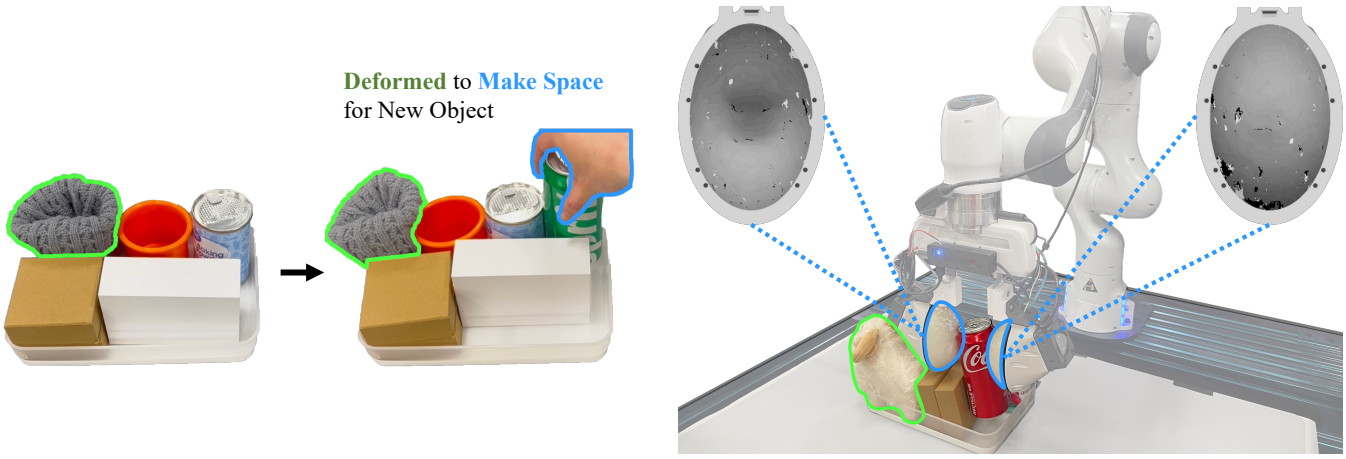
RoboPack: Learning Tactile-Informed Dynamics Models for Dense Packing

Bo Ai[1,3,4*] **Stephen Tian**[1*] **Haochen Shi**[1] **Yixuan Wang**[2]
Cheston Tan[3,4] **Yunzhu Li**[2] **Jiajun Wu**[1]
*Equal contribution

[1]Stanford University, USA [2]University of Illinois Urbana-Champaign, USA
[3]IHPC, Agency for Science, Technology and Research, Singapore
[4]CFAR, Agency for Science, Technology and Research, Singapore
https://robo-pack.github.io



(a) Dense Packing by a Human (b) Dense Packing by a Robot with Tactile Sensors

Fig. 1: **Tactile sensing for dense packing.** Tactile feedback is critical in tasks with heavy occlusion and rich contact, such as dense packing. (a) Humans rely on tactile sensations from their hands to navigate space and fit a water bottle into a suitcase. (b) Likewise, tactile sensing is crucial for robots to perform dense packing tasks, such as placing a can into a packed tray.

***Abstract*—Tactile feedback is critical for understanding the dynamics of both rigid and deformable objects in many manipulation tasks, such as non-prehensile manipulation and dense packing. We introduce an approach that combines visual and tactile sensing for robotic manipulation by learning a neural, tactile-informed dynamics model. Our proposed framework, RoboPack, employs a recurrent graph neural network to estimate object states, including particles and object-level latent physics information, from historical visuo-tactile observations and to perform future state predictions. Our tactile-informed dynamics model, learned from real-world data, can solve downstream robotics tasks with model-predictive control. We demonstrate our approach on a real robot equipped with a compliant Soft-Bubble tactile sensor on non-prehensile manipulation and dense packing tasks, where the robot must infer the physics properties of objects from direct and indirect interactions. Trained on only an average of 30 minutes of real-world interaction data per task, our model can perform online adaptation and make touch-informed predictions. Through extensive evaluations in both long-horizon dynamics prediction and real-world manipulation, our method demonstrates superior effectiveness compared to previous learning-based and physics-based simulation systems.**

I. Introduction

Imagine packing an item into a nearly full suitcase. As humans, we typically first form a visual representation of the scene and then make attempts to insert the object, *feeling* the compliance of the objects already inside to decide where and how to insert the new object. If a particular region feels soft, we can then apply additional force to make space and squeeze the new object in. This process is natural for us humans but very challenging for current robotic systems.

What would it take to produce adept packing capabilities in robots? Firstly, a robot needs to understand how its actions will affect the objects in the scene and how those objects will interact with each other. Dynamics models of the world predict exactly this: how the state of the world will change based on a robot's action. However, most physics-based dynamics models (e.g., physical simulators), assume full-state information and typically exhibit significant sim-to-real gaps, especially in unstructured scenes involving deformable objects.

At the same time, tasks such as dense packing present significant challenges due to severe occlusions among objects, creating partially observable scenarios where vision alone is insufficient to determine the properties of an object, such as its softness, or assess whether there is space for additional objects. For effective operation, the robot must integrate information from its actions and the corresponding tactile sensing into

its planning procedure. However, the optimal method for incorporating tactile sensing information into dynamic models is unclear. Naïvely integrating tactile sensing into a model's state space can perform poorly because the intricate contacts make tactile modeling a challenging problem, as we will also show empirically later on.

To tackle these challenges, in this work, we propose to 1) learn dynamics directly from *real* physical interaction data using powerful deep function approximators, 2) equip our robotic system with a compliant vision-based Soft-Bubble tactile sensor [22], and 3) develop a learning-based method for effective estimation of latent physics information from tactile feedback in interaction histories.

Because learning dynamics in raw pixel observation space can be challenging due to the problem's high dimensionality, we instead model scenes using keypoint particles [37, 29, 49, 48, 50]. Finding and tracking meaningful keypoint representations of densely packed scenes over time is itself challenging due to the proximity of objects and inter-occlusions. In this work, we extend an optimization-based point tracking system to preprocess raw observation data into keypoints.

We use the Soft-Bubble tactile sensor [22], which is ideal for tasks like dense packing, as it can safely sustain stress from the handheld object in all directions and provides high-resolution percepts of the contact force via an embedded RGB-D camera.

Finally, we propose an effective way to incorporate tactile information into our system by learning a separate state estimation module that incorporates tactile information from prior interactions and infers latent physics vectors that contain information that may be helpful for future prediction. This allows us to learn *tactile-informed* dynamics.

We call this system comprising keypoint-based perception, latent physics vector and state estimation from tactile information, dynamics prediction, and model-based planning **RoboPack**. We deploy RoboPack on two real-world settings—a tool-use manipulation and a dense packing task. These tasks involve multi-object interactions with complex dynamics that cannot be determined from vision alone. Furthermore, these settings are exceptionally challenging because, unlike prior work that only estimates the physical properties of the object held in hand, our tasks also require estimating the physical properties of objects with which the robot interacts *indirectly* through the handheld object.

We find that our method can successfully leverage histories of visuo-tactile information to improve prediction, with models trained on just 30 minutes of real-world interaction data per task on average. Through empirical evaluation, we demonstrate that RoboPack outperforms previous works on dynamics learning, an ablation without tactile information, and physics simulator-based methods in dynamics prediction and downstream robotic tasks. We further analyze the properties of the learned latent physics vectors and their relationship with interaction history length.

II. Related Work

A. Learning Dynamics Models

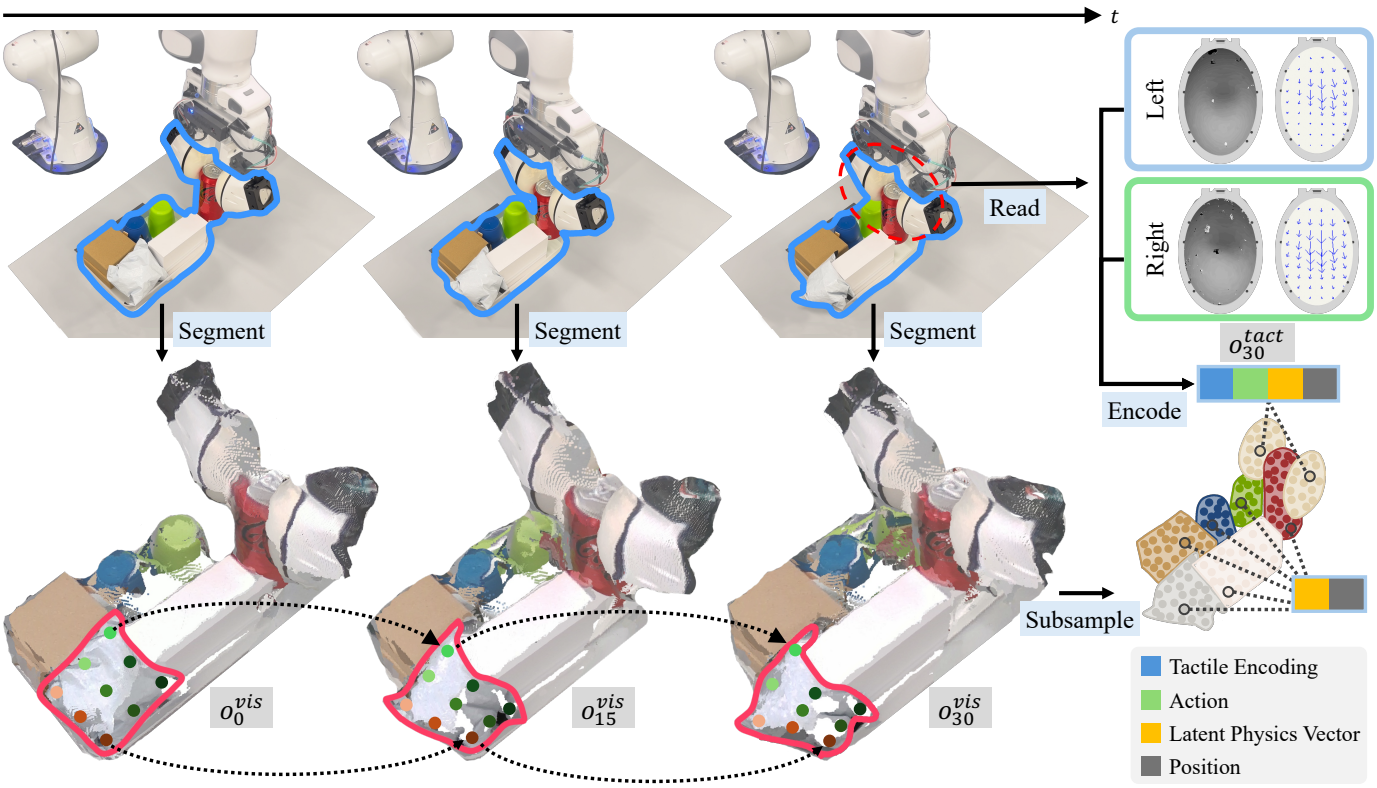
Simulators developed to model rigid and non-rigid bodies approximate real-world physics, often creating a significant sim-to-real gap [57, 17, 41]. To address this, we use a graph neural network (GNN)-based dynamics model trained directly on real-world robot interaction data, aligning with data-driven approaches for learning physical dynamics [42, 36]. Recent works have demonstrated inspiring results in learning the complex dynamics of objects such as clothes [34], ropes [5], and fluid [26], with various representations including low-dimensional parameterized shapes [38], keypoints [30], latent vectors [24], and neural radiance fields [31]. RoboPack, inspired by previous works [29, 47, 2], focuses on the structural modeling of objects with minimal assumptions about underlying physics. This approach overcomes the limitations of physics simulators by directly learning from real-world dynamics. Prior work on GNN-based dynamics learning [48, 49, 50, 55, 6] heavily relies on visual observations for predicting object dynamics, failing to capture unobserved latent variables that affect real-world dynamics, such as object physical properties. To address this challenge, our method incorporates tactile sensing into dynamics learning and leverages history information for state estimation, offering a robust solution to overcome the constraints of vision-only models.

B. Model-Free and Model-Based Reinforcement Learning

Reinforcement learning (RL) aims to derive policies directly from interactions. Our method contrasts with model-free RL approaches [40, 32, 12, 19, 27], by incorporating an explicit dynamics model, enhancing interpretability and including structured priors for improved generalization. Our work is closer to model-based RL [16, 13, 42, 46, 39, 62] in that we combine learned world models with planning via trajectory optimization. In particular, we learn world models in an offline manner from pre-collected interaction data, avoiding risky trial-and-error interactions in the real world. However, our approach is different from existing offline model-based RL [45, 59, 9, 54, 15] as it leverages multiple sensing modalities, i.e., tactile and visual perception. This multi-modal approach provides a more comprehensive understanding of both global geometry and the intricate local physical interactions between the robot gripper and objects. Moreover, our method addresses challenges in scenarios where visual observations are not always available. It uses tactile observation histories to estimate partially observable states, enabling online adaptation to different dynamics. This integration of offline model learning, multi-modal perception, and online adaptation equips our system with adaptive control behaviors for complex tasks.

C. Tactile Sensing for Robotic Manipulation

Tactile sensing plays an important role in both human and robot perception [7]. Among all categories of tactile sensors, vision-based sensors such as [60, 8, 25, 33] can achieve accurate 3D shape perception of their sensing surfaces. In our work, we use the Soft-Bubble tactile sensor [22] which



(a) 3D Point Tracking on Point Cloud Observations (b) Scene Representation

Fig. 2: **RoboPack's perception module.** (a) We construct a trajectory comprising particle representations of the scene, maintaining correspondence via 3D point tracking on the point cloud data. (b) These particles facilitate the creation of a visual scene representation, denoted as o_t^{vis}. For points representing the Soft-Bubble grippers, tactile encodings o_t^{tact} and latent physics vectors are integrated as extra attributes of the particles. We note that while the 3D point tracking module is needed at training time, during deployment the visual feedback can be replaced by predictions from our state estimator. This estimator auto-regressively predicts object particle positions from tactile interaction history and reduces reliance on dense visual feedback, which can be difficult to obtain due to visual occlusions.

offers a unique combination of compliance, lightweight design, robustness to continuous contact, and the ability to capture detailed geometric features through high-resolution depth images [22, 52]. Previous studies have successfully integrated vision and tactile feedback in robotic manipulation using parallel grippers [4, 10, 28] and dexterous hands [44, 53, 61]. In these tasks, vision effectively offers a comprehensive understanding of the scene's semantics, while tactile sensing delivers accurate geometry estimation for objects in contact that are often occluded. In our study, we explore the potential of integrating vision and tactile feedback for learning dynamics in tasks involving rich contact, occlusions, and a diverse set of objects with unknown physical properties, such as box pushing and dense packing.

III. Method

A. Overview

The objective of RoboPack is to manipulate objects with unknown physical properties in environments with heavy occlusions like dense packing. To formulate this problem, we define the observation space as $\mathcal{O}$, the state space as $\mathcal{S}$, and the action space as $\mathcal{A}$. Our goal is to learn a state estimator g that maps $\mathcal{O}$ to $\mathcal{S}$ and a transition function $\mathcal{T}$: $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$.

To efficiently learn dynamics from real-world multi-object interaction data, we would like to extract lower-dimensional representations of observations like keypoints. Furthermore, we require a mechanism to fuse tactile interaction histories into these representations without full tactile future prediction. Finally, to solve real robotic tasks, we need to leverage our learned model to plan robot actions.

Thus, our system has four main components: perception, state estimation, dynamics prediction, and model-predictive control, discussed in Section III-B, III-C, III-D, and III-E respectively. They are used together in the following way:

First, the perception system extracts particles from the scene as a visual representation o^{vis} and encodes tactile readings into latent embeddings o^{tact} attached to those particles.

Secondly, the state estimator g infers object states s from any prior interactions, which includes a single visual frame o_0^{vis}, the subsequent tactile observations $o_{0:t}^{tact}$, and the corresponding robot actions $a_{1:t-1}$:

$$\hat{s}_t = g(o_0^{vis}, o_{0:t}^{tact}, a_{1:t-1}). \quad (1)$$

Thirdly, to enable model-predictive control, we learn a dynamics prediction model f that predicts future states given the estimated current states and potential actions:

$$\hat{s}_{t+1} = f(\hat{s}_t, a_t). \tag{2}$$

Lastly, the future predictions are used to evaluate and optimize the cost of sampled action plans. The objective is to find a sequence of actions $a_0, ..., a_{H-1}$ to minimize a cost function $\mathcal{J}$ between the final states and a given target state s_g:

$$(a_0, ..., a_{H-1}) = \underset{a_0,...,a_{H-1} \in \mathcal{A}}{\arg\min} \mathcal{J}(\mathcal{T}(s_0, (a_0, .., a_{H-1})), s_g). \tag{3}$$

The robot executes the best actions and receives tactile feedback from the environment, with which it updates its estimates about object properties.

B. Perception

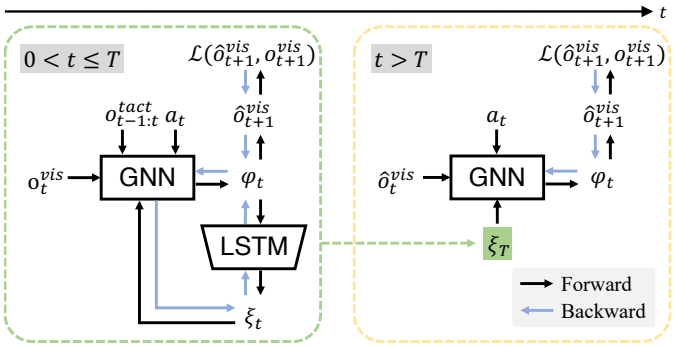
1) Visual Perception: Our visual perception module extends the formulation of D^{3}Fields [56], with an additional deformation term to handle non-rigid objects and mask-based closeness loss to better support multi-object scenes with occlusion. As shown in Figure 2(a), it takes in multi-view RGB-D observations and outputs tracked 3D keypoints for each object of interest. Critical for our training procedure, these keypoints maintain correspondences over time—a tracked point stays at the same region of an object throughout the trajectory.

First, we extract visual features for each object with a pre-trained DINOv2 model [43] and masks using Grounded SAM [43, 21, 35]. Through projection and interpolation, we can then compute semantic, instance, and geometric features for arbitrary 3D points. We initialize desired tracking points on object surfaces for an initial frame and formulate 3D keypoint tracking for subsequent frames as an optimization problem. The tracking objective has the following terms:

- **Distance to surface**. Use depth information to encourage points to be close to object surfaces.
- **Semantic alignment**. Align DINOv2 features between projected points in the current and initial frame.
- **Motion regularization**. Penalize large motion between consecutive frames to avoid jitter.
- **Mask consistency**. For multi-object packing settings with significant occlusion, we introduce an objective that constrains tracked points to be near the corresponding object masks, providing more consistent optimization signal for object pose than semantic alignment.

We optimize a translation and rotation transformation for each object with this objective. For deformable objects, we also predict axis-aligned shearing scales apart from a rigid transformation to track deformations.

2) Tactile Perception: As shown in the top right of Figure 2, our tactile perception module takes global force-torque and local force vectors as input and outputs embeddings for the tactile reading. Each Soft-Bubble tactile sensor provides its surface force distribution. This includes (1) shear force vectors $\{\langle q^x_{i,j}, q^y_{i,j}\rangle\}_{i,j}$, where i, j is the coordinate of a point on the



(a) State Estimator (b) Dynamics Predictor

Fig. 3: **RoboPack's dynamics module.** We perform state estimation and dynamics reasoning with a state estimator and a dynamics predictor respectively. (a) The state estimator auto-regressively predicts the positions of objects' particles and their latent physics vectors, reducing the dependency on dense visual feedback. (b) The dynamics predictor, conditioned on the estimated physics vectors, performs future prediction for planning. These modules share the same architecture, except that the state estimator has an LSTM that integrates history information and predicts physics parameters for each object.

2D surface of the bubble and x, y denote the vertical and horizontal axis of the tangent plane at that point, as well as (2) a global shear force torque vector and the overall force magnitude $\langle Q^x, Q^y, |Q|\rangle$. F^x, F^y are the mean of local force vectors across spatial dimensions, and $|Q|$ is defined as

$$|Q| = \sqrt{\max_{i,j} |q^x_{i,j}|^2 + \max_{i,j} |q^y_{i,j}|^2}, \tag{4}$$

3) Integrating Visual and Tactile Perception: As depicted in Figure 2(b), to integrate tactile observations with particle-based object representation, we first extract particles from the surface of the soft-bubble gripper by projecting the depth camera reading inside the gripper into 3D space. Next, we define a point-wise tactile signal as $\langle q^x_{i,j}, q^y_{i,j}, Q^x, Q^y, |Q|\rangle$ and train an auto-encoder that maps the point-wise signals independently into latent embeddings. Details regarding the auto-encoder architecture and training are available in Appendix A-A. We denote the collection of embeddings as the tactile observation o^{tact}. Lastly, we combine the object particles from the visual observation o^{vis} with the tactile sensor particles o^{tact} to form a unified particle representation of the scene.

C. State Estimation and Latent Physics Vector Inference

In real-world robotic manipulation, visual observations are not always available due to occlusion, but knowledge about object dynamics requires interactive feedback. In this work, we leverage tactile feedback to help estimate world states.

History information is often used to estimate the current state in POMDPs [1, 18, 23, 51]. Similarly, we seek to incorporate tactile history information into state estimation by employing a combination of graph neural networks (GNNs) and long-short term memory (LSTM), as shown in Figure 3(a).

We define our state as a tuple of object particles and an object-level latent physics vector, which capture the geometry and physics properties of objects respectively. In the following paragraphs, we describe how our method performs state estimation using history information.

At time $0 < t \leq T$, our state estimator g infers all states for $t = 1, ..., T$ autoregressively. Given the estimated previous state $\hat{s}_{t-1}$ and the tactile feedback at the previous and the current state $o^{tact}_{t-1:t}$, we construct a graph $G_{t-1} = \langle V_{t-1}, E_{t-1} \rangle$ with V_{t-1} as vertices and E_{t-1} as edges. For each node, $v_{i,t-1} = \langle x_{i,t-1}, c^o_{i,t-1} \rangle$, where $x_{i,t-1}$ is the particle position i at time $t-1$, and $c^o_{i,t-1}$ are particle attributes. The particle attributes contain (1) the previous and current tactile readings, $o^{tact}_{t-1:t}$, and (2) the latent physics vector of the object that the particle belongs to, $\xi_{\mathcal{M}_i,t-1}$, where $\mathcal{M}_i$ is the object index corresponds to the i-th particle, $1 \leq \mathcal{M}_i \leq Z$ and Z is the maximum number of objects in the scene. Formally, $c^o_{i,t-1} = \langle \xi_{t-1}, o^{tact}_{t-1:t} \rangle$. Note that here we implicitly assume that $\mathcal{M}$ is constant (i.e., objects only exhibit elastic and plastic deformations but not break apart), which generally holds for a large number of common manipulation tasks. Moreover, edges between pairs of particles are denoted $e_k = \langle u_k, v_k \rangle$, where u_k and v_k are the receiver and sender particle indices respectively, and $1 \leq u_k, v_k \leq |V_{t-1}|$ where k is the edge index. We construct graphs by connecting any nodes within a certain radius of each other.

Given the graph, we first use a node encoder f^{enc}_V and an edge encoder f^{enc}_E to obtain node and edge features, respectively:

$$h^v_{i,t-1} = f^{enc}_V(v_{i,t-1}), \qquad h^e_{k,t-1} = f^{enc}_E(e_{k,t-1}). \tag{5}$$

Then, the features are propagated through the edges in multiple steps, during which node effects are processed by neighboring nodes through learned MLPs. We summarize this procedure as f^{dec}_E, which outputs an aggregated effect feature for each node called ϕ_i:

$$\phi_{i,t-1} = f^{dec}_E(h^v_{i,t-1}, \sum_{k \in \mathcal{N}_i} h^e_{k,t-1})_{k=1,...,|E_{t-1}|}. \tag{6}$$

where $\mathcal{N}_i$ is a set of relations with particle i as the receiver.

Next, the model predicts node (particle) positions and updates the latent physics vector:

$$\hat{o}^{vis}_{i,t} = f^{dec}_V \left(h^v_{i,t-1}, \phi_{i,t-1} \right)_{i=1,...,|V_{t-1}|}, \tag{7}$$

$$\xi_{\eta,t}, m_t = f^{dec}_\xi \left(\sum_{\substack{i \\ \mathcal{M}_i = \eta}} h^v_{i,t-1}, \sum_{\substack{i \\ \mathcal{M}_i = \eta}} \phi_{i,t-1}, m_{t-1} \right)_{\eta=1,...,Z}. \tag{8}$$

where f^{dec}_ξ is an LSTM, m_t is its internal cell state at the current step, and $\xi_{\eta,t}$ is the updated physics latent vector for η-th object. At $t = 0$ the LSTM state m_0 is initialized as zero. The physics vector for each object is initialized as Gaussian noise: $\xi_{\eta,0} \sim \mathcal{N}(0, 0.1^2)$ for all η. All other encoder and decoder functions (i.e., f^{enc}_V, f^{dec}_V, f^{enc}_E, and f^{dec}_E) are MLPs.

D. Dynamics Prediction

After the state estimator produces an estimated state $\hat{s}_T = \langle \hat{o}^{vis}_T, \xi_T \rangle$ from the T-step history, our dynamics model predicts into the future to evaluate potential action plans. The dynamics predictor f is constructed similarly to the state estimator g, with two key differences: (i) it does not use tactile observations as input, and (ii) it is conditioned on frozen physics parameters estimated by g. Figure 3 illustrates this process. The forward prediction happens recursively: For a step $t > T$, we construct a graph in the same way as in Section III-C, but excluding tactile observations from the particle attributes, i.e., $c^o_{i,t} = \xi_t$. Then, the dynamics predictor infers the particle positions at the next step $\hat{o}^{vis}_{t+1}$ as formulated in Equations 5-7. The final state prediction is then $\hat{s}_{t+1} = \langle \hat{o}^{vis}_{t+1}, \xi_t \rangle$. Note that the estimated physics parameters are not modified by the dynamics predictor.

Training procedure and objective. We train the state estimator and dynamics predictor jointly end-to-end on trajectories of sequential interaction data containing observations and robot actions. For a training trajectory of length H, the state estimator estimates the first T states, and the dynamics predictor predicts all remaining states. The estimation and prediction are all computed autoregressively. The loss is computed only on visual observations:

$$\mathcal{L} = \frac{1}{H} \sum_{t=0}^{H-1} ||\hat{o}^{vis}_t - o^{vis}_t||^2_2. \tag{9}$$

Previous works [48, 50, 49] use the earth mover's distance (EMD) or chamfer distance (CD) as the training loss, but these provide noisier gradients because EMD requires estimating point-to-point correspondence and CD is prone to outliers. Instead, we use mean squared error (MSE) as the objective, enabled by the point-to-point correspondences from our 3D point tracking (Section III-B). The details of the architecture and training procedure of the state estimator and dynamics predictor are in Appendix A-B.

Note that the learning of the latent physics information is not explicitly supervised. The model is allowed to identify any latent parameters that enhance its ability to accurately estimate the current state and predict future outcomes. We provide an analysis on the learned physics parameters in Section V.

E. Model-Predictive Control

With the learned state estimator and dynamics predictor, we perform planning toward a particular goal by optimizing a cost function on predicted states over potential future actions. Concretely, we use Model Predictive Path Integral (MPPI) to perform this optimization [58].

Planning begins with sampling actions from an initial distribution performing forward prediction with the dynamics models. The cost is then computed on predicted states. Based on the estimated costs, we re-weight the action samples by importance sampling and update the distribution parameters. The process repeats for multiple interactions and we select the optimal execution plan.

Fig. 4: **Hardware overview**. Our experimental platform consists of a Franka Panda arm, two Soft-Bubble sensors, four RealSense D415 RGB-D cameras, and a diverse set of objects.

For computational efficiency, we execute the first K planning steps. While executing the actions, the robot records its tactile readings. After execution, it performs state estimation with the history of observations and re-plans for the next execution. More implementation details on planning can be found in Appendix C.

To summarize this section, a diagram of the entire system workflow including training and test-time deployment is available in Figure 10.

IV. Experimental Setup

A. Physical Setup

We set up our system on a Franka Emika Panda 7-DoF robotic arm. We use four Intel RealSense D415 cameras surrounding the robot and a pair of Soft-Bubble sensors for tactile feedback. We use 3D-printed connectors to attach the Soft-Bubble sensors to the robot. Each Soft-Bubble has a built-in RealSense D405 RGB-D camera. The RGB data are post-processed with an optical flow computation to approximate the force distribution over the bubble surface [22]. Our hardware setup is depicted in Figure 4.

B. Task Description

We demonstrate our method on two tasks where the robot needs to handle objects with unknown physical properties and significant visual occlusion: manipulating a box with an in-hand tool and dense packing.

1) Non-Prehensile Box Pushing: This task focuses on manipulating rigid objects with varying mass distributions using an in-hand rod. The objective is to push a box to a goal pose with the minimum number of pushes. The robot has access to tactile feedback at all steps but only visual observations in between pushes, which corresponds to the real-world feedback loop frequency. The task is much more challenging than usual pushing tasks because (i) the boxes have different dynamics yet the same visual appearance; (ii) the robot has little visual feedback to identify box configurations; and (iii) the in-hand object can rotate and slip due to the highly compliant Soft-Bubble grippers. This is why we emphasize that our task is non-prehensile. This leads to rather complex physics interactions. To achieve effective planning, the robot needs to identify the box's properties from the tactile interaction history and adjust its predictions of the rod and box poses.

(a) Training Object Set (b) Test Object Set

Fig. 5: **Object sets for the packing task.** The test objects are more complex than the training set visually, geometrically, and physically, to showcase the generalizability of our model.

We experiment with four boxes, each equipped with varying calibration weights attached to their inner layers to control their dynamics. We train our model on three of these boxes with identical visual appearances. During evaluation, we test our method on all four boxes including an additional one with a distinct visual appearance and mass distribution.

2) Dense Packing: The goal of this task is to place an additional object in an already densely packed box. Due to heavy occlusions during task execution, the robot does not have access to meaningful visual feedback during robot execution other than the initial frame, but again tactile signals are always observed. To place the object into the occupied box, the robot needs to identify potentially deformable regions with tactile information and make space for the object via pushing actions. The robot needs to avoid inserting into infeasible regions to prevent hardware and object damage. We specify the box that contains the object as the goal and the robot can insert the object at any position as long as it fits inside.

To test the generalizability of learned models, we create train and test object sets (Figure 5). The test objects differ from the training objects in of visual appearance, surface geometry, and physical properties. During evaluation, we consider scenarios with only training objects and those with half or more of objects from the test set.

C. Data Collection

To generate diverse and safe interaction behaviors, we use human teleoperation for data collection. In the *Non-Prehensile Box Pushing* task, for each weight configuration, we gather random interaction data for around 15 minutes. By "random", we refer to the absence of labels typically present in demonstration data. During these interactions, the end-effector approaches the box from various angles and contact locations, yielding diverse outcomes including translation and rotation, as well as relative movements between the in-hand object and

the bubble gripper. The dataset contains approximately 12000 total frames of interaction.

For dense packing, we collect approximately 20 minutes of teleoperated random interaction data with five unique objects, randomizing the initial configurations of the objects at the beginning of each interaction episode. Each episode includes various attempts at packing an object into the box and includes pushing and deforming objects, as well as in-hand slipping of the in-hand object in some trials. The dataset contains approximately 6000 total frames of interaction.

D. Action Space

Though our dynamics model is orthogonal to the action space, suitable action abstractions are important for efficient planning and execution.

1) Non-Prehensile Box Pushing: To reduce the planning horizon and number of optimized parameters, we sample macro-actions during planning, which are defined as a linear push and represented by i, θ, α, where i refers to the box particle index for end effector contact, θ denotes the angle of the push trajectory relative to the x-axis, and α represents the fraction of the distance covered before end effector-box contact along the entire push length. For dynamics prediction, the macro action is decomposed into smaller motions.

2) Dense Packing: As this task involves a large state space, we constrain the action space for planning efficiency. We first identify the outer objects in the box and compute feasible starting positions of actions nudging each object, determined by the geometric center of the object and its approximate radius. Then we sample straight-line push actions of varying lengths from each contact point towards the respective object centers. Similarly, the long push action is divided into small movements for dynamics prediction.

E. Planning Cost Functions

1) Non-Prehensile Box Pushing: We specify the goal state as a point cloud and use MSE as the cost function.

2) Dense Packing: We specify a 2D goal region by uniformly sampling points in the area underneath the tray. We use a cost function that (i) penalizes the objects in the box from being pushed out of the boundary, (ii) encourages the robot to make space for placing the in-hand object by maximizing the distance from target to object points, and (iii) rewards exploring different starting action positions. Mathematically, the loss function is

$$\mathcal{J}(\hat{o}_t, o_g, a_t) = \sum_{x \in \hat{o}_t} \min_{y \in o_g} ||x - y||_2 - \sum_{y \in o_g} \min_{x \in \hat{o}_t} ||x - y||_2 + r * \mathbb{1}_{[||a_{0,t}||_2 = 0]}, \quad (10)$$

where $\hat{o}_t$ is the predicted object particles in the box, o_g is the target point cloud, $||a_{0,t}||_2$ is the size of the first action, which is zero if it does not plan to switch to a different contact row, r is a negative constant, and $\mathbb{1}$ is an indicator function.

V. Experiments

In this section, we investigate the following questions.

i. Does integrating tactile sensing information from prior interactions improve future prediction accuracy?
ii. Do the latent representations learned by tactile dynamics models discover meaningful properties such as the physical properties of objects?
iii. Does our tactile-informed model-predictive control framework enable robots to solve tasks involving objects of unknown physical properties?

We first introduce our baselines and then present empirical results in the subsequent subsections.

A. Baselines and Prior Methods

We compare our approach against three prior methods and baselines, including ablated versions of our model, previous work on dynamics learning, and a physics-based simulator:

i. **RoboPack (no tactile)**: To study the effects of using tactile sensing in state estimation and dynamics prediction, we evaluate this ablation of our method, which zeroes out tactile input to the model.
ii. **RoboCook + tactile**: This approach differs from ours in that it treats the observations, i.e., visual and tactile observations $\langle o^{vis}, o^{tact} \rangle$, directly as the state representation, whereas RoboPack assumes partial observability of the underlying state and performs explicit state estimation. This can be viewed as an adaptation of previous work [29, 48, 50, 49] to include an additional tactile observation component. With this baseline, we seek to study different state representations and our strategy of separating state estimation from dynamics prediction.
iii. **Physics-based simulator**: We also compare our method to using a physics-based simulator for dynamics prediction after performing system identification of explicit physical parameters. We use heuristics to convert observed point clouds into body positions and orientations in the 2D physics simulator Pymunk [3]. For system identification, we estimate the mass, center of gravity, and friction parameters from the initial and current visual observations with covariance matrix adaptation [14].

The considered methods, including our approach, share some conceptual components with prior offline model-based reinforcement learning (RL) methods (Section II-B), although with very different concrete instantiations. Each method either learns the full environment dynamics, or in the case of *Physics-based simulator*, performs system identification from a static dataset. All compared methods use the dynamics models to perform model-predictive control via sampling-based planning. Specifically, *RoboPack (no tactile)* can be framed as a model-based RL method (e.g., [59, 11, 9]) that uses only sparse visual observations for model learning. On the other hand, *RoboCook + tactile* treats visual and tactile observations as the state, overlooking the partially observable nature of the task. Our upcoming results demonstrate that our integration of multi-modal perception and physical parameter estimation leads to superior performance in challenging task domains.

Task	Method	MSE *1e-3 ↓	EMD *1e-2 ↓	CD *1e-2 ↓
Box Pushing	**RoboPack**	**1.48 ± 0.14**	**2.97 ± 0.14**	**3.46 ± 0.13**
	RoboPack (no tactile)	1.75 ± 0.15	3.34 ± 0.15	3.80 ± 0.13
	RoboCook + tactile	2.11 ± 0.17	4.32 ± 0.16	5.40 ± 0.16
	Physics-based sim.	2.65 ± 0.18	4.11 ± 0.17	4.57 ± 0.16
Dense Packing	**RoboPack**	**0.070 ± 0.005**	**1.12 ± 0.036**	**2.01 ± 0.050**
	RoboPack (no tactile)	0.088 ± 0.006	1.18 ± 0.043	2.04 ± 0.058

TABLE I: **Long-horizon dynamics prediction results on the two task datasets.** Errors represent a 95% confidence interval.

B. Evaluating Dynamics Prediction

Results are summarized in Table I. On the *Non-Prehensile Box Pushing* task, RoboPack is significantly better than alternative methods in all metrics. Compared to RoboPack (no tactile), RoboPack can better estimate the mass distribution of the boxes, which is crucial in predicting the translation and rotation accurately. In contrast, when using tactile and visual observations directly as the state representation (RoboCook + tactile), the performance is even worse than RoboPack without tactile information. We hypothesize that this is because the model has very high errors in learning to predict future tactile readings because of the intricate local interactions between the Soft-Bubble grippers and the object. The difficulty in learning to predict tactile reading may distract the model from learning to predict visual observations accurately.

Comparing RoboPack to a physics-based simulator baseline, we find that the simulator performs poorly on dynamics prediction for a few potential reasons, including (i) limited visual feedback for performing system ID, and (ii) the simulator's parameter space may not capture the full range of real-world dynamics given the complex interactions between the compliant bubble and in-hand tool and rotating tool and the box. To illustrate the difference in model predictions, qualitative results are presented in Figure 6.

For the *Dense Packing* task, our model outperforms the best baseline on the pushing task, RoboPack (no tactile). We note that in this task, object movements are minimal and object deformation is the major source of particle motions. Metrics such as EMD and CD that emphasize global shape and distribution but are insensitive to subtle positional changes cannot differentiate the two methods in a statistically significant way. However, for the MSE loss, which measures prediction error for every point, RoboPack is significantly better than the baseline, indicating its ability to capture fine details of object deformation. This subtle performance difference between the two methods in dynamics prediction turns out to have a significant effect on real-world planning (Section V-D).

C. Analysis of Learned Physics Parameters

In this subsection, we seek to provide some quantitative and qualitative analyses of the latent representation learned by the state estimator. As it gives more direct control of object properties, we use our dataset collected for the *Non-Prehensile Box Pushing* task for the analysis.

To understand if the representation contains information about box types, we first attempt to train a linear classifier to test if there the features learned for different boxes are linearly separable in the latent space. We test the state estimator on 145-step trajectories in the testing data, which typically involves three to five pushes on the box. The classification accuracy of physics parameters ξ_t as more and more interaction information is processed is shown in Figure 7. It can be observed that as history information accumulates, the latent physics vectors become more indicative of the box type. In particular, the state estimator can extract considerable information in the first 20 steps, which is approximately the average number of steps it takes to complete an initial push. Furthermore, note that the state estimator only observes a history of no more than 25 steps during training, but it can generalize to sequences four times longer in this case.

To qualitatively inspect the learned representations, we perform principal component analysis, reducing the learned latent vectors from $\mathcal{R}^{16}$ to $\mathcal{R}^2$. Figure 7 shows the low-dimensional embeddings as the number of interaction time steps incorporated into the latents grows. We can see that as time progresses, the estimated latents become increasingly separated into clusters based on the physical properties (i.e., mass distributions in this case) of the manipulated object. The separation increases the most between $t = 1$ and $t = 20$, which is consistent with our observation in Figure 7 that longer

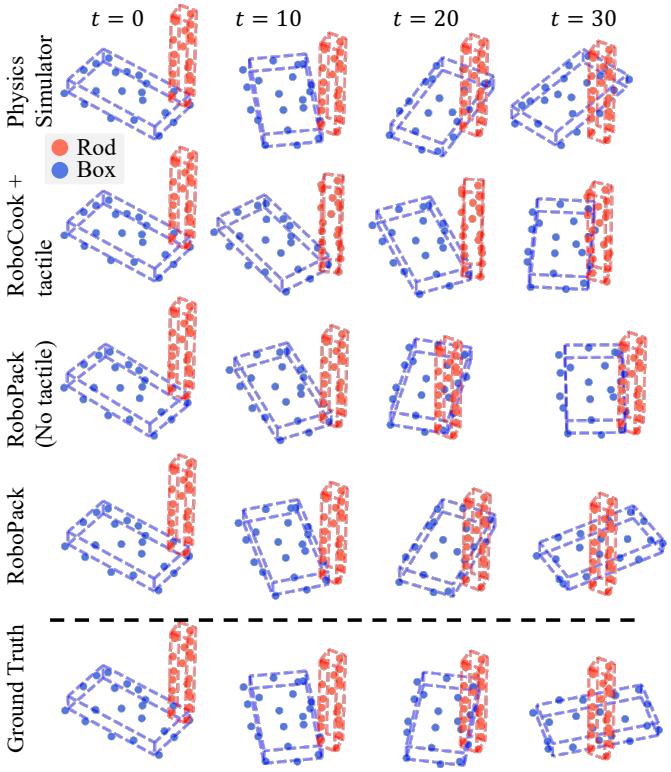


Fig. 6: **Qualitative results on dynamics prediction.** Predictions made by our model compared to baseline methods in the *Non-prehensile Box Pushing* task. Red dots indicate the rod and blue dots represent the box. Our method closely approximates the ground truth and outperforms all the baseline methods. For visualization, the blue dashed lines outline box contours and red dashed lines show in-hand object contours.

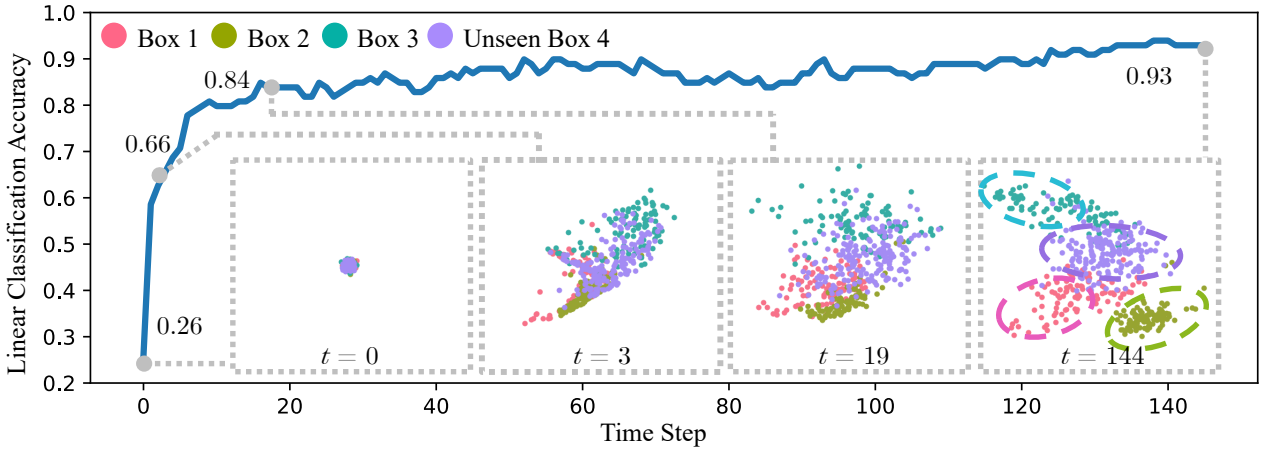


Fig. 7: **Analysis of learned physics parameters.** We assess our state estimator across 145-step trajectories and record the estimated physics parameters at each step. PCA visualizations at four distinct timesteps show that the physics parameters gradually form clusters by box type. We also employ a linear classifier trained on these parameters to accurately predict box types to demonstrate these clusters' linear separability. The classifier's improving accuracy across timesteps underscores the state estimator's proficiency in extracting and integrating box-specific information from the tactile observation history.

Method	Metric	 Box 1	 Box 2	 Box 3	 Box 4 (unseen)	Aggregated
RoboPack	MSE ↓	0.0164 ± 0.004	0.0165 ± 0.004	0.0137 ± 0.003	0.0156 ± 0.001	**0.0156 ± 0.002**
	# Pushes ↓	5.0 ± 1.20	5.40 ± 1.49	4.8 ± 1.24	6.0 ± 1.10	**5.3 ± 0.64**
	Success Rate ↑	4 / 5	4 / 5	4 / 5	4 / 5	**16 / 20**
RoboPack (no tactile)	MSE ↓	0.0612 ± 0.027	0.0141 ± 0.003	0.0250 ± 0.001	0.0264 ± 0.005	0.0317 ± 0.008
	# Pushes ↓	8.2 ± 0.99	5.0 ± 2.82	10.0 ± 0	8.2 ± 1.07	7.85 ± 0.63
	Success Rate ↑	2 / 5	4 / 5	0 / 5	2 / 5	8 / 20
RoboCook + tactile	MSE ↓	0.0459 ± 0.018	0.0607 ± 0.022	0.0418 ± 0.009	0.0438 ± 0.017	0.0480 ± 0.009
	# Pushes ↓	8.2 ± 1.21	7.4 ± 1.73	9.2 ± 0.72	8.8 ± 1.07	8.4 ± 0.64
	Success Rate ↑	2 / 5	2 / 5	1 / 5	1 / 5	6 / 20
Physics-based simulator	MSE ↓	0.0237 ± 0.004	0.0184 ± 0.003	0.0273 ± 0.012	0.0220 ± 0.004	0.0230 ± 0.003
	# Pushes ↓	8.4 ± 0.92	6.0 ± 0.18	7.4 ± 1.19	7.4 ± 1.49	7.3 ± 0.71
	Success Rate ↑	2 / 5	3 / 5	3 / 5	2 / 5	10 / 20

TABLE II: **Per-configuration results on the non-prehensile box pushing task**. We report the minimum error to goal across 10 plan executions per trial, trial success rates, and number of execution steps to solve the task. A trial is labeled as a success if it achieves an error lower than 0.02 for point-wise MSE within 10 pushes.

histories than a certain threshold yield marginal returns.

Collectively, the results indicate that our state estimator indeed learns information related to physical properties based on interaction histories.

D. Benchmarking Real-World Planning Performance

Next, we evaluate the performance of our approach in solving real-world robotic planning tasks.

For *Non-Prehensile Box Pushing*, we present quantitative results in Figure 9 and Table II. We can see that our method both achieves lower final error as measured by point-wise MSE (Table II) and makes progress toward goals more quickly (Figure 9) than other methods. The gap in performance between our model and RoboPack (no tactile) demonstrates the benefits of using tactile sensing in this task. While the physics-based simulator achieves the strongest performance of the baselines, it is not able to achieve as precise control as our method, taking more pushes to finish the task yet ending with higher MSE loss. We hypothesize this is because it can only infer dynamics of limited complexity via properties such as friction or mass center/moment. It also requires significant manual designs to construct the simulation for each task. Finally, RoboCook + tactile has the poorest control performance, consistent with its high dynamics prediction error on the test set. We hypothesize that the poor performance of this method is due to the difficulty of learning to predict future tactile observations, which are high-dimensional and sensitive to precise contact details.

For the *Dense Packing* task, we would ideally compare our method against the baseline with the best results on non-prehensile box pushing: the physics-based simulator. However, this is impractical for this task, because it is infeasible to obtain corresponding object models for the diverse and complex objects in this task or to estimate objects' explicit physics parameters without visual feedback. Thus, we compare against

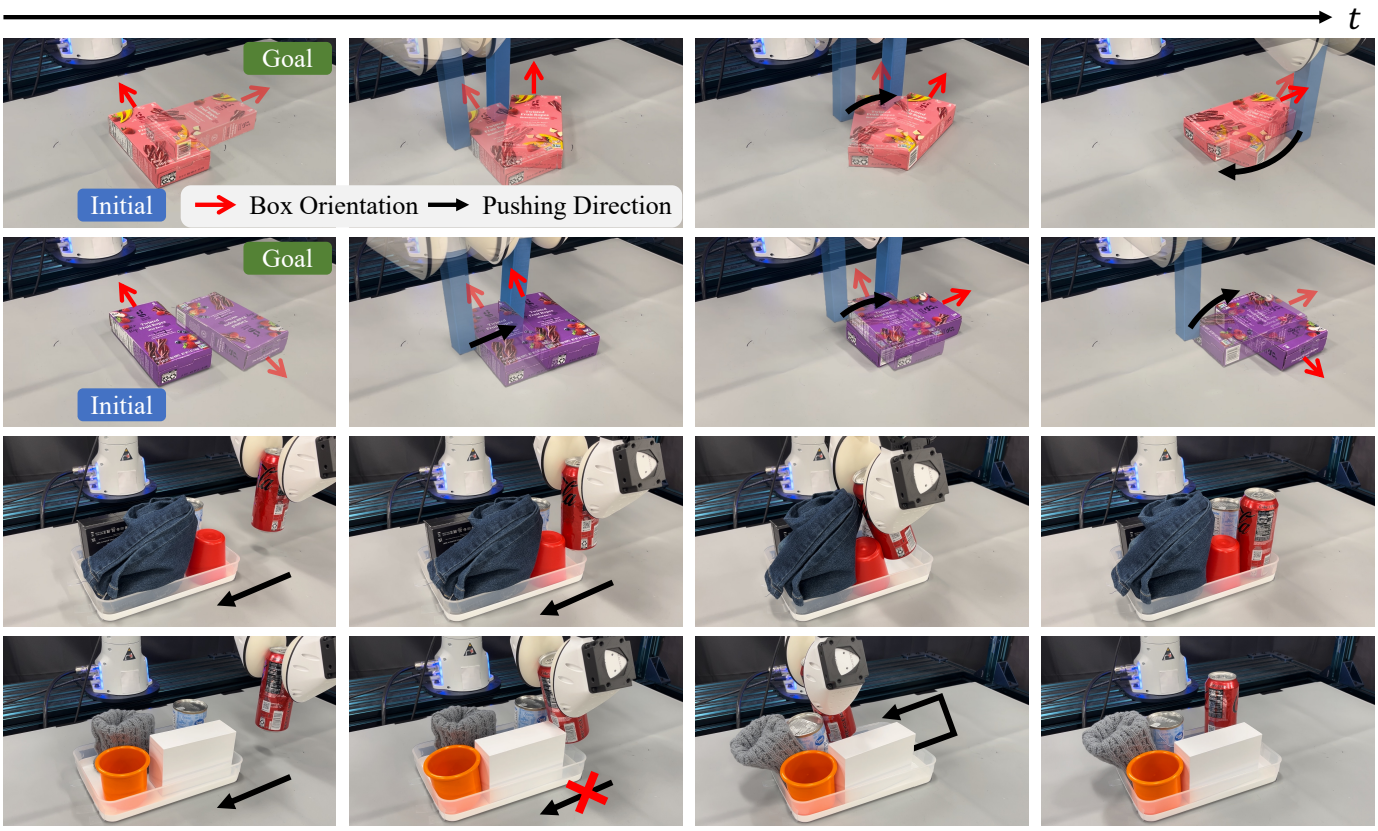


Fig. 8: **Non-prehensile box pushing and dense packing.** In the *Non-prehensile Box Pushing* task, we demonstrate that our robot can push a box with unknown mass distribution from a starting pose to a target pose. We show that our method can generalize to unseen targets and box configurations in the first two rows. In the *Dense Packing* task, we demonstrate that RoboPack effectively identifies feasible insertion rows in a tray, minimizing excessive force to prevent hardware damage for incorrect contact locations while taking pushing actions decisively at correct contact points for efficient task completion. The last two rows illustrate that our method can adapt to objects with different visual appearances, shapes, and deformability.

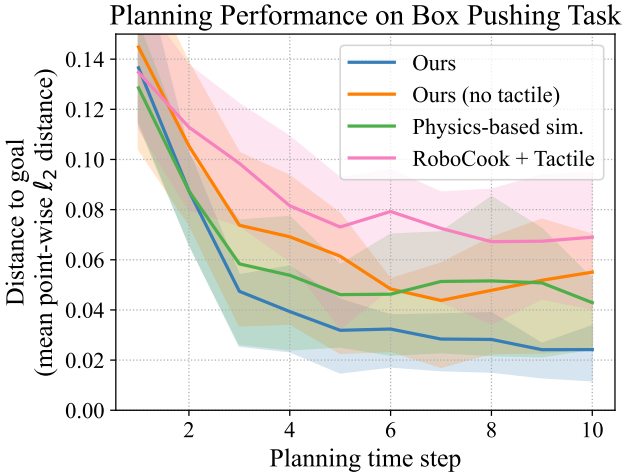


Fig. 9: **Real-world planning performance on the box pushing task.** Shaded regions denote the first and third quantiles. Note that different methods generally perform well on easier cases, leading to overlap between shadow regions. Our method has stable performance even for hard ones: its 75-percentile error is lower than the mean error of all other methods.

Method	Seen Objects	Unseen Objects
RoboPack	**12/15**	**10/15**
RoboPack (no tactile)	6/15	5/15

TABLE III: **Success rates on the dense packing task.** In the Unseen Objects setting, half or more of the objects in the tray are unseen. A trial is considered successful if the robot correctly determines feasible insertion locations and creates enough space (through deformation) to pack the object. The robot automatically attempts to pack the object when its end effector y-position exceeds a given threshold.

the best among the remaining baselines instead, i.e., RoboPack (no tactile). We test on scenarios containing only training objects (*Seen Objects*) as well as scenarios where half or more of the objects are from the test set (*Unseen Objects*). Results on both settings, shown in Table III, indicate that our method is more effective in identifying objects that are deformable or pushable, which consequently enables the robot to insert the object at feasible locations. Examples of our experiments are illustrated in Figure 8. Despite our method having only seen rectangular boxes and plastic bags in the training set,

it can generalize to objects with different visual appearances, geometries, and physical properties, such as the cups, cloth, and hat in the examples.

VI. Discussion

We presented RoboPack, a framework for learning tactile-informed dynamics models for manipulating objects in multi-object scenes with varied physical properties. By integrating information from prior interactions from a compliant visual tactile sensor, our method adaptively updates estimated latent physics parameters, resulting in improved physical prediction and downstream planning performance on two challenging manipulation tasks, *Non-Prehensile Box Pushing* and *Dense Packing*. We hope that this is a step towards robots that can seamlessly integrate information with multiple modalities from their environments to guide their decision-making.

In this paper we demonstrated our approach on two specific tasks, but our framework is generally applicable to robotic manipulation tasks using visual and tactile perception. To extend it to other tasks, one needs to adapt the cost function and planning module to the task setup, but the perception, state estimation, and dynamics prediction components are general and task-agnostic. For future work, we seek to develop dynamics models that can efficiently process higher-fidelity particles to model fine-grained object deformations. Integrating alternative trajectory optimization methods with our learned differentiable neural dynamics models is another promising direction. Finally, incorporating additional physics priors into the dynamics model could further improve generalization.

Acknowledgments

We thank the Toyota Research Institute for lending the SoftBubble sensor hardware. This work is in part supported by the Toyota Research Institute (TRI), the Stanford Human-Centered AI Institute (HAI), and Amazon. S.T. is supported by NSF GRFP Grant No. DGE-1656518. This work is also in part supported by an A*STAR CRF award to C.T.

References

[1] Bo Ai, Wei Gao, Vinay, and David Hsu. Deep visual navigation under partial observability. In *International Conference on Robotics and Automation (ICRA)*, 2022.

[2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and koray kavukcuoglu. Interaction Networks for Learning about Objects, Relations and Physics. In *Advances in Neural Information Processing Systems*, 2016.

[3] Victor Blomqvist. Pymunk, 2023. URL https://pymunk.org.

[4] Roberto Calandra, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H. Adelson, and Sergey Levine. More Than a Feeling: Learning to Grasp and Regrasp Using Vision and Touch. *IEEE Robotics and Automation Letters*, 2018.

[5] Peng Chang and Taşkın Padır. Model-Based Manipulation of Linear Flexible Objects with Visual Curvature Feedback. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2020.

[6] Haonan Chen, Yilong Niu, Kaiwen Hong, Shuijing Liu, Yixuan Wang, Yunzhu Li, and Katherine Rose Driggs-Campbell. Predicting Object Interactions with Behavior Primitives: An Application in Stowing Tasks. In *Conference on Robot Learning*, 2023.

[7] Ravinder S. Dahiya, Giorgio Metta, Maurizio Valle, and Giulio Sandini. Tactile Sensing—From Humans to Humanoids. *IEEE Transactions on Robotics*, 2010.

[8] Elliott Donlon, Siyuan Dong, Melody Liu, Jianhua Li, Edward Adelson, and Alberto Rodriguez. GelSlim: A High-Resolution, Compact, Robust, and Calibrated Tactile-sensing Finger. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[9] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex X. Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv:1812.00568*, 2018.

[10] Ruohan Gao, Yiming Dou, Hao Li, Tanmay Agarwal, Jeannette Bohg, Yunzhu Li, Li Fei-Fei, and Jiajun Wu. The Object Folder Benchmark : Multisensory Learning with Neural and Real Objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[11] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, 2018.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[13] Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *International Conference on Machine Learning*, 2019.

[14] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv:1604.00772*, 2016.

[15] Haoyang He. A survey on offline model-based reinforcement learning. *arXiv:2305.03360*, 2023.

[16] Todd Hester and Peter Stone. TEXPLORE: real-time sample-efficient reinforcement learning for robots. *Machine Learning*, 2013.

[17] Philipp Holl, Nils Thuerey, and Vladlen Koltun. Learning to Control PDEs with Differentiable Physics. In *International Conference on Learning Representations*, 2019.

[18] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

[19] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn,

Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv:2104.08212*, 2021.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations*, 2015.

[21] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[22] Naveen Kuppuswamy, Alex Alspach, Avinash Uttamchandani, Sam Creasey, Takuya Ikeda, and Russ Tedrake. Soft-bubble grippers for robust and perceptive manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[23] Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[24] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning Plannable Representations with Causal InfoGAN. In *Advances in Neural Information Processing Systems*, 2018.

[25] Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg Kammerer, Dinesh Jayaraman, and Roberto Calandra. DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor With Application to In-Hand Manipulation. *IEEE Robotics and Automation Letters*, 2020.

[26] Christian Legaard, Thomas Schranz, Gerald Schweiger, Ján Drgoňa, Basak Falay, Cláudio Gomes, Alexandros Iosifidis, Mahdi Abkar, and Peter Larsen. Constructing Neural Network Based Models for Simulating Dynamical Systems. *ACM Computing Surveys*, 2023.

[27] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 2016.

[28] Hao Li, Yizhi Zhang, Junzhe Zhu, Shaoxiong Wang, Michelle A. Lee, Huazhe Xu, Edward Adelson, Li Fei-Fei, Ruohan Gao, and Jiajun Wu. See, Hear, and Feel: Smart Sensory Fusion for Robotic Manipulation. In *Conference on Robot Learning*, 2023.

[29] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2019.

[30] Yunzhu Li, Antonio Torralba, Animashree Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos. In *Neural Information Processing Systems*, 2020.

[31] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3D Neural Scene Representations for Visuomotor Control. In *Conference on Robot Learning*, 2021.

[32] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[33] Changyi Lin, Han Zhang, Jikai Xu, Lei Wu, and Huazhe Xu. 9DTact: A compact vision-based tactile sensor for accurate 3D shape reconstruction and generalizable 6D force estimation. *IEEE Robotics and Automation Letters*, 2023.

[34] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning Visible Connectivity Dynamics for Cloth Smoothing. In *Conference on Robot Learning*, 2022.

[35] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. *arXiv:2303.05499*, 2023.

[36] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees. In *International Conference on Learning Representations*, 2018.

[37] Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. In *Conference on Robot Learning*, 2020.

[38] Carolyn Matl and Ruzena Bajcsy. Deformable Elasto-Plastic Object Shaping using an Elastic Hand and Model-Based Reinforcement Learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[39] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *International Conference on Learning Representations*, 2021.

[40] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[41] J. Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradSim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*, 2020.

[42] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and

Vikash Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. In *Conference on Robot Learning*, 2020.

[43] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *arXiv:2304.07193*, 2023.

[44] Haozhi Qi, Brent Yi, Sudharshan Suresh, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. General In-hand Object Rotation with Vision and Touch. In *Conference on Robot Learning*, 2023.

[45] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In Ali Jadbabaie, John Lygeros, George J. Pappas, Pablo A. Parrilo, Benjamin Recht, Claire J. Tomlin, and Melanie N. Zeilinger, editors, *Conference on Learning for Dynamics and Control*, 2021.

[46] Marc Rigter, Bruno Lacerda, and Nick Hawes. RAMBO-RL: robust adversarial model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

[47] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to Simulate Complex Physics with Graph Networks. In *International Conference on Machine Learning*, 2020.

[48] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. RoboCraft: Learning to See, Simulate, and Shape Elasto-Plastic Objects with Graph Networks. In *Robotics: Science and Systems*, 2022.

[49] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools. In *Conference on Robot Learning*, 2023.

[50] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. RoboCraft: Learning to see, simulate, and shape elasto-plastic objects in 3D with graph networks. *The International Journal of Robotics Research*, 2023.

[51] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *Advances in Neural Information Processing Systems*, 2010.

[52] H.J. Terry Suh, Naveen Kuppuswamy, Tao Pang, Paul Mitiguy, Alex Alspach, and Russ Tedrake. SEED: Series Elastic End Effectors in 6D for Visuotactile Tool Use. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[53] Sudharshan Suresh, Haozhi Qi, Tingfan Wu, Taosha Fan, Luis Pineda, Mike Lambeta, Jitendra Malik, Mrinal Kalakrishnan, Roberto Calandra, Michael Kaess, Joseph Ortiz, and Mustafa Mukadam. Neural feels with neural fields: Visuo-tactile perception for in-hand manipulation. *arXiv:2312.1346*, 2023.

[54] Stephen Tian, Frederik Ebert, Dinesh Jayaraman, Mayur Mudigonda, Chelsea Finn, Roberto Calandra, and Sergey Levine. Manipulation by feel: Touch-based control with deep predictive models. In *International Conference on Robotics and Automation (ICRA)*, 2019.

[55] Yixuan Wang, Yunzhu Li, Katherine Driggs-Campbell, Li Fei-Fei, and Jiajun Wu. Dynamic-Resolution Model Learning for Object Pile Manipulation. In *Robotics: Science and Systems*, 2023.

[56] Yixuan Wang, Zhuoran Li, Mingtong Zhang, Katherine Driggs-Campbell, Jiajun Wu, Li Fei-Fei, and Yunzhu Li. D^3fields: Dynamic 3d descriptor fields for zero-shot generalizable robotic manipulation. *arXiv:2309.16118*, 2023.

[57] R. Weinstein, J. Teran, and R. Fedkiw. Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Transactions on Visualization and Computer Graphics*, 2006.

[58] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *International Conference on Robotics and Automation (ICRA)*, 2016.

[59] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, 2022.

[60] Wenzhen Yuan, Siyuan Dong, and Edward H. Adelson. GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force. *Sensors*, 2017.

[61] Ying Yuan, Haichuan Che, Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Kang-Won Lee, Yi Wu, Soo-Chul Lim, and Xiaolong Wang. Robot Synesthesia: In-Hand Manipulation with Visuotactile Sensing. *arXiv:2312.01853*, 2023.

[62] Marvin Zhang, Sharad Vikram, Laura M. Smith, Pieter Abbeel, Matthew J. Johnson, and Sergey Levine. SOLAR: deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, 2019.

[63] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. VIOLA: Imitation learning for vision-based manipulation with object proposal priors. In *Conference on Robot Learning*, 2022.

APPENDIX A
MODEL ARCHITECTURE AND TRAINING

A. Tactile Autoencoder

Both the encoder and decoder are two-layer MLPs with hidden dimension 32 and ReLU activations. The encoder maps the raw point-wise tactile signal to latent space, then the decoder maps it back to the original dimension. The autoencoder is trained with MSE loss using the following hyper-parameters:

Hyperparameter	Value
Learning rate	5e-4
Optimizer	Adam [20]
Batch size	32
Latent space dimension	5

TABLE IV: **Hyperparameters for auto-encoder training.**

B. State Estimator and Dynamics Predictor

We use the same hyperparameters to train dynamics models for the nonprehensile box pushing and dense packing tasks, which are shown in Table V For graph construction, we connect any points within a radius of 0.15. We train the state estimator and dynamics model jointly, using sequences of length 25. To prevent the model from overfitting to a specific history length, which could vary at deployment time, we use the first k steps in a sequence as the history, $k \sim \text{Uniform}(0, 24)$. To stabilize training, we restrict the magnitude of the rotation component of predicted rigid transformations for a single step to be at most 30 degrees, which is much larger than any rotation that occurs in our datasets. Model training converges within 25 and 8 hours on the two tasks respectively with one NVIDIA RTX A5000 GPU.

For baselines RoboPack (no tactile) and RoboCook + tactile, we performed a hyper-parameter sweep and the optimal training parameters are the same as RoboPack described above.

Hyperparameter	Value
Learning rate	5e-4
Optimizer	Adam [20]
Batch size	4
Graph construction criteria	Radius
Graph connection radius	0.15m
Training sequence length	25 steps
Training history length	15 steps
# graph points per object	20
# graph points per tactile sensor	20
Node encoder MLP width	150
Node encoder MLP layers	3
Edge encoder MLP width	150
Edge encoder MLP layers	3
Edge effect MLP width	150
Edge effect MLP layers	3
Edge propagation steps	3
Latent physics vector size (dim(ξ))	16
Tactile encoding dimension (per point in o^{tact})	5

TABLE V: **Hyperparameters for dynamics model training.** We use the same hyperparameters for the nonprehensile box pushing and dense packing tasks.

APPENDIX B
HARDWARE SETUP

The hardware setup is depicted in Figure 4 in the main text.

Robot. We use a Franka Emika Panda robot arm, controlled using the Deoxys open-source controller library [63]. In our experiments, we use the `OSC_POSITION` and `OSC_YAW` controllers provided by the Deoxys library.

Sensors. We attach the Soft-Bubble sensors to the Franka Panda gripper using custom-designed 3D-printed adapters. We inflate both Soft-Bubble sensors to a width of 45mm measured from the largest distance sensor frame to the rubber sensor surface. While there can be slight variations in the exact amount of air in the sensor due to measurement error, we do not find this to be a significant cause of domain shift for learned models, likely because the signals that are used as input to our model are largely calculated using differences between the current reading and a reference frame captured when the gripper does not make contact with any object that we reset upon each inflation. While we contribute a novel method for integrating tactile information into the particle-based scene representation, the computation of raw tactile features described in Section III-B2 are computed by the Soft-Bubble sensor API [22] and is not part of our contribution.

APPENDIX C
PLANNING IMPLEMENTATION DETAILS

We provide hyperparameters for the MPPI optimizer that is used for planning with learned dynamics models in Table VI. We use the same planning hyperparameters for baselines as we do our method.

Hyperparameter	Box pushing	Dense packing
History length	22	25
Action sampler temporal correlation β*	0.2	N/A
MPPI # action samples	400	150
MPPI action horizon	20	80
MPPI # iterations	2	1
MPPI scaling temperature γ*	100	N/A
# steps executed before replanning K	20	45

TABLE VI: **Hyperparameters for real world planning experiments.** For the parameters denoted by *, we use the notation from Nagabandi et al. [42]. As introduced in Section III-E, K refers to the number of steps in the best plan found that is actually executed on the real robot before replanning. For box pushing it is the entire plan, while for dense packing it is 45 out of 80 steps.

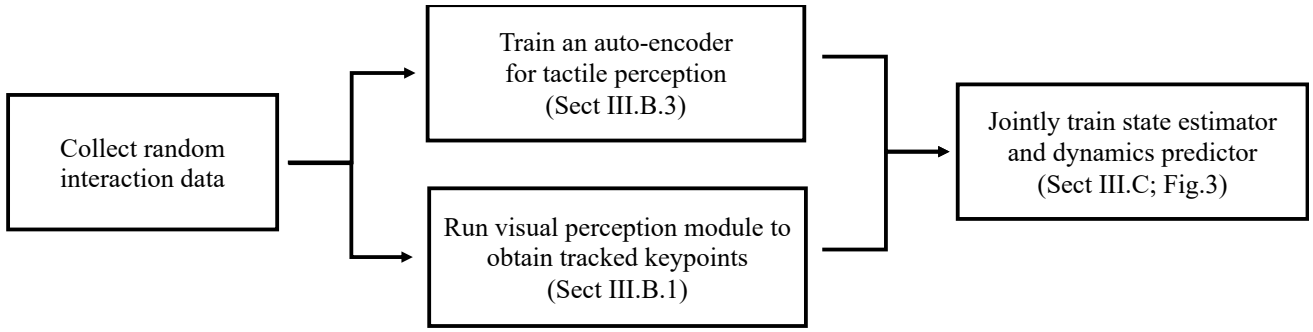
APPENDIX D
SYSTEM WORKFLOW

To present the offline training and online planning processes more clearly, a system diagram is provided in Figure 10.

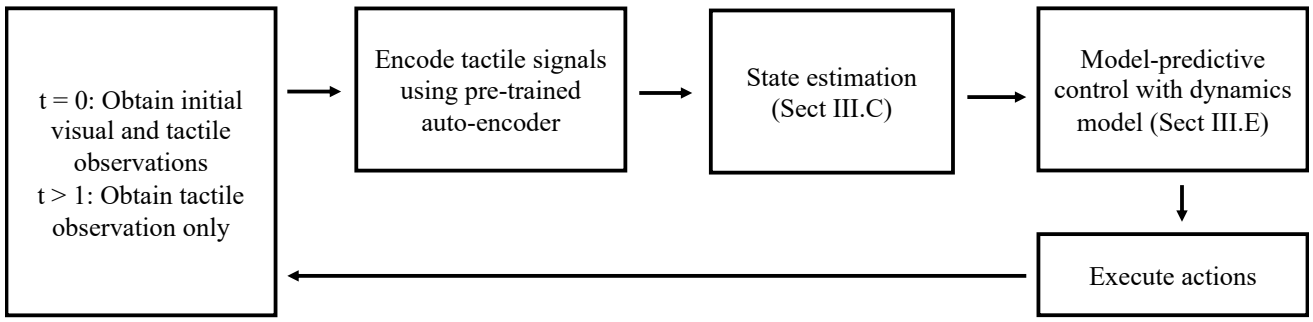
APPENDIX E
EXPERIMENTS

A. Box Configurations

For the non-prehensile box pushing task, we use boxes that have the same geometry different weight configurations to test



(a) Offline training



(b) Online planning

Fig. 10: **The complete workflow of the RoboPack system.** There are two main stages of the system: (a) offline training and (b) online planning.

the ability of each model to adapt its prediction based on the physical characteristics of each scenario. Specifically, we empty cardboard boxes of dimensions $18 \times 9.5 \times 3.8$ cm, and then add metal weights in the following configurations:

- **Box 1**: Two 100g weights placed at opposing corners of the box.
- **Box 2**: One 200g weight placed at the geometric center of the box.
- **Box 3**: No additional weight added.
- **Box 4 (unseen during training)**: This is the original unmodified box, which contains roughly uniformly distributed food items. The items are not affixed to the inner sides of the box, and there could be relative movement between the box and its contents if force is applied.

B. Qualitative Results on Planning

Additional qualitative results on non-prehensile box pushing and dense packing are presented in Figure 11 and Figure 12 respectively. Please additionally see our supplementary video for video examples of planning executions.

C. Physics-Based Simulator Baseline

Here we provide additional details about the physics-based simulator baseline used for the box pushing experiments in Section V.

First, we construct a 2D version of the task in the open source Pymunk simulator [3] that emulates a top-down view of the real scene. The simulated scene contains replicas of the rod and box produced by measuring the dimensions of the real versions of those objects.

Then, given two visual observations o^{vis}_{init} and o^{vis}_{final}(tracked points for each object) and a sequence of actions $\vec{a}$ taken by the robot, we perform system identification to optimize simulated parameters to fit the real system. Note that our method also uses only two visual observations from the history, but also can use tactile information. Because tactile simulation is not available, the baseline has access to just visual observations. To convert tracked points from real observations into simulator states, we project all points into 2D by truncating the z dimension, and then for each object we compute the object center with the spatial mean of points and the 2D rotation by finding the first two principal components of the 2D points with PCA. Thus the visual observations are converted into tuples $(pos^{rod}_{init}, rot^{rod}_{init}), (pos^{box}_{init}, rot^{box}_{init})$, $(pos^{rod}_{final}, rot^{rod}_{final}), (pos^{box}_{final}, rot^{box}_{final})$.

We optimize a vector of parameters $\vec{\mu} \in \mathbb{R}^5$, detailed in Table VII. We de-normalize values from $\vec{mu}$ to the actual system parameters and clamp them to prevent unrealistic values based on the minimum and maximum values shown. The initial standard deviation for optimization is $\sigma = 0.3$, which we found to work well empirically. The objective function is

$$\mathcal{L}(\vec{\mu}) = \|(pos^{box}_{final}, rot^{box}_{final}) - \textsc{sim}_{\vec{\mu}}(pos_{init}, rot_{init}, \vec{a})\|_2.$$

where $\textsc{sim}_{\vec{\mu}}(pos, rot, \vec{a})$ represents the box position and rotation after running a simulated trajectory with actions $\vec{a}$ in the Pymunk simulator starting from box and rod positions pos and rotations rot with simulator parameters set to $\vec{\mu}$.

We optimize the objective using CMA-ES, a gradient free

Hyperparameter	Initial value	Min	Max	Optimization space μ to sim. param p transform
Box mass	10	0.001	N/A	$p = 10(\mu + 1)$
Box friction	0.5	0.0001	N/A	$p = 0.5(\mu + 1)$
Moment of inertia	34520.83	10	N/A	$p = 35420.833(\mu + 1)$
Center of gravity x	0	-42.5	42.5	$p = 42.5\mu$
Center of gravity y	0	-90	90	$p = 90\mu$

TABLE VII: **Parameters optimized during system identification for the physics-based simulator baseline.** Initial values and scales are set such that when the parameters in the optimization space are $\vec{\mu} = 0$, the actual values in the physics simulator $\vec{p}$ are sensible defaults (see initial value column). Note for center of gravity, $(0, 0)$ refers to the geometric center of the object.

optimizer, using the implementation from https://github.com/CyberAgentAILab/cmaes. Parameters are initialized to have the center of mass at the center of the object uniformly distributed mass, and reasonable friction and mass defaults. We use a population size of 8 based on the implementation-suggested default of $4 + \lfloor 3 * \log(ndim) \rfloor$ and optimize for 100 generations.

Finally, we use the optimized set of parameters to perform forward prediction. After forward prediction, we convert the sequence of simulated 2D object positions into a sequence of pointcloud predictions by estimating a rotation matrix and translation (in 2D) and applying them to the 3D pointcloud for the initially provided observation. The z values (height) of all particles are assumed to be fixed at their initial values throughout the prediction.

APPENDIX F
TRACKING MODULE DETAILS

As described in Section III-B, after sampling initial sets of points for each object $\vec{p}_{init}$, we formulate point tracking as optimization for the points at each step $\vec{p}$. Specifically, the new points are computed as a 3D transformation of the points output at the previous step, represented by a rigid rotation $R \in \mathbb{R}^3$, translation $T \in \mathbb{R}^3$ and optional per-axis shearing $S \in \mathbb{R}^3$. The transform is a composition of rotation by R, scaling by S, and translation by T in that order. We abuse notation to sometimes use $\vec{p}$ for ease of reading, but $\vec{p}$ is a function of the actual optimized parameters R, S, T. Thus the optimization objective has the following loss terms:

1) **Distance to surface.**

$$\mathcal{L}_{depth}(\vec{p}) = \frac{1}{|\vec{p}|} \sum_{p \in \vec{p}} \max(0, depth_{interp(p)} - depth_{proj(\vec{p})})$$

where $depth_{interp(p)}$ is the depth estimation from interpolating information from multi-view depth observations, and $depth_{proj(\vec{p})}$ is the expected depth at each point when projected into each camera frame.

2) **Semantic alignment.**

$$\mathcal{L}_{align}(\vec{p}) = \frac{1}{|\vec{p}|} \sum_{p \in \vec{p}} \min(\|dinov2(p_{init}) - dinov2(p)\|_2, 30)$$

where $dinov2(p)$ represents the multi-view interpolated DinoV2 feature at the 3D point represented by p, and again p_{init} is the position of the point in the first frame (not necessarily immediately prior frame) of tracking.

Hyperparameter	Box pushing	Dense packing
Optimizer	Adam	Adam
LR schedule	Reduce on plateau	Reduce on plateau
Grad steps	200	200
Learning rate (T)	0.04	0.01
Learning rate (R)	0.04	0.1
Learning rate (S)	0.04	0.01
Use scale term	No	Yes
w_{depth}	1	1
w_{align}	1	1
w^T_{reg}	1e-3	3e3
w^R_{reg}	1e-3	1e2
w^S_{reg}	N/A	3e3
w_{mask}	100	15

TABLE VIII: **Loss weights for the tracking module.**

3) **Motion regularization.**

$$\mathcal{L}_{reg}(R, T, S) = w^T_{reg}\|R\|_2 + w^T_{reg}\|T\|_2 + w^S_{reg}\|S\|_2.$$

Motion regularization prevents tracked points from exhibiting high frequency jitter when the objects they are tracking do not move.

4) **Mask consistency.** We introduce a mask consistency loss. Intuitively, this loss tries to ensure that each pixel within a 2D mask for an object from a particular camera view should have a tracked point for that object that is close to that pixel when projected into that view.
Let the set of all views be V and the set of object masks in a particular view v be $M(v)$. Then the total number of masks points N is $N = \sum_{v \in V} \sum_{obj \in M(v)} |obj|$.
Concretely, this can be written as:

$$\mathcal{L}_{mask}(\vec{p}) = \frac{1}{N} \sum_{v \in V} \sum_{obj \in M(v)} \sum_{pix \in obj} \min_{p \in \vec{p}^{obj}} \|pix - proj(p, v)\|$$

where $proj(p, v)$ is the 2D projection of 3D point p into the image space of viewpoint v.

The overall objective is computed by weighting and combining these terms:

$$\mathcal{L}_{tracking} = w_{depth}\mathcal{L}_{depth} + w_{align}\mathcal{L}_{align} + \\ + w_{reg}\mathcal{L}_{reg} + w_{mask}\mathcal{L}_{mask}$$

The weights for each term as well as optimizer parameters are enumerated in Table VIII. The transformed points with the best loss after the total number of gradient steps is complete is output as the result.

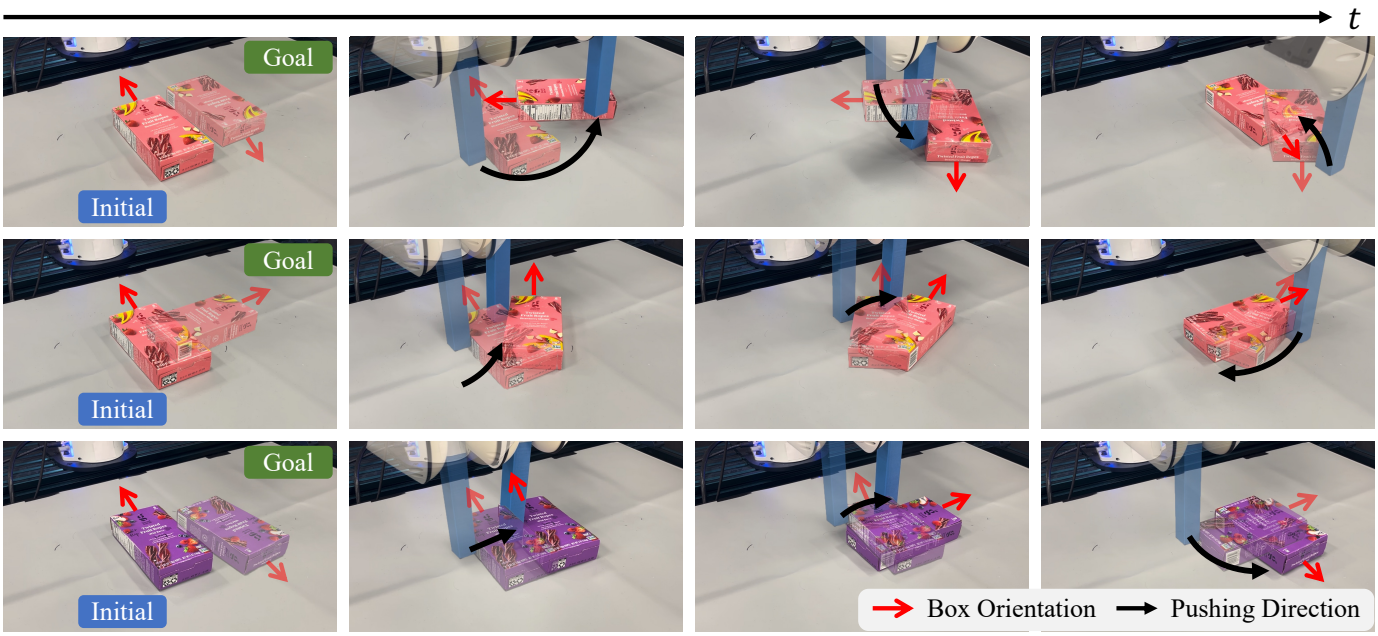


Fig. 11: **Non-prehensile box pushing.** We demonstrate our robot can push a box with unknown mass distribution from a starting pose to a target pose. Note that our box pushing is non-prehensile because the in-hand object is not fixed. We show that our method can generalize to unseen initial and target box poses in the first two rows and also previously unseen box configurations in the third row. A green arrow indicates the box's orientation, so boxes in rows 1 and 3 are flipped vertically.

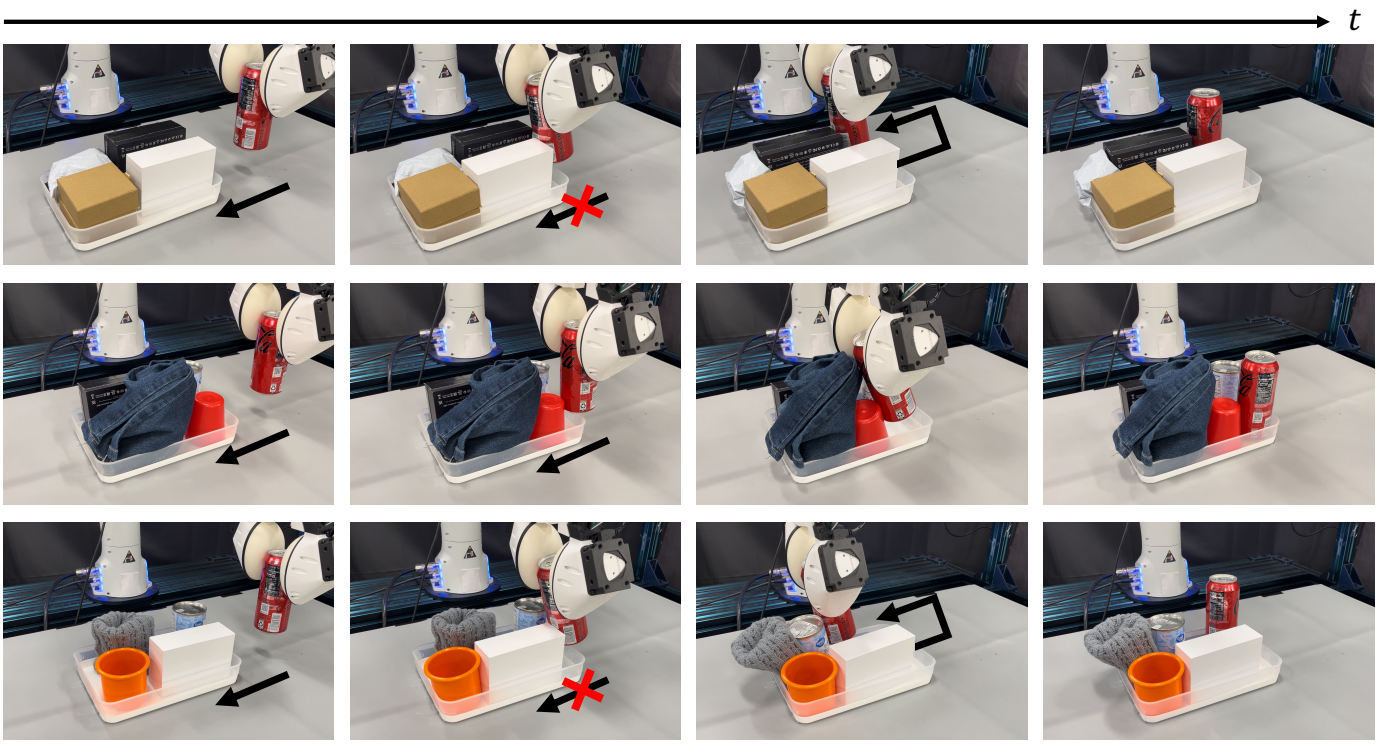


Fig. 12: **Dense packing with diverse object sets.** In the *Dense Packing* task, we demonstrate that RoboPack effectively identifies feasible insertion rows in a tray, minimizing excessive force on the robot to prevent hardware damage. The first row presents a set of objects from data collection, while subsequent rows illustrate our method's capability to adapt to objects with various visual appearances and different levels of deformability.