·Self-Supervised Multi-Modal Learning for Collaborative Robotic Grasp-Throw

Yanxu Hou, Zihan Fang, and Jun Li *Senior Member, IEEE*

***Abstract*—Accurate throwing like a human can expand the pick-and-place ability of a manipulator, which is significant but challenging in the field of robotics. Most existing robotic throwing methods neglect the mass of an object and air drag, not to mention the effect of a grasp on the subsequent throw, resulting in inaccurate throws. In this regard, we propose collaborative grasping and throwing learning (CGTL). It consists of a grasp agent with a grasping network (G-Net), a throw agent with a learning-based throw reference (LTR), and a multi-modal throw compensator network (MTC-Net). First, G-Net generates multi-channel grasp affordances for inferring grasps. Subsequently, LTR predicts a throw velocity reference by exploiting an air resistance estimation network (ARE-Net) and a projectile equation considering air drag. Meanwhile, MTC-Net uses multi-modal data to predict the compensation for the throwing velocity reference. Moreover, CGTL takes throwing performances into the reward of the grasp agent and the grasp affordances into the throw agent's observation to facilitate further accurate throwing. Finally, extensive experiments show that our CGTL outperforms its peers regarding throwing accuracy, especially when throwing different objects into new positions.**

***Index Terms*—Robotic throw, robotic grasp, multi-modal data, reinforcement learning, self-supervised learning**

I. INTRODUCTION

PICK-AND-PLACE is a fundamental robotic operation in various industries, such as warehousing, logistics, and manufacturing [1]–[5]. Grasp-and-throw is a beneficial extension to pick-and-place, often used in human operations. Realizing accurate throwing like a human is significant but challenging in robotics.

Currently, most existing robotic throwing systems focus on planning the throwing trajectories or controlling the throwing of given objects, e.g., [5]–[8]. They simplify an object to be thrown into basic geometric shapes, e.g., sphere, triangle, or cube, and leverage a standard projectile equation, i.e., a parabola, to fit the throwing motion given a landing position. Unfortunately, the standard projectile equation neglects crucial physical factors, including aerodynamics, inertia, friction, shape, and mass of the object to be thrown. Moreover, these methods assume that the gripper is located at the object's center of mass, but there are no means to guarantee it. As a result, they exhibit limited generalization to throw various objects and inaccurate land positions.

In recent years, numerous deep learning-based grasping approaches have emerged, such as [9]–[13][14]–[16]. They can grasp novel objects in unstructured environments, surpassing traditional force and form closure grasping methods [17], [18]. However, they rarely involve throwing. Learning-based throwing methods have recently been proposed [19]–[22]. For example, Zeng et al. [23] proposed the first robotic grasp-throw method for manipulating various objects. Nevertheless, they only consider visual clues but disregard the mass of the objects to be thrown, and their oversimplified projectile motion makes it hard to guarantee accuracy throwing. Therefore, accurately grasping and throwing various objects in robot fields remains challenging.

The complicated interdependent parameters involved in projectile motion [23] determine the accuracy of object throwing. These include the object's mass and center of mass, air resistance, grasp friction, and especially pre-throw conditions, i.e., grasp poses. Some parameters can be measured or estimated, e.g., an object's mass and air resistance, while others cannot, e.g., an irregular object's center of mass, grasp friction, and grasping poses. However, acquiring all the parameters to model projectile motion analytically is impractical.

Fortunately, exploiting multi-modal data from different aspects, e.g., the combination of visual features and the mass of objects, has great possibilities for deciding an accurate throw. In addition, deep reinforcement learning (DRL) enables robots to acquire desired behaviors using reward engineering, e.g., robotic grasp in [14], [16], and quadruped robot control in [24], [25], requiring no labeled samples. Consequently, applying DRL with well-designed rewards could improve throwing accuracy.

Inspired by the insight, we propose a self-supervised collaborative grasp-throw learning (CGTL) method to achieve accurate throws of various objects using multi-modal data. CGTL uses a Grasping Net (G-Net) to infer grasps and exploits a learning-based throwing reference (LTR) along with a multi-modal throwing compensation network (MTC-Net) to jointly predict throws. LTR calculates the throwing reference velocity using a projectile equation considering the object's mass and air resistance. We propose an air resistance estimation network (ARE-Net) to estimate the air resistance coefficient of the object to be thrown according to its visual

This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFF0500904, National Natural Science Foundation of China under Grant 61773115, and Shenzhen Fundamental Research Program under Grant JCYJ20190813152401690. (Corresponding author: Jun Li.)

Y. Hou, Z. Fang, and J. Li are with the Ministry of Education Key Laboratory of Measurement and Control of CSE, Southeast University, Nanjing 210096, China (e-mail: yxhou@seu.edu.cn; zhfang_seu@163.com; j.li@seu.edu.cn)

features and mass. Here, the mass is measured by a force sensor at the robot's wrist. MTC-Net infers compensation to the throwing reference velocity with the help of the multi-modal data considering the uncertain throw parameters. Furthermore, CGTL facilitates the cooperation between grasp and throw agents by reward engineering.

The main contributions of this work are as follows.

1) We propose LTR for predicting the aerial trajectory of an object to be thrown as a projectile equation considering air drag. In LTR, ARE-Net is designed to estimate the air resistance coefficient of the object to be thrown according to the object's mass and visual observations. LTR can produce a more accurate throw than the standard throwing projectile given a landing position.
2) CGTL is developed for jointly learning to grasp and to throw in self-supervision. CGTL comprises a throw agent with LTR and MTC-Net and a grasp agent with G-Net for predicting grasps and throws. We integrated throwing performances into the reward of the grasp agent and feed grasp affordances generated by G-Net into MTC-Net to facilitate the collaboration between the grasp and throw agents to reach accurate grasps and throws of various objects.
3) Extensive experiments are conducted on grasping and throwing actual metal workpieces and daily items. The results demonstrate that CGTL is superior to its peers in realizing accurate throwing, even in throwing unseen objects and throwing objects into new positions.

The remainder of this paper is organized as follows. LTR and CGTL are presented in Sections II and III. Experiments are conducted in Section IV. Section V concludes the paper.

II. Learning-based Throw Reference with Vision and Mass

The standard parabolic equations for projectile motion are insufficient to accurately model the aerial trajectory of an object in throwing, especially for various objects. We propose a learning-based throwing reference to predict the aerial trajectory of the object, considering its mass and visual features.

A. Formalization of Throws

A throw action is denoted by $t = (r, v, p_0, \alpha)$, where $r = (r_x, r_y, r_z)$ is the release position and $v = (v_x, v_y, v_z)$ is the release velocity of the object in throwing, $p_0 = (p_{x0}, p_{y0}, p_{z0})$ is an initial position of the grippers, and α is the direction of v. First, the robot grasps an object and move it to p_0 above the object before grasping. Second, the robot performs constant acceleration motion in Cartesian space and then switches to constant velocity when reaching the release velocity with $||v|| = \sqrt{v_x^2 + v_y^2 + v_z^2}$. Finally, the gripper releases the object when reaching r. Here, we suppose that t satisfies 1) v lies in plane (v_x, v_y), i.e., $v_z = 0$, and 2) the distance from p_0 to r is a constant c_d, and r_z is also a constant. Given an

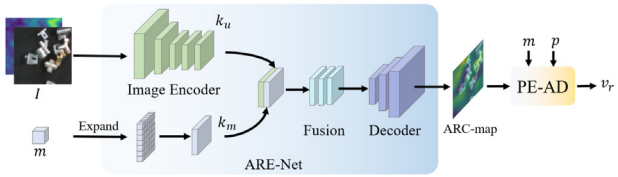


Fig 1. The architecture of ARE-Net. The image and mass encoder extract image and mass features, respectively. Then, the ARC-map is obtained through a fusion module and a decoder. Finally, PE-AD calculates the reference velocity based on the estimated air resistance coefficient, mass, and landing position.

expected land position $p = (p_x, p_y, p_z)$, α, r_x, and r_y can be calculated by,

$$\alpha = \arctan\frac{p_y - p_{y0}}{p_x - p_{x0}}, \tag{1}$$

$$r_x = p_{x0} + c_d \cos\alpha, \tag{2}$$

$$r_y = p_{y0} + c_d \cos\alpha. \tag{3}$$

In other words, α is the angle between x-axis and the line connecting p_0 to the target land position. In this way, a robot can perform t to throw an object to the expected land position by controlling $||v||$.

B. Projectile Equation with Air Drag

As known, air resistance is a primary factor affecting projectile motions [26]. Here, we use a projectile equation with air drag (PE-AD) to solve the initial velocity of a projectile v_0. Generally, at low speeds, air resistance is proportional to the velocity of objects, also known as Stokes drag,

$$f_k = -ks, \tag{4}$$

where k is the air resistance coefficient of the object being throwing and s is the object's speed. Substituting Eq. (4) into a parabola equation, we can obtain the projectile equation with air drag, i.e.,

$$x(t) = \frac{m}{k}(v_0 \cos\alpha)\left(1 - e^{-\frac{kt}{m}}\right), \tag{5}$$

$$y(t) = \frac{m}{k}\left(v_0 \sin\alpha + \frac{mg}{k}\right)\left(1 - e^{-\frac{kt}{m}}\right) - \frac{mg}{k}t, \tag{6}$$

where $x(t)$ and $y(t)$ are the coordinates of the object in throwing at time t in the robot frame, m is the object's mass, and g is the gravity acceleration. Here, $||v||$ is equal to v_0. Therefore, following Eqs. (5) and (6), we can obtain $||v||$, given m, g, α, k, and p. We employ a six-axis force sensor located at the robot's wrist to determine m,

$$m = \frac{F_z - F_{z0}}{g} \tag{7}$$

where F_{z0} and F_z are the initial force and the force after grasping an object along the z-axis, respectively. However, it is difficult to obtain the analytical solution of $||v||$ directly from Eqs. (5) and (6). Therefore, we use a classical numerical method, i.e., a modified Levenberg-Marquardt algorithm [27], to solve $||v||$. Nevertheless, in Eqs. (5) and (6), the determination of k remains a challenge. In this letter, k is approximately estimated from visual and mass information.

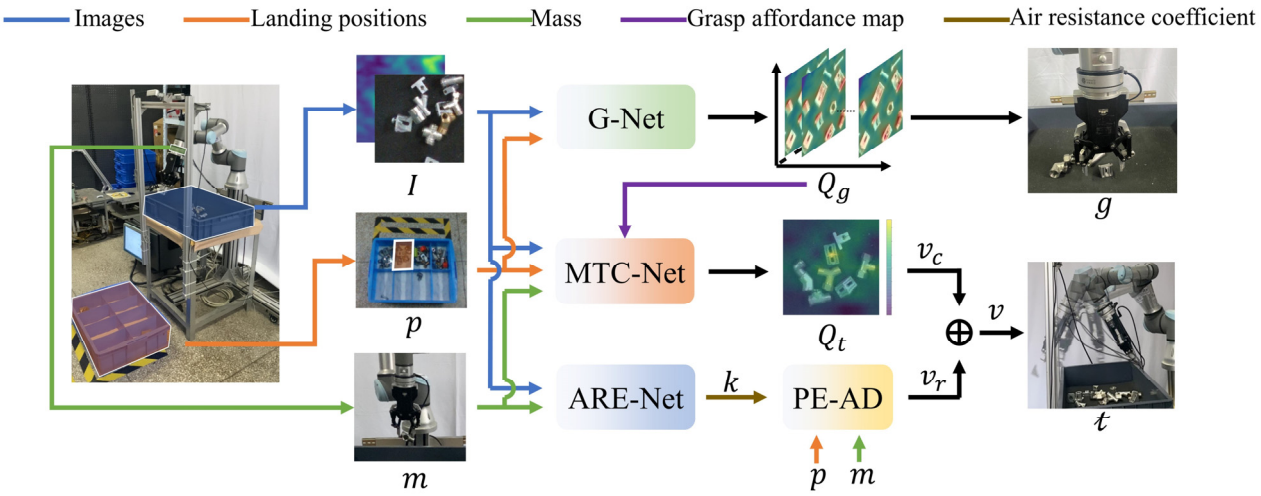


Fig 2.The architecture of CGTL. CGTL includes grasp and throw agents. The grasp agent uses G-Net to predict grasp actions g, and the throw agent has MTC-Net, ARE-Net, and PE-AD for predicting throwing actions t. ARE-Net uses an independent image encoder and G-Net. MTC-Net uses shared image and position encoders to extract RGB-D and position features.

C. ARE-Net for Predicting Air Resistance Coefficient

Our ARE-Net is proposed to estimate k based on the visual features and mass of the thrown object. ARE-Net takes the images and mass of the object as input and outputs an air resistance coefficient map (ARC-map). ARC-map is a two-dimensional map of the same size as the input image. Each pixel represents the air resistance coefficient of the object at the pixel level. ARE-Net, as shown in Fig.1, consists of an image encoder, a mass encoder, a fusion module, and a decoder. The image encoder consists of color and depth encoders to extract image features from RGB-D into a 7×7 shared feature map k_u, where the color and depth encoders exploit ResNet-50 [29], which is commonly used in image segmentation tasks. The mass encoder duplicates and expands the scalar feature m into a 7×7 mass feature map k_m. Then, the two feature maps are concatenated along the channel dimension such that each block of k_u has the same mass information. Finally, the fused features are fed into the decoder to output an ARC-map.

We obtain an actual air-resistant coefficient $\bar{k}$ to train ARE-Net. Assume that an object is thrown to the actual position $\bar{p}$ with velocity $||v||$. We can calculate the actual air resistance $\bar{k}$ by solving Eqs. (5) and (6). Since the same objects in the ARC-map share the same $\bar{k}$, we segment the mask of the object I_m by using Mask-RCNN [28] and then assign masks in I_m with $\bar{k}$. Bitwise multiplication of I_m and I yields a masked image. We use the Smooth L1 loss to update AER-Net accordingly.

$$L(k,\bar{k}) = \begin{cases} \frac{1}{2}(\bar{k}-k)^2, & |\bar{k}-k| < \delta \\ \delta\left(|\bar{k}-k| - \frac{1}{2}\delta\right), & otherwise \end{cases} \quad (8)$$

where δ is used to eliminate the influence of outliers. We pass gradients only through the mask on which the grasping is executed. All other pixels backpropagate with 0 loss. Section IV-D details the data collection of $\bar{k}$ and the training of ARE-Net.

D. Learning-based Throw Reference

By utilizing Eqs. (6), (7), and ARE-Net, we can construct LTR. First, we capture the image of the object using an RGB-D camera and then grasp the object and obtain its mass using a six-axis force sensor mounted at the robot's end. Next, the image and mass are fed to ARE-Net to obtain k. Finally, given m, α, and p, we calculate the throwing reference velocity v_r according to Eqs. (6) and (7). LTR can implement more accurate throws than the standard projectile equation, considering no air drag.

III. Collaborative Grasping and Throwing Learning

CGTL, as shown in Fig. 2, is proposed to achieve accurate throwing. CGTL employs grasping and throwing agents to learn actions through trial and error jointly.

A. G-Net for Predicting Grasps

A grasp $g = (u, \phi)$, where u is the grasp position, and ϕ is the gripper's rotation around Z-axis. We design a Grasping Network (G-Net), as shown in Fig. 3, to infer grasps from visual information. G-Net takes p, a color heightmap I_c, and a depth heightmap I_d captured by an RGB-D camera and outputs a grasp affordance map, where the grasp affordance map represents grasps at the pixel level [15], [16]. Each pixel at the grasp affordance indicates the probability of successfully grasping an object at that pixel. Similar to our prior work [14], G-Net directly outputs a multi-channel grasp affordance map $Q_g = \{q_i\}_{i\in 1:C}$ to avoid redundant rotation on the input image The shape of Q_g is $C \times H \times W$, where $C = 18$ represents the number of rotation angles, and H and W are the height and width of I_c. Each q_i has the same size as the input image. G-Net consists of a shared image encoder, a shared position encoder, a fusion module, and a decoder. The shared image encoder exploits ResNet-50 to extract image features from RGB-D into a 7×7 shared feature map G_u. The

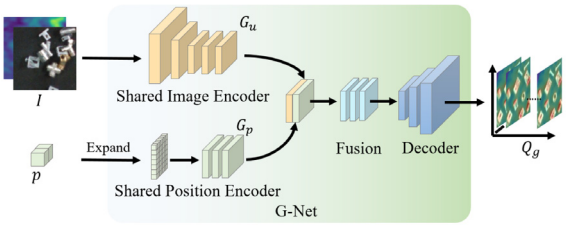


Fig 3. The architecture of G-Net.

shared position encoder transforms $p = (p_x, p_y)$ into a one-dimensional feature of length 49 via a fully connected layer and then be reshaped into a 7×7 two-dimensional features G_p. Note that, G_u and G_p are also used in MTC-Net. The fusion module, consisting of several convolution layers and ReLU activation functions, takes G_u and G_p as inputs and outputs a fusion feature G_{up}. Finally, the decoder uses two deconvolution layers and an upsample operation to upsample G_{up} feature to the same size as I_c.

B. MTC-Net with LTR for Predicting Throws

We calculate $||v||$ by adding a throwing compensation velocity v_c into a throwing reference velocity v_r.

$$||v|| = v_r + v_c, \tag{9}$$

v_r is predicted by using LTR, and v_c is inferred by MTC-Net for compensating the uncertainties caused by unmeasurable parameters. The combination of a reference controller LTR and MTC-Net can be considered deep residual reinforcement learning, which is more effective than traditional deep reinforcement learning in robot control, e.g., [30]–[33].

MTC-Net is designed, as shown in Fig.4, to predict v_c from multi-modal data, taking I_c, I_d, m, Q_g, and p as inputs and generating the throwing compensation to velocity map Q_t, where each pixel in Q_t is v_c. In MTC-Net, the object's mass provides another perspective to understand the object's physical properties. Furthermore, the grasp affordances provide valuable grasping insight for compensating throwing reference velocity. MTC-Net is illustrated in Fig.4. The grasp affordance encoder uses two convolution layers and a ReLU to extract Q_g into G_g. Then, MTC-Net concatenates k_u, G_u, G_p, and G_g along the channel and inputs them into the fusion module to obtain a downsampled feature l. G_u and G_p are predicted by the shared image and position encoders in G-Net. We duplicate and expand p_x, p_y, and m into 2D features with the same size of l. These 2D features are concatenated with l along the channel and input into the fusion module. The fusion module includes several convolutional modules and ReLU activation functions to fuse these multi-modal features. Finally, the decoder consisting of multiple deconvolutions and activation functions outputs Q_t. v_c is obtained by indexing the values at the grasping position in Q_t.

C. CGTL via Self-Supervision

CGTL consists of a grasp agent with G-Net and a throw agent with LTR and MTC-Net to predict grasps and throws. Meanwhile, it facilitates the agents' collaboration through reward engineering. Learning grasps and throws by trial and

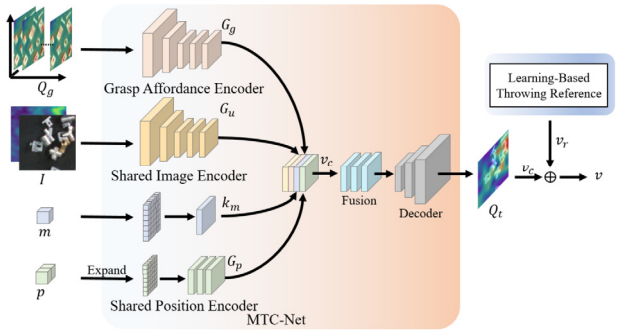


Fig 4. The architecture of MTC-Net. MTC-Net consists of a grasp affordance encoder, a shared image encoder, a mass encoder, and a shared position encoder for extracting multi-modal features. Then, a fusion module and a decoder fuse and expand the features to a velocity map Q_t with the exact size of the input image. Finally, LTR and Q_t are combined to predict the throwing velocity.

error can be modeled as a Markov Decision Process (MDP) $< o, a, r, \rho >$, where o, a, r, and ρ denote observations, actions, rewards, and state transitions, respectively. For the grasp agent, given an observation $o_g = (I_c, I_d, p)$, robots perform g according to G-Net and receive grasp reward r_g. We use Double Deep Q-Learning [34] to train G-Net $Q_{\theta g}$ with parameter θ_g, where $Q_{\theta g}$ can be regarded as an action-value function in Q-learning. We can infer g by

$$g = \underset{g}{argmax} Q_{\theta g}(o_g, g). \tag{10}$$

We collect samples (o_g, g, r_g) and store them into a replay buffer D. Then, we sample data from D and update G-Net by

$$\theta_g \leftarrow \theta + \alpha \Delta_\theta \mathcal{L}(Q_g, y_g) \tag{11}$$

where $\mathcal{L}(Q_g, y_g)$ is a loss function and y_g is the target in Q-learning.

$$y_g = \mathbb{E}_{t=1}^{T}[\gamma^{t-1} r_g] \tag{12}$$

where $\gamma_g = 1$ is a future reward discount. A mean squared error is used as $\mathcal{L}$ to train G-Net,

$$\mathcal{L}(Q_g, y_g) = \frac{1}{B}\sum_{i}^{B}(Q_g - y_g)^2 \tag{13}$$

We also incorporate throwing performance into r_g to promote the collaboration between grasp and throw agents.

$$r_g = \begin{cases} r_0 + r_t & if\ f_g = 1 \\ 0 & if\ f_g = 0 \end{cases} \tag{14}$$

where r_0 is a primary reward and r_t is a throw performance. r_t is obtained by measuring the distance between the actual landing position and the expected one.

For the throw agent, given an observation $o_t = (I_c, I_d, Q_g, m, p)$, robots perform t according to MTC-Net and LTR and receives throw reward r_t. We also use Double Deep Q-Learning to train MTC-Net $Q_{\theta t}$ with parameter θ_t. $Q_{\theta t}$ is trained in the same way as training $Q_{\theta g}$. r_t is derived by measuring the distance between the expected and actual land positions.

$$r_t = \sqrt{(x_p - x)^2 + (y_p - y)^2}, \tag{15}$$

where the actual land position (x_p, y_p, z_p) is detected by

measuring the image difference between the images of the landing zone before and after throwing an object.

D. Training Details

ARE-Net, G-Net, and MTC-Net are implemented using PyTorch, and an ADAM optimizer with a learning rate of 10^{-3} is applied to train them, with a learning rate decay mechanism to improve optimization performance. Before training G-Net and MTC-Net, we train the ARE-Net in an actual robot system. A heuristic grasping strategy is employed to achieve a grasping success rate of 40%, in which objects are randomly grasped and thrown into expected land positions. We collect a dataset containing 1000 samples for training the ARE-Net. In addition, we clip the gradients during training ARE-Net to avoid gradient explosion and vanishing.

We train G-Net and MTC-Net by attempting to grasp and throw various objects into a multi-compartment storage box. Note that, the shared image and position encoders in G-Net are pretrained on the ImageNet dataset and frozen during training MTC-Net. The robot first acquires a scene image from the RGB-D camera and infers g using G-Net. The robot then executes g. If g is successful, the throw agent predicts $||v||$ based on LTR and MTC-Net. Finally, the robot throws the object and obtains r_t. When the robot finishes throwing objects, we return them to the robot workspace. Note that, during training MTC-Net, we directly use a pre-trained ARE-Net. We execute a total of 2000 attempts, each action lasting about 10 seconds, for a total of about 5.5 hours of training.

E. Evaluation Metrics

The grasp success rate (*GSR*) is used to evaluate grasping performance.

$$GSR = \frac{M_g}{N_g}, \tag{16}$$

where N_g and M_g are the times of all grasps and successful grasps, respectively.

The throw success rate (*TSR*) is used to evaluate the throwing performance.

$$TSR = \frac{M_T}{M_g} \tag{17}$$

where M_T is the number of successful throws. A successful throw means an object is thrown into the desired target compartment. The actual landing position of the object is measured by the image difference between the images of the landing zone before and after the throw.

The completion rate (*CR*) is used to evaluate the performance of grasping and throwing together,

$$CR = \frac{M_{gt}}{N_g}, \tag{18}$$

where M_{gt} is the times of successful throws.

IV. Experiments

A. Experimental setup

To examine the proposed method, we conduct extensive experiments on metal workpieces and daily objects, as shown in Fig. 5(a), with masses ranging from 10 grams to 200 grams. The experimental system consists of two RealSense 435 cameras and a UR3 robot with a two-fingered gripper, as illustrated in the left of Fig. 2. A six-axis force sensor installed at the UR3's wrist is used to obtain the mass of an object. The storage box with eight compartments is located outside the UR3 workspace. The experiments are performed on a server installed Ubuntu 18.04 with an Intel Xeon CPU E5-2620 v3 @ 2.4 GHz and an NVIDIA RTX 3090.

B. Methods for Comparison

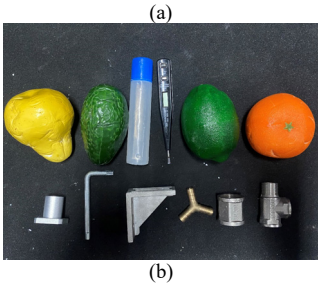
We compare our CGTL with the following methods. (1) A method for residual physics-based grasping and throwing (RPGT) [23]; (2) A method exploiting G-Net to perform grasping and an ideal projectile equation to perform throwing (G-IPE); (3) A method that uses G-Net and LTR to predict grasps and throws (G-LTR); (4) A variant of CGTL with independent image encoders and position encoders in G-Net and MTC-Net, named as CGTL-I; and (5) A variant of CGTL without fusing throwing performance into the grasp reward (CGTL^{-}).

We train the grasping and throwing modules of RPGT using the heuristic grasping strategy presented in Section III-C. The G-Net in G-IPE, G-LTR, and CGTL^{-} are trained similarly to CGTL, but the throwing performance in grasp rewards is removed. The MTC-Net in CGTL^{-} is also trained like CGTL, but the grasp affordances are removed from its inputs. CGTL-I's independent image and position encoders of G-Net and MTC-Net are trained separately. The training process of our CGTL's grasp and throw agents is illustrated in Fig.6.

C. Grasping and Throwing Various Objects

We first evaluate the prediction performance of ARE-Net for three rounds by randomly selecting 10% of the dataset in Section III-D as the test dataset. The mean square error between $\bar{k}$ and k is 0.00445. The method is utilized in Section II-C to test ARE-Net for three runs each 20 times, and the resultant mean square error is 0.00823. Then, predicting air resistance by ARE-Net is affected by various uncertain

(a)



(b)

Fig 5. Experimental objects. (a) Metal workpieces and daily objects and (b) unseen objects.

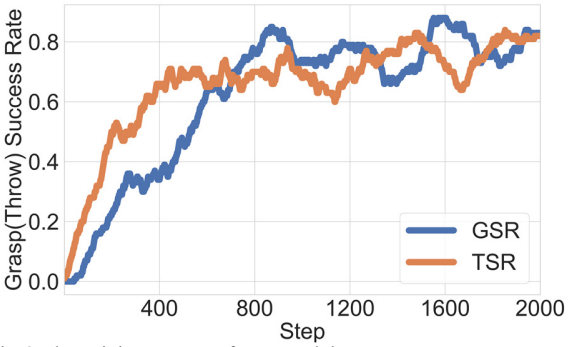


Fig 6. The training process of grasp and throw agents.

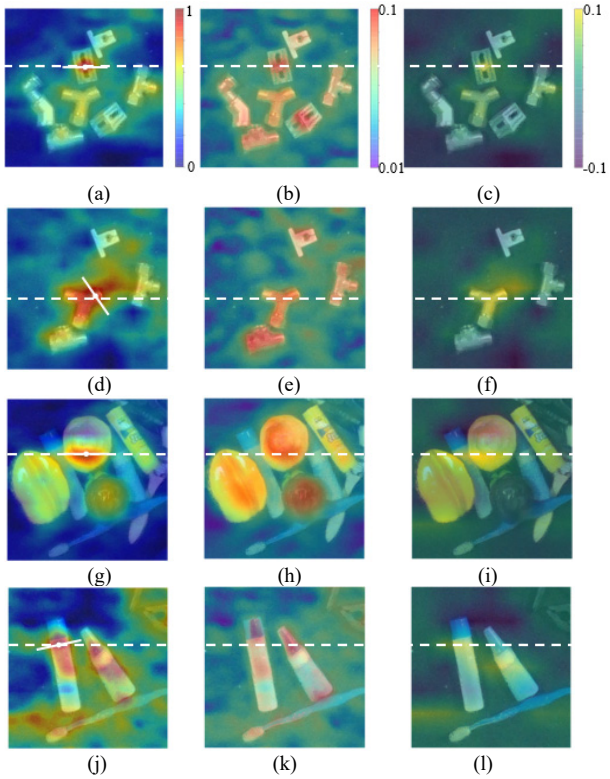


Fig 7. Grasp affordances, ARC-map, and throwing compensation velocity map generated by our CGTL. Grasp affordances, ARC-map, and throwing compensation velocity map are 224 × 224 pixels. The first, second, and third columns show the Grasp affordances, ARC-map, and throwing compensation velocity map in grasping and throwing various objects, respectively. In the first column, the grasps are marked with white broken lines.

factors, such as the accuracy of the landing point position and the grasping position of the object to be thrown. However, overall, LTR containing ARE-Net is superior to the ideal parabolic equations in predicting landing points.

Figure 7 illustrates Q_g, ARC-Map, and Q_t in grasping and throwing objects to its target compartment. Q_g can indicate a successful grasp, and the grasped position nears the object's center of mass, as shown in Fig. 7(a). Objects with similar shapes and masses have similar k, as shown in Fig. 7(b), where brighter colors indicate a greater k. In our experimental setup, k ranges from 0.01 to 0.1, mostly around 0.06.

We compare CGTL with its peers in grasping and throwing various objects, and the results are shown in Table I, where each method is attempted in three rounds every 20 times, a total of 60 times. CGTL achieves the highest *TSR* of 84.3% and *CR* of 71.7%. It is because (1) CGTL uses an air drag-included projectile equation to calculate v_r, while RPGT uses an ideal projectile equation as its throwing reference; (2) MTC-Net considers multi-modal data, including visual observations and object mass, while RPGT only considers visual observations; and (3) CGTL employs an enhanced grasp reward to promote the collaboration between grasping and throwing.

Compared with CGTL$^-$, CGTL, and CGTL-I achieve greater *TSR* and *CR*, which indicates that adding throwing performance into the grasping reward can promote the synergy between grasping and throwing. CGTL$^-$ outperforms G-LTR in *TSR* and *CR*, indicating that MTC-Net can mitigate throwing uncertainties, such as grasp friction and pre-throw conditions. G-LTR outperforms G-IPE in *TSR* and *CR*, implying that LTR is more accurate and suitable as the

Table I Comparison of Methods in Grasping and Throwing Various Objects

	GSR		*TSR*		*CR*	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*
RPGT	85.0%	5.0%	80.4%	4.6%	68.3%	7.6%
G-IPE	78.3%	5.8%	74.5%	8.8%	58.3%	2.9%
G-LTR	81.7%	2.9%	75.5%	6.3%	61.7%	5.8%
CGTL$^-$	83.3%	5.8%	82.0%	4.9%	68.3%	2.9%
CGTL-I	81.7%	2.9%	**85.7%**	3.8%	70.0%	5.0%
CGTL	**85.0%**	8.7%	84.3%	4.7%	**71.7%**	5.8%

Table II Comparison of Methods in Grasping and Throwing Unseen Objects

	GSR		*TSR*		*CR*	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*
RPGT	80.0%	8.7%	72.9%	4.7%	58.3%	7.6%
G-IPE	76.7%	2.9%	60.9%	2.2%	46.7%	2.9%
G-LTR	81.7%	5.8%	63.3%	6.5%	51.7%	7.6%
CGTL$^-$	76.7%	2.9%	73.9%	5.7%	56.7%	2.9%
CGTL-I	80.0%	5.0%	**81.3%**	1.2%	65.0%	5.0%
CGTL	**83.3%**	5.8%	80.0%	7.7%	**66.7%**	2.9%

Table III Comparison of Methods in Grasping and Throwing into New Position

	GSR		*TSR*		*CR*	
	Mean	*Std*	*Mean*	*Std*	*Mean*	*Std*
RPGT	75.0%	5.0%	64.4%	5.5%	48.3%	2.9%
G-IPE	68.3%	2.9%	63.4%	7.3%	43.3%	5.8%
G-LTR	71.7%	5.8%	65.1%	7.3%	46.7%	7.6%
CGTL$^-$	75.0%	5.0%	68.9%	9.7%	51.7%	5.0%
CGTL-I	78.3%	2.9%	76.6%	9.1%	60.0%	5.0%
CGTL	**81.7%**	5.8%	**77.6%**	8.1%	**63.3%**	2.9%

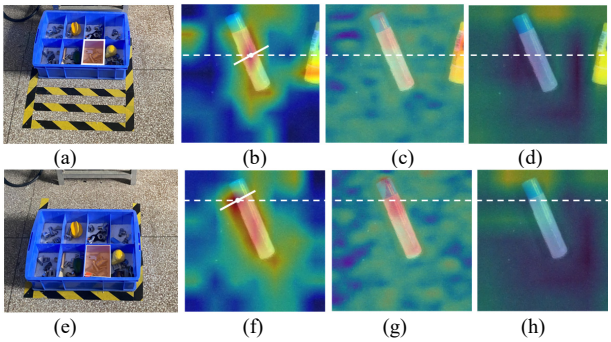


Fig. 8 Grasp affordances, ARC-map, and throwing compensation velocity map generated by our CGTL in throwing stick-shaped to a farther position. (a) and (e) show a near and a far position. The second, third, and fourth columns show the grasp affordances, ARC-map, and throwing compensation velocity map generated by our CGTL, respectively.

Table IV Ablation Study of CGTL

	v_r	m	Q_t	*GSR*	*TSR*	*CR*
CGTL(1)				76.7%	67.4%	51.7%
CGTL(2)	√			78.3%	76.6%	60.0%
CGTL(3)	√	√		80.0%	83.3%	66.7%
CGTL(4)	√		√	80.0%	77.1%	61.7%
CGTL	√	√	√	**85.0%**	**84.3%**	**71.7%**

reference velocity than the ideal projectile equation. Furthermore, G-LTR achieves comparable *TSR* to RPGT. It suggests that LTR can achieve comparable results to state-of-the-art throwing methods even without a throwing velocity compensation. Also, LTR's ARE-Net can reasonably estimate various objects' air resistance coefficients. However, G-LTR and G-IPE are inferior to RPGT in *CR*. It demonstrates that they fail to make accurate throws through grasping due to the independent learning of grasping and throwing agents. CGTL achieves higher *GSR* than CGTL-I, suggesting that the pre-trained ResNet-50 in the shared image encoder can better extract visual clues for grasping. However, CGTL achieves slightly lower *TSR* compared to CGTL-I. It may be attributed to the fact that independent image and position encoders in the grasping and throwing agents are more conducive to extracting specific visual and positional features for the grasping and throwing tasks.

D. Grasping and Throwing Unseen Objects

We evaluated the generalization of CGTL and its peers on unseen objects, as shown in Fig. 5(b). The results are presented in Table II. Our CGTL achieves the best *TSR* of 80.0% and *CR* of 66.7%. CGTL and CGTL-I outperform CGTL$^-$, indicating that they can generalize the collaboration between grasping and throwing of unseen objects. CGTL is superior to CGTL-I in *TSR* and *CR*. It reflects that the shared image and position encoder can generalize to CGTL and overpass G-LTR in *TSR* and *CR*. It substantiates that MTC-Net has a solid generalization to compensate for the throwing reference velocity for unseen objects. Meanwhile, G-LTR achieves similar *TSR* and *CR* with G-IPE in throwing unseen objects. It suggests that the generalization of LTR is like that of the ideal projectile equation for novel objects. The reason is that the air resistance coefficient of an object is not only related to its visual and mass information but also influenced by other factors, such as surface roughness and the object's density.

E. Grasping and Throwing Objects into New Positions

We experiment to evaluate the generalization ability of CGTL and its peers in throwing objects to new positions. To achieve this, we move the eight-compartment storage box 10 cm outwards along the y-axis. Table III shows that CGTL achieves the best *TSR* and *CR*. It reflects that CGTL can grasp and throw objects to new landing positions. Additionally, we investigate the collaboration between grasping and throwing, i.e., whether grasps can enhance throws. We move the eight-compartment storage box 30 cm outwards along the *y*-axis, where throwing velocities exceed the maximum velocity of UR3, as shown in Fig.8(e). For instance, as shown in Fig.8(f), CGTL tends to grasp the end of the stick-shaped object to enable a farther throw. Namely, our CGTL can throw objects farther in the same throwing velocity by grasping at different positions on objects.

F. Ablation Study of CGTL

We conduct ablation experiments to examine the effects with and without object mass, grasp affordance, and throwing reference velocity in CGTL. CGTL(1) does not include any of these factors. CGTL(2) only incorporates throwing reference velocity. CGTL(3) includes throwing reference velocity and object mass. CGTL(4) contains throwing reference velocity and grasp affordance. Table IV shows that all three variables contribute to our CGTL in varying degrees. Note that the throwing reference velocity significantly impacts *TSR* compared to the other two variables. It shows that our LTR can more accurately throw objects than traditional methods. Moreover, CGTL(3) outperforms CGTL(2) in *TSR*, indicating that considering an object's mass significantly improves the throw performance of CGTL. This is probably because the mass of an object plays a vital role in determining its aerial projectile and compensating for the uncertainty caused by intractable factors in throwing. In addition, adding grasp affordances to the observation of MTC-Net improves *TSR*. It demonstrates that grasp affordance can facilitate accurate throwing through grasping.

V. Conclusion

In this work, we propose CGTL, a method comprising grasp and throw agents to achieve accurate grasping and throwing. The grasp agent utilizes G-Net to generate multi-channel grasp affordances for inferring grasps. The throw agent predicts a throwing velocity from multi-modal data by jointly inferring a throwing velocity reference and its compensation using LTR and MTC-Net. LTR exploits the projectile equation considering the mass and air resistance of objects being thrown and uses ARE-Net to predict its air resistance coefficients. The collaboration between grasping and throwing is achieved by incorporating the throwing performance into the reward of the grasp agent and the

grasping affordance into the observation of the throw agent. Finally, we extensively compare CGTL with the existing grasp and throw method RPGT and four variants of CGTL, i.e., G-IPE, G-LTR, CGTL^{-}, and CGTL-I, to grasp and throw various objects. The experimental results show that our CGTL outperforms its peers in grasping and throwing various objects, especially achieving the best performances in throwing objects into new positions. Our ongoing work is to study grasp-throw learning by considering the effects of force and torque together.

ACKNOWLEDGMENTS

The authors would like to thank Kingal Automation Technology (Zhenjiang) Co. for providing the UR3 robot for the experiments and the reviewers for their constructive comments, which greatly helped us improve this letter.

REFERENCES

[1] M. Gupta and G. S. Sukhatme, "Using Manipulation Primitives for Object Sorting in Cluttered Environments," *IEEE Trans. Autom. Sci. Eng*., vol. 12, no. 2, pp. 608–614, 2015.

[2] T. Kiyokawa, J. Takamatsu, and S. Koyanaka, "Challenges for Future Robotic Sorters of Mixed Industrial Waste : A Survey," *IEEE Trans. Autom. Sci. Eng.*, vol. PP, pp. 1–18.

[3] C. H. Corbato, M. Bharatheesha, J. van Egmond, J. Ju, and M. Wisse, "Integrating Different Levels of Automation: Lessons from Winning the Amazon Robotics Challenge 2016," *IEEE Trans. Ind. Informatics*, vol. 14, no. 11, pp. 4916–4926, 2018.

[4] D. Guo, H. Liu, B. Fang, F. Sun, and W. Yang, "Visual Affordance Guided Tactile Material Recognition for Waste Recycling," *IEEE Trans. Autom. Sci. Eng*., pp. 1–9, 2021.

[5] M. Leveziel, G. J. Laurent, W. Haouas, M. Gauthier, and R. Dahmouche, "A 4-DoF Parallel Robot with a Built-in Gripper for Waste Sorting," *IEEE Robot. Autom. Lett.,* vol. 7, no. 4, pp. 9834–9841, 2022.

[6] T. Senoo, A. Namiki, and M. Ishikawa, "High-speed throwing motion based on kinetic chain approach," Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2008, pp. 3206–3211.

[7] G. Hassan et al., "Time-Optimal Pick-and-Throw S-Curve Trajectories for Fast Parallel Robots," *IEEE/ASME Trans. Mechatronics*, pp. 1–11, 2022.

[8] Y. Gai, Y. Kobayashi, and T. Emaru, "Motion control and optimization of a ball throwing robot with a flexible arm," in *Proc. 12th International Conference on Motion and Vibration Control*, 2014, vol. 7, no. 7, pp. 937–945.

[9] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Rob. Res.,* vol. 34, no. 4–5, pp. 705–724, 2015.

[10] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *Int. J. Rob. Res*., vol. 39, no. 2–3, pp. 183–201, 2020.

[11] L. Berscheid, P. Meisner, and T. Kroger, "Self-Supervised Learning for Precise Pick-and-Place without Object Model," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4828–4835, 2020.

[12] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Rob. Res*., vol. 37, no. 4–5, pp. 421–436, 2018.

[13] Y. Hou and J. Li, "Learning 6-DoF grasping with dual-agent deep reinforcement," *Rob. Auton. Syst*., vol. 166, p. 104451, 2023.

[14] Y. Hou, J. Li, and I. Chen, "Self-Supervised Antipodal Grasp Learning with Fine-Grained Grasp Quality Feedback in Clutter," *IEEE Trans. Ind. Electron.*, vol. PP, pp. 1–8, 2023.

[15] A. Zeng *et al*., "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," *Int. J. Rob. Res*., Aug. 2019.

[16] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning Synergies between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 4238–4245, 2018.

[17] J. Bohg, A. Morales, T. Asfour, and D. K. Member, "Data-Driven Grasp Synthesis - A Survey," Trans. Robot., vol. 30, no. 2, pp. 289–309, 2014.

[18] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *IEEE International Conference on Robotics and Automation*, 2000, vol. 1, pp. 348–353.

[19] Z. Fang, Y. Hou, and J. Li, "A Pick-And-Throw Method for Enhancing Robotic Sorting Ability via Deep Reinforcement Learning," *Proc. - 2021 36th Youth Acad. Annu. Conf. Chinese Assoc. Autom. YAC*, vol. 2, no. 3, pp. 479–484, 2021.

[20] J. S. Hu, M. C. Chien, Y. J. Chang, S. H. Su, and C. Y. Kai, "A ball-throwing robot with visual feedback," *IEEE/RSJ 2010 Int. Conf. Intell. Robot. Syst*., pp. 2511–2512, 2010.

[21] M. Monastirsky, O. Azulay, and A. Sintov, "Learning to Throw With a Handful of Samples Using Decision Transformers," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 576–583, 2023.

[22] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Bjorkman, "Deep predictive policy training using reinforcement learning," *IEEE Int. Conf. Intell. Robot. Syst*., vol. 2017-Septe, pp. 2351–2358, 2017.

[23] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "TossingBot: Learning to Throw Arbitrary Objects with Residual Physics," *IEEE Trans. Robot*., vol. 36, no. 4, pp. 1307–1319, 2020.

[24] G. Bellegarda and A. Ijspeert, "CPG-RL: Learning Central Pattern Generators for Quadruped Locomotion," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 12547–12554, 2022.

[25] C. Yu, "Multi-Modal Legged Locomotion Framework With Automated Residual Reinforcement Learning," *IEEE Robot. Autom. Lett*., vol. 7, no. 4, pp. 10312–10319, 2022.

[26] G. W. Parker, "Projectile motion with air resistance quadratic in the speed," *Am. J. Phys*., vol. 45, no. 7, pp. 606–610, 1977.

[27] C. Kanzow, N. Yamashita, and M. Fukushima, "Levenberg-Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints," *J. Comput. Appl. Math.*, vol. 172, no. 2, pp. 375–397, 2004.

[28] H. Kaiming, G. Gkioxari, D. Piotr, and G. Ross, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, p. 9.

[29] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, "Deep Residual Learning for Image Recognition," *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, p. 8.

[30] S. Devlin and D. Kudenko, "Theoretical considerations of potential-based reward shaping for multi-agent systems," 10th Int. Conf. Auton. Agents Multiagent Syst. 2011, *AAMAS* 2011, vol. 1, pp. 209–216, 2011.

[31] M. Zhou, Z. Liu, P. Sui, Y. Li, and Y. Y. Chung, "Learning implicit credit assignment for cooperative multi-agent reinforcement learning," *Adv. Neural Inf. Process. Syst.*, 2020.

[32] T. Johannink et al., "Residual Reinforcement Learning for Robot Control," *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada, 2019, pp. 6023-6029.

[33] A. Iscen, K. Caluwaerts, and J. Tan et al., "Policies Modulating Trajectory Generators," *Proceedings of The 2nd Conference on Robot Learning,* 2018.

[34] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *in 30th AAAI Conference on Artificial Intelligence*, 2016, pp. 2094–2100.