TossNet: Learning to Accurately Measure and Predict Robot Throwing of Arbitrary Objects in Real Time With Proprioceptive Sensing

Lipeng Chen, *Member, IEEE*, Weifeng Lu, *Student Member, IEEE*, Kun Zhang, *Student Member, IEEE*, Yizheng Zhang, Longfei Zhao, *Member, IEEE*, and Yu Zheng, *Senior Member, IEEE*

***Abstract*—Accurate measuring and modeling of dynamic robot manipulation (e.g., tossing and catching) is particularly challenging, due to the inherent nonlinearity, complexity, and uncertainty in high-speed robot motions and highly dynamic robot–object interactions happening in very short distances and times. Most studies leverage extrinsic sensors such as visual and tactile feedback toward task or object-centric modeling of manipulation dynamics, which, however, may hit bottleneck due to the significant cost and complexity, e.g., the environmental restrictions. In this work, we investigate whether using solely the on-board proprioceptive sensory modalities can effectively capture and characterize dynamic manipulation processes. In particular, we present an object-agnostic strategy to learn the robot toss dynamics of arbitrary unknown objects from the spatio-temporal variations of robot toss movements and wrist-force/torque (F/T) observations. We then propose TossNet, an end-to-end formulation that jointly measures the robot toss dynamics and predicts the resulting flying trajectories of the tossed objects. Experimental results in both simulation and real-world scenarios demonstrate that our methods can accurately model the robot toss dynamics of both seen and unseen objects, and predict their flying trajectories with superior prediction accuracy in nearly real-time. Ablative results are also presented to demonstrate the effectiveness of each proprioceptive modality and their correlations in modeling the toss dynamics. Case studies show that TossNet can be applied on various real robot platforms for challenging tossing-centric robot applications, such as blind juggling and high-precise robot pitching.**

***Index Terms*—Data-driven modeling, dynamic manipulation, proprioceptive sensing, trajectory prediction.**

Manuscript received 29 October 2023; revised 7 April 2024; accepted 28 May 2024. Date of publication 18 June 2024; date of current version 27 June 2024. This paper was recommended for publication by Associate Editor C. Della Santina and Editor J. Bohg upon evaluation of the reviewers' comments. *(Corresponding author: Yu Zheng.)*

Lipeng Chen, Yizheng Zhang, Longfei Zhao, and Yu Zheng are with the Tencent Robotics X, Shenzhen 518057, China (e-mail: lipengchen@tencent.com; yizhengzhang@tencent.com; longfeizhao@tencent.com; petezheng@tencent.com).

Weifeng Lu is with the Department of Biomedical Engineering, City University of Hong Kong, Hong Kong, and also with the Tencent Robotics X, Shenzhen 518057, China (e-mail: weifenglu2-c@my.cityu.edu.hk).

Kun Zhang is with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, and also with the Tencent Robotics X, Shenzhen 518057, China (e-mail: kun.zhang@connect.ust.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2024.3416009, provided by the authors.

Digital Object Identifier 10.1109/TRO.2024.3416009

NOMENCLATURE

Symbol	Meaning.
$\mathcal{X}$	Vector of multimodal robot proprioceptive perceptions.
$\mathcal{Y}$	Object's flying trajectory.
$\boldsymbol{x}$	Position and orientation of the robot end-effector.
$\dot{\boldsymbol{x}}$	Translational and angular velocity of the robot end-effector.
$\boldsymbol{f}$	Force/torque (F/T) value perceived by the F/T sensor.
m	Number of perception points.
T	Temporal interval between every two perception points.
$\boldsymbol{e}$	Learned implicit representation of robot toss dynamics.
$\boldsymbol{p}$	Position of the object.
$\boldsymbol{o}$	Orientation of the object parameterized in angle-axis.
$\hat{\boldsymbol{p}}$	Predicted position of the object.
$\hat{\boldsymbol{o}}$	Predicted orientation of the object.
$\mathbf{R}$	Orientation of the object parameterized in rotation matrix.
r	Robot's common reference frame.
b	Prescribed body frame of the object.
b′	Mock body frame of the object for proprioceptive tossing.
g	Prescribed grasp frame.
M	Object's mass.
$\boldsymbol{\mathcal{I}}$	Object's moment of inertia about the CoM.
$\boldsymbol{v}$	Translational velocity of the object.
$\boldsymbol{\omega}$	Angular velocity of the object.
$\boldsymbol{F}$	Total force acting on the object's CoM.
$\boldsymbol{\tau}$	Total torque acting on the object's CoM.
θ	Model weights to be learned during training.

I. INTRODUCTION

ROBOT manipulation has been gradually shifting from kinematic, static, and quasistatic to more dynamic applications, thanks to recent advances in robot actuation and perception [1], [2], [3]. One typical scenario of dynamic manipulation is object tossing, which endows robots with the extrinsic dexterity to reorient [4], [5], regrasp [6], [7], [8], or transport objects [9], [10], [11] by the delicate exploit of toss dynamics.

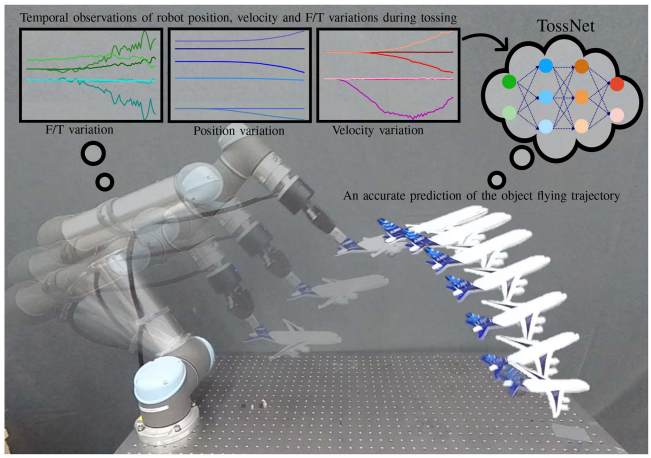


Fig. 1. Learning from the robot's sensation of tossing. Like the human's *ball sense*, we present TossNet, an object-agnostic learning approach that leverages solely the on-board proprioceptive sensory modalities, such as spatio-temporal robot motions and F/T observations, to accurately perceive the characteristics of robot tossing of arbitrary unknown objects and predict their entire flying trajectories in real-time.

Fig. 2. Dependence of toss dynamics on robot grasps. Even if the robot tosses a same object with the same toss movements, different robot grasps on the object will produce different flying trajectories of the tossed object.

However, the underlying difficulties of accurately measuring, sensing, and estimating dynamic manipulation, particularly the motion dynamics of manipulated objects, still remain as one of the most significant barriers for these dynamic applications to come into real practices [1], [2], [12]. Therefore, we investigate herein the use of solely the intrinsic proprioceptive sensory modalities, such as the observed spatio-temporal robot motion and force/torque (F/T) variations, to achieve a lightweight but effective and accurate modeling strategy of dynamic manipulation. As shown in Fig. 1, we focus on the robot tossing of arbitrary unknown rigid objects, a harder class of dynamic manipulation instances. We propose TossNet, an end-to-end formulation that jointly learns the characteristics of robot tossing of arbitrary objects and predicts the resulting flying trajectories of tossed objects with proprioceptive sensory observations in real-time.

Robot tossing by nature consists of continuous and highly dynamic interactions among the robot, the tossed object, and the environment (e.g., air drag). Modeling the toss dynamics accordingly requires accurately measuring not only the relevant physical properties of involved agents, e.g., the inertia and mass distribution of the tossed object, but also the characteristics of their interactions, such as the robot toss movement and the grasp on the tossed object. For example, as shown in Fig. 2, if a robot grasps an object from different locations, the same robot toss movement can produce entirely different flying trajectories of the tossed object. Not uncommonly, as shown in Fig. 3, the robot grasp can also displace from the initial location progressively

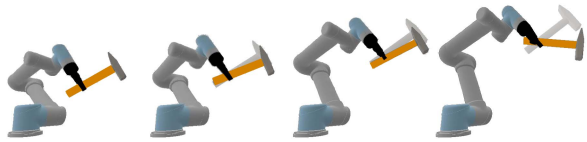


Fig. 3. Displacement of a robot grasp during a toss. The robot's grasp on the object can displace dynamically during tossing due to the accelerated tossing motion and the limited frictional forces. A subtle slip of robot grasp on the tossed object can result in entirely different object trajectories.

during tossing, e.g., the object may slip within the robot gripper due to the accelerated robot motion and the limited friction forces. Such variations are subtle and highly complex due to the close and strong robot–object interactions, and thus hard to be captured with conventional exteroceptive sensory solutions. For example, visual sensors [13], [14], [15], despite being lightweight and easy to deploy, are frequently confronted with difficulties and bottlenecks, e.g., severe noises and occlusion, particularly in tightly-coupled and environment-constrained applications. Recent studies have also been attempting the augmented proprioceptive sensory modalities, such as tactile and visuotactile sensing [16], [17] for dynamic manipulation. Even though promising results have been obtained, e.g., for contact modeling [8], [18], slip detection [19], [20], and physical properties reasoning [8], [21], the underlying complexity and cost in fabrication, calibration, and deployment limit their applications to only specific tasks in well-controlled environments [22]. There still lacks a lightweight and reproducible strategy to accurately capture and model dynamic manipulation.

Predicting the flying trajectory of a rigid object, on the other hand, has been well addressed using the analytical model of rigid dynamics, and/or as nonlinear regression problems with visual observations [23], [24], [25], [26]. Despite significant progress, vision-based methods typically rely on the historical visual observations of the object's state to recognize and predict its follow-up trajectories. Their efficacy and effectiveness heavily hinge on the quality and duration of the historical observations. Hence, these methods usually require the use of multiple high-speed cameras mounted externally to capture the entire object trajectory within their field of view. In addition, many of these approaches rely on prior knowledge about the tracked objects, such as their physical properties, to ensure accurate predictions. In this context, while these methods may be well-suited for tracking objects over long distances and durations, such as catching a far-flying human-tossed object [9], [25], [26], [27], they are frequently confronted with multiple practical barriers when it comes to tracking and predicting the motions of robot-tossed objects. First, occlusions and impairments in visual observations can hardly be avoided in robot-centered environments, particularly during dynamic and rapid movements. Second, most tossing-centric applications, such as robot juggling, typically happen over very short distances and periods, where it is challenging for vision-based methods to gather a sufficient amount of historical visual observations to ensure accurate and high-quality predictions. Third, most tossing-centric applications are also time-sensitive. For example, robot juggling requires real-time and accurate predictions of the object's movements, so that

the robot can have enough time to track and catch the flying objects. However, balancing the need for an ample history of visual observations with the requirement for real-time prediction efficiency presents a dilemma for vision-based methods, as they have to continuously process the visual streams to update the predictions of object trajectories. Currently, there is still a lack of an efficient and reliable method for predicting object motions in dynamic robot manipulation.

We envision from the sequential nature of robot tossing, proposing that the flying dynamics of a robot-tossed object depends mainly on its physical properties and the characteristics of the preceding robot tossing process. In other words, while ignoring the effects of air drag, if the robot toss dynamics can be accurately captured, the resulting flying trajectory of the tossed object can be inferred directly even without visually observing the object in flying. Such an inference process has been commonly applied in human dynamic manipulation: think of how we can, for example, play basketball with foresight. With proper training and practice, an experienced player can anticipate the exact flying trajectory of a basketball (even at the first contact) immediately after holding and tossing it out. This is essentially achieved by the so-called *ball sense* [28], a psychophysical sensation and judgment of the result of a shot or a movement the player develops from lengthy training and practice. Further, even though visual observations can be used, the human relies mainly on the proprioceptive sensation, e.g., the tactile touch, arm muscle tension, and tossing strength, to perceive the properties of the manipulated basketball and the characteristics of a tossing movement. With such an implicit understanding of basketball tossing, the human can reason the result of a toss effortlessly and accurately. In other words, the player learns a model of basketball tossing, which maps from the proprioceptive sensations of tossing a basketball to its resulting fly trajectories. Moreover, the sensation/modeling of the tossing and the inference of the basketball trajectory, in fact, happen almost synchronously and are done immediately after the basketball is tossed out.

In this light, we propose and address the joint task of measuring robot tossing of arbitrary unknown rigid objects and forecasting the subsequent object flying trajectories (see Fig. 1). Inspired by the human's decision process, we build an object-agnostic neural representation of robot tossing of arbitrary objects. It leverages only the multimodal robot proprioceptive features, including the robot motion and the wrist F/T observations. We present TossNet, an end-to-end deep sequence-to-sequence (S2S) neural network, which models and maps the robot toss dynamics to an accurate estimation of the resulting object flying trajectory in real-time. We envision that the proposed capability can further boost the robot's dexterity and versatility in both industrial and domestic applications, and help lead to more capable and collaborative robots.

The proposed TossNet relies on two major components, namely, a set of encoders and a decoder.

1) The encoders are built with long short-term memory (LSTM) modules [29], [30], [31]. Each encoder takes a sequence of single- (or cross-)modal proprioceptive observations of the robot tossing process of an unknown object as inputs to learn the singular (or correlated) features. They are then applied together to build an implicit neural representation of the toss dynamics. Compared with vision or other intrinsic sensory modalities (e.g., tactile), our methods employ only the on-board and well-tested motion encoders and the wrist F/T sensor, making our methods more lightweight and robust, particularly for dynamic robot applications in cluttered and unstructured environments. We demonstrate that our methods can obtain an accurate approximation of the toss dynamics by extracting and fusing multiple single-modal proprioceptive features. In parallel, while every single modality of the proprioceptive observations expresses the toss dynamics to a certain level, we investigate and confirm that the spatio-temporal correlations among different sensory modalities can also contain critical features that facilitate the modeling of toss dynamics. We extract such correlated features using a fine-grained fusion strategy that further improves the modeling accuracy of toss dynamics. Ablation studies are also presented to demonstrate the effectiveness of each modality and their correlations in modeling the toss dynamics.
2) The decoder is also built with LSTM modules, which leverage solely the above learned toss dynamics to extract the flying trajectory of the tossed object in an autoregressive manner. We propose a novel strategy to parameterize the object trajectories consistently in proprioceptive robot tossing. It allows our method to correctly associate the flying trajectories of arbitrary unknown objects with the learned toss dynamics from proprioceptive perceptions. We demonstrate that our method is capable of predicting the object trajectory much more accurately than state-of-the-art methods on a variety of toss scenarios. Moreover, similar to the human's decision process, our method uses only on-board proprioceptive sensors, and can predict the entire long-duration object trajectories in nearly real-time: 1) We construct the proprioceptive encoders with LSTM modules, which enables our method to learn the toss dynamics at runtime and in parallel with robot tossing. 2) Our decoder does not depend on the prior knowledge of tossed objects or the visual observations of the object's flying trajectories. Instead, our method predicts the entire flying trajectory of a tossed object from solely the learned toss dynamics, and therefore can be done in one shot and immediately upon the robot tossing in real-time. Such features make our method an ideal compensation to vision-based methods toward more accurate and efficient prediction of object motions, especially in dynamic robot manipulation.

Experimental results show that our methods can be employed on various real robot platforms, and are capable of accurately modeling their toss dynamics on both seen and *unseen* objects and predicting their seconds-long trajectories in only tens of milliseconds. We demonstrate that our methods can be used in a variety of tossing-centric robot applications, including blind juggling and high-precision pitching. We hope that our simple yet effective pipeline of using proprioceptive sensing could serve

as a strong paradigm to model the robot toss dynamics, and also contribute to other dynamic manipulation modalities, e.g., robot pushing and batting [1], [2], [3].

The rest of this article is organized as follows. Section II reviews the related work. Section III gives a detailed description of the proposed method (TossNet) and a novel strategy to unify the parameterization of object motions in proprioceptive robot tossing. The experimental results are presented in Section IV to evaluate the proposed methods. Finally, Section V concludes this article and discusses some future work.

II. Related Work

Modeling of Toss Dynamics: Robot tossing by nature relies mainly on high-speed and time-varying physical interactions between the robot and object, and therefore accurately sensing and modeling of toss dynamics is challenging [1], [3], [12]. Many previous studies devoted to tossing and other similar instances [32], [33], [34], [35] assume the physical properties of all agents are known a priori, and approximate the task dynamics based on human insights into physics like the frictional rigid body dynamics [36], [37], [38], [39]. However, in most cases of interest, these analytical methods often observe considerable limitations in model accuracy and generalization: A complete knowledge of dynamics is never a realistic assumption, and standard analytical models are only rough approximations due to the unknown nonlinearities inherent in involved agents and their interactions, e.g., the wear and tear of robot end-effector from repeated tossing. Recently, learning models have been proven to be promising in compensating for this lack of nonlinearities in analytical dynamics. These methods are largely task-centric, which address unknown nonlinearities at various levels by learning straightforwardly at least a partial or supplementary model using regression techniques. A large body of previous works have considered direct or augmented modeling of physical properties of each individual agent [21], [40], [41], [42] and their contacts [43], [44], [45], [46] in dynamic manipulation tasks. However, this line of work requires identifying exactly the physical parameters of importance or the very source of unknown nonlinearities in dynamics, which can be quite tricky. In contrast, some early studies have demonstrated the dynamics can be directly learned at the task level to reduce learning degrees of freedom [47], [48]. This line of methods typically builds a complete or residual mapping from robot control parameters to future task states [11], [49], [50], and therefore provides a broader range of data-driven corrections that can compensate for noisy observations and dynamics that are not explicitly modeled. Our work is similar in spirit to these, where we formulate the problem of modeling toss dynamics as to learn a task-dependent mapping from the multimodal proprioceptive observations to a low-dimensional task embedding (see Section III).

Further, rather than modeling the robot toss dynamics directly, a large majority of existing studies focus on the task of target-throwing, i.e., throwing an object into a prescribed target container or toward a specific landing location. This line of research typically involves building an *inverse* dynamics model that maps the object's 2-D landing targets to the robot's control commands [11], [51], [52], [53]. For example, the recent work by Zeng et al. [11] has proposed TossingBot, a rigid robot system that can grasp from a clutter of random objects and throw them into a set of prescribed target boxes. The proposed method leverages a precaptured RGB-D observation before robot tossing to approximate the object-centric dynamics, and maps the object's 2-D landing locations (i.e., planer positions) to the robot's throwing commands (i.e., release velocities). Similarly, another two recent works by Bianchi et al. [51], [52] proposed an inverse dynamics model that can control a soft robot arm to toss four spherical objects into a set of predefined target boxes. This line of works can be particularly efficient and useful in many real-world robot applications, such as cleaning and sorting [54], [55]. Different from these studies, our work can be broadly regarded as a *forward* dynamics model that leverages multimodal proprioceptive observations to capture and model the robot tossing of arbitrary unknown objects, and predict their entire 6-D flying trajectories in real-time. Compared to previous methods, our approach showcases a blend of generality, versatility, reliability, and accuracy in handling the robot tossing of arbitrary unknown objects. As demonstrated in Section IV-D, our approach can tackle a broader range of more complex and challenging tossing-centric robot tasks, including precise pitching and blind juggling, surpassing the capabilities demonstrated in previous works.

Sensing of Robot Tossing: A variety of sensors have been applied in the sensing and estimation of dynamic manipulation, which can be roughly classified into proprioceptive and exteroceptive modalities [12]. Exteroceptive sensory modalities like visual, proximity, and tactile [6], [22], [56], [57], [58], typically provide observations on the states of external agents, which may often suffer from impaired and noisy observation by unstructured environments, and observe significant complexity and difficulties in sensing dynamic manipulation. For example, tactile sensors rely on delicate deployment, laborious calibration, and postprocessing. Proprioceptive sensors like F/T, on the other hand, observe directly physical states of the robot, such as position, velocity, and motor torque, and therefore, are well suited for modeling interactions in more dynamic and robot-centric manipulation [5], [59], [60], [61]. For example, Homberg et al. [61] presented a soft hand capable of detecting the hand configurations and identifying manipulated objects using proprioceptive sensors, such as resistive force and bend sensors. A recent work by Lloyd and Lepora [62] presented a hybrid method for reactive control of goal-driven robot pushing, which employs both tactile feedback to predict the states of the object in pushing, and robot proprioceptive feedback to perceive the states of the robot end-effector. In this regard, we propose to use historic observations of robot motion and F/T information to capture toss dynamics (see Section III-A).

Predicting Trajectory of Fast-Flying Objects: It has been long recognized as a fundamental problem in robotics, which for example allows robots to catch [4], [63], [64], bat [65], [66], and regrasp objects [6], [67] dynamically. Theoretically, rigid body dynamics can be applied to compute the object trajectory, e.g., using recursive Newton–Euler equations [68], [69], [70]. However, these closed-form solutions usually require

accurate knowledge of physical properties (e.g., mass, moment of inertia, position of CoM, etc.) and initial movement state (e.g., initial position, velocity, etc.) of objects that are difficult to estimate. We compare our method with analytical physics in Section IV-B. Many previous systems devoted to interacting with fast-moving objects assume a known model of object motion dynamics, e.g., simple ballistic and parabolic motion, and fit and estimate unknown model parameters from a small number of throwing demonstrations [23], [24], [25]. This line of methods has achieved satisfactory prediction accuracy, which, however, is mostly limited to a narrow range of simple objects like balls, and focused solely on their translational trajectories. Data-driven learning of motion dynamics, on the other hand, alleviates the dependency on predefined analytical models and prior knowledge of objects. For example, the seminal works by Kim et al. [26], [63] learnt a second-order dynamical system with regression techniques like support vector regression and Gaussian mixture to model dynamics of uneven free-flying objects, such as bottle and hammer. We compare our method with this line of methods, particularly Gaussian process regression (GPR) [71] in Section IV-B. More recently, deep neural networks (DNN), e.g., LSTM and CNN have been extensively applied to learn nonlinear dynamics of moving objects with high accuracy [72]. In this paradigm, a model of object dynamics is first trained offline from demonstrations in a supervised manner, which then tracks and predicts the motion of arbitrary objects recursively from online visual observations [73], [74], [75].

Our work also uses deep learning techniques for predicting flying trajectories of random unknown objects. However, rather than building a direct model of object dynamics from visual observations, we learn a forward mapping from toss dynamics to object dynamics. In addition to higher prediction accuracy, building such a mapping brings multiple benefits. First, our method predicts the complete flying trajectory of an arbitrarily tossed object all at once and immediately after the object is released. This feature makes our method well-suited for a range of time-sensitive applications, such as robot catching of fast-flying objects and blind juggling [39], [63]. Second, as noted previously, most previous prediction methods benefit from direct accurate observations of object states in flight, which on the other hand, necessitates the use of exteroceptive motion capture sensors, e.g., high-speed stereo cameras [76]. This limits the applications of these methods to wide empty spaces, where unimpaired and sufficient observations can be ensured [27], [77], [78]. In contrast, our method does not require prior knowledge or any observations of object dynamics, and thus can be more reliable, lightweight, and robust.

III. Proposed Method

We formulate the task of predicting the flying trajectory of an arbitrary unknown rigid object tossed by the robot (see Fig. 4), as to learn a joint function $\phi^{\theta} : \mathcal{X} \rightarrow \mathcal{Y}$, which takes a collection of multimodal robot proprioceptive perceptions $\mathcal{X}$ during tossing as inputs to model the toss dynamics, and then decodes the resulting object flying trajectory $\mathcal{Y}$ starting from the moment $(t = 0)$ that the robot releases the object.

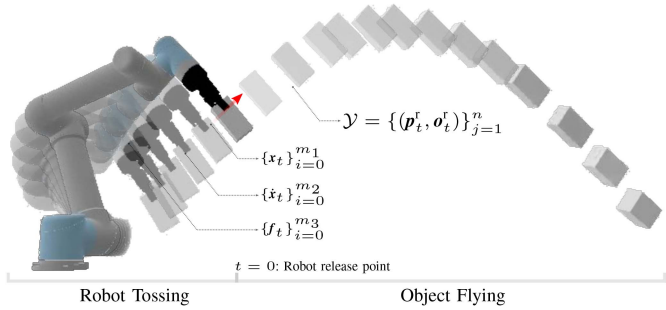


Fig. 4. Problem definition. Our method takes a collection of multimodal robot proprioceptive observations $\mathcal{X} = \{\{\boldsymbol{x}_t\}_{i=0}^{m_1}, \{\dot{\boldsymbol{x}}_t\}_{i=0}^{m_2}, \{\boldsymbol{f}_t\}_{i=0}^{m_3}\}$ perceived at the robot tossing stage as inputs to model the toss dynamics, and then outputs the resulting object flying trajectory $\mathcal{Y} = \{(\boldsymbol{p}_t^{\mathrm{r}}, \boldsymbol{o}_t^{\mathrm{r}})\}_{j=1}^{n}$ starting from the moment $(t = 0)$ that the robot releases the object.

A. Toss Dynamics Model With Multimodal Proprioceptive Observations

Our method first builds a model of toss dynamics from the robot's multimodal proprioceptive perceptions during tossing as follows:

$$\boldsymbol{e} = \phi_{\mathrm{toss}}(\mathcal{X}) \tag{1}$$

where the proprioceptive observations $\mathcal{X}$ record the historical spatio-temporal variations of the robot tossing motions and 6-D F/T values perceived at the *robot tossing* stage. The embedding vector $\boldsymbol{e}$ represents the learned toss dynamics characterized mainly by the robot tossing movements, the physical proprieties of the tossed object, and the robot–object interactions. We detail the role of each proprioceptive modality below in measuring the toss dynamics.

Consider, a robot tosses a random unknown object (see Fig. 4). The history of the robot's toss movements, such as the temporal variations of the robot's position and velocity, directly reveals the motion characteristics of the toss action. Meanwhile, from the perspective of the object's flying dynamics, since the object is kinematically connected to the robot before the robot releases its gripper, the robot's toss movement largely decides the initial flying state of the tossed object. For example, while ignoring the effects of the robot grasp and physical properties of the object (e.g., the location of the object CoM), the initial position and velocity of the object at the release point $(t = 0)$ are roughly equal to those of the robot end-effector.

In this regard, our method first extracts the motion characteristics of the toss action from the historical observations of the robot's position and velocity at the robot tossing stage. Our method takes two temporal sequences, $\{\boldsymbol{x}_t\}_{i=0}^{m_1}$ and $\{\dot{\boldsymbol{x}}_t\}_{i=0}^{m_2}$, $t = (i - m_{1/2})\mathrm{T}_{1/2}$, where $\boldsymbol{x}_t \in \mathrm{SE}(3)$ denotes the position and orientation, and $\dot{\boldsymbol{x}}_t \in \mathbb{R}^6$ denotes the translational and angular velocity of the robot end-effector at the time t. m_* and T_* denote the number of sampling points for perception and the temporal interval between every two neighboring points, respectively. $m_*\mathrm{T}_*$ thus decides the observation duration of the corresponding sensory modality. Note that the robot motion can also be encoded by the movements of arm joints, which is related to that of the robot end-effector by forward kinematics. However,

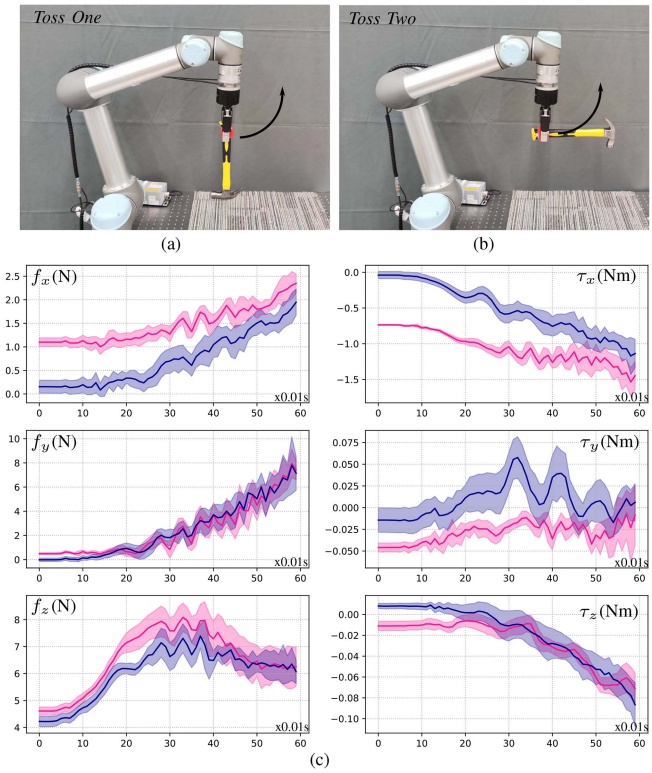


Fig. 5. F/T variations of different robot toss actions exhibit distinct spatio-temporal profiles. Spatio-temporal F/T variations can quantitatively perceive the object properties and interaction characteristics of different robot toss actions, e.g., the robot grasp and movements. (a) and (b) Two toss actions which differ only in the grasp locations on the object. Arrowed arcs indicate the moving direction and path of the robot end-effector during tossing. (c) F/T variations of the above two toss actions in ten runs: Blue curves represent the F/T variations of *Toss One* and the pink curves represent those of *Toss Two*. Shaded region represents the deviation, and the solid curve represents the mean of ten runs of each toss action.

from a perspective of learning efficiency, we use the temporal observation of the end-effector kinematics directly.

In addition to the motion characteristics, our method leverages the F/T perceptions to capture and compensate for the variations of toss dynamics inherited from the physical properties of the tossed object and the robot–object interactions, particularly the robot grasps. We employ a wrist-mounted 6-D F/T sensor between the robot arm and the end-effector (see Fig. 4) to capture the F/T variations at the robot tossing stage. The F/T variations of different robot toss actions exhibit distinct spatio-temporal profiles. Taking Fig. 5 for example, different robot grasps on a same object lead to quite different F/T variations.

Our method takes the temporal sequence $\{\boldsymbol{f}_t\}_{i=0}^{m_3}$, where $\boldsymbol{f}_t \in \mathbb{R}^6$ denotes the F/T value at the time t and $t = (i - m_3)\mathrm{T}_3$. Note the F/T observations can also perceive the robot motion characteristics, as the displacement and acceleration of the robot end-effector also affect the value of the connected F/T sensor. In this regard, the robot motion and F/T observations are spatio-temporally correlated, while such correlations can potentially facilitate the modeling of robot tossing. We present experimental results that further evaluate the effectiveness of each above sensory modality and their correlations on modeling the robot toss dynamics in Section IV-C.

Leveraging F/T for dynamic manipulation perception is promising. First, F/T perception has been widely recognized as an efficient, lightweight, and reliable modality to perceive interaction dynamics [79]. In addition, the F/T sensors, as standardized sensory tools, are accessible on most robot systems, allowing our methods to be easily transferred to different robots and other dynamic manipulation scenarios, e.g., robot pushing [80]. We provide experimental results that show our methods can be successfully applied on different robot systems for challenging tossing-centric applications in Section IV-D.

B. Object Flying Trajectory Parameterization and Prediction

Our method then learns another function that directly maps the learned feature of toss dynamics $\boldsymbol{e}$ to the flying trajectory $\mathcal{Y}$ of the tossed object as follows:

$$\mathcal{Y} = \phi_{\text{obj}}(\boldsymbol{e}) \tag{2}$$

which in turn depends on how the object trajectory $\mathcal{Y}$ is parameterized to correctly formulate the above mapping, particularly when the toss dynamics $\boldsymbol{e}$ is captured from only the robot's proprioceptive perception.

Trajectory Parameterization: Our method uses a *mock* body frame attached to the object at each toss action to parameterize the object's flying trajectories in proprioceptive robot tossing. Conventionally, the complete pose of a rigid object can be parameterized as the translation and rotation of a predefined body frame w.r.t. a reference frame. However, it can happen that different object trajectories can be mapped to the same proprioceptive robot features or vice versa, if they are parameterized with random body frames. It gives rise to the *learning inconsistency* between the robot toss dynamics learned from proprioceptive observations and the parameterized object's flying trajectories.

There are two reasons accounting for such inconsistency in proprioceptive robot tossing. On the one hand, the body frames (i.e., their origins and orientations) of different objects can be assigned w.r.t. different reference conventions. It, therefore, leads to inconsistent parameterization of the object trajectories. Take the example in Fig. 6(a), consider two different objects with *identical* physical properties of importance (e.g., mass, CoM, and moment of inertia). If they are applied under a same robot toss action (i.e., same robot grasp and toss movements), their flying motions under the action are supposed to be identical. Meanwhile, the perceived robot motions and F/T variations for such two actions are also identical, giving correctly a same toss dynamics learned by (1). However, if their body frames (dashed frames) are assigned differently, their parameterized trajectories (dashed curves) will be different, leading to the so-called inconsistency.

On the other hand, even if the object body frames are assigned w.r.t. a same reference convention, there can still exist inconsistencies between the learned toss dynamics and the object trajectories. For example in Fig. 6(b), consider a symmetric and uniformly-distributed object, if the robot tosses the object with the same toss movements but grasps it from different sides (indicated by the letter "A"), the proprioceptive observations (and thus the learned toss dynamics) for such two robot toss actions will

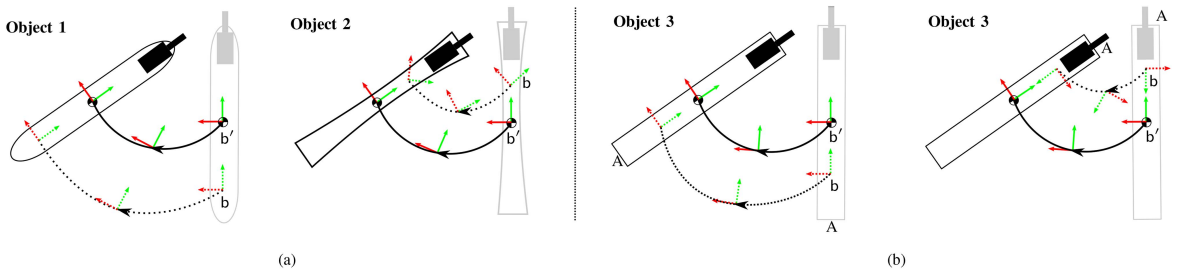


Fig. 6. Learning inconsistency in proprioceptive robot tossing. Parameterizing the object trajectory with the conventional body frame (the dashed one) can give rise to *inconsistency* between the robot toss dynamics learned from the proprioceptive observations and the parameterized object flying trajectories. The solid and dashed curves represent the same object trajectory but are parameterized with the mock body frame and a conventional body frame, respectively. (a) For two different objects with identical physical properties of importance (e.g., mass, CoM, and moment of inertia), if they are assigned with different body frames (dashed frames), a same toss action (and thus the same learned toss dynamics) will be mapped to differently parameterized object trajectories (dashed curves). (b) For a symmetric and uniformly-distributed object, if the robot tosses the object with the same toss movements but grasps it from different sides (indicated by the letter "A"), the proprioceptive observations for such two toss actions will be identical. However, the object trajectories (dashed curves) parameterized with the conventional body frame are different. For both cases, the object trajectories parameterized with the mock body frames (solid curves) are consistent with the learned proprioceptive robot toss dynamics.

be identical. However, the obtained object trajectories (dashed curves) parameterized with the conventional body frame (dashed frame) are different, resulting in the inconsistency as well. In fact, the robot toss dynamics learned from proprioceptive observations determines the relative trajectory of the tossed object, i.e., the object's spatial displacements from its initial pose at the robot releasing point, rather than its absolute trajectory defined w.r.t. the body frame. Therefore, to correctly formulate the mapping (2) for proprioceptive robot tossing, the initial object pose ($t = 0$) should be removed from the parameterized flying trajectory.

To address the consistency issue, i.e., to make the model in (2) learnable, we assume for each tossed object a *mock* body frame b' located at the object's CoM. The object position $\boldsymbol{p}_t^{\mathsf{r}} \in \mathbb{R}^3$ at time t can then be parameterized as the translational displacement of the mock body frame w.r.t. the robot frame r. Note, however, our model is agnostic to the object CoM (and all other physical properties), but can infer it accurately from the learned robot toss dynamics. Assigning the mock body frame b' to the object CoM provides two major benefits. First, the object CoM is unique and its motion pattern is consistent among different objects. Second, the motion dynamics of the object CoM is relatively simple and, therefore, more friendly to the learning methods.

Simply assigning the object body frame to its CoM, however, can still be confronted with the learning inconsistency. For example, even if the dashed body frame in Fig. 6(b) is located at the exact location of the object CoM, the parameterized object trajectories will still be different in the object orientation. To further address the orientation consistency, as shown in Figs. 7 and 8, for each arbitrary object and at the beginning of each toss action of the object, we assign the mock body frame b' (solid frame) with a fixed initial toss orientation $\mathbf{R}_{\mathsf{b}'}^{\mathsf{g}} \in \mathrm{SO}(3)$ w.r.t. the robot gripper.[1] This is to eliminate the inconsistency in the object orientation due to the body frame and initial object pose. In this way, the object trajectories parameterized with the mock body frame b' are consistent with the learned proprioceptive robot toss

[1]Initial orientation in practice is defined w.r.t. the frame of F/T sensor, which is kinematically connected to the robot gripper.

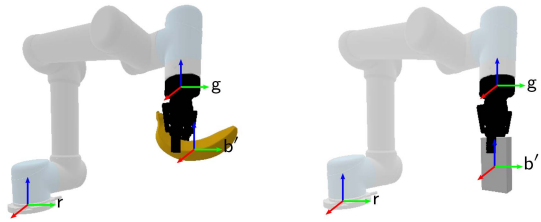


Fig. 7. Consistent trajectory parameterization in proprioceptive robot tossing. We assign a *mock* body frame b' for each toss action to eliminate the learning inconsistency in proprioceptive robot tossing. The mock body frame b' is located at the object CoM, and with a constant initial toss orientation $\mathbf{R}_{\mathsf{b}'}^{\mathsf{g}}$ w.r.t. the robot gripper g for all toss actions of arbitrary objects.

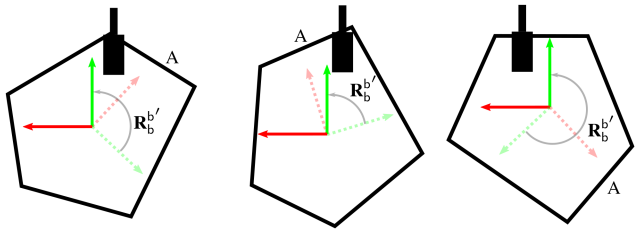


Fig. 8. Consistent trajectory parameterization with a mock body frame. Robot can grasp an object with different grasp poses w.r.t. the gripper, whereas the mock body frame (the solid one) b' is always aligned identically w.r.t. the robot gripper. If there exists a predefined body frame (the dashed one) b, the object trajectory w.r.t. the body frame can then be obtained with a constant transformation matrix $\mathbf{T}_{\mathsf{b}}^{\mathsf{b}'}$.

dynamics. For example, the object trajectories parameterized with b' (solid curves) in Fig. 6 are identical w.r.t. the same proprioceptive features. Note that the initial orientation of the frame b' can be randomly selected but should be consistent across different toss actions and tossed objects w.r.t. the robot gripper. Here for simplicity, we set the initial orientation to be identical to that of the robot gripper, i.e., $\mathbf{R}_{\mathsf{b}'}^{\mathsf{g}} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Then, the object pose $\boldsymbol{o}_t^{\mathsf{r}}$ (more specifically, the relative displacement of the object orientation) can be defined as the rotational displacement of the frame b' w.r.t. the robot frame r. In particular, we parameterize $\boldsymbol{o}_t^{\mathsf{r}}$ in the form of angle-axis, i.e.,

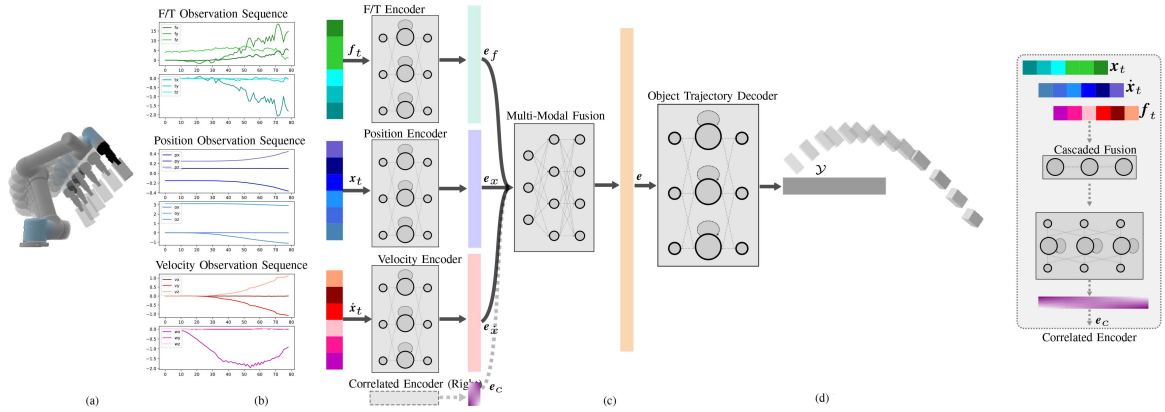


Fig. 9. Model architecture. We employ an encoder-decoder architecture to learn (1) and (2) jointly. Encoder processes the temporal sequence of robot proprioceptive observations $\mathcal{X}$ to obtain an embedding of toss dynamics $\boldsymbol{e}$, from which the decoder generates a prediction of the object trajectory $\mathcal{Y}$. We use a fully connected layer (referred to as the *multimodal fusion*) to fuse multiple single-modal toss features $\{\boldsymbol{e}_x, \boldsymbol{e}_{\dot{x}}, \boldsymbol{e}_f, \boldsymbol{e}_c\}$. We also employ a separate correlated encoder with a fusion layer (referred to as the *cascaded fusion*) to process synchronized multimodal proprioceptive observations at each timestep.

$\boldsymbol{o}_t^{\mathrm{r}} \in \mathbb{R}^3$ to eliminate the value dependency in parameterization. Note if there exists a prescribed body frame b, the object pose parameterized w.r.t. b can be easily transformed by multiplying the rotation matrix $\mathbf{R}_{\mathrm{b}}^{\mathrm{b}'}$.

In this context, we discretize and parameterize the object trajectory as $\mathcal{Y} = \{(\boldsymbol{p}_t^{\mathrm{r}}, \boldsymbol{o}_t^{\mathrm{r}})\}_{j=1}^{n}$ and $t = j\mathrm{T}$, where n denotes the number of the trajectory waypoints and T denotes the temporal interval between every two neighboring waypoints. $n\mathrm{T}$ denotes the trajectory duration. Different from visual prediction or other physics-based methods, our method leverages no explicit observation or modeling of the object dynamics. Rather, it decodes the entire object trajectory from the previously learned toss dynamics. This is beneficial: Our method eliminates the prediction dependency on the prior (visual) observations of object motions; it accurately predicts the flying trajectories of robot-tossed objects in real-time; and more importantly, it does not rely on the physical model of the tossed objects, but only the aforementioned proprioceptive perceptions of robot tossing, making the prediction more simple but effective. We present experimental analysis to evaluate the performance of our method on various seen and unseen objects in Section IV.

C. Model Architecture

We use an encoder-decoder architecture (see Fig. 9) to learn the functions defined by (1) and (2) jointly. The encoder processes the temporal sequence of robot proprioceptive observations $\mathcal{X}$ to obtain an embedding of the toss dynamics $\boldsymbol{e}$, from which the decoder yields a prediction of object flying trajectory $\mathcal{Y}$.

We propose two model architectures for the encoder to accommodate the synchronization scenarios that can happen during real tossing-centric robot applications. In most cases, since different sensory modalities can hardly be sampled synchronously due to the varied sampling frequency and transmit latency, the temporal perceptions of different modalities cannot be synchronized or aligned. In these cases, the encoder processes each sensory modality separately to extract their corresponding single-modal toss features $\{\boldsymbol{e}_x, \boldsymbol{e}_{\dot{x}}, \boldsymbol{e}_f\}$, respectively, and then fuses them together to obtain the toss embedding $\boldsymbol{e}$. In some other cases, the temporal observations of different sensory modalities can be obtained synchronously. As aforementioned, these observations of different modalities can be correlated in both temporal and spatial domains. For example, during robot tossing, the robot motion and F/T observations are correlated at each waypoint. Such correlations can contain additional features or clues that facilitate the learning of toss dynamics. In this regard, the encoder also fuses directly the synchronized observations of different sensory modalities at each waypoint to extract their correlation feature $\boldsymbol{e}_c$ with a separate module. The correlation feature $\boldsymbol{e}_c$ is then added and fused together with the single-modal features $\{\boldsymbol{e}_x, \boldsymbol{e}_{\dot{x}}, \boldsymbol{e}_f, \boldsymbol{e}_c\}$ to obtain the toss dynamics $\boldsymbol{e}$. We present experimental results to evaluate the effectiveness of each sensory modality and their correlations in modeling the toss dynamics in Section IV-C.

As depicted in Fig. 9, according to the sequential nature of the input proprioceptive perceptions and the output discretized object trajectory, we apply LSTM module [29], [30], [31], a class of recurrent neural networks for sequence processing, to construct both encoder and decoder. Each LSTM module consists of multiple layers of connected LSTM cells. As shown in Fig. 10, the compact forms of the equations for the forward pass of an LSTM cell are as follows:

$$\begin{cases} \boldsymbol{u}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{h}_{t-1}, \boldsymbol{a}_t] + \boldsymbol{b}_f) \\ \boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{h}_{t-1}, \boldsymbol{a}_t] + \boldsymbol{b}_i) \\ \boldsymbol{g}_t = \tanh(\boldsymbol{W}_c[\boldsymbol{h}_{t-1}, \boldsymbol{a}_t] + \boldsymbol{b}_c) \\ \boldsymbol{c}_t = \boldsymbol{u}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \\ \boldsymbol{y}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{h}_{t-1}, \boldsymbol{a}_t] + \boldsymbol{b}_o) \\ \boldsymbol{h}_t = \boldsymbol{y}_t \odot \tanh(\boldsymbol{c}_t) \end{cases} \tag{3}$$

where $\boldsymbol{W}_f$, $\boldsymbol{W}_i$, $\boldsymbol{W}_c$, and $\boldsymbol{W}_o$ are the coefficient matrices, and $\boldsymbol{b}_f$, $\boldsymbol{b}_i$, $\boldsymbol{b}_c$, and $\boldsymbol{b}_o$ are the bias vectors. These parameters are learned during training. $\boldsymbol{a}_t$, $\boldsymbol{h}_t$, and $\boldsymbol{c}_t$ denote, respectively, the input state, hidden state, and cell state at time t. $\boldsymbol{u}_t$, $\boldsymbol{i}_t$, $\boldsymbol{g}_t$, and $\boldsymbol{y}_t$ denote the input gate, forget gate, cell gate, and output gate at

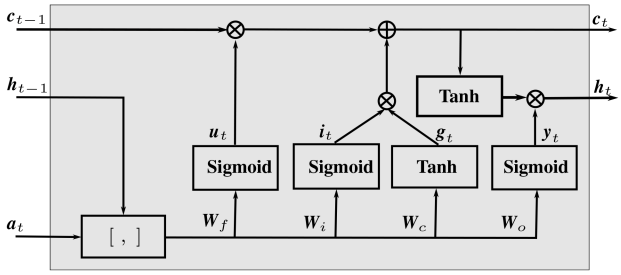


Fig. 10. Structure of an LSTM cell. Sigmoid and Tanh denote the sigmoid and hyperbolic tangent activation functions, respectively. [,] denotes the concatenation operation.

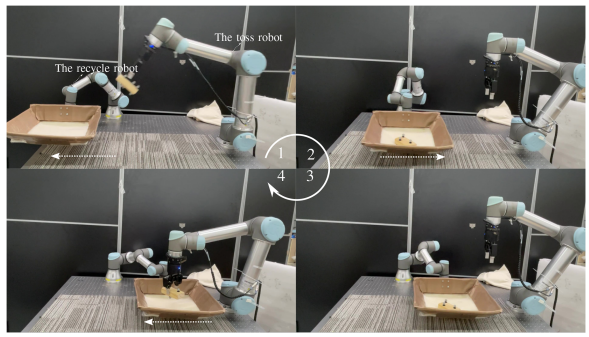


Fig. 11. Autonomous data collection for training. We build a dual-robot system for data collection that can reset itself with self-supervision. The recycling robot (left) collects and provides the object with a shallow box, and the toss robot picks the object and tosses it out with random toss actions. The robot motion and F/T values at the tossing stage are recorded with onboard sensors (ATI F/T mini45) as the input features, whereas the object trajectories are recorded with OptiTrack as the ground truth labels.

time t, respectively. $\sigma(\cdot)$ denotes the sigmoid activation function, and $\odot$ denotes the Hadamard product.

The encoder contains four LSTM modules to process three temporal sequences of single-modal observations and one sequence of synchronized multimodal observations, respectively. We apply a simple cascaded layer (referred to as the cascaded fusion) to concatenate the synchronized observations of different modalities, and a fully connected layer (referred to as the multimodal fusion) to fuse multiple single- and cross-modal toss features. The decoder contains one single LSTM module to decode the sequence of object trajectory in an autoregressive manner. We name our model for the asynchronous toss scenarios as TossNet, and the model for the **syn**chronous scenarios as syn-TossNet. We compare the proposed methods with several other state-of-the-art methods in Section IV-B.

D. Training and Implementation

1) Loss Function: The network is trained with a L2 loss function between the ground truth and the predicted object trajectories, which is written as follows:

$$\mathcal{L} = w_{\mathrm{p}} \left(\sum_{j=0}^{n} ||\boldsymbol{p}_j - \hat{\boldsymbol{p}}_j||^2 \right) /n + w_{\mathrm{o}} \left(\sum_{j=0}^{n} ||\boldsymbol{o}_j - \hat{\boldsymbol{o}}_j||^2 \right) /n \tag{4}$$

where $\boldsymbol{p}_j$ and $\hat{\boldsymbol{p}}_j$ denote the ground truth and predicted object position at jth waypoint, respectively. $\boldsymbol{o}_j$ and $\hat{\boldsymbol{o}}_j$ denote the ground truth and predicted object orientation at jth waypoint, respectively. w_{p} and w_{o} are tunable weights for the position loss and orientation loss, respectively.

2) Data Collection: To efficiently collect diverse data to train our models, we build a dual-robot system for autonomous data collection that can reset itself with self-supervision (see Fig. 11). The data collection process is recorded and provided in the Supplementary Material. During each collection cycle, a *recycling robot* (left) first passes an object to the *toss robot* (right) with a broad shallow box. The toss robot picks up the object with a random firm grasp and moves to an initial toss pose. Then, it selects and executes a random toss action. In the meanwhile, the recycling robot returns to catch the object and then shakes the box gently so that the object can be relocated to the box center to facilitate the next collection cycle. The robot's proprioceptive perceptions from the initial toss point to the gripper release point are recorded with the onboard sensors (robot position and velocity and ATI F/T mini45) as input features, and the object flying trajectory from the gripper release point to the object landing point is recorded with an optical motion capture system (OptiTrack) as the ground truth. In addition, we also developed a similar simulation environment with PyBullet [81], which acts as a consistent, controlled, and easy-regulated environment for fair and large-scale model evaluation. The autonomous data collection process and simulation environment are demonstrated in the video of the Supplementary Material.[2]

3) Implementation: During implementation, as shown in Algorithm 1, the recurrent nature of LSTM modules enables our proposed TossNet to model the toss dynamics in parallel with the robot tossing process, by processing the updated sensory observations at runtime. This, in fact, eliminates the computational time of task perception and modeling from the conventional *perception-model*-prediction pipeline. Further, different from the vision-based prediction, which has to observe the object trajectory and predict its follow-ups recursively, once the robot releases its gripper, our method can output the entire object trajectory without the need to further observe the object. It, therefore, further reduces the *prediction* time from the pipeline. This is particularly advantageous for time-critical toss applications, e.g., robot juggling [4], [9]. We provide a detailed analysis of the time complexity of our method in Section IV-C and a variety of real-world robot applications using our method in Section IV-D.

IV. Experiments

This section presents a series of experimental results in both simulated and real settings. We aim to evaluate 1) the accuracy and efficiency of our method (TossNet) to model the toss dynamics from robot proprioceptive observations and predict the object flying trajectories from the learned toss dynamics (see Section IV-B), 2) the effectiveness of each sensory modality and their correlations in modeling the toss dynamics (see

[2]Video of the real robot experiments is available online at https://youtu.be/yBz9debQyg

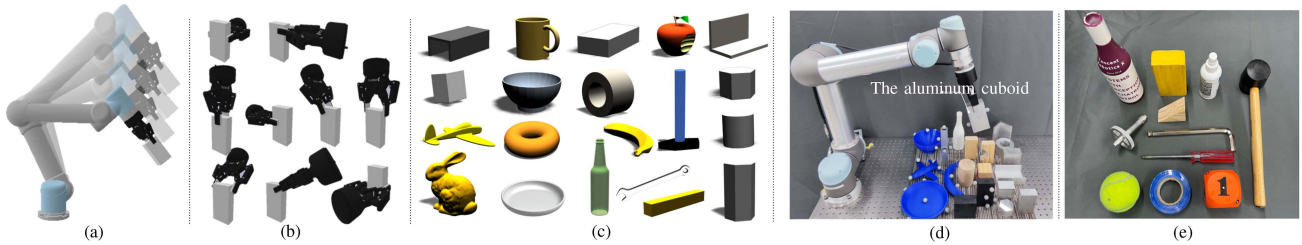


Fig. 12. Construction of datasets. We collect three datasets $\{\mathcal{M}_s, \mathcal{G}_s, \mathcal{O}_s\}$ in the simulated environment to systematically evaluate if our method can capture the variations of (a) robot motion, (b) grasp, and (c) physical proprieties of the tossed objects in modeling the toss dynamics. (d) We collect two real-world toss datasets $\{\mathcal{G}_r, \mathcal{O}_r\}$ corresponding to the varied robot grasps and twenty different objects to train and evaluate our method. (e) We also include a dataset of real *unseen* objects for the independent model evaluation. The real objects are varied in geometries such as cylinders, cubes, and polyhedrons, and materials including rubber, aluminium, wood, and ABS resin.

Algorithm 1: TossNet and *syn*-TossNet.

Input: Sensory observation $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$, $\boldsymbol{f}$
Output: The predicted object trajectory $\mathcal{Y}$
while robot tossing ($t \leq 0$) **do**
  if $\boldsymbol{x}_t/\dot{\boldsymbol{x}}_t/\boldsymbol{f}_t$ updated **then**
    $\boldsymbol{e}_{x/\dot{x}/f} \leftarrow \text{LSTM}_{x/\dot{x}/f}(\boldsymbol{x}_t/\dot{\boldsymbol{x}}_t/\boldsymbol{f}_t)$
    if synchronous **then**
      $\boldsymbol{e}_c \leftarrow \text{LSTM}_c(\boldsymbol{x}_t, \dot{\boldsymbol{x}}_t, \boldsymbol{f}_t)$
if synchronous **then**
  $\boldsymbol{e} \leftarrow \text{Fuse}(\boldsymbol{e}_x, \boldsymbol{e}_{\dot{x}}, \boldsymbol{e}_f, \boldsymbol{e}_c)$
else
  $\boldsymbol{e} \leftarrow \text{Fuse}(\boldsymbol{e}_x, \boldsymbol{e}_{\dot{x}}, \boldsymbol{e}_f)$
$\mathcal{Y} \leftarrow \text{LSTM}_{\text{decoder}}(\boldsymbol{e})$
return $\mathcal{Y}$

Section IV-C), and 3) the deployment performance of our method on different real robot systems and its potential usage on various tossing-centric robot applications (see Section IV-D). To this end, we build a variety of robot tossing datasets on both seen and unseen objects. We compare our method with several state-of-the-art baseline methods by evaluating their prediction accuracy of object trajectories on various tossing scenarios. We apply our method to challenging real-world tossing-centric robot applications such as blind juggling and precise pitching.

A. Experimental Settings and Datasets

We collect a range of tossing datasets in both simulated and real robot settings to train and evaluate our methods on both seen and unseen objects. The data collection process has been detailed in Section III-D.

Simulated Toss Datasets: We collect three simulated datasets $\{\mathcal{M}_s, \mathcal{G}_s, \mathcal{O}_s\}$ [see Fig. 12(a)–(c)] for a systematic and large-scale model evaluation.

1) The motion dataset $\mathcal{M}_s$ contains 10 K robot toss actions. In each trial, the robot tosses one single cuboid block with a same grasp but releases the block at different positions and velocities [see Fig. 12(a)]. This dataset is primarily used to evaluate the capability of our method to capture robot motions in modeling the toss dynamics.
2) The grasp dataset $\mathcal{G}_s$ contains 20 K toss actions where the robot tosses one single cuboid block with both randomized release positions, velocities, and robot grasps on the block [see Fig. 12(b)]. This is to further evaluate the model's capability to capture robot grasps (in addition to robot motions) in modeling the robot toss dynamics.
3) The object dataset $\mathcal{O}_s$ contains 20 K toss actions where the robot tosses twenty different objects [see Fig. 12(c)], with randomized release positions, velocities, and robot grasps. This is to further evaluate the method's capability to capture object properties in modeling the toss dynamics.

Real-World Toss Datasets: Similarly, we collect two real-world datasets, $\mathcal{G}_r$ and $\mathcal{O}_r$, corresponding to toss actions characterized with randomized robot grasps and further different objects (in addition to random robot motions). $\mathcal{G}_r$ contains 1 K robot toss actions of an aluminum cuboid and $\mathcal{O}_r$ contains 2 K toss actions of twenty different objects [see Fig. 12(d)].

Unseen Objects: We also include a dataset of *unseen* objects [see Fig. 12(e)], where each object does not appear in the above training datasets. We collect 400 random toss actions on 11 real objects. Each toss action is generated with randomized release position, velocity, and robot grasp. This dataset is built only for the independent model evaluation and never joins the training pipeline.

For each involved object, its physical properties of importance are calculated automatically with SolidWorks and its corresponding robot grasps are sampled randomly from an offline grasp dataset generated with GraspIt [82]. Note these object information is just for data generation but agnostic to our method. We particularly include a set of nonhomogeneous objects in the simulation datasets for an unbiased evaluation. Most real objects are characterized by symmetrical geometries, such as cylinders, cubes, and polyhedrons, and are made of uniform materials including rubber, aluminium, wood, and ABS resin. Note this is only for the simplicity of data generation, e.g., obtaining the ground truth of object physical properties and achieving stable robot grasps on these objects, while our method is in fact general for all kinds of rigid objects.

For both simulated and real-world scenarios, we use a UR5 arm mounted with a Robotiq 2F-85 gripper. To reduce the side effects of motion noises, in the real-world scenario, we set the robot control mode to be velocity control, so the robot motion

can be smoother at both position and velocity levels. We use the ATI F/T mini45 sensor to collect the F/T variations during robot tossing.

Training and Metrics: Each dataset is split into 70% for training, 20% for validation, and 10% for testing. Furthermore, to make full use of the datasets and for the unbiased model evaluation, k-folder ($k = 5$) cross-validation is applied to training. The training and testing procedure is applied to all baseline methods for a fair comparison.

We evaluate the trajectory prediction accuracy on the object position and orientation separately with two different metrics as follows:

1) Average displacement error (ADE) [83]: ADE refers to the mean square error between all waypoints on the predicted trajectories and their corresponding ground truth values. It evaluates the overall prediction accuracy with the average error distribution along the predicted trajectories. Note that here, we report the *rooted* mean square error as ADE for easier understanding.
2) Final displacement error (FDE) [31]: FDE refers to the mean of absolute displacements between the predicted final waypoints and their corresponding ground truth values. Compared to ADE, it further illustrates how the prediction error evolves along the predicted trajectories, and is typically larger than ADE. Indeed, minimizing FDE is as important as minimizing ADE in trajectory prediction, as most methods can suffer from error propagation. From a practical perspective, minimizing FDE in trajectory prediction is even more crucial, as it is often followed by a catch action (e.g., robot juggling) at a waypoint on the latter trajectory segment.

B. Baseline Studies

We first compare our method against several baseline methods in modeling the robot toss dynamics, by evaluating their performance of trajectory prediction on the above toss datasets.

1) Analytical: It computes the object flying trajectory autoregressively based on the equations of rigid body dynamics, i.e., the Newton–Euler equations as follows:

$$\begin{cases} \boldsymbol{F} = M\dot{\boldsymbol{v}} \\ \boldsymbol{\tau} = \boldsymbol{\mathcal{I}}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{\mathcal{I}}\boldsymbol{\omega} \end{cases} \tag{5}$$

where $\boldsymbol{F}$, $\boldsymbol{\tau}$ denote the total force and torque acting on the object's CoM, respectively. M and $\boldsymbol{\mathcal{I}}$ denote the object's mass and the moment of inertia, respectively, about the object's CoM. $\boldsymbol{v}$ and $\boldsymbol{\omega}$ denote the object's translational and angular velocities, respectively. Specifically, the ground truth values of the object's physical properties, including its mass, CoM, and moment of inertia are measured and used in the evaluation, which, however, are agnostic to other baselines and our method. In addition, the initial motion state of the tossed object, i.e., the object position and velocity at the robot release point, is also required and approximated using the transformed object pose w.r.t. that of the robot end-effector, assuming the robot grasp on the object remains fixed during tossing. Note one may argue that it is unfair for the analytical method to use the approximated initial *object* state rather than its true value. As claimed in previous sections, the object state in highly dynamic robot manipulations can hardly be captured due to the complex interactions (e.g., grasp slip) and sensor limitations. In contrast, the *robot* state can be easily obtained with the onboard proprioceptive encoders. In fact, it is indeed unfair for other baseline methods, since they can leverage no prior knowledge of the object but only the proprioceptive observations of robot tossing.
2) MLP: It regresses the object trajectory directly from the approximated initial motion state of the tossed object at release with a multilayer perceptron (MLP). We use a four-layered MLP and each hidden layer contains 128 hidden neurons.
3) e-MLP: It decodes the object trajectory with a similar MLP as above but from a vector of task embedding $\boldsymbol{e}$ [defined by (1)] learned by our method (TossNet), rather than from the initial motion state of the tossed object. This is to evaluate the capability of our method to accurately model the robot toss dynamics from proprioceptive observations, and the effects of learned toss dynamics on improving the prediction accuracy of object trajectories.
4) e-GPR: It also takes advantage of the task embedding $\boldsymbol{e}$ learned by our method (TossNet), but based on Gaussian processes (GP) [71], [84], [85] to regress the object trajectory. We build a separate GP for each dimension of the object trajectory, assuming that their values change independently [71].
5) AttenS2S: It represents the pipeline of **atten**tion-based DNN methods, e.g., Transformer [86], for S2S processing. Briefly, similar to our method, it encodes the toss dynamics from proprioceptive observations with a set of attention modules, and then decodes the object trajectory from the learned embeddings with another set of attention modules.
6) ConvS2S: It represents another pipeline of deep learning methods using temporal **conv**olutional operations for S2S processing [87].
7) TossNet (ours): It represents our method to jointly model robot toss dynamics and predict the object trajectories. Compared with AttenS2S and ConvS2S, our method is built with LSTM modules as detailed in Section III.
8) Analy-TossNet: It combines the pure data-driven TossNet with analytical physics. It differs from TossNet by predicting the object trajectory as residuals from an initial approximation of the object trajectory based on physics, as detailed in the analytical method (Analytical).

The ADE and FDE results are summarized in Tables I and II and Tables III and IV, respectively, where position errors are in meters, orientation errors are in radians, and standard deviations are in parentheses. Overall, the four DNN-based methods (last four rows) achieve higher prediction accuracy than other lines of methods, by at least one order of magnitude in most cases. Further, our proposed method (TossNet) excels among the four deep learning methods for both seen and unseen objects in predicting both object position and orientation.

TABLE I
BASELINE RESULTS IN SIMULATION

Method	$\mathcal{M}_s$		$\mathcal{G}_s$		$\mathcal{O}_s$	
	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)
Analytical	4.23e-2(3.80e-3)	1.42e+0(1.27e-2)	3.34e-2(1.37e-2)	1.43e+0(3.29e-2)	3.50e-2(2.34e-2)	1.42e+0(2.87e-2)
MLP	3.32e-1(5.96e-2)	3.20e-1(5.80e-2)	3.43e-1(5.80e-2)	3.42e-1(6.08e-2)	3.44e-1(6.16e-2)	3.37e-1(6.44e-2)
e-MLP	1.19e-1(3.04e-2)	1.19e-1(1.94e-2)	1.54e-1(3.04e-2)	1.23e-1(2.90e-2)	1.64e-1(2.58e-2)	1.59e-1(2.31e-2)
e-GPR	5.58e-2(2.26e-2)	1.45e-2(2.68e-2)	5.58e-2(2.34e-2)	1.36e-2(2.74e-2)	5.56e-2(2.21e-2)	1.61e-2(2.62e-2)
ConvS2S	8.10e-3(4.50e-3)	1.27e-2(1.48e-2)	8.40e-3(5.50e-3)	2.23e-2(4.28e-2)	1.27e-2(6.10e-3)	2.42e-2(3.42e-2)
AttenS2S	4.50e-3(2.40e-3)	6.90e-3(8.80e-3)	5.50e-3(2.80e-3)	1.07e-2(9.50e-3)	8.50e-3(4.20e-3)	9.91e-3(3.21e-3)
TossNet (ours)	3.30e-3(1.70e-3)	**4.70e-3(3.70e-3)**	**3.70e-3(1.90e-3)**	**6.50e-3(4.50e-3)**	**3.10e-3(1.90e-3)**	**6.80e-3(6.80e-3)**
Analy-TossNet	**3.20e-3(3.30e-3)**	4.70e-3(4.10e-3)	4.10e-3(2.10e-3)	6.90e-3(1.00e-3)	4.10e-3(2.20e-3)	8.03e-3(5.10e-3)

The rooted ADE of baseline methods on Simulated datasets.

TABLE II
BASELINE RESULTS IN REAL WORLD

Method	$\mathcal{G}_r$		$\mathcal{O}_r$		Unseen Objects	
	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)
Analytical	1.01e-1(2.51e-2)	1.68e+0(2.49e-2)	1.12e-1(2.86e-2)	1.69e+0(2.06e-2)	1.23e-1(2.79e-2)	1.80e+0(1.99e-2)
MLP	1.98e-1(2.09e-2)	3.12e-1(3.15e-2)	1.93e-1(1.87e-2)	2.82e-1(2.11e-2)	2.21e-1(1.86e-2)	2.69e-1(1.87e-2)
e-MLP	1.04e-1(9.90e-3)	1.48e-1(1.66e-2)	9.19e-2(1.68e-2)	1.47e-1(1.71e-2)	9.94e-2(1.32e-2)	1.51e-1(1.40e-2)
e-GPR	8.47e-2(1.85e-2)	1.22e-1(6.41e-2)	9.03e-2(1.66e-2)	1.48e-1(5.06e-2)	8.34e-2(1.49e-2)	1.91e-1(4.35e-2)
ConvS2S	2.65e-2(7.50e-3)	6.59e-2(3.17e-2)	3.52e-2(7.47e-3)	6.46e-2(4.92e-2)	3.32e-2(4.95e-3)	5.61e-2(4.74e-2)
AttenS2S	**1.53e-2(6.29e-3)**	4.01e-2(1.86e-2)	2.07e-2(5.56e-3)	3.70e-2(1.84e-2)	2.11e-2(5.11e-3)	3.41e-2(1.63e-2)
TossNet (ours)	1.57e-2(6.94e-3)	3.84e-2(1.71e-2)	**1.91e-2(5.26e-3)**	**3.54e-2(1.78e-2)**	**1.81e-2(4.48e-3)**	3.23e-2(1.27e-2)
Analy-TossNet	1.59e-2(5.71e-3)	**3.82e-2(1.75e-2)**	2.22e-2(5.35e-3)	3.65e-2(1.63e-2)	1.99e-2(4.01e-3)	**2.92e-2(1.01e-2)**

Rooted ADE of baseline methods on Real-World datasets.

TABLE III
BASELINE RESULTS IN SIMULATION

Method	$\mathcal{M}_s$		$\mathcal{G}_s$		$\mathcal{O}_s$	
	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)
Analytical	4.32e-2(6.05e-3)	1.51e+0(1.81e-2)	4.10e-2(1.38e-2)	1.45e+0(6.93e-2)	3.62e-2(6.05e-3)	1.49e+0(1.81e-2)
MLP	3.41e-1(2.67e-1)	2.87e-1(1.99e-1)	3.51e-1(2.56e-1)	3.58e-1(2.07e-1)	3.51e-1(2.67e-1)	2.87e-1(1.99e-1)
e-MLP	1.37e-1(8.11e-2)	1.38e-1(6.70e-2)	1.63e-1(1.16e-1)	1.37e-1(7.20e-2)	1.67e-1(1.04e-1)	1.19e-1(9.48e-2)
e-GPR	7.16e-2(3.39e-2)	1.59e-2(3.43e-2)	6.77e-2(3.36e-2)	1.72e-2(4.08e-2)	6.93e-2(3.37e-2)	1.80e-2(3.17e-2)
ConvS2S	3.55e-2(2.05e-2)	2.00e-2(3.56e-2)	4.85e-2(2.65e-2)	5.15e-2(8.72e-2)	3.55e-2(2.05e-2)	2.00e-2(3.56e-2)
AttenS2S	8.30e-3(5.50e-3)	**1.05e-2(2.29e-2)**	8.95e-3(6.75e-3)	2.19e-2(5.25e-2)	8.30e-3(5.51e-3)	1.05e-2(2.29e-2)
TossNet (ours)	6.50e-3(4.95e-3)	1.08e-2(1.79e-2)	**6.30e-3(7.20e-3)**	**2.13e-2(1.49e-2)**	**6.30e-3(4.95e-3)**	**1.01e-2(1.07e-2)**
Analy-TossNet	**6.10e-3(4.88e-3)**	1.11e-2(1.71e-2)	6.30e-3(7.60e-3)	2.29e-2(5.44e-2)	6.70e-3(5.02e-3)	1.04e-2(1.80e-2)

FDE of all baseline methods on the Simulated datasets.

TABLE IV
BASELINE RESULTS IN REAL-WORLD

Method	$\mathcal{G}_r$		$\mathcal{O}_r$		Unseen Objects	
	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)
Analytical	1.37e-1(5.33e-2)	1.69e+0(7.31e-2)	1.62e-1(5.66e-2)	1.72e+0(5.36e-2)	2.01e-1(3.09e-2)	1.81e+0(5.35e-2)
MLP	2.50e-1(1.77e-1)	4.45e-1(2.10e-1)	2.49e-1(1.73e-2)	4.03e-1(2.06e-1)	2.66e-1(1.52e-2)	3.91e-1(2.21e-1)
e-MLP	1.32e-1(8.47e-2)	2.23e-1(1.01e-1)	1.26e-1(8.65e-2)	1.89e-1(1.08e-1)	1.34e-1(4.25e-2)	2.01e-1(9.16e-2)
e-GPR	7.30e-2(3.70e-2)	3.13e-1(1.63e-1)	9.61e-2(3.36e-2)	2.50e-1(1.27e-1)	9.04e-2(2.31e-2)	2.15e-1(9.13e-2)
ConvS2S	5.38e-2(2.76e-2)	1.57e-1(8.70e-2)	4.49e-2(2.53e-2)	1.62e-1(8.45e-2)	5.11e-2(1.89e-2)	1.14e-1(6.01e-2)
AttenS2S	3.20e-2(1.63e-2)	9.46e-2(4.89e-2)	2.69e-2(1.54e-2)	8.86e-2(4.80e-2)	**2.62e-2(1.43e-2)**	7.85e-2(3.21e-2)
TossNet (ours)	**3.15e-2(1.82e-2)**	**7.09e-2(4.68e-2)**	**2.62e-2(1.45e-2)**	8.40e-2(4.52e-2)	2.68e-2(1.33e-2)	**7.06e-2(3.59e-2)**
Analy-TossNet	3.16e-2(2.00e-2)	9.15e-2(4.82e-2)	2.65e-2(1.53e-2)	**8.25e-2(4.57e-2)**	2.65e-2(1.39e-2)	7.25e-2(3.57e-2)

FDE of all baseline methods on the Real-World datasets.

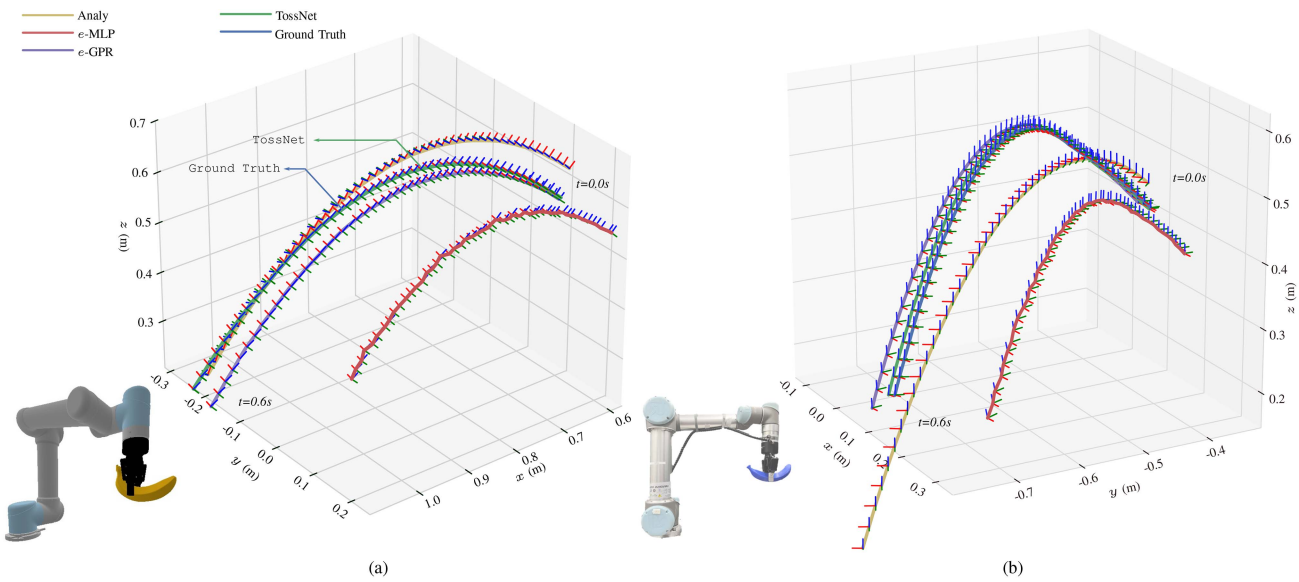


Fig. 13. Predicted flying trajectories of a robot tossed banana by five baseline methods in (a) Simulation. (b) Real World. Each trajectory starts from the robot release point ($t = 0$) and lasts for 0.6 s ($t = 0.6$). In both cases, the trajectory predicted by our method (the green line) aligns best with the ground truth (the blue line) trajectory. In addition, the prediction errors by our method accumulate at a very slow rate and remain in a small range. (a) Ground truth and four predicted trajectories for a simulated banana. (b) Ground truth and four predicted trajectories for a real 3-D-printed banana.

Specifically, taking solely the approximated initial motion state of the tossed object, MLP fails to model the toss dynamics. It thus performs the worst among all baseline methods in trajectory prediction. The analytical method performs slightly better than MLP, since it is built on a ground rule of physics and exposed to more inputs, i.e., the ground truth physical properties of the tossed objects. However, the analytical method assumes a deterministic system, which is rarely the case in dynamic manipulation (this will be further discussed later with Fig. 13), and, therefore, also shows limited prediction accuracy. By absorbing a task feature vector $\boldsymbol{e}$ produced by our method (TossNet), e-MLP achieves a much higher prediction accuracy of the object trajectories than pure MLP. It proves that our method can obtain an accurate estimation of the robot toss dynamics from proprioceptive observations. While with the same task embedding $\boldsymbol{e}$, e-GPR outperforms e-MLP in both simulation and real scenarios, indicating that GPR is more expressive than MLP in decoding object trajectories from the learned toss dynamics. However, it still falls short compared to DNN-based methods, particularly in terms of error deviation.

Among the four DNN-based methods, AttenS2S achieves a comparable prediction accuracy to our method. However, compared with TossNet, AttenS2S (attention-based) has a higher model complexity, which has been proved to easily lead to model overfitting especially while being trained using limited data [88]. On the other hand, attention-based methods (e.g., Transformer [86]) by principle require obtaining the entire input sequence in one shot for task modeling (i.e., to compute the so-called attention scores), which may pose further computational delay in real-world toss-centric applications, particularly when the input sequence is long and multimodal. In contrast, as previously mentioned in Section III, our method consumes the input sequence recurrently and, therefore, can be parallelized with robot tossing, i.e., it can perceive and model the robot toss action at runtime, which is advantageous for time-critical tossing applications.

When it comes to physics-based methods, it is widely proven that leveraging physics can potentially improve the performance of data-driven prediction methods, as the optimization can be possibly steered from an initial physics-based solution of already high quality. However, as the analytical method (first row in Tables I–IV) fails to provide a reasonably accurate approximation of the object trajectory (especially in orientation), the prediction accuracy of the physics-augmented TossNet (Analy-TossNet, last row in Tables I–IV) does not show apparent improvements compared with pure TossNet.

Fig. 13 plots the predicted flying trajectories for a simulated and a real 3-D-printed robot-tossed banana respectively, produced by our and four selected baseline methods. Each trajectory starts from the robot release point (t=0) and lasts for 0.6 s (t=0.6). Overall, for both objects, the trajectories predicted by our method (the green lines) align best with their ground truth trajectories (the blue lines and obtained with OptiTrack). In addition, it can be observed the prediction errors of all methods grow larger along the trajectories, mainly due to the issue of error propagation. However, the error by our method accumulates at a very slow rate and remains in a small range, which can also be observed by comparing ADE and FDE in Tables I (respectively II) and III (respectively IV). On the one hand, there are two major reasons accounting for this phenomenon. First, the object trajectory typically deviates more dramatically as the object flies, and therefore gets more difficult to fit. Meanwhile, the prediction error can propagate forward and accumulate along the predicted trajectory, which is the major drawback of autoregressive decoding. On the other hand, as also shown in Tables III and IV, our method still outperforms in minimizing the final perdition errors.

TABLE V
ABLATION RESULTS IN SIMULATION

Method	$\mathcal{M}$-s		$\mathcal{G}$-s		$\mathcal{O}$-s	
	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)
r-TossNet	7.20e-3(4.00e-3)	8.20e-3(8.20e-3)	2.13e-2(1.38e-2)	6.46e-2(7.74e-2)	1.13e-2(7.80e-3)	2.03e-2(4.12e-2)
ft-TossNet	5.20e-3(2.90e-3)	6.90e-3(5.90e-3)	1.02e-2(5.10e-3)	1.03e-2(9.10e-3)	9.60e-3(4.70e-3)	1.70e-2(1.57e-2)
TossNet (ours)	3.30e-3(1.70e-3)	4.70e-3(3.70e-3)	3.70e-3(1.90e-3)	6.50e-3(4.50e-3)	**3.10e-3(1.90e-3)**	6.80e-3(6.80e-3)
syn-TossNet (ours)	**3.00e-3(2.10e-3)**	**4.40e-3(3.60e-3)**	**2.30e-3(3.10e-3)**	**4.15e-3(4.06e-3)**	4.50e-3(4.40e-3)	**6.15e-3(4.17e-3)**

Rooted ADE of all ablation methods on Simulated datasets.

TABLE VI
ABLATION RESULTS IN REAL-WORLD

Method	$\mathcal{G}$-r		$\mathcal{O}$-r		Unseen Objects	
	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)
r-TossNet	4.67e-2(3.64e-2)	1.39e-1(7.10e-2)	5.59e-2(4.83e-2)	1.53e-1(6.03e-2)	4.71e-2(3.92e-2)	1.31e-1(5.88e-2)
ft-TossNet	3.29e-2(1.38e-2)	7.26e-2(3.34e-2)	3.58e-2(1.07e-2)	8.82e-2(2.84e-2)	3.33e-2(1.21e-2)	6.98e-2(3.16e-2)
TossNet (ours)	**1.57e-2(6.94e-3)**	3.84e-2(1.71e-2)	1.91e-2(5.26e-3)	3.54e-2(1.78e-2)	1.81e-2(4.48e-3)	3.23e-2(1.27e-2)
syn-TossNet (ours)	1.98e-2(4.70e-3)	**3.49e-2(2.23e-2)**	**1.28e-2(5.95e-3)**	**2.79e-2(1.93e-2)**	**1.14e-2(4.52e-3)**	**2.71e-2(1.37e-2)**

Rooted ADE of ablation methods on Real-World datasets.

In other words, in addition to the high-prediction accuracy, our method achieves a slower rate of error propagation, which is critical for long-horizon iterative trajectory prediction.

It also shows in Fig. 13 that the actual initial motion state of the tossed object (i.e., the starting point at the ground truth object trajectory) can differ from the approximated one predicted by the analytical method (i.e., the starting point at the trajectory predicted by the analytical method). It demonstrates that due to highly complex interaction dynamics, the relative transformation between the robot gripper and the tossed object can indeed change during tossing, leading to an indeterminate offset between the approximated initial object state and its true value. In fact, in addition to dynamic grasp displacements, there exist more dynamic uncertainties that are supposed to be considered (but can be hardly captured with extrinsic sensors as discussed before) to accurately model the toss dynamics.

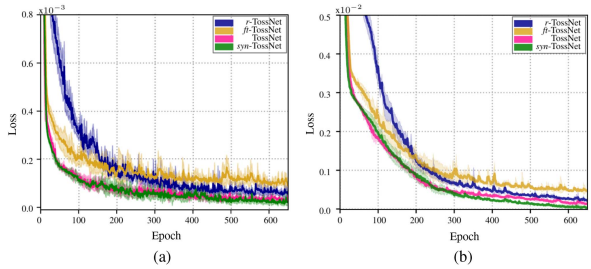


Fig. 14. Learning curves of four model variants along the learning epochs. Briefly, TossNet (pink) and syn-TossNet (green), which absorb the historical observations of both sensory modalities outperform r-TossNet (blue) and ft-TossNet (orange), which absorb only one single modality. syn-TossNet achieves a higher prediction accuracy than TossNet by absorbing additional multimodal correlated features. The solid line represents the mean error of fivefold runs, and the shaded region represents the standard deviation of the training runs at each epoch.

C. Ablation Studies

We also conduct a series of ablation studies to evaluate the effectiveness of each sensory modality in modeling the toss dynamics and predicting the resulting object trajectory. Briefly, we compare four model variants as follows.

1) r-TossNet: It leverages only the robot motion sequence at the robot tossing stage to encode the robot toss dynamics and decode the resulting object trajectory.
2) ft-TossNet: It leverages only the F/T reading sequence to encode the toss dynamics.
3) TossNet (ours): Both robot motion and F/T sequences are employed but processed separately to encode the toss dynamics. This applies to the scenarios where the robot motion and F/T perceptions during robot tossing can only be sampled asynchronously.
4) syn-TossNet (ours): In addition to separate processing, the method also fuses and processes robot motion and F/T perceptions together to model the toss dynamics. This applies to the scenarios where the robot motion and F/T perceptions can be sampled synchronously. It also acts as a baseline to study the spatio-temporal correlations of different sensory modalities, and their effects on learning the tossing dynamics, as discussed before in Section III-C.

The ADE and FDE results of the above models are summarized in Tables V–VIII. Position errors are presented in units of meters, and orientation errors are in radians. Standard deviations are in parentheses. Fig. 14 also shows their learning curves on the dataset $\mathcal{O}_s$ and $\mathcal{O}_r$, respectively. Overall, TossNet and syn-TossNet which absorb the historical observations of both sensory modalities outperform r-TossNet and ft-TossNet which absorb only one single modality, i.e., the robot motion and F/T variations, respectively. Comparing r-TossNet and ft-TossNet (rows 1 and 2 in Tables V–VIII), we see that while ft-TossNet achieves higher prediction accuracy, r-TossNet converges faster than ft-TossNet as shown by their learning curves in Fig. 14. TossNet shows both advantages of fast convergence and high-prediction accuracy.

The results demonstrate that the robot motions reflect more explicitly the motion characteristics of a robot toss action, and thus can more directly relate the robot toss dynamics and the

TABLE VII
ABLATION RESULTS IN SIMULATION

Method	$\mathcal{M}$-s		$\mathcal{G}$-s		$\mathcal{O}$-s	
	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)
r-TossNet	8.10e-3(4.05e-3)	2.21e-2(3.01e-2)	2.56e-2(1.33e-2)	6.95e-2(4.63e-2)	1.91e-2(1.88e-2)	4.99e-2(4.27e-2)
ft-TossNet	7.85e-3(3.75e-3)	1.53e-2(2.16e-2)	1.38e-2(9.60e-3)	2.28e-2(1.89e-2)	1.55e-2(1.01e-2)	3.10e-2(2.52e-2)
TossNet (ours)	6.50e-3(4.95e-3)	1.08e-2(1.79e-2)	6.30e-3(7.20e-3)	2.13e-2(1.49e-2)	6.30e-3(4.95e-3)	1.01e-2(1.07e-2)
syn-TossNet (ours)	**6.05e-3(4.20e-3)**	**9.75e-3(3.60e-3)**	**4.03e-3(8.50e-3)**	**1.55e-2(1.15e-2)**	**6.01e-3(4.73e-3)**	**9.34e-3(1.13e-2)**

FDE of all ablation methods on the Simulated datasets.

TABLE VIII
ABLATION RESULTS IN REAL WORLD

Method	$\mathcal{G}$-r		$\mathcal{O}$-r		Unseen Objects	
	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)
r-TossNet	6.14e-2(3.17e-2)	1.76e-1(9.54e-2)	5.85e-2(3.55e-2)	1.79e-1(9.41e-2)	5.85e-2(3.55e-2)	1.79e-1(9.41e-2)
ft-TossNet	5.70e-2(3.41e-2)	1.83e-1(9.04e-2)	4.90e-2(2.52e-2)	1.39e-1(7.33e-2)	4.40e-2(1.93e-2)	1.00e-1(6.53e-2)
TossNet (ours)	3.15e-2(1.82e-2)	**7.09e-2(4.68e-2)**	2.62e-2(1.45e-2)	8.40e-2(4.52e-2)	2.68e-2(1.33e-2)	7.26e-2(3.59e-2)
syn-TossNet (ours)	**3.11e-2(1.12e-2)**	8.02e-2(1.35e-2)	**2.21e-2(1.21e-2)**	**8.14e-2(3.25e-2)**	**2.02e-2(1.18e-2)**	**5.93e-2(3.25e-2)**

FDE of all ablation methods on the Real-World datasets.

TABLE IX
PREDICTION ERROR (ADE) OF TOSSNET ON THE DATASETS $\mathcal{O}_\text{S}$, $\mathcal{O}_\text{R}$ AND UNSEEN OBJECTS WITH INPUT SEQUENCES OF VARIED LENGTH {0.1 S, 0.5 S, 1.0S}

Seq. length/s	$\mathcal{O}_\text{s}$		$\mathcal{O}_\text{r}$		Unseen Objects	
	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)	Position (m)	Orientation (rad)
0.1	7.90e-3(1.90e-3)	1.93e-2(5.81e-3)	2.41e-2(5.40e-3)	6.21e-2(1.52e-2)	3.39e-2(6.00e-3)	6.59e-2(2.52e-2)
0.5	4.90e-3(2.90e-3)	9.20e-3(6.12e-3)	**1.88e-2(5.21e-3)**	5.39e-2(4.33e-2)	1.95e-2(4.99e-3)	5.11e-2(2.01e-2)
1.0	**3.10e-3(1.90e-3)**	**6.80e-3(6.80e-3)**	1.91e-2(5.26e-2)	**3.54e-2(1.78e-2)**	**1.81e-2(4.48e-3)**	**3.23e-2(1.27e-2)**

Overall, the prediction accuracy deteriorates quickly as the input sequence gets short, especially in orientation.

object flying dynamics, speeding up the learning process. On the other hand, the robot motion modality contains no clues on the robot–object interactions (e.g., robot grasp) and the physical properties of the tossed object. Therefore, r-TossNet cannot model the toss dynamics accurately, leading to limited prediction accuracy of the resulted object trajectory. The F/T values, particularly its spatio-temporal variations, in contrast, can reflect implicitly not only the motion characteristics of a robot toss action, but also contain rich information on the robot grasp and physical properties of the object. Therefore, ft-TossNet improves the modeling accuracy of robot toss dynamics and, in turn, enhances the prediction accuracy of object trajectories.

In addition, it can be observed in both Tables V–VIII and Fig. 14, *syn*-TossNet (last row in Tables V–VIII) achieves a higher prediction accuracy than TossNet. *syn*-TossNet differs from TossNet only by extracting and fusing additional multimodal correlated features in modeling the toss dynamics. It indicates that: 1) Our method can effectively extract the spatio-temporal correlations among synchronous F/T and motion observations in robot tossing, and 2) with the correlated multimodal features, our method can obtain a more accurate estimation of the robot toss dynamics.

Input Sequence Length: To further evaluate the effects of input sequence on modeling the robot toss dynamics and predicting the resulting object trajectories, we also experiment by providing the model (TossNet) with the input sequences of varied horizon length, i.e., $\{0.1, 0.5, 1.0\}$ s. The results of prediction error (ADE) on the datasets $\mathcal{O}_\text{s}$, $\mathcal{O}_\text{r}$ and unseen objects are summarized in Table IX. Overall, the prediction accuracy deteriorates quickly as the input sequence gets short, especially in object orientation. This is mainly because the shorter the input sequence, the less information the sequence contains to characterize the toss dynamics. For other experiments in this section, we set the length of the input sequence to be 1.0 s.

Model Inference Time: We also evaluate the model inference time, i.e., the time for the model to predict the object trajectory of a certain duration, which is critical for trajectory prediction in real tossing-centric applications. For example, in robot juggling, if it takes too much time for the model to output the predictions of object trajectories, there can be no enough reaction time for the robot to move and catch the tossed object.

We evaluate the model to predict the object trajectory of varied durations. The model inference time for the object trajectory of $\{0.1, 0.2, 0.5\}$ s are $\{12.2(0.1), 14.5(0.1), 20.1(0.3)\}$ ms, respectively, (standard deviations are in parentheses). In other words, the model inference time increases linearly with trajectory duration. In addition, compared with vision-based or other trajectory prediction methods which rely on the online observations of the objects to predict their follow-up trajectories autoregressively, our method uses only proprioceptive observations of robot tossing and predicts the object trajectory in one shot. Therefore, our model is much more computationally efficient in trajectory prediction and flexible for tossing-centric applications.

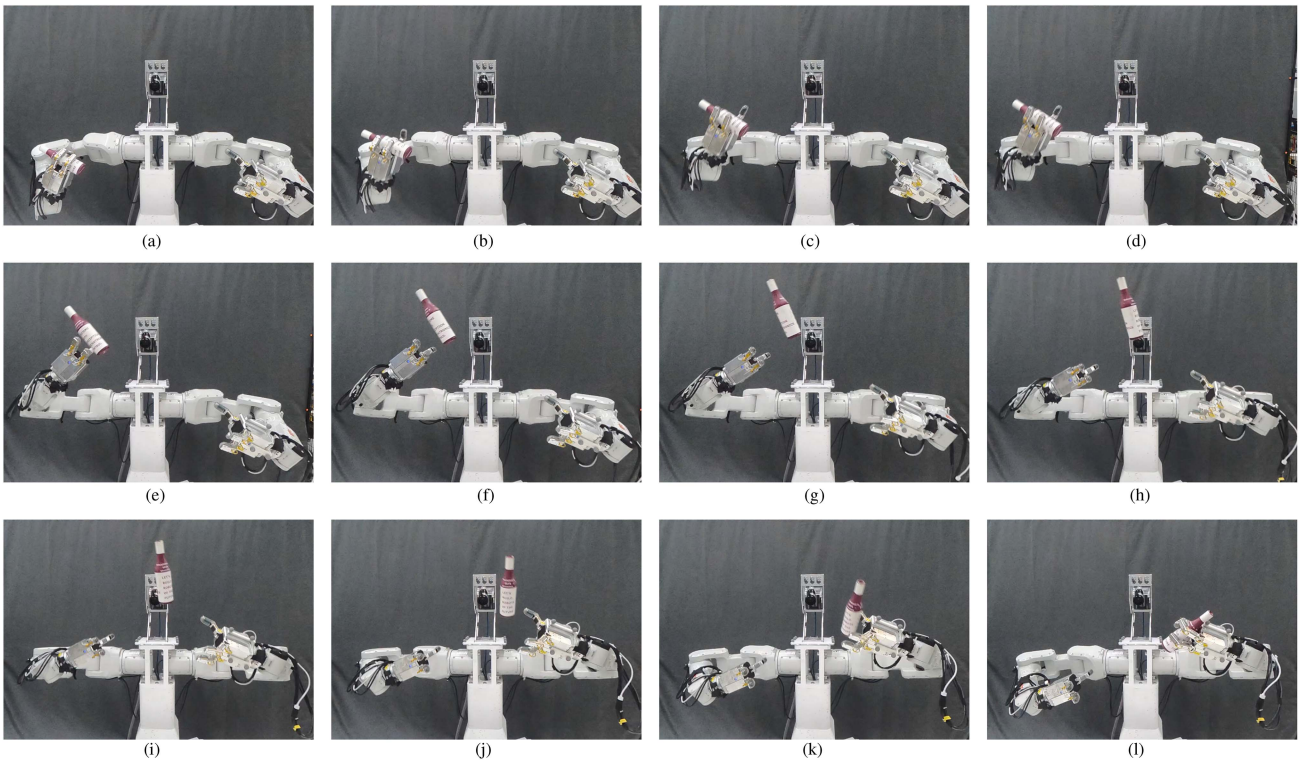


Fig. 15. Blind juggling robot system. With TossNet, a dual-arm IRB-1100 robot with multifingered hands can easily play blind juggling by tossing and catching a plastic bottle. As TossNet can predict the entire object trajectory accurately in real-time, the robot achieves a success rate of 100% in 20 juggling experiments. (a)–(d) Right arm tosses a bottle out. (e) Right arm releases the bottle and TossNet predicts the whole flying of the bottle in one shot. (e)–(l) Left arm moves to the intercept point ($t = 0.5$ s) on the predicted trajectory and catches the bottle stably.

D. Real Robot Applications

We apply our method to a variety of scenarios to further evaluate its deployment performance on real robots and demonstrate its potential usage in tossing-centric applications. All real robot experiments are recorded and can be found in the video of the Supplementary Material.

Different Robot Deployment: In addition to UR5 with which we discuss and demonstrate our method mostly, we also experiment with IRB-1100 from ABB, a higher performance industrial robot, to evaluate the deployment performance of our method on different robot systems. Before deployment, TossNet is re-trained with data collected from an IRB robot tossing an empty plastic bottle (see Fig. 15). Quantitatively, the position/orientation ADE and FDE results of the prediction trajectories on the IRB robot are 1.0e-3(1.51e-4)/2.0e-3(1.1e-3), and 1.5e-3(1.4e-4)/4.5e-3(2.7e-3), respectively. It demonstrates that, in addition to the UR5 robot, TossNet works well on the real IRB robot and further reduces the prediction errors to the millimeter/degree scale.

Blind Juggling Robot System: In this scenario, a dual-arm IRB-1100 robot with multifingered hands is programmed to play *blind* juggling by tossing and catching a plastic bottle (see Fig. 15). Blind juggling is challenging, particularly in nonvisual and short-distanced scenarios. It requires the robot to accurately and efficiently predict/track the object's motion and move swiftly to the intercept point to catch the flying object.

Specifically, after the right arm (for picking) picks and tosses out the bottle, i.e., after collecting all proprioceptive observations, TossNet predicts and outputs the flying trajectory of the bottle immediately to the robot controller. The controller then applies a *direct catching* strategy where the left arm (for catching) moves to the intercept point (the predicted object location at 0.5 s) directly on the predicted trajectory and closes its fingers according to the predicted timing.

We perform the juggling experiments 20 times and observe a success rate of 100%. It mainly benefits from two factors: 1) Our method guarantees high-prediction accuracy of the object trajectories such that the robot can follow and catch the object at the right intercept locations; and 2) Different from methods relying on the online observations of the object trajectory, our method predicts the entire trajectory in real-time and in one shot before the object flies out, which leaves sufficient time for the robot to move to intercept and catch the object in time.

Precise Pitching to Target: Beyond the object trajectory prediction, in this scenario, the UR5 robot is set to project a set of unseen objects into a moving *small* container (see Fig. 16). The container poses are randomly generated in the toss-reachable space and provided to TossNet before each experiment. Compared to target-throwing tasks showcased in previous literature [11], [51], precise-pitching exhibits a much narrower error tolerance. It requires the object's trajectory to precisely accommodate both the position and orientation of the target container.

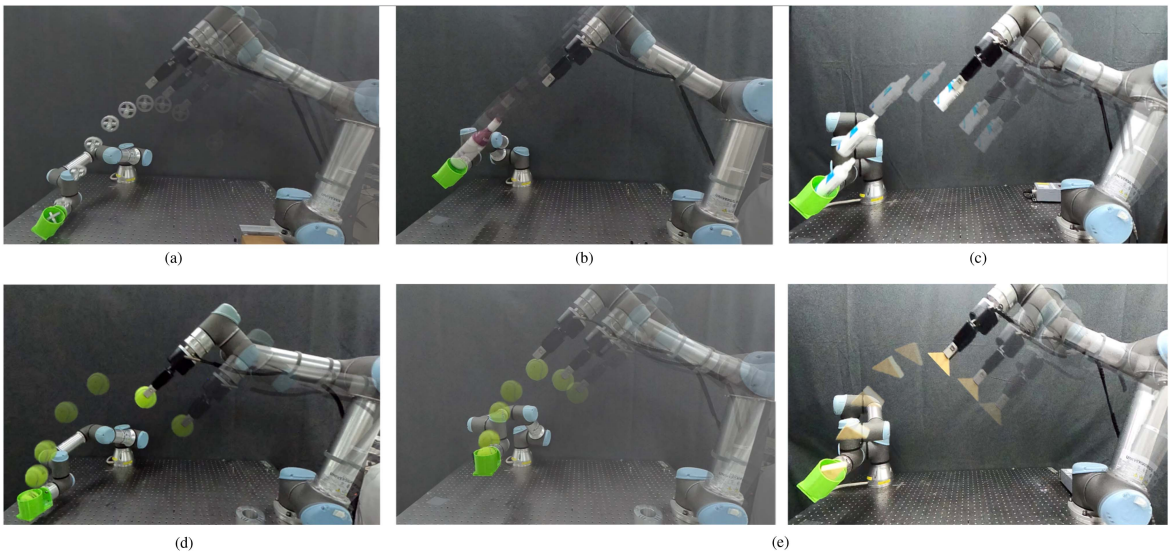


Fig. 16. Precise pitching-to-target. Robot pitches a set of unseen objects precisely into a *small* container with random target poses. Success rates for the above object in 20 experiments are (a) 60%, (b) 80%, (c) 90%, (d) 75%, and (e) 80%, respectively. The diameter of the target container is only 7.0 cm and the size of the axis-aligned bounding box for each tossed object is in parentheses. The pitching tasks here are particularly challenging, since the size of the container is very small compared with those of the tossed objects. (a) Mechanical assembly (5:5 × 5:5 × 6:5cm3), (b) Plastic bottle (5:7 × 5:7 × 22:0cm3), (c) Spray bottle (4:0 × 4:0 × 13:5cm3), (d) Ball with different pitching target poses (7:0 × 7:0 × 7:0cm3), (e) Toy brick (4:2 × 3:0 × 8:4cm3).

Since TossNet is a forward model, we use it as a trajectory predictor to search for approximate toss actions using bisection searching. Specifically, given an unseen object and a reachable pose of the static cup, we first select a random stable grasp on the object and sample a release position and velocity for the robot end-effector. Subsequently, we apply the cubic spline [89] and slerp [90] methods for translational and rotational interpolations, respectively. This yields a highly smooth trajectory for robot throwing. By performing the mocked throwing trajectory (i.e., the robot does not actually release the object), we obtain the corresponding F/T observations, which contain rich information on the tossing dynamics, such as the object properties and robot grasps. We then employ our method to predict the resulting object's flying trajectory, which determines if the object can fit into the target cup. This process is repeated and accelerated using bisection searching until a robot action that can successfully toss the object into the target cup is found.

We leverage the model trained with dataset $\mathcal{O}_s$ [see Fig. 12(d)] to make predictions, while the pitched objects are all *unseen* [see Fig. 12(e)] to TossNet, i.e., they are not included in training the model. We run 20 experiments with random target pitching poses for each object and observe an average success rate of 77.0%. On the one hand, the primary reason for failure stems from prediction errors, particularly in object orientation. On the other hand, the pitching tasks here are particularly challenging, given the close resemblance in size between the container and the tossed objects. For example, the diameter of the ball [see Fig. 16(d)] is 7.0 cm, which is the same as the container, leaving almost no error tolerance for pitching.

In addition, although our method can accurately measure and model the toss dynamics from robot proprioceptive observations, contributing to the high prediction accuracy of object motions and success rates of precise robot pitching, executing a sampled robot toss action, however, can be easily disturbed and diverged with noise. In this regard, rather than searching and executing in an open manner, we aim to include our method in a closed loop to achieve more precise, adaptive, and controllable robot tossing in the future. To this end, a controller can be built on top of the proposed TossNet, to reactively adjust and control the toss actions based on real-time predictions of TossNet using proprioceptive observations, and to toss random objects into dynamically moving containers. More advanced searching strategies, such as golden section searching and optimization-based searching [52], [91], can also be incorporated to accelerate the searching process.

V. Conclusion

We have presented TossNet, an LSTM-based method that can jointly model the robot toss dynamics and predict the flying trajectories of arbitrary robot-tossed rigid objects. Contrary to many previous studies, which rely on tailored and/or exteroceptive sensors, e.g., stereo cameras, our method leverages only the onboard proprioceptive robot motion and F/T sensors. It makes our method an effective, lightweight, and reproducible strategy, particularly for dynamic manipulation. We qualitatively show that our proposed method outperforms state-of-the-art methods in modeling the toss dynamics with proprioceptive perceptions. In addition, our method can accurately predict the entire object trajectories without online observations in one shot and in nearly real-time. These combined strengths contribute to high success rates when applying our proposed method to a variety of challenging toss-centric applications, such as blind robot juggling and precise pitching. Our proposed work is expected to have a long-term impact in areas such as dynamic and nonprehensile manipulation [1], [2], [3]. By pushing the boundaries of what

is achievable with proprioceptive feedback and sensor fusion, our work paves a new way for handling complex real-world manipulation tasks.

Future work will include extending our method to model the robot tossing of unknown objects with more complex physical properties, e.g., deformable and fluid objects, and other instances of dynamic manipulation such as pushing and swinging. In addition, although our method exhibits remarkable generalization due to the proprioceptive perception strategy and consistent trajectory parameterization in proprioceptive robot tossing, some robot-specific factors, such as the gripper dynamics, highly depend on the hardware and may not generalize well across different robot platforms. In this regard, in addition to gathering a larger amount of in-distribution data from a diverse array of robot platforms, strategies on domain generalization and shift [92], [93] will also be investigated to improve the model's generalization across different hardwares. We also seek to explore robot tossing in the context of human–robot interaction [9], [25], [27], [94]. In these settings, human-oriented factors such as human motions, safety, and comfort, are to be systematically formulated and addressed to facilitate the development of more dynamically capable and collaborative robot assistants. To this end, imitation-based learning methods to retarget, learn, and reproduce human–human throw and catch skills to robots will also be investigated.

REFERENCES

[1] M. T. Mason and K. M. Lynch, "Dynamic manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1993, pp. 152–159.

[2] K. M. Lynch and M. T. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *Int. J. Robot. Res.*, vol. 18, pp. 64–92, 1999.

[3] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1711–1718, Jul. 2018.

[4] M. M. Schill and M. Buss, "Robust ballistic catching: A hybrid system stabilization problem," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1502–1517, Dec. 2018.

[5] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, 2019, Art. no. eaat8414.

[6] N. Furukawa, A. Namiki, S. Taku, and M. Ishikawa, "Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2006, pp. 181–187.

[7] N. C. Dafle et al., "Extrinsic dexterity: In-hand manipulation with external forces," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 1578–1585.

[8] C. Wang, S. Wang, B. Romero, F. Veiga, and E. Adelson, "SwingBot: Learning physical features from in-hand tactile exploration for dynamic swing-up manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5633–5640.

[9] J. Kober, M. Glisson, and M. Mistry, "Playing catch and juggling with a humanoid robot," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2012, pp. 875–881.

[10] H. Frank, N. W.-Wojtasik, B. Hagebeuker, G. Novak, and S. Mahlknecht, "Throwing objects–A bio-inspired approach for the transportation of parts," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2006, pp. 91–96.

[11] 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, Aug. 2020.

[12] H. I. Christensen and G. D. Hager, "Sensing and estimation," in *Springer Handbook of Robotics*. Berlin, Germany: Springer, 2016, pp. 91–112.

[13] Y. Shirai and H. Inoue, "Guiding a robot by visual feedback in assembling tasks," *Pattern Recognit.*, vol. 5, no. 2, pp. 99–108, 1973.

[14] R. Sharma, J.-Y. Herve, and P. Cucka, "Dynamic robot manipulation using visual tracking," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1992, pp. 1844–1845.

[15] F. Aghili, "A prediction and motion-planning scheme for visually guided robotic capturing of free-floating tumbling objects with uncertain dynamics," *IEEE Trans. Robot.*, vol. 28, no. 3, pp. 634–649, Jun. 2012.

[16] M. Costanzo, "Control of robotic object pivoting based on tactile sensing," *Mechatronics*, vol. 76, 2021, Art. no. 102545.

[17] A. Grover, C. Grebe, P. Nadeau, and J. Kelly, "Under pressure: Learning to detect slip with barometric tactile sensors," 2021, *arXiv:2103.13460*.

[18] Z. Xu et al., "COCOI: Contact-aware online context inference for generalizable non-planar pushing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 176–182.

[19] S. Dong, W. Yuan, and E. H. Adelson, "Improved gelsight tactile sensor for measuring geometry and slip," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 137–144.

[20] J. W. James and N. F. Lepora, "Slip detection for grasp stabilization with a multifingered tactile robot hand," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 506–519, Apr. 2021.

[21] P. Nadeau, M. Giamou, and J. Kelly, "Fast object inertial parameter identification for collaborative robots," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 3560–3566.

[22] T. Bi, C. Sferrazza, and R. D'Andrea, "Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5761–5768, Jul. 2021.

[23] W. Hong and J.-J. E. Slotine, "Experiments in hand-eye coordination using active vision," in *Proc. Conf. Exp. Robot. IV*, 1997, pp. 130–139.

[24] U. Frese et al., "Off-the-shelf vision for a robotic ball catcher," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2001, pp. 1623–1629.

[25] M. Riley and C. G. Atkeson, "Robot catching: Towards engaging human-humanoid interaction," *Auton. Robots*, vol. 12, no. 1, pp. 119–128, 2002.

[26] S. Kim and A. Billard, "Estimating the non-linear dynamics of free-flying objects," *Robot. Auton. Syst.*, vol. 60, no. 9, pp. 1108–1122, 2012.

[27] D. Carneiro, F. Silva, and P. Georgieva, "Robot anticipation learning system for ball catching," *Robotics*, vol. 10, no. 4, 2021, Art. no. 113.

[28] A. Almazov, "Ball sense," Accessed: Jun. 21, 2024. [Online]. Available: https://alvin-almazov.com/tennis-eng/ball-sense/

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Conf. Int. Speech Commun. Association, Interspeech*, 2014, pp. 338–342.

[31] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. F.-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 961–971.

[32] Y. Gai, Y. Kobayashi, Y. Hoshino, and T. Emaru, "Motion control of a ball throwing robot with a flexible robotic arm," *Int. J. Comput. Inf. Eng.*, vol. 7, no. 7, pp. 937–945, 2013.

[33] W. Mori, J. Ueda, and T. Ogasawara, "A 1-DoF dynamic pitching robot that independently controls velocity, angular velocity and direction of a ball," *Adv. Robot.*, vol. 24, no. 5/6, pp. 921–942, 2010.

[34] T. Senoo, A. Namiki, and M. Ishikawa, "High-speed throwing motion based on kinetic chain approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 3206–3211.

[35] D. M. Lofaro, R. Ellenberg, P. Oh, and J.-H. Oh, "Humanoid throwing: Design of collision-free trajectories with sparse reachable maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1519–1524.

[36] D. E. Stewart, "Rigid-body dynamics with friction and impact," *SIAM Rev.*, vol. 42, no. 1, pp. 3–39, 2000.

[37] R. Ronsse, P. Lefèvre, and R. Sepulchre, "Sensorless stabilization of bounce juggling," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 147–159, Feb. 2006.

[38] R. Ronsse, P. Lefevre, and R. Sepulchre, "Rhythmic feedback control of a blind planar juggler," *IEEE Trans. Robot.*, vol. 23, no. 4, pp. 790–802, Aug. 2007.

[39] P. Reist and R. D'Andrea, "Design and analysis of a blind juggling robot," *IEEE Trans. Robot.*, vol. 28, no. 6, pp. 1228–1243, Dec. 2012.

[40] K. Muelling, J. Kober, and J. Peters, "Learning table tennis with a mixture of motor primitives," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2010, pp. 411–416.

[41] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2015, pp. 127–135.

[42] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song, "DensePhysNet: Learning dense physical object representations via multi-step dynamic interactions," *Robot.: Sci. Syst.*, 2019.

[43] N. Fazeli, S. Zapolsky, E. Drumwright, and A. Rodriguez, "Learning data-efficient rigid-body contact models: Case study of planar impact," in *Proc. Conf. Robot Learn.*, 2017, pp. 388–397.

[44] Y. Jiang, J. Sun, and C. K. Liu, "Data-augmented contact model for rigid body simulation," in *Proc. Annu. Learn. Dyn. Control Conf.*, 2022, pp. 378–390.

[45] S. Pfrommer, M. Halm, and M. Posa, "Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations," in *Proc. Conf. Robot Learn.*, 2021, pp. 2279–2291.

[46] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Rev. Phys.*, vol. 3, no. 6, pp. 422–440, 2021.

[47] E. W. Aboaf, C. G. Atkeson, and D. J. Reinkensmeyer, "Task-level robot learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1988, pp. 1309–1310.

[48] E. W. Aboaf, S. M. Drucker, and C. G. Atkeson, "Task-level robot learning: Juggling a tennis ball more accurately," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1989, pp. 1290–1295.

[49] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman, "Deep predictive policy training using reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2351–2358.

[50] A. Kloss, S. Schaal, and J. Bohg, "Combining learned and analytical models for predicting action effects from sensory data," *Int. J. Robot. Res.*, vol. 41, no. 8, pp. 778–797, 2022.

[51] D. Bianchi, M. G. Antonelli, C. Laschi, A. M. Sabatini, and E. Falotico, "Softoss: Learning to throw objects with a soft robot," *IEEE Robot. Autom. Mag.*, 2023, early access, Sep. 29, 2023, doi: 10.1109/MRA.2023.3310865.

[52] D. Bianchi, M. G. Antonelli, C. Laschi, A. M. Sabatini, and E. Falotico, "Learning-based inverse dynamic controller for throwing tasks with a soft robotic arm," in *Proc. Int. Conf. Informat. Control, Automat., Robot.*, 2023, pp. 424–432.

[53] B. Huang et al., "Dynamic handover: Throw and catch with bimanual hands," in *Proc. Conf. Robot. Learn.*, 2023, pp. 1887–1902.

[54] J. Wu et al., "Tidybot: Personalized robot assistance with large language models," *Auton. Robots*, vol. 47, no. 8, pp. 1087–1102, 2023.

[55] F. Raptopoulos, M. Koskinopoulou, and M. Maniadakis, "Robotic pick-and-toss facilitates urban waste sorting," in *Proc. IEEE Int. Conf. Automat. Sci. Eng.*, 2020, pp. 1149–1154.

[56] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik, "Learning visual predictive models of physics for playing billiards," 2015, *arXiv:1511.07404*.

[57] S. E. Navarro, B. Hein, and H. Wörn, "Capacitive tactile proximity sensing: From signal processing to applications in manipulation and safe human-robot interaction," in *Soft Robotics: Transferring Theory to Application*. Berlin, Germany: Springer, 2015, pp. 54–65.

[58] T. Senoo, Y. Yamakawa, S. Mizusawa, A. Namiki, M. Ishikawa, and M. Shimojo, "Skillful manipulation based on high-speed sensory-motor fusion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 1611–1612.

[59] Y. Zhu et al., "Reinforcement and imitation learning for diverse visuomotor skills," *Robot.: Sci. Syst.*, 2018.

[60] L. Han, W. Xu, B. Li, and P. Kang, "Collision detection and coordinated compliance control for a dual-arm robot without force/torque sensing based on momentum observer," *IEEE/ASME Trans. Mechatron.*, vol. 24, no. 5, pp. 2261–2272, Oct. 2019.

[61] B. S. Homberg, R. K. Katzschmann, M. R. Dogar, and D. Rus, "Robust proprioceptive grasping with a soft robot hand," *Auton. Robots*, vol. 43, pp. 681–696, 2019.

[62] J. Lloyd and N. F. Lepora, "Goal-driven robotic pushing using tactile and proprioceptive feedback," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1201–1212, Apr. 2021.

[63] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1049–1065, Oct. 2014.

[64] S. S. M. Salehian, M. Khoramshahi, and A. Billard, "A dynamical system approach for softly catching a flying object: Theory and experiment," *IEEE Trans. Robot.*, vol. 32, no. 2, pp. 462–471, Apr. 2016.

[65] T. Senoo, A. Namiki, and M. Ishikawa, "High-speed batting using a multi-jointed manipulator," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2004, pp. 1191–1196.

[66] H.-M. Joe, J. Lee, and J.-H. Oh, "Dynamic nonprehensile manipulation of a moving object using a batting primitive," *Appl. Sci.*, vol. 11, no. 9, 2021, Art. no. 3920.

[67] N. C. Dafle et al., "Regrasping objects using extrinsic dexterity," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 2560–2560.

[68] K. S. Bhat, S. M. Seitz, J. Popović, and P. K. Khosla, "Computing the physical parameters of rigid-body motion from video," in *Proc. Eur. Conf. Comput. Vis.*, 2002, pp. 551–565.

[69] B. Beigzadeh, M. N. Ahmadabadi, A. Meghdari, and A. Akbarimajd, "A dynamic object manipulation approach to dynamic biped locomotion," *Robot. Auton. Syst.*, vol. 56, no. 7, pp. 570–582, 2008.

[70] S. Kansal and S. Mukherjee, "Kinematic and dynamic analysis of a dexterous multi-fingered delta robot for object catching," *Robotica*, vol. 40, no. 8, pp. 2878–2908, 2022.

[71] K. Nguyen, J. Krumm, and C. Shahabi, "Gaussian process for trajectories," 2021, *arXiv:2110.03712*.

[72] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer networks for trajectory forecasting," in *Proc. Int. Conf. Pattern Recognit.*, 2021, pp. 10 335–10 342.

[73] K. Mironov, I. Vladimirova, and M. Pongratz, "Processing and forecasting the trajectory of a thrown object measured by the stereo vision system," *IFAC-PapersOnLine*, vol. 48, no. 11, pp. 28–35, 2015.

[74] K. Mironov, "Transport by robotic throwing and catching: Accurate stereo tracking of the spherical object," in *Proc. Int. Conf. Ind. Eng., Appl., Manuf.*, 2017, pp. 1–6.

[75] N. Qadeer, J. H. Shah, M. Sharif, M. A. Khan, G. Muhammad, and Y.-D. Zhang, "Intelligent tracking of mechanically thrown objects by industrial catching robot for automated in-plant logistics 4.0," *Sensors*, vol. 22, no. 6, 2022, Art. no. 2113.

[76] Y. Liu, P. Sun, and A. Namiki, "Target tracking of moving and rotating object by high-speed monocular active vision," *IEEE Sensors J.*, vol. 20, no. 12, pp. 6727–6744, Jun. 2020.

[77] S. Kim, E. Gribovskaya, and A. Billard, "Learning motion dynamics to catch a moving object," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2010, pp. 106–111.

[78] K. Dong, K. Pereida, F. Shkurti, and A. P. Schoellig, "Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 6718–6725.

[79] G. Bätz, B. Weber, M. Scheint, D. Wollherr, and M. Buss, "Dynamic contact force/torque observer: Sensor fusion for improved interaction control," *Int. J. Robot. Res.*, vol. 32, no. 4, pp. 446–457, 2013.

[80] J. Stüber, C. Zito, and R. Stolkin, "Let's push things forward: A survey on robot pushing," *Front. Robot. AI*, vol. 7, 2020, Art. no. 8.

[81] E. Coumans and Y. Bai, "Pybullet, A python module for physics simulation for games, robotics and machine learning," pp. 2016–2018 . [Online]. Available: http://pybullet.org

[82] A. T. Miller and P. K. Allen, "Graspit! A versatile simulator for robotic grasping," *IEEE Robot. Autom. Mag.*, vol. 11, no. 4, pp. 110–122, Dec. 2004.

[83] S. Pellegrini, A. Ess, K. Schindler, and L. V. Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 261–268.

[84] E. J. Heravi and S. Khanmohammadi, "Long term trajectory prediction of moving objects using Gaussian process," in *Proc. Int. Conf. Robot. Vis. Signal Process.*, 2011, pp. 228–232.

[85] S. Becker, R. Hug, W. Hübner, and M. Arens, "An evaluation of trajectory prediction approaches and notes on the trajnet benchmark," 2018, *arXiv:1805.07663*.

[86] A. Vaswani et al., "Attention is all you need," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[87] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1243–1252.

[88] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[89] S. McKinley and M. Levine, "Cubic spline interpolation," *College Redwoods*, vol. 45, no. 1, pp. 1049–1060, 1998.

[90] E. B. Dam, M. Koch, and M. Lillholm, *Quaternions, interpolation and animation*, vol. 2. Copenhagen, Denmark: Datalogisk Institut, Københavns Univ., 1998.

[91] S. Mirjalili and S. Mirjalili, "Genetic algorithm," *Evol. Algorithms Neural Netw.: Theory Appl.*, vol. 780, pp. 43–55, 2019.

[92] J. Cui and J. Trinkle, "Toward next-generation learned robot manipulation," *Sci. Robot.*, vol. 6, no. 54, 2021, Art. no. eabd9461.

[93] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4396–4415, Apr. 2023.

[94] E. J. Carter, M. N. Mistry, G. P. K. Carr, B. A. Kelly, and J. K. Hodgins, "Playing catch with robots: Incorporating social gestures into physical interactions," in *Proc. IEEE Int. Symp. Robot Hum. Interactive Commun.*, 2014, pp. 231–236.

Lipeng Chen (Member, IEEE) received the Ph.D. degree in computer science from the Robotic Manipulation Lab, University of Leeds, Leeds, U.K., in 2019.

He was a Postdoctoral Researcher with the University of Edinburgh, Edinburgh, U.K. He is currently a Senior Research Scientist with Tencent Robotics X, Shenzhen, China. His research interests include autonomous robotic manipulation and physical human–robot collaboration.

Weifeng Lu (Student Member, IEEE) received the B.S. degree in optical information science and technology from the School of Science, Xi'an Jiaotong University, Xi'an, China, in 2014, and the M.S. degree in control engineering from the Department of Automation, Tsinghua University, Beijing, China, in 2018. He is currently working toward the Ph.D. degree in biomedical engineering with City University of Hong Kong, Hong Kong, China.

His research interests include human–robot interaction and reinforcement learning.

Kun Zhang (Student Member, IEEE) received the B.E. degree in machine design and manufacturing from Harbin Engineering University, Harbin, China, in 2016, and the M.S. degree in electromechanical engineering from the Department of Electromechanical Engineering, University of Macau, Macau, China, in 2019. He is currently working toward the Ph.D. degree in electronic and computer Engineering with the Department of Electronic and Computer Engineering, the Hong Kong University of Science and Technology, Hong Kong, China.

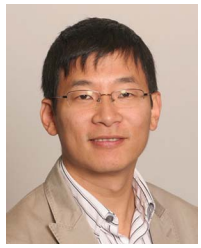
His research interests include robotic perception and manipulation.

Yizheng Zhang received the B.E. degree in automation from the Xi'an University of Technology, Xi'an, China, in 2018, and the M.S. degree in computer science from ShanghaiTech University, Shanghai, China, in 2021.

He is currently a Robotics Engineer with Tencent Robotics X, Shenzhen, China. His research interests include legged locomotion, robotic manipulation, and reinforcement learning algorithms for robotics.

Longfei Zhao (Member, IEEE) received the Ph.D. degree in mechanical engineering from Quebec University, Montreal, QC, Canada, in 2015.

From 2016 to 2018, he was a Senior Roboticist with Kinova Robotics, Boisbriand, Canada. He is currently a Principal Research Scientist with Tencent Robotics X, Shenzhen, China. His research interests include robot control theory, motion planning, and human–robot interaction and collaboration.



Yu Zheng (Senior Member, IEEE) received the dual B.E. degrees in mechanical engineering and computer science, and the Ph.D. degree in mechatronics from Shanghai Jiao Tong University, Shanghai, China, in 2001 and 2007, respectively. He also received the M.S. and Ph.D. degrees in computer science from the University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, in 2011 and 2013, respectively.

From 2010 to 2014, he was with Disney Research Pittsburgh. From 2014 to 2018, he was an Assistant Professor with the Department of Electrical and Computer Engineering, the University of Michigan-Dearborn. In 2018, he joined Tencent Robotics X, Shenzhen, China, where he is currently a Principal Research Scientist and the Team Lead of Control Center. His research interests include multicontact/multibody robotic systems, robotic grasping and manipulation, legged locomotion, and geometric algorithms for robotics.