robotics

Article

Robot Anticipation Learning System for Ball Catching †

Diogo Carneiro, Filipe Silva and Petia Georgieva *

Department of Electronics, Telecommunications and Informatics, Institute of Electronics and Informatics Engineering of Aveiro (IEETA), University of Aveiro, 3810-193 Aveiro, Portugal; diogo.carneiro@ua.pt (D.C.); fmsilva@ua.pt (F.S.)

* Correspondence: petia@ua.pt

† This paper is an extended version of our paper published in Carneiro, D.; Silva, F.; Georgieva, P. The role of early anticipations for human-robot ball catching. In Proceedings of the 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Torres Vedras, Portugal, 25–27 April 2018.

Abstract: Catching flying objects is a challenging task in human–robot interaction. Traditional techniques predict the intersection position and time using the information obtained during the free-flying ball motion. A common pain point in these systems is the short ball flight time and uncertainties in the ball's trajectory estimation. In this paper, we present the Robot Anticipation Learning System (RALS) that accounts for the information obtained from observation of the thrower's hand motion before the ball is released. RALS takes extra time for the robot to start moving in the direction of the target before the opponent finishes throwing. To the best of our knowledge, this is the first robot control system for ball-catching with anticipation skills. Our results show that the information fused from both throwing and flying motions improves the ball-catching rate by up to 20% compared to the baseline approach, with the predictions relying only on the information acquired during the flight phase.

Keywords: human–robot interaction; ball catching; trajectory prediction; anticipation learning; neural network

Citation: Carneiro, D.; Silva, F.; Georgieva, P. Robot Anticipation Learning System for Ball Catching. *Robotics* **2021**, *10*, 113. https://doi.org/10.3390/robotics10040113

Academic Editors: Ciro Natale and Marco Costanzo

Received: 31 July 2021
Accepted: 6 October 2021
Published: 15 October 2021

1. Introduction

The interception of moving objects is central to several benchmark robotic tasks such as ball-in-the-cup [1–3], batting [4–6], ball-catching [7–11], juggling [12–14] and playing table tennis [15–18]. This research focuses on demonstrating the combined performance of visual tracking systems and motion control algorithms in highly dynamic environments. In particular, catching a flying ball is a challenging task due to the demanding spatial–temporal constraints, which require coordination between visual, planning and control systems to get the hand to the right place at the right time [19]. While humans learn to accomplish this task with relative ease, this is still not the case for robotic systems, particularly in short-distance scenarios. The robot's actions to catch a flying ball are hampered by delays and noise in both sensors and actuators [20]. The sensory noise contributes to uncertainties in the ball's trajectory prediction and creates difficulties in the control of the robot's endpoint. Furthermore, frequent re-prediction of the target-catching location is required, as new observations become available. From the perspective of control, this progressive refinement of the desired catching point requires the online re-planning of robot motion, with object flight times that may last for around one second.

Traditional robot control systems for catching objects are built on visual information for the object in flight. Given the short flight time, only robot systems equipped with fast visual detection and tracking, sophisticated control architectures and computational power can perform the task successfully. To address this issue, this paper presents the Robot Anticipation Learning System (RALS) to predict feasible catching points in advance, using observations of the thrower's motion before the ball is released. These anticipation skills gain extra time for the robot hand to start moving in the targeted direction before the opponent finishes throwing.

This paper is an extension of the work that was originally presented in [21], where the role of early anticipation in human–robot ball catching was introduced. Here, we explain the RALS methodological framework in detail and we provide more comprehensive tests. Several computer simulations are conducted to demonstrate the effectiveness of the proposed solution in a scenario where the human subject throws a ball and the robot catches it. The experiments demonstrated that the proposed anticipation mechanism significantly improved the ball-catching rate compared to the baseline approach, where the predictions rely only on information acquired during the ball's flight phase.

In summary, the contributions of this paper are three-fold:

- RALS is developed, the first robot control system for catching flying objects with anticipation skills, using visual information from the thrower's hand motion.
- A learning mechanism is implemented to map the noisy vision information into a prediction of the ball's position and velocity at the moment of release.
- RALS was implemented and successfully evaluated for different levels of sensor noise and limits of the robot joint velocities.

The rest of the paper is organized as follows. Section 2 reviews related works and main concepts. The experimental setup is presented in Section 3. Section 4 presents the design of the RALS system. The ball-catching results with and without the anticipation mechanism are discussed in Section 5. Finally, conclusions are drawn in Section 6.

2. Related Work

The numerous approaches to solve the ball-catching problem differ in the complexity of the robot model, the way they predict the object's trajectory, and the adopted online trajectory generation method. Earlier works often show robots with few degrees-of-freedom (DOF) and simple motion generation algorithms. In the pioneering works of Slotine [22,23], a 4-DOF WAM arm (Barret Technology) and an active vision system with output information at 60 Hz were used. In these works, the catching point corresponds to the closest point of the ball's trajectory to the base of the robot, while the end-effector assumes a perpendicular orientation with respect to the ball's trajectory. The trajectory used to catch the ball is planned in Cartesian-space, using a 3rd-order polynomial function, which requires an inverse kinematics algorithm running in the control loop. The best performance results had a 70–80% success rate for similar launches.

Nishiwaki et al. [24] addressed both the falling-ball task and the ball-catching task using a humanoid robot, Saika, using an active vision system consisting of two-CCD cameras. In this work, the end-effector reached the catching point through an inverse kinematics model using a three-layered neural network model. The vertical trajectory of the ball was approximated by a quadratic function through a weighted least squares method, giving more importance to the most recent observations. Frese et al. [25] used a 7-DOF DLR-LWR-II arm equipped with a basket and an off-the-shelf stereo vision system that acquires and processes images at 50 Hz (a combination of two cameras). The catching point selection considered two criteria. First, a choice of location that is near to the robot's end-effector. Second, a catching point that is far away from the robot, to avoid physical constraints such as joint limits. The catching configuration was calculated to ensure a perpendicular orientation of the basket with respect to the ball's trajectory. In their experiments, the robot succeeded in 2/3 of its attempts to catch the ball. Most of the failures occurred due to the system's (camera and lens) limited horizontal field of view, resulting in delays in the visual system.

Studies in the following years frequently used more complex robots and advanced techniques for motion planning; human imitation has attracted significant interest. Riley and Atkeson [26] explored this idea to create human-like behaviours, encoded by movement primitives based on nonlinear dynamics. These Programmable Pattern Generators were adapted to catching a ball with a 30-DOF humanoid robot equipped with a baseball glove, although the end-effector's orientation was not considered. The trajectory estimation of the flying ball used a stereo vision system at 60Hz. The catching point was derived from the intersection of the estimated trajectory with a horizontal plane placed at a given height. Park et al. [27] proposed an evolutionary algorithm for the ball-catching task based on a motion database, created off-line through imitation learning. The database was initially filled with kinematic data extracted from human motions. Then, a data-driven evolutionary optimization was employed to provide human-like reaching with minimal torques in real-time. The proposed framework was validated in a humanoid robot with a 6-DOF arm, equipped with a vision system installed on the head. Kim et al. [28] conducted their ball-catching experiments in the iCub humanoid robot by learning from human demonstrations acquired using a glove and the X-Sens motion capture suit. The work's focus is on the synchronization of the robot's movements with that of a flying ball. The proposed approach controls the timing of robot motions encoded with both Gaussian Mixture Models (GMMs) and Dynamic Movement Primitives (DMPs).

Additional challenges have been addressed in more recent works, such as the inclusion of more complex robots. For example, most of the works discussed above give little importance to the hand control and the grasping strategy. In [29], Bauml et al. addressed the joint control of a DLR-LWR-III arm with 7-DOFs and a 12-DOF DLR-Hand-II hand. The motion control is formulated as a nonlinear optimization problem subject to constraints in terms of workspace, maximum joint velocities, and limits on joint angles. Later, the same authors extended their research to allow for up to two balls to be caught simultaneously using the mobile humanoid robot Rollin Justin [7]. In addition to the degrees-of-freedom of the arms, torso and the mobile platform itself, a 2-DOF pan-tilt unit ensured the ball was in the field of view of the stereo vision system.

Mobile manipulator systems are of interest due to their extended workspace and dexterity in a scenario where accurate high-speed motions are required. Dong et al. [30] proposed a framework based on a hierarchical optimization scheme for real-time trajectory generation. The higher-level kinematic planner is formulated as a nonlinear optimization problem, solved through sequential quadratic programming, while the low-level kinematic planner is solved by quadratic programming. The joint control is driven by an inverse dynamics learning method, which accounts for a state-of-the-art success rate of 85.3%. The balls are thrown by a human subject at around 4 m, resulting in a flight time of about 1 s. The experimental setup includes a 6-DOF manipulator (a UR10 arm from Universal Robotics) rigidly mounted on a 3-DOF omnidirectional mobile platform and the gold standard VICON system for ball estimation and trajectory prediction at 100 Hz.

Another recent topic of research is the possibility of catching arbitrary objects with uneven shapes. In most studies, the object in flight is a ball and the flight trajectory is approximated by a parabola, while the effect of air drag and other forces are ignored. However, uneven flying objects require their dynamical modelling to obtain robust predictions about translational and rotational behaviour. The catching of arbitrary objects with a complex flying behaviour was addressed by Kim et al. [8]. A learning framework is proposed to teach a robot to catch flying objects through an observation of demonstrations encoded by dynamical systems. In this way, the robotic system learns a model of the arm's movements, as well as the dynamics of the flying object, while offline, based on prior information such as mass, shape, or inertia. The prediction system is combined with a probabilistic model to obtain the distribution of optimal catching configurations. The advanced system can coordinate the motion of the arm, hand and fingers to catch a hammer, a tennis racket, a bottle or a cardboard box. Simulations were provided with the iCub humanoid robot, and experiments with the 7-DOF Kuka LWR 4+ arm. Following the same framework, Salehian et al. [9] proposed a strategy in which the robot's hand follows the object's trajectory for a short period of time, to allow for more time to close the fingers. The control law is expressed as a linear system, whose parameters are approximated by Gaussian mixture models (GMMs). In contrast, Yu et al. [31] proposed a neural acceleration estimator to tackle the task of motion prediction for in-flight uneven objects without any prior information. The experimental results show that htis was effective in terms of prediction accuracy and generalization performance for uneven objects in public datasets and real-world experiments with a UR5 robot.

The majority of the previous works concentrate on the classical problem of visual estimation of the ball's trajectory (model-based approach) to anticipate the catching point. However, some others [32–35] continuously used visual information in the feedback control loop (visual serving approaches). This focuses on the relationship between the pose of the object in-flight, the robot's pose and the projection of visual features in the image plane. Sato et al. [35] proposed an eye-in-hand configuration for visual servoing control of high-speed ball catching using a robot arm with 7-DOFs. The system comprises a multi-fingered hand in which 8 small cameras are attached and two external fixed high-speed cameras (500 frames per second). On the one hand, the external cameras deal with predicting the 3D trajectory of the flying ball and the desired catching point. On the other hand, the multi-vision hand provides the visual information needed to correct the hand's position and orientation using a visual serving control.

Among the most recent works are the contributions by Schill and Buss [10] and Ardakani et al. [36]. The former addresses the existing challenge of the reliable task execution of robots catching objects in-flight, such that experimental success in ballistic catching becomes predictable and robust. The authors adapted a hybrid systems framework approach, focusing on the Zeno behaviour of bouncing balls to enable the robotic catching of spherical objects. A dynamical system parametrization enables dynamically feasible offline motions using hybrid bouncing ball formalisms. The success prediction and dynamic feasibility are solved through optimization-based motion planning. The authors provide solutions to the different phases of joint robot–robot throwing and catching. The experimental setup consists of two 2-DOF robots, symmetrically mounted on a vertical plane. They consider a joint robot–robot scenario, which does not require visual feedback, in which each robot can perform the throwing and the catching task. The robustness, with respect to uncertainties in the impact model and object's state accuracy, is quantified. The latter proposes the use of model predictive control (MPC) to solve the problem of point-to-point trajectory generation in a ball-catching scenario, while respecting physical limitations and real-time requirements.

The most evident finding in the literature of robot ball-catching is the lack of works emphasizing the advantages of earlier anticipations based on observations of the thrower's movement. The existence of great flexibility in the use of visual information by humans was reported in [37]. For this purpose, the participant's access to earlier information regarding both the thrower's action and the ball's trajectory was manipulated. By recording the whole-body postural control, the performance of each participant was evaluated, considering three conditions: only the thrower's action is available, only the information about the flying ball is available, and all visual information is available. Their study revealed that movements were initiated earlier when visual information was available, prior to ball flight, resulting in improved performance. Along the same line, the recognition of human actions and intentions is recognised as essential for human–robot interaction [38]. This concept of earlier anticipation was explored in the context of human–robot table tennis [18]. In the context of human–robot ball-catching, this idea was first addressed by the authors in [21] and will be revisited in this article, which provides an extended analysis of the novel approach in the form of a simulation-based testbed.

3. Experimental Setup

3.1. Simulation Scenario

This section describes the main blocks involved in the simulation of the human–robot ball-catching task, as well as the various assumptions imposed in this study (see Figure 1). First, a scenario is assumed in which a human performs underarm ball throwing, and the robot (at about 4 m) tries to catch the balls entering its reachable space. Second, the projectile motion is approximated by a curved parabolic path, assuming that the only force acting on the flying object is gravity (i.e., air resistance is neglected). Regarding, the robot catcher, this study considers a 3-DOF articulated manipulator (RRR), mounted upside down so that, in the home position, it is fully extended down. The world coordinate frame is fixed to the ground, with the x-axis oriented towards the human thrower, while the z-axis of both coordinate frames, S_w and S_0, are vertically aligned with each other. The robot's base, as well as the origin of the base frame S_0, are located at a height of 1.70 m above the ground.

The robot's link lengths, from the base to the end-effector, are the following: $l_1 = 0.1$ m (base-shoulder), $l_2 = 0.3$ m (shoulder-elbow) and $l_3 = 0.3$ m (elbow-wrist). The robot's reachable workspace is further reduced due to the physical limits imposed on the joints according to the following inequalities: $|q_1| < 150°$ (vertical joint), $|q_2| < 105°$ (shoulder joint) and $0 < q_3 < 135°$ (elbow joint). The flying ball and the robot's end-effector are modelled as point source objects, while the problems of hand orientation and grasping are not addressed. In this work, the desired catching point is determined from all valid solutions inside the reachable workspace, as the trajectory point of the flying ball that is closest to the end-effector. This target point is constantly updated as new observations become available. At the beginning of each trial, the arm adopts an initial configuration where the elbow angle is at a right angle, aligned with the x-direction.

Focusing on control, the progressive refinement of the desired catching point requires the online re-planning of the robot's kinematic chain so that the end-effector reaches the right place in time. This work adopts the so-called kinematics control in the motion control problem. This is based on an online inverse kinematic transformation that computes the reference joint velocities corresponding to an assigned end-effector velocity direction, derived as the difference between the catching point and the end-effector's position. The solution to the inverse kinematics problem is based on the inversion of the manipulator's analytical Jacobian matrix and the use of feedback corrections as a result of re-estimating the ball's trajectory. The online closed-loop algorithm is implemented in discrete-time form, with a time interval of 10 ms. The objective is to find, at instant t_k, a joint velocity vector $\dot{q}_k$ that allows the end-effector to move as quickly as possible towards the desired catching point under the velocity constraints on the actuators. For that purpose, a scaling factor is applied to the joint velocity vector, such that the maximum possible values are always used, while the end-effector remains in the desired direction.

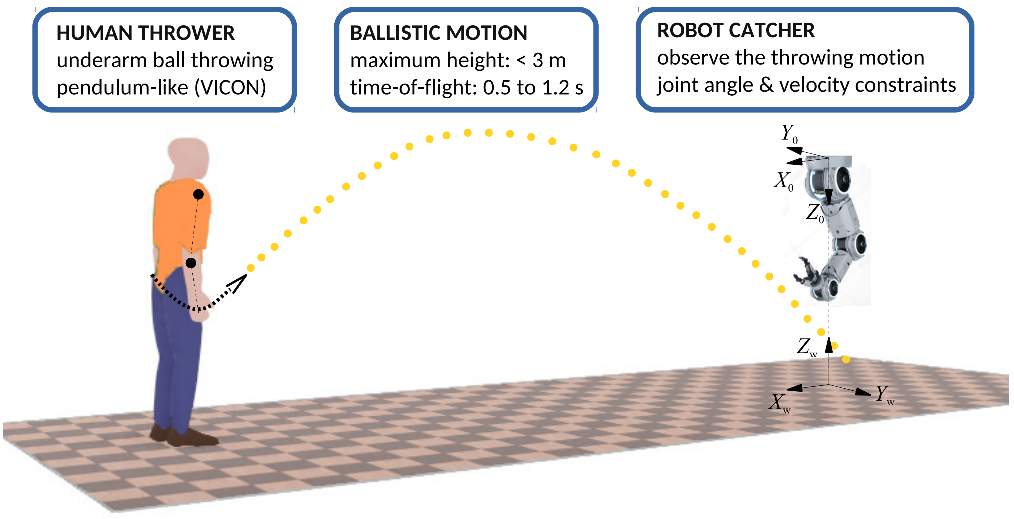


Figure 1. Overview of the simulated human–robot ball catching task decomposed into three main blocks, namely, human thrower, ballistic motion, and robot catcher. The main assumptions underlying the study are also highlighted.

3.2. Human Demonstrations of Underarm Throwing

The generation of throwing trajectories was supported by an analysis of human demonstrations acquired from two subjects playing catch at a distance of about 4 m from each other. Motion capture was performed in a human motion analysis laboratory, equipped with a VICON optoelectronic system with eight infrared cameras. A standard upper-body marker set was attached to both subjects and two additional markers were attached to the ball. The 3D coordinates of each marker were collected at 100 Hz and stored for offline analyses with customized software written in Matlab (Mathworks, MA, USA).

Figure 2 illustrates the projection of the 3D thrower's hand data in a 2D vertical plane aligned with the projectile motion with the axes x horizontal and y vertically up. The start marker (circles 'o') indicates the start of the retraction phase, in which the hand is pushed back while the arm stretches. The release marker (asterisks '$\star$') was obtained by observing the distance between the coordinates of the ball and the hand. The analysis indicates the existence of regularity in the shape of the generated trajectories, as well as a small variability in the duration of the movement's execution time. Therefore, a single demonstration that ignores the retraction phase was extracted from the real data as the basis of implementation of the throwing generator.

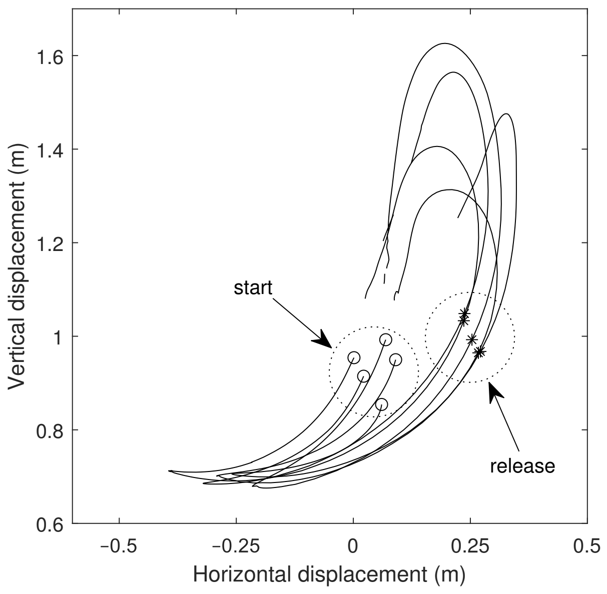


Figure 2. Spatial coordinates of the thrower's hand for multiple launches represented in the same reference frame. The circles 'o' are the start of the movement and the asterisks '$\star$' are the moment at which the ball is released.

3.3. Data Generation in Preparatory and Ballistic Phases

In this study, catching the ball involves observing the thrower's action, as well as observing the free motion as it approaches the robot. This subsection describes the data-generation process required to both train the feedforward neural network (core of RALS) and simulate the complete throwing–catching task. The simulation includes the ground truth data of the ball's behaviour and raw data representing the effects of noisy measurements. The complete motion of the ball is divided into a preparatory phase, followed by a ballistic phase. The preparatory phase lasts from the beginning of the throw until the moment the ball is released. This is followed by the ballistic phase, in which the ball's motion is completely determined by the laws of physics, whatever the trajectory during the initial phase. From the mechanical perspective, the path of the ball during flight takes place in a vertical plane, parallel to the gravity vector. The generation of the ball trajectory takes place over the entire simulation period, according to a given sequence.

The first step is to express the ballistic motion as a function of the following input parameters: (i) the initial position vector of the ball P_i represented by the coordinates (x_i, y_i, z_i) in the world reference frame; (ii) the final position vector of the ball P_f when it intercepts the ground plane is represented by the coordinates (x_f, y_f, z_f) in the world reference frame; (iii) the time t_f elapsed from the initial position P_i until the ball reaches the target final position P_f. These parameters assume random values with uniform distribution, within the following ranges: $x_i \in [3.5; 3.75]$, $y_i \in [-0.1; 0.1]$ and $z_i \in [0.8; 1.4]$; $x_f \in [-3.0; 0.0]$, $y_f \in [-1.0; 1.0]$ and $z_f = 0$; and $t_f \in [0.5; 1.5]$. Restrictions on the highest point of the object above the ground and on the possible range of the time-of-flight (i.e., from the launch to the catch) are also included. In this study, the maximum height is limited to 3 m and the time-of-flight (ToF) ranges from 0.5 to 1.2 s. A common problem in the above parametrization algorithm is that some of the generated trajectories may be invalid. Therefore, feasibility verification is considered, such that any trajectory outside the reachable workspace, or any trajectory that is unable to satisfy the imposed constraints, is discarded. The second step requires a throwing trajectory that verifies the previously established conditions in terms of the horizontal and vertical components of the ball's position and velocity at release. For that purpose, a polynomial interpolation was used on the selected demonstration data. The algorithm provides trajectory smoothing along with the necessary requirements. Gaussian noise is added to the generated trajectories to represent the measurement noise that corrupts the observed samples.

3.4. Release Parameters versus Catching Point

The projectile motion is revisited by including specific aspects of a ball-catching task, such as the robot's reachable space and the target-catching point. More concretely, this subsection establishes analytical relationships between the physical parameters of the projectile motion and the desired catching point, assuming negligible air resistance. For reasons of simplicity, the analysis is performed by assuming a two-dimensional projectile motion modelled by parabolic equations. In this context, the independent horizontal and vertical displacements, as function of time t, can be written as:

$$\begin{aligned} x &= x_0 + v_{0x} t \\ y &= y_0 + v_{y0} t - \frac{1}{2} g t^2 \end{aligned} \tag{1}$$

where (x_0, y_0) and (v_{0x}, v_{0y}) are, respectively, the initial position and initial velocity of the object, and g is the gravitational constant ($g = 9.81\ \mathrm{m/s^2}$). The maximum height that the projectile reaches above the ground is obtained from the velocity equation in the vertical direction for $v_y = 0$, that is:

$$0 = |\vec{v}_0| sin\theta - g t_h = v_{0y} - g t_h \tag{2}$$

where $\vec{v}_0$ is the initial velocity vector and t_h is the time taken to reach the maximum height, given by $t_h = \frac{v_{0y}}{g}$. From the vertical displacement in (1), we obtain,

$$h^{max} = y_0 + \frac{v_{0y}^2}{g} - \frac{1}{2} g \frac{v_{0y}^2}{g^2} = y_0 + \frac{v_{0y}^2}{2g} \tag{3}$$

Assuming a restriction on the highest point of the projectile h^{max}, Equation (3) provides a superior limit to the initial velocity in the y-direction, as follows:

$$v_{0y}^{max} = \sqrt{2g(h^{max} - y_0)} \tag{4}$$

Equation (1) can also be used to determine the set of trajectories that intercept a given target point (x_t, y_t). By eliminating t, we obtain the following equality:

$$(y_t - y_0) v_{0x}^2 - (x_t - x_0) v_{0y} v_{0x} + \frac{1}{2} g (x_t - x_0)^2 = 0 \tag{5}$$

By specifying the initial velocity in the y-direction, we can obtain the required velocities in the x-direction by solving a quadratic equation. It should be noted that, although there may be two solutions, only one is valid, since we consider that the interception with the target point occurs during the descending phase of the ballistic motion. At the same time, a discriminant $(b^2 - 4ac)$ of less than zero indicates that there is no solution, providing a lower limit on the initial velocity in the y-direction, as follows:

$$v_{0y}^{min} = \sqrt{2g(y_t - y_0)} \tag{6}$$

Figure 3 illustrates an example of multiple ballistic trajectories that intercept the same target point. The black curve is the single one where the target point coincides with the catching point, i.e., the closest point to the end-effector. At the same time, the evaluation of the ToF for each ballistic trajectory follows a similar procedure by solving a quadratic equation. The equation used to evaluate the ToF with respect to the initial velocity in the y-direction, producing the same velocity limit as expressed in (6), is the following:

$$\frac{1}{2}gt^2 - v_{0y}t - (y_t - y_0) = 0 \tag{7}$$

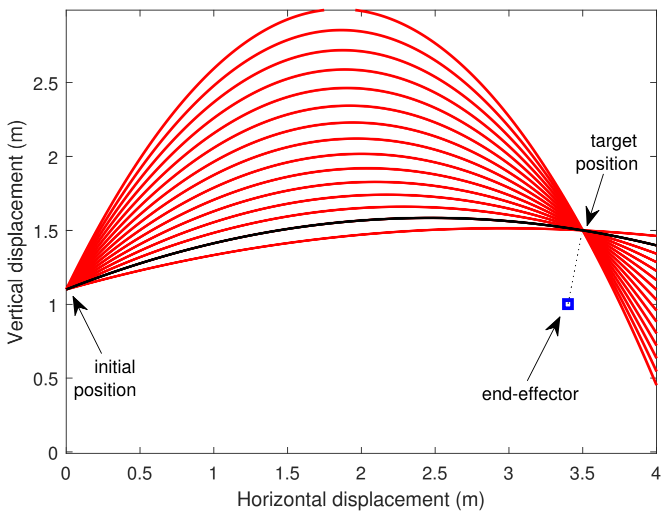


Figure 3. Example of multiple ballistic trajectories of a projectile launched at the initial position $(x_0, y_0) = (0, 1.1)$ meters with different initial velocities (v_{0x}, v_{0y}), while intercepting the same target point $(x_t, y_t) = (3.5, 1.5)$ meters inside the robot's workspace. The robotic arm is shown in blue along with the workspace in dashed lines.

At the end of this study, we provide an intuitive insight to the relationship between the initial velocities of the ball at the moment it is launched ($t = 0$) and the feasibility of the generated movement (see Figure 4). An invalid movement results from various circumstances, namely, it is outside the robot's workspace, and exceeds the limits imposed on the maximum height of the ball h^{max} or on the ToF. Movements that intercept the reachable workspace and comply with these limits are considered valid.

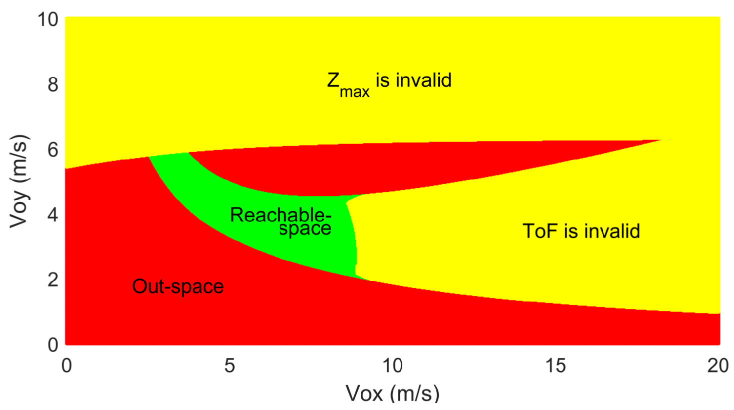


Figure 4. From a total of 10,000 launches, generated with initial velocities that are uniformly distributed in the range $0 < v_{ox} < 20$ m/s and $0 < v_{oy} < 10$ m/s), 33.3% are out of the reachable space (red), 5.4% are inside the reachable space (green), and 38.9% are inside the reachable space. However, these are invalid since the maximum height (h^{max}) is greater than 3 m. A total of 22.4% are inside the reachable space, but are invalid, since the time-of-flight (ToF^{min}) is less than 0.4 s.

4. The Anticipation Learning System

4.1. Anticipation Model

As mentioned previously, the robot ball-catching task is divided into two sequential motion phases, reflecting the ball trajectory before and after it has been released, as follows:

- Phase A (throwing phase)—refers to the sequence of movements from the hand of the thrower, before the ball is released:
- Phase B (ballistic phase)—refers to the ball's motion after release.

In other works, the prediction of the catching point and the robot's motion-planning and execution are performed in phase B. During this phase, the trajectory of the free-flying ball is modeled as a parabola, and the interception point is predicted from the estimated curve, which triggers a relevant robot movement. However, to obtain a reliable estimation of the ball trajectory, the robot needs to wait until sufficient visual information is acquired. This may result in insufficient time to move to the catching point, particularly in short-distance scenarios and in cases with limitations on joint velocity. We pose the following general anticipation question: Can we make use of the vision information during phase A to improve the robot's planning and ultimately increase the ball-catching success rate?

To test this hypothesis, we define the ball state in phase A as the following sequence: $S_t^{(A)} = \{S_{pos,t}^{(A)}, S_{vel,t}^{(A)}\}_{t=0,\dots,T}$, where $S_{pos,t}^{(A)} \in \mathbb{R}^3$ and $S_{vel,t}^{(A)} \in \mathbb{R}^3$ denote the ball spatial positions and velocities at the sampling moment t. T is the time of ball release, i.e., the transition moment from phase A to phase B. Since, in state A, the ball is in the hand of the throwing opponent, a natural assumption here is that the ball state and the thrower's hand motion are the same variables. Now, the general anticipation question is reformulated into the following specific task: Predict the ball state (position and velocity) at the moment it is released (i.e., $S_T^{(A)}$), from sequential observations of the thrower's hand motion (a multidimensional sequence of the first L samples, $L < T$ of the ball state $S^{(A)}$),

$$S_T^{(A)} = f(S_{pos,0}^{(A)}, S_{pos,1}^{(A)}, \dots S_{pos,L-1}^{(A)}, S_{vel,0}^{(A)}, S_{vel,1}^{(A)}, S_{vel,L-1}^{(A)}) \quad (8)$$

If the initial conditions of the flying ball are reliably predicted, its trajectory can be estimated as a parabolic motion and the robot arm can move in the estimated direction earlier (in phase A). Given the nonlinear nature of the mapping function and the inherent variability of the task execution, a Neural Network (NN) seems to be the natural choice to fit the regression function (Equation (8)). NNs are known to have good generalization properties and a well-established training framework; therefore, we did not explore other regressors. However, alternative options may be also considered. Two prediction scenarios were studied, as follows:

- Joint Position-Velocity prediction—Prediction of the ball's position and velocity at the start of the ballistic phase ($\bar{S}_T^{(A)}$) with one NN model. This corresponds to a regression between a ($6L$)-dimensional input (L samples of spatial position and velocity) and a six-dimensional output (spatial position and velocity at the transition time T),

$$S_{t=0,\ldots L-1}^{(A)} = \{S_{pos,0,\ldots L-1}^{(A)}, S_{vel,0,\ldots L-1}^{(A)}\} \in \mathbb{R}^{6L} \tag{9}$$

$$\bar{S}_T^{(A)} = \{S_{pos,T}^{(A)}, S_{vel,T}^{(A)}\} \in \mathbb{R}^6 \tag{10}$$

- Separated Position and Velocity prediction—Separate prediction of the ball initial position ($\bar{S}_{pos,T}^{(A)} \in \mathbb{R}^3$) and initial velocity ($\bar{S}_{vel,T}^{(A)} \in \mathbb{R}^3$) at the moment of release,

$$\bar{S}_{pos,T}^{(A)} = f_{pos}(S_{pos,t}^{(A)}, S_{vel,t}^{(A)}, t = 0, \ldots L-1) \tag{11}$$

$$\bar{S}_{vel,T}^{(A)} = f_{vel}(S_{pos,t}^{(A)}, S_{vel,t}^{(A)}, t = 0, \ldots L-1) \tag{12}$$

The underlying idea of the proposed anticipation model is to gain extra time for the robot arm to start moving in the targeted direction before the opponent finishes throwing. In phase B, the robot continues to adjust its catching position based on dynamically collected information of the flying ball. However, if the anticipation phase is successful, the robot is better prepared for the incoming ball. Its arm is expected to be closer to a feasible catching point, therefore improving the interception success rate.

4.2. *Anticipation Model Training*

A standard (shallow) NN architecture, with one hidden layer (HL) and the logistic type of HL neurons, was trained. The output layer has six linear units that reflect the specific regression task of the network. A dataset of 3000 ball catching episodes (trials) was generated based on a human demonstration (see Section 3.3), and divided into 1500 trials for training, 1000 for cross-validation and 500 for testing. The maximum number of training iterations was set to 5000; however, the over-fitting risk was handled by stopping the training early, after 250 iterations of increasing validation error. The final structure of the models was optimized after tuning the most sensitive hyper-parameters, the input memory L and the number of hidden layer units. L determines how many samples of visual information on the thrower's hand motion are required before a reliable prediction of the ball's release state can be provided. During the experiments, 36 samples ($\mathbb{R}^6$ vectors) from phase A were collected to represent the ball state sequence $S^{(A)}$. The last sample is the ball release state.

4.2.1. Model 1 (Joint Position-Velocity Prediction)

The Mean Squared Error (MSE) between the target release ball state ($\bar{S}_T^{(A)}$), known from m demonstration trajectories, and the state ($S_T^{(A)}$) predicted by the model was the loss function of Model 1, to be minimized with the scaled conjugate gradient method:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\bar{S}_T^{(A)} - S_T^{(A)})^2 \tag{13}$$

As shown in Figure 5, the MSE decreases when the memory L increases, i.e., the more visual information is collected, more accurate the predictions. However, predictions need to be made early enough to effectively provide the robot with extra time to start moving in the estimated direction. $L = 20$ was chosen as a trade-off between prediction accuracy and the extra time needed for task execution. The NN input is then $S_{1,..,20}^{(A)} = \{S_{pos,1,\ldots,20}^{(A)}, S_{vel,1,\ldots,20}^{(A)}\} \in \mathbb{R}^{120}$.

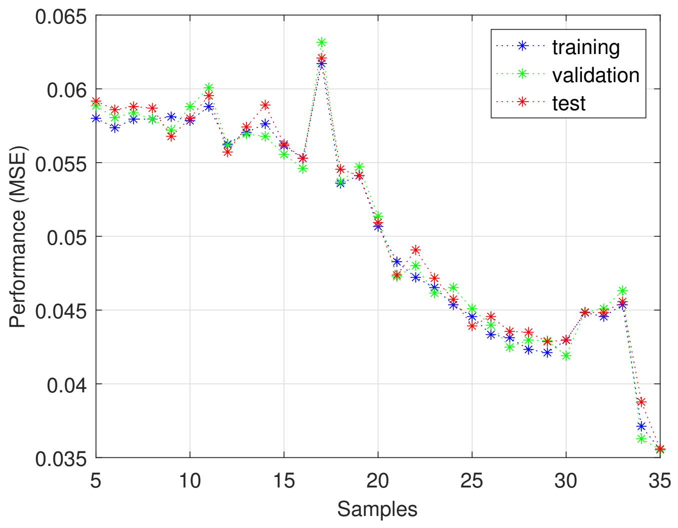


Figure 5. Model 1 hyper-parameters' optimisation: MSE vs. Neural Network (NN) input memory (samples *L*).

The training, validation and test curves shown in Figure 5 are close, which indicates a lack of over-fitting. The number of HL units was chosen after a grid search in the range [2, 20] and the results are shown in Figure 6. Five HL neurons were taken as a trade-off between the error rate and the model complexity. To obtain a better insight into the effect of the sensor noise on the predicted variables, we designed the following experiment. Starting from an ideal (reference) throwing trajectory, we generated a number of noisy trajectories by adding uniform random noise (40 dB SNR). We then applied the trained NN model to predict the ball position and velocity at the release state from these noisy observations. The predicted values are shown in Figures 7 and 8. We could observe is that the velocity is significantly more affected by the noise (particularly in X and Z directions) than the position. To address this issue, we decided to split the prediction of the position and velocity into two distinct models.

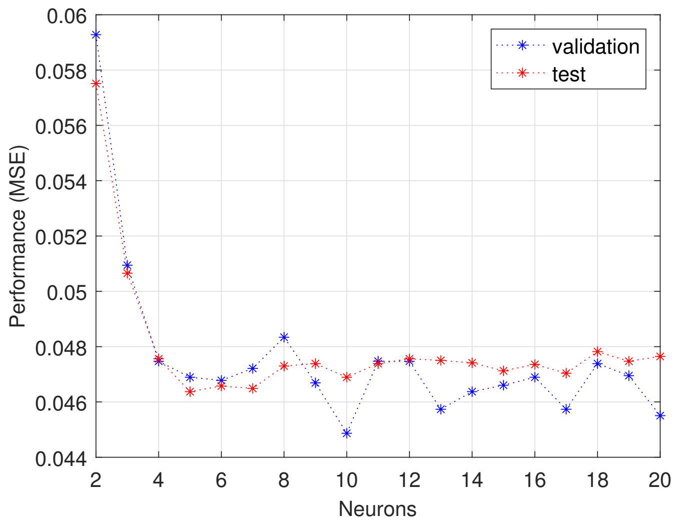


Figure 6. Model 1 hyper-parameters' optimisation: MSE vs. Hidden Layer (HL) neurons.

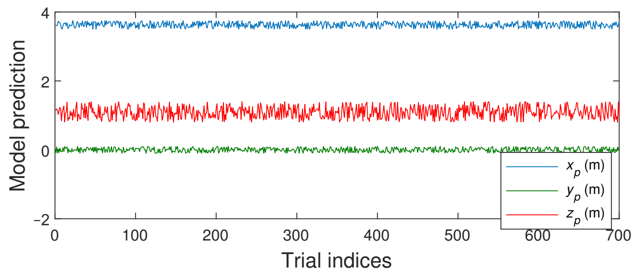


Figure 7. Prediction of the ball's position (x_p, y_p, z_p) at the release state after corrupting the observation samples with noise (40 dB SNR).

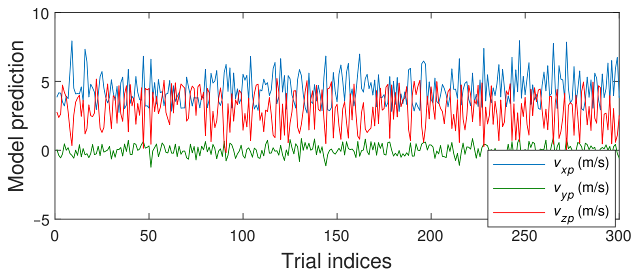


Figure 8. Prediction of the ball's velocity (v_{xp}, v_{yp}, v_{zp}) at the release state after corrupting the observation samples with noise (40 dB SNR).

4.2.2. Model 2 (Separated Position and Velocity Prediction)

In this scenario, the prediction models were trained separately, with the following MSE loss functions to be minimised:

$$MSE_v = \frac{1}{m}\sum_{i=1}^{m}[(\bar{S}_{v,T}^{(A)} - S_{v,T}^{(A)})^2]_v, v = \{pos, vel\} \tag{14}$$

Here, we choose to keep the same number of HL neurons as used in Model 1 (i.e., five neurons) and evaluate only the effect of the network memory. From the results in Figures 9 and 10, it is interesting to observe that the MSEs of both position and velocity are significantly lower than the overall prediction error in Model 1 (see Figure 5) for the entire range of HL neurons. At the same time, the MSE position gradually decreases, while more samples are accumulated, which is not the case with the velocity MSE. Therefore, $L = 20$ was chosen as a good trade-off between the waiting time before making predictions and the prediction accuracy. These results clearly suggest that using separate models to estimate the position and velocity leads to an increased capacity to filter noise coming from different distributions. In the next sections, Model 2 is implemented as the learning component of RALS to study the role of early anticipation. Algorithm 1 summarizes the steps involved in its implementation for human–robot ball catching.

Algorithm 1: RALS-based Ball Catching Process.

```
PROCEDURE: Anticipation Model Training (offline)
    $\bar{S}^{(A)}_{pos,T}$, $\bar{S}^{(A)}_{vel,T}$ ⇐ train separate NN models to predict the ball position and velocity
     at release state after observing the first $L$-samples of the thrower's motion.
END PROCEDURE
PROCEDURE: Simulation (online)
for $i = 1 : 1 : endSimulation$ do
PHASE A (throwing phase)
    if (ball in hand & L-samples) then
        $\bar{S}^{(A)}_{pos,T}$, $\bar{S}^{(A)}_{vel,T}$ ⇐ NN models generate the initial conditions of the flying ball;
        Compute the ball's trajectory with a parabolic motion; Compute potential
        catching point; move the robot arm in the target direction.
    end if
PHASE B (ballistic phase)
    if (ball flying & S-samples) then
        Switching time for recursive estimation of the ball's trajectory through least
        square optimization; Move the robot arm in direction to the estimated
        catching point.
    end if
    if (ball is catchable) then
        $r^d$ ⇐ determine the catching point as the closest one to the current end-effector
        position; $\dot{q}$ ⇐ move the arm using a Jacobian-based IK algorithm and read
        current joint angles;
        if Distance(Ball,EndEffector) < 2 cm then
            Ball caught—exit the simulation loop;
        end if
    else
        Move the end-effector towards the last catchable point or stay in the
        current state;
    end if
END FOR
END PROCEDURE
```

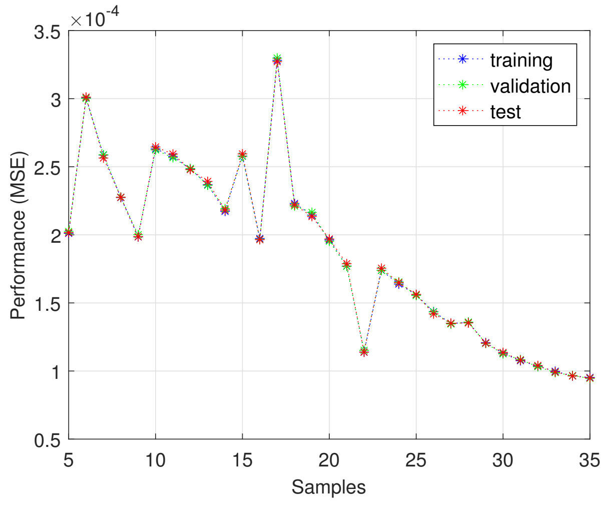


Figure 9. Model 2 hyper-parameters' optimization (position prediction model): MSE vs. Neural Network (NN) input memory.

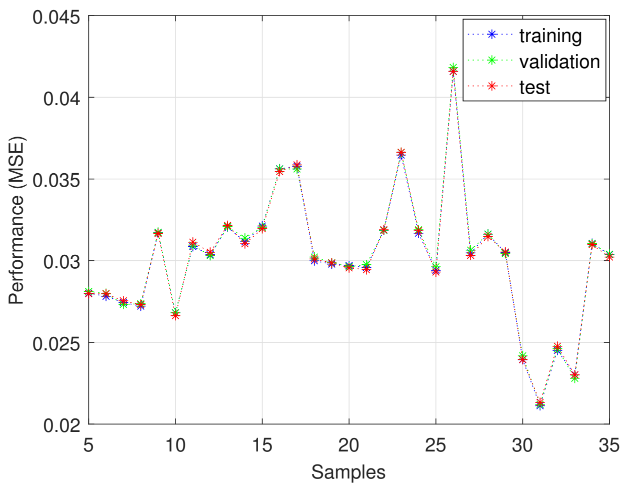


Figure 10. Model 2 hyper-parameters' optimization (velocity prediction model): MSE vs. Neural Network (NN) input memory.

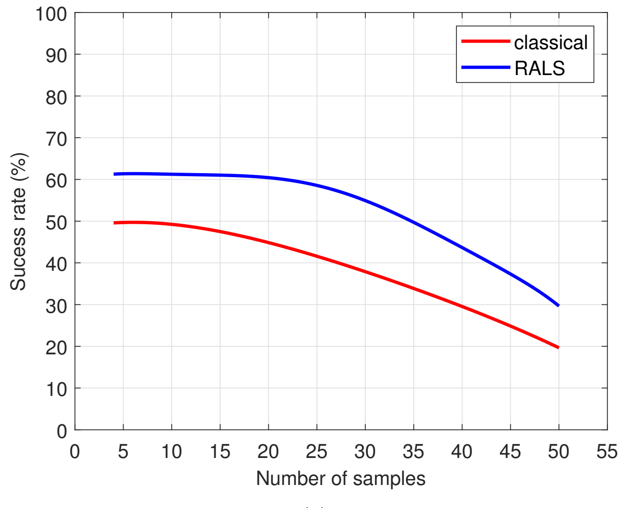
5. Results

This section aims to evaluate the proposed RALS framework in different test scenarios, such as variations in the switching time between the perception phase and the robot's control, in the signal-to-noise ratio (SNR) applied to the ball trajectory (i.e., sensory measurements) and the maximum velocities available at the robot's joints. For the purpose of evaluation, a set of 1000 valid launches is selected from the generated trajectories, i.e., those in which the ball intercepts the robot's reachable workspace and those that satisfy the imposed constraints on the maximum height the ball can achieve, in a range between 0.4 and 1.2 s for the time-of-flight. A successful catch occurs whenever the distance between the ball's centroid and the end-effector position is less than 2 cm. The control actions are stopped as soon as the end-effector is sufficiently close to the target, i.e., below the threshold limit of 2 mm. A failed attempt can be explained by an inaccurate estimation of the ball's state (bad prediction failure) or the robot's inability to reach the desired catching point in time (no-time failure). The experiments were carried out by considering that the arm is aligned with the x-axis of the reference coordinate system, adopting an optimal initial configuration in terms of the manipulability index [39].

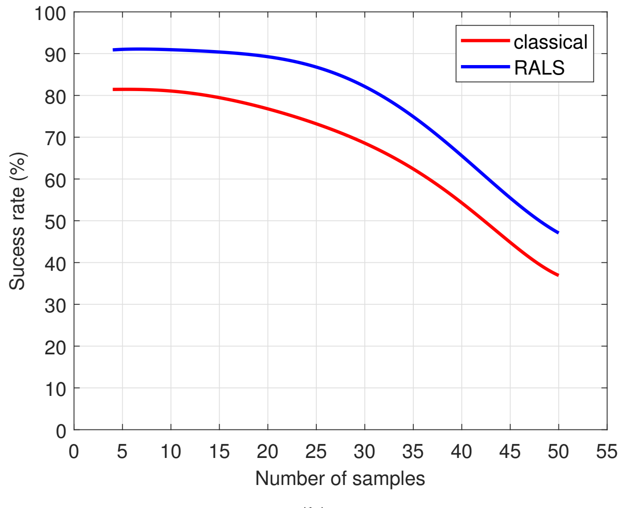
5.1. Switching Time between Perception and Action

In the throwing phase, the robotic arm immediately starts moving towards the predicted catching point, guided by RALS. Once the ball is released (ballistic phase), the robot has to decide when to switch from RALS prediction to the classical (baseline) prediction of the ball trajectory through least square optimization. Regarding the classical method for ball-catching without anticipation skills, the switching time refers to the waiting time before the robot arm starts moving in the targeted direction. In both cases, the switching time is expressed as the number of observations (samples S) of the ball in-flight.

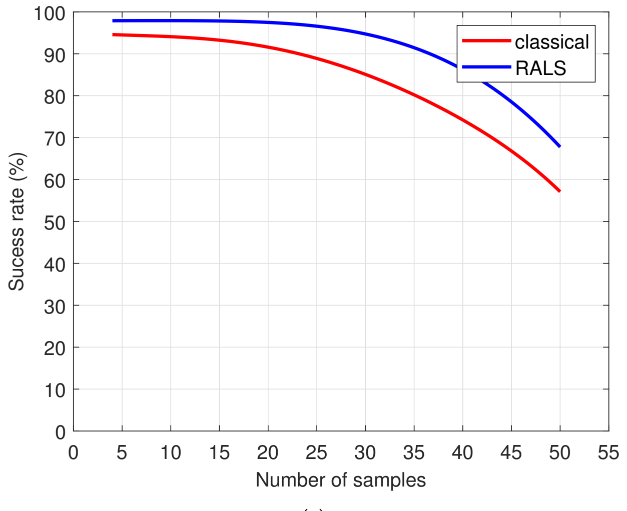
Figure 11 shows the catching success rate as a function of the switching time for ball-catching with a neural network (RALS) and without a neural network (classical). For these experiments, the sensory signals were corrupted, with 40dB SNR noise, and the maximum robot joint angular velocities are: (a) 90 deg/s, (b) 135 deg/s and (c) 180 deg/s. The first observation is that the RALS-based strategy (blue line) outperforms the baseline in all scenarios. Naturally, the success rate improvements are more significant for lower-speed robots. Second, the highest success rate occurs when the switching time is in the first 10–20 samples of the flying ball. Presumably, the later the robot arm starts moving, less likely it is to catch the ball. However, the RALS-based strategy shows that the success rate is almost constant for a considerably larger number of samples. This means that there is no advantage in re-planning the robot's movement too early, since the success rate is at an almost constant level until the 20th-sample. In the same line of thought, Figure 12 depicts the catching failure due to the lack of time taken to reach the ball, denoted as the no-time failure rate, for the same maximum joint velocities.



(**a**)



(**b**)



(**c**)

Figure 11. Success rate as a function of the switching time (in samples) with and without RALS. The maximum joint velocities are: (**a**) 90 deg/s, (**b**) 135 deg/s and (**c**) 180 deg/s.

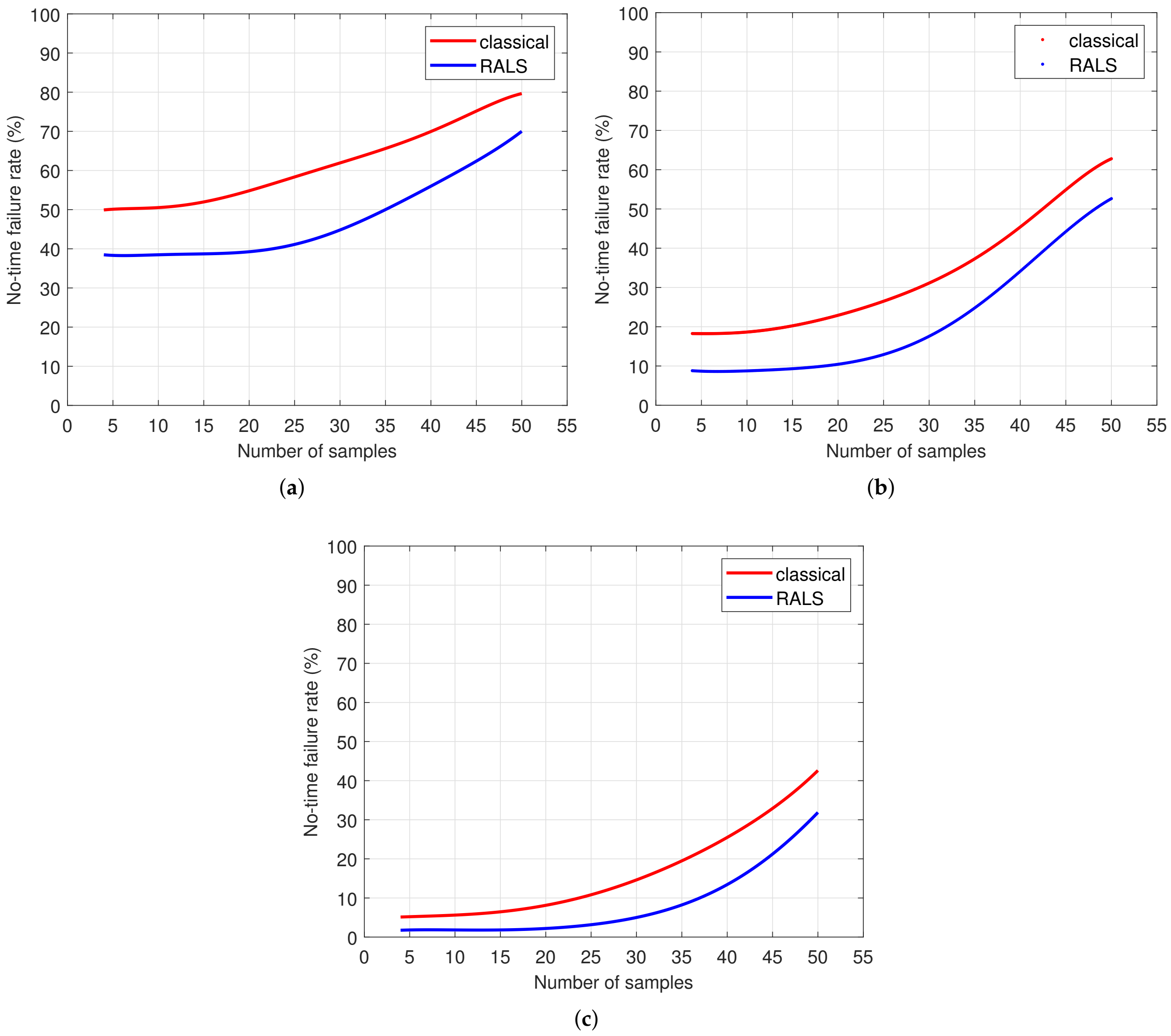


Figure 12. No-time failure rate as a function of the switching time (in samples) with and without RALS. The maximum joint velocities are: (**a**) 90 deg/s, (**b**) 135 deg/s and (**c**) 180 deg/s.

5.2. Signal-to-Noise Variation

Figure 13 compares the success rate as a function of the SNR level on the sensory data for a switching time of S = 20 samples and two different maximum joint velocities: (a) 90 deg/s and (b) 135 deg/s. The same level of noise was applied to both the throwing and ballistic phases. Although low SNR and low robot joint velocities affect clearly both methods, RALS outperforms the baseline method with an average success rate leverage of 20%. Neural networks are known for their good data representation capacity and filtering properties. Therefore, it is not surprising that, in the range of 36–50 dB, for the most typical case of 135 deg/s, the success rate remains above 90%. The baseline method is more sensitive to noise, achieving a success rate close to 80% only for high SNR levels. Figure 14 complements the previous results, showing the evolution of the bad prediction failure rate with the SNR under the same conditions.

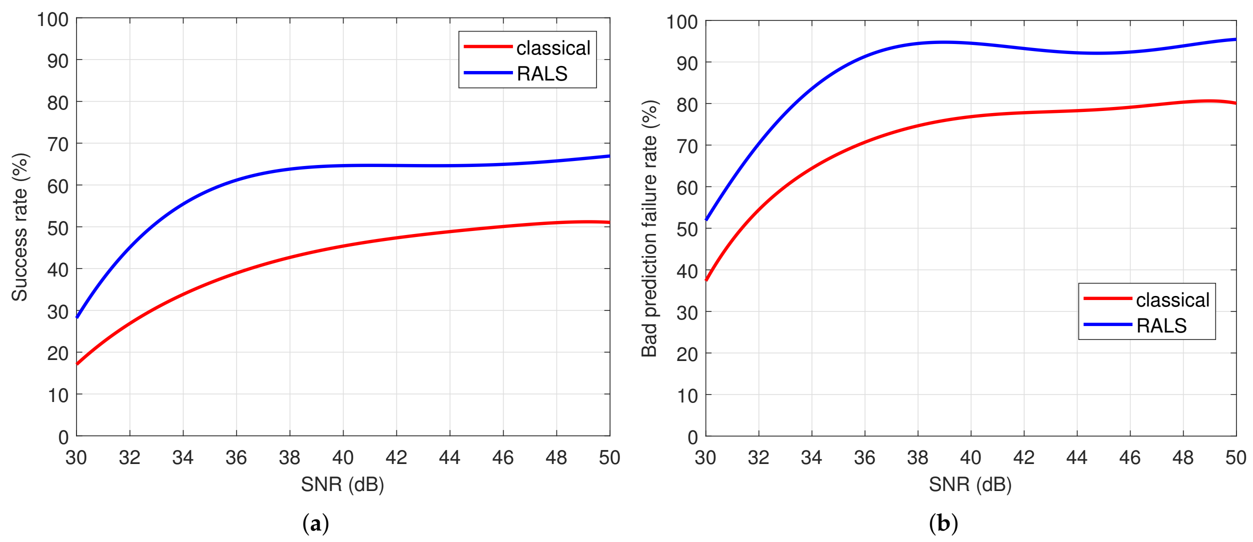


Figure 13. Success rate as a function of the SNR in the observed samples for maximum joint velocities of: (**a**) 90 deg/s and (**b**) 135 deg/s.

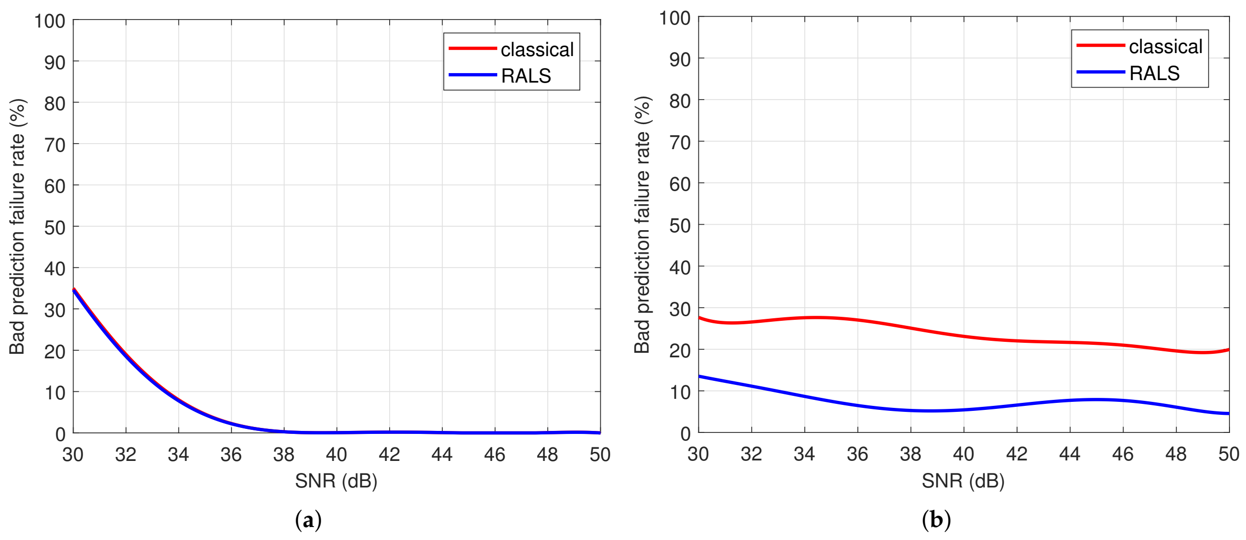


Figure 14. Bad prediction failure rate as a function of the SNR in the observed samples for maximum joint velocities of: (**a**) 90 deg/s and (**b**) 135 deg/s.

5.3. Maximum Joint Velocity Variation

As seen previously, the maximum angular velocities of the robot's joints have a major influence on the catching performance. Figure 15a shows the success rate as a function of the maximum velocity, assuming the same bound for all joints. The simulation was set up with a switching time of 20 samples and an SNR of 40 dB. The advantage of using RALS-based early predictions is clearly noted, with success rate improvements of about 10–12% over almost the entire range. The bar chart in Figure 15b allows for a more detailed comparison between the two methods under analysis. Each experience is classified as: (i) "Only-RALS" if the anticipation method caught the ball, but the classical method could not, (ii) "Only-classical" if the RALS method could not catch the ball, but the classical method succeeded, and (iii) "Successful in both" if both methods catch the ball. The RALS improves the success rate by about 12% over almost the entire range.

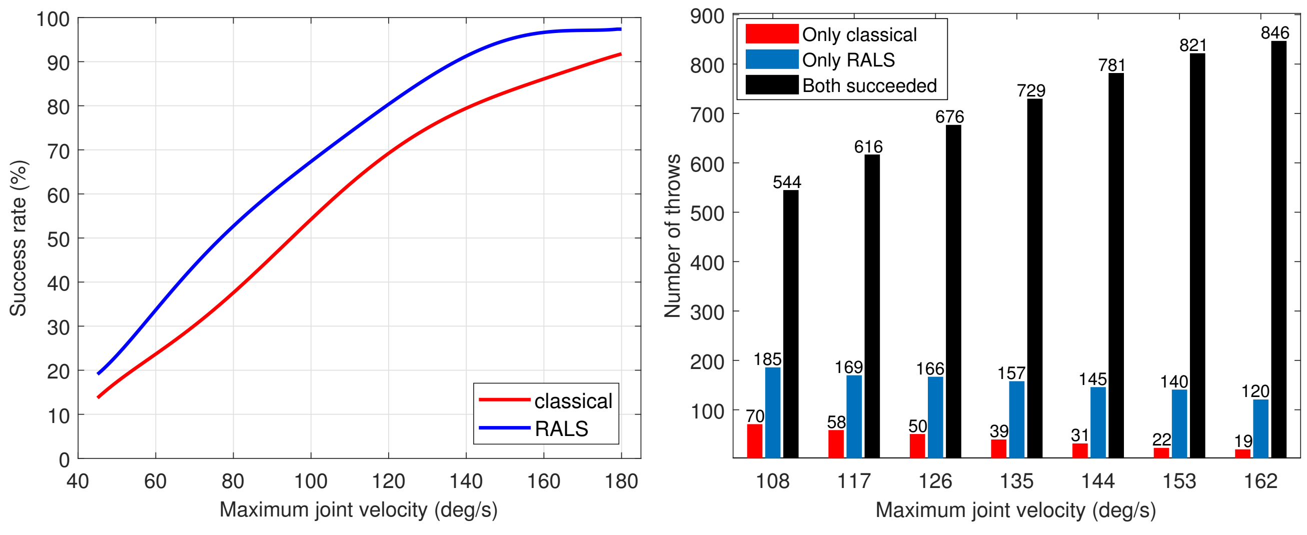


Figure 15. (**a**) Success rate as a function of the maximum joint velocities with and without RALS. (**b**) Number of successes unique to each method and those where both succeeded in catching the ball.

6. Conclusions

In this paper, we presented Robot Anticipation Learning System (RALS) that predicts the ball state (position and velocity) at the moment it is released, from observations of the thrower's motion before the ball is released (the so-called anticipation, preparatory phase). Based on the prediction, the ball-catching point is estimated much earlier and the robot starts to approach this point. RALS improved the robot ball-catching rate by up to 20% compared to the baseline approach, where the predictions rely only on information received during the free-flying ball motion. To the best of our knowledge, this is the first autonomous robot control system for ball-catching that enhances the motion-planning policy with information from the stage when the ball is still in the hand of the throwing opponent.

Author Contributions: Conceptualization, D.C., F.S. and P.G.; investigation, D.C.; writing—original draft preparation, F.S. and P.G.; writing—review and editing, F.S. and P.G.; supervision, F.S. and P.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: This work was supported by the Foundation for Science and Technology (FCT) in the context of the project UIDB/00127/2020.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nemec, B.; Ude, A. Reinforcement learning of ball-in-a-cup playing robot. In Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics, ROBIO 2011, Karon Beach, Thailand, 7–11 December 2011. [CrossRef]
2. Kober, J.; Peters, J. Policy search for motor primitives in robotics. *Mach. Learn.* **2011**, *84*, 171–203. [CrossRef]
3. Bujarbaruah, M.; Zheng, T.; Shetty, A.; Sehr, M.; Borrelli, F. Learning to Play Cup-and-Ball with Noisy Camera Observations. In Proceedings of the 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), Hong Kong, China, 20–21 August 2020. [CrossRef]
4. Senoo, T.; Namiki, A.; Ishikawa, M. Hybrid Trajectory Generation of an Articulated Manipulator for High-speed Batting. *J. Robot. Soc. Jpn.* **2006**, *24*, 515–522. [CrossRef]
5. Muelling, K.; Kroemer, O.; Lampert, C.H.; Schölkopf, B. Movement Templates for Learning of Hitting and Batting. In *Learning Motor Skills*; Springer Tracts in Advanced Robotics; Springer: Cham, Switzerland, 2014. [CrossRef]

6. Jia, Y.B.; Gardner, M.; Mu, X. Batting an in-flight object to the target. *Int. J. Robot. Res.* **2019**, *38*, 451–485. [CrossRef]
7. Bäuml, B.; Birbach, O.; Wimböck, T.; Frese, U.; Dietrich, A.; Hirzinger, G. Catching flying balls with a mobile humanoid: System overview and design considerations. In Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots, Bled, Slovenia, 26–28 October 2011. pp. 513–520.
8. Kim, S.; Shukla, A.; Billard, A. Catching objects in flight. *IEEE Trans. Robot.* **2014**, *30*, 1049–1065. [CrossRef]
9. Salehian, S.S.M.; Khoramshahi, M.; Billard, A. A dynamical system approach for softly catching a flying object: Theory and experiment. *IEEE Trans. Robot.* **2016**, *32*, 462–471. [CrossRef]
10. Schill, M.M.; Buss, M. Robust Ballistic Catching: A Hybrid System Stabilization Problem. *IEEE Trans. Robot.* **2018**, *34*, 1502–1517. [CrossRef]
11. Kao, S.T.; Ho, M.T. Ball-catching system using image processing and an omni-directional wheeled mobile robot. *Sensors* **2021**, *21*, 3208. [CrossRef] [PubMed]
12. Rizzi, A.A.; Koditschek, D.E. Progress in spatial robot juggling. In Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France, 12–14 May 1992. [CrossRef]
13. Reist, P.; Drandrea, R. Design and analysis of a blind juggling Robot. *IEEE Trans. Robot.* **2012**, *28*, 1228–1243. [CrossRef]
14. Poggensee, K.L.; Li, A.H.; Sotsaikich, D.; Zhang, B.; Kotaru, P.; Mueller, M.; Sreenath, K. Ball Juggling on the Bipedal Robot Cassie. In Proceedings of the European Control Conference 2020 (ECC 2020), St. Petersburg, Russia, 12–15 May 2020. [CrossRef]
15. Andersson, R.L. Aggressive Trajectory Generator for a Robot Ping-Pong Player. *IEEE Control Syst. Mag.* **1989**, *9*, 15–21. [CrossRef]
16. Mülling, K.; Kober, J.; Peters, J. A biomimetic approach to robot table tennis. *Adapt. Behav.* **2011**, *19*, 359–376. [CrossRef]
17. Li, H.; Wu, H.; Lou, L.; Kühnlenz, K.; Ravn, O. Ping-pong robotics with high-speed vision system. In Proceedings of the 2012 12th International Conference on Control Automation Robotics & Vision (ICARCV), Guangzhou, China, 5–7 December 2012. [CrossRef]
18. Wang, Z.; Boularias, A.; Mülling, K.; Schölkopf, B.; Peters, J. Anticipatory action selection for human–robot table tennis. *Artif. Intell.* **2014**, *247*, 399–414. [CrossRef]
19. Peper, L.; Bootsma, R.J.; Mestre, D.R.; Bakker, F.C. Catching balls: How to get the hand to the right place at the right time. *J. Exp. Psychol. Hum. Percept. Perform.* **1994**, *20*, 591. [CrossRef] [PubMed]
20. Cesqui, B.; Russo, M.; Lacquaniti, F.; d'Avella, A. Grasping in one-handed catching in relation to performance. *PLoS ONE* **2016**, *11*, e0158606. [CrossRef] [PubMed]
21. Carneiro, D.; Silva, F.; Georgieva, P. The role of early anticipations for human-robot ball catching. In Proceedings of the 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Torres Vedras, Portugal, 25–27 April 2018; pp. 10–16.
22. Hove, B.; Slotine, J.J.E. Experiments in robotic catching. In Proceedings of the 1991 American Control Conference, Boston, MA, USA, 26–28 June 1991; pp. 380–386.
23. Hong, W.; Slotine, J.J.E. Experiments in hand-eye coordination using active vision. In *Experimental Robotics IV*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 130–139.
24. Nishiwaki, K.; Ionno, A.; Nagashima, K.; Inaba, M.; Inoue, H. The humanoid saika that catches a thrown ball. In Proceedings of the 6th IEEE International Workshop on Robot and Human Communication, RO-MAN'97 SENDAI, Sendai, Japan, 29 September–1 October 1997; pp. 94–99.
25. Frese, U.; Bauml, B.; Haidacher, S.; Schreiber, G.; Schäfer, I.; Hahnle, M.; Hirzinger, G. Off-the-shelf vision for a robotic ball catcher. In Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180), Maui, HI, USA, 29 October–3 November 2001; Volume 3, pp. 1623–1629.
26. Riley, M.; Atkeson, C.G. Robot catching: Towards engaging human-humanoid interaction. *Auton. Robot.* **2002**, *12*, 119–128. [CrossRef]
27. Park, G.R.; Kim, K.; Kim, C.; Jeong, M.H.; You, B.J.; Ra, S. Human-like catching motion of humanoid using evolutionary algorithm (ea)-based imitation learning. In Proceedings of the RO-MAN 2009—The 18th IEEE International Symposium on Robot and Human Interactive Communication, Toyama, Japan, 27 September–2 October 2009; pp. 809–815.
28. Kim, S.; Gribovskaya, E.; Billard, A. Learning motion dynamics to catch a moving object. In Proceedings of the 2010 10th IEEE-RAS International Conference on Humanoid Robots, Nashville, TN, USA, 6–8 December 2010; pp. 106–111.
29. Bäuml, B.; Wimböck, T.; Hirzinger, G. Kinematically optimal catching a flying ball with a hand-arm-system. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 2592–2599.
30. Dong, K.; Pereida, K.; Shkurti, F.; Schoellig, A.P. Catch the Ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021. [CrossRef]
31. Yu, H.; Guo, D.; Yin, H.; Chen, A.; Xu, K.; Wang, Y.; Xiong, R. Neural Motion Prediction for In-flight Uneven Object Catching. *arXiv* **2021**, arXiv:2103.08368v1.
32. Allen, P.K.; Timcenko, A.; Yoshimi, B.; Michelman, P. Automated Tracking and Grasping of a Moving Object with a Robotic Hand-Eye System. *IEEE Trans. Robot. Autom.* **1993**, *9*, 152–165. [CrossRef]
33. Deguchi, K.; Sakurai, H.; Ushida, S. A goal oriented just-in-time visual servoing for ball catching robot arm. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Nice, France, 22–26 September 2008. [CrossRef]

34. Cigliano, P.; Lippiello, V.; Ruggiero, F.; Siciliano, B. Robotic Ball Catching with an Eye-in-Hand Single-Camera System. *IEEE Trans. Control Syst. Technol.* **2015**, *23*, 1657–1671. [CrossRef]
35. Sato, M.; Takahashi, A.; Namiki, A. High-speed catching by multi-vision robot hand. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021. [CrossRef]
36. Ardakani, M.M.G.; Olofsson, B.; Robertsson, A.; Johansson, R. Real-time trajectory generation using model predictive control. In Proceedings of the 2015 IEEE International Conference on Automation Science and Engineering (CASE), Gothenburg, Sweden, 24–28 August 2015; doi:10.1109/CoASE.2015.7294220. [CrossRef]
37. Stone, J.A.; Maynard, I.; North, J.S.; Panchuk, D.; Davids, K. Emergent perception–action couplings regulate postural adjustments during performance of externally-timed dynamic interceptive actions. *Psychol. Res.* **2015**, *79*, 829–843. [CrossRef] [PubMed]
38. Sigurdsson, G.A.; Russakovsky, O.; Gupta, A. What Actions are Needed for Understanding Human Actions in Videos? *arXiv* **2017**, arXiv:1708.02696.
39. Yoshikawa, T. *Foundations of Robotics: Analysis and Control*; MIT Press: Cambridge, MA, USA, 1990.