Autonomous Air-Hockey Playing Cobot Using Optimal Control and Vision-Based Bayesian Tracking

Ahmad AlAttar(✉), Louis Rouillard, and Petar Kormushev

Robot Intelligence Lab, Imperial College London, London, UK
{a.alattar19,louis.rouillard-odera17,p.kormushev}@imperial.ac.uk
https://www.imperial.ac.uk/robot-intelligence/

Abstract. This paper presents a novel autonomous air-hockey playing collaborative robot (cobot) that provides human-like gameplay against human opponents. Vision-based Bayesian tracking of the puck and striker are used in an Analytic Hierarchy Process (AHP)-based probabilistic tactical layer for high-speed perception. The tactical layer provides commands for an active control layer that controls the Cartesian position and yaw angle of a custom end effector. The active layer uses optimal control of the cobot's posture inside the task nullspace. The kinematic redundancy is resolved using a weighted Moore-Penrose pseudo-inversion technique. Experiments with human players show high-speed human-like gameplay with potential applications in the growing field of entertainment robotics.

Keywords: Air hockey · Cobot · Bayesian tracking · Analytic Hierarchy Process · Autonomous robot · Entertainment robotics

1 Introduction

Robots are expanding from strictly industrial applications to closer interactions with humans. Two major applications of such close encounters are in home-care robotics [5] and entertainment robotics [10], which naturally raises the question of how to make user interaction with robots both enjoyable and safe. In this paper, we focus on air hockey which is a challenging example of an arcade game where humans can physically play against robots. An air-hockey table is a constrained 2D environment which is ideal for testing high-speed robot motion planning. However, strategies to master the game involve trajectory prediction, high puck speed, and uncertainty management. Therefore, air hockey presents an easy-to-learn yet hard-to-master task for robots to perform effectively against a skilled opponent.

In this work, we program a Panda collaborative robot (cobot) arm, hereinafter referred to as the *Panda arm*, to play air hockey, as shown in Fig. 1. This cobot is relatively new to robotics research, only released in 2017 by

K. Althoefer et al. (Eds.): TAROS 2019, LNAI 11650, pp. 358–369, 2019.
https://doi.org/10.1007/978-3-030-25332-5_31

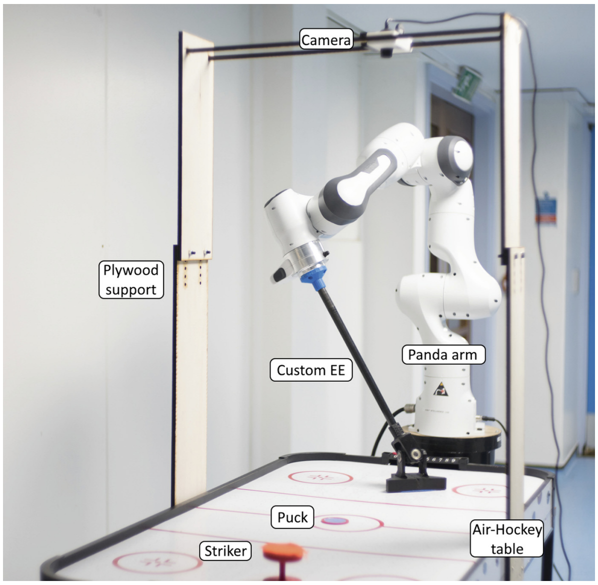

Fig. 1. General setup for the air-hockey task. A Sony PlayStation Eye camera is fixed on top of the table using a plywood structure. The puck and striker are colored with bright tones for ease of filtering. The Panda arm stood at the far side of the table with its custom-made end effector.

Franka Emika GmbH [6]. Recent works have implemented robotic entertainment applications using different approaches and other robotic arms. In [10], a dual system composed of a tactics layer governing patterns of gameplay and a skill layer governing individual shots is introduced with a custom robotic arm. In [8], a Barett WAM arm is used with a three-layer system: motion control, short-term strategy and long-term strategy - adapting its playstyle to the opponent. Other studies also used various control strategies such as learning from movement primitives [1], task-switching [2], with fuzzy control [14], using weak points of the human vision [9], and more recently using Deep Reinforcement Learning [12].

This paper proposes a novel architecture for air-hockey playing cobots using Bayesian filtering for perception, nullspace-based posture optimization for robot motion, and a combined Analytic Hierarchy Process (AHP) and probabilistic tactical framework for human-like behavior. The proposed architecture comprises two layers: (1) a tactical layer, implementing the high-speed perception, motion prediction, and strategy planning; and (2) an active layer, implementing the robot movement execution via optimal control. The two layers are introduced in Sects. 2 and 3, followed by experimental results in Sect. 4.

2 Tactical Layer: Determination of Cobot Actions

In a competitive game, the puck can cross the entire length of the air-hockey table in a mere hundred milliseconds, which is similar in speed to the human vision reflex [13]. A human player needs to predict the trajectory of the puck using the movement of the opponent's striker and the configuration of the table, which is implemented for the cobot using a Bayesian framework that mimics the decision-making process of humans [4,16].

2.1 Computer Vision: Kalman Filtering-Based Predictive Approach

A Sony PlayStation Eye camera with a (320×240) pixels resolution and a 187 Hz frame rate is used to detect the position of the puck at a high speed. The 2D table frame is represented by a rectangle $(W \times L)$ $(width \times length)$ associated with a Cartesian coordinate system (x, y). Calibration between the air-hockey table frame and the camera frame is done using a least-squares method. Moreover, a 2D affinity that optimally matched the (x^c, y^c) camera frame to the (x^r, y^r) cobot frame is obtained, having the form $\left[x^r \; y^r \; 1\right]^T = S\left[x^c \; y^c \; 1\right]^T$ where S is a homogeneous transform. A method based on the Moore-Penrose pseudo inverse is used to determine the coefficients of S along with training points from the camera frame and the cobot frame [7].

A flowchart representing the general functioning of the tactical layer is shown in Fig. 2. Positions of the puck and the human player's striker (hereon referred to as the *striker*) are obtained using color filtering and contouring. The striker and the puck are tracked to be able to predict the puck trajectory before it is hit by the human player, so that the cobot has a longer time window to move before the puck arrives at a position within its reach. For this matter, Kalman filtering [11] is used as it provides a simple framework to deal with state prediction and uncertainty under a Gaussian assumption:

$$\overrightarrow{X_\mu} = \left[x \; y \; v_x \; v_y\right]^T, \quad C = \begin{bmatrix} 1 \, 0 \, 0 \, 0 \\ 0 \, 1 \, 0 \, 0 \end{bmatrix}, \quad \overrightarrow{Y} = C\overrightarrow{X_\mu}$$

Where x, y is Cartesian position, v_x, v_y is Cartesian speed, $\overrightarrow{X_\mu}$ is the object's state vector expected value, C is the measurement transition matrix and $\overrightarrow{Y}$ is the measurement vector. Along with $\overrightarrow{X_\mu}$, the state of the object is associated with the co-variance matrix Σ. Both define a normally distributed random variable. Acceleration is ignored as it is modified frequently and non-linearly by external sources (mainly human and cobot actions). At any given time in the future, the predicted position of the object can be obtained by repetitively applying the operations:

$$\overrightarrow{X_\mu}(t + dt) = A\overrightarrow{X_\mu}(t) \tag{1}$$

$$\Sigma(t + dt) = A\Sigma(t)A^T \tag{2}$$

where dt is the time-step since the last measurement and A is the state transition matrix. Equation (1) returns a position in $\mathbb{R}^2$ that neglects puck bounces against the table walls. For those bounces, we make the assumption of a perfectly elastic model with infinite inertia for the walls compared to the puck.

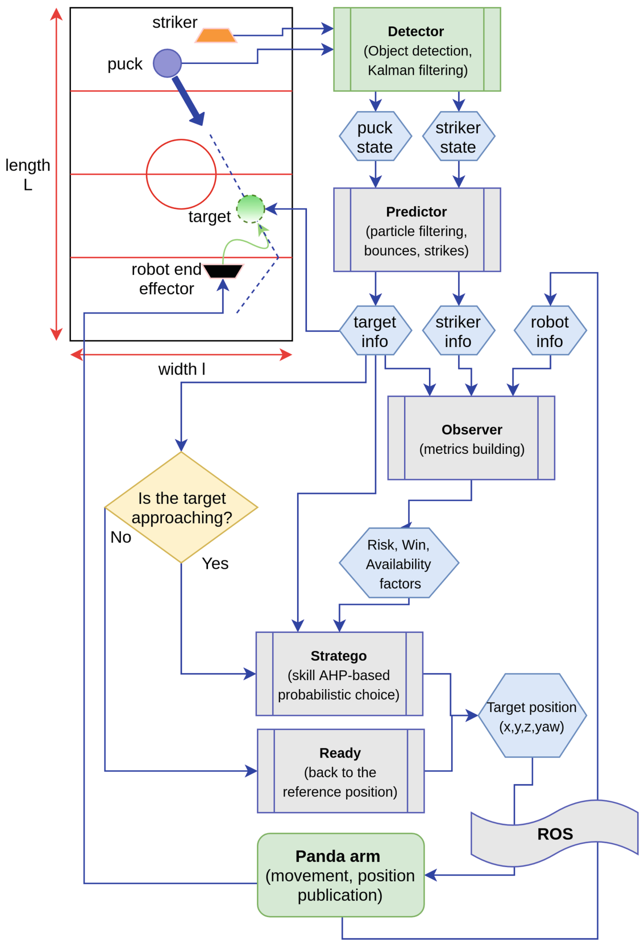
striker
puck
length L
target
robot end effector
width l
Detector (Object detection, Kalman filtering)
puck state
striker state
Predictor (particle filtering, bounces, strikes)
target info
striker info
robot info
Observer (metrics building)
Is the target approaching?
No
Yes
Risk, Win, Availability factors
Stratego (skill AHP-based probabilistic choice)
Ready (back to the reference position)
Target position (x,y,z,yaw)
ROS
Panda arm (movement, position publication)

Fig. 2. General tactical layer flow chart.

The table is modelled with two walls at $y = 0$ and $y = W$. For a single bounce on the $y = W$ wall, for example, this results in these equations:

$$\overrightarrow{X_\mu} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \overrightarrow{X_\mu} + \begin{bmatrix} 0 \\ 2W - y \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

$$\Sigma \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \Sigma \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}^T \tag{4}$$

Equation (3) introduces a non-linear operation, which is not suitable for the Kalman framework. To deal with this limitation, we use a variant of the particle filter [3] as follows:

- We sample a population $\overrightarrow{X_i}$, $i = 1..N$, $N > 1000$ from the normal distribution defined by $\overrightarrow{X_\mu}$ and Σ
- For each particle $\overrightarrow{X_i}$ we repetitively apply Eqs. (1) and (3) for future predictions
- The future state expected value and co-variance can be obtained as:

$$\overrightarrow{X}_{\mu\, fut.\, st.} = \frac{1}{N}\sum_{k=0}^{N}\overrightarrow{X_i} = \overrightarrow{X}', \quad \Sigma_{fut.\, st.} = \frac{1}{N}\sum_{k=0}^{N}(\overrightarrow{X_i} - \overrightarrow{X}') * (\overrightarrow{X_i} - \overrightarrow{X}')^T$$

Estimates for future states when the puck is close to the table limits are theoretically improved by this method when $N \to \infty$. A quicker estimate can always be obtained by simply applying Eqs. (3) and (4) to the expected value of the state at time $(t + n \times dt)$ for $n > 0$.

To deal with puck-striker contacts, we make the assumption of an infinite striker inertia compared to the puck inertia. With $\overrightarrow{X}_{puck}$ and Σ_{puck} referring to the puck's state and $\overrightarrow{X}_{striker}$ referring to the striker's state we reason as follows:

- We have the puck and striker states evolve with Eqs. (1) and (2) until either a time limit is reached or the distance between puck and striker is inferior to the sum of their respective radii;
- If the latter is true, the surface between the two objects is defined by its normal unit vector $\left[\cos(\theta)\ \sin(\theta)\right]^T$:

$$\overrightarrow{X}^{s.\,ref}_{puck} = \overrightarrow{X}_{puck} - \overrightarrow{X}_{striker}, \quad \theta = \operatorname{atan2}(\overrightarrow{X}^{s.\,ref}_{puck}[1], \overrightarrow{X}^{s.\,ref}_{puck}[0])$$

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & \cos(\theta) & -\sin(\theta) \\ 0 & 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

- The puck then elastically bounces on the surface:

$$M = R_\theta * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * R_\theta^{-1}$$

$$\overrightarrow{X}_{puck} \leftarrow \overrightarrow{X}_{striker} + M * \overrightarrow{X}^{s.\,ref}_{puck}, \quad \Sigma_{puck} \leftarrow M * \Sigma_{puck} * M^T$$

By combining all the previous points, we update the estimated puck state vector using camera measurements. Using this updated state, we then predict the puck's future state expected value and co-variance matrix at any given time by taking into account bounces on the walls and human player's potential strikes. This provides an effective online state-estimate filtering framework, with measures of uncertainty in future predictions which is useful for strategy-making.

2.2 Strategy-Making: AHP-Based Probabilistic Behavior

The filtered state and equations specified in the previous section are sufficient to determine the set of future positions and velocities for the puck up to a certain point. Performing a Principal Component Analysis (PCA) on the co-variance matrix of the puck's position determines the first principal component: the corresponding eigenvalue is taken as a measure of the uncertainty of the prediction, and the eigenvector is taken as the Cartesian direction of maximum uncertainty. Combining those elements, among the set of future puck positions, we find the closest to the current cobot end-effector position $\overrightarrow{x}_{cobot}$. We name this optimal position $\overrightarrow{x}_{target}$ and its corresponding speed $\overrightarrow{v}_{target}$. We also determine the corresponding time of arrival, how uncertain that prediction is, and the safest axis on which we maximize the probability to reach the puck, i.e. the direction defined by the aforementioned eigenvector. This set of variables is hereon referred to as the *target*. To obtain diverse gameplay, following the methods of [10] we built a set of skills, standard actions that depend on the cobot and puck relative positions. This ensemble is hereon referred to as the *skillset*:

– Ready: the cobot returns to its own goal position;
– Defend: the cobot moves to $\overrightarrow{x}_{target}$ and stops;
– Stop: the cobot intercepts the puck at $\overrightarrow{x}_{target}$ with a speed $0.5\overrightarrow{v}_{target}$
– Counter: the cobot intercepts the puck at $\overrightarrow{x}_{target}$ with a speed $-2\overrightarrow{v}_{target}$
– Cut: the cobot intercepts the puck at $\overrightarrow{x}_{target}$ from $\overrightarrow{x}_{cobot}$
– Smash: the cobot strikes the puck at $\overrightarrow{x}_{target}$ towards the opponent's goal
– Rally: the cobot strikes the puck at $\overrightarrow{x}_{target}$ towards the human player's goal with a bounce on one of the walls

Once this skillset is defined, we then specify at any point during the game what kind of event should trigger the execution of a skill by the cobot. For this purpose, we explored different possible trigger events, for instance human player's strikes, bounces on the walls, and time elapsed. Most of them resulted in an erratic behavior for the cobot, either not reacting to an obvious incoming shot or continuously triggering inconsistent skills. We thus settle on a simple event: when $\overrightarrow{x}_{target}$ is predicted to be on the cobot's side of the table with a speed not indicating a movement towards the other side. Once the cobot decides that an action needs to be taken, we then specify a way to discriminate the skills in the skillset to select the most relevant. We do not systematically select the most relevant skill to provide the cobot's gameplay with some unpredictability. To discriminate between the skills, we combine the frameworks of [8] and [10]:

– We define three normalized metrics continuously updated during the game by combining at any given time the target, cobot, and striker information: risk factor $\mathcal{R}$ (assessing the risk for the cobot to concede a goal), win factor $\mathcal{W}$ (assessing the possibility to score a goal) and availability factor $\mathcal{A}$ (assessing how easy it is to reach the target and how much the prediction can be trusted). For each factor, 0 indicates a bad context for the cobot and 1 a good context.

- As in [8], using an AHP [15], we define a static Pairwise Comparison Matrix (PCM), describing the relative preference for a skill compared to another one according to a given criteria ($\mathcal{R}$, $\mathcal{W}$ or $\mathcal{A}$). In our case, we define three (7×7) matrices Φ_R, Φ_W, Φ_A respectively associated to the (7×1) normalized first principal components $\overrightarrow{\varphi}_R$, $\overrightarrow{\varphi}_W$, $\overrightarrow{\varphi}_A$.
- When an event triggers the choice for a skill at time t, each skill is assigned to a probability specified by the (7×1) probability vector $\overrightarrow{\mathbb{P}}$ as follow:

$$\mathcal{S}(t) = \mathcal{R}(t) + \mathcal{W}(t) + \mathcal{A}(t)$$
$$\mathcal{R}'(t) = \frac{\mathcal{R}}{\mathcal{S}} \;\; \mathcal{W}'(t) = \frac{\mathcal{W}}{\mathcal{S}} \;\; \mathcal{A}'(t) = \frac{\mathcal{A}}{\mathcal{S}}$$
$$\overrightarrow{\mathbb{P}}(t) = \begin{bmatrix} \vdots & \vdots & \vdots \\ \overrightarrow{\varphi}_R & \overrightarrow{\varphi}_W & \overrightarrow{\varphi}_A \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \mathcal{R}' \\ \mathcal{W}' \\ \mathcal{A}' \end{bmatrix}$$

- As in [10], a skill is then randomly sampled from the skillset according to $\overrightarrow{\mathbb{P}}$ and executed by the cobot. The skill is stopped when the predicted time of arrival of the puck at the target is reached. If the puck now moves towards the human's side of the table, the cobot performs a Ready skill, otherwise a trigger event runs the previous steps again.

This framework uses the rich information provided by the target to select a skill in a considered yet randomized way among several possibilities. The uncertainty management and unpredictability resulting from this approach mimic human behavior [4,16], creating an interesting and variable gameplay.

3 Active Layer: Implementation of Cobot Actions

The active layer in the proposed architecture implements the robot movement execution via optimal control. The main goal is to maximize the end-effector velocity without exceeding the individual joint-level speed limits. Compared to setups in [10] or [14], the usage of a 7-DoF robot may be suboptimal since the workspace is only a 2D plane. However, the Panda arm's safe interaction with users makes it a suitable option for safe gameplay in close human proximity. Note that the z-dimension (normal to the air-hockey table) and the roll, pitch and yaw orientations are not constrained by design and need to be dealt with inside the controller.

3.1 Software Implementation: Translation of PD Commands into Joint Velocities

Desired cobot end-effector (EE) Cartesian positions are sent by the tactical layer through ROS at a frequency of 500 Hz. A real-time Linux kernel was used to minimize the control latency. Four dimensions are being controlled as described in the next section. These dimensions represent the (x_d, y_d) desired position in the air-hockey table frame, the z_d dimension that needs to be fixed so that the EE stays in contact with the table, and the orientation of the EE in the table frame that is equivalent to its $yaw = \alpha_d$ angle.

The Panda arm's library allows the return of a (4×4) transformation matrix T^0_{EE}, linking the EE frame to the cobot reference frame. Using this matrix, the cobot current position $(x_c, y_c, z_c, \alpha_c)$ is:

$$x_c = T^0_{EE}[0,3], \quad y_c = T^0_{EE}[1,3], \quad z_c = T^0_{EE}[2,3], \quad \alpha_c = \text{atan2}(T^0_{EE}[1,0], T^0_{EE}[0,0])$$

The difference between desired and current Cartesian position serves as an input to a PD controller to obtain a desired Cartesian acceleration $\ddot{\overrightarrow{\mathbb{X}}}$. Using a custom symmetric positive definite weighting matrix W we then construct the equivalent Cartesian impedance matrix Λ and the Jacobian Moore-Penrose weighted pseudo inverse ${J_4}^\dagger$ [7]:

$$\Lambda = (J_4 W J_4^T)^{-1}, \quad J_4^\dagger = W J_4^T \Lambda$$

Where J_4 is a (4×7) EE Jacobian matrix with the four (x, y, z, α) dimensions. Supposing that we know the (7×1) joint velocity vector $\dot{\overrightarrow{\mathbb{Q}}}$ at the previous timestep, we can derive the joint acceleration $\ddot{\overrightarrow{\mathbb{Q}}}$ minimizing the equivalent kinetic energy using W as a mass matrix [7]:

$$\dot{\overrightarrow{\mathbb{X}}} = J_4 \dot{\overrightarrow{\mathbb{Q}}} \implies \ddot{\overrightarrow{\mathbb{X}}} = \dot{J}_4 \dot{\overrightarrow{\mathbb{Q}}} + J_4 \ddot{\overrightarrow{\mathbb{Q}}}, \quad \ddot{\overrightarrow{\mathbb{Q}}}_{move} = J_4^\dagger (\ddot{\overrightarrow{\mathbb{X}}} - \dot{J}_4 \dot{\overrightarrow{\mathbb{Q}}})$$

Compared to a situation where we control all six Cartesian dimensions to operate a movement on the (x, y, z, α) dimensions, using the custom weight matrix W and the three redundant DoF of the Panda arm allows a better distribution of the joint speed, further away from upper limits that would cap the EE movement velocity.

However, this local optimization process can result in long movements, such as drifting of the arm joint positions (for instance towards positions minimizing the gravitational potential energy), that ultimately reach joint limits, thus stopping the process. To prevent this drifting, given the arm joint positions $(q_1, ..., q_7)$, we implement a gradient descent optimization with damping on the cost function Ω inside the three DoF nullspace:

$$\Omega = \frac{1}{2} \sum_{i=1}^{7} (q_i(t) - q_i^0)^2, \quad \overrightarrow{\nabla} \Omega = \begin{bmatrix} q_1 - q_1^0 \\ \vdots \\ q_7 - q_7^0 \end{bmatrix}, \quad \overrightarrow{\Psi} = -\mathcal{P}_n \overrightarrow{\nabla} \Omega - \mathcal{D}_n \dot{\overrightarrow{\nabla}} \Omega$$

where $(q_1^0, ..., q_7^0)$ are chosen so that the arm kept a straight position as much as possible, and that $\mathcal{P}_n$, $\mathcal{D}_n$ created a slow, non-oscillating, non-overshooting movement inside the nullspace, disturbing as less as possible the EE movement.

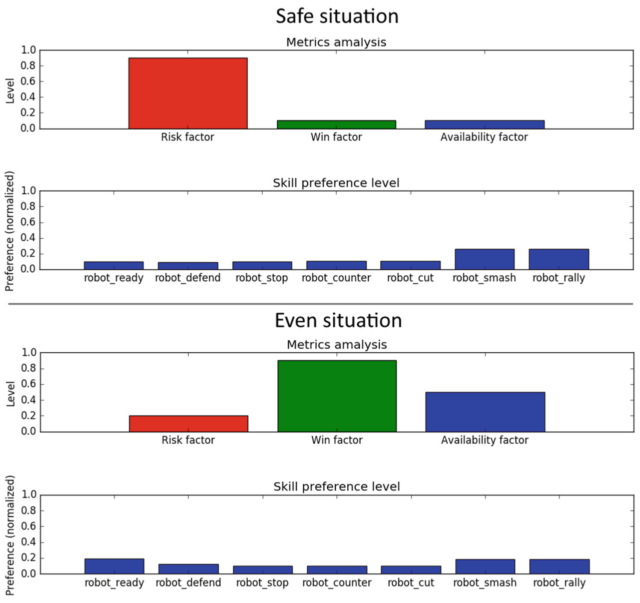

Fig. 3. Comparison of skillset weighing in a typically safe (top) and more even (bottom) gamewise situation. The skill weighing varies in the 2 situations yet we can observe a rather equal distribution in both cases, which guarantees an unpredictable gameplay.

3.2 Hardware Implementation: Universal-Joint Based Striker

As explained in the previous section, we choose to limit the EE control to the four (x, y, z, α) dimensions. To maximize the cobot EE Cartesian speed with a capped set of joint velocities, we choose to build a custom, partially 3D-printed EE that transformed a typical six DoF "full-body" movement into a four DoF "wrist-like" movement as detailed in Fig. 4. To linearly multiply the EE speed at a given joint rotation speed, we base our design around a 46 cm-long carbon fiber rod. The tip of the rod, considered as the EE, is linked to the flange frame by the transformation matrix T_{EE}^{flange}.

The transformation T_{EE}^{flange} can be passed (along with a mass and inertia matrix) to the Panda arm internal models to automatically adapt kinetic and dynamic functions (notably for the Jacobian calculations used in the previous section). With this design, by slightly orienting and translating the flange of the Panda arm, we obtained an amplified EE movement at the tip of the rod. Moreover, to transmit $\alpha = yaw$ movements on the cobot striker (to orient it inside the air-hockey table frame), we use a 3D-printed combination of a striker with a universal joint (see Fig. 4B): rotations around the carbon rod axis are fully transmitted to the cobot striker's α while keeping *roll* and *pitch* dimensions free. The latter are mechanically constrained by the design to always keep the cobot striker's surface parallel to the air-hockey table. As the z dimension of the cobot EE is fixed in the controller, and with a rather large flat surface for the striker,

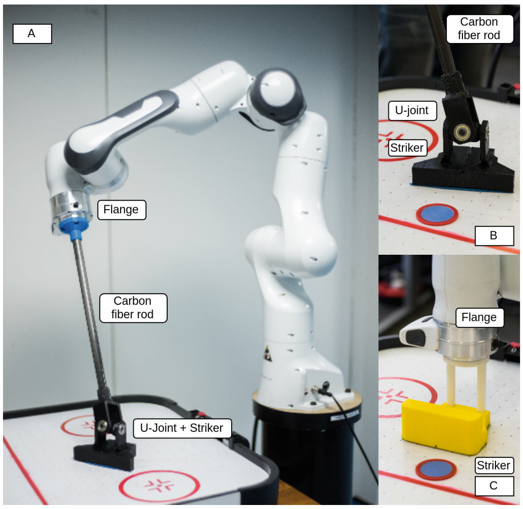

Fig. 4. Overview of the custom end effector. (A) General view of the Panda arm. (B) Close view on the universal joint/striker combination. (C) For comparison purpose, close view on a "full body motion" striker prototype linked to the Panda flange: the whole Panda arm mass has to be moved to displace the striker.

the system naturally adapted its *roll* and *pitch* to the controlled *yaw* dimension. This mechanical design helps the software minimize the joint velocity for a given Cartesian EE speed by delegating some dimensions' control to the mechanics, therefore liberating DoF for the software optimization.

4 Experimentation Results and Analysis

During the experimentation phase, the cobot demonstrates robust gameplay, able to adapt to a majority of situations[1]. The main limitations lie in the hardware side and include difficulty to perform EE impulsive movements necessary for strong strikes and vibrations of the actuators at very high frequency due to constant small re-adaptations of the desired EE position. The efficacy of some of the frameworks introduced in this work is assessed in this section.

4.1 Strategy Making: Evolution of Skillset Usage with Game State

The combination of the AHP framework with a probabilistic choice help create a varied yet logical gameplay as represented in Fig. 3. We deliberately choose not to implement any pairwise strong difference in the *PCM* matrices, as this can create a rather deterministic gameplay, but this factor can be tuned to adjust the

[1] A video demonstration of the system is available at: https://www.imperial.ac.uk/robot-intelligence/videos/.

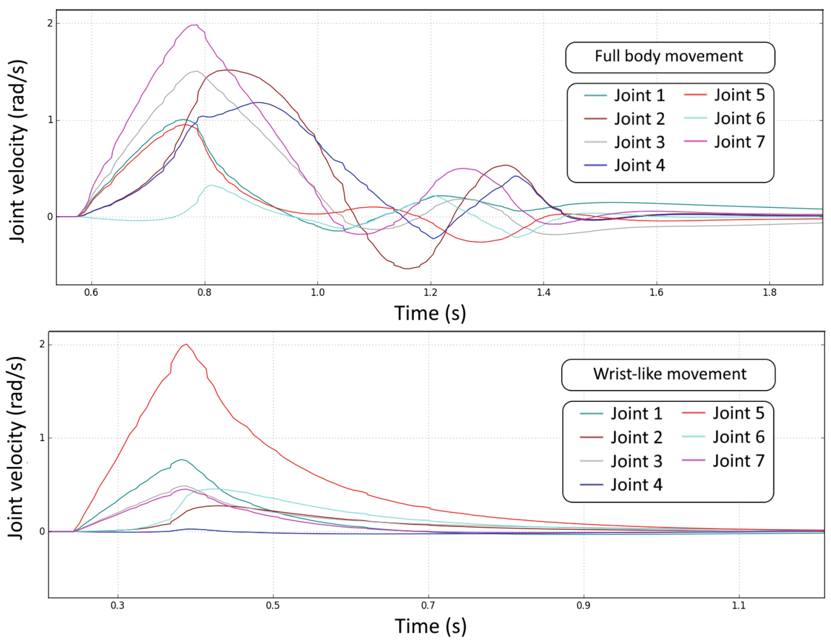

Fig. 5. For a given arm +30 cm translation on the y axis, comparison of joint velocity between (top) a fully constrained "full-body" motion and (bottom) a partially freed "wrist-like" movement. "Wrist-like" movement allows for a more evenly distributed joint velocity, preventing some joints to reach their velocity limit.

efficiency of the cobot. This result validates the similar approach performed in [10] by showing a comparable evolution of skill choice weighing depending on the game situation. We, however, propose AHP as a more unified and transparent way to assign probability weights regarding different criteria.

4.2 Active Layer: Preparation of Joint Velocity

The combination of the custom universal-joint based EE hardware and the software weighted Moore-Penrose technique yield a better distribution of the joint velocities as shown in Fig. 5. For the "wrist-like" movement, we can observe that the majority of the joint speed is limited to small values away from joint limits, with no unnecessary compensation of rotation between different joints. This gives rise to a higher potential for faster more complex movements of the EE without risking any instability or process breaking issues.

5 Conclusion

In this work, an air-hockey playing cobot, both safe and challenging for human opponents, is introduced. Inherent uncertainties in the predictions, as a by-product of Kalman filtering, induced a more natural process of state prediction. Along with the AHP-based probabilistic technique that is used for skill selection, this architecture is a step forward towards a more human-like gameplay in the expanding field of entertainment robotics. Throughout this research, we found that the limited joint speeds of the arm combined with the constrained environment were two limitations hard to overcome for a high-speed high-frequency application such as air hockey. Nonetheless, this research highlights the potential application of cobots, such as the Panda arm, for *safe* human interactions in entertainment robotics.

References

1. Bentivegna, D.C., Atkeson, C.G.: A framework for learning from observation using primitives. In: Kaminka, G.A., Lima, P.U., Rojas, R. (eds.) RoboCup 2002. LNCS (LNAI), vol. 2752, pp. 263–270. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45135-8_20
2. Bishop, B., Spong, M.: Vision-based control of an air hockey playing robot. IEEE Control Syst. Mag. **19**(3), 23–32 (1999)
3. Djuric, P.M., et al.: Particle filtering. IEEE Signal Process. Mag. **20**(5), 19–38 (2003)
4. Ernst, M.O., Banks, M.S.: Humans integrate visual and haptic information in a statistically optimal fashion. Nature **415**, 429 EP (2002)
5. Falck, F., Doshi, S., Smuts, N., Lingi, J., Rants, K., Kormushev, P.: Human-centered manipulation and navigation with Robot DE NIRO. In: IROS 2018 Workshop: Towards Robots that Exhibit Manipulation Intelligence, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, October 2018
6. Franka Emika GmbH: Panda arm (2017). https://www.franka.de/
7. Nakamura, Y.: Advanced Robotics: Redundancy and Optimization, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (1990)
8. Namiki, A., Matsushita, S., Ozeki, T., Nonami, K.: Hierarchical processing architecture for an air-hockey robot system. In: 2013 IEEE International Conference on Robotics and Automation, pp. 1187–1192. IEEE (2013)
9. Ogawa, M., et al.: Development of air hockey robot improving with the human players. In: IECON 2011, pp. 3364–3369 (2011)
10. Shimada, H., Kutsuna, Y., Kudoh, S., Suehiro, T.: A two-layer tactical system for an air-hockey-playing robot. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6985–6990. IEEE (2017)
11. Sorenson, H.: Kalman Filtering Techniques, Advances in Control Systems, vol. 3. Elsevier, Amsterdam (1966)
12. Taitler, A., Shimkin, N.: Learning control for air hockey striking using deep reinforcement learning. In: 2017 International Conference on Control, Artificial Intelligence, Robotics Optimization (ICCAIRO), pp. 22–27 (2017)
13. Thorpe, S., Fize, D., Marlot, C.: Speed of processing in the human visual system. Nature **381**, 520 EP (1996)
14. Wang, W.J., Tsai, I.D., Chen, Z.D., Wang, G.H.: A vision based air hockey system with fuzzy control. In: Proceedings of the International Conference on Control Applications, vol. 2, pp. 754–759 (2002)
15. Whitaker, R.: The analytic hierarchy process - what it is and how it is used. Math. Model. **9**, 161–176 (1987)
16. Wolpert, D.M., Flanagan, J.R.: Motor prediction. Curr. Biol. **11**(18), R729–R732 (2001)