Efficient and Reactive Planning for High Speed Robot Air Hockey

Puze Liu[1], Davide Tateo[1], Haitham Bou-Ammar[2,3], and Jan Peters[1]

***Abstract*— Highly dynamic robotic tasks require high-speed and reactive robots. These tasks are particularly challenging due to the physical constraints, hardware limitations, and the high uncertainty of dynamics and sensor measures. To face these issues, it's crucial to design robotics agents that generate precise and fast trajectories and react immediately to environmental changes. Air hockey is an example of this kind of task. Due to the environment's characteristics, it is possible to formalize the problem and derive clean mathematical solutions. For these reasons, this environment is perfect for pushing to the limit the performance of currently available general-purpose robotic manipulators. Using two Kuka Iiwa 14, we show how to design a policy for general-purpose robotic manipulators for the air hockey game. We demonstrate that a real robot arm can perform fast-hitting movements and that the two robots can play against each other on a medium-size air hockey table in simulation.**

I. INTRODUCTION

Recent hardware and software advances allow robots to depart static industrial surroundings to novel real-world *dynamic* environments with (potentially) fast-moving objects interaction involved. In such contexts, real-world constraints, estimation uncertainties, and dynamical interactions pose many challenges. For instance, hardware restrictions limit the design of high-speed trajectories, target tracking errors can lead to complete task failures, and fast-moving objects increase the likelihood of collisions with other obstacles or humans. Therefore, the design of swift, precise, and reactive movements is key to the success of such systems equipping robots with the essential tools to cope with ever-changing conditions.

Although designing task-specific robotic solution is always a possibility to tackle the aforementioned challenges, empowering *general-purpose* robots to solve dynamic tasks is desirable when the task is not well specified. Such a robot could face dangerous tasks performed previously by a human if equipped with appropriate end-effectors. However, the progress towards effective general-purpose robotic systems in dynamic environments has been limited due to a variety of factors. A possible step towards better understanding those limitations is to consider a more restricted dynamic environment, such as air hockey.

The air hockey task is a 2D constrained environment characterized by fast puck movements and high uncertainty. Two factors cause this high uncertainty: firstly, air flows through small holes on the table surface to reduce the friction between the puck and the table, creating an uneven airflow distribution, resulting in high uncertainty and fast movements. Secondly, the collision behavior between the cylindrical puck and mallet or the table's borders produces highly variable trajectories, as it is sensitive to small differences in the system state. This environment shows all the fundamental aspects of dynamic tasks: it requires robots to perform high-speed trajectories that reach the robot capability with low reaction time. However, it is relatively simple and controlled, letting us formalize the problem rigorously and perform appropriate scientific validation.

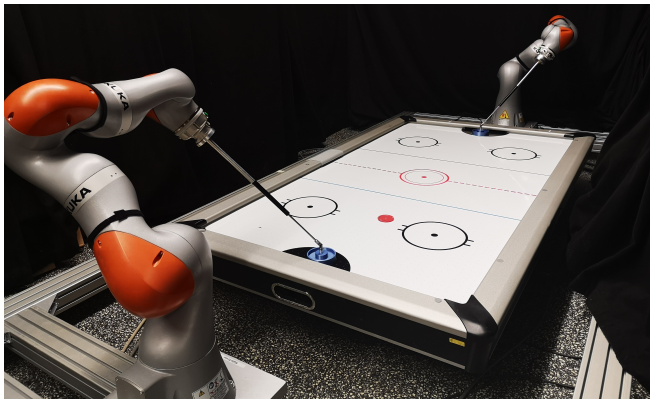


Fig. 1: Robot air hockey using two KUKA LBR IIWA 14.

Although many previous works have tried to solve this task [1], [2], [3], few have focused on general-purpose manipulators [4] and they are not able to show a highly dynamic behavior. Instead, our objective is to show how a general-purpose arm can achieve performances close to the task-specific 2DoF robots by employing advanced optimization techniques. To prove our claim, we consider a real air hockey system with the table size $216\text{cm} \times 122\text{cm}$. We mount two KUKA LBR IIWA 14 arms at each end of an air hockey table. The robots play against each other using a two-level policy. The high-level policy selects the appropriate tactic i.e., hitting, defending, etc., while the low-level one computes the required trajectory using planning and optimization techniques.

The main contribution of this work is a novel trajectory optimization technique. We employ a null space optimization algorithm that leverages the robot redundancy to generate high-speed motion while satisfying the joints' position and velocity constraints. We prove the effectiveness of this technique in real robot puck hitting. We also show the two

[1]Department of Computer Science, Technische Universität Darmstadt, Germany {puze,davide}@robot-learning.de, jan.peters@tu-darmstadt.de

[2] Huawei R&D London, United Kingdom haitham.ammar@huawei.com

[3] University College London (UCL), Honorary position.

robot arms playing reasonably fast against each other in a simulator. Finally, we use Bayesian Optimization (BO) to tune the physical parameters of the simulation. This approach does not require gradients, enabling us to use standard simulator platforms and easily consider non-differentiable dynamics.

A. Problem Statement

Air Hockey is a two players game. We refer to the two players as *home* and *away*. Let $\mathcal{T} \subset \mathbb{R}^2$, be the planar surface of a rectangular Air Hockey Table with dimension $l \times w$, with $w < l$. The table is divided symmetrically in two surfaces, each of length $\frac{l}{2}$, $\mathcal{T}_{\text{home}}$ and $\mathcal{T}_{\text{away}}$. Let $\mathcal{G}_{\text{home}}, \mathcal{G}_{\text{away}} \subset \mathbb{R}^2$ be two players' goal areas, placed at the two short sides of the table. Let $\boldsymbol{r} \in (\mathcal{T} \cup \mathcal{G}_{\text{home}} \cup \mathcal{G}_{\text{away}})$ and $\phi \in (-\pi, \pi]$ be, respectively, the position and orientation of a circular puck of radius R_{mallet}. Let $x_{\text{home}}, x_{\text{away}}$ be the position of players' mallets, both of radius R_{mallet}. Each player p can modify the puck trajectory, while the puck is in his own area $\mathcal{T}_{\text{p}}$, by causing a collision between the mallet and the puck i.e., $\|\boldsymbol{r} - \boldsymbol{x}_p\|_2 = R_{\text{puck}} + R_{\text{mallet}}$. The puck can change direction also by colliding with the boundaries of the table, excluding the boundaries between $\mathcal{T}$ and the goal areas $\mathcal{G}$. The objective of each player p is to maximize their score

$$\max \sum_{i}^{N} \sum_{t}^{T_i} \mathbb{1}(\boldsymbol{r}_t \in \mathcal{G}_{\neg p}),$$

where N is the number of game rounds, T_i is the length of round i, and $\neg p$ is the opponent.

Given that the mallet's position should stay on the table surface to hit effectively, a robotic agent must be able to execute fast trajectories on the planar manifold. As this task is tremendously difficult at high speed, the objective is to minimize the trajectory error w.r.t. the plane surface in the z axis. We add a mechanical compliant end-effector to cope with the error.

B. Related Work

Robotics researchers have a long-standing interest in Air Hockey. Such interest lies in the fact that Air Hockey is a very dynamic game that can exhibit interesting robot dynamics and tactics.

One of the first examples of a robotic Air Hockey system is presented in [5], where the authors designed a planar, 3-link, redundant manipulator that can play using cameras. In a subsequent paper [1], the authors described in detail the dynamics of the game, focusing specifically on collision modeling. One of the most outstanding Air Hockey platforms has been presented in [2], where a 4Dof Barrett Wam arm has been used as a planar robot to play air hockey fast. In the paper, the authors present a hierarchical architecture that considers low-level control as well as short and long-term strategies. It is possible to find another example of a planar robot playing Air Hockey in [3], where the authors implement a two-layer tactical system. The only known general-purpose manipulator to solve this task can be found in [4], where the authors use a Franka Emika Panda Arm. The proposed tactical system is strongly inspired on [2] and [3]. However, the system's performance is quite far from the task-specific ones, which use robots in a planar configuration. Up to our knowledge, our platform is the only general-purpose manipulator able to play air hockey with reasonable performances on a medium-size table.

Many works focus on simulated Air Hockey environments. In [6], the authors developed an automatic calibration procedure that allows the system to automatically tune the parameters that affect the puck's motion e. g., the restitution coefficients. In [7], the authors have studied how to optimize the attack motion against a human opponent. In [8], the authors developed a modified version of the DQN algorithm [9], incorporating suboptimal demonstrations. The authors show that the proposed approach outperforms other deep learning baselines.

The air hockey task has also been used in the Robot Learning community. An example is [10], where the authors use a very simplified air hockey task as a benchmark. In this work, the authors learn a latent representation of the opponent's tactics. However, the used environment is extensively limited and not comparable to other works that focus on the air hockey task. These limitations highlight the need for a fully functional air hockey platform. Thus, a more complex environment is also beneficial for the Robot Learning community. This task can be used as a test bench for many different Robot Learning areas: ranging from learning dynamical movements to multiagent learning and human-robot interaction.

II. PRELIMINARIES

A. Manipulator kinematics

We consider general-purpose, redundant, 7DoF manipulators. A robot configuration is fully described by the joint angles. The joint space is the space of all the possible joint angles: $\boldsymbol{q} = \{q_i | i \in \{1 \dots 7\}, q_i^{\min} \leq q_i \leq q_i^{\max}\}$. We can compute the end-effector pose $\boldsymbol{x}$ in a given configuration q by using the forward kinematics i.e., $\boldsymbol{x} = \text{FK}(\boldsymbol{q})$. The space of all allowed end-effector poses is called the task space. For any pose in the task space, we can compute one (or multiple) configurations in terms of joint positions using the inverse kinematics i.e., $\boldsymbol{q} = \text{IK}(\boldsymbol{x})$. For a redundant 7DoF manipulator, $\boldsymbol{q} \in \mathbb{R}^7$ and $\boldsymbol{x} \in \text{SE}(3)$. In this setting, the end-effector pose does not fully describe the system's configuration due to the extra degree of freedom originating from the robot's redundancy. From this fact, it follows that, for most configurations, the end-effector pose does not fully describe the system's configuration. There is still a degree of freedom due to the robot's redundancy.

The relation between the joint velocities $\dot{\boldsymbol{q}}$ and task space velocities $\dot{\boldsymbol{x}}$ is given through the Jacobian Matrix $\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$. For a redundant manipulator, it is possible to compute the desired joint velocities given a target end-effector velocity by solving (II-A) as follows

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}(\boldsymbol{q})\dot{\boldsymbol{x}} + \boldsymbol{N}\dot{\boldsymbol{q}}, \tag{1}$$

with the generalized inverse $\boldsymbol{J}^{\#}$ and the null space projection matrix $\boldsymbol{N} = \boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J}$. A common choice of the generalized inverse is the pseudoinverse $\boldsymbol{J}^{\dagger} = \boldsymbol{J}^{\mathsf{T}}(\boldsymbol{J}\boldsymbol{J}^{\mathsf{T}})^{-1}$. Here, the second component describes the joint velocities that change the robot's redundancy while fixing the end-effector pose.

Instead of using the null space projecting matrix, it is possible to parameterize the null space velocities using the coordinate system induced by the null space basis. We can rewrite (1) as $\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}(\boldsymbol{q})\dot{\boldsymbol{x}} + \boldsymbol{E}_{\boldsymbol{N}}\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is the velocity in the null space coordinates, $\boldsymbol{E}_{\boldsymbol{N}} = \mathrm{Null}(\boldsymbol{J})$ is the matrix of basis vectors spanning the null space of $\boldsymbol{J}$. When considering the 7DoF robot and the end-effector $\boldsymbol{x} \in \mathrm{SE}(3)$, the null space matrix becomes a single basis vector, $\boldsymbol{\alpha}$ becomes a scalar. This formulation is particularly useful to perform non-linear optimization.

An important metric that can be used to choose a given configuration is the Measure of Manipulability (MOM) $w = \sqrt{|\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}^{\mathsf{T}}(\boldsymbol{q})|}$. The MOM metric describes the distance to the singularity, as it is non-negative and becomes zero only at the singularities. MOM is also useful to understand in which configuration the joint velocities produce the highest end-effector velocity.

B. Bayesian Optimization

We use BO as a tool for portable and flexible system identification. In BO, the objective is to *efficiently* maximize a (stochastic) black-box model based on function-value information, i.e., $\max_{\boldsymbol{\theta}\in\Theta} f(\boldsymbol{\theta})$ with $\Theta \subseteq \mathbb{R}^d$ being the d-dimensional search domain. Algorithms of this type are sequential in nature. At each round i, an input $\boldsymbol{\theta}_i \in \Theta$ is selected and a black-box function value $f(\boldsymbol{\theta}_i)$ is observed. The goal is to rapidly (in terms of regret) approach $\boldsymbol{\theta}^{\star} = \arg\max_{\boldsymbol{\theta}\in\Theta} f(\boldsymbol{\theta})$. Since both $f(\cdot)$ and $\boldsymbol{\theta}^{\star}$ are unknown, solvers need to trade off exploitation and exploration during the search process.

To achieve this goal, typical BO algorithms operate in two steps. In the first, a Bayesian model is learned, while in the second an acquisition function determining new queries is maximized. Next, we briefly survey both those steps.

1) BO with Gaussian Processes (GPs): Gaussian process regression [11] offers a flexible and sample efficient modelling alternative to reason about $f(\cdot)$. In those forms, designers impose a GP prior on latent functions, which are fully specified by a mean function $m(\boldsymbol{\theta})$ and a covariance kernel $k_{\boldsymbol{\lambda}}(\boldsymbol{\theta}, \boldsymbol{\theta}')$ with $\boldsymbol{\lambda}$ being kernel hyper-parameters. Assuming Gaussian likelihood noise with variance σ and given a dataset $\mathcal{D}_i = \{\boldsymbol{\theta}_l, y_l \equiv f(\boldsymbol{\theta}_l)\}_{l=1}^{n_i}$ with n_i denoting all gathered data up to the i^{th} round, one can compute output predictions on novel input queries $\boldsymbol{\theta}^{\star}_{1:q}$. This is achieved through the predictive posterior which is given by $f(\boldsymbol{\theta}^{\star}_{1:q})|\mathcal{D}_i, \boldsymbol{\lambda} \sim \mathcal{N}\left(\boldsymbol{\mu}_i(\boldsymbol{\theta}^{\star}_{1:q}; \boldsymbol{\lambda}), \boldsymbol{\Sigma}_i(\boldsymbol{\theta}^{\star}_{1:q}; \boldsymbol{\lambda})\right)$ with

$$\begin{aligned}\boldsymbol{\mu}_i(\boldsymbol{\theta}^{\star}_{1:q}; \boldsymbol{\lambda}) &= \overbrace{\boldsymbol{K}^{(i)}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}^{\star}_{1:q}, \boldsymbol{\theta}_{1:n_i})}^{\boldsymbol{A}^{(i)}} \tilde{\boldsymbol{K}}^{(i)}_{\boldsymbol{\lambda}} \boldsymbol{y}_{1:n_i}, \\ \boldsymbol{\Sigma}_i(\boldsymbol{\theta}^{\star}_{1:q}; \boldsymbol{\lambda}) &= \boldsymbol{K}^{(i)}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}^{\star}_{1:q}, \boldsymbol{\theta}^{\star}_{1:q}) - \boldsymbol{A}^{(i)} \tilde{\boldsymbol{K}}^{(i)}_{\boldsymbol{\lambda}} \boldsymbol{A}^{\mathsf{T},(i)},\end{aligned} \tag{2}$$

where we have used $\boldsymbol{\theta}_{1:n_i}$ and $\boldsymbol{\theta}^{\star}_{1:q}$ to concatenate all n_i inputs and q queries, and $\tilde{\boldsymbol{K}}^{(i)}_{\boldsymbol{\lambda}} = [\boldsymbol{K}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_{1:n_i}, \boldsymbol{\theta}_{1:n_i}) + \sigma \boldsymbol{I}]^{-1}$. In the latter $\boldsymbol{K}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_{1:n_i}, \boldsymbol{\theta}_{1:n_i}) \in \mathbb{R}^{n_i \times n_i}$ denotes the covariance matrix that is computed such that $[\boldsymbol{K}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_{1:n_i}, \boldsymbol{\theta}_{1:n_i})]_{k,l} = k_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_k, \boldsymbol{\theta}_l)$.

The remaining ingredient needed in a GP pipeline is a process to determine the unknown hyper-parameters $\boldsymbol{\lambda}$ given a set of observations. In standard GPs [11], $\boldsymbol{\lambda}$ are fit by minimising the negative log likelihood leading us to the following optimization problem

$$\min_{\boldsymbol{\lambda}} \mathcal{J}(\boldsymbol{\lambda}) = \frac{1}{2}\det\left(\tilde{\boldsymbol{K}}^{(i)}_{\boldsymbol{\lambda}}\right) + \frac{1}{2}\boldsymbol{y}^{\mathsf{T}}_{1:n_i}\tilde{\boldsymbol{K}}^{(i)}_{\boldsymbol{\lambda}}\boldsymbol{y}_{1:n_i} + \frac{n_i}{2}\log 2\pi.$$

2) Acquisition Function Maximization: Acquisition functions trade off exploration and exploitation for determining new probes to evaluate by utilising statistics from the posterior $p_{\boldsymbol{\lambda}}(\cdot) \equiv p(f(\cdot)|\mathcal{D}_i, \boldsymbol{\lambda})$. Among various types, we focus on three myopic acquisitions being Expected Improvement (EI) [12], Probability of Improvement (PI) [13], and Upper Confidence Bound (UCB) [14].

EI Acquisition: In EI, one determines new queries by maximizing expected gain relative to the function values observed so far [12]. In a batch form, an EI acquisition is defined as

$$\alpha_{\mathrm{EI}}(\boldsymbol{\theta}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{p_{\boldsymbol{\lambda}}(\cdot)}\left[\max_{j\in 1:q}\left\{\mathrm{ReLU}(f(\boldsymbol{\theta}_j) - f(\boldsymbol{\theta}^{+}_i))\right\}\right],$$

where $\boldsymbol{\theta}_j$ is the j^{th} vector in the batch $\boldsymbol{\theta}_{1:q}$, $\boldsymbol{x}^{+}_i$ is the best performing input so far and $\mathrm{ReLU}(a) = \max\{0, a\}$.

PI Acquisition: To assess a new batch of probes, PI measures the probability of acquiring gains in the function value compared to $f(\boldsymbol{\theta}^{+}_i)$. Such a probability is measure through an expected left-continuous Heaviside function, $\mathbb{1}(\cdot)$, as follows

$$\alpha_{\mathrm{PI}}(\boldsymbol{\theta}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{p_{\boldsymbol{\lambda}}(\cdot)}\left[\max_{j\in 1:q}\left\{\mathbb{1}\{f(\boldsymbol{\theta}_j) - f(\boldsymbol{\theta}^{+}_i)\}\right\}\right],$$

UCB Acquisition: In this type of acquisition, the learner trades off the mean and variance of the predictive distribution to gather new query points for function evaluation

$$\alpha_{\mathrm{UCB}}(\boldsymbol{\theta}_{1:q}, \mathcal{D}_i) = \mathbb{E}_{p_{\boldsymbol{\lambda}}}\left[\max_{j\in 1:q}\{\mu_i(\boldsymbol{\theta}_j; \boldsymbol{\lambda}) + \sqrt{\beta\pi/2}|\gamma_i(\boldsymbol{\theta}_j; \boldsymbol{\lambda})|\}\right],$$

where $\mu_i(\boldsymbol{\theta}_j; \boldsymbol{\lambda})$ is the posterior mean given in Equation 2 and $\gamma_i(\boldsymbol{\theta}_j; \boldsymbol{\lambda}) = f(\boldsymbol{\theta}_j) - \mu_i(\boldsymbol{\theta}_j; \boldsymbol{\lambda})$.

With modeling and acquisition ingredients defined, BO runs a loop that iteratively fits a GP model (Section II-B.1) and then determines a batch of new probes to evaluate by maximizing acquisitions from Section II-B.2.

III. OPTIMIZATION OF HITTING MOVEMENT

Hitting is the most challenging movement to perform in the air hockey task. It is necessary to accelerate the puck to high velocities, such that two robots can play autonomously against each other. To obtain this result, we need to maximize the mallet's speed when the collision with the puck happens.

Unfortunately, it can be hard to achieve this objective with a general-purpose, 7-DoF manipulator. While the maximum joint velocity can be sufficient for high-speed point-to-point movements, the environment's constraints, i.e., the

plane manifold of the air hockey game, introduce many problematic issues. Specifically, these constraints are located in two different spaces (the task space and the joint space). The classical approach consists of planning and inverse kinematics. This method solves the problem in the task space and doesn't consider the joint velocity constraints. Thus, it is not suitable for high-speed movements. Conversely, optimizing in the joint space leads to many high-dimensional constrained non-linear optimization problems, which do not fit the real-time requirements.

In our approach, we plan a collision-free Cartesian trajectory based on the start, hitting, and stop points such that the Cartesian constraints are satisfied. Then, we propose a linear constrained Quadratic Programming (QP) to compute desired joint velocities on each trajectory point. Unlike the simple joint trajectory optimization method, where both objective and constraints are nonlinear, we can compute the solution of QP fast and easily. With our approach, we can find a hitting trajectory in real-time satisfying both Cartesian and joint constraints.

Moreover, we optimize the joint configuration at the hitting point to maximize the robot's performance. At the hitting point, instead of constructing a non-linear, high-dimensional (position + velocity) optimization problem, we decompose the problem into two lower-dimensional problems. Firstly, we perform a position-only Nonlinear Programming (NLP) by maximizing the manipulability along the hitting direction. Secondly, we propose two methods to find the maximum end-effector velocity reachable in that configuration. Given the desired hitting configuration, we present the Anchored Quadratic Programming (AQP), a variant of the original QP, which significantly improves the hitting performance.

A. Cartesian Trajectory Planning

Typical trajectory planning methods are not applicable in the air-hockey task, for example, the cubic/quintic polynomials could exceed the table's boundaries, the Bezier curve [15] does not provide a monotonic velocity profile (the maximum velocity doesn't occur at the hitting point). Therefore, we design a trajectory planning method composed of two segments: *hitting*, *stop*. As illustrated in Fig. 2, each segments of the trajectory is composed of two parts: *linear part* and *arc part*. Given the start and the stop point, the tightened boundaries are firstly determined to prevent a motion in an unnecessary direction. Then we find two cross-points between the tightened boundaries and the line that passes through the hit point along the hitting direction. The *arc part* is tangent to the line segment of start/endpoint and middle point as well as the line segment of middle point and hit point with the maximum arc radius. The *linear part* links the arc to the unconnected point.

The planed path is parameterized by the arc length $\boldsymbol{x} = \boldsymbol{g}(s)$. For each segment, we use a quartic polynomial to define the profile of the arc length $s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t + 3 + a_4 t^4$. We employ the positions and velocities boundary conditions as the desired position and velocity of each segment and set the acceleration boundary conditions

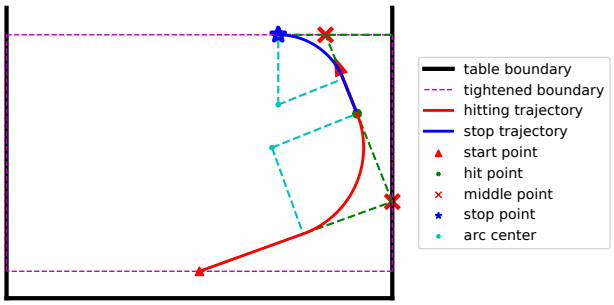


Fig. 2: Collision-free hitting path planning. We use two auxiliary middle points to find the path. The arc radius is maximized to reduce angular acceleration.

at two endpoints as $\ddot{s}(0) = \ddot{s}(t_f) = 0$, the arc length is thus guaranteed to be monotonic in the time interval $[0, t_f]$.The total motion time is $t_f = 2s_f / v_{\text{hit}}$, where v_{hit} is the speed at one of the two endpoints that is greater than 0.

B. Weighted Null Space Optimization (QP)

To achieve high-speed trajectories, we optimize the null space velocity at every point of the trajectory. We attach a universal joint to the end-effector (Section V-A) and focus purely on the positions of the end-effector in task space, the concerned Jacobian reduced to $\boldsymbol{J}_p \in \mathbb{R}^{3\times n}$. Let $\boldsymbol{b} = \boldsymbol{J}_{\boldsymbol{p}}^{\dagger}\left(\frac{\bar{\boldsymbol{x}} - \boldsymbol{x}}{\Delta T} + \bar{\boldsymbol{v}}\right)$ be the joint velocities for the trajectory tracking, $\boldsymbol{x}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{v}}$ are respectively actual, desired cartesian position and desired cartesian velocity. The objective is to minimize the weighted joint velocity as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\left\|\boldsymbol{b} + \boldsymbol{E_N}\boldsymbol{\alpha}\right\|_{\boldsymbol{W}}^2 \\ \text{s.t.} \quad & \dot{\boldsymbol{q}}^{\min} \le \boldsymbol{b} + \boldsymbol{E_N}\boldsymbol{\alpha} \le \dot{\boldsymbol{q}}^{\max}, \end{aligned} \tag{3}$$

where $\boldsymbol{E_N} \in \mathbb{R}^{n\times(n-3)}$ is the *null space matrix* that each column is one basis vector of the $\text{Null}(\boldsymbol{J_p})$ and $\boldsymbol{\alpha} \in \mathbb{R}^{n-3}$ are entries on each basis vector.

It should be pointed out that, when the $\boldsymbol{W}$ is set to be an identity matrix, the optimal solution is $\boldsymbol{\alpha} = \boldsymbol{0}$ when the constraints are not violated. In practice, we apply higher weights on the shoulder and the elbow joints because these joints afford more loads and have lower velocity limits. Finally, we can obtain the next step velocity and position by Euler integration

$$\begin{aligned} \dot{\boldsymbol{q}}_{\text{next}} &= \boldsymbol{b} + \boldsymbol{E_N}\boldsymbol{\alpha}^*, \\ \boldsymbol{q}_{\text{next}} &= \boldsymbol{q}_{\text{current}} + \dot{\boldsymbol{q}}_{\text{next}}\Delta t. \end{aligned}$$

C. Hitting Configuration Optimization (NLP)

To find the best configuration for the hitting movement, we consider the manipulability along a specific hitting direction $\boldsymbol{v}$, as presented in [16]. We maximize the following quantity at the hitting point $\boldsymbol{p}$

$$\max_{\boldsymbol{q}} \ \left\|(\boldsymbol{v}^{\mathsf{T}}\boldsymbol{J}_p(\boldsymbol{q}))\right\|_2, \qquad \text{s.t. } \text{FK}_p(\boldsymbol{q}) = \boldsymbol{p}, \tag{4}$$

And $\text{FK}_p(\boldsymbol{q})$ is the forward kinematics w.r.t positions (x, y, z).

D. Computing the Maximum Hitting Velocity

To compute the maximum hitting velocity at the selected hitting configuration, we can use two different strategies: 1) the least square solution, which can be computed easily in closed form 2) the linear programming approach. The least-square approach produces a feasible solution while the linear programming method results in the maximum theoretical velocity on that configuration.

a) Least square solution: We parameterize the hitting velocity as a scalar speed value η and a hitting direction $\boldsymbol{v}$, such that $\boldsymbol{v}_{\max} = \eta \boldsymbol{v}$. The least square solution is $\dot{\boldsymbol{q}} = \boldsymbol{J}_{\boldsymbol{p}}^{\dagger}(\boldsymbol{q}^*)\boldsymbol{v}_{\max}$. Thus, the maximum possible joint velocity can be determined by the minimum ratio of the absolute value of i-th joint velocity $\dot{q}_i$ and its maximum $\dot{q}_i^{\max}$

$$\eta = \min_i \left(\frac{\dot{q}_i^{\max}}{|\dot{q}_i|} \right), \quad i \in \{1, \cdots, n\}.$$

b) Linear Programming (LP): Instead of using least squared solution, we can construct a linear program on the null space as

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \boldsymbol{v}^{\mathsf{T}} \boldsymbol{J}_{\boldsymbol{p}}(\boldsymbol{q}^*) \left(\boldsymbol{E}_{\boldsymbol{v}^\perp}(\boldsymbol{q}^*)\boldsymbol{\alpha} \right), \\ \text{s.t.} \quad & \dot{\boldsymbol{q}}^{\min} \leq \boldsymbol{E}_{\boldsymbol{v}^\perp}(\boldsymbol{q}^*)\boldsymbol{\alpha} \leq \dot{\boldsymbol{q}}^{\max}, \end{aligned} \tag{5}$$

where $\boldsymbol{v}^\perp \in \mathbb{R}^3$ is the orthogonal complement space of $\boldsymbol{v} \in \mathbb{R}^3$, which in our case is a plane orthogonal to the hitting direction, $\boldsymbol{E}_{\boldsymbol{v}^\perp}(\boldsymbol{q}^*) = \text{Null}\left[(\boldsymbol{v}^\perp)^{\mathsf{T}} \boldsymbol{J}_{\boldsymbol{p}}(\boldsymbol{q}^*) \right]$ is the basis vectors spanning the null space of $(\boldsymbol{v}^\perp)^{\mathsf{T}} \boldsymbol{J}_{\boldsymbol{p}}(\boldsymbol{q}^*)$.

E. Anchored Null Space Optimization (AQP)

The QP optimization mentioned in Section III-B is a local optimizer that focuses only on the current step. It can lead to the local optima with undesirable redundancy configuration, e.g., lower elbow configurations can cause collisions with the table. To avoid this problem, we can modify the optimization problem to an AQP

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \left\| (\boldsymbol{b} + \boldsymbol{E}_{\boldsymbol{N}}\boldsymbol{\alpha}) - \dot{\boldsymbol{q}}^a \right\|_{\boldsymbol{W}}^2, \\ \text{s.t.} \quad & \dot{\boldsymbol{q}}^{\min} \leq \boldsymbol{b} + \boldsymbol{E}_{\boldsymbol{N}}\boldsymbol{\alpha} \leq \dot{\boldsymbol{q}}^{\max}, \end{aligned} \tag{6}$$

where $\dot{\boldsymbol{q}}^a = z\frac{(\boldsymbol{q}^a - \boldsymbol{q})}{c}$ is a reference velocity that leads to the anchored configuration $\boldsymbol{q}^a$ which is solved from hitting configuration optimization and z is the phase variable, which linearly increasing from 0 in the hitting segment and linearly decreasing to 0 in the stop segment and c is a scale constant. Instead of minimizing the weighted joint velocity in (3), we minimize the distance to the reference velocity. When $z = 0$ at the hitting point, the objective is the same as (3) and when $z = 1$ at the two ends of the trajectory, the objective tries to find the closest configuration to the reference.

IV. SYSTEM IDENTIFICATION

For in-depth safe experimentation, we resort to building an accurate simulator of the puck's behavior in Gazebo. We realize a set of 7 tunable parameters that are summarized in Table I. Due to the difficulty associated with differentiating through a Gazebo simulator, we follow a Bayesian optimization scheme (Section II-B) for allocating optimal configurations of those parameters. Our black-box function consists of two terms responsible for errors between the puck's position and orientation as generated by Gazebo versus real data gathered using the Optitrack system, i.e., $\{\langle \boldsymbol{r}_t^{(i)}, \phi_t^{(i)} \rangle_{t\geq 1}\}_{i\in[1:N]}$ with N being the total number of collected trajectories. In short, we define the black box as an average over real-world traces as follows:

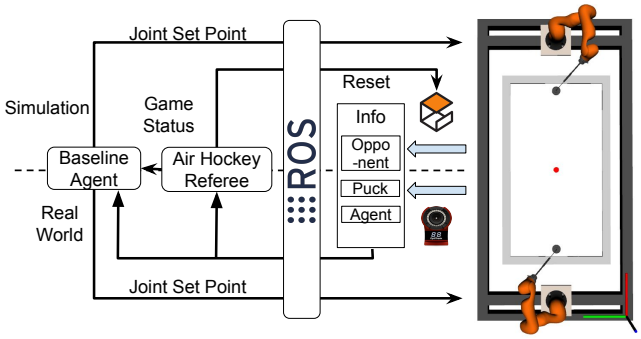


Fig. 3: Framework of the air hockey system. Real world and simulation share the same structure.

$$f(\boldsymbol{\theta}) = \mathbb{E}_{i\sim\mathcal{U}[1:N]} \left[\frac{1}{T} \sum_{t=1}^{T} ||\Delta r_t^{(i)}(\boldsymbol{\theta})||_2^2 + 0.2 \|\Delta\phi_t^{(i)}(\boldsymbol{\theta})\|_{\angle}^2 \right],$$

where $\Delta r_t^{(i)} = \boldsymbol{r}_t^{(i)} - \hat{\boldsymbol{r}}_t^{(i)}(\boldsymbol{\theta})$ and $\Delta\phi_t^{(i)} = \phi_t^{(i)} - \hat{\phi}_t^{(i)}(\boldsymbol{\theta})$ with $\{\langle \hat{\boldsymbol{r}}_t^{(i)}(\boldsymbol{\theta}), \hat{\phi}_t^{(i)}(\boldsymbol{\theta}) \rangle_{t\geq 1}\}_{i\in[1:N]}$ being positions and orientations gathered from Gazebo under a specific setting of the parameters $\boldsymbol{\theta}$ from Table I. In the above equation we use $\mathcal{U}(\cdot)$ to denote a uniform distribution over the trajectories, while $\angle$ represents the (shortest) angular distance between $\phi_t^{(i)}$ and $\hat{\phi}_t^{(i)}(\boldsymbol{\theta})$.

Unfortunately, BO as described in Section II-B faces two significant challenges when attempting to optimize for $\boldsymbol{\theta}$. First, nonlinearities and puck collisions induce a non-Gaussian (heteroscedastic) likelihood noise violating Gaussianity assumptions inherent to standard GPs, and second, the choice of the acquisition to optimize is vague, whereby different acquisitions can lead to conflicting minima [17]. To circumvent the aforesaid problems, we adapt techniques from Heteroscedastic Evolutionary Bayesian Optimisation (HEBO) [17] to tune the parameters in Table I. Precisely, we utilize input and output warping [18], [19] to map data distributions to ones that closely resemble Gaussian densities and rather adopt a multi-objective acquisition to determine

parameter	min	max
restitution long-side rim	0.5	1
restitution short-side rim	0.5	1
friction coefficient rim	0	0.5
friction coefficient table-surface	0	0.5
puck spinning	0	0.5
puck linear velocity decay	0	0.01
puck angular velocity decay	0	0.01

TABLE I: Parameters for system identification

Pareto-optimal solution between $\alpha_{\text{EI}}(\cdot)$, $\alpha_{\text{PI}}(\cdot)$, and $\alpha_{\text{UCB}}(\cdot)$, i.e., solving $\max_{\boldsymbol{\theta}\in\Theta}(\alpha_{\text{EI}}(\cdot), \alpha_{\text{PI}}(\cdot), -\alpha_{\text{UCB}}(\cdot))$; see [17].

V. SYSTEM ARCHITECTURE

A. Hardware & Software

To demonstrate the effectiveness of general robotic systems in tackling dynamic tasks, we created an air hockey game between two Kuka LBR IIWA 14 manipulators. We mounted those arms at either end of an air hockey table and equipped each with a custom-designed end-effector of length 515 mm. The end-effector was composed of an aluminum rod, a 10 N gas spring, and a universal joint connected to a mallet. Our choice of a gas spring reduced contact forces to the table in case of vertical control error and as such increased safety by diminishing exerted pressures from the robotic arms. Additionally, the universal joint passively adapts row and pitch angles of the end-effector to ensure that the mallet's surface is parallel to the table. The cylindrical symmetry of the mallet warrants collisions which are invariant to yaw angles. Additionally, we positioned six Optitrack Flex 13 cameras with (1920×1680) pixel resolution and frame rates of 120 Hz above the air hockey table. We enabled effective object tracking by placing distinctive markers on both the puck and the table allowing us to achieve a 1mm tracking precision.

On the software side, we built our framework using ROS in both real-world and simulation as illustrated in Fig. 3. In the real system, the puck and table poses were determined using the Optitrack system, while in simulation those poses were directly provided by Gazebo. Finally, we introduced a referee node that established the game's status and score. This node handles game logic by observing the state of the robot and the puck, performed checks of faulty states, and reset the puck's position in case of a scored goal or when the puck was unreachable.

B. State Estimation

With our hardware setting presented, we now focus on the procedure by which we gather puck data (i.e., position and velocity) that we later use for both fine-tuning the simulator and for optimizing hitting movements (Section III). Given that the puck's position, $\boldsymbol{r}_t$, and orientation ϕ_t are attainable from the Optitrack system, we adopt an Extended Kalman filter to estimate linear and angular velocities $\dot{\boldsymbol{r}}_t$ and $\dot{\phi}_t$.

To estimate these quantities, we use an Extended Kalman Filter with the following transition model

$$\begin{aligned}
\boldsymbol{r}_{t+1} &= \boldsymbol{r}_t + \dot{\boldsymbol{r}}_t \Delta T + \boldsymbol{\varepsilon}_{\boldsymbol{r}}, \\
\dot{\boldsymbol{r}}_{t+1} &= \dot{\boldsymbol{r}}_t - (d\dot{\boldsymbol{r}}_t + \boldsymbol{c}(\dot{\boldsymbol{r}}))\Delta T + \boldsymbol{\varepsilon}_{\dot{\boldsymbol{r}}}, \\
\phi_{t+1} &= \text{wrap_angle}(\phi_t + \dot{\phi}_t \Delta T + \varepsilon_{\phi}), \\
\dot{\phi}_{t+1} &= \dot{\phi}_t + \varepsilon_{\dot{\phi}},
\end{aligned}$$

where ε_k is the (Gaussian) noise specific to the state variable k, $\boldsymbol{c}(\cdot)$ is the friction term vector where we define every component i as

$$c_i(\dot{\boldsymbol{r}}) = \begin{cases} c, & |\dot{r}_i| > 0 \\ 0 & \text{otherwise.} \end{cases}$$

As the Optitrack system can also track rigid bodies' orientation, the observed variables are $\boldsymbol{r}$ and ϕ. After each prediction step, we apply the collision model from [1]. Every time a collision happens, we don't update the covariance matrix, as it is hard to model the bounce uncertainty.

We use an ellipsoidal gating procedure for the Optitrack measure. To compute the gate, we use the innovation covariance matrix and a gate probability of 90% i.e., we reject the 10% of the possible measurement which belongs to the queue of our estimated distribution. As the Optitrack is very unlikely to produce outliers, we interpret the outliers as an unexpected collision. For this reason, instead of rejecting the measurement, we reset the track using the new information and reset the state covariance matrix to the initial one i.e., the identity matrix.

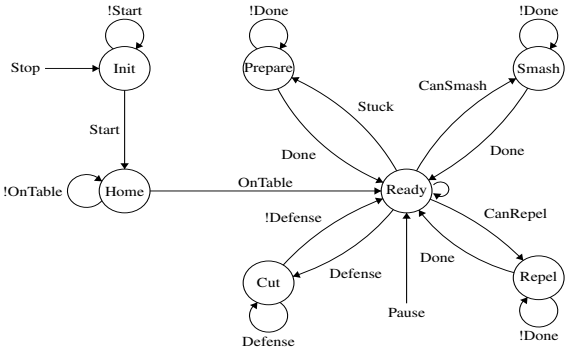
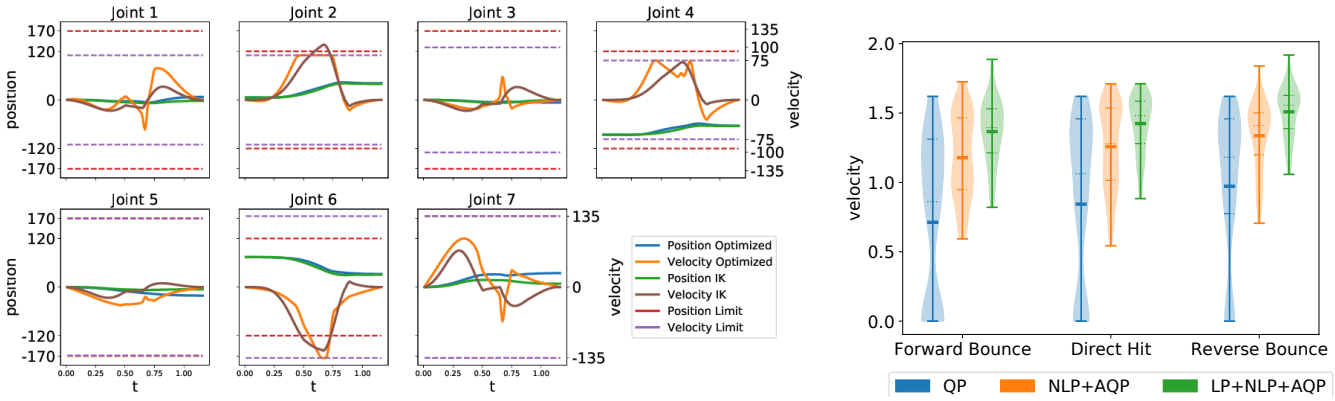


Fig. 4: State machine of the high-level policy

C. Agent Strategy

Following previous works [3], we use a two-layer hierarchical architecture to control the robot. At a high level, we leverage a classical finite state machine to implement the tactics. The high-level strategies that we propose are very similar to the one presented in the previous literature [3], [4]. We show the full state machine in Fig. 4. The *Init* state is a safe state where the robot positions the mallet at a safe height from the table. When we stop the game, the robot reverts to this safe state, where no action can occur. Once we start the game, the robot enters the *Home* state, where it moves the mallet on the table. Once both robots have positioned their mallets on the table, The system enters the *Ready* state, which keeps the robot on the base position until an event occurs. The *Ready* state is active while the game is paused.

All the other possible states are reachable from the *Ready* state. The *Smash* state can be activated when the puck velocity is not too high. It hits the puck strongly and aims at the goal. The *Repel* state quickly responds to a fast incoming puck that is not risky, by hitting back the puck in the incoming direction. The *Prepare* state is used when the puck is stationary and close to the table's borders. It moves the



(a) Optimized solution and inverse kinematic solution

(b) Hitting velocity for different algorithms

Fig. 5: Hitting movement Optimization. (a) shows that the velocity limit gets violated on joint 2 for planning and inverse kinematic approach. Our approach apportions the velocity of joint 2 into other joints. (b) shows that NLP signific antly improves the hitting performance and eliminates the failures. LP tries to reach the theoretical maximum of the configuration.

puck in a better position to be able to hit it strongly. Finally, the *Cut* state is responsible for defending the goal. It tries to move the incoming puck at the sides to gain control of it.

VI. EXPERIMENTAL EVALUATION

In this section, we present the experimental results for the hitting movement and the system identification procedure. In the attached video, we demonstrate the result of the Kalman filter, the system identification, the state machine, two robots playing the game autonomously in simulation, and the hitting performance on the real robot. Demonstrations can be found in the video.

A. Hitting Performance

In Fig. 5a we compare the trajectories obtained using only inverse kinematics on the planned path with the ones optimized by LP + NLP + AQP. The proposed optimization makes better use of the velocity range of each joint while obtaining similar joints paths. This technique exploits the null space manifold such that we avoid exceeding the velocity bounds. In the shown example, instead of overshooting the velocity limit on joint 2, the optimized trajectory apportions the velocity to other joints. We can also observe that multiple joints are reaching the velocity limit at the same time, which means robot capability gets exploited.

The violin plot in Fig. 5b shows the performance of each component in the proposed optimization. We selected three hitting types i.e., hitting directly towards the goal, make a bounce towards the opposite side (reverse bounce), or the same side of the puck (forward bounce). For each hitting type, we evaluate the movement on a grid of 180 puck hitting positions. The trajectories are planned with the time step size of 0.01s. In Table II is reported the elapsed time statistics for every algorithm. For every hitting point, we perform up to 10 trials of hitting optimization. For QP, the initial attempting velocity is set to be $2m/s$, other methods use the velocity solved from Section III-D. If the optimization succeeds, we stop the optimization and obtain the calculation time. If the optimization fails, the velocities are scaled down by a factor of 0.9 and retry. If all trials fail, we record the hitting velocity as zero and the optimization time at termination.

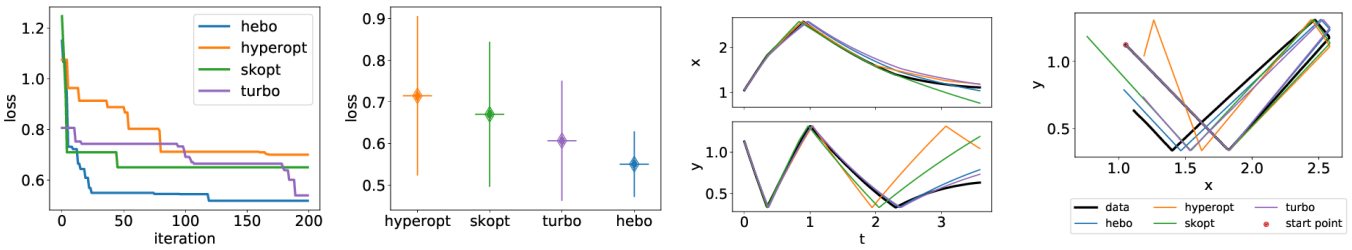
Method	Mean	Min	Max	Median
QP	263.58	30.65	531.73	243.96
NLP + AQP	132.88	23.03	284.54	126.07
LP + NLP + AQP	234.26	43.26	498.84	222.01

TABLE II: Optimization time (*ms*) for different methods

From the violin plots, it is clear that the initial nonlinear optimization (NL) (4) together with the anchored quadratic programming (AQP) (6) drastically improves the chances of successfully compute a hitting trajectory compared to the pure quadratic programming (QP) approach using (3). We record no hitting failure in the tested position with NL. The linear programming step (LP) shown in (5) to compute the maximum velocity allows us to find fast-hitting solutions at the cost of double computation time. This higher computation time is mostly due to the failure of optimization, because there is no feasible solution to reach the desired velocity following the Cartesian path. By employing a reasonable prediction time, e.g., 1.5 seconds, it is possible to use any of the proposed algorithms to hit the puck, as the optimization, except for the most extreme positions, is sufficiently fast to allow the hitting movement of the robot. This issue is not crucial in practical cases: when the optimization takes longer it is not possible to obtain a high hitting velocity therefore it may not be worth trying the hitting movement in the first place. This fact allows us to play the game reactively with the robot, as proved in simulation.

B. System Identification

We optimize the parameters for 200 iterations. At each iteration, we sample 5 candidate parameter vectors from the acquisition function and we evaluate the loss over the whole dataset. In our experiments, HEBO significantly outperformed all other BO algorithms, leading to trajectories that are closer to the ones in the training dataset, as is shown



(a) Learning curve

(b) Best results over 3 runs, with 95% confidence intervals

(c) Example trajectory in time

(d) Example trajectory in x-y plane

Fig. 6: Results for Bayesian optimization.

in Fig. 6c and 6d. From the learning curve in Fig. 6a, it is clear that not only HEBO finds a better solution faster than the other algorithms, but also the samples it selects are on average better than the ones chosen by competing algorithms. We also observe superior performances when averaging among different independent runs, as presented in Fig 6b. While the results are sufficiently good to be used in a simulation, there is still some discrepancy between the real trajectory and the simulated one. This discrepancy is unavoidable since it's impossible to have a perfect model of the air hockey table. Indeed, irregularities of the plane and air flow play a major role in the system, particularly at slow velocities, as Fig. 6c shows.

VII. CONCLUSION

In this paper, we studied and presented a robotic system that is able to play air hockey, a simple, but challenging dynamic task. We proved that commercially available general-purpose industrial manipulators, such as the Kuka Iiwa LBR 14, can perform strong hits. We proposed an optimization method that exploits the redundancy of the manipulator coping with joint velocity limits. In the simulated environment, we showed two robots playing against each other successfully, while our optimization algorithm runs in real-time on a desktop machine. We presented a simple framework, based on BO, to tune the parameters of the environment's dynamic model. This framework only requires black-box access to a simulator: it doesn't require any further knowledge of the system.

This work opens many interesting research lines. An interesting research direction is to perform an in-depth study of BO techniques applied to non-differentiable dynamics, such as the collision, and compare with standard techniques. Another research direction would be to improve the optimization by considering dynamics properties, and not only kinematics. Finally, we need to test the whole game on the real system to demonstrate the capabilities of existing systems and our approach.

ACKNOWLEDGMENT

This project was supported by the CSTT fund from Huawei Tech R&D (UK). The support provided by China Scholarship Council (No. 201908080039) is acknowledged.

REFERENCES

[1] C. B. Partridge and M. W. Spong, "Control of planar rigid body sliding with impacts and friction," *International Journal of Robotics Research (IJRR)*, vol. 19, no. 4, pp. 336–348, 2000.

[2] A. Namiki, S. Matsushita, T. Ozeki, and K. Nonami, "Hierarchical processing architecture for an air-hockey robot system," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[3] H. Shimada, Y. Kutsuna, S. Kudoh, and T. Suehiro, "A two-layer tactical system for an air-hockey-playing robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[4] A. AlAttar, L. Rouillard, and P. Kormushev, "Autonomous air-hockey playing cobot using optimal control and vision-based bayesian tracking," in *International Conference Towards Autonomous Robotic Systems (TAROS)*, 2019.

[5] B. E. Bishop and M. W. Spong, "Vision based control of an air hockey playing robot," *IEEE Control Systems Magazine*, vol. 19, no. 3, 1999.

[6] H. Alizadeh, H. Moradi, and M. N. Ahmadabadi, "Automatic calibration of an air hockey robot," in *International Conference on Robotics and Mechatronics (ICRoM)*, pp. 107–112, IEEE, 2013.

[7] K. Igeta and A. Namiki, "Algorithm for optimizing attack motions for air-hockey robot by two-step look ahead prediction," in *IEEE/SICE International Symposium on System Integration*, pp. 465–470, 2017.

[8] A. Taitler and N. Shimkin, "Learning control for air hockey striking using deep reinforcement learning," in *International Conference on Control, Artificial Intelligence, Robotics Optimization*, 2017.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] A. Xie, D. P. Losey, R. Tolsma, C. Finn, and D. Sadigh, "Learning latent representations to influence multi-agent interaction," in *Conference on Robot Learning (CoRL)*, 2020.

[11] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[12] J. Močkus, "On bayesian methods for seeking the extremum," in *Optimization techniques IFIP technical conference*, Springer, 1975.

[13] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, 1964.

[14] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[15] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

[16] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann, "Manipulability analysis," in *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, pp. 568–573, IEEE, 2012.

[17] A. I. Cowen-Rivers, W. Lyu, R. Tutunov, Z. Wang, A. Grosnit, R. R. Griffiths, H. Jianye, J. Wang, and H. B. Ammar, "An empirical study of assumptions in bayesian optimisation." unpublished.

[18] P. Kumaraswamy, "A generalized probability density function for double-bounded random processes," *Journal of hydrology*, 1980.

[19] G. E. Box and D. R. Cox, "An analysis of transformations," *Journal of the Royal Statistical Society: Series B (Methodological)*, 1964.