

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 Tie³, Haozhuo Zheng⁴, Haoyu Zhou⁵, Weikun Peng¹, Debang Wang¹, Tianrun Hu¹, Tianyi Chen⁶, Zhouliang Yu⁷, 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] 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 Vision-Language Models (VLMs) serve as vast repositories of knowledge that can enhance robotic capabilities. For instance, LLMs have demonstrated superior capacity in decomposing complex tasks, such as meal preparation, into subtasks, significantly improving robot planning and reasoning processes. Recently, researchers have started investigating LLMs and VLMs in robotics to develop

* denotes equal contribution

† denotes the corresponding author

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

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

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

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

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

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

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



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.

robot foundation models. These efforts either leverage pre-trained LLMs or VLMs as high-level planners to execute a set of pretrained low-level skills [2]–[4], or train low-level robot actions on massive datasets [5]–[7]. While these models excel at high-level task planning and semantic understanding, they face challenges with the complexity of 3D object geometry, deformations, force constraints, and comprehending different robot manipulator topologies during low-level manipulations. In other words, the breadth and adaptability of low-level manipulation capabilities have received less attention compared to high-level task planning in robotics. Thus, it is 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.

To build a manipulation foundation model, the core problem is defining a unified task formulation for all robot manipulation tasks, similar to auto-regressive prediction in GPTs [8], mask prediction in BERT [9], and vision-language contrastive learning in CLIP [10]. This formulation must: 1) handle arbitrary objects, including articulated rigid objects, solid rigid objects, and deformable objects; 2) support a diverse range of manipulation tasks, such as grasping, pulling, re-orientating, and folding; 3) adapt to cross-platform integration, accommodating various robot designs and configurations; and 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 to meet the above requirements. We formulate the manipulation foundation task as a contact synthesis task. Specifically, the foundation model \mathcal{F} takes as input: 1) object point cloud and physical attributes (e.g., friction coefficients, Young’s modulus), 2) task descriptions as target motion, 3) robot manipulator descriptions as point clouds, and 4) manipulation region masks if required (e.g., grasping a pan’s handle). The foundation model \mathcal{F} outputs the contact points, contact forces, or post-contact motion to enable the robot to manipulate the object towards its target motion. We generated a large-scale annotated dataset to train the model. We tested our ManiFoundation model in diverse experiments, verifying its effectiveness in both simulations and real-world applications. In summary, our primary contributions are:

- We propose a comprehensive framework for foundation models in general-purpose robotic manipulation via contact synthesis, achieving broad generalization across various objects, robots, and tasks.
- We implement the ManiFoundation model, which includes neural network backbones for visual and physical feature extraction and an iterative contact wrench optimization method to generate feasible robot poses while avoiding collisions and penetrations.
- We develop a pipeline to generate a large-scale annotated dataset covering diverse objects and robot hands, which will be publicly available on our project website.
- We conduct extensive experiments to verify the effectiveness of our framework in both simulations and real-world settings, covering various articulated rigid and deformable object manipulation tasks.

II. RELATED WORK

A. Foundation Models in Robotics

Foundation models in robotics [11] have recently garnered significant attention. SayCan [2] grounds LLMs using value functions of pretrained skills. VoxPoser [3] combines affordances and constraints from LLMs and vision-language models into 3D value maps for motion planners. Codes-as-policies [4] leverages LLMs to write robot policy code from natural language instructions. Robotics Transformer (RT-1) [5] was trained on 130k real-world data points, covering 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] collects a dataset from 22 robots with 527 robotic skills, showing that scaling data improves generalization across multiple robot platforms. Our work focuses on developing a foundation model specifically designed for manipulation skills via contact synthesis, supporting general robotic manipulation with broad generalization across diverse deformable objects, robots, and tasks.

B. Learning Visual Affordances

Learning affordances [12]–[14] has received increasing attention for robotic manipulation, including grasping [15]–[17] and articulated object manipulation [18]–[20].

Where2Act [18] predicts the actionable point-on-point cloud for primitive actions such as pushing and pulling. Vision-Robotic Bridge (VRB) [21] 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 divided into two parts: a task-planning module that determines the object trajectory, and a manipulation module that guides the robot to achieve this trajectory. For any given task, we assume LLMs/VLMs, other models, or differentiable simulators with digital twins can convert it into a sequence of object point clouds as subgoals. Examples of this step for tasks in our experiments are provided on our website. Note that this aspect is not our primary focus. Here, we aim to develop a universal model that guides robots to manipulate objects towards the target pose or shape. The model should generalize across arbitrary objects, robots, and tasks (target shape or motion).

We present our ManiFoundation model as a solution to the *contact synthesis* problem. Specifically, the model \mathcal{F} takes as input: 1) manipulated object descriptions, including the object point cloud and physical properties; 2) task descriptions as target motion; 3) robot descriptions as point clouds; and 4) manipulation region masks if needed (e.g., a pan’s handle). The model outputs contact points and contact forces to manipulate the object towards its target motion. Formally, let a point cloud with N points and normals on each point be $\mathcal{P} = (x_1, n_1), \dots, (x_N, n_N) \in \mathbb{R}^{N \times 6}$, 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 clouds, 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 a contact solution $M_C, M_S = \mathcal{F}(\mathcal{P}_o, \mathcal{P}_h, \mathcal{M}, \mathcal{R}, \Gamma)$ that satisfies:

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

where $M_C = \{c_i\}_{i=1}^H, c_i \in \mathbb{R}^3$ is the predicted contact points with H points on the object point cloud, and $M_S = \{s_i\}_{i=1}^H, s_i \in \mathbb{R}^3$ 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. Figure 2 provides a pipeline overview. The following sections describe each module in detail.

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 to generate contact points

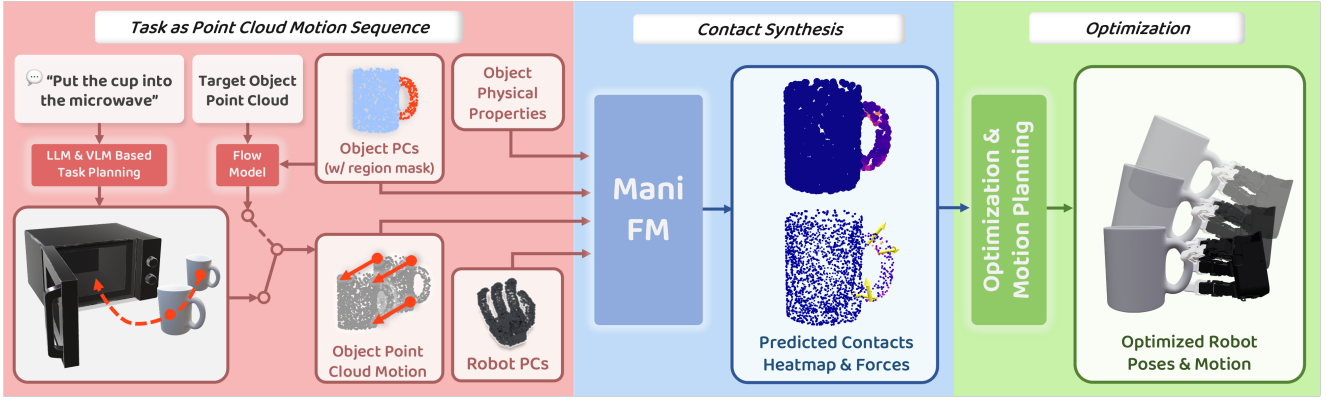


Fig. 2. The pipeline of our ManiFoundation model. Left: we decompose a manipulation task to a sequence of object point cloud motions from either VLM-based planning or a flow model. Middle: we train a ManiFoundation network to predict the contact point and force heatmap for each motion of the sequence.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

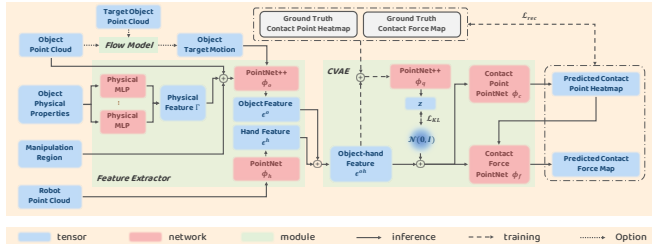


Fig. 3. Overview of our ManiFoundation network. The feature extractor module incorporates the information from both object and robot point clouds, and the CVAE module generates the contact point and force maps on the given object.

and 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, using *PointNet++* [22] (ϕ_o) for \mathcal{P}_o and *PointNet* [23] (ϕ_h) for \mathcal{P}_h . We chose *PointNet* for the manipulators due to its simpler architecture and fewer parameters, making it suitable for the limited variety of manipulators. In contrast, *PointNet++* is better suited for extracting features from the more diverse and complex object point clouds. To address 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 wide variety of objects, each with distinct physical properties that are crucial for manipulation. These properties dictate different manipulation strategies. For example, a handkerchief and a card of the same shape require different approaches: spinning a handkerchief involves grasping the four corners, while rotating a card involves applying friction to other areas. These differences necessitate encoding physical attributes to distinguish between different types of objects, leading to varied contact points and force solutions.

In this work, we encode different physical properties with different Multi-layer Perceptions (MLPs) $\psi_i : \mathbb{R}^1 \rightarrow \mathbb{R}^{10}$, $i = 1, \dots, 6$. The same MLP processes the same physical properties across different points. Specifically, we encode friction coefficients for all objects, density and elasticity coefficients [24] for 2D deformable objects, and Young’s modulus and Poisson’s ratio [25] 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 $\Gamma = MMP(\psi_1(\gamma_1), \dots, \psi_6(\gamma_6)) \in \mathbb{R}^{10}$, where γ_i is the i -th physical property.

This design offers several benefits. First, it can handle objects with a wide range of physical properties. Second, if additional properties are needed to identify new object types, new MLPs can be easily added to map these properties to feature vectors. These new vectors are then combined with existing ones and fed into the mean pooling layer.

b) *Manipulation Region Selection*: Certain manipulation tasks have constraints that specify which parts of an object can be manipulated, often due to human preferences or environmental factors. For example, robots are typically preferred to grasp the handle of a teapot or pan and cannot grasp the bottom of a cup when it’s on a table. To address these constraints, we design a manipulation region mask, denoted as $\mathcal{R} = \{r_i\}_{i=1}^N \in \mathbb{R}^N$. $r_i = 1$. Here, $r_i = 1$ indicates that the i -th point of \mathcal{P}_o is allowed to contact, while $r_i = 0$ means it is not. This mask can be assigned by humans or generated based on environmental constraints.

For deformable objects, specifying point-wise target motion is challenging. We use a scene flow model [26] to process both current and target point clouds to deduce the target motion. For articulated or rigid objects, determining the target motion 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:

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

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 $\epsilon^{oh} = \{\epsilon_i^{oh}\}, i = 1, \dots, N_o + N_h$. Next, the following CVAE will fuse them to get the final output.

2) *CVAE for Contact Point and Force/Motion*: In practice, multiple contact points 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. We chose CVAE instead of diffusion models for its simpler structure, faster inference speed, and multimodal generation capabilities as demonstrated in our experiments.

Specifically, during training, the ground truth contact point heatmap $M_{C'}$ and force/motion map $M_{S'}$ together with ϵ^{oh} are encoded into latent variables $z \in \mathbb{R}^{64}$ by a posterior *PointNet++* ϕ_q : $z = \phi_q(\epsilon^{oh}, M_{C'}, M_{S'})$, 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 *PointNets* ϕ_c and ϕ_f to predict the contact point heatmap M_C and contact force/motion map. We then use non-maximum suppression (NMS) [27] 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_C as an extra input, we predict the force map M_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:

$$\mathcal{L} = \lambda_C \mathcal{L}_{MSE}(M_{C'}, M_C) + \lambda_S \mathcal{L}_{MSE}(M_{S'}, M_S) + \lambda_{KL} \mathcal{D}_{KL}(\phi_q(\epsilon^{oh}, M_{C'}, M_{S'}) \parallel \mathcal{N}(0, I)), \quad (3)$$

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

B. Robotic Hand Pose Refinement

Given the contact points predicted by our network, we can directly manipulate deformable objects by simply grasping the contact points to achieve the target motion. However, 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. For a multi-fingered hand, we iterate all possible matches between fingertips and contact points (e.g., $4! = 24$ matches for a four-fingered hand). For each match, we fix the robot hand in its neutral joint values as a rigid object and apply the

Iterative Closest Point (ICP) algorithm to align the fingertips with the predicted contact points. The resulting hand 6D pose for each match is used to generate a contact heatmap, and the best match, based on the highest Intersection over Union (IoU) score between the generated and predicted heatmaps, is selected. This initial alignment provides a good starting pose for the robot palm, roughly aligning fingertip positions with predicted contact points, thereby shortening subsequent optimization time and improving success rates.

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

$$\hat{\mathbf{w}} = \frac{2}{\Delta t^2} \begin{bmatrix} \Delta\mathbf{x}/M \\ \Delta\theta/I \end{bmatrix}, \quad (4)$$

$$\mathbf{w} = \hat{\mathbf{w}} / \|\hat{\mathbf{w}}\|, \quad (5)$$

where object mass and inertia are denoted as M and I , which are computed approximately by estimating the volume of the object. We normalize $\hat{\mathbf{w}}$ to target wrench \mathbf{w} for simplicity. In practice, we care more about the direction of the 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 \mathbf{f}^* exerted by m fingertips onto the object to minimize the norm of wrench residual \mathbf{r} in Eqn. 6.

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

$$\mathbf{f}_{1:m}^*(\mathbf{w}, \mathbf{q}) = \arg \min_{\mathbf{f}_{1:m}} \left\| \mathbf{r}(\mathbf{w}, \mathbf{q}, \mathbf{f}_{1:m}) \right\|^2, \quad (7)$$

$$\text{s.t. } \mathbf{f}_{1:m} \in FC(\mu). \quad (8)$$

where $G_{1:m}$ is the grasp mapping matrices that maps 3d contact force at fingertips to 6d wrench applied to the object. $FC(\mu)$ is the friction cone with coefficient μ guaranteeing that contact forces satisfy Coulomb's law [28]. During optimization, fingertips are not required to always be in contact with the object surface. If fingertip i is away from the surface, G_i is calculated from its position \mathbf{p}_i and the normal vector at its projection onto the surface $\tilde{\mathbf{n}}_i$.

The optimization problem to improve grasp quality is:

$$\min_{\mathbf{q}} \left\| \mathbf{r}^*(\mathbf{w}, \mathbf{q}) \right\|^2 + \sum_{i=1}^m \left| \Phi(\mathbf{p}_i(\mathbf{q})) \right|^2, \quad (9)$$

$$\text{s.t. } \mathbf{q} \in [\mathbf{q}_{min}, \mathbf{q}_{max}], \quad (10)$$

$$\Phi(\mathbf{c}_{1:n}(\mathbf{q})) \geq \epsilon_{1:n}, \quad (11)$$

$$\left\| \mathbf{c}_i(\mathbf{q}) - \mathbf{c}_j(\mathbf{q}) \right\| \geq \epsilon_i + \epsilon_j, \quad (12)$$

where $\mathbf{r}^*(\mathbf{w}, \mathbf{q}) = \mathbf{r}(\mathbf{w}, \mathbf{q}, \mathbf{f}_{1:m}^*(\mathbf{w}, \mathbf{q}))$ is the minimum wrench residual. Our optimization objective in Eqn. 9 comprises two terms. The first term is the minimized norm of wrench residual, which measures the current grasp's capability to generate the desired object motion. The second term is the sum of the distance between fingertips and the object's surface, calculated using the object's signed distance

field (Φ) and the fingertips' positions $\mathbf{p}_{1:m}$ obtained from forward kinematics. Specifically, for any 3D point \mathbf{x} , we locate the nearest points $\bar{\mathbf{p}}$ on the object's surface and the outward normal $\bar{\mathbf{n}}$. $\Phi(\mathbf{p}) = \bar{\mathbf{n}} \cdot (\mathbf{p} - \bar{\mathbf{p}})$, with $\frac{\partial \Phi(\mathbf{x})}{\partial \mathbf{x}} = \bar{\mathbf{n}}$.

We also add a set of constraints to make the resulting hand pose practical. Eqn. 10 describes the robot joint limits. For simplicity, we use a set of n spheres with different radius $\epsilon_{1:n}$ to represent the collision shape of the robot hand. The spheres are attached to the fingers and palm, so the centers' positions $\mathbf{c}_{1:n}(\mathbf{q})$ can be obtained from forward kinematics. Eqn. 11 guarantees no penetration between the hand and object and Eqn. 12 guarantees no self-penetration.

We solve this nonlinear optimization problem iteratively by performing Taylor expansion to Eqn. 9-12 in the neighborhood of \mathbf{q} , $\mathbf{f}_{1:m}^*$ and obtaining a convex optimization problem (second-order cone programming) which could be solved quickly by CVXPY [29] as follows:

$$\min_{\delta \mathbf{q}, \delta \mathbf{f}_{1:m}} \left\| \mathbf{r}^*(\mathbf{w}, \mathbf{q}) + \sum_{i=1}^m \frac{\partial G_i(\mathbf{q})}{\partial \mathbf{q}} \mathbf{f}_i^* \delta \mathbf{q} + G_i(\mathbf{q}) \delta \mathbf{f}_i \right\|^2 \quad (13)$$

$$+ \sum_{i=1}^m \left| \Phi(\mathbf{p}_i) + \frac{\partial \Phi(\mathbf{p}_i)}{\partial \mathbf{p}_i} \frac{\partial \mathbf{p}_i(\mathbf{q})}{\partial \mathbf{q}} \delta \mathbf{q} \right|^2, \quad (14)$$

$$\text{s.t. } \mathbf{f}_{1:m}^* + \delta \mathbf{f}_{1:m} \in FC(\mu), \quad (15)$$

$$\mathbf{q} + \delta \mathbf{q} \in [\mathbf{q}_{min}, \mathbf{q}_{max}], \quad (16)$$

$$\Phi(\mathbf{c}_{1:n}) + \frac{\partial \Phi(\mathbf{c}_{1:n})}{\partial \mathbf{c}_{1:n}} \frac{\partial \mathbf{c}_{1:n}(\mathbf{q})}{\partial \mathbf{q}} \delta \mathbf{q} \geq \epsilon_{1:n}, \quad (17)$$

$$\left\| \mathbf{c}_i - \mathbf{c}_j + \left(\frac{\partial \mathbf{c}_i(\mathbf{q})}{\partial \mathbf{q}} - \frac{\partial \mathbf{c}_j(\mathbf{q})}{\partial \mathbf{q}} \right) \delta \mathbf{q} \right\| \geq \epsilon_i + \epsilon_j, \quad (18)$$

$$|\delta \mathbf{q}| \leq s_q, \quad |\delta \mathbf{f}_{1:m}| \leq s_f. \quad (19)$$

In the above equations, the Jacobians $\frac{\partial \mathbf{p}(\mathbf{q})}{\partial \mathbf{q}}$ and $\frac{\partial \mathbf{c}(\mathbf{q})}{\partial \mathbf{q}}$ are provided by the simulation Jade [30]. The exact formula of grasp mapping matrices and the corresponding gradients are derived in supplementary. Besides the constraints derived from the original problem Eqn. 9-12, we also add Eqn. 19 restricting $\delta \mathbf{q}$ and $\delta \mathbf{f}_{1:m}$ in the neighborhood of zero, so that Taylor expansion is reasonable. We then update the joint values of the robot hand by $\mathbf{q} \leftarrow \mathbf{q} + \delta \mathbf{q}^*$. The convergence conditions are $\|\mathbf{r}^*(\mathbf{w}, \mathbf{q})\| < 10^{-6}$ and $|\Phi(\mathbf{p}_{1:m}(\mathbf{q}))| < 0.005$. We set the maximum number of iterations as 100. The optimization is successful if it reaches convergence within 100 iterations, otherwise, it fails and terminates after 100 iterations. The successful output of this iterative convex optimization is the joint values of the robot hand and associated torques to generate the wrench closest to the target wrench.

We can provide force-closure grasp optimization by setting $\mathbf{w} = [0, 0, 0, 0, 0, 0]^T$, 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 [28].

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, $\mathbf{r}^J(P^J, \mathbf{w}, \mathbf{q}, \mathbf{f}_{1:m}) = \sum_{i=1}^m P^J(G_i(\mathbf{q})\mathbf{f}_i) - \boldsymbol{\tau}$, where $\boldsymbol{\tau}$ is the target joint torque and P^J is the projection matrix. If we apply a wrench \mathbf{w} to the articulated object, the resulting torque at the joint will be $P^J\mathbf{w}$. Take the revolution joint as an example, where \mathbf{a} is the axis direction and \mathbf{o} is any point on the axis. The

corresponding projection matrix is $P^J = [(\mathbf{o} \times \mathbf{a})^T, \mathbf{a}^T]$. 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. Refer to our website for more details.

a) Training Set: For fair and convenient network comparison, we select a small $\sim 3\text{K}$ training dataset, consisting of $\sim 1\text{K}$ rigid, $\sim 1\text{K}$ clothes, and $\sim 1\text{K}$ 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 $\sim 200\text{K}$ training dataset to evaluate whether our model can scale up with more data.

b) Testing Set: To systematically evaluate the 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. Each test set contains ~ 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 model output?
- Q4: Can our CVAE-based network generate diverse proposals with the same input?
- Q5: Can manipulation region mask constrain the network contact heatmaps and the optimization result?
- Q6: How effective is our grasp optimization?
- Q7: Does combining network and optimization improve task-oriented grasp generation? (See Supp.)
- Q8: Can our model generalize to novel robot manipulators? (See Supp.)
- Q9: How does our proposed framework perform on real robot experiments with real point clouds? (See Supp.)

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,

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.

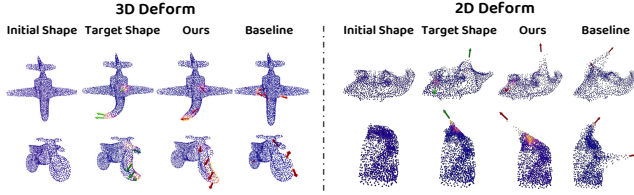


Fig. 4. Visualizations of evaluations on deformable objects are shown. The point cloud colors represent the contact heatmap, with purple indicating low values and yellow indicating high values. Arrows denote the force or motion direction. This representation is consistent across all figures.

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 [24] for 2D deformable objects and SoftMAC [25] for 1D/3D deformable objects. We apply contact forces to the object and obtain the rollout point cloud in simulators. The success is determined by the distance between the rollout point cloud and the target point cloud. Detailed thresholds and parameters are in Supp.

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 fingertip using Eqn. 6. We chose such a baseline because we are the first to train a unified model for contact synthesis on objects of various types. **Ours(SD)** stands for training our model only on Single Dataset ($\sim 1K$ rigid, $\sim 1K$ clothes, $\sim 1K$ 3D deformation respectively instead of together). **Ours** is for our network trained on a $\sim 200K$ dataset training data.

To answer Q1, as shown in Tab. I, **Ours** has an average success rate of over 90, 85, 80% over rigid, 3D deformable, 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 illustrate 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 performance loss, compared to training on a single object type.

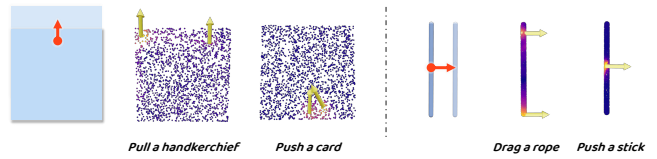


Fig. 5. Visualizations of how physical properties affect network outputs. Blue squares and cylinders are objects. Overlapping images with transparency represent the moved object. The orange arrow represents the motion.

2) **Physical Properties:** To understand the impact of physical properties (in Q3), we conducted two experiments shown in Fig. 5, evaluating how variations in physical properties influence our model’s behavior. We used two shapes: a thin square and a long cylinder. The square represented a handkerchief (deformable) or a card (rigid). The cylinder represented a rope (deformable) or a stick (rigid). We used identical point clouds and motions, altering only the physical properties to observe the outcomes. Fig. 5 shows that for the thin square, our model pulls the forward corners for the handkerchief, reflecting its pliability, and pushes from the rear for the rigid card. For the long cylinder, our model drags both ends of the rope and pushes the middle of the stick. These results demonstrate that our model adjusts its outputs based on the physical properties, indicating it learns to reason about object dynamics.

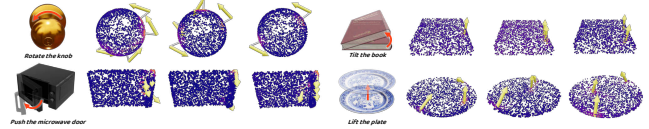


Fig. 6. Visualization of multimodal contact syntheses generated by our CVAE-based model.

3) **Multimodal Syntheses:** To answer Q4—whether our model can generate diverse solutions from the same input—we tested four novel objects not in our training dataset. For each object, we selected a common motion, such as rotating a knob or tilting a book. Using these inputs, we sampled different $z \in \mathbb{R}^{64}$ values from $\mathcal{N}(0, I)$ and visualized the outputs. As shown in Fig. 6, our CVAE module allows our model to generate diverse, physically plausible contact outputs. For example, to rotate a knob, the model provides solutions like holding the upper or the lower side.

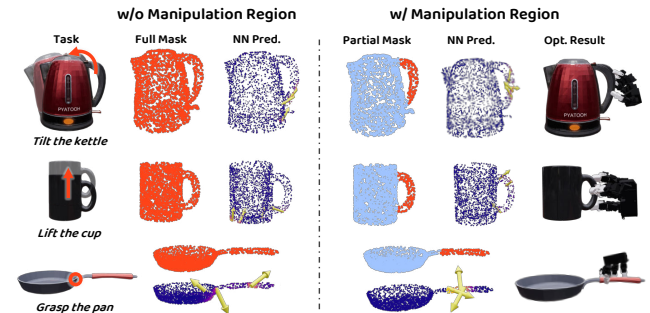


Fig. 7. Visualizations of how manipulation region mask input affects our model prediction and the optimization result.

4) **Manipulation Region Selection:** To explore whether a manipulation region mask can 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

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

TABLE II

HAND POSE REFINEMENT AND DIRECT GRASP COMPARISONS. T AND F INDICATE TASK-ORIENTED GRASP AND FORCE-CLOSURE GRASP.

handles. We assessed the model’s performance with and without a manually selected manipulation region mask, and the results are shown in Fig. 7. The mask guides the model to favor these regions. Without a mask, it may suggest contact points suitable for the intended motion but impractical for manipulation. For instance, lifting a cup from its bottom on a table is unfeasible. Applying a mask to the handle helps the model propose viable contact points, preventing potential collisions. The effectiveness of this approach is illustrated by the robot hand poses generated by our model, shown on the right side of 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 and force-closure grasps. Task-oriented grasp requires the grasp pose to produce a specific target wrench. As derived in Sec. III-B.3, we use the target wrench as the manipulation task description for rigid objects. Our evaluation includes both task-oriented and force-closure grasp scenarios over 1000 objects in our dataset, generating 20 distinct grasps per object, totaling 20k task-oriented and 20k force-closure grasps. In task-oriented evaluations, the target wrench for each grasp is randomly sampled. 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 initialize each method identically. We retrieve a ground truth 6D palm pose from our dataset that can exert the target wrench or achieve force-closure. This 6D palm pose is placed 30cm away from the object, and then a random 60-degree rotation and 5cm translation are added, with all finger joints set at rest positions. Under this initialization, we run our robotic hand pose refinement and compare it with a direct grasp method. The direct grasp method moves the robot hand directly towards the object’s center, closes the fingers upon contact, and calculates the forces for each fingertip using Eqn. 6.

Our analysis includes three types of robotic hands: Barrett [31], LeapHand [32], and ShadowHand [33]. As shown in Table II, our refinement module achieves significantly higher success rates than the baseline for both scenarios and all tested hands.

C. Real World Experiments

In this section, we evaluate our system’s performance in manipulating various rigid objects, clothes, and 3D de-

formable objects in real-world settings, as shown in Fig. 1. Detailed descriptions are provided in the Supp. **Rope Rearrangement** We designed an experiment for a Kinova MOVO robot arm to rearrange a randomly reset rope into the letters ”NUS.” We performed the task 10 times for each letter, achieving a 90% success rate, indicating our model’s accuracy in generating grasp points for deformable objects like ropes. **Breakfast Preparation** We designed a complex breakfast preparation experiment using a single Flexiv robot arm with a LeapHand manipulator. The tasks included opening the fridge door (articulated object), taking out the milk box (rigid object), placing the milk box on the table, picking up a piece of bread, putting it into the toaster, opening the toaster, and placing the cooked bread on a plate. The bread is treated as a 3D deformable object. **Cloth Folding** We designed a cloth folding experiment for a Kinova MOVO robot arm to fold a T-shirt. The robot grasps the point with the highest heatmap value and moves to the predicted target point.

VI. CONCLUSION AND LIMITATIONS

We introduce a framework for developing a manipulation foundation model for contact synthesis. This model takes inputs like point clouds, physical properties, target motions, and region masks to output contact points and forces or motions. Tested on various objects, including articulated, rigid, and deformable items, our model achieves average success rates of around 90% in both simulations and real-world settings.

Our model can be improved in the following aspects. First, our framework focuses on quasi-static tasks by separating the trajectory into discrete motions at each time step. Future work could extend the framework to high-dynamic tasks by considering multi-step motions or the entire trajectory at each timestep. Second, our model focuses on single-hand fingertip contact. Future work could extend the framework to multi-hand surface contact situations. Thirdly, our current model has about 5 million parameters. In the future, we plan to scale up by increasing the amount of data and the number of network parameters.

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. Chormanski, T. Ding, D. Driess, A. Dube, 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] 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.
- [13] 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.
- [14] 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. Kulis, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 1916–1927. [Online]. Available: <https://proceedings.mlr.press/v205/lin23c.html>
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak, “Affordances from human videos as a versatile representation for robotics,” 2023.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] 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.
- [28] S. P. Boyd and B. W. P. “Fast computation of optimal contact forces,” *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1117–1132, 2007.
- [29] 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.
- [30] 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.
- [31] B. Technology, “Barrett bh8-282 3-fingered gripper,” 2019. [Online]. Available: <https://www.barrett.com/about-barretthand>
- [32] K. Shaw, A. Agarwal, and D. Pathak, “Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning,” *Robotics: Science and Systems (RSS)*, 2023.
- [33] Bwonikrobotics, “Barrett bh8-282 3-fingered gripper,” 2019. [Online]. Available: <https://www.allegrohand.com/>
- [34] 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.
- [35] Z. Yu, J. Fu, Y. Mu, C. Wang, L. Shao, and Y. Yang, “Multireact: Multimodal tools augmented reasoning 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>
- [36] 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.
- [37] 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.
- [38] 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.
- [39] 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.
- [40] Q. Zhou and A. Jacobson, “Thing10k: A dataset of 10,000 3d-printing models,” *arXiv preprint arXiv:1605.04797*, 2016.
- [41] 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.
- [42] 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.
- [43] 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.
- [44] 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. 20428–20438.
- [45] 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.
- [46] 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.
- [47] 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.