One-Shot Manipulation Strategy Learning by Making Contact Analogies

Yuyao Liu[1,2*], Jiayuan Mao[1*], Joshua B. Tenenbaum[1], Tomás Lozano-Pérez[1], Leslie Pack Kaelbling[1]
[1]Massachusetts Institute of Technology
[2]Institute for Interdisciplinary Information Sciences, Tsinghua University
liuyuyao21@mails.tsinghua.edu.cn {jiayuanm,jbt}@mit.edu {tlp,lpk}@csail.mit.edu.

***Abstract*— We present a novel approach, MAGIC (manipulation analogies for generalizable intelligent contacts), for one-shot learning of manipulation strategies with fast and extensive generalization to novel objects. By leveraging a reference action trajectory, MAGIC effectively identifies similar contact points and sequences of actions on novel objects to replicate a demonstrated strategy, such as using different hooks to retrieve distant objects of different shapes and sizes. Our method is based on a two-stage contact-point matching process that combines global shape matching using pretrained neural features with local curvature analysis to ensure precise and physically plausible contact points. We experiment with three tasks including scooping, hanging, and hooking objects. MAGIC demonstrates superior performance over existing methods, achieving significant improvements in runtime speed and generalization to different object categories. Website: https://magic-2024.github.io/.**

I. Introduction

A hallmark of human intelligence is flexible tool use: humans can quickly acquire new manipulation "strategies" from just a handful of demonstrations and apply these strategies across various scenarios, including generalization to novel objects of unseen categories. For example, as illustrated in Fig. 1, even from a single demonstration of using a hook to reach distant objects or putting hangers on a rod, we can generalize to different object positions, sizes, and diverse categories, such as hangers and mugs.

Traditionally, two main approaches have been widely studied to build machines that can flexibly use tools: model-based and analytic approaches which take novel scenarios and goals and use built-in physical models to compute plans [1]–[4], and policy learning, which leverages various types of priors (e.g., object-based and part-based models) and pretrained neural features for generalization [5]–[8]. However, both approaches have their limitations. Model-based planning generalizes well given accurate object and physical models. However, it is slow and usually does not benefit from learning. Policy learning approaches, on the other hand, are very efficient at performance time but usually exhibit limited generalization to novel objects and scenarios, particularly when the shape of the novel objects differs significantly from objects seen during training, such as generalizing from hangers to mugs.

In this paper, we present a novel approach, MAGIC (manipulation analogies for generalizable intelligent contacts), for one-shot manipulation strategy learning. Shown in Fig. 1, given a *single* reference action trajectory (e.g., using a hook to reach for a distant object) and a novel scenario (e.g., with different tools and different objects), the goal of the algorithm is to generate a sequence of robot actions that apply a "similar" strategy to the test objects specified by users: in this example, having the target object moving along a certain direction for a given distance. MAGIC extends two critical insights into a broad class of manipulation strategies. First, many strategies such as hooking, hanging, hammering, pushing, reaching [9], [10], stacking, pouring [11]–[13], and cutting [13] can be characterized by a sequence of contact waypoints (i.e., the order in which contacts between objects and robot bodies are made); second, these contacts are characterized by forceful affordances between object pairs: a specific pair of contact points on two objects would enable the application of forces along certain directions. However, searching for contact points that would enable the specific affordance is generally challenging due to complex constraints on reachability, collision avoidance, and motion stability.

MAGIC tackles these challenges by combining data-driven and analytic approaches to generate contact waypoints in novel scenarios. In particular, it first extracts the sequence of contacts among objects in the reference trajectory and then proposes (pairs of) contact points that have similar global and local shape properties as the contact points in the reference, which can be used as guidance for motion planning or motion retargeting. Finally, it leverages a physical simulator to select the contact and the action sequence that achieves the goal. Our key innovation lies in a novel global-to-local matching algorithm to find functional correspondences between the target objects and reference objects. Intuitively, a "good" contact point would satisfy both a global and a local matching property. First, the points on two objects should be on similar parts of the global shape (e.g., in the hook-using example, we need a contact point on the tool that is at the end of a long rod). Second, the hooking contact point should have a matched local curvature with the target object being hooked so that we can execute the actions stably. Therefore, we propose to use a pretrained visual feature-based correspondence matching to resolve the global matching property. This enables us to quickly search over different parts of the objects but the resulting contact point is usually not precise. Next, we use a local curvature-based matching algorithm to find the best contact point within a local region of the previously proposed

* indicates equal contribution. Work done while Yuyao Liu was a visiting student at MIT.

contact point, which gives us precise and physically plausible (e.g., collision-free and physically stable) contacts.

Overall, MAGIC tackles the problem of one-shot manipulation strategy learning by making analogies in contact waypoints. We validate the effectiveness of our approach on three challenging tasks: scooping a ball against a concave arc with a spoon, hanging a mug onto a mug tree, and using tools to hook objects of varying sizes. Compared to global shape-matching algorithms, our framework achieves significant improvements when the reference objects are from different categories than the test objects. Compared to local shape-matching and simulation-based approaches, our framework is orders of magnitude faster — for most test objects, we need to run simulations on fewer than three candidate contact points to find a solution. Finally, compared to pretrained feature-matching-based approaches, our method finds more precise and physically plausible solutions.

II. Related Work

Our algorithm is inspired by contact-based modeling techniques used in robotic manipulation. Various approaches have been developed for planning manipulations in contact space, involving both rigid bodies and robotic hands [14]–[22]. Techniques such as CMGMP [1], [2], [23] and the work of [3] perform search over the exponential space of possible contact sequences to determine possible contact modes using three basic types: fixed, separating, and sliding contact, while [4] learns the types of the contact sequence from a single demonstration to guide the planning around them. By contrast, in this paper, we focus on leveraging global and local shape matching to generate analogical contact points using a single demonstration, which significantly improves the runtime efficiency of existing methods as well as the generalizations to unseen objects.

Recently, researchers have explored learning object affordances [9], [11], [24]–[28], usually from a large set of demonstrations. In contrast, this paper focuses on generating trajectories with novel objects from a single demonstration leveraging pretrained visual features and shape analysis algorithms, requiring no additional training data.

For many years, people have attempted to harness shape information to guide tool usage through mathematical analysis [29]–[33] and data-driven approaches [5], [10], [34]–[37]. These methods often necessitate intricate human specifications or extensive in-domain datasets. In contrast, our approach integrates the off-the-shelf visual models, pretrained on general image datasets, with the generic geometrical property of curvature. This combination allows us to achieve effective and efficient tool-using guidance with minimal human intervention and without the need for large specialized datasets.

III. One-Shot Manipulation Strategy Learning

We propose MAGIC (manipulation analogies for generalizable intelligent contacts), a novel approach for fast and generalizable manipulation strategy learning from a single demonstration. Fig. 2 shows the overall framework. First, we extract the contact points among objects from the reference trajectory. Subsequently, we find the candidate contact points that support the functional affordances on unseen test objects through a global-to-local matching process, in which we first perform coarse correspondence point matching using pretrained vision transformer (ViT) features DINOv2 [38] (Section III-B), followed by a local alignment based on shape curvatures to find stable local contact patches (Section III-C). Finally, the candidate contact points and their matching scores can be used for tool selection, for retargeting reference object trajectories, or as waypoints for motion planning (Section III-D). The generated trajectories will be verified in a physical simulation and then output to the robot for execution.

A. Problem Definition

We introduce the task of one-shot manipulation strategy learning. Specifically, we start with a single demonstration involving the $\mathrm{SE}(3)$ trajectories of objects that execute a particular manipulation strategy to achieve a goal (e.g., hanging a hanger) in the reference scene. Our goal is to generate the trajectories of objects in a target scene that apply a similar manipulation strategy to a different object (e.g., attaching a mug to a mug tree). We simplify this problem by assuming the robot is only manipulating one of the objects (i.e., the "tool" that the robot is holding), and the desired motion can be explained by achieving a sequence of contact waypoints. Many rigid-body tool-using tasks such as hooking, poking, stacking, and funneling, all fall into this category. For now, we assume that the strategy to apply and the manipulanda in both the reference and target scenes (e.g., the hanger or the mug) are already identified, and later we will discuss strategies for selecting the best tool object to accomplish a certain task. Therefore, the problem can be cast as making the analogy between the reference object motion and the target object motion, taking into consideration other environmental constraints such as robot reachability and collisions.

Based on our insights into decomposing manipulation trajectories into contact point sequences, this task further reduces to establishing a functional correspondence between the demonstrated object and the novel object, which can be represented as an $\mathrm{SE}(3)$ transformation between two rigid bodies for each contact waypoint. For tasks considered in this paper, we only consider scenarios where there is a single contact waypoint responsible for the target motion (but it generalizes to finding correspondences in multiple waypoints), and there is a reduced degree of freedom that can be reasoned about in 2D. In particular, we only model contacts on a canonical 2D cross-sectional view of the objects. For example, when we consider hooking objects on a table, we only consider object motions and shape properties on the surface that is perpendicular to the table's surface normal. This is equivalent to assuming access to a canonical pose of the object: for hooks, this will be the top-down view when the hook is placed on the table; for mugs, this will be the side view that contains the handle of the mug. In this paper, we assume that this canonical view is given, and in general, it can be predicted by external modules. Therefore, now, our goal is to establish a single correspondence between two 2D

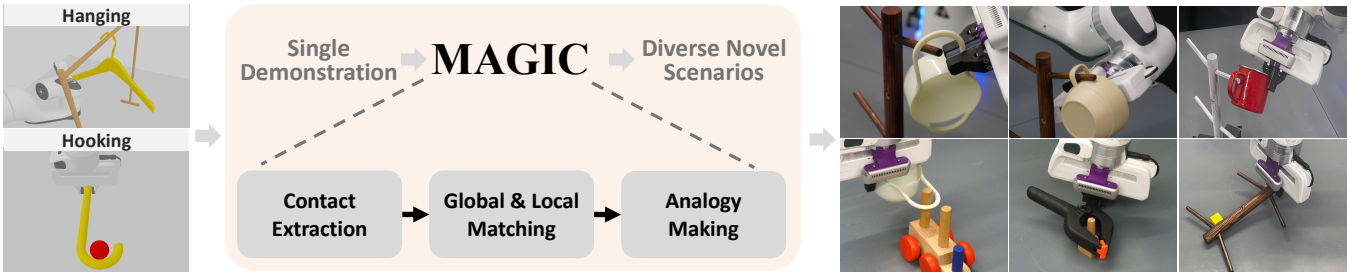


Fig. 1: We introduce MAGIC (Manipulation Analogies for Generalizable Intelligent Contacts), a pipeline that is capable of learning manipulation strategies from single demonstrations and applying them to novel objects.

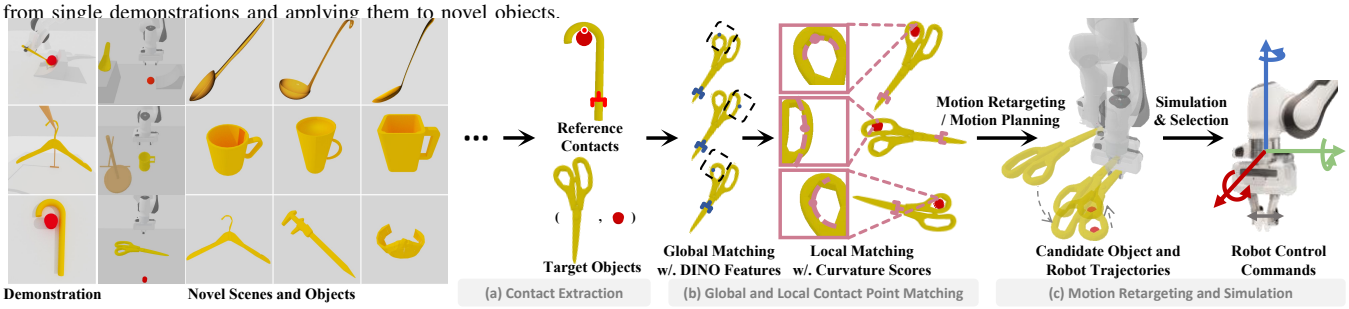


Fig. 2: The overall pipeline of MAGIC. (a) We first extract contact points from the reference trajectory. (b) Then, we compute a global and local contact point matching score to select candidate contact points on novel objects. (c) The generated contact points will be used for motion retargeting or motion planning, and the final motion will be simulated and verified by a physical simulator.

images of the objects in E(2) (translations, rotations, and reflections in 2D) plus scaling; and then we can recover the SE(3) correspondences based on the canonical object poses.

Overall, the input to MAGIC is an image I_T of the reference object T, an image $I_{T'}$ of the target object, and a point of interest p_T on the reference image. Our goal is to find the corresponding point $p_{T'}$ on $I_{T'}$. When we have two objects interacting: the tool object T being directly held by the robot (e.g., the hook) and another object O in contact with T (e.g., the target object we want to hook), our goal would be to find a contact point pair $p_{T'}$ on $I_{T'}$ and $p_{O'}$ on $I_{O'}$ given the reference $\langle T, O, p_T, p_O \rangle$ that maximizes a score function: $score(X, X')$, where $X = \langle T, O, p_T, p_O \rangle$ is the reference contact and $X' = \langle T', O', p_{T'}, p_{O'} \rangle$ is the target contact. In this paper, we employ a two-stage global-to-local matching process, therefore, the *score* function will be composed of two parts: a global matching score s_{dino}, and a local matching score s_{curv}:

$$score\left(X, X'\right) = s_{dino}(T, T', p_T, p_{T'}) + \lambda \cdot s_{curv}(X, X'),$$

where λ is a constant hyperparameter. p_O and p_T are automatically extracted using the contact information in simulation and will be manually annotated for real-world objects (they can also be automatically recovered using video-based contact point detector such as [39]). After generating top candidates for $p_{T'}$ and $p_{O'}$ based on *score*, for additional validation, we will simulate the computed trajectory with the target object and select the first one that succeeded in the task in simulation. All tasks studied in this paper involve only stable quasi-static motion. Therefore, they can be simulated using mild assumptions on point clouds, uniform object densities, and frictions. When the simulation is unavailable in challenging dynamic tasks (e.g., hammering), we fall back to using the highest-scoring contact match.

B. Global Contact Point Matching with Pretrained Features

The global matching score s_{dino} is only computed between the tool objects T and T'. In this stage, we will find a candidate set of contact points "globally" on the target object T', leveraging visual features pretrained on large image datasets for capturing global and semantic correspondences [40]. Specifically, we adopt DINOv2 [40] as our visual feature extractor. Fig. 3a shows the pipeline: we first extract visual features for both objects (with different image rotations and reflections) and then find a candidate set of matching points.

Feature extraction. We feed I_T and $I_{T'}$ into DINOv2 respectively, and get feature maps F_T and $F_{T'}$. In practice, to eliminate the effects of rotation and reflection in 2D, we apply horizontal flips and 12 rotations on $I_{T'}$ and get a total of 24 images $\{F^i_{T'}\}_{i=1}^{24}$. Next, we apply a principal component analysis (PCA) on the feature vectors. Given feature maps $F_T, F^i_{T'} \in \mathbb{R}^{n \times n \times d}$, let $flatten(F)$ be flattened version of F in $\mathbb{R}^{n^2 \times d}$, we have $W = \text{PCA}_{d,d'}\left\{flatten(F_T), \{flatten(F^i_{T'})\}_{i=1}^{24}\right\}$, where d and d' are the original feature dimension and the feature dimension after reduction, respectively, and $W \in \mathbb{R}^{d \times d'}$ is the matrix of principal component vectors. The output is $\widetilde{F} = F \times W \in \mathbb{R}^{n \times n \times d'}$.

Matching by patches. Next, we aggregate the features within a local region (instead of a single point) so as to increase the receptive field. Formally, let $patch(p)$ denote the $m \times m$ local area centered at point p, we find a point across all rotations of the target image $p_{T'}$ that maximizes the patch-aggregated cosine similarity with the reference point p_T:

$$s_{dino}\left(T, T', p_T, p_{T'}\right) = \sum_{\substack{p \in patch(p_{T'}), \\ p' \in patch(p_T)}} \left\langle \widetilde{F}^i_{T'}(p), \widetilde{F}_T(p') \right\rangle.$$

We select the top $k = 3$ matches with the highest s_{dino} to perform the local contact point matching in the next stage.

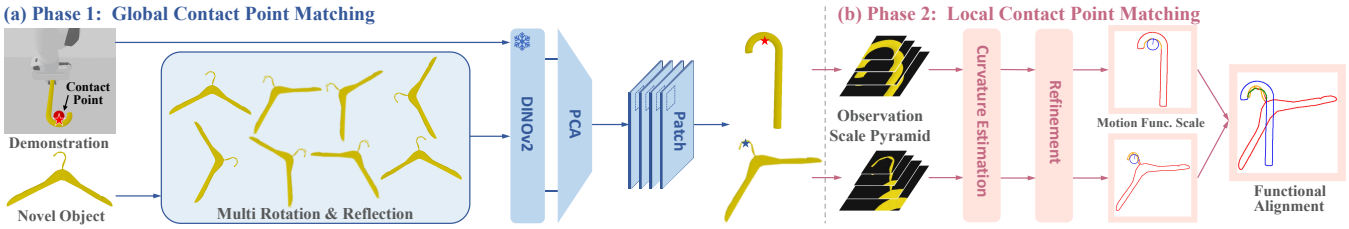


Fig. 3: Global and Local Contact Point Matching. The contact point matching process consists of two phases: (a) global matching using DINOv2 [38] features; (b) local matching involving multi-scale curvature estimation and refinements using irrelevant point suppression and convexity matching.

C. Local Contact Point Matching with Curvature Estimation

Although DINOv2 can propose contact points within a coarse global region, it is not accurate enough in terms of local geometric properties to support collision-free and stable physical motion. We leverage curvatures to refine the correspondence matching. Illustrated in Fig. 3b, our local alignment has four steps. We first construct a multiple observation scale pyramid: we estimate the curvatures of the reference contact point and the contact point proposed by DINOv2 on the target object at multiple scales. Then, based on the normal direction and the sign of the estimated curvatures, we perform irrelevant point suppression and convexity matching to find a physically plausible contact point for each scale. After that, we repeat the curvature estimation process on the updated contact points to get a more accurate curvature and normal direction. Finally, we accept the best contact point across scales.

Curvature estimation at a given scale. Given an image of the object with segmentation, we first use the Canny edge detector [41] to get the object edges. We define the *observation scale* as the radius of the region centered at the point of interest (e.g., the contact point) on the object edge. For a certain observation scale s of the point $\mathbf{c}$, we can use the edge points $\{(x_i,y_i)\}_{i=1}^n$ inside s to estimate the magnitude κ of the curvature and the radius of curvature r. Specifically, we first find the direction $\mathbf{x}'$ with the largest variance on $\{(x_i,y_i)\}_{i=1}^n$, and construct a local coordinate system $\mathbf{x}'\mathbf{y}'$ centered at $\mathbf{c}$, then represent $\{(x_i,y_i)\}_{i=1}^n$ in $\mathbf{x}'\mathbf{y}'$ to get $\{(x'_i,y'_i)\}_{i=1}^n$. Afterwards, we fit a parabola $y'=ax'^2$ on points $\{(x'_i,y'_i)\}_{i=1}^n$, i.e.,

$$a = \arg\min_a \sum_{i=1}^{n} \left(y'_i - a x'^2_i\right)^2,$$

then by the definition of curvature, we have $\kappa = 2|a|, r = 1/\kappa$..

Now, for a pyramid of observation scales $\{s_j\}_{j=1}^m$, we can compute a series of radii of curvature $\{r_j\}_{j=1}^m$. The *motion functional scale* is defined as s_j, where $j = \arg\min_{j'} \left|\frac{s_{j'}}{r_{j'}} - \alpha\right|$, where α is a parameter, and in practice, we select $\alpha = 3.5$ for all objects across all experiments. This constant corresponds to a scenario where the observation scale is similar to the radius of curvature estimated using the points inside the observation scale. Furthermore, the sign of the curvature (that is, whether the point on the curve is convex or concave) can be computed based on the mask of the object. Given both the tool object T' and target object O', we define the local matching score as:

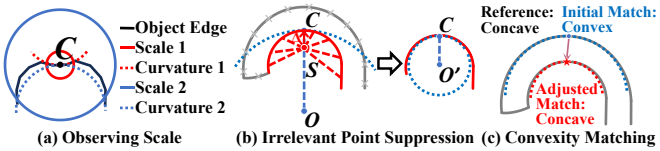


Fig. 4: **(a)** The observing scale at which we perform estimation has effect on both the magnitude and the sign of he estimated curvature. **(b)** Filtering out the irrelevant points can give a more accurate curvature estimation. **(c)** We perform local search with in the observation region to find the contact point with the correct convexity.

$$s_{curv}\left(\langle T,O,p_T,p_O\rangle,\langle T',O',p_{T'},p_{O'}\rangle\right) = \left|\frac{r(p_T)}{r(p_O)} - \frac{r(p_{T'})}{r(p_{O'})}\right|.$$

Irrelevant point suppression and convexity matching. Direct application of the curvature estimation algorithm is not robust enough for two reasons. First, the choice of the observation scale will significantly affect the estimation of the curvature. As illustrated in Fig. 4a, selecting a very small scale (scale 1) will make the estimation sensitive to local noise. Second, for large observation scales, irrelevant points (i.e., points on a different "edge" of the object) will be included and inject noise in the estimation, especially for thin objects as shown in Fig. 4b.

To eliminate these irrelevant points around the contact point of interest, we compute the curvature twice. First, we include all points within the observation scale. After computing the initial contact point C on the edge of the object and its curvature, we pick a point S that is at a small distance Δ from the contact point C along the direction of the radius of curvature $\overrightarrow{CO}$ in the initial estimation. Then, we select all points on the same "edge" as C on the side of S. This is done by emitting "rays" in all directions from S and for each ray selecting the point that is hit first. We then use these points to estimate the curvature again, as shown on the right of Figure 4b. Moreover, we ensure that the curvature sign (i.e., the convexity) of the corresponding point in the target image is consistent with that in the reference image by performing a local search within the observation region of the target image, which is depicted in Fig. 4c.

D. Generating Object Motions by Making Contact Analogies

So far we have presented a general mechanism for finding geometric functional correspondence between reference and target objects, and it can be used in many downstream modules. In the one-shot manipulation strategy learning setting, we consider three particular use cases.

Object motion retargeting. This is the most straightforward way to generate target object motions assuming there is

a single contact that accounts for the motion. Given the reference contact point p_T, the direction vector $\mathbf{v}$ of the view angle, the normal vector $\mathbf{n}_T$ of p_T estimated from curvature calculation, we can construct two local frames for the reference and the target object, and directly retarget the reference object motion by aligning two local frames.

Motion planning based on contact waypoints and force directions. We can generate object motion by leveraging analogical contact points as waypoints for a motion planner. For example, if our goal is to hang an object at a particular position, we can first compute the hanging pose by making analogies with the reference object pose and use a collision-free motion planner to generate the robot motion. This also applies to hook-using tasks where the goal is specified as to apply a particular force along a target direction on an object.

Tool selection. Our pipeline also supports tool selection, where the goal is to select one "tool" object (e.g., a hook) from a set of available objects that can best execute the strategy in the new scenario, given the other object to manipulate (e.g., the object to be hooked). We apply our algorithm on all available tools, and select the tool with the highest score.

IV. Experiments

In this work, we have conducted experiments both in simulation and in the real world. For simulation experiments, we adopt the simulation environment SAPIEN 2 [42] using a Franka Emika Panda arm. We evaluate different one-shot manipulation strategy learning algorithms on three tasks, as shown in Fig. 2: **(*Scooping*)** scooping balls of different sizes with various spoons against a concave arc, given a demonstration with a reference spoon; **(*Hanging*)** hanging a mug onto a mug tree, given a demonstration of hanging a hanger on a rod; **(*Hooking*)** selecting a tool from a set to hook objects of varying shapes and sizes, given a demonstration of hooking a ball with a hook. For all tasks, we have two variants: *Floating-Gripper (FG)* where the tool will be manipulated by a floating gripper, and the harder *Arm* variant where the tool needs to be grasped and manipulated by the robot. In this variant, we use a general antipodal grasp sampler [43] and RRT-Connect [44] for generating robot trajectories, of which we utilize MPlib* as the implementation. For *Hooking*, we also have a variant where there is only a single tool object provided to the robot (no tool selection). For all tasks, we only provide a single demonstration of object motion trajectories, the canonical view, and the pair of contact points on reference objects.

Baselines. We implemented two sets of baselines: global shape-matching and local shape-matching methods. For global shape-matching methods, we use two different methods: principal component analysis (PCA) and iterative closest point with FPFH features (FPFH+ICP) [45]–[47] on object point clouds to find the best transformation that would align the target object and the reference object. Then, we retarget the object motion or transform the waypoints. For local shape-matching baselines, we also implemented two methods: *DINO matching* which directly finds the best functional correspondence based on the DINOv2 features, and *curvature matching* which uses the random contact point sampling method from [4] and applies the curvature filtering algorithm from MAGIC to rank all contact points. Since this method involves random sampling of contact points and simulation, we cap its runtime to 180 seconds and use the contact point with the highest matching score found at that time. Note that all methods except for the sampling-based curvature-matching algorithm have very small variances across different runs (our algorithm is completely deterministic). Therefore, we do not include performance variances for different methods. We do not compare with methods such as NDF [5], KETO [9], and GIFT [10] in the main experiment, because they assume access to a collection of 3D object models for representation learning, but we provide a case study of comparing MAGIC with KETO in the following section.

*https://github.com/haosulab/MPlib

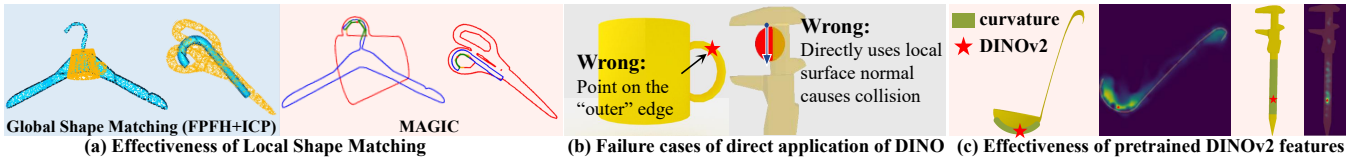
Method	Scooping		Hanging		Hooking		
	FG	Arm	FG	Arm	No Tool Sel.	FG	Arm
PCA	27.6	16.7	0.0	0.0	5.0	0.0	0.0
FPFH+ICP	32.9	27.2	0.0	0.0	6.7	0.0	0.0
DINO Matching	55.7	41.4	23.9	21.6	49.2	66.7	24.0
Curvature-Only	13.9	5.6	68.7	38.1	60.0	76.7	3.3
MAGIC	**94.4**	**65.4**	**92.9**	**84.3**	**65.0**	**100**	**63.3**

TABLE I: Average success rates (%) of 3 tasks. '*floating gripper* (FG)' indicates manipulation by directly setting positions and velocities of objects, focusing only on object-object interactions, excluding the robot. All methods except for Curvature-Only run almost deterministically. Curvature-Only has an average std. of 6.6%.

A. Experiment Results in Simulation

Table I summarizes the overall performances of variants of our methods and other baselines. The success rates are computed over 6 spoons and 3 objects for *Scooping*, 134 mugs for *Hanging* adopted from [5], and 4 tools and 5 objects for *Hooking* adopted from [4]. Overall, our model MAGIC performs the best. We break down our analysis into the following bullet points.

Local shape matching can significantly improve performance over global shape matching. We compare MAGIC with global shape matching methods, PCA and FPFH+ICP. As shown in Table I, global shape matching is ineffective because it lacks an understanding of the local contact points. Fig. 5(a) illustrates an example of the correspondence established by global shape matching (left) and MAGIC (right). To further demonstrate the effectiveness of local shape matching, we also provide an additional study on the mug-hanging task in Table II, where the demonstration includes the trajectory of hanging a held-out mug on the mug tree (i.e., intra-category transfer), rather than the hanger. In this setting, FPFH+ICP achieves a success rate of 41.1% (*c.f.* 94.0% for MAGIC). The results indicate that PCA and FPFH+ICP (as well as the DINO matching algorithm) can sometimes successfully achieve intra-category transfer by aligning overall shapes. However, they perform poorly at inter-category transfer due to the lack of local contact point alignment. By contrast, MAGIC excels in both intra-category and inter-category transfers by accurately identifying contact points that satisfy both global and local constraints.



(a) Effectiveness of Local Shape Matching **(b) Failure cases of direct application of DINO** **(c) Effectiveness of pretrained DINOv2 features**

Fig. 5: (a) Global shape matching fails to provide effective functional alignment due to the lack of awareness of local contact points. **(b)** Local curvature helps identify more accurate contact points and contact normals. **(c)** Pretrained DINOv2 features enhance data efficiency by selecting the optimal contact (the red star) and grasping points from those that curvature alone cannot distinguish (the green region). We also visualize the DINOv2 matching heatmaps.

	PCA	FPFH+ICP	DINO Match	Curvature-Only	MAGIC
Hanging (intra-cat.)	5.2	41.1	64.9	67.9	**94.0**
Hanging (inter-cat.)	0.0	0.0	23.9	68.7	**92.9**

TABLE II: Average success rates (%) of *Hanging* (intra-category transfer vs. inter-category transfer). While PCA and FPFH+ICP can occasionally achieve intra-category transfer, they fail to perform inter-category transfer. The experiments are conducted on the *FG* variant.

Local curvature matching is crucial, in addition to pretrained features. The comparison between MAGIC and other methods that do not use local curvature information (e.g., PCA, FPFH+ICP, and DINO Match) shows the importance of local curvature matching. As an illustration, as shown in Fig. 5(b), the contact points proposed solely by the pretrained model are likely to fail due to incorrect curvature sign and incorrect interaction direction. By contrast, MAGIC leverages local curvature information to identify the appropriate contact points and correct force direction through the curvature estimation. Moreover, on the *Hooking* task, pairwise curvature matching plays an important role in selecting the most appropriate tool for different objects, as shown in the comparison between the *no-tool-selection* and the *FG* variant in the hooking task.

Using pretrained features as guidance significantly improves contact point selection. In Table I, MAGIC shows superior performance over the sampling-based curvature matching algorithm. In principle, without noise in curvature computation and given a sufficient number of sampled points, the curvature-based method would be capable of finding the best contact. However, it will be very slow (e.g., in [4], the authors set the timeout to 600 seconds for the search). In this work, we leverage the global and semantic correspondence provided by DINOv2 features to reduce the searching space of local geometry matching. In Fig. 5(c), we visualize how pretrained DINOv2 features can improve data efficiency by identifying the best contact and grasping points that curvature alone cannot differentiate.

Moreover, we provide additional comparison of different pre-trained features such as DIFT [48] and SD-DINO [40] in Table III. The results indicate that DINOv2 not only provides better guidance for shape matching but it also runs faster. This difference can be attributed to the distinct pretraining tasks and model architectures of DINOv2. DINOv2 employs self-supervised learning techniques, involving contrastive learning and a teacher-student training paradigm, where the teacher network has access to global views and distills knowledge to the student network — which has access to both global and local views [38]. This approach enhances the model's understanding of global information. By contrast, diffusion models are trained on generation tasks through a denoising process, which focuses more on local textures [49]. Regarding model structures, DINOv2 uses Vision Transformer (ViT) [50] with an attention mechanism applied to patches of the input image, whereas diffusion models are based on U-Net [51] and process the input and output as full-resolution images, requiring more computation time.

Method	Scooping	Hanging	Hooking	Avg. Time
MAGIC (DIFT + Curvature)	62.2	81.3	**100**	60.8s
MAGIC (SD-DINO + Curvature)	73.8	91.8	96.0	50.7s
MAGIC (DINOv2 + Curvature)	**94.4**	**92.9**	**100**	**34.0s**

TABLE III: Additional studies on the choice of pretrained image features. We substitute DINOv2 with SD-DINO [40] and DIFT [50]. We present the average success rates (%) and average time consumed of the methods. The experiments are conducted on the *FG* variant.

Further comparison with dataset-dependent methods. KETO [9] and GIFT [10] introduce keypoints as the guidance for tool-using strategies and utilize self-supervised learning on the positive samples obtained from trial-and-error with the tools from a large tool dataset. While MAGIC also introduces a similar concept, contact points, we propose to consider contacts as pairs of contact points. In contrast, keypoints only focus on the tool object. Moreover, both KETO and GIFT need a dataset of tools, while MAGIC only requires one single demonstration and can be generalized to tools with very different shapes.

In Table IV, we provide additional experiment results comparing MAGIC with KETO over the *Hammering* and *Pushing* tasks from KETO. Since KETO requires a decently large dataset of tool meshes to perform self-supervised learning, which is not available for those tasks in our main experiment, we perform experiments on their tasks. The single demonstration for MAGIC is composed of an image of a reference hammer taken in the simulator and the pixel coordinates of the function point and the effect point. The results indicate that MAGIC outperforms KETO on both tasks. To further investigate the generalizability, we provide the success rate of transferring from hammers to hammers and non-hammers, respectively. For both intra-category and inter-category generalization, MAGIC performs better than KETO.

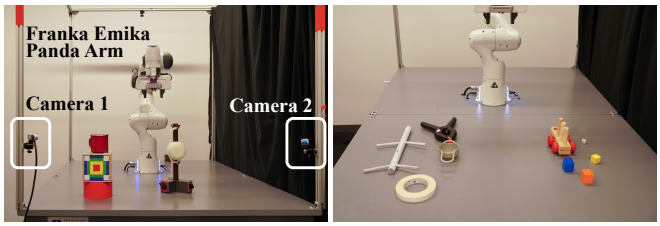
Ablation studies. Table V provides additional ablation studies on different design choices of our method MAGIC. First, there

Method	Hammering			Pushing
	Hammers → Hammers	Hammers → Non-hammers	Average	Average
KETO	66.4	46.3	56.4	71.7
MAGIC	**75.6**	**53.7**	**64.4**	**76.3**

TABLE IV: Comparison between MAGIC and KETO [9] (Learning Keypoint Representations for Tool Manipulation). We present the average success rates (%). We also investigate the intra-category and inter-category generalizablity of MAGIC and KETO, indicated by Hammers → Hammers and Hammers → Non-hammers, respectively.

Method	Scooping	Hanging	Hooking	Avg. Time
DINO Matching (Best Baseline)	55.7	23.9	66.7	**12.4s**
MAGIC (DINOv2 + Curvature)	**94.4**	92.9	**100**	34.0s
(w/o. Feature PCA and Patch)	71.7	**93.3**	62.0	33.8s
(w/o. Two-Step Curvature)	83.3	53.7	75.6	14.1s
(w/o. Simulation Verification)	77.8	71.6	50.0	19.7s

TABLE V: Ablation studies. We present the average success rates (%) and average runtime of all methods, on the *FG* variant.



(a) Hanging (b) Hooking

Fig. 6: Real-world experiment setup.

is a significant performance drop observed in the scooping and hooking tasks if we remove feature PCA and patch-based score aggregation. This demonstrates that both methods reduce noise in the feature maps and improve the accuracy of contact point matching. Second, we consider removing the irrelevant point suppression and convexity matching. The model still surpasses vanilla DINO matching, which demonstrations the effectiveness of curvature estimation, but we see a clear performance drop compared with the full algorithm MAGIC. Finally, we show that removing simulation verification significantly degrades the performance.

B. Real-World Experiments

To evaluate the effectiveness of MAGIC in real-robot systems, we conducted experiments on *Hanging* and *Hooking* utilizing a Franka Emika Panda Arm with a parallel gripper, as depicted in Fig. 6. We capture the RGB and depth image of the scene with two RealSense D435 cameras, whose extrinsics are calibrated to the frame of the base link of the robot.

We perform contact point matching on the RGB image captured by the camera, then import a reconstructed (possibly partial) object mesh from the RGBD camera into the simulator SAPIEN 2 for motion retargeting, motion planning, and verification. To reconstruct the mesh of the objects from the RGBD image, we first use Segment Anything model [52] to get the mask of the objects, and extract the corresponding point cloud of each object. Then, we remove outliers with DBSCAN [53] and select the cluster with the largest number of points. We then perform object completion by projecting the point cloud down to the table plane, and using Alpha Shape [54] to reconstruct the surfaces of the object mesh.

To provide a quantitative assessment, we executed a total of 10 trials for each object. We report the success rates of various methods for *Hanging* across three visually distinct cups and *Hooking* across four tools from different categories, as shown in Fig. 7. MAGIC has consistently achieved the highest performance, which validates the effectiveness and practical applicability of MAGIC in the real world. We also illustrate the contact point matching found by MAGIC on more real-world objects in Fig. 8.

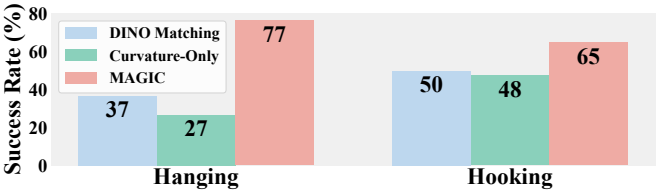


Fig. 7: Average success rates of real-world experiments.

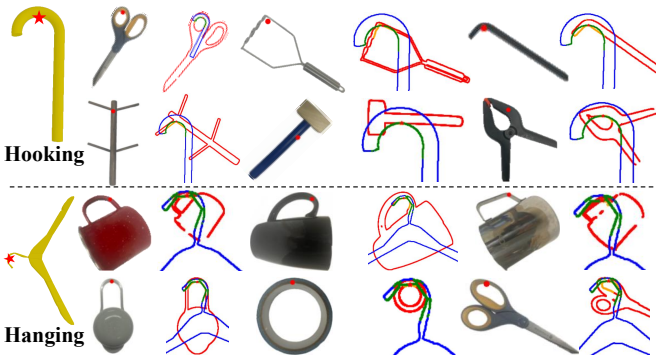


Fig. 8: Contact point matching results on real-world objects.

V. Conclusion

We have presented MAGIC, a framework for one-shot manipulation strategy learning, enabling robots to quickly adapt and execute tool-using tasks with novel objects. We integrate both data-driven and analytical shape-matching algorithms for the best of both worlds, to quickly generate precise and physically plausible contact points. Noteably, our algorithm is generic in the sense that we do not need task-specific information for the matching algorithm other than the reference contact waypoints. Experiments on three representative tasks illustrate the effectiveness of our method.

Limitations. Currently, our pipeline is designed for making contact analogies for a single contact patch between two objects. Future work should consider interactions among multiple objects with multiple contact points, and extend the current shape-based matching criteria to consider forceful affordance. Another future direction would be to consider predicting or searching over 2D views of objects for correspondence matching. Moreover, our overall framework of coarse-to-fine and semantic-to-geometric matching can be extended to 3D pretrained features [55]–[57] and curvatures.

Acknowledgements. We gratefully acknowledge support from NSF grant 2214177; from AFOSR grant FA9550-22-1-0249; from ONR MURI grant N00014-22-1-2740; and from ARO grant W911NF-23-1-0034; from MIT Quest for Intelligence; from the MIT-IBM Watson AI Lab; from ONR Science of AI; and from Simons Center for the Social Brain. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

References

[1] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact Mode Guided Sampling-Based Planning for Quasistatic Dexterous Manipulation in 2D," in *ICRA*, 2021. 1, 2

[2] ——, "Contact Mode Guided Motion Planning for Quasidynamic Dexterous Manipulation in 3D," in *ICRA*, 2022. 1, 2

[3] T. Pang, H. T. Suh, L. Yang, and R. Tedrake, "Global Planning for Contact-Rich Manipulation via Local Smoothing of Quasi-Dynamic Contact Models," *T-RO*, 2023. 1, 2

[4] J. Mao, T. Lozano-Pérez, J. B. Tenenbaum, and L. P. Kaelbling, "Learning Reusable Manipulation Strategies," in *CoRL*, 2023. 1, 2, 5, 6

[5] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann, "Neural Descriptor Fields: SE(3)-Equivariant Object Representations for Manipulation," in *ICRA*, 2022. 1, 2, 5, 9

[6] W. Liu, J. Mao, J. Hsu, T. Hermans, A. Garg, and J. Wu, "Composable Part-Based Manipulation," in *CoRL*, 2023. 1

[7] H. Huang, F. Lin, Y. Hu, S. Wang, and Y. Gao, "CoPa: General Robotic Manipulation through Spatial Constraints of Parts with Foundation Models," *arXiv:2403.08248*, 2024. 1

[8] Y. Zhu, A. Lim, P. Stone, and Y. Zhu, "Vision-Based Manipulation from Single Human Video with Open-World Object Graphs," *arXiv:2405.20321*, 2024. 1

[9] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese, "KETO: Learning Keypoint Representations for Tool Manipulation," in *ICRA*, 2020. 1, 2, 5, 6

[10] D. Turpin, L. Wang, S. Tsogkas, S. Dickinson, and A. Garg, "GIFT: Generalizable Interaction-aware Functional Tool Affordances without Labels," in *RSS*, 2021. 1, 2, 5, 6

[11] Z. Lai, S. Purushwalkam, and A. Gupta, "The Functional Correspondence Problem," in *CVPR*, 2021. 1, 2

[12] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki, "Graph-Structured Visual Imitation," in *CoRL*, 2020. 1

[13] R. Xu, F.-J. Chu, C. Tang, W. Liu, and P. A. Vela, "An Affordance Keypoint Detection Network for Robot Manipulation," *RA-L*, vol. 6, no. 2, pp. 2870–2877, 2021. 1

[14] J. C. Trinkle and J. J. Hunter, "A Framework for Planning Dexterous Manipulation," in *ICRA*, 1991. 2

[15] X. Ji and J. Xiao, "Planning Motions Compliant to Complex Contact States," *IJRR*, vol. 20, no. 6, pp. 446–465, 2001. 2

[16] M. Yashima, Y. Shiina, and H. Yamaguchi, "Randomized Manipulation Planning for a Multi-Fingered Hand by Switching Contact Modes," in *ICRA*, 2003. 2

[17] G. Lee, T. Lozano-Pérez, and L. P. Kaelbling, "Hierarchical Planning for Multi-Contact Non-Prehensile Manipulation," in *IROS*, 2015. 2

[18] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of Complex Behaviors Through Contact-Invariant Optimization," *ACM Transactions on Graphics (ToG)*, vol. 31, no. 4, pp. 1–8, 2012. 2

[19] K. Hauser and J.-C. Latombe, "Multi-Modal Motion Planning in Non-Expansive Spaces," *IJRR*, vol. 29, no. 7, pp. 897–915, 2010. 2

[20] J.-P. Sleiman, J. Carius, R. Grandia, M. Wermelinger, and M. Hutter, "Contact-Implicit Trajectory Optimization for Dynamic Object Manipulation," in *IROS*, 2019. 2

[21] B. Aceituno-Cabezas and A. Rodriguez, "A Global Quasi-Dynamic Model for Contact-Trajectory Optimization in Manipulation," in *RSS*, 2020. 2

[22] E. Huang, X. Cheng, Y. Mao, A. Gupta, and M. T. Mason, "Autogenerated Manipulation Primitives," *IJRR*, 2023. 2

[23] J. Xiao and X. Ji, "Automatic Generation of High-Level Contact State Space," *IJRR*, vol. 20, no. 7, pp. 584–606, 2001. 2

[24] A. Gupta and L. S. Davis, "Objects in Action: An Approach for Combining Action Understanding and Object Perception," in *CVPR*, 2007. 2

[25] Y. Zhu, Y. Zhao, and S.-C. Zhu, "Understanding Tools: Task-Oriented Object Modeling, Learning, and Recognition," in *CVPR*, 2015. 2

[26] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, "Learning Task-Oriented Grasping for Tool Manipulation from Simulated Self-Supervision," *IJRR*, vol. 39, no. 2-3, pp. 202–216, 2020. 2

[27] K. Mo, L. J. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani, "Where2act: From Pixels to Actions for Articulated 3D Objects," in *CVPR*, 2021. 2

[28] Y. Ju, K. Hu, G. Zhang, G. Zhang, M. Jiang, and H. Xu, "Robo-ABC: Affordance Generalization Beyond Categories via Semantic Correspondence for Robot Manipulation," in *ECCV*, 2024. 2

[29] H. Asada and M. Brady, "The Curvature Primal Sketch," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1, pp. 2–14, 1986. 2

[30] C. G. Jensen, W. E. Red, and J. Pi, "Tool Selection for Five-Axis Curvature Matched Machining," *Computer-Aided Design*, vol. 34, no. 3, pp. 251–266, 2002. 2

[31] S. Brandi, O. Kroemer, and J. Peters, "Generalizing Pouring Actions Between Objects using Warped Parameters," in *Humanoids*, 2014. 2

[32] D. Rodriguez and S. Behnke, "Transferring Category-based Functional Grasping Skills by Latent Space Non-Rigid Registration," *RA-L*, vol. 3, no. 3, pp. 2662–2669, 2018. 2

[33] O. Biza, S. Thompson, K. R. Pagidi, A. Kumar, E. van der Pol, R. Walters, T. Kipf, J.-W. van de Meent, L. L. Wong, and R. Platt, "One-shot Imitation Learning via Interaction Warping," in *CoRL*, 2023. 2

[34] S. Thompson, L. P. Kaelbling, and T. Lozano-Perez, "Shape-Based Transfer of Generic Skills," in *ICRA*, 2021. 2

[35] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "kPAM: KeyPoint Affordances for Category-Level Robotic Manipulation," in *ISRR*, 2019. 2

[36] W. Gao and R. Tedrake, "kPAM 2.0: Feedback Control for Category-Level Robotic Manipulation," *RA-L*, vol. 6, no. 2, pp. 2962–2969, 2021. 2

[37] B. Wen, W. Lian, K. Bekris, and S. Schaal, "You Only Demonstrate Once: Category-Level Manipulation from Single Visual Demonstration," in *Robotics: Science and Systems 2022*, 2022. 2

[38] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "DINOv2: Learning Robust Visual Features without Supervision," *TMLR*, 2023. 2, 4, 6

[39] K. Ehsani, S. Tulsiani, S. Gupta, A. Farhadi, and A. Gupta, "Use the Force, Luke! Learning to Predict Physical Forces by Simulating Effects," in *CVPR*, 2020. 3

[40] J. Zhang, C. Herrmann, J. Hur, L. P. Cabrera, V. Jampani, D. Sun, and M.-H. Yang, "A Tale of Two Features: Stable Diffusion Complements DINO for Zero-Shot Semantic Correspondence," in *NeurIPS*, 2023. 3, 6

[41] J. Canny, "A Computational Approach to Edge Detection," *T-PAMI*, vol. PAMI-8, no. 6, pp. 679–698, 1986. 4

[42] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, "SAPIEN: A SimulAted Part-based Interactive ENvironment," in *CVPR*, 2020. 5

[43] V.-D. Nguyen, "The Synthesis of Force-Closure Grasps," Ph.D. dissertation, Massachusetts Institute of Technology, 1985. 5

[44] S. M. LaValle and J. J. Kuffner Jr, "Randomized Kinodynamic Planning," *IJRR*, vol. 20, no. 5, pp. 378–400, 2001. 5

[45] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," in *ICRA*, 2009. 5

[46] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *TPAMI*, no. 5, pp. 698–700, 1987. 5

[47] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," *arXiv:1801.09847*, 2018. 5

[48] L. Tang, M. Jia, Q. Wang, C. P. Phoo, and B. Hariharan, "Emergent Correspondence from Image Diffusion," in *NeurIPS*, 2023. 6

[49] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *NeurIPS*, 2020. 6

[50] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *ICLR*, 2021. 6

[51] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *MICCAI*, 2015. 6

[52] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment Anything," in *ICCV*, 2023. 7

[53] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *KDD*, 1996. 7

[54] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the Shape of a Set of Points in the Plane," *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551–559, 1983. 7

[55] W. Shen, G. Yang, A. Yu, J. Wong, L. P. Kaelbling, and P. Isola, "Distilled Feature Fields Enable Few-Shot Language-Guided Manipulation," in *CoRL*, 2023. 7

[56] Y. Wang, Z. Li, M. Zhang, K. Driggs-Campbell, J. Wu, L. Fei-Fei, and Y. Li, "D^3Fields: Dynamic 3D Descriptor Fields for Zero-Shot Generalizable Robotic Manipulation," in *CoRL*, 2024. 7

[57] N. S. Dutt, S. Muralikrishnan, and N. J. Mitra, "Diffusion 3D Features (Diff3F): Decorating Untextured Shapes with Distilled Semantic Features," in *CVPR*, 2024. 7

[58] 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:1512.03012*, 2015. 9

APPENDIX

Visualization of Diverse Novel Objects. In Fig. 9, we present the demonstrations and diverse novel objects for each task.

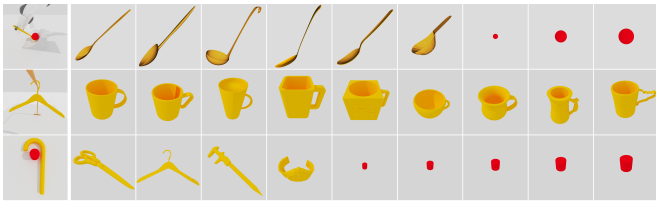


Fig. 9: Reference objects and diverse novel objects. We evaluate the ability of one-shot manipulation strategy learning by providing a single demonstration for three tasks: *Scooping*, *Hanging*, and *Hooking* (shown in the first column), and then testing on various unseen instances.

1) Scooping: Demonstrated on a reference spoon and a reference ball; evaluated on 4 spoons, a soup ladles, a measuring cup and 3 different balls.

2) Hanging: Demonstrated on hanging a hanger to a rod; evaluated on hanging mugs to a mug tree [5]. where the meshes of the mugs are adopted from ShapeNet [58]. We filter out those meshes not suitable for hanging (e.g., mugs without handles), and use 134 mugs for evaluation.

3) Hooking: Demonstrated on hooking a reference object with a hook; Evaluated on hooking 5 different cylinders with 4 tools (a pair of scissors, a hanger, a caliper and a watch).

Visualization of trajectories for simulation tasks. In Fig. 10, 11 and 12, we provide two example trajectories of each task generated by MAGIC.

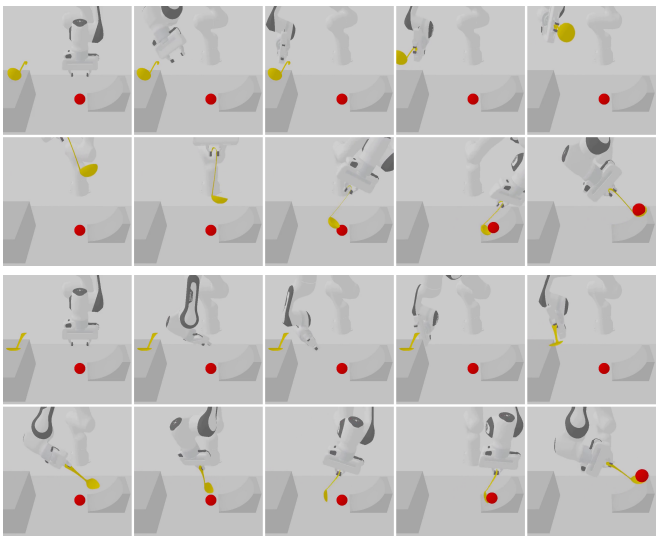


Fig. 10: Two example episodes of *Scooping*. in simulation.

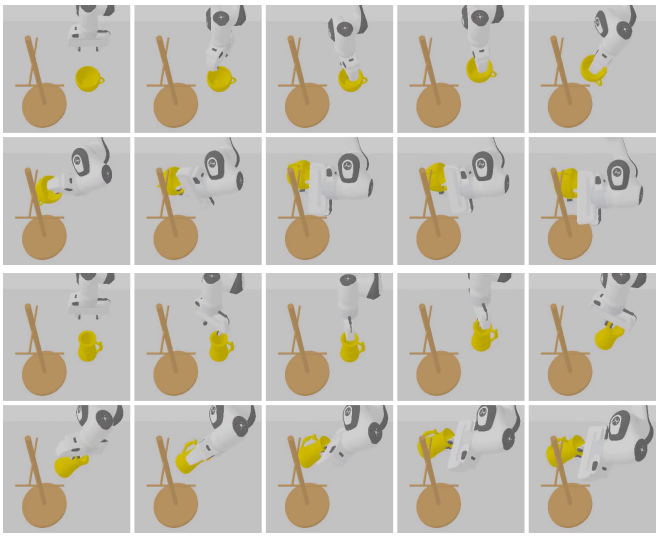


Fig. 11: Two example episodes of *Hanging* in simulation.

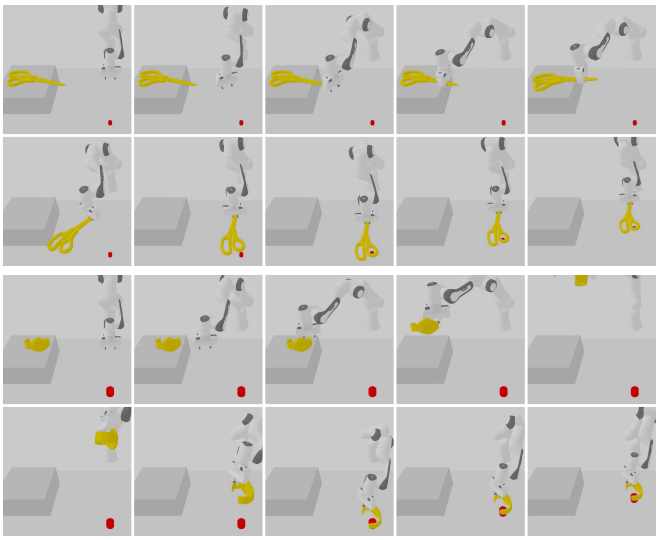


Fig. 12: Two example episodes of *Hooking* in simulation.

Visualization of trajectories for real-world tasks. We demonstrate an example trajectory of *Hanging* and *Hooking* in Fig. 13 and 14, respectively.

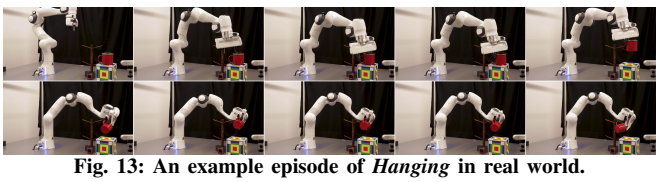


Fig. 13: An example episode of *Hanging* in real world.

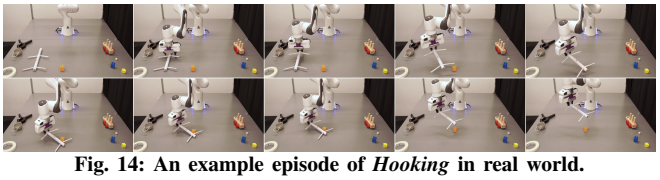


Fig. 14: An example episode of *Hooking* in real world.