ManiFoundation Model for General-Purpose Robotic Manipulation of Contact Synthesis with Arbitrary Objects and Robots

Zhixuan Xu[1]*, Chongkai Gao[1]*, Zixuan Liu[2]*, Gang Yang[1]*, Chenrui Tie[3], Haozhuo Zheng[4], Haoyu Zhou[5], Weikun Peng[1], Debang Wang[1], Tianyi Chen[6], Zhouliang Yu[7], Lin Shao[1]

***Abstract*— To substantially enhance robot intelligence, there is a pressing need to develop a large model that enables general-purpose robots to proficiently undertake a broad spectrum of manipulation tasks, akin to the versatile task-planning ability exhibited by LLMs. The vast diversity in objects, robots, and manipulation tasks presents huge challenges. Our work introduces a comprehensive framework to develop a foundation model for general robotic manipulation that formalizes a manipulation task as contact synthesis. Specifically, our model takes as input object and robot manipulator point clouds, object physical attributes, target motions, and manipulation region masks. It outputs contact points on the object and associated contact forces or post-contact motions for robots to achieve the desired manipulation task. We perform extensive experiments both in the simulation and real-world settings, manipulating articulated rigid objects, rigid objects, and deformable objects that vary in dimensionality, ranging from one-dimensional objects like ropes to two-dimensional objects like cloth and extending to three-dimensional objects such as plasticine. Our model achieves average success rates of around 90%. Supplementary materials and videos are available on our project website at https://manifoundationmodel.github.io/.**

I. Introduction

Foundation models [1] that are trained on broad data are becoming versatile tools for numerous tasks, revolutionizing fields like natural language processing and computer vision. Large Language Models (LLMs) and large Vision-Language Models (VLMs) serve as vast repositories of knowledge that can potentially enhance robotic capabilities. For instance, LLMs have exhibited their superior capacity to decompose complex tasks such as meal preparation into a series of subtasks, leading to substantial enhancements in robot planning and reasoning processes. Recently, people have started to investigate LLMs or VLMs in robotics to develop robot foundation models. These works either leverage pretrained LLMs or VLMs as high-level planners to call a set of pretrained low-level skills [2]–[4], or directly train low-level robot actions on massive datasets [5]–[7]. While these models excel at high-level task planning and semantic understanding, they face substantial challenges when confronted with the extremely high complexity of 3D object geometry, deformations, force constraints, and the lack of comprehension of different robot manipulator topologies when deploying robots to actually perform low-level manipulations. In other words, the breadth and adaptability of low-level manipulation capabilities have received considerably less attention compared to the high-level task-planning abilities in robotics. It is thus essential to build a *manipulation foundational model* that equips robots with the requisite low-level manipulation proficiency to engage with the physical world and competently execute a diverse range of tasks.

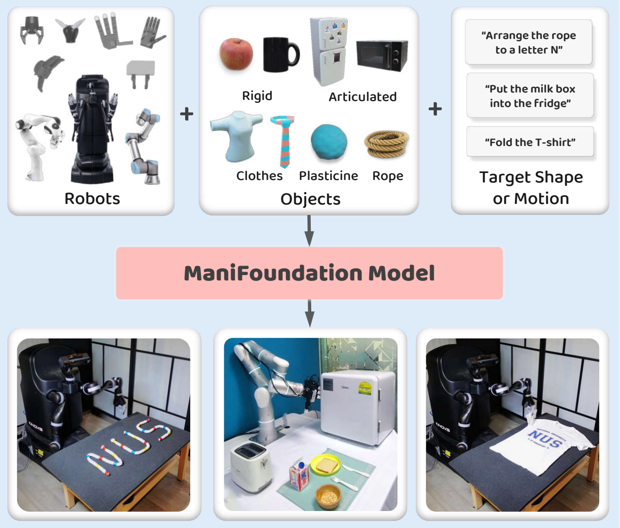

Fig. 1. We propose a ManiFoundation Model that can generalize over a diverse range of robots and objects, and perform various kinds of manipulation tasks based on 3D point cloud input.

To build a manipulation foundation model, the core problem is to define a unified task formulation for all kinds of robot manipulation tasks, just as the auto-regressive prediction task in GPTs [8], the mask prediction task in BERT [9], and the vision-language contrastive learning task in CLIP [10]. This formulation must: 1) handle arbitrary objects, including articulated rigid objects, solid rigid ob-

* denotes equal contribution

[1] Zhixuan Xu, Chongkai Gao, Gang Yang, Weikun Peng, Debang Wang, and Lin Shao are with the Department of Computer Science, National University of Singapore. ariszxxu@gmail.com, gaochongkai@u.nus.edu, yg.matinal@gmail.com, debang@u.nus.edu, linshao@nus.edu.sg

[2] Zixuan Liu is with Tsinghua Shenzhen International Graduate School, Tsinghua University. zx-liu21@mails.tsinghua.edu.cn

[3] Chenrui Tie is with School of Electronics Engineering and Computer Science, Peking University. crtie@pku.edu.cn

[4] Zheng Haozhuo is with Department of Mathematics, National University of Singapore. muztaga2@gmail.com

[5] Haoyu Zhou is with Department of Mechanical Engineering, National University of Singapore zhouhaoyu01@u.nus.edu

[6] Tianyi Chen is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University tianyyiii@gmail.com

[7] Zhouliang Yu is with the Division of Emerging Interdisciplinary Areas, Hongkong University of Science and Technology zhouliangyu@link.cuhk.edu.cn

jects, and deformable objects; 2) support a diverse range of manipulation tasks, like grasping, pulling, re-orientating, and folding; 3) adapt for cross-platform integration, accommodating diverse robot designs and configurations; 4) integrate seamlessly with LLMs/LVMs for effective task execution and long-horizon planning.

In this work, we propose a foundation model framework for low-level robotic manipulation, aiming to meet the above requirements. We formulate the manipulation foundation task as the contact synthesis task. Specifically, the foundation model $\mathcal{F}$ takes as input 1) object point cloud and physical attributes (such as friction coefficients, Young's modulus, etc), 2) task descriptions represented as target motion, 3) robot manipulator descriptions as point clouds, 4) manipulation region masks if needed, for example, the task requires the robot to grasp the pan's handle. The foundation model $\mathcal{F}$ outputs the contact points, contact forces or post-contact motion, so that robots can manipulate the object toward its target motion. We generate a large-scale annotated dataset to train the model. We test our ManiFoundation model over diverse experiments to verify its effectiveness both in simulation and the real world. In summary, our primary contributions include:

- We propose a holistic framework towards foundation models for general-purpose robotic manipulation via contact synthesis, achieving broad generalization over various objects, robots, and tasks.
- We implement the ManiFoundation model consisting of both neural network backbones for visual and physical feature extraction and an iterative contact wrench optimization method that generates feasible robot poses and avoids collisions and penetrations.
- We develop a comprehensive pipeline to generate a large-scale dataset with annotations covering diverse objects and robot hands. This dataset will be made publicly available on our project website.
- We conduct extensive experiments to verify the effectiveness of our proposed framework both in simulation and real-world experiments over various manipulation tasks including (articulated) rigid object manipulations and deformable object manipulations.

II. Related Work

A. Foundation Models in Robotics

Foundation models in robotics [11] recently received a lot of attention. SayCan [2] grounds LLMs using value functions of pretrained skills. VoxPoser [3] produces affordances and constraints from LLMs and vision-language models to combine as 3D value maps for motion planners. Codes-as-policies [4] use LLMs to write robot policy code leveraging natural language instructions. RoboCat [12] is self-improving AI agent for robotics, learning tasks across various arms and autonomously generating new data to enhance its skills. Robotics Transformer (RT-1) [5] is trained over 130k real-world data containing over 700 tasks collected by 13 robots over 17 months. RT-2 [6] enhances vision-language-action models using web and robotic data. RT-X [7] gathers a dataset from 22 robots with 527 robotic skills and shows that scaling data improves the generalization capabilities between multiple robot platforms. Our work focus on developing the foundation model specifically designed for manipulation skills via contact synthesis to support general robotic manipulation with the broad generalization over objects including diverse deformable objects, robots, and tasks.

B. Learning Visual Affordances

Learning affordances [13]–[15] has received increasing attention for robotic manipulation, including grasping [16]–[18] and articulated object manipulation [19]–[21]. Where2Act [19] predicts the actionable point-on-point cloud for primitive actions such as pushing and pulling. Affordance Diffusion [22] synthesizes complex interactions (i.e., an articulated hand) with a given object. Vision-Robotic Bridge (VRB) [23] train a visual affordance model on large-scale videos with human behavior to predict contact points and a post-contact trajectory learned from human videos. Our model provides contact points for diverse robot hands and objects, including dexterous hands and deformable objects, together with contact forces and post-contact motions to guide robot low-level action execution.

III. ManiFoundation Model Design

A robotic manipulation task can be decomposed into two parts: a task-planning module that gives the object trajectory, and a manipulation module that guides robots to achieve this trajectory. For any given robot task, we assume LLMs/VLMs, other models, or differentiable simulators with digital twins, can convert it to a sequence of object point clouds as waypoint subgoals. Examples of this step for tasks in our experiments are provided in our website. Note this aspect is not the primary focus of us. Here we investigate to develop a universal model that guide robots to manipulate objects towards the target pose or shape. The model should generalize over arbitrary objects, robots, and tasks (target shape or motion).

We formulate our ManiFoundation model as a *contact synthesis* problem. Specifically, our foundation model $\mathcal{F}$ takes as input 1) manipulated object descriptions including object point cloud and physical properties, 2) task descriptions as target motion, 3) robot descriptions as point clouds, 4) manipulation region masks if needed, for example, the task requires the robot to grasp the pan's handle. It outputs the contact points and contact forces so that robots can manipulate the object toward its target shape or motion. Formally, let a point cloud with N points and normals on each point be $\mathcal{P} = \{(x_1, n_1), \cdots, (x_N, n_N)\} \in \mathbb{R}^{N\times 6}$, where $x_i \in \mathbb{R}^3$ is the position and $n_i \in \mathbb{R}^3$ is the inward normal vector of the i-th point. Let $\mathcal{P}_o$ and $\mathcal{P}_h$ be the input object and robot hand point cloud respectively. Let $\mathcal{M} = \{m_i\}_{i=1}^N, m_i \in \mathbb{R}^3$ be the task motion that should be achieved on each point of the object. Let the physical feature of the object be Γ and the region mask be $\mathcal{R}$. The manipulation foundation problem is defined as finding

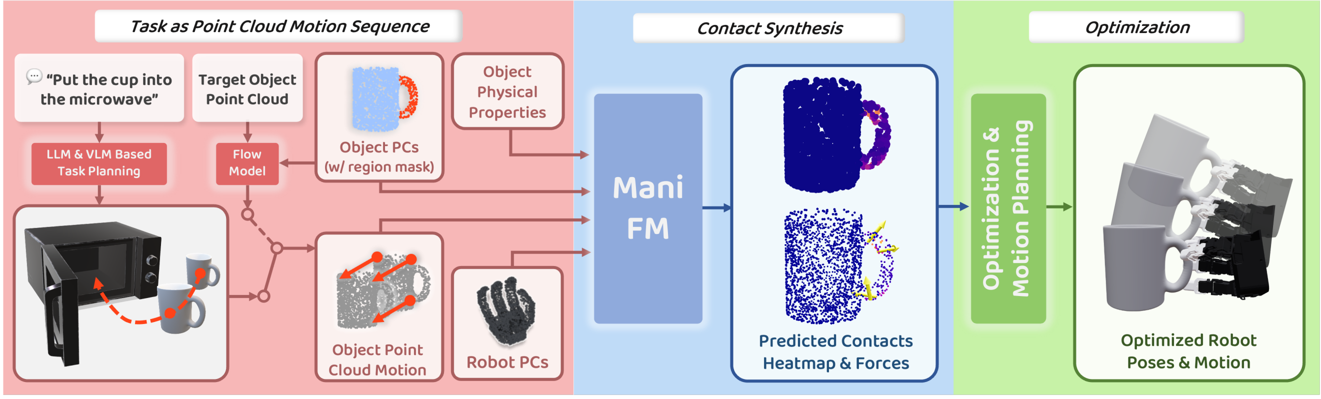
Task as Point Cloud Motion Sequence
"Put the cup into the microwave"
Target Object Point Cloud
Object PCs (w/ region mask)
Object Physical Properties
LLM & VLM Based Task Planning
Flow Model
Object Point Cloud Motion
Robot PCs
Contact Synthesis
Mani FM
Predicted Contacts Heatmap & Forces
Optimization
Optimization & Motion Planning
Optimized Robot Poses & Motion

Fig. 2. The pipeline of our Manifoundation model. Left: we decompose a manipulation task to a sequence of object point cloud motions from either VLM-based planning or a flow model. Middle: we train a ManiFoundation network to predict the contact point and force heatmap for each motion of the sequence. Right: we acquire the robot pose for execution based on optimization with the initial results from the contact point and force heatmaps.

a contact solution $M_{\mathcal{C}}, M_{\mathcal{S}} = \mathcal{F}(\mathcal{P}_o, \mathcal{P}_h, \mathcal{M}, \mathcal{R}, \Gamma)$ that satisfies:

$$M_{\mathcal{C}}, M_{\mathcal{S}} = \min \|\mathbf{f}(\mathcal{P}_o, \mathcal{P}_h, \Gamma, M_{\mathcal{C}}, M_{\mathcal{S}}) - \mathcal{M}\|, \quad (1)$$

where $M_{\mathcal{C}} = \{c_i\}_{i=1}^{H}, c_i \in \mathbb{R}^3$ is the predicted contact point positions with H points on the object point cloud, and $M_{\mathcal{S}} = \{s_i\}_{i=1}^{H}, s_i \in \mathbb{R}^3$ is the contact force corresponds to each contact point c_i. $\mathbf{f}$ stands for the forward dynamics function that outputs the object per-point motion given the contact points and forces. An overview of our foundation model is shown in Fig. 2. In the following parts, we describe the modules one by one.

A. The ManiFoundation Network

The whole network structure is illustrated in Figure 3. There are mainly two parts of our network: the Feature Extractor and the Conditional Variational Autoencoder (CVAE). The feature extractor is designed to encode information from the object and robot manipulator, while the CVAE module is designed to fuse extracted features and generate contact point and force heatmaps for execution. We introduce the details of each module in the following sections.

1) Feature Extractor: We encode the object point cloud and robot manipulator point cloud separately with a *PointNet++* [24] ϕ_o for $\mathcal{P}_o$ and a *PointNet* [25] as ϕ_h for $\mathcal{P}_h$. However, to deal with the diversity in both objects and task requirements, we also design specific modules before these feature extractors.

a) Physical Properties Encoder: Our model accommodates a large variety of object types, each with distinct physical properties. These properties are crucial for object manipulation, as they indicate different manipulation strategies. For example, consider a handkerchief and a piece of card of the same shape. To spin the handkerchief, one needs to grasp the four corners. However, to rotate a card you can apply friction to other areas. This leads to totally different contact points and forces solutions, thus encoding the physical attributions is necessary to distinguish different kinds of objects.

In this work, we encode different physical properties with different Multi-layer Perceptions (MLPs) $\psi_i : \mathbb{R}^1 \rightarrow \mathbb{R}^{10}, i = 1, \cdots, 6$. The same MLP processes the same physical properties across different points. Specifically, we encode friction coefficients for all objects, density and elasticity coefficients [26] for 2D deformable objects, and Young's modulus and Poission's ratio [27] for 1D and 3D deformable objects. Each physical property is a one-dimensional value on each point of $\mathcal{P}_o$. All physical features extracted by separate MLPs are aggregated using Masked Mean Pooling (MMP), where the missing physical properties on certain types of objects are masked. Thus, for each point of $\mathcal{P}_o$, the extracted physical feature Γ is $\Gamma = MMP(\psi_1(\gamma_1), \cdots, \psi_6(\gamma_6)) \in \mathbb{R}^{10}$, where γ_i is the i-th physical property.

This design offers several benefits. Firstly, it can handle objects possessing a wide and variable array of physical properties. Secondly, if additional physical properties are needed to identify new object types, new MLPs can be seamlessly added to map these new properties to feature vectors, which are then combined with previous feature vectors and fed into the mean pooling layer.

b) Manipulation Region Selection: In practice, certain manipulation tasks have extra constraints that emphasize which part of an object can be manipulated. These constraints often stem from human preferences or environmental constraints. For instance, it's generally preferred for robots to grasp the handle of a teapot or pan, and a robot cannot grasp the bottom of a cup when it is placed on the table.

To accommodate these constraints, we design a manipulation region mask to encode the region information, denoted as $\mathcal{R} = \{r_i\}_{i=1}^{N} \in \mathbb{R}^N$. $r_i = 1$ means the i-th point of $\mathcal{P}_o$ is allowed to contact, otherwise $r_i = 0$. In practice, this mask can be assigned by humans or generated from environmental constraints. Our model is trained to predict contact points exclusively within the manipulation region mask.

For deformable objects, specifying the point-wise target motion can be challenging. Our model accommodates this by accepting the target point cloud of the deformable objects as input. A scene flow model [28] processes both the current

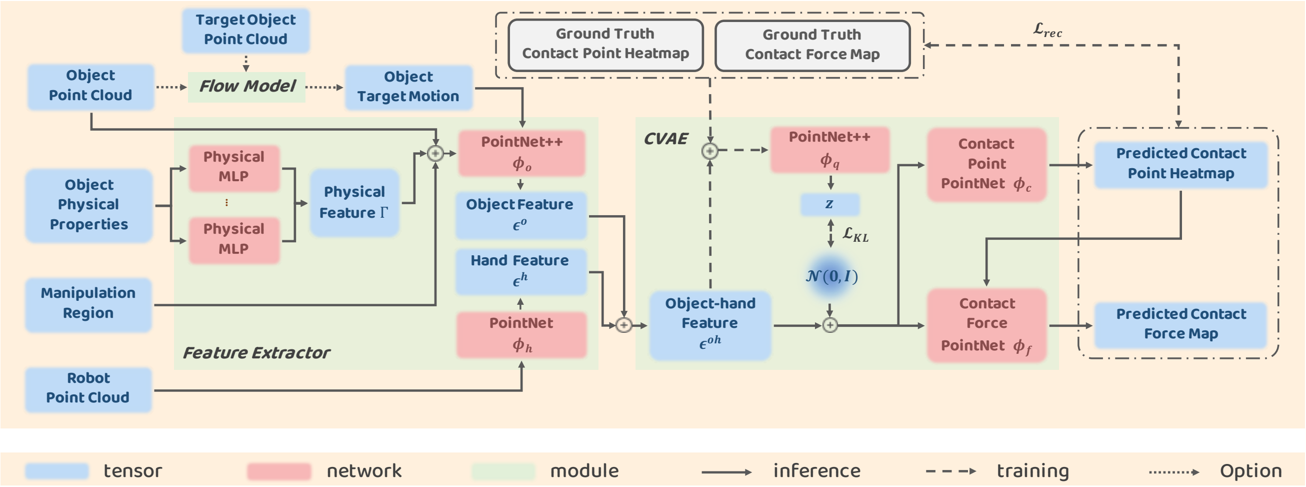
Target Object Point Cloud
Object Point Cloud
Flow Model
Object Target Motion
Ground Truth Contact Point Heatmap
Ground Truth Contact Force Map
$\mathcal{L}_{rec}$
Object Physical Properties
Physical MLP
Physical MLP
Physical Feature Γ
PointNet++ ϕ_o
Object Feature ϵ^o
Hand Feature ϵ^h
PointNet ϕ_h
Manipulation Region
Robot Point Cloud
Feature Extractor
CVAE
PointNet++ ϕ_q
z
$\mathcal{L}_{KL}$
$\mathcal{N}(0,I)$
Object-hand Feature ϵ^{oh}
Contact Point PointNet ϕ_c
Contact Force PointNet ϕ_f
Predicted Contact Point Heatmap
Predicted Contact Force Map
tensor
network
module
inference
training
Option

Fig. 3. Overview of our Manifoundation network. The feature extractor module incorporates the information from both object and robot point clouds, and the CVAE module generates the contact point and force maps on the given object.

and target point clouds to deduce the required target motion. In contrast, for articulated or rigid objects, determining the target motion from the current to the target point cloud is straightforward.

In summary, ϕ_o takes as inputs object point cloud $\mathcal{P}_o$, the task motion $\mathcal{M}$, the physical feature Γ, and the manipulation region mask $\mathcal{R}$, and output the extracted feature vector for each point of the object. ϕ_h take as input only the hand point cloud $\mathcal{P}_h$. That is:

$$\begin{aligned}\epsilon_i^o &= \phi_o(x_i, n_i, m_i, r_i, \Gamma), i = 1, \cdots, N_o.\\ \epsilon_i^h &= \phi_h(x_i, n_i), i = 1, \cdots, N_h.\end{aligned} \tag{2}$$

We then perform max pooling on the extracted hand features to get a global hand feature ϵ^h and concatenate each object point feature ϵ_i^o with ϵ^h to get a combined feature point cloud $\epsilon^{oh} = \{\epsilon_i^{oh}\}, i = 1, \cdots, N_o + N_h$. Next, the following CVAE will fuse them to get the final output.

2) CVAE for Contact Point and Force/Motion: In practice, multiple contact point and force combinations can achieve the same target motion for an object. For instance, to advance a cube, one might push from behind or pull from its sides. To generate a range of solutions, we utilize a Conditional Variational Autoencoder (CVAE) for creating contact point heatmap and force/motion heatmap. The contact point heatmap, with dimensions $\mathbb{R}^{N\times 1}$, assigns a probability score to each point indicating its likelihood of being a contact point. The force/motion map, sized $\mathbb{R}^{N\times 3}$, represents either the contact force or the post-contact motion at each point.

Specifically, during training, the ground truth contact point heatmap $M_{\mathcal{C}'}$ and force/motion map $M_{\mathcal{S}'}$ together with ϵ^{oh} are encoded into latent variables $z \in \mathbb{R}^{64}$ by a posterior *PointNet++* ϕ_q: $z = \phi_q(\epsilon^{oh}, M_{\mathcal{C}'}, M_{\mathcal{S}'})$, and z will be concatenated on each point of the condition feature ϵ^{oh} to be sent to following decoders. During the inference phase, the prior latent variables are sampled from a Gaussian distribution. To get the contact points and force/motion maps , we use two *PointNet*s ϕ_c and ϕ_f to predict the contact point heatmap $M_{\mathcal{C}}$ and contact force/motion map. We then use non-maximum suppression (NMS) [29] to select H contact points $\mathcal{C}$, where H is less than or equal to the finger number of the hand. With the predicted contact heatmap $M_{\mathcal{C}}$ as an extra input, we predict the force map $M_{\mathcal{S}}$ of shape $\mathbb{R}^{N\times 3}$, composed of 3D vectors on each point of the object by ϕ_f. We select the forces/motions on the chosen contact points as the final contact force/motion $\mathcal{S}$.

The training objectives of the whole network include the mean-square loss between the predicted and ground-truth contact point and force heatmaps, as well as the CVAE training loss, that is:

$$\begin{aligned}\mathcal{L} = &\lambda_{\mathcal{C}}\mathcal{L}_{MSE}(M_{\mathcal{C}'}, M_{\mathcal{C}}) + \lambda_{\mathcal{S}}\mathcal{L}_{MSE}(M_{\mathcal{S}'}, M_{\mathcal{S}})\\ &+ \lambda_{KL}\mathcal{D}_{KL}(\phi_q(\epsilon_{oh}, M_{\mathcal{C}'}, M_{\mathcal{S}'}) \,||\, \mathcal{N}(0, I)),\end{aligned} \tag{3}$$

where $\lambda_{\mathcal{C}}, \lambda_{\mathcal{S}}, \lambda_{KL}$ are hyperparameters for loss weights. $\mathcal{N}(0, I)$ is a standard Guassian distribution.

B. Robotic Hand Pose Refinement

Given the contact points predicted by our network, we can directly manipulate deformable objects by simple grasping the contact points to achieve the target motion. However, for articulated and rigid object manipulation generally requires the coordinated effort of all fingers, while the contact points predicted by our network may not lead to feasible joint values of the robotic hand that satisfy the robot kinematics and avoid penetrations between robots and objects. Thus, we develop an optimization module for robotic hand pose refinement.

1) Pose Initialization: We infer the initial hand pose from the predicted contact points described in Sec. III-A.2. Specifically, for a multi-fingered hand, we can iterate all possible matches between fingertips and the predicted contact points. For instance, with four contact points and a four-fingered robotic hand, there would be $4! = 24$ possible matches. For each match, we consider the robot hand in its neutral joint poses as a rigid object and apply the Iterative Closest Point (ICP) algorithm to align robot fingertip points with contact points on the object. The result of ICP is the hand 6D pose for this match. Under this hand 6D pose, we generate a contact heatmap by applying a Gaussian kernel on the object point cloud at locations nearest to the fingertip positions. The best match, yielding the highest Intersection over Union (IoU) score between the generated heatmap and the predicted contact point heatmap, is selected. This initial 6D pose is then used for the robotic hand pose refinement.

2) Target Motion and Target Wrench: Given the target motion of the rigid object, our system uses inverse dynamics to infer the target wrench $\boldsymbol{w}$ for producing the specified motion. Specifically, the target motion over time interval Δt involves a rotation $\Delta\boldsymbol{\theta}$ followed by a translation $\Delta\boldsymbol{x}$, with $\Delta\boldsymbol{\theta}$ indicating a rotation around the axis $\frac{\Delta\boldsymbol{\theta}}{|\Delta\boldsymbol{\theta}|}$ through an angle of $|\Delta\boldsymbol{\theta}|$ radians. It can be calculated as follows:

$$\hat{\boldsymbol{w}} = \frac{2}{\Delta t^2}\begin{bmatrix}\Delta\boldsymbol{x}/M \\ \Delta\boldsymbol{\theta}/I\end{bmatrix}, \tag{4}$$

$$\boldsymbol{w} = \hat{\boldsymbol{w}}/\|\hat{\boldsymbol{w}}\|, \tag{5}$$

where object mass and inertia are denoted as M and I, which are computed approximately by estimating the volume of the object. We normalize $\hat{\boldsymbol{w}}$ to target wrench $\boldsymbol{w}$ for practical simplicity. In practice, we care more about the direction of wrench and leave low-level force/torque control to take care of action to move objects towards target shapes.

3) Wrench Optimization: Given a target wrench and robot hand's joint poses, we calculate the optimal contact forces $\boldsymbol{f}^*$ exerted by m fingertips onto the object to minimize the norm of wrench residual $\boldsymbol{r}$ in Eqn. 7.

$$\boldsymbol{r}(\boldsymbol{w},\boldsymbol{q},\boldsymbol{f}_{1:m}) = \sum_{i=1}^{m} G_i(\boldsymbol{q})\boldsymbol{f}_i - \boldsymbol{w}, \tag{6}$$

$$\begin{aligned}\boldsymbol{f}^*_{1:m}(\boldsymbol{w},\boldsymbol{q}) = \underset{\boldsymbol{f}_{1:m}}{\arg\min}\ &\Big\|\boldsymbol{r}(\boldsymbol{w},\boldsymbol{q},\boldsymbol{f}_{1:m})\Big\|^2,\\ \text{s.t. } &\boldsymbol{f}_{1:m} \in FC(\mu),\end{aligned} \tag{7}$$

where $G_{1:m}$ is the grasp mapping matrices that maps 3d contact force at fingertips to 6d wrench applied to the object and $FC(\mu)$ is the friction cone with coefficient μ guaranteeing that contact forces satisfy Coulomb's law [30]. We do not require fingertips to always touch the object's surface through optimization. If fingertip i is away from the surface, G_i is calculated from its center position $\boldsymbol{p}_i$ and the normal vector at its projection onto the surface $\bar{\boldsymbol{n}}_i$.

The optimization problem that improves the hand pose $\boldsymbol{q}$ to enhance the grasp quality is formulated as:

$$\min_{\boldsymbol{q}} \Big\|\boldsymbol{r}^*(\boldsymbol{w},\boldsymbol{q})\Big\|^2 + \sum_{i=1}^{m}\Big|\Phi\big(\boldsymbol{p}_i(\boldsymbol{q})\big)\Big|^2, \tag{8}$$

$$\text{s.t. } \boldsymbol{q} \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}], \tag{9}$$

$$\Phi\big(\boldsymbol{c}_{1:n}(\boldsymbol{q})\big) \geq \epsilon_{1:n}, \tag{10}$$

$$\Big\|\boldsymbol{c}_i(\boldsymbol{q}) - \boldsymbol{c}_j(\boldsymbol{q})\Big\| \geq \epsilon_i + \epsilon_j, \tag{11}$$

where $\boldsymbol{r}^*(\boldsymbol{w},\boldsymbol{q}) = \boldsymbol{r}\big(\boldsymbol{w},\boldsymbol{q},\boldsymbol{f}^*_{1:m}(\boldsymbol{w},\boldsymbol{q})\big)$ is the minimum wrench residual. Our optimization objective in Eqn. 8 comprises two terms. The first term is the minimized norm of wrench residual, which measures the current grasp's capability to induce the desired motion in the object. The second term is the aggregate distance between fingertips and the object's surface, defined using the object's signed distance field (Φ) and the fingertip center positions $\boldsymbol{p}_{1:m}$ obtained from forward kinematics. for any 3D point $\boldsymbol{x}$, we locate the nearest points $\bar{\boldsymbol{p}}$ on the object's surface and the outward normal $\bar{\boldsymbol{n}}$. $\Phi(\boldsymbol{p}) = \bar{\boldsymbol{n}}\cdot(\boldsymbol{p}-\bar{\boldsymbol{p}})$, with $\frac{\partial\Phi(\boldsymbol{x})}{\partial\boldsymbol{x}} = \bar{\boldsymbol{n}}$. The second term in Eqn. 8 quantifies the sum of the distances between the fingertip centers and the object's surface.

We also add a set of constraints to make the resulting hand pose practical. Eqn. 9 describes the robot joint limits and we want the system penetration-free. For simplicity, we use a set of n spheres with different radii $\epsilon_{1:n}$ to represent the complex collision shape of robot hand. The spheres are attached to the fingers and palm, so the center positions $\boldsymbol{c}_{1:n}(\boldsymbol{q})$ are obtained from forward kinematics. Eqn. 10 guarantees no penetration between the hand and object and Eqn. 11 guarantees no self-penetration within the fingers and palm of the robot hand.

We solve this nonlinear optimization problem iteratively by performing Taylor expansion to Eqn. 8-11 in the neighborhood of $\boldsymbol{q}$, $\boldsymbol{f}^*_{1:m}$ and obtaining a convex optimization problem (second-order cone programming) which could be solved quickly by CVXPY [31] as follows:

$$\min_{\delta\boldsymbol{q},\delta\boldsymbol{f}_{1:m}} \Big\|\boldsymbol{r}^*(\boldsymbol{w},\boldsymbol{q}) + \sum_{i=1}^{m}\frac{\partial G_i(\boldsymbol{q})}{\partial\boldsymbol{q}}\boldsymbol{f}^*_i\delta\boldsymbol{q} + G_i(\boldsymbol{q})\delta\boldsymbol{f}_i\Big\|^2 \tag{12}$$

$$+\sum_{i=1}^{m}\Big|\Phi(\boldsymbol{p}_i) + \frac{\partial\Phi(\boldsymbol{p}_i)}{\partial\boldsymbol{p}_i}\frac{\partial\boldsymbol{p}_i(\boldsymbol{q})}{\partial\boldsymbol{q}}\delta\boldsymbol{q}\Big|^2, \tag{13}$$

$$\text{s.t. } \boldsymbol{f}^*_{1:m} + \delta\boldsymbol{f}_{1:m} \in FC(\mu), \tag{14}$$

$$\boldsymbol{q} + \delta\boldsymbol{q} \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}], \tag{15}$$

$$\Phi(\boldsymbol{c}_{1:n}) + \frac{\partial\Phi(\boldsymbol{c}_{1:n})}{\partial\boldsymbol{c}_{1:n}}\frac{\partial\boldsymbol{c}_{1:n}(\boldsymbol{q})}{\partial\boldsymbol{q}}\delta\boldsymbol{q} \geq \epsilon_{1:n}, \tag{16}$$

$$\Big\|\boldsymbol{c}_i - \boldsymbol{c}_j + (\frac{\partial\boldsymbol{c}_i(\boldsymbol{q})}{\partial\boldsymbol{q}} - \frac{\partial\boldsymbol{c}_j(\boldsymbol{q})}{\partial\boldsymbol{q}})\delta\boldsymbol{q}\Big\| \geq \epsilon_i + \epsilon_j, \tag{17}$$

$$|\delta\boldsymbol{q}| \leq s_q,\ |\delta\boldsymbol{f_{1:m}}| \leq s_f. \tag{18}$$

In the above equations, the Jacobians $\frac{\partial\boldsymbol{p}(\boldsymbol{q})}{\partial\boldsymbol{q}}$ and $\frac{\partial\boldsymbol{c}(\boldsymbol{q})}{\partial\boldsymbol{q}}$ are provided by the simulation Jade [32]. The exact formula of grasp mapping matrices and the corresponding gradients are derived in supplementary. Besides the constraints derived from the original problem Eqn. 8-11, we also add Eqn. 18 restricting $\delta\boldsymbol{q}$ and $\delta\boldsymbol{f_{1:m}}$ in the neighbourhood of zero, so that Taylor expansion is reasonable. We then update the joint values of the robot hand by $\boldsymbol{q} \leftarrow \boldsymbol{q} + \delta\boldsymbol{q}^*$. The convergence conditions are $\|\boldsymbol{r}^*(\boldsymbol{w},\boldsymbol{q})\| < 10^{-6}$ and $|\Phi\big(\boldsymbol{p}_{1:m}(\boldsymbol{q})\big)| < 0.005$. We set the maximum number of iterations as 100. The optimization is successful if it reaches convergence in 100 iterations, otherwise, it fails and terminates after 100 iterations. The successful output of these iterative convex optimization is the joint values of the robot hand and associated torques to generate wrench closest to target wrench.

We can provide force-closure grasp optimization by setting $\boldsymbol{w} = [0,0,0,0,0,0]^T$, because when the supporting wrench set of the resulting grasp contains the neighborhood of origin point in wrench space, it can also expand to the whole wrench space by scaling the contact forces [30].

The DoF (degree of freedom) for articulated rigid bodies is less than six (wrench space). So we extend the definition of wrench residual to joint space, $\boldsymbol{r}^J(P^J, \boldsymbol{w}, \boldsymbol{q}, \boldsymbol{f}_{1:m}) = \sum_{i=1}^{m} P^J\big(G_i(\boldsymbol{q})\boldsymbol{f}_i\big) - \boldsymbol{\tau}$, where $\boldsymbol{\tau}$ is the target joint torque and P^J is the projection matrix. If we apply a wrench $\boldsymbol{w}$ to the articulated object, the resulting torque at the joint will be $P^J\boldsymbol{w}$. Take the revolution joint as an example, where $\boldsymbol{a}$ is the axis direction and $\boldsymbol{o}$ is any point on the axis. The corresponding projection matrix is $P^J = \big[(\boldsymbol{o} \times \boldsymbol{a})^T, \boldsymbol{a}^T\big]$. After replacing the definition of wrench residual, the other parts of optimization are the same as free objects.

IV. Dataset

We create a large-scale comprehensive annotated dataset that includes articulated/rigid objects and deformable objects in 1D/2D/3D forms. See Appendix C, D, E, and our website for more details.

a) Training Set: For fair and convenient network comparison, we select a small ∼3K training dataset, consisting of ∼1K rigid, ∼1K clothes, and ∼1K 3D deformation object examples. Each example contains the input of our network: oriented object point cloud, target point cloud, target motion, object physical properties, manipulation region mask, manipulator point cloud, and the ground truth contact point heatmap. The rigid dataset part contains data generated with PandaGripper, KinovaHand, and LeapHand. We also have a ∼200K training dataset to evaluate whether our model can scale up with more data.

b) Testing Set: To systematically evaluate the model performance, we design $2 \times 3 = 6$ test sets. The "3" here stands for three types of objects (rigid, cloth, 3D deformable). The "2" here is for two test groups: Seen Object & Unseen Motion and Unseen Object & Unseen Motion. Specifically, the Seen Object & Unseen Motion test group is designed to assess whether our model can learn to manipulate an object to reach an unseen arbitrary goal after seeing the object in the training set. The Unseen Object & Unseen Motion test group is designed to evaluate our model's generalization to novel objects. All 6 test sets contain ∼200 examples.

V. Experiments

In this section, we conduct experiments to answer the following questions:

Q1: What is the overall performance of our model?

Q2: How does training on different object categories affect our performance?

Q3: How do physical properties affect the output of the network?

Q4: Can our CVAE-based network generate diverse proposals with the same input?

Q5: Can manipulation region mask affect and constrain the network contact heatmaps and the optimization result?

Q6: How effective is our optimization algorithm compared to other approaches on force-closure grasp optimization?

Methods SR	Seen Object & Unseen Motion			Unseen Object & Unseen Motion		
	Rigid	3D Deform	2D Deform	Rigid	3D Deform	2D Deform
Baseline	23.5%	2.0%	0.6%	22.5%	2.5%	3.0%
Ours(SD)	**92.5%**	88.5%	86.23%	97.0%	82.0%	84.0%
Ours(3K)	91.0%	85.0%	**89.22%**	96.0%	82.5%	82.5%
Ours	91.5%	**91.0%**	83.23%	**97.5%**	**85.5%**	**84.5%**

TABLE I

Success rates of the network over different experiment settings.

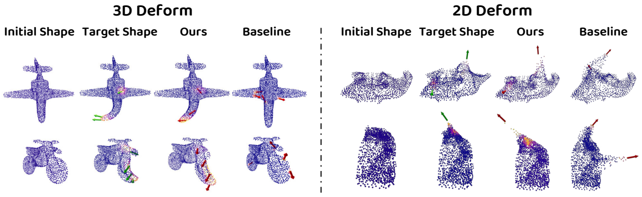

Fig. 4. Visualizations of evaluation on deformable objects. The color of the point cloud represents the contact heatmap, with purple representing low values and yellow representing high values. Arrows are the force or motion direction. The representation of all visualizations of this figure is consistent with the latter figures.

Q7: Does combining network and optimization improve task-oriented grasp generation? (See Supplementary)

Q8: Can our model generalize to novel robot manipulators? (See Supplementary)

Q9: How does our proposed framework perform on real robot experiments with real point clouds? (See Supplementary)

To report quantitatively results, we evaluate our methods by success rate all in simulations. Success for a test example is defined for each material as follows:

For (articulated) rigid objects, Success is (1) finding a hand pose and finger joint values to grasp the object without penetration; (2) moving the object with the optimized contact force in simulation assuming precise mass and inertia, reaching a 6D pose whose average point-wise distance to the target point cloud is no more than 1mm.

For deformable objects: We apply the predicted contacts on the objects in simulation. We use DiffCloth [26] for 2D deformable objects and SoftMAC [27] for 1D/3D deformable objects. We can calculate the object point cloud after applying contact forces in simulators. The success rate is defined based on the distance between the results from the object point cloud and the target point cloud. Detailed thresholds and parameters are put in the supplementary.

A. Neural Network Evaluation

1) Overall Performance & Data Fusion:

To answer Q1 and Q2, we train our model described in Sec. III-A on the above 3K training set denoted as **Ours(3K)** and compare it with three approaches: **Baseline**, **Ours(SD)**, and **Ours**. **Baseline** randomly selects the same number of contact points as **Ours**, applying forces or motions in random directions for deformable objects. For articulated/rigid object, **Baseline** sets the robot hand/fingers at the maximum open status, then moves it toward the object's center, closing the fingers upon contact with the object's surface. **Baseline** calculate the forces of each fingerstip using Eqn. 7. We chose such a baseline because we are the first to train a unified model for contact synthesis on objects of various types. We

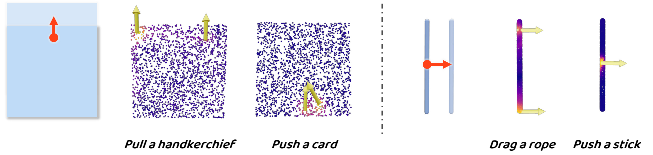

Fig. 5. Visualizations of how physical properties affect network outputs. Blue squares and cylinders are object meshes. Overlapping images with transparency represent the moved object. The orange arrow represents the motion of the object.

put other less related baselines in supplementary. **Ours(SD)** stands for training our model only Single Dataset (∼1K rigid, ∼1K clothes, ∼1K 3D deformation respectively instead of together). **Ours** contains our network trained on a larger dataset of ∼200K training data.

To answer Q1, as shown in Tab. I, **Ours** has an average success rate of over 90, 85, 80% over rigid objects, 3D deformable objects, and 2D deformable objects. It indicates our pipeline can synthesize reasonable contacts for diverse objects with broad generalization. Meanwhile, the low success rate of **Baseline** and the visualization 4 illustrates the difficulty of each task.

To address Question 2, we conducted a comparison between **Ours(3K)** and **Ours(SD)**. The **Ours(3K)** have comparable or slightly better performance. This reflects that our model is capable of effectively integrating various object types without any degradation in performance, compared to training exclusively on a single object type.

2) Physical Properties: To further understand the impact of the physical properties (in Q3), we conducted two experiments depicted in Fig. 5, evaluating how variations in physical properties influence our model's behavior. The experiments involved two shapes: a thin square and a long cylinder. The square, when associated with the physical properties of cloth or a rigid body, represents a deformable handkerchief or a rigid card, respectively. Similarly, the cylinder, with the properties of a 3D deformable object or a rigid body, symbolizes a rope or a stick. We use identical point clouds and motions to our model, altering only the physical properties, and observing the outcomes.

Fig. 5 demonstrates that to advance the thin square, our model opts to pull the forward corners, reflecting the handkerchief's pliability, and pushes from the rear for the rigid card. Pushing the handkerchief from behind would cause significant deformation. For the long cylinder, our model drags both ends on the rope and pushes the middle point given stick's rigidity. The above result illustrates that our model can adjust its outputs based on the input of physical properties, indicating our model learns to reason about the dynamics of the object.

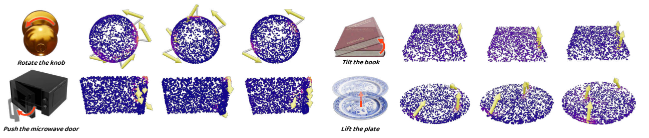

Fig. 6. Visualization of multimodal contact syntheses generated by our CVAE-based model.

3) Multimodal Syntheses: To answer Q4 - whether our model can generate diverse solutions given the same input, we choose four novel objects that are not in our training dataset. For each object, we choose a common desired object motion such as rotating the knob and tilting the book. Given the above input, we draw different $z \in \mathbb{R}^{64}$ samples from $\mathcal{N}(0, I)$ and visualize their outputs. As shown in Fig. 6, thanks to our CVAE module design, our method can generate diverse physically plausible contact outputs given the same input. For example, to rotate a knob, our model can provide solutions that include both holding the upper side and manipulating the lower side of the knob.

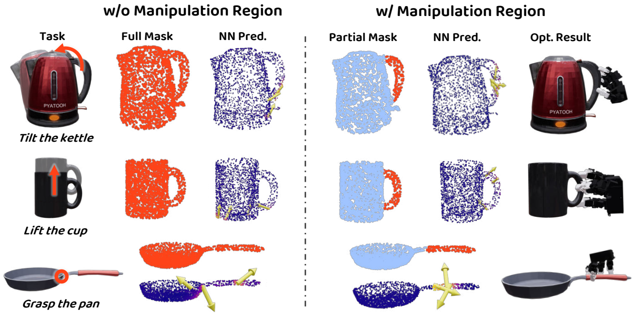

Fig. 7. Visualizations of how manipulation region mask input affects our model prediction and the optimization result.

4) Manipulation Region Selection: To explore whether a manipulation region mask can effectively guide the network's contact point predictions (Q5), we tested our model with three new everyday objects and tasks, not included in our training set. These objects have specific manipulation areas, like handles. We assessed the model's performance with and without a manually selected manipulation region mask, and the results are displayed in Fig. 7. The mask leads the model to favor these regions. Without a mask, it may suggest contact points suitable for the intended motion but unsuitable for practical manipulation. For instance, lifting a cup from its bottom on a table is unfeasible. Applying a mask to the handle helps the model to propose viable contact points, preventing potential collisions. This approach's effectiveness is depicted in the robot hand poses generated by our model, shown on the right side of Fig. 7. This method's success is illustrated by our model's generated hand poses, presented in Fig. 7. Although generating the ideal manipulation mask isn't our focus, our model includes an API for users to define their manipulation regions.

B. Evaluating Robot Hand Pose Optimization

We evaluate the effectiveness of our proposed robotic hand pose refinement for task-oriented grasp and force-closure grasp. Task-oriented grasp requires the grasp pose to produce a specific target wrench. Note that we have derived a relationship between the target wrench and target motion for rigid objects in Sec. III-B.3. For simplicity in this evaluation, we use the target wrench as the manipulation task description. Our evaluation encompasses both task-oriented grasp and force-closure grasp scenarios over 1000 objects in our dataset. For each object, 20 distinct grasps are generated. Together, 20k task-oriented grasp and 20k force-closure grasp. In task-oriented evaluations, the target wrench for each grasp is randomly sampled. For this experiment, a grasp is considered successful if it meets three criteria: the residual wrench is less than 10^{-6}, the distance between the fingertip and the object is less than 5mm, and there are no intersections between fingers or between fingers and the object.

For a fair comparison, we provide every method with the same initialization. We retrieve a ground truth 6D palm pose that can exert the target wrench or in force-closure from our dataset. We put this 6D palm pose 30cm away from the object and add a random 60-degree rotation and 5 cm translation to the ground truth palm pose and set all other finger joints at rest joint values. Under this initialization, we run our robotic hand pose refinement and compare it with a direct grasp method. The direct grasp method directly moves the robot hand toward the object's center, closing the fingers upon contact with the object's surface and calculating the forces of each fingertip using Eqn. 7.

Furthermore, our analysis includes three types of robotic hands: Barrett [33], LeapHand [34], and ShadowHand [35]. The results are shown in Table. II. Our refinement module provides much higher success rates than the baseline for both scenarios and all considered hands.

Method	Barrett		LeapHand		Shadow	
	Task-oriented SR | Force-closure SR					
Baseline	T 95.2%	F 96.3%	T 91.9%	F 95.1%	T 77.7%	F 81.9%
Ours	T 16.5%	F 13.2%	T 22.1%	F 23.7%	T 10.4%	F 18.9%

TABLE II

Hand pose refinement and direct grasp comparisons. T and F indicate task-oriented grasp and force-closure grasp.

C. Real World Experiments

In this section, we evaluate our system's performance of manipulating various rigid, clothes, and 3D deformation objects in multiple real-world settings, as shown in Figure 1. We put detailed descriptions in the supplementary. **Rope Rearrangement** We design a rope rearrangement experiment for a Kinova MOVO robot arm to rearrange a random reset rope to letters "NUS". We perform 10 times for each letter in "NUS". Our model achieves 90% success rate, indicates that our model is accurate in generating grasp points for deformation objects like ropes. **Breakfast Preparation** We design a complex breakfast preparation experiment for a single Flexiv robot arm and a LeapHand as the manipulator. In this setting, the robot will open the fridge door, take out the milk box, place the milk box on the table, pick up a piece of bread, put the bread into the toaster, open the toaster, take out the cooked bread and place it to the plate. Among these objects, the fridge door is an articulated object, while the milk box is a normal rigid body object, and the bread is treated as a 3D deformation object. **Cloth Folding** We design a cloth folding experiment for a Kinova MOVO robot arm to fold a T-shirt. We extract the T-shirt point clouds using SAM [36]. Movo will then grasp the point with the highest heatmap value and move to the predicted target point.

VI. Conclusion and Limitations

We introduce a framework that develops a manipulation foundation model via contact synthesis, taking inputs like point clouds, physical properties, target motions, and region masks to output contact points and forces or motions. Our model, tested on a variety of objects including articulated, rigid, and deformable items, demonstrates average success rates of around 90% across simulations and real-world applications.

Our model can be further improved in the following aspects. First, our framework focuses on quasi-static tasks because we separate the whole trajectory into discrete motions at each time step. Future works can seek to extend our framework to high-dynamical tasks by considering multiple-step motions or the whole trajectory at each timestep. Secondly, our model focuses on single-hand finger-point contact. Future works can seek to extend our framework to multiple hand surface contact situations.

References

[1] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[2] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[3] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[4] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *arXiv preprint arXiv:2209.07753*, 2022.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[6] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[7] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models," *arXiv preprint arXiv:2310.08864*, 2023.

[8] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[10] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[11] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, *et al.*, "Foundation models in robotics: Applications, challenges, and the future," *arXiv preprint arXiv:2312.07843*, 2023.

[12] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, *et al.*, "Robocat: A self-improving foundation agent for robotic manipulation," *arXiv preprint arXiv:2306.11706*, 2023.

[13] T.-T. Do, A. Nguyen, and I. Reid, "Affordancenet: An end-to-end deep learning approach for object affordance detection," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 5882–5889.

[14] L. Yen-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin, "Learning to see before learning to act: Visual pre-training for manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7286–7293.

[15] Y.-C. Lin, P. Florence, A. Zeng, J. T. Barron, Y. Du, W.-C. Ma, A. Simeonov, A. R. Garcia, and P. Isola, "Mira: Mental imagery for robotic affordances," in *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 1916–1927. [Online]. Available: https://proceedings.mlr.press/v205/lin23c.html

[16] P. Mandikal and K. Grauman, "Learning dexterous grasping with object-centric visual affordances," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 6169–6176.

[17] J. Borja-Diaz, O. Mees, G. Kalweit, L. Hermann, J. Boedecker, and W. Burgard, "Affordance learning from play for sample-efficient policy learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6372–6378.

[18] Y.-H. Wu, J. Wang, and X. Wang, "Learning generalizable dexterous manipulation from human grasp affordance," in *Conference on Robot Learning*. PMLR, 2023, pp. 618–629.

[19] K. Mo, L. J. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani, "Where2act: From pixels to actions for articulated 3d objects," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 6813–6823.

[20] R. Wu, Y. Zhao, K. Mo, Z. Guo, Y. Wang, T. Wu, Q. Fan, X. Chen, L. Guibas, and H. Dong, "Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects," *arXiv preprint arXiv:2106.14440*, 2021.

[21] Y. Wang, R. Wu, K. Mo, J. Ke, Q. Fan, L. J. Guibas, and H. Dong, "Adaafford: Learning to adapt manipulation affordance for 3d articulated objects via few-shot interactions," in *European Conference on Computer Vision*. Springer, 2022, pp. 90–107.

[22] Y. Ye, X. Li, A. Gupta, S. De Mello, S. Birchfield, J. Song, S. Tulsiani, and S. Liu, "Affordance diffusion: Synthesizing hand-object interactions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 22 479–22 489.

[23] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak, "Affordances from human videos as a versatile representation for robotics," 2023.

[24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[26] Y. Li, T. Du, K. Wu, J. Xu, and W. Matusik, "Diffcloth: Differentiable cloth simulation with dry frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 1, pp. 1–20, 2022.

[27] M. Liu, G. Yang, S. Luo, C. Yu, and L. Shao, "Softmac: Differentiable soft body simulation with forecast-based contact model and two-way coupling with articulated rigid bodies and clothes," 2023.

[28] G. Wang, Y. Hu, Z. Liu, Y. Zhou, M. Tomizuka, W. Zhan, and H. Wang, "What matters for 3d scene flow network," in *European Conference on Computer Vision*. Springer, 2022, pp. 38–55.

[29] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *18th international conference on pattern recognition (ICPR'06)*, vol. 3. IEEE, 2006, pp. 850–855.

[30] S. P. Boyd and B. Wegbreit, "Fast computation of optimal contact forces," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1117–1132, 2007.

[31] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[32] G. Yang, S. Luo, and L. Shao, "Jade: A differentiable physics engine for articulated rigid bodies with intersection-free frictional contact," *arXiv preprint arXiv:2309.04710*, 2023.

[33] B. Technology, "Barrett bh8-282 3-fingered gripper," 2019. [Online]. Available: https://www.barrett.com/about-barretthand

[34] K. Shaw, A. Agarwal, and D. Pathak, "Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning," *Robotics: Science and Systems (RSS)*, 2023.

[35] Bwonikrobotics, "Barrett bh8-282 3-fingered gripper," 2019. [Online]. Available: https://www.allegrohand.com/

[36] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

[37] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang, "Grounded sam: Assembling open-world models for diverse visual tasks," 2024.

[38] Z. Yu, J. Fu, Y. Mu, C. Wang, L. Shao, and Y. Yang, "Multireact: Multimodal tools augmented reasoning-acting traces for embodied agent planning," in *6th Robot Learning Workshop NeurIPS 2023: Pretraining, Fine-Tuning, and Generalization with Large Scale Models*, 2023. [Online]. Available: https://openreview.net/forum?id=pXDr36kovo

[39] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3859–3866.

[40] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, "Objaverse: A universe of annotated 3d objects," *arXiv preprint arXiv:2212.08051*, 2022.

[41] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[42] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "Abc: A big cad model dataset for geometric deep learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9601–9611.

[43] Q. Zhou and A. Jacobson, "Thingi10k: A dataset of 10,000 3d-printing models," *arXiv preprint arXiv:1605.04797*, 2016.

[44] H. Geng, H. Xu, C. Zhao, C. Xu, L. Yi, S. Huang, and H. Wang, "Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7081–7091.

[45] J. Huang, Y. Zhou, and L. Guibas, "Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups," *arXiv preprint arXiv:2005.11621*, 2020.

[46] X. Wei, M. Liu, Z. Ling, and H. Su, "Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–18, 2022.

[47] B. Zhou, H. Zhou, T. Liang, Q. Yu, S. Zhao, Y. Zeng, J. Lv, S. Luo, Q. Wang, X. Yu, *et al.*, "Clothesnet: An information-rich 3d garment model repository with simulated clothes environment," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 20 428–20 438.

[48] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 209–216.

[49] R. Shi, Z. Xue, Y. You, and C. Lu, "Skeleton merger: an unsupervised aligned keypoint detector," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 43–52.

APPENDIX

A. High-level Task and Motion Planning via LLM

In our work, we develop an LLM/LVM-based object motion planner that can generate collision-free dense trajectories that follow high-level language instructions $\mathcal{I}$ from users that are long-horizon and require contextual understanding of the embodied world. In our work, the planner consists of the following components: 1) an LLM/LVM that is prompted with the description of the task to generate sparse 6D arrays that might contain infeasible waypoints or collision, 2) An open-vocabulary object detector to obtain spatial-geometrical information of the relevant objects, or even portions of the object (e.g. "the body of the microwave" or "the handle of the cup"), and 3) a sampling-based motion planner to generate collision-free dense trajectory. To obtain a sparse trajectory that roughly describes the pose change of manipulated objects from the initial states to the goal states, the planner first transforms the $\mathcal{I}$ (e.g. "Put the cup in the microwave" and "Pull the rope into the shape of N, U and S") into several decompositions $\mathcal{I} \rightarrow (i_1, i_2, ..., i_n)$. To obtain precise geometrical information about each object, We employ the multimodal detector [37] to generate 2D bounding boxes around each object, using object names as prompts, these 2D coordinates are then projected onto 3D space to gather the spatial-geometry information corresponding to the robot perspective coordinates. With the augmentation geometric information of objects, the LLM then generates a trajectory τ_i of the manipulated objects for each manipulation phase described by the sub-instruction i_t in a chain-of-thought [4], [38] style, where each waypoint consists of 6-DoF object pose. However, trajectories proposed by the LLM are not entirely reliable, as the text description and image of the environment does not contain all the comprehensive physics information of the embodied environment, and as the environment is not fully observable to the LLM/VLM, so that the waypoints may contain infeasible states or collisions with the surrounding environment. To avoid giving the foundation model infeasible waypoints, we adopt a sampling-based motion planning method to interpolate the sparse trajectory τ_i to a collision-free and feasible dense trajectory $OPT(\tau_i)$.

B. Middle-level Motion Planning via Simulation

To enable flexible collision checking while motion planning, we represent all the objects with their point clouds and load them in the Jade simulator [32]. We adopt fcl [39] for real-time collision checking for objects with the surroundings in the simulation environment. We use the following RRT-connect motion planning algorithm 1 to find a collision-free dense trajectory.

We create a large-scale comprehensive annotated dataset that includes articulated/rigid objects and deformable objects in 1D/2D/3D forms, as shown in B.

C. Articulated/Rigid Bodies

Object Pre-processing We collect 100K+ object models from 1K+ categories in Objaverse [40], ShapeNet [41], ABC [42], Thingi10K [43], and GAPartNet [44]. From these datasets, we select 100K+ objects in 1K+ categories and normalize all models into a unit box and augment each object by randomly scaling them with 4 sizes between 0.05 and 0.4. Then we remesh them into manifolds [45], abandon those with low volume and translate the obtained mesh so that its coordinate origin coincides with its center of mass. Finally, for simulation purposes, we create collision meshes for every object mesh through convex decomposition using CoACD [46].

Algorithm 1 Generate Collision-Free Dense Trajectory using RRT-Connect

```
Require: Sparse trajectory $\tau_i$ from LLM/LVM, initial state $s_{\text{init}}$, goal state $s_{\text{goal}}$, obstacle set $\mathcal{O}$
Ensure: Dense trajectory $OPT(\tau_i)$ that is collision-free
  Initialize $T_{\text{start}}$ with root $s_{\text{init}}$
  Initialize $T_{\text{goal}}$ with root $s_{\text{goal}}$
  while $T_{\text{start}}$ and $T_{\text{goal}}$ not connected do
    $s_{\text{rand}} \leftarrow$ SampleRandomState()
    $s_{\text{near}} \leftarrow$ NearestNeighbor($T_{\text{start}}$, $s_{\text{rand}}$)
    $s_{\text{new}} \leftarrow$ Steer($s_{\text{near}}$, $s_{\text{rand}}$)
    if not InCollision($s_{\text{new}}$, $\mathcal{O}$) then
      Add $s_{\text{new}}$ to $T_{\text{start}}$
      if CanConnect($s_{\text{new}}$, $T_{\text{goal}}$) then
        Connect $T_{\text{start}}$ and $T_{\text{goal}}$ through $s_{\text{new}}$
      end if
    end if
    Swap($T_{\text{start}}$, $T_{\text{goal}}$)
  end while
  $OPT(\tau_i) \leftarrow$ PathBetweenTrees($T_{\text{start}}$, $T_{\text{goal}}$)
  $OPT(\tau_i) \leftarrow$ OptimizePath($OPT(\tau_i)$)
  return $OPT(\tau_i)$
```

Annotations Our dataset contains millions of training examples. Each training example is composed of the input and the ground truth label. The input of each example is the object oriented point cloud $\mathcal{P}_o$, robot manipulator's point cloud $\mathcal{P}_h$ at its rest pose, task motion $\mathcal{M}$, manipulation region $\mathcal{R}$, and friction coefficient. The ground truth label contains the ground truth contact point heatmap and contact force heatmap. We propose a unified sampling-based method for large-scale contact synthesis. We adopted a hierarchical sampling approach to obtain suitable contact training examples. For a scaled object model, we sample points on its outer surface and obtain their corresponding normals. Since solving the inverse kinematics (IK) for multiple fingers with a floating base is relatively challenging and time consuming, we propose first sample palm poses facing the object, solve IK for each finger, and perform combinations on all finger solutions. We then sample contact forces within all the contact friction cones. At last, we calculate the sum of the wrenches exerted by all fingers to the object. Here we focus on point contact between the tips of robotic hands and the object surface. For the manipulation region defined as $\mathcal{R} = \{r_i\}_{i=1}^{N}$, there are 50% probability that $r_i = 1$ for all $i = 1 \ldots N$, representing no constraints on the output contact heatmap. Conversely, the points where $r_i = 1$ encompass the

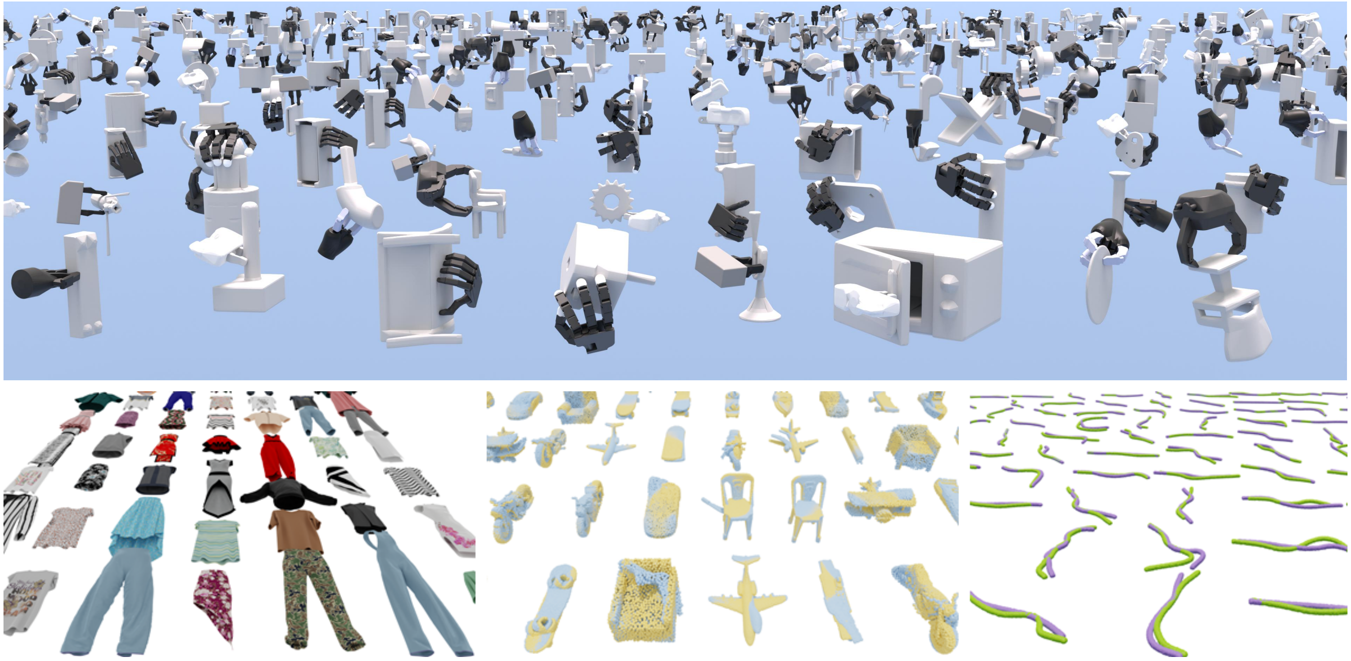

Fig. 8. Dataset visualization. Our large-scale comprehensive annotated dataset includes articulated/rigid and deformable objects in 1D/2D/3D forms.

ground truth contact points along with points in their vicinity.

D. 2D Deformable Object

Object Pre-processing We collect 2K+ clothes meshes from 4 categories and 19 sub-categories in ClothesNet [47]. For each clothes mesh, we normalize it into a unit box and simplify the mesh with quadric decimation [48] to under 3500 vertices. Objects with more than one connected component are filtered out.

Annotations Our dataset contains 100K+ training examples. Each training example has the same data format as rigid body. The physical properties of the cloth include its density, frictional coefficient, and elasticity coefficients. The manipulation region are all set to one. Our 2D deformable dataset have two parts. One part is collected during clothes folding. The other is collected during random drags. Specifically, we first rotate the clothing so that the front side faces upwards, and let it free-falls in the DiffCloth [26] simulator until it lands flat on a plane, defining this as the initial object state. Then for the first part, we define a set of contact point based on key points detected by Skeleton Merger [49] and generate target motion trajectories for folding the cloth.

E. 3D Deformable Object

Object Pre-processing For 3D Deformable objects, we collect 1K object models from 10 categories in ShapeNet [41]. All the objects are normalized into a unit ball. Then, we use farthest point sampling (FPS) to sample 2048 points on the surface. We use SoftMAC [27], a particle-based deformable object simulator, to simulate the motion of the objects when external forces are added.

Annotations For each object, we collect 100 training examples. So, the total size of our 3D deformable object dataset is 100K. Each training example has the same data format as rigid body. The physics property includes friction coefficient, Young's modulus, and Poisson ratio. Our data annotation process is structured as follows: For an object composed of particles, we initially apply the K-means clustering method to divide all points into 100 clusters. Subsequently, we randomly select one group and then randomly choose a force to apply to all particles in this group. Clustering is adopted because the entire object is composed of 2048 particles. Applying a force to just one particle would have a minimal impact on the object's overall shape. Therefore, for a noticeable deformation in the entire object, we apply the same force to all particles within a selected cluster simultaneously.

F. Simulation Experiments Metric for Deformable Objects

We apply the predicted contacts on the objects in simulation. We use DiffCloth [26] for 2D deformable objects and SoftMAC [27] for 1D/3D deformable objects. We can calculate the object point cloud after applying contact forces in simulators. The success rate is defined based on the distance between the results from the object point cloud and the target point cloud.

Specifically, for 2D deformable objects, we evaluate using 50 points with the most motion. A prediction fails if the max L2 error exceeds 0.5m, or over 50% of the selected points have an L2 error over 0.3m. For 3D deformable objects, we select 100 points. Success requires an average L2 error under 0.5cm and an average distance for selected points under 5cm. During evaluation, all point clouds are normalized into a cube with side length of 2m.

Method	LeapHand		Allegro	
	SOUM	UOUM	SOUM	UOUM
Baseline	23.5%	22.5%	18.0%	19.5%
Ours	91.5%	97.5%	94.0%	94.0%
Ours w/o WBR	24.0%	24.0%	17.0%	16.5%
Noisy GT w/ WBR	94.5%	95.0%	-	-

TABLE III
Rigid body grasp performance.

G. Ablation Study on Robot Hand Pose Refinement

For Q7 and Q8, we conduct the following experiments on the same rigid-body test set as the first experiment, with Seen Object & Unseen Motion denoted as SOUM and Unseen Object & Unseen Motion denoted as UOUM for simplicity. We report the success rate for LeapHand, as shown in Tab. III, to answer Q7 and validate the performance with the couple of our network and wrench-based refinement. **Baseline** is the same as the first experiment, our wrench-based refinement is denoted as **WBR**. **Ours w/o WBR** replaces **WBR** with directly closing fingers as **Baseline** does, while **Noisy GT w/ WBR** stands for using ground truth hand poses with some Gaussian noise to enable our wrench-based refinement. As for the performance of our network, we can see that **WBR** significantly improves our network, based on the results of **Ours w/o WBR** and **Ours**. Furthermore, **Ours** gives a remarkably better performance than **Baseline**, and an at least not weaker performance than **Noisy GT** which can be regarded as a high-quality initialization.

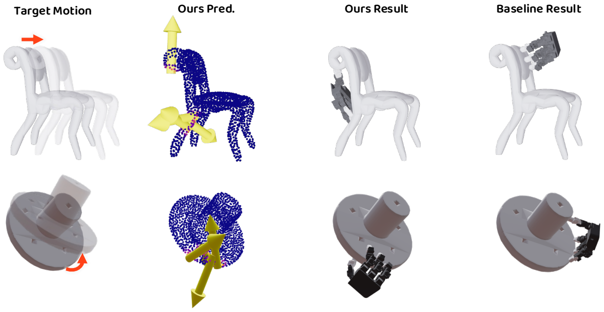

Fig. 9. Visualization of evaluation on rigid bodies

The success of **Ours** and the failure of **Baseline** can be illustrated in Fig. 9. As for a big object (the first row in Fig. 9), **Ours** pushes it on the back to move it to the right, while as for a small object (the second row in Fig. 9), **Ours** grasps its edge to lift it on one side. However, without **WBR**, **Baseline** can provide neither a contact solution to directly achieve the target motion, nor a force-closure grasp.

H. Generalization over Novel Robot Hands

Our network is trained on multiple robot hands, we evaluate if our model has the ability to generalize to novel robot hands, such as Allegro. In other words, we have not trained our model with the data annotated by Allegro hand. Using the Allegro hand, **Ours** attained an average accuracy of 94.0% on the test set, compared to 18.7% for the **Baseline** (described in), demonstrating effective generalization to novel robotic hands. So the generalization performance is shown in Tab. III to answer Q8. The challenge of the wrench-based task on Allegro is reflected by the results of **Baseline**. However, It is surprising to see that **Ours** achieves a competitive performance, proving that our network does have the potential of novel robot hands generalization.

Letter	Mean Dis(m)	Max Dis(m)	Success Rate
N	0.0210	0.0376	10/10
U	0.0186	0.0394	9/10
S	0.0183	0.0384	8/10

TABLE IV
Rope Rearrangement Experiment Results.

I. Real World Experiments

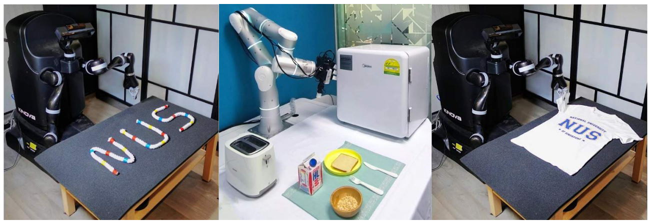

Fig. 10. Realworld experiment settings.

In this section, we evaluate our system's performance of manipulating various rigid, articulated rigid and deformable objects in several real-world settings.

Rope Rearrangement We design a rope rearrangement experiment for a Kinova MOVO robot arm to rearrange a random reset rope to letters "NUS". We perform 10 times for each letter in "NUS". Specifically, we take RGBD images for both the current scene and the goal scene, and get the rope point cloud with SAM [36]. We then approximate the rope point clouds with several cylinders so that we can calculate the per-point target motion by obtaining the cylinder's planar rotation and translation from current to goal scene. We iteratively grasp the point with the highest predicted contact heatmap value and move to the predicted target point. A test case visualization is shown in Fig. 11. As a result, our model achieves 90% success rate, as shown in Tab. IV, indicating that our model is accurate in generating grasp points for deformation objects like ropes.

Breakfast Preparation We design a complex breakfast preparation experiment for a single Flexiv robot arm and a LeapHand as the manipulator. In this setting, the robot will open the fridge door, take out the milk box, place the milk box on the table, pick up a piece of bread, put the bread into the toaster, open the toaster, take out the cooked bread and place it to the plate. Among these objects, the fridge door is an articulated object, while the milk box is a normal rigid body object, and the bread is treated as a 3D deformation object.

Cloth Folding We design a cloth folding experiment for a Kinova MOVO robot arm to fold a T-shirt. We extract the

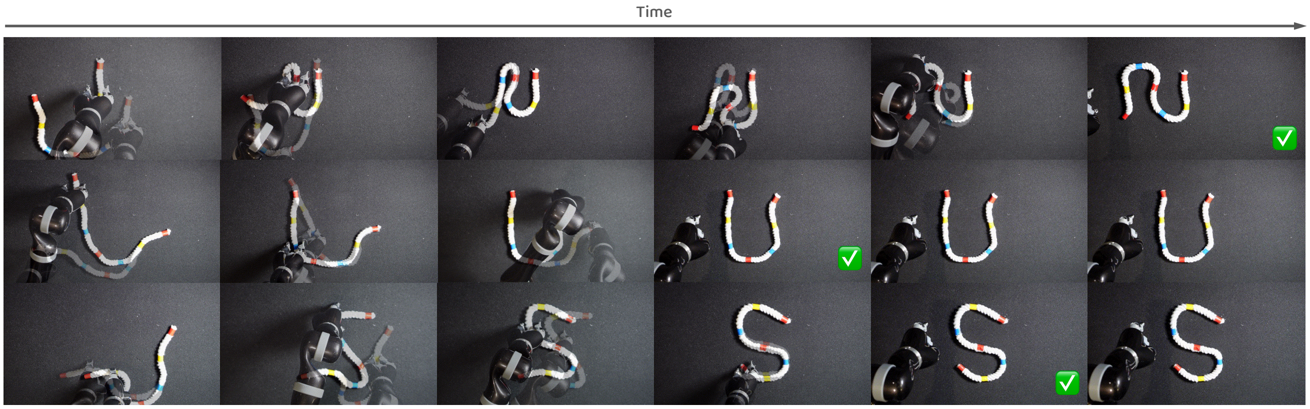

Fig. 11. Test Cases of Rope Rearrangement.

T-shirt point clouds using SAM [36]. We design 3 substages of clothes folding and obtain the per-point target motion in DiffCloth [26]. Movo will grasp the point with the highest heatmap value and move to the predicted target point.