{\text{player}}}).$$

Given the relative velocity $\Delta \bar{v}(0) < 0$ (robots moving towards each other) at the current time instance $t = 0$, it takes T_b to reach $\Delta \bar{v}(T_b) = 0$, where

$$T_b = -\tau_{\text{player}} \ln \left(\frac{\Delta a_{\max}}{\Delta a_{\max} - \Delta \bar{v}(0)} \right) \geq 0.$$

Therefore, the safety constraint for two robots to remain at least $D_s > 0$ distance apart can be formulated as

$$\begin{aligned} &\|\Delta \boldsymbol{p}_{ij}\| + \int_0^{T_b} \Delta \bar{v}(t)\ \mathrm{d}t \geq D_s, \quad \forall i \neq j, \\ \implies &\|\Delta \boldsymbol{p}_{ij}\| + T_b \Delta a_{\max} + \tau_{\text{player}} \Delta \bar{v}(0) \geq D_s, \quad \forall i \neq j. \end{aligned} \tag{26}$$

Plugging (VII-A) into (26), we can formulate h_{ij} whose 0-level set encodes the safe states for robots i and j as

$$\begin{aligned} h_{ij}(\Delta \boldsymbol{p}_{ij}, \Delta \boldsymbol{v}_{ij}) &= (\Delta a_{\max} - \Delta \bar{v}(0)) \\ &\cdot \exp\left(\frac{\|\Delta \boldsymbol{p}_{ij}\| + \tau_{\text{player}} \Delta \bar{v}(0) - D_s}{\tau_{\text{player}} \Delta a_{\max}} \right) - \Delta a_{\max}. \end{aligned} \tag{27}$$

Given an extended class $\mathcal{K}_\infty$ function $\alpha : \mathbb{R} \to \mathbb{R}$ (for example, $\alpha(r) = r$), h_{ij} is a valid control barrier function if it satisfies the condition that $\dot{h}_{ij} \geq -\alpha(h_{ij})$, which can be formulated as linear constraints on the control variable u:

$$\dot{h}_{ij} = \frac{\partial h_{ij}}{\partial \begin{bmatrix} \Delta \boldsymbol{p}_{ij} \\ \Delta \boldsymbol{v}_{ij} \end{bmatrix}} \begin{bmatrix} \Delta \dot{\boldsymbol{p}}_{ij} \\ \Delta \dot{\boldsymbol{v}}_{ij} \end{bmatrix} \geq -\alpha(h_{ij}), \tag{28}$$

where

$$\begin{bmatrix} \Delta \dot{\boldsymbol{p}}_{ij} \\ \Delta \dot{\boldsymbol{v}}_{ij} \end{bmatrix} = \begin{bmatrix} \Delta \boldsymbol{v}_{ij} \\ \frac{-1}{\tau_{\text{player}}}(\Delta \boldsymbol{v}_{ij} - (\boldsymbol{u}_i - \boldsymbol{u}_j)). \end{bmatrix}$$

The exact expression of (28) is omitted here for visual clarity, but it differs considerably from the one in [11] due to the damped dynamics in our system.

B. Collision Avoidance using Quadratic Programs

To guarantee collision avoidance among robots, we can formulate a quadratic program that minimally changes the nominal control $\hat{\boldsymbol{u}}_i$ (e.g., produced from trajectory planners

Strategy	**BL**	**DMPC**	**CMPC**
BL	1.6 - 2.2	0.2 - 3.2	1.8 - 1.6
DMPC	-	1.4 - 1.6	4.3 - 0.2
CMPC	-	-	1.0 - 0.8

TABLE II: Average Scores in Tournament 1 - Collisions allowed

Team	**BL**	**DMPC**	**CMPC**
BL	1.8 - 2.4	0.4 - 5.2	0.8 - 1.0
DMPC	-	1.4 - 1.7	1.9 - 0.3
CMPC			1.1 - 1.9

TABLE III: Average Scores in Tournament 2 - No collision enforced via CBF

described in Sec.VI) such that the actual control command $\boldsymbol{u}_i$ respects the CBF constraint (28) and the input constraint.

$$\begin{aligned} \boldsymbol{u}^* = \operatorname{argmin} u \quad & \sum_{i=1}^{N} \|\boldsymbol{u}_i - \hat{\boldsymbol{u}}_i\|^2 \\ \text{s.t.} \quad & \text{CBF constraints in (28)}, \quad \forall i \neq j, \\ & \|\boldsymbol{u}_i\|_\infty \leq u_{\max}, \qquad\quad \forall i \in \mathcal{M} \end{aligned} \tag{29}$$

where $\|\cdot\|_\infty$ denotes the infinity norm. To emphasize, the CBF constraints should only be added if the two robots i and j are moving towards each other, i.e., $\Delta\bar{v} < 0$. Due to the convexity of the problem, safe controls can be solved at every time step in a centralized fashion very efficiently.

VIII. Results

In this Section, we present some of the maneuvers performed by the described approach, as well as a comparison of the different strategies by creating a virtual tournament and using Monte-Carlo simulations to evaluate the outcome.

A. Bounce kicking

For a depiction of the successful execution of the bounce kick using trajectory optimization with contact dynamics, the interested reader is directed to our code repository, which contains a directory with select video results including the bounce kick.

B. Virtual Tournaments via Monte Carlo simulations

The strategies taken into account for our tournaments are the Baseline strategy (Part VI-A), the DMPC strategy (Part VI-B) and the Centralized-MPC strategy (Part VI-C). Each strategy is hand-tuned to the best of each author's knowledge, with slightly different parameters, and they slightly differ for the kicking velocity (4m/s vs 5m/s in CMPC).

The competition is designed so that every team plays against every other team for 15 matches (with an exception for the baseline strategy, which was run for 5 matches), where the maximum score of each match is 4. Matches differ due to the presence of actuation noise introduced in simulation. In order to take into account the "aggressiveness" of each team and the damage that it may cause to other robots, we additionally introduce a damage score, which is proportional to the induced squared velocity change from collision, so that higher-speed collisions cause more damage. To further take into account the aggressiveness of each team (e.g. the number of collision with other agents), we separate our tournament in two categories. In the first category, we allow the players to collide with each other, but we monitor their aggressiveness via the damage score. In the second category of tournament, we enforce "fairness" by using the CBF-based controller, in addition to each team's strategy, to deviate the trajectory of players about to collide. The CBF-based controller acts as an additional perturbation to the state of the game, allowing us to evaluate 1) which strategy is more respectful of the imposed constraints, and 2) which strategy is more robust to the external disturbances caused by the "referee" CBF-controller.

The results are shown in Table II for the tournament without CBF-controller, and in Table III for the tournament with CBF-based controller. In Appendix B have additionally included the histograms of the scores and of the *damage scores* for each team playing against every other team.

From the results of the Monte-Carlo simulations in Table II we can observe that the winning strategy appears to be the Decentralized-MPC method, which consistently outperforms the other two approaches, scoring an average of 4.0 points more when playing against the CMP, and 3.0 points more than the baseline approach. The Centralized-MPC, despite the high expectations, performs poorly, barely reaching the baseline approach. This is possibly caused by the tuning of the approach, as well as the advantage that a more aggressive strategy may have. This insight is also confirmed by the fact that, when fairness is enforced via the CBF-controller, the gap between Centralized-MPC and Decentralized-MPC decreases.From the damage score in Appendix B, we can observe that the centralized formulation (with its inter-player collision avoidance) indeed reduced the damage score for the agents. With some surprise, we also observe that the CBF has a small, if any, effect on the ability of teams to score.

IX. Conclusion

In this work we have presented a comparison of different trajectory optimization strategies and formulations for the control of a hybrid autonomous system, applying these techniques to the problem of multi-players air hockey. The comparison has been possible thanks to the in-house built simulator, which accurately models impact and dynamics, as well as a control-barrier function controller which allowed us to evaluate the effects of collisions on the dynamics of the game. Lessons learn include that contact dynamics in dynamic, adversarial environments are very challenging, that physical intuition can be very useful when tuning hybrid optimization strategies, that MIQP is hard! and that sometimes a simple strategy/controller can go a long way.

Acknowledgment

We would like to thank the teaching staff for 6.832, both for teaching the course and also for the periodic guidance they provided for the direction and methods of this project.
SURVEY KEYWORD: underactuated

REFERENCES

[1] R. D'Andrea, T. Kalmar-Nagy, P. Ganguly, and M. Babish, "The cornell robocup team," in *Robot Soccer World Cup*. Springer, 2000, pp. 41–51.

[2] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[3] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE control systems magazine*, vol. 22, no. 1, pp. 44–52, 2002.

[4] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 375–382, 2019.

[5] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Nonlinear model predictive control for multi-micro aerial vehicle robust collision avoidance," *arXiv preprint arXiv:1703.01164*, 2017.

[6] A. Bemporad, F. Borrelli, and M. Morari, "Piecewise linear optimal controllers for hybrid systems," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 2. IEEE, 2000, pp. 1190–1194.

[7] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. D. Ames, E. Feron, and M. Egerstedt, "The robotarium: A remotely accessible swarm robotics research testbed," *CoRR*, vol. abs/1609.04730, 2016. [Online]. Available: http://arxiv.org/abs/1609.04730

[8] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[9] A. Donev, S. Torquato, and F. H. Stillinger, "Neighbor list collision-driven molecular dynamics simulation for nonspherical particles. i. algorithmic details ii. applications to ellipses and ellipsoids," 2004.

[10] T. Marcucci and R. Tedrake, "Mixed-integer formulations for optimal control of piecewise-affine systems," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 230–239.

[11] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015.

APPENDIX A
CONTROL BARRIER FUNCTIONS

The dynamics of the players (1) without noise are in the control affine form:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}, \tag{30}$$

where $\boldsymbol{x} \in X \subseteq \mathbb{R}^n$ is the state, $\boldsymbol{u} \in U \subseteq \mathbb{R}^m$ is the input, and f and g are Lipschitz continuous. The set C is safe if

$$\boldsymbol{x}(0) \in C \implies \boldsymbol{x}(t) \in C \quad \forall t \geq 0. \tag{31}$$

Before we introduce the definition of control barrier functions, we note that an extended class $\mathcal{K}_\infty$ function $\alpha : \mathbb{R} \to \mathbb{R}$ is strictly increasing, $\alpha(0) = 0$, $\alpha(r) \to \infty$ as $r \to \infty$, and $\alpha(r) \to -\infty$ as $r \to -\infty$ [8].

Definition 1 (Control barrier function [8])**.** *Consider the dynamical system in control affine form* (30)*, and let $C \in D \subset \mathbb{R}^n$ be the super zero level set of a continuously differentiable function $h : D \to \mathbb{R}$, i. e. $C = \{\boldsymbol{x} \in D \subset \mathbb{R}^n \mid h(\boldsymbol{x}) \geq 0\}$. If there exists a Lipschitz continuous extended class $\mathcal{K}_\infty$ function α such that, for the system* (30) *and for all $\boldsymbol{x} \in D$,*

$$\sup_{\boldsymbol{u} \in U} \{L_f h(\boldsymbol{x}) + L_g h(\boldsymbol{x})\boldsymbol{u}\} \geq -\alpha(h(\boldsymbol{x})), \tag{32}$$

then h is a control barrier function. $L_f h(\boldsymbol{x})$ and $L_g h(\boldsymbol{x})$ denote the Lie derivatives of h in the directions of the vector fields f and g, respectively.

The following theorem summarizes two important properties of control barrier functions, which can be used to ensure both the forward invariance and the asymptotic stability of the safe set.

Theorem 1 ([8])**.** *Consider a dynamical system in control affine form* (30)*, and the super zero level set $C \subset \mathbb{R}^n$ of a continuously differentiable function h, defined as above. Then, any Lipschitz continuous controller $\boldsymbol{u}$ such that* (32) *holds for all $\boldsymbol{x} \in D$ renders the set C forward invariant and asymptotically stable, or in other words,*

$$\boldsymbol{x}(0) \in C \Rightarrow \boldsymbol{x}(t) \in C \quad \textit{for all } t \geq 0 \quad \textit{(forward invariance)}, \tag{33}$$

$$\boldsymbol{x}(0) \notin C \Rightarrow \boldsymbol{x}(t) \to \in C \quad \textit{as } t \to \infty \quad \textit{(asymptotic stability)}, \tag{34}$$

where $\boldsymbol{x}(0)$ denotes the state $\boldsymbol{x}$ at time $t = 0$, and $\boldsymbol{x}(t) \to \in C$ signifies that $\boldsymbol{x}$ approaches the set C.

With the forward invariance property, we can use control barrier functions to ensure that if the system starts in the safe set (e.g., no collisions), then it will remain in the safe for all future time. The asymptotic statbility property ensures that, if the system is driven outside the safe set (e.g., by noise), the system approaches the safe set eventually.

APPENDIX B
TOURNAMENT RESULTS

A. Tournament 1: No Control Barrier Function

1) Same-type: In this part we compare each strategy playing again adversaries using the same strategy, as shown in Figures 5 through 10.

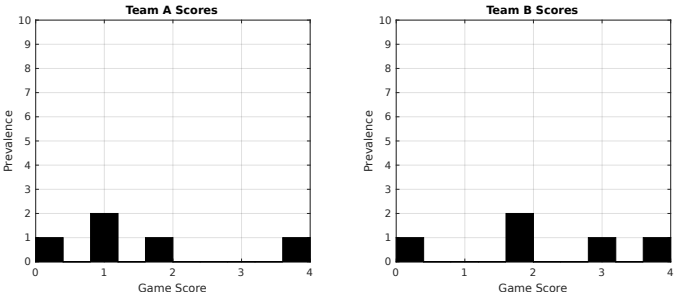

Fig. 5: Score of Baseline Strategy Vs. Baseline Strategy, no CBF

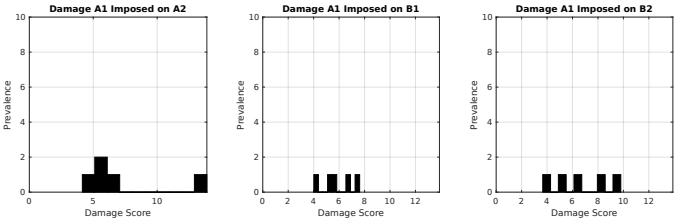

Fig. 6: Damage score of Baseline Strategy Vs. Baseline Strategy, no CBF

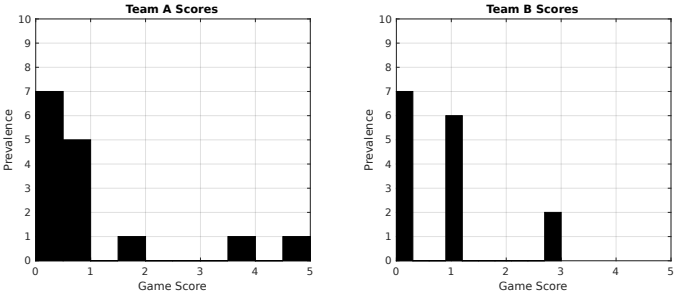

Fig. 7: Score of Centralized-MPC vs Centralized-MPC, no CBF

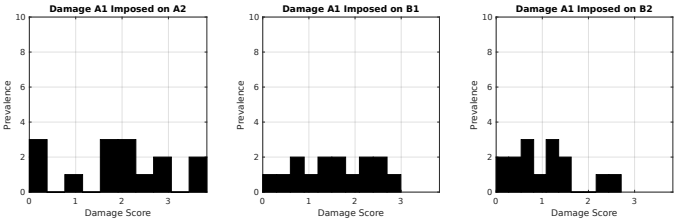

Fig. 8: Damage score of Centralized-MPC vs Centralized-MPC, no CBF

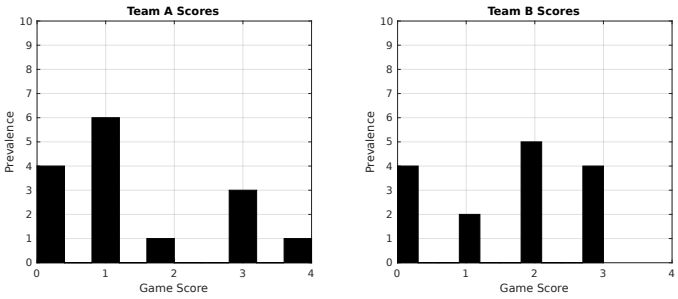

Fig. 9: Score of Decentralized-MPC vs Decentralized-MPC, no CBF

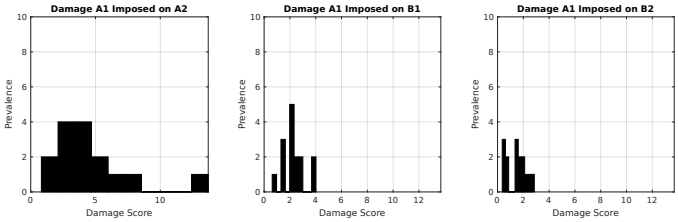

Fig. 10: Damage score of Decentralized-MPC vs Decentralized-MPC, no CBF

2) Different-type: In this part we compare each strategy playing again adversaries using a different strategy, as shown in Figures 11 through 16.

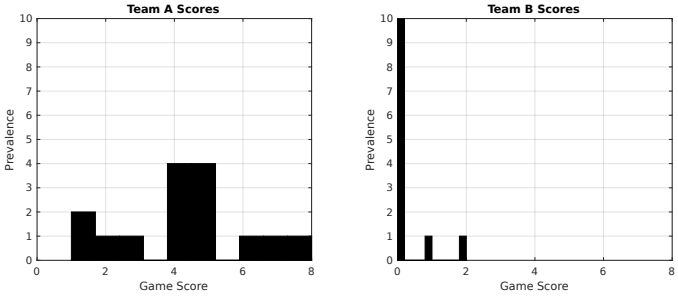

Fig. 11: Score of Decentralized-MPC vs Centralized-MPC, no CBF

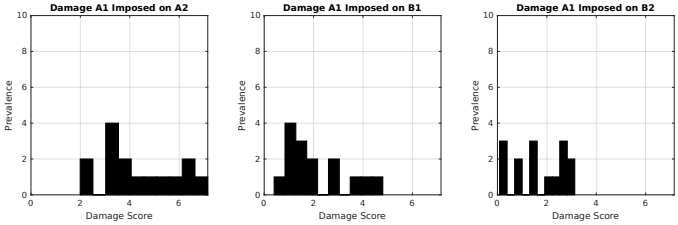

Fig. 12: Damage score of Decentralized-MPC vs Centralized-MPC, no CBF

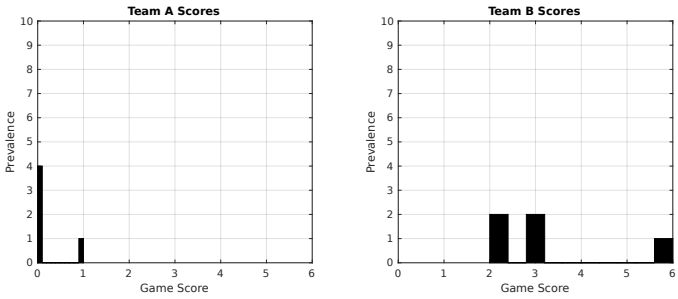

Fig. 13: Score of Baseline vs Decentralized-MPC, with CBF

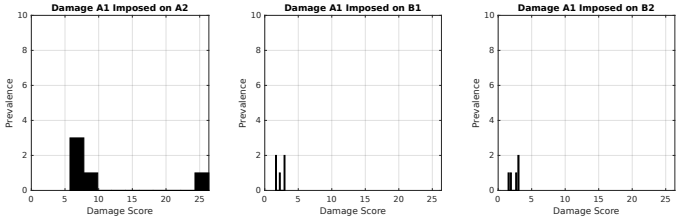

Fig. 14: Score of Baseline vs Decentralized-MPC, with CBF

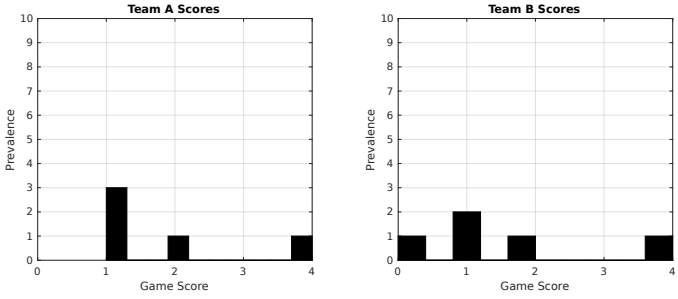

Fig. 15: Score of Baseline vs Centralized-MPC, with CBF

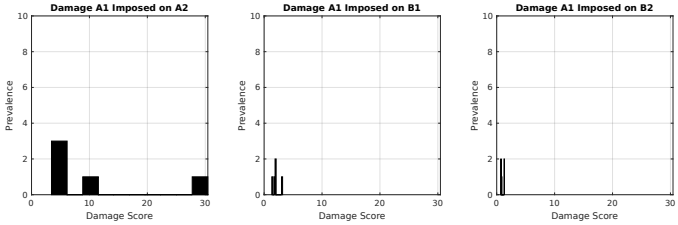

Fig. 16: Damage score of Baseline vs Centralized-MPC, with CBF

B. Tournament 2: With Control Barrier Function

1) Same-type: In this part we compare each strategy playing again adversaries using the same strategy with control barrier functions, as shown in Figures 17 through 22.

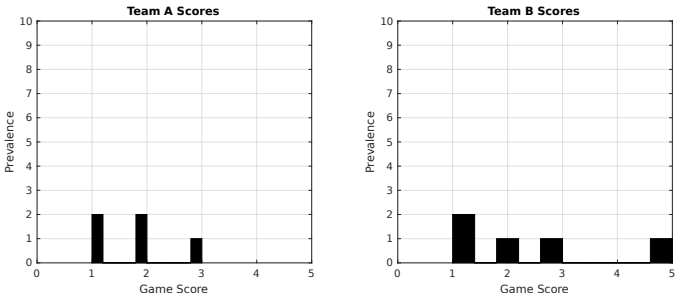

Fig. 17: Score of Baseline Strategy Vs. Baseline Strategy, with CBF

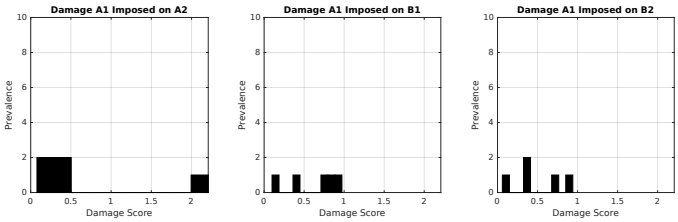

Fig. 18: Damage score of Baseline Strategy Vs. Baseline Strategy,with CBF

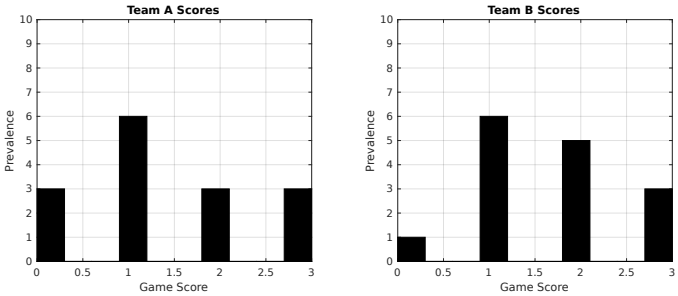

Fig. 19: Score of Decentralized-MPC vs Decentralized-MPC, with CBF

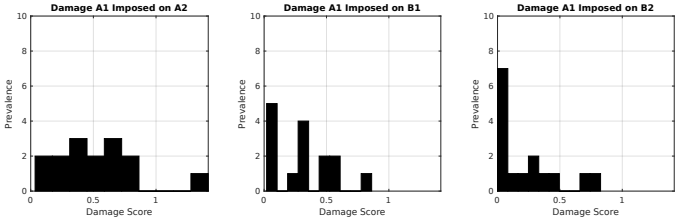

Fig. 20: Score of Decentralized-MPC vs Decentralized-MPC, with CBF

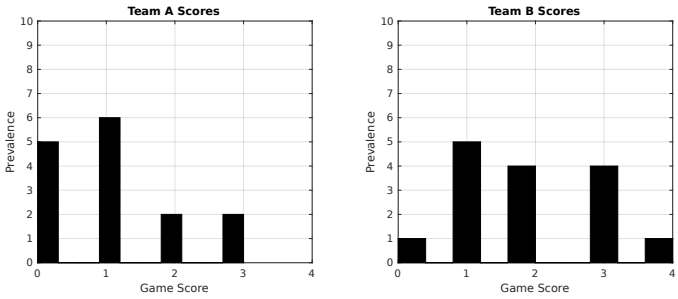

Fig. 21: Score of Centralized-MPC vs Centralized-MPC, with CBF

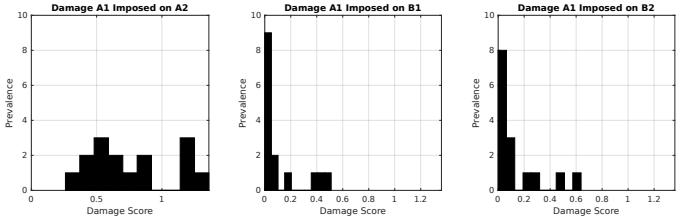

Fig. 22: Damage score of Centralized-MPC vs Centralized-MPC, with CBF

2) Different-type: In this part we compare each strategy playing again adversaries using a different strategy with control barrier functions, as shown in Figures 23 through 28.

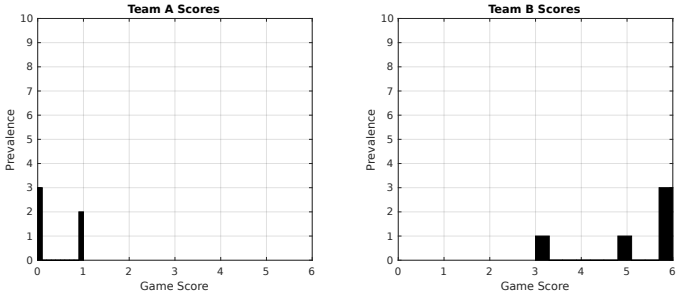

Fig. 23: Score of Baseline vs Decentralized-MPC, with CBF

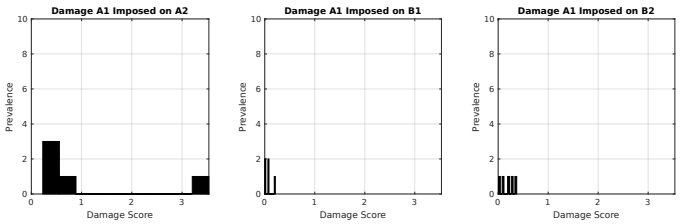

Fig. 24: Score of Baseline vs Decentralized-MPC, with CBF

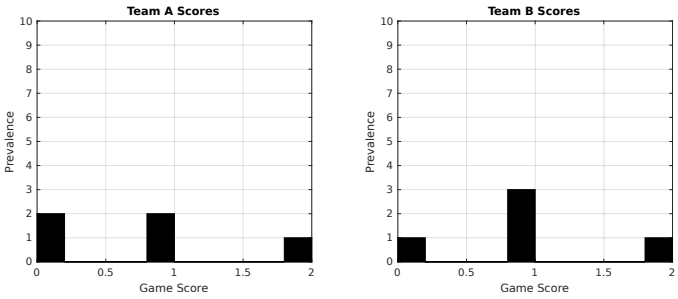

Fig. 25: Score of Baseline vs Centralized-MPC, with CBF

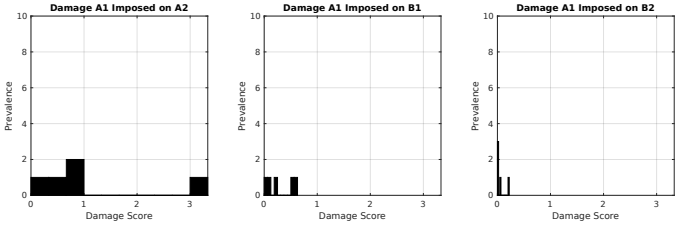

Fig. 26: Damage score of Baseline vs Centralized-MPC, with CBF

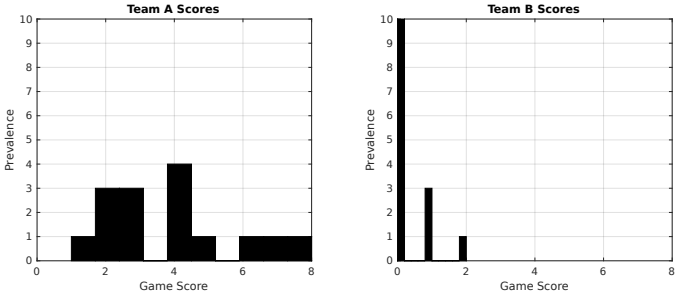

Fig. 27: Score of Decentralized-MPC vs Centralized-MPC, with CBF

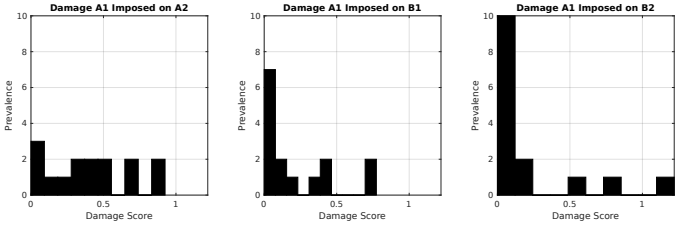

Fig. 28: Damage score of Decentralized-MPC vs Centralized-MPC, with CBF