2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Learning to Throw Objects Safely in Multi-Obstacle Environments

Mohammadreza Kasaei[2], Klemen Voncina[1], and Hamidreza Kasaei[1]

***Abstract*— Robotic throwing enables fast and efficient object placement beyond the robot's immediate workspace, but reliable throwing in cluttered environments remains underexplored. Existing approaches, such as TossingBot, learn throwing strategies from visual input but assume obstacle-free settings. In this paper, we address the problem of throwing objects into a target basket while avoiding obstacles placed randomly in the scene. We introduce a potential field state representation that compactly encodes both basket attraction and obstacle repulsion on a fixed-size grid, enabling reinforcement learning (RL) policies to generalize across arbitrary numbers and configurations of obstacles. The policy is initialized from kinesthetic demonstrations and optimized in simulation using three state-of-the-art RL algorithms (SAC, DDPG, TD3). Among these, SAC achieves the most consistent performance across scenarios. We compare the potential field representation against explicit state encodings and demonstrate that it achieves higher success rates and better scalability to unseen obstacle configurations. Real-robot experiments with unseen throwable objects confirm robust sim-to-real transfer, achieving up to 90% success in cluttered scenes. These results demonstrate that PFR provides a practical and robust representation for safe and efficient robotic throwing in unstructured environments. A video showcasing our experiments has been attached to the paper as supplementary material.**

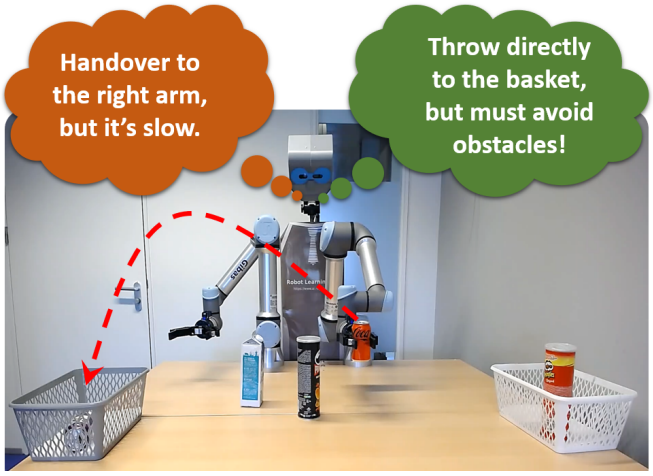

Fig. 1: Overview of our throwing task. The robot must decide between a slow object-handover or a faster throwing action. Since obstacles may block the trajectory, we treat the problem as a **Safe Reinforcement Learning** challenge, where the goal is to achieve reliable throws while avoiding collisions.

I. Introduction

In automated distribution and sorting logistics systems, speed and efficiency play a critical role [1]. Developing robots with the ability to swiftly pick up and throw objects can significantly enhance throughput and operational efficiency. Throwing, in particular, enables fast and precise object placement, reduces the need for complex robotic traversal, and extends the robot's effective workspace [2]. By allowing robots to toss objects instead of carrying them, throughput can be improved in warehouses, automated fulfillment centers, and waste-sorting systems.

Recent works have demonstrated the feasibility of robotic throwing. For example, TossingBot [3] learns grasping and throwing policies from visual input, but assumes obstacle-free environments. Other approaches [4], [5] rely on hand-crafted or analytic motion kernels, limiting adaptability in cluttered settings. However, practical environments rarely allow direct throwing trajectories: obstacles such as walls, bins, or other items must be considered to ensure successful placement.

An illustrative example is shown in Fig. 1. Suppose the robot is asked to place an object into the gray basket. Since the basket lies outside the maximum kinematic reach of the left arm, a simple placement action is not feasible. One option would be to perform a hand-over to the right arm and then place the object into the basket, but this is a relatively slow process and reduces system throughput. Alternatively, the robot can throw the object directly into the basket. However, this introduces additional challenges: the robot must decide whether a safe throwing trajectory exists in the presence of obstacles, and if so, how to execute it reliably. This motivates formulating the problem as a safe reinforcement learning task, where the robot learns to maximize throwing success while avoiding collisions.

In this paper, we address the problem of throwing objects into a target basket in the presence of multiple, randomly placed obstacles. We introduce a potential field state representation that compactly encodes both basket attraction and obstacle repulsion on a fixed grid. Unlike explicit state encodings that scale poorly with the number of obstacles, the potential field provides a fixed-dimensional and physically meaningful input to the learning algorithm, enabling policies to generalize across arbitrary numbers and configurations of obstacles.

To avoid unsafe random exploration in cluttered scenes, we initialize the throwing motion using kinesthetic teaching [6]. These demonstrations provide a safe kernel that bootstraps reinforcement learning and ensures stable early

[1] Klemen Voncina and Hamidreza Kasaei are with the Department of Artificial Intelligence, University of Groningen, The Netherlands. Emails: k.voncina@student.rug.nl, hamidreza.kasaei@rug.nl

[2] Mohammadreza Kasaei is with the School of Informatics, University of Edinburgh, UK. Email: m.kasaei@ed.ac.uk

training. The kernel is then modulated by policies trained with reinforcement learning. We compare three state-of-the-art RL algorithms (SAC, DDPG, TD3) through this project and analyze their performance on the throwing task. The key contributions of our work are:

- We formulate robotic throwing with obstacles as a safe reinforcement learning problem, where the robot must learn to throw while avoiding collisions (Fig. 1). We leverage kinesthetic teaching to initialize the throwing kernel safely, reducing unsafe exploration during early RL training.
- We introduce a compact potential field state representation that encodes basket and obstacle information in a fixed grid, enabling scalable throwing policies that generalize across arbitrary numbers of obstacles.
- We conduct extensive experiments in both simulation and on a real robot, showing that policies trained with potential fields achieve high success rates and transfer effectively to unseen objects and obstacle configurations.

II. Related work

Robotic throwing has been studied through a variety of approaches, ranging from analytic motion planning and physics-based models to modern reinforcement learning methods [7], [8], [9], [10], [11]. Analytical approaches typically rely on explicit models of object dynamics and release kinematics, enabling precise but brittle solutions. Learning-based methods, on the other hand, allow robots to adapt to unknown dynamics and unmodeled effects, but most prior works focus on obstacle-free scenarios or encode obstacles explicitly in the state, which does not scale to complex environments.

In this section, we first review analytical methods for throwing and highlight their limitations in unstructured settings. We then discuss learning-based methods, emphasizing recent works that combine reinforcement learning with kernel modulation. Finally, we position our contribution within this landscape.

A. Analytical methods

These approaches aim to explicitly model object dynamics, throwing kinematics, and release timing to generate precise throwing trajectories. Early works focused on deterministic motion planning and rigid-body dynamics models for controlled throwing.

For instance, sampling-based motion planning methods have been successfully applied to throwing tasks. Zhang et al. [12] introduced dynamic intermediate state objectives to plan throwing motions effectively. Similarly, LaValle and Kuffner[13] and Kunz and Stilman [14] proposed kinodynamic planning techniques that account for velocity and acceleration constraints, making them suitable for throwing tasks.

More recently, adaptive and impact-driven planning methods have emerged. Liu et al. [10] developed a model for adaptive mobile manipulator throwing, while Zermane et al. [1] proposed impact-driven logistic planning to optimize object release timing. Heavily constraining the problem, such as proposed by Myashita et al. [15], may also help to restrict the solution space sufficiently to approach the problem analytically.

Although these analytical methods can achieve high accuracy, they struggle in dynamic and unstructured environments, where real-world uncertainties (e.g., object properties, obstacle variations, and sensor noise) are difficult to model explicitly [16], [17], [18], [19]. For example [17] describes a throwing/catching robot that uses padded manipulators to avoid having to deal with the complex dynamics of a ball bouncing off a hard surface when being caught. In [15], the robotic manipulator consists of a single joint to constrain the degrees of motion that need to be accounted for. Even [3] constrain the parameter kernel by fixing the release height of the robot, therefore only having to deal with parameters for speed, angle, and release timing. Schwarke et al. introduced a curiosity-driven learning approach for joint locomotion and manipulation tasks, where in an obstacle-free environment, their robot performs a joint location and manipulation task involving throwing a package from a table to a bin [20]. Our work differs from these approaches by leveraging deep reinforcement learning (RL), which can adapt to such uncertainties without requiring handcrafted models.

B. Learning methods

To address the limitations of analytical methods, researchers have explored data-driven and reinforcement learning-based throwing. Kober et al. [5] propose a method that combines reinforcement learning and physical modeling. A significant breakthrough in RL-based throwing came with TossingBot [3], which integrates residual physics modeling into an end-to-end learning framework. This allows the robot to learn object dynamics implicitly, reducing dependence on predefined models. However, TossingBot primarily focuses on obstacle-free scenarios, limiting its applicability in cluttered environments.

More recent RL-based works have considered throwing with obstacles. Kasaei et al. [4] introduced an RL framework for throwing into a moving basket while avoiding a single obstacle. In their approach, the pose of the obstacle is explicitly included in the state vector. While effective for a fixed number of obstacles, this design is not scalable: each additional obstacle increases the state dimension and requires training a new policy for that specific configuration. In contrast, our method employs a compact potential field representation that encodes both basket attraction and obstacle repulsion in a fixed-size grid, allowing a single policy to generalize across arbitrary numbers and configurations of obstacles.

Our approach further differs by leveraging kinesthetic teaching to safely initialize the throwing kernel. Rather than requiring the policy to explore from scratch, demonstrations provide a safe starting point for reinforcement learning in cluttered scenes. We systematically evaluate the scalability of this approach across varying numbers of obstacles and

show transfer to unseen objects, highlighting the robustness of our framework.

III. Methods

A. Preliminaries and Problem Formulation

We formulate robotic throwing as a Markov Decision Process (MDP) with continuous state and action spaces. An MDP is defined by the tuple $(s_t, a_t, p(s_{t+1}|s_t, a_t), r(s_t, a_t))$, where s_t is the state at time t, a_t the action taken, $p(s_{t+1}|s_t, a_t)$ the transition dynamics, and $r(s_t, a_t)$ the reward. The objective of reinforcement learning is to find a policy $\pi(a|s)$ that maximizes the expected discounted return $R_t = \mathbb{E}[\sum_{i=t}^{\infty} \gamma^{i-t} r_{i+1}]$ with discount factor $\gamma \in [0, 1]$. Since our action space is continuous and directly parameterizes the throwing kernel, we employ off-policy algorithms designed for continuous control: Deep Deterministic Policy Gradient (DDPG) [21], Twin Delayed DDPG (TD3) [22], and Soft Actor-Critic (SAC) [23]. These algorithms reuse samples from a replay buffer to improve sample efficiency, making them well-suited for robotic domains where interactions are costly.

We study the task of throwing an object into a target basket in the presence of randomly placed obstacles. The policy receives an observation of the scene and must output throwing parameters that modulate a kernel initialized by kinesthetic teaching. Each episode consists of one throwing attempt: the object is placed in front of the robot, a basket is placed at a random reachable location, and between 0–5 obstacles are placed in the workspace (Fig. 4). The policy is rewarded if the throw lands successfully in the basket without colliding with obstacles.

B. Kinesthetic Teaching for Safe Initialization

To avoid unsafe random exploration at the start of RL training, we use kinesthetic teaching [6]. A human demonstrator physically guides the robot arm through a throwing motion, which defines an initial *throwing kernel* (see Fig. 2). This kernel encodes joint trajectories that are safe and physically feasible, serving as a structured prior. The RL policy does not imitate the demonstration directly, but instead learns to modulate kernel parameters—initial and final shoulder joint values, release time, and throw duration—to adapt to different objects and obstacle configurations. Thus, kinesthetic teaching is not used for personalization, but as a mechanism for safe exploration.

Fig. 2: Kinesthetic teaching of the throwing kernel. A human demonstrator physically guides the robot arm through a throwing motion. This provides a safe initial kernel that bootstraps RL training and avoids unsafe random exploration.

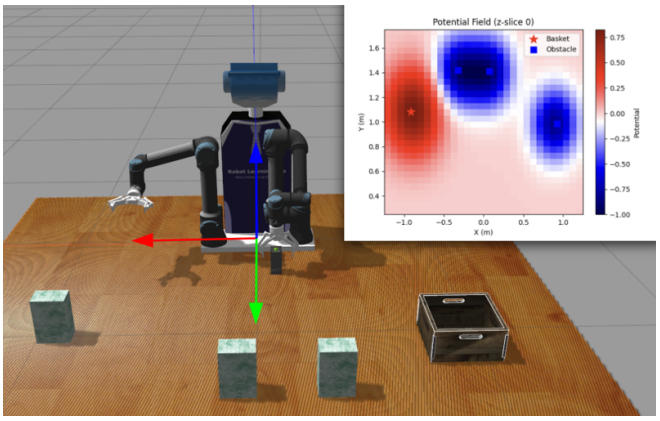

Fig. 3: Example of the Potential Field Representation (PFR). The robot's workspace is discretized into a fixed grid, where the basket generates an attractive potential (red) and obstacles generate repulsive potentials (blue). This yields a structured, fixed-size encoding that generalizes across different numbers of obstacles.

C. State Representation

The state representation combines both robot-centric variables and task-level information. At its core, each state includes a *proprioceptive block* consisting of the initial and final values of the shoulder joint (j_i, j_f), and a *timing block* capturing the gripper release time t_r and the overall motion duration θ. Together, these four variables describe the kernel parameters that directly modulate the throwing action. In addition, the state contains the position of the throwable object (x_o, y_o, z_o) and the basket (x_b, y_b, z_b) expressed in the robot base frame, as well as their relative distances (d, d_x, d_y). Including explicit distances accelerates learning by allowing the policy to reason about spatial relationships without inferring them solely from raw coordinates.

The main difference between the variants of our method lies in how obstacles are represented. In the **Explicit Pose Representation (EPR)**, each obstacle is encoded by appending its center position and its relative distance to the robot frame. This produces a compact, low-dimensional observation vector that allows fast convergence, but its dimensionality grows linearly with the number of obstacles, making it suitable for scenarios with known obstacles. Therefore, we construct an *observation space* for each of our test conditions. For example, the state of an environment containing the throwable, the goal object, and no obstacles would look like this $s = (proprio, timing, pos_o, pos_g, dst_g) \in \mathbb{R}^{13}$. To this basic state, we can then add the representation of an obstacle as well. Thus, this environment with one obstacle would record a state in the following format; $s = (proprio, timing, pos_o, pos_g, dst_g, pos_{obs}, dst_{obs}) \in \mathbb{R}^{19}$ any further obstacles each add 6 more observable values to the state. Of course, these variables are bounded by certain constraints. All the positional variables will appear within the pre-determined range of the robot's workspace.

In contrast, the **Potential Field Representation (PFR)**

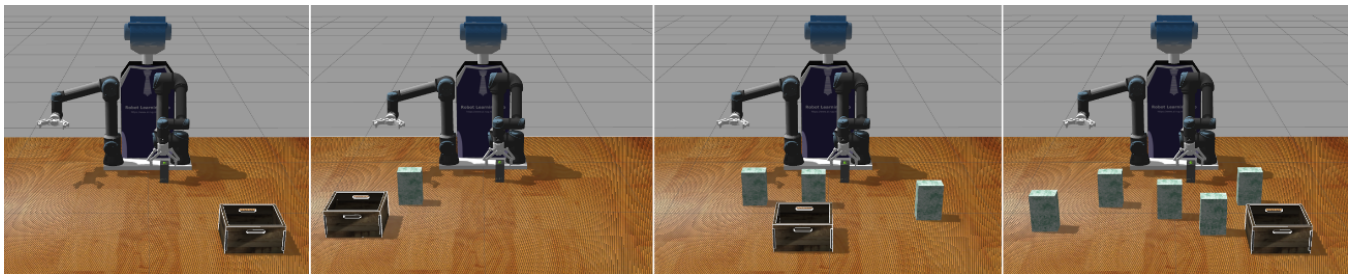

Fig. 4: Examples of environment configurations: no obstacles (left), 1 obstacle, 3 obstacles, and 5 obstacles (right). The robot must learn which obstacles block the throwing trajectory and which do not.

encodes all obstacles and the basket jointly in a fixed-size grid covering the robot's workspace (see Fig. 3). The workspace is discretized into an $M \times M$ grid (with $M = 15$). Each cell stores a continuous potential value: attraction from the basket (positive) and repulsion from obstacles (negative), decaying smoothly with distance. This produces a structured representation whose dimensionality is independent of the number of obstacles, allowing a single policy to generalize across arbitrary obstacle counts and configurations. We then introduce a convolutional neural network (CNN) encoder for the PFR. Instead of flattening, the potential field is treated as a single-channel image of size $(1, M, M)$, which is processed by a CNN backbone to extract spatial features. These features are then concatenated with proprioceptive variables (joint values, release time, and throw duration) before being passed to the policy network. This design preserves the geometric structure of the environment, enabling the policy to more effectively exploit the smooth gradients of the potential field and the shape of obstacles. While PFR requires more training interactions due to its higher dimensionality, it provides scalability and a physically meaningful encoding that supports policies trained once to be deployed across arbitrary numbers of obstacles.

D. Action Space

The action space directly parameterizes the throwing kernel that the robot executes. Each action is a four-dimensional continuous vector, $a_t = (j_i, j_f, t_r, \theta)$, where j_i and j_f denote the initial and final values of the shoulder joint, t_r is the release time of the gripper, and θ is the duration of the throwing motion. Given these parameters, the controller generates a trajectory in which the shoulder joint moves from j_i to j_f, the second joint extends with a fixed amplitude provided by the kinesthetic demonstration, and the gripper opens at t_r within the total motion duration θ. This formulation allows RL to modulate a safe kernel initialized from demonstration rather than learning raw joint trajectories from scratch, thereby reducing the risk of unsafe exploration and focusing the search on physically meaningful throwing motions.

E. Environment Setup

A simulated environment is implemented using Gazebo and ROS [24], [25], which mirrors the real robot with two UR5e arms and an Xtion vision system mounted on the head of the robot. Each training episode consists of a single step, therefore, two observations (before and after) and one action. In each experiment, an object is placed in front of the robot to be thrown. The goal basket is placed randomly in the robot's vicinity. If there are obstacles in the environment, they are placed randomly in the scene, ensuring that they do not collide with each other, the goal basket, or a throwable object. The random placement of obstacles, even when some may appear distant from the basket or outside the robot's immediate workspace, serves a purpose in our study (refer to Fig. 4, the last two images on the right). It allows the robot to discern and learn which objects should be considered obstacles preventing the throwing trajectory toward the target basket and which objects are not obstructive to reaching the target position. By exposing the system to various spatial configurations, including those where certain obstacles seem less relevant, we enable the robot to develop a nuanced understanding of the relevant obstacles and refine its decision-making process during the learning phase.

F. Reward

The reward for the action is calculated based on the state as recorded at the end of the training episode. Firstly, a few conditions are checked. If an obstacle has been moved (caused by collision) the reward is set to 0, regardless of whether it reaches the goal, if the object is inside the goal box (checked by distance and axis-aligned bounding box collision calculation), the reward is 1, otherwise, the reward is calculated using the following function:

$$r = \rho - \sigma \tag{1}$$

where the reward term ρ is described by

$$\rho = 0.9e^{-100d_g^2} + 0.1e^{-2d_g^2} \tag{2}$$

where the choice of exponents e^{-100} and e^{-2} ensures rapid decay for large errors and allows gradual improvement, respectively. The e^{-100} term decays extremely fast, meaning that a throw far from the goal gets almost no reward, strongly discouraging bad throws, while the e^{-2} term decays more slowly, meaning even suboptimal throws receive some reward, helping the policy refine its performance incrementally. This approach prevents the reward from becoming sparse. The penalty term σ is given by

$$\sigma = (1 - e^{-10d_r^2}) \tag{3}$$

d_g is the same distance that appears in the observation between the object and obstacle at the final position of the training episode and d_r is the relative distance of the object to the goal measured as the difference between d_g at the start of the episode and d_g at the end of the episode. Additionally, should this term be positive (the object is closer than it was at the start), $d_r = 0$.

This ensures that we only penalize the throw if the object ended up further away after the throwing action, meaning the robot threw it in a random direction.

IV. Experiments and Results

We validate our approach through extensive experiments in both simulation and on a real robotic platform. Our evaluation has two main objectives: (i) to compare two alternative obstacle encodings, the Explicit Pose Representation (EPR) and the Potential Field Representation (PFR), in terms of learning efficiency and scalability, and (ii) to assess the generalization of learned throwing policies to unseen objects and their transferability from simulation to the real robot. Performance is measured in terms of the *throwing success rate*, defined as the ratio of successful throws that land inside the basket to the total number of attempts.

A. Training Setup

Policies are trained in simulation using the milk box object as the primary throwable and then evaluated on both the milk box and three previously unseen objects (banana, coke can, and sneaker). For EPR, policies are trained for 100k timesteps, while PFR requires 250k timesteps due to its higher-dimensional observation space. We consider four obstacle scenarios with $0, 1, 3,$ and 5 obstacles randomly placed in the workspace (Fig. 4). Training in simulation allows safe and efficient exploration, after which the best-performing policies are transferred zero-shot to the real robot.

Our implementation is based on Stable-Baselines3 [26], and Table I summarizes the hyperparameters used across SAC, TD3, and DDPG. The perception pipeline for real-robot tests (Fig. 7) builds on prior work in object detection and grasp synthesis [27], [28], [29], [30].

B. Ablation on Potential Field Grid Resolution

We further investigated the effect of grid resolution in the Potential Field Representation (PFR). In this experiment, we compared three grid sizes: 10×10, 15×15, and 30×30.

Figure 5 illustrates representative potential fields at two different resolutions. With a coarse 10×10 grid, the field is blocky, and small obstacles are represented by only a few cells, making their influence less precise. Increasing the resolution to 15×15 produces smoother gradients, while a 30×30 grid yields even finer details, capturing obstacle boundaries and basket attraction regions more clearly.

Quantitatively, we observe that increasing the grid size improves generalization to unseen objects and cluttered environments. For example, with SAC in the three-obstacle scenario, success rates for the milk box improve from 0.89 with a 10×10 grid to 0.96 with a 15×15 grid, and 0.97 with a 20×20 grid. However, the computational cost also increases: the 10×10 grid converges after approximately 200k steps, while the 20×20 grid requires closer to 300k steps for stable performance. These results reveal a trade-off between training efficiency and spatial fidelity. Coarser grids train faster but lose fine-grained obstacle information, whereas denser grids yield smoother and more discriminative potential fields at the expense of longer training. In practice, we find the 15×15 grid offers the best balance between efficiency and accuracy, and we adopt this setting in subsequent experiments.

TABLE I: List of hyper-parameters used in this study. For the CNN-based PFR, the potential field grid is treated as a single-channel image and passed through a two-layer convolutional encoder before concatenation with proprioceptive features.

Parameter	Value
#hidden layers (actor/critic MLPs)	2
#hidden units per layer	256
#samples per minibatch	256
optimizer	Adam
learning rate	3×10^{-4}
batch size (SAC)	256
batch size (TD3)	100
batch size (DDPG)	100
#epochs	50K
discount (γ)	0.99
replay buffer size	$50K$
nonlinearity	ReLU
target update rate (τ)	0.005
target update interval	1
gradient steps	1
CNN Encoder (for PFR)	
input channels	1 (potential field)
Conv layer 1	16 filters, 3×3 kernel, stride 1, ReLU
Conv layer 2	32 filters, 3×3 kernel, stride 1, ReLU
Flatten	fully connected + concat with proprioception
final linear layer	256 units, ReLU

C. Simulation Results

Table II reports the performance of policies trained on the milk box when evaluated on both the trained object and three unseen objects across different obstacle scenarios. Among the algorithms, SAC consistently outperforms TD3 and DDPG, maintaining success rates above 90% with up to three obstacles and showing greater robustness as clutter increases. In contrast, TD3 accuracy drops sharply in heavily cluttered scenes, while DDPG exhibits inconsistent generalization across objects. Interestingly, despite its elongated shape, the banana transfers relatively well under SAC, whereas the soda can and sneaker prove more challenging, reflecting

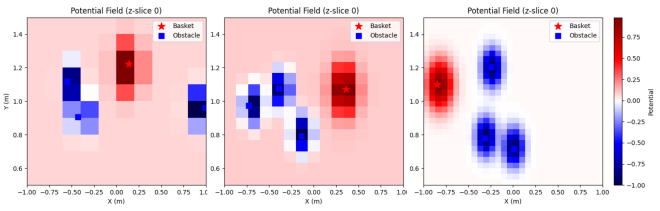

Fig. 5: Effect of grid resolution on the Potential Field Representation (PFR). (*left*) A 10×10 grid provides a coarse and blocky representation, where small obstacles occupy only a few cells. (*middle*) A 15×15 grid offers a clearer encoding of basket attraction and obstacle repulsion. (*right*) A 30×30 grid produces smooth gradients and sharper obstacle boundaries, capturing fine details at the expense of increased training cost.

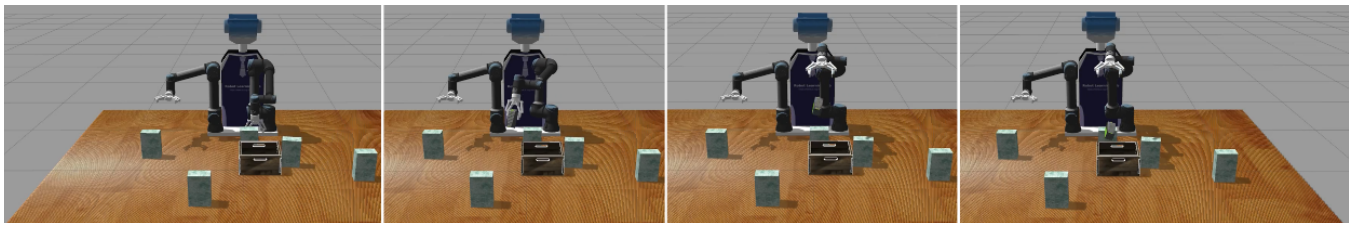

Fig. 6: Sequence of snapshots showing the policy successfully delivers the object into the target basket while navigating around multiple obstacles, demonstrating its ability to generate safe and effective throws in cluttered environments.

TABLE II: Throwing accuracy for the proposed model. Bold indicates the trained object; others are unseen test objects.

Alg.	#Obst	**Milk**		Banana		Coke Can		Sneaker	
		EPR	PFR	EPR	PFR	EPR	PFR	EPR	PFR
SAC	0	0.95	0.97	0.91	0.94	0.85	0.93	0.94	0.94
SAC	1	0.95	0.97	0.92	0.94	0.81	0.91	0.91	0.93
SAC	3	0.93	0.96	0.83	0.90	0.76	0.90	0.90	0.93
SAC	5	0.86	0.92	0.75	0.83	0.73	0.82	0.82	0.87
TD3	0	0.92	0.93	0.72	0.89	0.71	0.89	0.84	0.89
TD3	1	0.87	0.91	0.74	0.86	0.43	0.83	0.78	0.81
TD3	3	0.79	0.84	0.66	0.86	0.35	0.75	0.71	0.81
TD3	5	0.70	0.79	0.48	0.81	0.40	0.75	0.57	0.76
DDPG	0	0.88	0.91	0.50	0.65	0.51	0.72	0.79	0.84
DDPG	1	0.92	0.92	0.67	0.67	0.58	0.67	0.83	0.84
DDPG	3	0.92	0.90	0.70	0.68	0.70	0.67	0.88	0.80
DDPG	5	0.83	0.87	0.67	0.68	0.69	0.68	0.74	0.81

the difficulty of generalizing to objects with different mass distributions and centers of gravity.

A key result is that the PFR often surpasses the EPR. For example, with SAC and three obstacles, PFR reaches 0.90 accuracy on the Coke Can compared to 0.76 with EPR. Similarly, under TD3, PFR improves performance on the Banana from 0.66 to 0.86 in the same setting. Even with DDPG, which generally lags behind SAC and TD3, PFR remains competitive; in the five-obstacle scenario with the Sneaker, PFR outperforms EPR (0.81 vs. 0.74). These results highlight the benefit of preserving spatial structure through CNN-based encoding rather than flattening or relying on explicit pose lists. When comparing EPR and PFR more broadly, a clear trade-off emerges between efficiency and scalability. EPR converges faster, typically reaching stable performance after about 100k training steps, but requires a fixed obstacle count and thus a separate policy for each scenario. In contrast, PFR requires longer training (250k steps) but produces a single policy that generalizes to arbitrary numbers and configurations of obstacles, making it more suitable for deployment in unstructured environments. This demonstrates that while EPR is efficient for controlled settings, PFR provides a scalable and robust representation for safe throwing in cluttered environments.

Although our policies achieve high success rates in simulation, we observed three recurring sources of failure:
(i) Collisions with obstacles: In cluttered scenarios, some throws result in trajectories that intersect with one or more obstacles. This typically occurs when the policy selects joint angles or release timings that generate shallow parabolic arcs. Upon contact, the object either bounces unpredictably or drops short of the basket.
(ii) Underthrows and overthrows: Another common failure arises when the throwing release timing is incorrectly predicted. In these cases, the object either falls short of the basket (underthrow) or overshoots it entirely (overthrow). While less frequent in obstacle-free conditions, these errors become more common as the environment grows more cluttered and valid throwing windows shrink.
(iii) Suboptimal exploration in cluttered scenes: With three or more obstacles, the solution space narrows considerably, making policy optimization more challenging. Occasionally, the policy converges prematurely to a safe but suboptimal strategy, such as consistently throwing away from obstacles but also missing the basket.

These failure modes highlight the inherent difficulty of learning robust throwing policies in highly constrained environments.

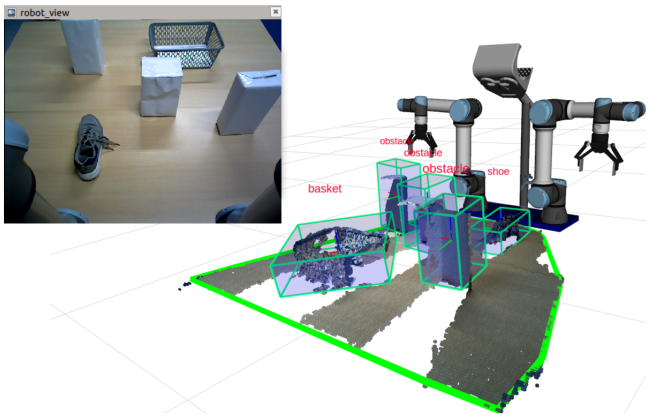

Fig. 7: Example output of our real-robot perception system, showing object detection and recognition results.

D. Real Robot Experiments

To directly compare the transferability of EPR and PFR, we evaluated both approaches on the real robot with three obstacles using the SAC algorithm. Each policy was trained in simulation and deployed zero-shot on the real robot without fine-tuning. For each configuration, we performed 20 throwing trials using a sneaker as the throwable object and tested with both a basket of comparable size to the one used in simulation and a larger basket approximately 1.5× wider (see Fig. 8). The model was provided with the same state information as in simulation; however, all perception data were obtained from an integrated perception pipeline described in [30], [27], [29], which performs object detection, recognition, and pose estimation in real time. Figure 7 illustrates an example output of the perception

system, showing accurate detection of multiple objects and their corresponding 6D poses.

Table III summarizes the success rates. With the normal basket, PFR achieves higher accuracy (70%) than EPR (60%), demonstrating better robustness to perception noise and release-time variability. The advantage of PFR is even more pronounced with the larger basket, where it consistently reaches 90%, while EPR remains was 70%). These results confirm the scalability advantage of PFR: whereas EPR requires explicit encoding of each obstacle and thus is sensitive to pose estimation errors, PFR maintains a fixed state representation and generalizes more effectively to real-world clutter. While success rates are slightly lower than in simulation, the relative trend between EPR and PFR remains consistent, highlighting the robustness of sim-to-real transfer.

By comparing the obtained results, we draw two key conclusions. First, the learned policies transfer effectively from simulation to the real robot, achieving high accuracy even with unseen throwable objects such as the sneaker. Second, performance is influenced by the target basket geometry: larger baskets mitigate the effects of perception noise and release-time variability, leading to higher success rates. Together, these findings confirm the feasibility of deploying the proposed method in real-world scenarios and demonstrate that our approach enables both safe and effective throwing in cluttered environments.

Despite the overall high success rates, we observed three recurring sources of failure during the real-robot experiments.

(i) Inaccurate object pose estimation: This refers to the challenge of precisely determining the position and orientation (pose) of the object in the robot's environment [31]. This lag in tracking accuracy can lead to errors in predicting the object's trajectory during the throw, resulting in landing the object near, but not inside, the target basket.

(ii) Delay in gripper command execution: This issue arises due to the way the gripper is controlled. Unlike direct control, where the user can open and close the gripper instantly, it is governed by the robot's controller, adding a multi-step process. First, the user sends a command to the robot's controller, which then relays it to the gripper. However, the timing of this process is subject to network conditions and the robot's current operational state. Consequently, there can be a delay in the execution of gripping commands, potentially affecting the precision and timing of the object release during the throw. [32] proposed an approach for addressing such releasing time uncertainty.

(iii) Unstable grasp pose selection: A grasp that is stable for pick-and-place manipulation is not necessarily optimal for throwing, as the grasp configuration directly influences dynamics, object orientation, and resulting trajectory [33], [34]. Moreover, if the selected grasp pose is not stable or secure, it can lead to difficulties in maintaining a firm grip on the object during the throwing motion. These instabilities can result in deviations from the desired trajectory, contributing to failures in accurately delivering the object to the target basket.

These identified factors collectively contribute to the challenges faced in achieving successful object throws, in real robot experiments. Addressing these issues is crucial for improving the accuracy and reliability of the throwing process.

TABLE III: Real-robot throwing success rates (20 trials per setting) with SAC under EPR and PFR.

Basket Size	#Trials	SAC-EPR	SAC-PFR
Normal (sim-sized)	20	0.60 (12/20)	0.70 (14/20)
Large (1.5×)	20	0.70 (14/20)	0.90 (18/20)

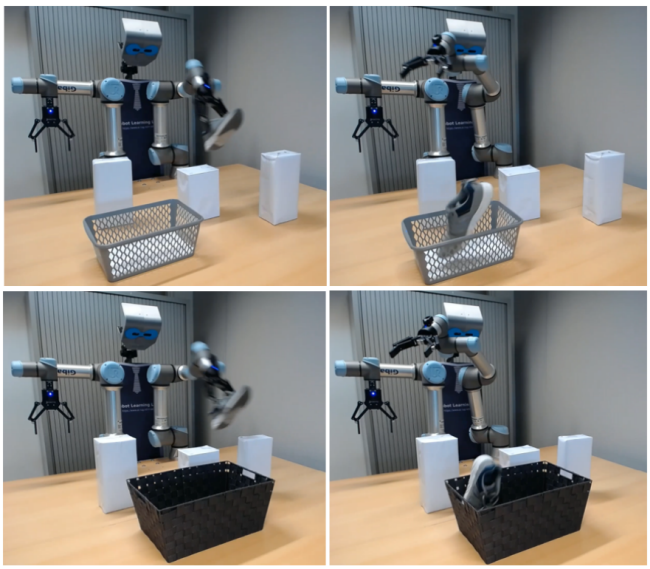

Fig. 8: Throwing a sneaker object into the basket while avoiding obstacles: (*top row*) an experiment with a basket size similar to the training phase; (*bottom row*) an experiment with a basket size 1.5 times larger than the one used during training.

V. Conclusion

In this work, we addressed the problem of safe object throwing in cluttered environments, where a robot must deliver objects into a target basket while avoiding collisions with obstacles. We explored two complementary state representations: the Explicit Pose Representation (EPR), which directly encodes obstacle positions but requires a fixed obstacle count, and the Potential Field Representation (PFR), which provides a scalable, fixed-size encoding independent of the number of obstacles. To enhance the effectiveness of PFR, we introduced a CNN-based encoder that preserves the spatial structure of the potential field, enabling the policy to learn obstacle boundaries and basket attraction gradients more effectively.

Through extensive simulation experiments, we showed that SAC consistently outperforms TD3 and DDPG, achieving high success rates across both EPR and PFR. Importantly, PFR not only closes the performance gap with EPR but, in many cases, surpasses it, while retaining the scalability advantage of handling arbitrary obstacle configurations. Real-

robot experiments further confirmed the feasibility of our approach: policies trained in simulation transferred effectively to the physical robot, achieving up to 90% success with unseen objects in cluttered scenarios.

Overall, this work demonstrates that potential field representations combined with reinforcement learning offer a practical and robust solution for safe robotic throwing. By balancing accuracy, scalability, and transferability, our approach brings robotic throwing a step closer to real-world deployment in logistics, manufacturing, and service applications. Looking ahead, several exciting directions remain open. One avenue is to extend the framework to dynamic multi-robot settings, where one robot throws an object, and another robot is responsible for catching it, or dual-arm coordination for throwing larger or heavier objects. Another promising avenue is to extend our framework to dynamic environments with other policy learning approaches, such as Diffusion Policy or Diffusion Policy Optimization (DPPO), where both obstacles and targets may move unpredictably.

REFERENCES

[1] A. Zermane, N. Dehio, and A. Kheddar, "Planning impact-driven logistic tasks," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2184–2191, 2024.

[2] F. Raptopoulos, M. Koskinopoulou, and M. Maniadakis, "Robotic pick-and-toss facilitates urban waste sorting," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1149–1154.

[3] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," 2019.

[4] H. Kasaei and M. Kasaei, "Throwing objects into a moving basket while avoiding obstacles," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3051–3057.

[5] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," 2011.

[6] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, 2012, pp. 391–398.

[7] M. Bombile and A. Billard, "Dual-arm control for coordinated fast grabbing and tossing of an object: Proposing a new approach," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 127–138, 2022.

[8] B. Forrai, T. Miki, D. Gehrig, M. Hutter, and D. Scaramuzza, "Event-based agile object catching with a quadrupedal robot," *arXiv preprint arXiv:2303.17479*, 2023.

[9] M. Bombile and A. Billard, "Bimanual dynamic grabbing and tossing of objects onto a moving target," *Robotics and Autonomous Systems*, p. 104481, 2023.

[10] Y. Liu, A. Nayak, and A. Billard, "A solution to adaptive mobile manipulator throwing," in *2022 Ieee/Rsj International Conference On Intelligent Robots And Systems (Iros)*. IEEE, 2022, pp. 1625–1632.

[11] M. Monastirsky, O. Azulay, and A. Sintov, "Learning to throw with a handful of samples using decision transformers," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 576–583, 2022.

[12] Y. Zhang, J. Luo, and K. Hauser, "Sampling-based motion planning with dynamic intermediate state objectives: Application to throwing," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2551–2556.

[13] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[14] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3713–3719.

[15] H. Miyashita, T. Yamawaki, and M. Yashima, "Control for throwing manipulation by one joint robot," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1273–1278.

[16] F. Lombai and G. Szederkényi, "Throwing motion generation using nonlinear optimization on a 6-degree-of-freedom robot manipulator," in *2009 IEEE International Conference on Mechatronics*. IEEE, 2009, pp. 1–6.

[17] M. Mason and K. Lynch, "Dynamic manipulation," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, vol. 1, 1993, pp. 152–159 vol.1.

[18] Y. Gai, Y. Kobayashi, Y. Hoshino, and T. Emaru, "Motion control of a ball throwing robot with a flexible robotic arm," *International Journal of Computer and Information Engineering*, vol. 7, no. 7, pp. 937–945, 2013.

[19] C. H. Kim and S. Sugano, "Executing optimized throwing motion on robot arm with free joint," *Advanced Robotics*, vol. 30, no. 24, pp. 1571–1578, 2016.

[20] C. Schwarke, V. Klemm, M. Van der Boon, M. Bjelonic, and M. Hutter, "Curiosity-driven learning of joint locomotion and manipulation tasks," in *Proceedings of The 7th Conference on Robot Learning*, vol. 229. PMLR, 2023, pp. 2594–2610.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[22] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[25] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[26] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[27] H. Kasaei and M. Kasaei, "Mvgrasp: Real-time multi-view 3d object grasping in highly cluttered environments," *Robotics and Autonomous Systems*, vol. 160, p. 104313, 2023.

[28] H. Kasaei, S. Luo, R. Sasso, and M. Kasaei, "Simultaneous multi-view object recognition and grasping in open-ended domains," *arXiv preprint arXiv:2106.01866*, 2021.

[29] S. H. Kasaei, N. Shafii, L. S. Lopes, and A. M. Tomé, "Interactive open-ended object, affordance and grasp learning for robotic manipulation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3747–3753.

[30] S. H. Kasaei, M. Oliveira, G. H. Lim, L. S. Lopes, and A. M. Tomé, "Towards lifelong assistive robotics: A tight coupling between object perception and manipulation," *Neurocomputing*, vol. 291, pp. 151–166, 2018.

[31] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, "Self-supervised 6d object pose estimation for robot manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3665–3671.

[32] Y. Liu and A. Billard, "Tube acceleration: robust dexterous throwing against release uncertainty," *IEEE Transactions on Robotics*, 2024.

[33] R. Newbury, M. Gu, L. Chumbley, A. Mousavian, C. Eppner, J. Leitner, J. Bohg, A. Morales, T. Asfour, D. Kragic, *et al.*, "Deep learning approaches to grasp synthesis: A review," *IEEE Transactions on Robotics*, 2023.

[34] A. Ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.