Dynamics-Guided Diffusion Model for Robot Manipulator Design

Xiaomeng Xu[1], Huy Ha[1,2], Shuran Song[1,2]
[1]Stanford University, [2]Columbia University
https://dgdm-robot.github.io

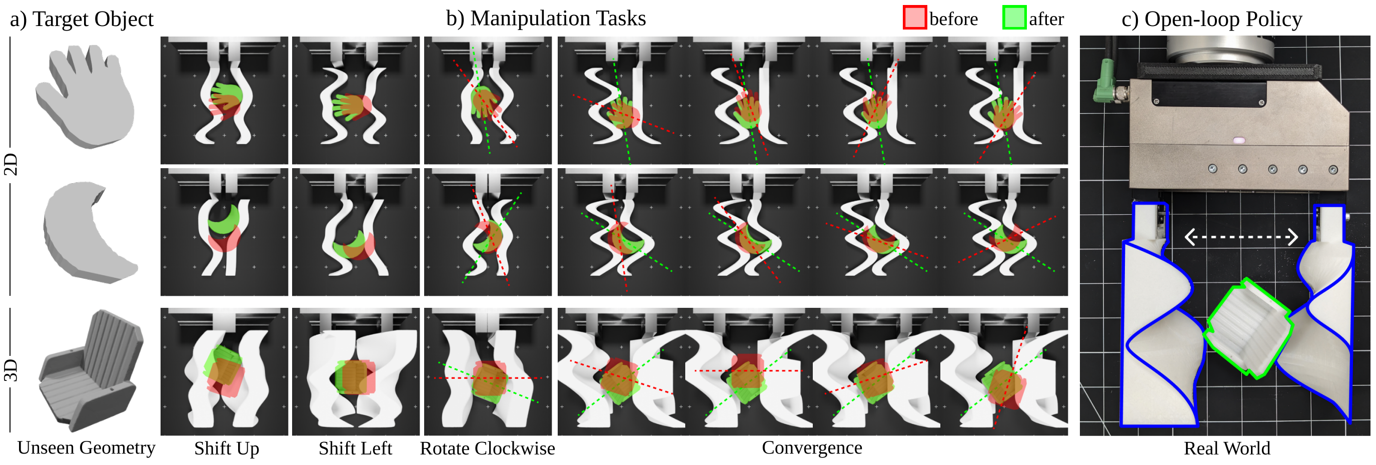

Fig. 1: **Task-specific Designs without Task-specific Training.** Given input objects (a), our algorithm learns to generate diverse manipulator geometries tailored to unseen manipulation tasks (b), which can be deployed with an open-loop motion (c) (*i.e.*, closing in parallel). Our framework generates designs for new tasks and objects in seconds, unlocking rapid design iteration cycles.

***Abstract*—We present Dynamics-Guided Diffusion Model (DGDM), a data-driven framework for generating manipulator geometry designs for a given manipulation task. Instead of training different design models for each task, our approach employs a learned dynamics network shared across tasks. For a new manipulation task, we first decompose it into a collection of individual motion targets which we call target interaction profile, where each individual motion can be modeled by the shared dynamics network. The design objective constructed from the target and predicted interaction profiles provides a gradient to guide the refinement of finger geometry for the task. This refinement process is executed as a classifier-guided diffusion process, where the design objective acts as the classifier guidance. We evaluate our framework on various manipulation tasks, under the sensor-less setting using only an open-loop parallel jaw motion. Our generated designs outperform optimization-based and unguided diffusion baselines relatively by 31.5% and 45.3% on average manipulation success rate. With the ability to generate a design within 0.8 seconds, our framework could facilitate rapid design iteration and enhance the adoption of data-driven approaches for robotic mechanism design.**

I. Introduction

Mechanical intelligence refers to the utilization of mechanical design to solve tasks and adapt to new challenges [36]. A substantial body of evidence in both natural [5] and artificial systems [39] has demonstrated that well-customized embodiments can significantly simplify an agent's perception and control, thereby enhancing overall robustness [42].

Despite its advantages, mechanical intelligence in robotics has recently been overshadowed by the rapid development of its counterpart, "action intelligence", where the agent focuses on inferring different actions for different tasks, assuming a fixed mechanical embodiment design. Namely, recent advancements in data-driven approaches [8, 33, 49, 69] have demonstrated action intelligence that can be directly applied to novel tasks and environments [14, 28]. In contrast, learning for mechanical design has largely focused on single task optimization [20, 60] or heavily engineered objective functions that could not be reused for new design task [7, 25, 34, 37, 48, 63]. In practice, this means automating task-specific design typically involves recollecting training data for every scenario, which is too expensive to be practical. Therefore, we investigate the following question:

Can we automate task-specific mechanical design without task-specific training?

We introduce **Dynamics-Guided Diffusion Model**, a framework that generates manipulator geometry designs in seconds, with no task-specific training and no perception - only a parallel jaw closing motion. From tasks as simple as object translation to complex tasks requiring sequential interactions such as pose convergence (Fig. 2), our framework's designs achieve drastically different manipulation goals with geometry changes that are highly adapted to the task and object. To enable task-agnostic pretraining for task-specific designs, our framework consists of two key technical contributions, each answering a key research question:

- **How to represent the task space?** The task repre-

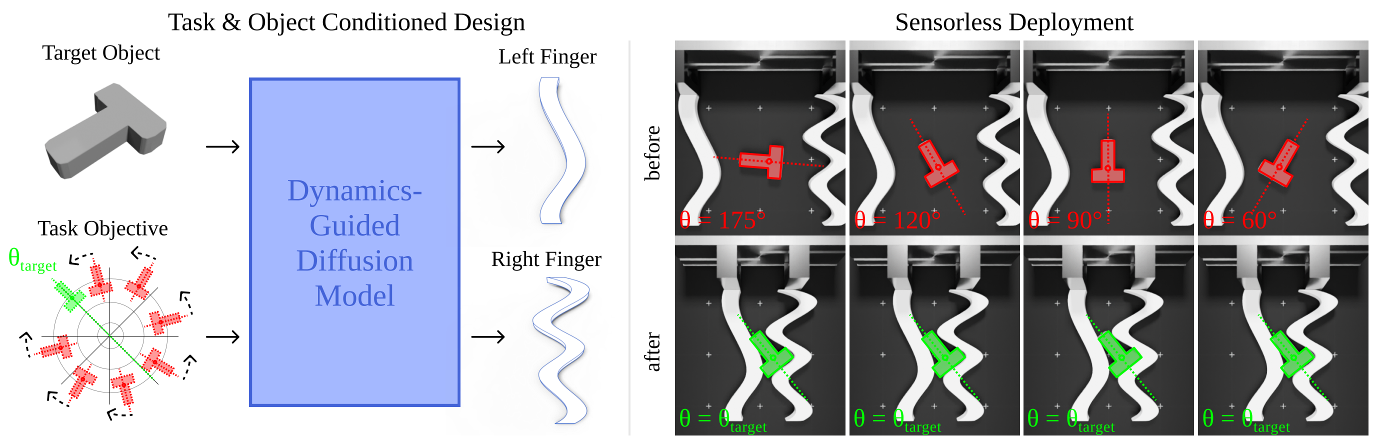

Fig. 2: **The Convergence Task.** Throughout this paper, we use convergence as an illustrative example. The goal of convergence is to design fingers that always reorient a target object to a specified orientation θ_{target} when closing the gripper in parallel. In industrial settings, this task is **useful** because it funnels the objects from arbitrary poses to a small set of θ_{target}, thereby simplifying downstream perception/planning. Despite its utility, designing for convergence can be highly **counter-intuitive**. An experienced engineer may take many design cycles to develop a reasonable design for just one object and must repeat this process for each new object. In contrast, our framework generates a functional design for a novel object within seconds, all without any prior training specific to this task.

sentation has to be expressive enough to capture the wide range of manipulation tasks while being compact enough to be readily learned from data. Our key insight is that many manipulation tasks can be *decomposed* into a collection of individual motion targets that specify how *each* object should move under *each* initial pose. We call the collection of individual motions **interaction profile**. While the final composed objective is specific to the task, each of the individual motions can be modeled by a generic dynamics model that is reusable across tasks.

- **How to facilitate efficient and robust search?** As the design space grows, the distribution of good designs often becomes multi-modal, and the task of generating promising yet diverse design candidates becomes challenging. To address this issue, we propose a **dynamics-guided diffusion model**. First, with the dynamics network, we can infer the current interaction profile for an object and the current fingers. The design objective constructed by comparing the current with the target (*i.e.*, task-specific) interaction profiles gives us a gradient on how to update the finger. This guidance is incorporated into diffusion denoising steps similarly to classifier guidance [11]. Tuning the guidance scale allows users to trade off the exploration of diverse candidates and the exploitation of the most promising modes.

We demonstrate results on both 2D and 3D objects with a variety of manipulation objectives ranging from primitive to complex and single- to multi-object objectives, all under a sensor-less setting, where the initial pose of the object is unknown. Experiments in simulation and the real world demonstrate that designs generated by our approach achieve high task performance, with 31.5% and 45.3% relative success rate improvements compared to optimization and unguided diffusion generation results.

II. Related Work

A. Manual End-effector Design

Domain-specific end-effector designs have a long history in robotics, which expands the system's capabilities while simplifying perception and control. The diverse array of manipulator designs we see today (serial [2, 47], parallel [18], multi-finger [19, 59], platform-based [15, 38], dexterous [29, 35], underactuated [6, 9, 10, 24, 36]) typically start as the result of many trail-and-error iterations by expert engineers. Though many analytical design methods have been developed [3, 12, 30, 31, 44, 55, 65], heavy manual efforts are still needed to discover optimal and occasionally counter-intuitive designs in practice, which significantly hinders the development of designs for new applications. For instance, for complex manipulation tasks such as convergence (Fig. 2), previous works only deal with 2D planar polygons, utilizing manual or analytical designs of grasping policy [1, 17] or gripper geometry [67].

B. Analytical Optimization for Automatic End-effector Design

To alleviate the manual efforts, previous works have explored optimization approaches to manipulator design. Though non-linear optimization is possible [57], this approach typically requires careful task-specific formulation of objectives and constraints. Therefore, first-order optimization of morphology only [62] or both morphology and control [32, 57, 63] is more common, but requires careful initialization (task-specific parameterization [62], cage-based deformation [34, 63], or heuristics [32]). Further, tasks involving complex contact modes, such as the ones we consider, are known to yield biased and high variance gradients in differentiable simulators [34, 56]. Importantly, all manual efforts involved in setting up an optimization problem are typically not transferrable to new tasks or objects, and thus require a significant amount of manual labor for each new task.

C. *Data-driven Robot Hardware Design*

Fundamentally, data-driven approaches offer an improvement over optimization-based approaches by facilitating the transfer of knowledge from one scenario to another, thus reducing costs at inference and optimization stages. A common approach to leveraging data is to distill the evaluation process into a value network, which takes the embodiment parameterization as input and outputs the design's task-specific performance. This value network can then be used to guide a search/optimization procedure, which is often more efficient than evaluating the design in simulation. This direction has been explored in the realm of gripper design [20] and locomotion [64, 68], to guide optimization [20, 26, 60] or graph search [25, 61, 64, 68]. Another common approach is to learn a generative model of the design space, which effectively compresses the design space into a low-dimensional continuous latent space. This makes offline optimization via gradient-descent [20, 26] or online optimization via trial-and-error rollouts of random latent-space samples [20, 25, 60] significantly more efficient. Finally, when co-optimizing morphology and control, leverage control experience from prior embodiment evaluations can significantly improve the efficiency and accuracy of new embodiment evaluations [7, 48].

All these data-driven approaches require task-specific data, yet only work well when provided with enough data for robust generalization. We posit that this requirement on a large amount of task-specific data has hindered the mass adoption of data-driven approaches to robot hardware design. In this work, we aim at eliminating the requirement on task-specific data when designing manipulators for a new task, by leveraging dynamics as the shared structure between manipulation tasks.

III. Approach

A. *Interaction Profiles as Task Specification*

Compact, Expressive Task Representation. Requirements for a manipulation system are incredibly diverse, ranging in what initial poses are allowed, what objects are considered, and what the desired effects are. Parameterizing the space of manipulation task objectives call for, first and foremost, a representation **expressive** enough to capture tasks ranging from shifting objects upwards to temporally-extended sequential tasks such as pose convergence (Fig. 2). Moreover, this representation should be **compact** - containing only the necessary information to capture how the object interacts with the finger, such that it is efficient to evaluate/learn. For instance, modeling the detailed physics states such as in differentiable simulators [27, 34, 63] is expressive, but forward integrating the dynamics over the time horizon for every finger evaluation is expensive.

Interaction Profiles. Our key insight is that many manipulation tasks can be decomposed into a collection of individual motion targets that specify how each object should move under each initial pose. By combining motions from all objects and initial poses under consideration, we get a complete profile of how the current manipulator will interact with the target objects - the "interaction profile". Below, we introduce the notation and demonstrate how interaction profiles can be used to specify manipulation tasks.

Denote by o and m the object shape and manipulator shape. When o is at the initial planar pose $p = (\theta, x, y)$, closing m once will change o's pose by $\Delta p = (\Delta\theta, \Delta x, \Delta y)$, dictated by the manipulator-object interaction dynamics $\mathcal{D}$. We refer to scalar-valued functions f defined on top of Δp as motion objectives and aggregate these motion objectives among all initial poses p and objects o to get the design objective F.

Example: Multi-object Shift Up

Suppose we wish to design a manipulator that shifts a set of objects upwards. We can define each motion objective as

$$f(o,m,p) = \Delta y(o,m,p) \tag{1}$$

where Δy is shorthand for the y-translation component of Δp. The design objective for this task can then be aggregated from (1) as

$$F(m) = \sum_o \sum_p f(o,m,p) \tag{2}$$

By its design, interaction profiles naturally scale to varying ranges of initial poses and objects. Thus, from the previous example, using a larger set of initial poses and objects will yield an objective that is more robust to different initial poses and tailored to more objects. Since each motion objective is conditioned on p, this approach also allows for objectives dependent on initial poses, as we'll illustrate in the example below.

Example: Pose Convergence

The goal of pose convergence is to rotate an object to a target orientation θ_{target} (Fig. 2). A well-designed convergence manipulator can funnel a wide range of initial configurations into a single orientation - the target orientation θ_{target} - with no perception, no closed-loop control, only a parallel gripper closing motion on repeat. How the object should rotate depends on the initial orientation θ relative to the target orientation θ_{target}:

$$f(o,m,p) = \begin{cases} \Delta\theta(o,m,p) & \text{if } \theta \in [\theta_{\text{target}} - \pi, \theta_{\text{target}}] \\ -\Delta\theta(o,m,p) & \text{if } \theta \in [\theta_{\text{target}}, \theta_{\text{target}} + \pi] \end{cases} \tag{3}$$

The objective for this task can then be aggregated from (3) analogously to (2).

If $\nabla_m F$ can be efficiently computed, then we can use this gradient information to find a pair of fingers m that achieves the task. To achieve this, we propose to represent $\mathcal{D}$ as a neural network and train it using data generated from interactions between random finger-object pairs. Subsequent sections will delve into the learning of dynamics (§ III-B) and the use of diffusion processes for guiding manipulator design (§ III-C).

B. Dynamics Network

The dynamics network $\mathcal{D} : (o,m,p) \mapsto \Delta p$ aims to learn a general model of how a random distribution of fingers interacts with a distribution of objects. Importantly, it provides gradients of the design objective with respect to the finger representation. The following paragraphs describe the details of the dynamics network's training procedure.

Shape Representation. We choose cubic Bézier curves and surfaces as the shape representation for the manipulator geometry m, for their flexibility and ubiquity in parameterizing shapes [4, 16]. Control points are grid sampled along the length (and height in 3D) of the finger while the remaining y-coordinate of the control points determines its protrusion outwards/inwards. We represent object shape o as a point cloud by surface sampling the object mesh. For 2D objects, we sample 100 2D points from their contour surfaces. For 3D objects, we sample 512 3D points from all surfaces.

Motion Representation. We represent object motion under interaction as a three-dimensional vector consisting of delta rotation along the z-axis, delta translation along the x-axis, and delta translation along the y-axis, denoted as $\Delta p = (\Delta\theta, \Delta x, \Delta y)$.

Network Architecture. In the 2D manipulation case, we use an MLP as the network architecture. First, we transform sampled object initial poses p with a high-frequency positional encoding - a trick typically used to combat over smoothing of neural networks [40]. Then, o and m are passed through separate 2-layer multi-layer perceptrons (MLPs) with 256 hidden dimensions, before being concatenated together with the high-frequency pose embedding. Finally, the resulting embedding is passed through an 8-layer MLP with 256 hidden dimensions to get the predicted object motion Δp. In 3D, we use a PointNet++ [43] encoder to encode object geometry, whereas all other parts of the architecture are shared between 2D and 3D tasks.

Training Data Generation. Since our dynamics network training only requires task-agnostic data, its data generation happens once for all tasks. We sample random object and manipulator pairs, load them into MuJoCo [58] simulation environment, and measure Δp after a single parallel jaw closing interaction. For the 2D case, we generate 321 planar object shapes by first extracting contours of 2D icon images from the Icons-50 dataset [21], and then extruding it to a planar shape. In the 3D setting, we select 164 objects from Google's Scanned Objects Dataset [13, 66], filtering out objects whose scales are out of bound or tilt over during interaction. We randomly sample 1024 manipulator geometry parameters m from a uniform distribution. For each object-fingers pair, we grid sample 360 initial orientations and 5×5 initial positions, then run forward simulations to get delta object poses after a single gripper close. In total, we use $321\times1024\times360\times25$ data points for training the 2D dynamics network and $164\times1024\times360\times25$ data points for training the 3D dynamics network.

Design Objective Gradient Evaluation. To design manipulators that are generalizable to more initial poses p and objects o, the design objective F (2) can be evaluated for a wider range of poses and objects. Concretely, we can grid-sample p's and a set of o's and evaluate each motion objective f in parallel along the batch dimension. The design objective gradient $\nabla_m F$ is attained by aggregating the gradients along the pose/object batch dimension. For each new design, evaluating $\nabla_m F$ takes 0.16 seconds on average, which makes it efficient to run in the inner loop of iterative design procedures. In this project, we grid-sample 360 orientations and 5×5 positions, getting a 360×5×5 dimensional motion profile.

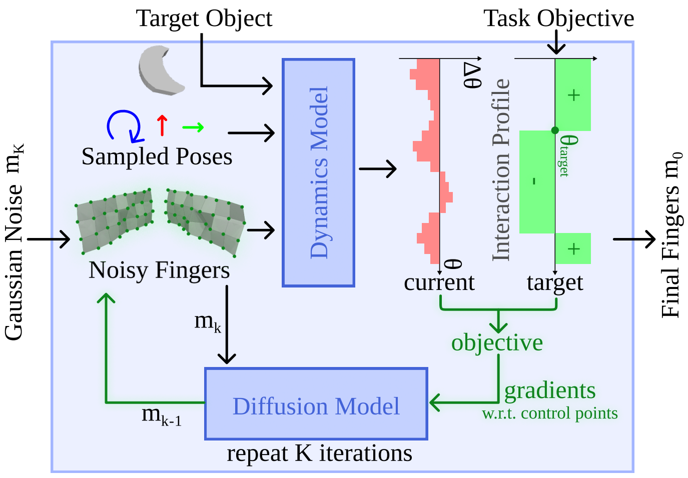

Fig. 3: **Dynamics-Guided Diffusion Model.** Our approach generates finger shapes given a target object and task, specified as a target interaction profile (§ III-A) describing how the object should move under different initial configurations. This target can be compared with the dynamics network's prediction of the current interaction profile, which is used to construct a differentiable objective (§ III-B). Gradients of the objective are then used to iteratively guide the reverse denoising process of a manipulator shape diffusion model (§ III-C). By averaging gradients across more sampled poses and target objects, our framework can generate fingers that are robust to a wider range of initial poses and tailored to more objects.

Modelling Interactions instead of Modelling Contacts. Differentiable simulators [27, 34, 54, 60, 63] are another popular choice for providing $\nabla_m F$, but suffer from two major limitations. First, soft contact models, such as penalty-based methods used by Xu et al., are known to yield biased and high-variance gradients [34, 56]. Further, such gradients need to be computed for each simulation timestep, which is computationally expensive for long-horizon interactions. Instead of modeling individual contacts as in differentiable simulators, our dynamics network learns to capture the temporally extended finger-object interaction. This data-driven approach can be trained on physically accurate simulated data, avoiding the limitations associated with soft contacts. Moreover, our dynamics network exhibits flexibility by seamlessly generalizing to novel objects and tasks at test time, allowing for the construction of objectives without extra data generation/training.

C. Dynamics-Guided Diffusion Model

Given design objective gradients from $\mathcal{D}$ (§ III-B), the obvious approach is to perform gradient descent on the finger geometry [20, 32, 34, 57, 63]. However, the distribution of good designs are often multi-modal, which means gradient descent approaches quickly get stuck in local minima. Instead, to efficiently navigate through the large and multi-modal design space, we extend classifier guidance [11], an iterative diffusion model sampling approach that enables a balance between multi-modal diversity and task-specific guidance.

Diffusion Model. Diffusion models [23, 50] are a class of probabilistic generative models that generate samples from an underlying distribution through iterative denosing. A diffusion model $\varepsilon_\theta(m_k)$ predicts the noise added to a sample [23], which is trained by first adding noise ε to m_0 to produce noised sample m_k, and then apply a mean-squared error loss between the ground truth noise and predicted noise $||\varepsilon_\theta(m_k)-\varepsilon||^2$. To generate samples from a diffusion model, we start with a Gaussian noise m_K and gradually predict less-noisy samples $m_{K-1}, m_{K-2}, \ldots$, until final sample m_0. The process involves repeatedly predicting m_{k-1} from m_k, which is the reverse noising process of modeling the distribution $p_\theta(m_{k-1}|m_k)$. Specifically, we employ Denoising Diffusion Implicit Models (DDIMs) [51] as the sampling method.

We sample manipulator shape parameters m from a uniform distribution and train a diffusion model on this distribution. The diffusion model $\varepsilon_\theta(m_k)$ with 1D UNet architecture [46] predicts noise added to a sample m_k. We use DDIM for diffusion sampling process $p_\theta(m_{k-1}|m_k)$ with 15 training denoising iterations and 5 inference iterations, and the Square Cosine noise scheduler [41].

Background on Classifier Guidance. Many techniques are developed to condition diffusion model on certain inputs or guide the reverse diffusion process with priors, such as concatenation, cross-attention [45], classifier-free guidance [22], *etc*. Among these approaches, classifier guidance [11] can guide the reverse noising process towards desired samples with an unconditional diffusion model. It requires a classifier $p_\phi(l|m_k)$, where m_k is the sample, l is the class label, and ϕ is the classification network. To condition the sampling process on label l, each reverse noising transition is:

$$p_{\theta,\phi}(m_k|m_{k+1},l) = Zp_\theta(m_k|m_{k+1})p_\phi(l|m_k) \tag{4}$$

where Z is a normalizing constant. Classifier guidance can be applied to DDIM sampling method. Leveraging the connection between diffusion models and score matching proposed by [52, 53], a score function for $p(m_k)$ can be derived from the diffusion model:

$$\nabla_{m_k}\log p_\theta(m_k) = -\frac{1}{\sqrt{1-\bar{\alpha}_k}}\varepsilon_\theta(m_k) \tag{5}$$

where $\alpha_k := 1-\beta_k, \beta_k$ is the variance of Gaussian noise added to samples at denoising step k, and $\bar{\alpha}_k := \prod_{s=1}^{k}\alpha_s$. The score function for $p_\theta(m_k)p_\phi(l|m_k)$ is:

$$\begin{aligned}\nabla_{m_k}\log(p_\theta(m_k)p_\phi(l|m_k)) &= \nabla_{m_k}\log p_\theta(m_k) + \nabla_{m_k}\log p_\phi(l|m_k)\\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_k}}\varepsilon_\theta(m_k) + \nabla_{m_k}\log p_\phi(l|m_k)\end{aligned} \tag{6}$$

A new noise prediction can be defined as:

$$\hat{\varepsilon}(m_k) := \varepsilon_\theta(m_k) - \sqrt{1-\bar{\alpha}_k}\nabla_{m_k}\log p_\phi(l|m_k) \tag{7}$$

Then DDIM can be performed with the modified noise prediction for conditioned sampling.

Algorithm 1 Dynamics guided DDIM sampling, given a diffusion model $\varepsilon_\theta(m_k)$, design objective $F(m_k)$ constructed from dynamics network, and gradient scale s.

```
Input: design objective $F(\cdot)$, gradient scale $s$
$m_K \leftarrow$ sample from $\mathcal{N}(0,\mathbf{I})$
for all $k$ from $K$ to 1 do
    $\hat{\varepsilon} \leftarrow \varepsilon_\theta(m_k) - s\sqrt{1-\bar{\alpha}_k}\nabla F(m_k)$
    $m_{k-1} \leftarrow \sqrt{\bar{\alpha}_{k-1}}\left(\frac{m_k-\sqrt{1-\bar{\alpha}_k}\hat{\varepsilon}}{\sqrt{\bar{\alpha}_k}}\right) + \sqrt{1-\bar{\alpha}_{k-1}}\hat{\varepsilon}$
end for
return $m_0$
```

Dynamics Guidance & the Effects of Guidance Scaling. To guide the design generation towards manipulation design objectives, we extend classifier guidance to use interaction profiles instead, which we term dynamics guidance. Concretely, we replace classifier gradients with $\nabla_{m_k}F(m_k)$ to guide the DDIM sampling process (Algo. 1). By extending classifier guidance, we not only enable guiding unconditional diffusion models with task-specific gradients, but also inherit an elegant way to trade off diversity and performance. Since $s\nabla_m \log p(l|m) = \nabla_m \log \frac{1}{Z}p(l|m)^s$, where s is the gradient scale and Z is an constant, increasing s has the effect of sharpening the sampling distribution towards $p(l|m)^s$. Dhariwal and Nichol showed that when s is larger the distribution becomes sharper and generated samples have higher fidelity, while smaller scale leads to more diverse samples. We observe the similar effect of different gradient scales [11] from our generated results, as shown in Fig. 5.

IV. Evaluation

We design experiments and a suite of manipulation tasks, aimed at answering the following questions:

1) Does our approach design effective manipulators customized for a range of unseen tasks and 2D/3D objects?
2) Does dynamics-guided diffusion generation provide better performance than gradient descent optimization? What are the design strategies the algorithm learns?
3) How does different conditioning (*e.g.*, multiple objects) impact the final design?
4) How do designs from our approach transfer to real hardware?

Manipulation Tasks & Metrics. We categorize our suite into two difficulty levels.

		Primitive						Complex			
		up	down	left	right	clock	counter	rotate	clockup	clockleft	converge
2D	Unguided	56.8	82.1	82.9	80.4	46.9	58.5	74.0	36.4	36.9	61.7°
	Opt.	79.5	53.3	81.3	94.0	48.8	**73.2**	78.9	29.3	49.4	73.7°
	DGDM	**88.2**	**92.0**	**96.7**	**97.7**	**60.8**	72.0	**79.3**	**62.8**	**63.7**	**83°**
3D	Unguided	43.0	43.8	80.4	87.9	41.2	33.5	64.2	30.3	33.1	63.6°
	Opt.	47.4	66.3	86.3	88.1	59.1	52.0	66.9	29.2	37.7	60°
	DGDM	**81.5**	**75.1**	**95.1**	**97.2**	**69.9**	**65.0**	**83.0**	**57.1**	**58.2**	**72.5°**

TABLE I: **Single Object Evaluation**. We evaluate on primitive task objectives including shift up, shift down, shift left, shift left, rotate clockwise, rotate counterclockwise, and complex objectives including rotate either way, rotate clockwise and shift up, rotate clockwise and shift down, and convergence. We report average success rates (%) on all tasks, with the exception of convergence, which is reported in degrees of the largest convergence range.

1) **Primitive objectives** involve single-axis object movements or rotations in $SE2$ space, such as shifting up or rotating counterclockwise. We consider all 6 primitive objectives and report **average success rates**, defining success based on predefined thresholds (see supplementary material for details).
2) **Complex objectives** combine multiple primitive objectives to parameterize a broad range of manipulation tasks. For example, the convergence objective includes rotation objectives with target directions and magnitudes based on initial object orientation. Additionally, we explore clock-up, clock-left, and rotate objectives to showcase our approach's flexibility in composing orthogonal or conflicting objectives. For convergence, we report the **maximum convergence range** in degrees, indicating the broadest range of initial orientations leading to a consistent final orientation within a small tolerance (5 degrees). Meanwhile, other complex tasks are evaluated on success rates.

Ablations. First, removing our task-specific guidance from our dynamics model yields the **Unguided** baseline, which generates task-agnostic manipulators using our geometry diffusion model. Second, removing our diffusion model yields the **Opt.** baseline, which optimizes the manipulator control points using gradients of the task objective through our dynamics network, common for many prior works [20, 32, 34, 63]. To mitigate performance variance due to initialization, we run each approach 16 times per object-task pair and select the best performance, then average among objects.

Evaluation Procedure. We evaluate each approach on held-out objects (8 in 2D, 6 in 3D) and manipulation tasks. Each finger Bézier representation is extruded into a 3D mesh and mounted to a WSG50 gripper performing a fixed open-close action. To investigate orientation robustness, we grid-sample 360 planar initial orientations and calculate average success rates after the first open-close action for all tasks except convergence. Observing continued object movement, we report convergence metrics after the 40th open-close action.

		Primitive						Complex		
		up	down	left	right	clock	counter	rotate	clockup	clockleft
2D	Unguided	55.8	79.8	77.1	80.2	44.7	56.4	68.3	35.2	35.2
	Opt.	78.6	50.3	79.2	93.8	46.1	**71.4**	74.3	25.0	49.0
	DGDM	**83.8**	**88.1**	**99.3**	**94.3**	**61.3**	68.4	**78.4**	**62.4**	**63.8**
3D	Unguided	40.1	40.8	75.8	87.9	34.7	29.2	61.7	29.2	25.2
	Opt.	42.4	66.4	77.9	86.6	40.3	39.2	67.0	22.6	34.3
	DGDM	**89.7**	**66.8**	**96.1**	**95.4**	**69.3**	**58.2**	**77.6**	**44.2**	**37.9**

TABLE II: **Multi-object Evaluation** (Average success rates %).

A. *Experiment Results*

*1) **Task-specific manipulators**:* Despite having no task-specific training, our dynamics and diffusion model can generate tailored fingers for a wide variety of scenarios, surpassing the unguided baseline (*i.e.*, uniform control points) *across all primitive and complex tasks*. The advantages of generating custom fingers over random counterparts become more pronounced as the design requirements escalate. For instance, our approach exhibits a +16.6% improvement over the unguided baseline in 2D primitive objectives, a figure that expands to 20.0% in 2D complex objectives (see Table I). A similar trend is observed when transitioning from 2D to 3D objects (+18.0% over Unguided in 2D, +23.4% over Unguided in 3D, Table I) and from single-object to multi-object designs (+18.0% over Unguided in single-obj 2D, +18.6% over Unguided in multi-obj 2D, Table II).

In each of these scenarios, where task complexity, design space, and the target object set grow, a human expert designer would face significantly increased time and effort. However, our approach, leveraging the gradient aggregation from individual motion objectives (per initial configuration, task, and object), offers a simple yet effective solution to handle progressively complex design requirements. As long as users can articulate how *each* object should move from *each* initial pose, this specification can be seamlessly incorporated into the diffusion denoising process.

*2) **Robustness & efficient search with guided diffusion**:* The Opt. baseline and DGDM share the same task objective gradients from our learned dynamics network, differing only in how this gradient information is incorporated. The baseline uses gradient descent, a method that requires upwards of 18 and 24 minutes to converge in 2D and 3D (for 16 samples), respectively, and is prone to local minima. In contrast, our approach utilizes classifier-guidance with a diffusion denoising process, which strikes a balance between exploring different modes (by introducing Gaussian noise) and exploiting the current mode (using the gradient of design objective) through a guidance scaling factor (Fig. 5). This results in +12.8% and +10.4% higher success rates than the baseline in 2D and 3D primitive objectives, respectively. Additionally, our diffusion models prove stable even with diffusion processes as short as 5 timesteps, translating to an average design time of 13 and 54 seconds in 2D and 3D (for 16 samples), respectively.

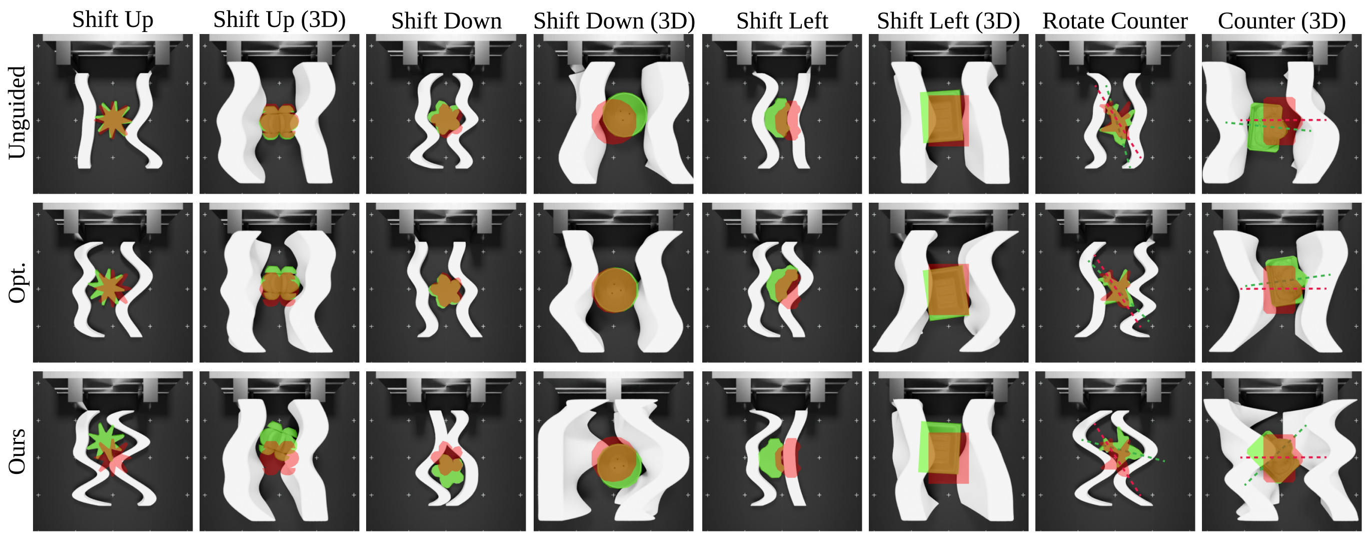

Fig. 4: **Results on Primitive Objectives.** We generate manipulators for primitive objectives that involve motion along one dimension in $SE2$ space. Red and green object masks denote object configurations before and after interaction respectively, overlaid with the image after closing in the manipulator. Compared with baseline methods, finger shapes produced by our method achieve the task much more effectively.

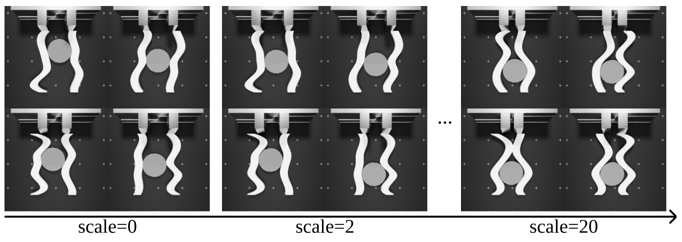

Fig. 5: **Effect of Scaled Guidance.** From left to right we increase scaled guidance in the diffusion process. The increased scale will enforce more task guidance and achieve higher task performance (shifting down), however, reduce the diversity of generated designs. By tuning this parameter, the user could trade-off between diversity and task guidance of the candidate designs at test time.

*3) **Emergent design for convergence**:* What strategies do our designs employ to consistently achieve convergence from a broader range of initial orientations compared to the unguided baseline (+21.3° in 2D, +8.9° in 3D, Table I)? From Fig. 2, we identify two emergent design patterns not commonly observed in the baseline algorithm:

1) **Push-and-Catch**: In the majority of our convergence designs, one finger features a bulge that pushes the object into the hollow cavity of the other finger (Fig. 6 a,b,d-f,h). Interestingly, this cavity should *roughly* complement the object's size and shape at the convergence point, but not precisely. We hypothesize that this extra wiggle room (most prominent in Fig. 6 e,f) supports a broader range of initial orientations by allowing extreme orientations to "slide" into the optimal orientation, caught by the cavity.
2) **Parallel-Align**: When objects exhibit symmetric flat edges, our designs utilize two parallel surfaces on each finger to align these edges (Fig. 6 c,g). Here, the parallel surfaces not only align with the object's flat edges but also with each other, reinforcing the object from both sides towards the target convergence orientation.

In both of these strategies, the pair of fingers must simultaneously exploit object **geometry** (*i.e.*, size, shape, symmetry) and **physics** (sliding), coordinating their geometry to achieve the most effective convergence. This emergent coordination between the pair of finger designs arises from our dynamics guidance and is not commonly observed in the baseline.

*4) **Specialized or generalized designs for multi-object scenarios**:* It is commonplace for automation pipelines to handle various object types with differing geometries. In such cases, opting for a more generic design that performs well across all objects may be advantageous, as opposed to fabricating one optimal design for each object. However, customizing manipulators for multiple objects quickly becomes unwieldy, as a gripper designed to achieve the task with one object might cause undesirable effects on another object. For example, a manipulator pair designed to rotate a T-shape clockwise could inadvertently move a house- and car-shaped object up without rotation (Fig. II, top row, 'rotate clockwise').

Approaching this challenge from a compositional perspective intuitively appears promising, requiring two key skills. First, the designer must enumerate a *diverse set of design candidates* that achieve the task for each object, including designs that may not be optimal. Second, the designer must sift through these candidates to prioritize common design patterns effective across all objects. Ultimately, these common elements can be composed into a unified design suitable for all considered objects. In other words, design approaches unable to navigate a large, multi-modal design space or filter through a vast set of design candidates may struggle to achieve this multi-object design scenario.

Our algorithm demonstrates the ability to discover this compositional design intuition by effectively balancing diversity (§ IV-A2) with object- and task-specific performance guidance (§ IV-A1). From Fig 7, we observe that the multi-object fingers for "shift down" (first column) combine the

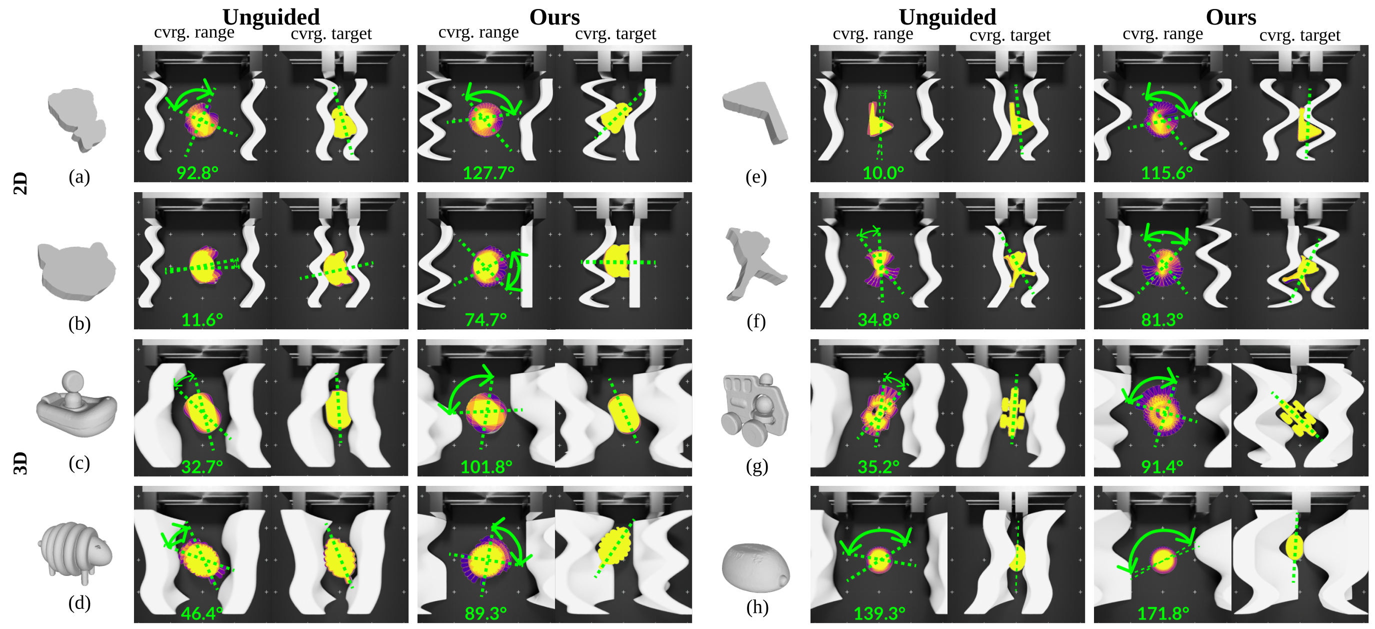

Fig. 6: **Convergence Results.** For each pair of finger designs, we show the range of initial orientations ("cvrg. range") which converges to the same convergence mode ("cvrg. target"). In the generated designs (under "ours"), we observe two emergent design patterns: a bulge on one finger pushes the object into the other finger with a hollow cavity (a,b,d-f,h), and two parallel surfaces on either finger "catch" and align objects with symmetric flat edges (c,g). In both strategies, the pair of fingers must simultaneously exploit object **geometry** and **physics** (sliding) and coordinate their own geometry to achieve the most effective convergence.

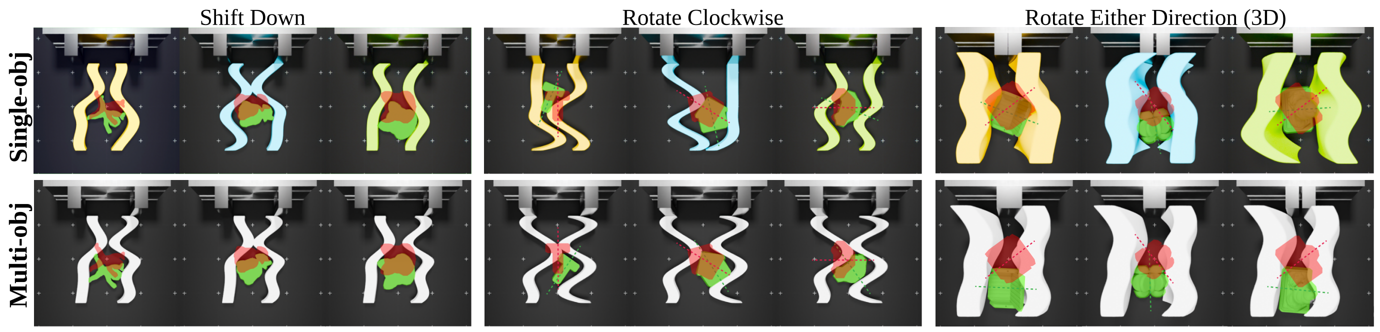

Fig. 7: **Specialized or Generalized Design in Multi-object Scenarios.** Our approach can be flexibly conditioned on individual objects and generate a specialized design for each object [Top] or simultaneously conditioned on multiple objects at once and generate one design for all objects [Bottom]. This flexibility allows the user to trade-off between performance (single-object) and generality (multi-object) based on the task and system requirements.

left finger from one design and the right finger from another design. Meanwhile, the left multi-object finger for "rotate clockwise" is derived from the left single-object finger for the house. In contrast, the unguided baseline lacks task-specific guidance, hindering its ability to guide its diverse generations toward a common design effective for all objects. On the other extreme, the Opt. baseline often gets stuck in a local minimum, unable to effectively explore a multi-modal design space. These limitations are reflected in Table II, with our approach achieving $+18.6\%$ and $+23.4\%$ higher success rates than the unguided baseline in 2D and 3D, respectively, and $+14.7\%$ and $+17.6\%$ higher success rates than the Opt. baseline in 2D and 3D, respectively.

Naturally, we acknowledge that multi-object finger designs must sacrifice some performance compared to the single-object scenario (-1.5% in 2D, -5.2% in 3D) to achieve more object-level generality. This balance between generality and task-specific performance is a fundamental trade-off in mechanism design in automation, with the optimal compromise dependent on the specific application. These results illustrate that designs from our approach can vary along this spectrum without additional training, providing a valuable tool for designers to explore the trade-off in their design space.

5) ***Real-world evaluation with Sim2Real transfer:*** We show real-world results of all tasks for both 2D and 3D cases by mounting the 3D printed designs on a WSG50 (Fig. 8). Surprisingly, our Sim2Real transfer was significantly simplified due to our sensor-less formulation - with no perception and closed-loop control, only geometry and physics impacted the transferability. Overall, our real-world results highlight the

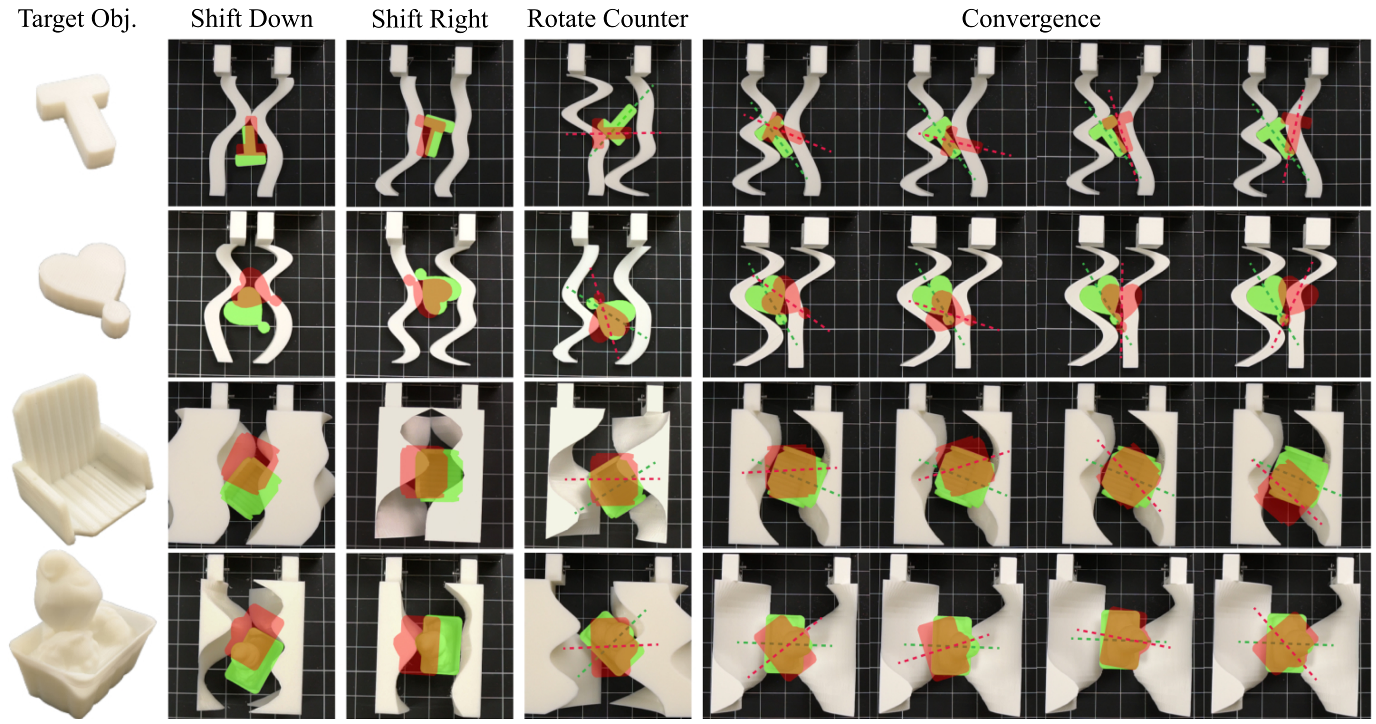

Fig. 8: **Real-world Results.** We manufacture manipulators generated by DGDM and execute the open-loop parallel closing motion. Behaviors in simulation successfully transfer to the real world. Red and green masks denote object configurations before and after interaction respectively.

robustness of our designs to unmodeled plastic deformations in simulation and manufacturing imperfections. For instance, our convergence fingers typically achieve convergence much faster in the real world because the lower friction coefficient of 3D printing PLA material allows the object to slide into the cavity more easily (§ IV-A3). These results are best viewed in the video attached in the supplementary material.

V. Conclusion and Future Directions

We present Dynamics-Guided Diffusion Model, a versatile framework for the rapid generation of diverse and tailored manipulator geometry designs for unseen tasks. This task-agnostic framework lays the groundwork to enable more rapid experimentation and future research. To expand the scope of our work beyond geometry design, investigating co-optimizing articulation structure, materials, and policy would unlock broader applications. To support a better interface for users, it is an interesting future direction to automatically detect when different objectives conflict with each other (*e.g.*, in multi-object scenarios) using their gradients, and informing the user on the feasibility of the design requirements. We hope that our framework contributes to the wider adoption of data-driven approaches in the domain of robotic mechanism design.

Acknowledgements

We thank Zhenjia Xu, Cheng Chi, Mandi Zhao, Zeyi Liu, Austin Patel, Chuer Pan, Yihuai Gao, Dominik Bauer, Samir Gadre, Mengda Xu and John So for their thoughtful discussions and helpful feedback on initial drafts of the manuscript. This work was supported in part by the NSF Award #2143601, #2037101, and #2132519. We would like to thank Google for the UR5 robot hardware. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

[1] Bernardo Aceituno-Cabezas, Jose Ballester, and Alberto Rodriguez. Certified grasping. *The International Journal of Robotics Research*, 42(4-5):249–262, 2023.

[2] J Angeles and C Lopez-Cajun. The dexterity index of serial-type robotic manipulators. *ASME Trends and Developments in Mechanisms, Machines and Robotics*, 7984, 1988.

[3] Haruhiko Asada and J Granito. Kinematic and static characterization of wrist joints and their optimal design. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 244–250. IEEE, 1985.

[4] Robert C Beach. *An introduction to the curves and surfaces of computer-aided design*. Van Nostrand Reinhold Co., 1991.

[5] David N Beal, Franz S Hover, Michael S Triantafyllou, James C Liao, and George V Lauder. Passive propulsion in vortex wakes. *Journal of Fluid Mechanics*, 549:385–402, 2006.

[6] Lionel Birglen, Thierry Laliberté, and Clément M Gosselin. *Underactuated robotic hands*, volume 40. Springer, 2007.

[7] Tianjian Chen, Zhanpeng He, and Matei Ciocarlie. Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning. *arXiv preprint arXiv:2008.04460*, 2020.

[8] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[9] Matei Ciocarlie and Peter Allen. A design and analysis tool for underactuated compliant hands. In *2009 IEEE/RSJ International conference on intelligent robots and systems*, pages 5234–5239. IEEE, 2009.

[10] Matei Ciocarlie and Peter Allen. Data-driven optimization for underactuated robotic hands. In *2010 IEEE International Conference on Robotics and Automation*, pages 1292–1299. IEEE, 2010.

[11] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[12] Keith L Doty, Claudio Melchiorri, Eric M Schwartz, and Claudio Bonivento. Robot manipulability. *IEEE Transactions on Robotics and Automation*, 11(3):462–468, 1995.

[13] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items, 2022. URL https://arxiv.org/abs/2204.11918.

[14] Gilbert Feng, Hongbo Zhang, Zhongyu Li, Xue Bin Peng, Bhuvan Basireddy, Linzhu Yue, ZHITAO SONG, Lizhi Yang, Yunhui Liu, Koushil Sreenath, and Sergey Levine. Genloco: Generalized locomotion controllers for quadrupedal robots. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1893–1903. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/feng23a.html.

[15] Eugene F Fichter. A stewart platform-based manipulator: general theory and practical construction. *The international journal of robotics research*, 5(2):157–182, 1986.

[16] Hetal N Fitter, Akash B Pandey, Divyang D Patel, and Jitendra M Mistry. A review on approaches for handling bezier curves in cad for manufacturing. *Procedia Engineering*, 97:1155–1166, 2014.

[17] Kenneth Y Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10(2-4):201–225, 1993.

[18] Clement Gosselin and Jorge Angeles. The optimum kinematic design of a planar three-degree-of-freedom parallel manipulator. 1988.

[19] Clement Gosselin, Frederic Pelletier, and Thierry Laliberte. An anthropomorphic underactuated robotic hand with 15 dofs and a single actuator. In *2008 IEEE International Conference on Robotics and Automation*, pages 749–754. IEEE, 2008.

[20] Huy Ha, Shubham Agrawal, and Shuran Song. Fit2form: 3d generative model for robot gripper form design. In *Conference on Robot Learning*, pages 176–187. PMLR, 2021.

[21] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697*, 2018.

[22] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[23] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[24] HaoTse Hsiao, Jiefeng Sun, Haijie Zhang, and Jianguo Zhao. A mechanically intelligent and passive gripper for aerial perching and grasping. *IEEE/ASME Transactions on Mechatronics*, 27(6):5243–5253, 2022.

[25] Jiaheng Hu, Julian Whitman, Matthew Travers, and Howie Choset. Modular robot design optimization with generative adversarial networks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4282–4288. IEEE, 2022.

[26] Jiaheng Hu, Julian Whitman, and Howie Choset. Glso: Grammar-guided latent space optimization for sample-efficient robot design automation. In *Conference on Robot Learning*, pages 1321–1331. PMLR, 2023.

[27] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*, pages 6265–6271. IEEE, 2019.

[28] Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4455–4464. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/huang20d.html.

[29] Steve C Jacobsen, John E Wood, DF Knutti, and Klaus B Biggers. The utah/mit dextrous hand: Work in progress. *The International Journal of Robotics Research*, 3(4):21–50, 1984.

[30] Sung-Gaun Kim and Jeha Ryu. New dimensionally homogeneous jacobian matrix formulation by three end-effector points for optimal design of parallel manipulators. *IEEE Transactions on Robotics and Automation*, 19 (4):731–736, 2003.

[31] Charles A Klein and Bruce E Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *The international journal of robotics research*, 6(2):72–83, 1987.

[32] Milin Kodnongbua, Ian Good Yu Lou, Jeffrey Lipton, and Adriana Schulz. Computational design of passive grippers. *arXiv preprint arXiv:2306.03174*, 2023.

[33] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[34] Mengxi Li, Rika Antonova, Dorsa Sadigh, and Jeannette Bohg. Learning tool morphology for contact-rich manipulation tasks with differentiable simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1859–1865. IEEE, 2023.

[35] Hong Liu, Ke Wu, Peter Meusel, Nikolaus Seitz, Gerd Hirzinger, MH Jin, YW Liu, SW Fan, T Lan, and ZP Chen. Multisensory five-finger dexterous hand: The dlr/hit hand ii. In *2008 IEEE/RSJ international conference on intelligent robots and systems*, pages 3692–3697. IEEE, 2008.

[36] Qiujie Lu, Nicholas Baron, Guochao Bai, and Nicolas Rojas. Mechanical intelligence for adaptive precision grasp. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4530–4536. IEEE, 2021.

[37] Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *Conference on Robot Learning*, pages 854–869. PMLR, 2020.

[38] Ou Ma and Jorge Angeles. Optimum architecture design of platform manipulators. In *Fifth International Conference on Advanced Robotics' Robots in Unstructured Environments*, pages 1130–1135. IEEE, 1991.

[39] Tad McGeer. Passive dynamic walking. *The international journal of robotics research*, 9(2):62–82, 1990.

[40] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[41] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[42] Rolf Pfeifer and Gabriel Gómez. Morphological computation–connecting brain, body, and environment. *Creating brain-like intelligence: From basic principles to complex intelligent systems*, pages 66–83, 2009.

[43] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[44] Alberto Rodriguez and Matthew T Mason. Effector form design for 1dof planar actuation. In *2013 IEEE International Conference on Robotics and Automation*, pages 349–356. IEEE, 2013.

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[47] J Kenneth Salisbury and John J Craig. Articulated hands: Force control and kinematic issues. *The International journal of Robotics research*, 1(1):4–17, 1982.

[48] Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9798–9805. IEEE, 2019.

[49] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[50] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[51] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[52] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

[53] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[54] Andrew Spielberg, Allan Zhao, Yuanming Hu, Tao Du, Wojciech Matusik, and Daniela Rus. Learning-in-the-loop optimization: End-to-end control and co-design of soft robots through learned deep latent representations. *Advances in Neural Information Processing Systems*, 32, 2019.

[55] Leo J Stocco, Septimiu E Salcudean, and Farrokh Sassani. On the use of scaling matrices for task-specific robot design. *IEEE Transactions on Robotics and Automation*, 15(5):958–965, 1999.

[56] HJ Terry Suh, Max Simchowitz, Kaiqing Zhang, Tao Pang, and Russ Tedrake. Pathologies and challenges of using differentiable simulators in policy optimization for contact-rich manipulation. In *ICRA 2022 Workshop: Reinforcement Learning for Contact-Rich Manipulation*, 2022.

[57] Orion Taylor and Alberto Rodriguez. Optimal shape and motion planning for dynamic planar manipulation. *Autonomous Robots*, 43:327–344, 2019.

[58] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent*

Robots and Systems, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

[59] Nathan Thatcher Ulrich. Grasping with mechanical intelligence. Technical report, 1988.

[60] Tsun-Hsuan Wang, Juntian Zheng, Pingchuan Ma, Yilun Du, Byungchul Kim, Andrew Spielberg, Joshua Tenenbaum, Chuang Gan, and Daniela Rus. Diffusebot: Breeding soft robots with physics-augmented generative diffusion models. *arXiv preprint arXiv:2311.17053*, 2023.

[61] Julian Whitman, Raunaq Bhirangi, Matthew Travers, and Howie Choset. Modular robot design synthesis with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10418–10425, 2020.

[62] Adam Wolniakowski, Jimmy A Jorgensen, Konstantsin Miatliuk, Henrik Gordon Petersen, and Norbert Kruger. Task and context sensitive optimization of gripper design using dynamic grasp simulation. In *2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 29–34. IEEE, 2015.

[63] Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. An end-to-end differentiable framework for contact-aware robot design. *arXiv preprint arXiv:2107.07501*, 2021.

[64] Jie Xu, Andrew Spielberg, Allan Zhao, Daniela Rus, and Wojciech Matusik. Multi-objective graph heuristic search for terrestrial robot design. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 9863–9869. IEEE, 2021.

[65] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.

[66] Kevin Zakka. Scanned Objects MuJoCo Models, 7 2022. URL https://github.com/kevinzakka/mujoco_scanned_objects.

[67] Mike Tao Zhang and Ken Goldberg. Gripper point contacts for part alignment. *IEEE Transactions on Robotics and Automation*, 18(6):902–910, 2002.

[68] Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

[69] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Appendix

I. Tasks and metrics

We provide more details about the evaluation manipulation tasks, and introduce more metrics for each task in this section.

Primitive objectives:

- **Shift up**: Make the object shift upward (along the negative x-axis) under all initial poses. Metrics are the average success rate after one closure $S_x^- \uparrow (\%)$, average delta translation along the x-axis after one closure $\Delta x \downarrow (cm)$, and average final x coordinate after the 40th gripper closure $x \downarrow (cm)$. The metrics are averaged among trajectories with different initial poses. The motion objective is $f(o,m,p) = -\Delta x(o,m,p)$, then the design objective can be aggregated from $f(o,m,p)$ as $F(m) = \sum_o \sum_p f(o,m,p)$.
- **Shift down**: Make the object shift downward (along the positive x-axis) under all initial poses. Metrics are average success rate $S_x^+ \uparrow (\%)$, average delta translation along the x-axis $\Delta x \uparrow (cm)$, and average final x coordinate $x \uparrow (cm)$. The motion objective is $f(o,m,p) = \Delta x(o,m,p)$.
- **Shift left**: Make the object shift leftward (along the negative y-axis) under all initial poses. Metrics are average success rate $S_y^- \uparrow (\%)$, average delta translation along the y-axis $\Delta y \downarrow (cm)$, and average final y coordinate $y \downarrow (cm)$. The motion objective is $f(o,m,p) = -\Delta y(o,m,p)$.
- **Shift right**: Make the object shift rightward (along the positive y-axis) under all initial poses. Metrics are average success rate $S_y^+ \uparrow (\%)$, average delta translation along the y-axis $\Delta y \uparrow (cm)$, and average final y coordinate $y \uparrow (cm)$. The motion objective is $f(o,m,p) = \Delta y(o,m,p)$.
- **Rotate clockwise**: Make the object rotate clockwise (negative delta rotation around the z-axis) under all initial poses. Metrics are average success rate $S_\theta^- \uparrow (\%)$, average delta rotation around the z-axis $\Delta\theta \downarrow (^\circ)$, and average final orientation $\theta \downarrow (^\circ)$. The motion objective is $f(o,m,p) = -\Delta\theta(o,m,p)$.
- **Rotate counterclockwise**: Make the object rotate counterclockwise (positive delta rotation around the z-axis) under all initial poses. Metrics are average success rate $S_\theta^+ \uparrow (\%)$, average delta rotation around the z-axis $\Delta\theta \uparrow (^\circ)$, and average final orientation $\theta \uparrow (^\circ)$. The motion objective is $f(o,m,p) = \Delta\theta(o,m,p)$.

Complex objectives:

- **Rotate**: Make the object rotate either clockwise or counterclockwise under all initial poses. The metrics are average success rate $S_\theta^{+-} \uparrow (\%)$, the average absolute value of delta rotation around the z-axis $|\Delta\theta| \downarrow (^\circ)$, and the average absolute value of final orientation $|\theta| \downarrow (^\circ)$. The motion objective is $f(o,m,p) = [\Delta\theta(o,m,p)]^2$.
- **Rotate clockwise and shift up**: Make the object rotate clockwise and shift up under all initial poses. The metrics are average success rate $S_\theta^- \& S_x^- \uparrow (\%)$, average delta rotation around the z-axis $\Delta\theta \downarrow (^\circ)$, average final orientation $\theta \downarrow (^\circ)$, average delta translation along the x-axis after one closure $\Delta x \downarrow (cm)$, and average final x coordinate after the 40th closure $x \downarrow (cm)$. The motion objective is $f(o,m,p) = -\Delta\theta(o,m,p) - \Delta x(o,m,p)$.
- **Rotate clockwise and shift left**: Make the object rotate clockwise and shift left under all initial poses. The metrics are average success rate $S_\theta^- \& S_y^- \uparrow (\%)$, average delta rotation around the z-axis $\Delta\theta \downarrow (^\circ)$, average final orientation $\theta \downarrow (^\circ)$, average delta translation along the y-axis $\Delta y \downarrow (cm)$, and average final y coordinate $y \downarrow (cm)$. The motion objective is $f(o,m,p) = -\Delta\theta(o,m,p) - \Delta y(o,m,p)$.
- **Convergence**: Make the object always end in a fixed final pose under a range of initial poses. The metric is the maximum convergence range $R_c^{max} \uparrow (^\circ)$, the largest range of initial orientations leading to a consistent final orientation within a small tolerance. We report the maximum convergence range within the tolerance of 3°, 5°, and 10°, respectively.

II. Additional results

We report results on all the tasks and metrics described above in Tab. III, Tab. IV, Tab. V, and Tab. VI. The evaluation procedure is the same as described in the main paper, where we run each approach 16 times per object-task pair and select the best performance, then average among test objects. DGDM outperforms baselines consistently on both discrete metrics (*e.g.* average success rate) and continuous metrics (*e.g.* delta transformation and final transformation).

		up			down			left			right			clock			counter		
		$S_x^- \uparrow$	$\Delta x \downarrow$	$x \downarrow$	$S_x^+ \uparrow$	$\Delta x \uparrow$	$x \uparrow$	$S_y^- \uparrow$	$\Delta y \downarrow$	$y \downarrow$	$S_y^+ \uparrow$	$\Delta y \uparrow$	$y \uparrow$	$S_\theta^- \uparrow$	$\Delta \theta \downarrow$	$\theta \downarrow$	$S_\theta^+ \uparrow$	$\Delta \theta \uparrow$	$\theta \uparrow$
	Unguided	56.8	-0.2	-1.3	82.1	0.3	1.9	82.9	-0.5	-1.2	80.4	0.5	1.4	46.9	-1.5	-9.5	58.5	2.3	11.1
2D	Opt.	79.5	-0.4	-2.2	53.3	0.4	2.3	81.3	-0.6	-2.4	94.0	0.7	1.7	48.8	**-2.9**	-8.8	73.2	3.6	9.6
	DGDM	**88.2**	**-0.4**	**-3.1**	**92.0**	**0.5**	**3.7**	**96.7**	**-0.7**	**-2.4**	**97.7**	**0.8**	**2.1**	**60.8**	-2.6	**-12.3**	**72.0**	**3.6**	**14.2**
	Unguided	43.0	-0.1	-0.6	43.8	0.1	1.0	80.4	-0.2	-1.2	87.9	0.2	0.9	41.2	-1.1	-4.9	33.5	0.5	2.7
3D	Opt.	47.4	-0.1	-1.3	66.3	0.2	1.7	86.3	-0.3	-1.3	88.1	0.3	1.6	59.1	-1.9	-5.0	52.0	1.2	4.6
	DGDM	**81.5**	**-0.2**	**-1.6**	**75.1**	**0.2**	**1.7**	**95.1**	**-0.4**	**-1.8**	**97.2**	**0.4**	**1.9**	**69.9**	**-2.3**	**-7.7**	**65.0**	**2.2**	**6.3**

TABLE III: Single Object Primitive Objectives Evaluation

		rotate			clock-up					clock-left					convergence		
		$S_\theta^{+-} \uparrow$	$\|\Delta\theta\| \uparrow$	$\|\theta\| \uparrow$	$S_\theta^- \& S_x^- \uparrow$	$\Delta\theta \downarrow$	$\theta \downarrow$	$\Delta x \downarrow$	$x \downarrow$	$S_\theta^- \& S_y^+ \uparrow$	$\Delta\theta \downarrow$	$\theta \downarrow$	$\Delta y \downarrow$	$y \downarrow$	$R_c^{max}(3°) \uparrow$	$R_c^{max}(5°) \uparrow$	$R_c^{max}(10°) \uparrow$
	Unguided	74.0	3.5	18.2	36.4	-1.5	-9.5	-0.2	-1.3	36.9	-1.5	-9.5	-0.5	-1.2	56.5	61.7	68.7
2D	Opt.	78.9	4.5	18.9	29.3	-1.8	-4.2	-0.3	-1.0	49.4	-2.9	-9.0	-0.6	-2.1	69.6	73.7	82.1
	DGDM	**79.3**	**4.5**	**20.1**	**62.7**	**-3.3**	**-14.5**	**-0.4**	**-3.6**	**63.7**	**-3.2**	**-10.1**	**-0.7**	**-2.4**	**78.4**	**83**	**89.3**
	Unguided	64.2	2.2	16.5	30.3	-1.1	-4.9	-0.1	-0.6	33.1	-0.5	-2.7	-0.2	-1.2	52.2	63.6	67.8
3D	Opt.	66.9	2.4	15.7	29.2	-1.1	-2.7	-0.1	-0.5	37.7	-1.3	-3.3	-0.3	-1.0	50.3	60	72.3
	DGDM	**83.0**	**3.1**	**21.0**	**57.1**	**-2.4**	**-6.8**	**-0.2**	**-0.9**	**58.2**	**-2.9**	**-11.1**	**-0.5**	**-1.7**	**68.8**	**72.5**	**81.5**

TABLE IV: Single Object Complex Objectives Evaluation

		up			down			left			right			clock			counter		
		$S_x^- \uparrow$	$\Delta x \downarrow$	$x \downarrow$	$S_x^+ \uparrow$	$\Delta x \uparrow$	$x \uparrow$	$S_y^- \uparrow$	$\Delta y \downarrow$	$y \downarrow$	$S_y^+ \uparrow$	$\Delta y \uparrow$	$y \uparrow$	$S_\theta^- \uparrow$	$\Delta \theta \downarrow$	$\theta \downarrow$	$S_\theta^+ \uparrow$	$\Delta \theta \uparrow$	$\theta \uparrow$
	Unguided	55.8	-0.2	-1.3	79.8	0.3	1.3	77.1	-0.5	-1.1	80.3	0.5	1.3	44.7	-1.5	-3.5	56.4	2.1	6.2
2D	Opt.	78.6	-0.4	-1.0	50.3	0.3	1.0	79.2	-0.6	-1.0	93.8	0.7	1.7	46.1	**-2.9**	-7.8	71.4	**3.5**	7.4
	DGDM	**83.8**	**-0.4**	**-3.0**	**88.1**	**0.4**	**3.4**	**99.3**	**-0.7**	**-2.5**	**94.3**	**0.7**	**2.1**	**61.3**	-2.4	**-10.5**	**68.4**	3.3	**16.5**
	Unguided	40.1	-0.1	-0.5	40.8	0.1	0.9	75.8	-0.2	-0.8	87.9	0.2	0.6	34.7	-0.5	-0.8	29.2	0.1	2.5
3D	Opt.	42.4	-0.1	-0.4	66.4	0.2	1.0	77.9	-0.2	-0.4	86.6	0.3	0.8	40.3	-1.1	-1.9	39.2	**1.9**	3.5
	DGDM	**89.7**	**-0.2**	**-1.5**	**66.8**	**0.2**	**1.2**	**96.1**	**-0.5**	**-1.8**	**95.4**	**0.4**	**1.3**	**69.3**	**-2.0**	**-5.2**	**58.2**	1.6	**3.5**

TABLE V: Multi-object Primitive Objectives Evaluation

		rotate			clock-up					clock-left				
		$S_\theta^{+-} \uparrow$	$\|\Delta\theta\| \uparrow$	$\|\theta\| \uparrow$	$S_\theta^- \& S_x^- \uparrow$	$\Delta\theta \downarrow$	$\theta \downarrow$	$\Delta x \downarrow$	$x \downarrow$	$S_\theta^- \& S_y^+ \uparrow$	$\Delta\theta \downarrow$	$\theta \downarrow$	$\Delta y \downarrow$	$y \downarrow$
	Unguided	68.3	3.2	13.4	35.2	-1.5	-3.5	-0.2	-0.8	35.2	-1.0	-3.0	-0.5	-1.0
2D	Opt.	74.3	3.8	7.2	25.0	-1.4	-0.1	-0.1	-0.2	49.0	-2.9	-4.6	-0.4	-0.7
	DGDM	**78.4**	**4.2**	**15.8**	**62.4**	**-3.3**	**-10.6**	**-0.4**	**-3.6**	**63.8**	**-3.0**	**-6.3**	**-0.7**	**-2.4**
	Unguided	61.7	2.1	15.8	29.2	-0.8	-0.4	-0.1	-0.5	25.2	0.1	2.5	-0.2	-0.7
3D	Opt.	67.0	2.3	9.5	22.6	0.1	-0.1	-0.1	-0.3	34.3	-0.9	-1.2	-0.2	-0.5
	DGDM	**77.6**	**2.5**	**21.6**	**44.2**	**-1.6**	**-7.6**	**-0.2**	**-1.2**	**37.9**	**-2.3**	**-8.9**	**-0.5**	**-2.3**

TABLE VI: Multi-object Complex Objectives Evaluation