

Figure 8: Qualitative results of *D-Cubed* using the LEAP hand in a real-world experiment. The LEAP hand effectively deforms the object, exhibiting similar deformation as observed in the simulation.

A Additional Analysis

A.1 Ablation of Additional Gradient Guidance

The performance disparity when *D-Cubed* also uses gradient guidance with the proposed gradient-free sampling is reported in Fig. 7. As gradient guidance does not demonstrate a statistically significant improvement in the score compared to *D-Cubed* without gradient guidance, the increased time required by the simulator to calculate gradients used for gradient guidance is not warranted. Additionally, this result adds further evidence to the premise that gradients from differentiable simulators are often sparse and uninformative [13].

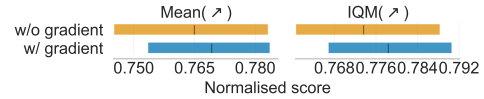


Figure 7: Comparison of performance with and without additional gradient guidance in our method. We report Mean and Interquartile Mean (IQM) of improvement in EDM averaged across all six tasks.

A.2 Qualitative Results in Real-World Environments

We qualitatively investigate whether an optimised trajectory from simulation can be transferred to real-world environments. Due to hardware limitations, we use a LEAP hand, a low-cost dexterous hand [51], instead of the Shadow hand [21]. Thus, the simulator is modified by replacing the Shadow hand with the LEAP hand. We evaluate the trajectory transfer on the *Flip* task, as this benchmark task makes the least number of simplifying assumptions compared to the real world. The other tasks permit object interpenetration with the table and unrealistic floating behaviour of the deformable object, rendering the evaluation impractical. In this experiment, given the known start state of the deformable object, we transfer the sequence of actions optimised in simulation to the real-world environment and control the hand in an open-loop manner. As shown in Fig. 8, the hand successfully *flips* the deformable object so that the object is folded within the hand in the real-world environment.

B Data collection Details

A task-agnostic play dataset of representative robot hand motions, including finger closing and opening and wrist movement, is collected. We use RGB data from a RealSense D435 camera to track human hand motion and re-target the human hand pose to a robot hand in the SAPIEN simulator [52], inspired by prior work [39]. We collect the play data for a duration of only 20 minutes, which corresponds to around 50K data points.

511 C Training Details

512 C.1 VAE

513 The VAE encoder and decoder both consist of a 4 layer LSTM [53] with 256 neurons per layer. In
 514 this work, we use a subsequence of actions with $H = 10$ to learn the skill-latent space. The VAE is
 515 trained using the Adam optimiser [54] with a learning rate of $1e-4$.

516 C.2 Latent Diffusion Models

517 We use the transformer architecture used in NanoGPT [55]. We report further hyperparameter details
 518 of the transformer denoiser network and diffusion in Table 2 and Table 3.

Table 2: Transformer Denoiser Network Hyperparameters

Parameter	Value
Optimiser	Adam
Learning rate	$1e-4$
Minibatch size	256
Embedding dimension	312
Batch size	256
Number of layers	6
Self-attention heads	4

Table 3: Diffusion Hyperparameters

Parameter	Value
Number of diffusion timesteps	200
Noise schedule	cosine
Noise schedule parameters s	0.008

519 C.3 D-Cubed Details

520 We report hyperparameters of D-Cubed used during trajectory optimisation steps in Table 4.

Table 4: D-Cubed Optimisation Hyperparameters

Parameter	Value
Number of diffusion timesteps	200
Number of samples	5

521 D Baseline Method Details

522 As we report the scores for gradient-based trajectory optimisation (TrajOpt) and PPO from prior
 523 work [14], we refer the reader to the prior work for further details.

524 D.1 MPPI

525 *MPPI* baseline samples 30 trajectories with a horizon of 15 steps. These parameters are chosen
 526 because they result in an optimisation time similar to *D-Cubed*. We report the hyperparameters for

MPPI in Table 5 in detail. In this experiment, we employ the publicly available MPPI implementation¹.

Table 5: MPPI and Skill-based MPPI Hyperparameters

Parameter	Value
planning horizon	15
Number of samples	30
Temperature	1.0
Initial noise mean	0.0
Initial noise std	1.0

D.2 Skill-based MPPI

Skill-based MPPI baseline samples skill-latent representations for effective exploration of the state space. We use the same hyperparameters as those of MPPI, except that the action sampled from a Gaussian distribution is skill-latent representations instead of low-level actions.

D.3 LDM w/ Gradient Guidance

LDM w/ Gradient Guidance baseline leverages gradient guidance [31] to generate a desired trajectory. In particular, first-order gradients from the differentiable physics simulator are used to guide the reverse process of the latent diffusion model. In our experiments, we denoise a noisy trajectory without gradient guidance for the first half of the diffusion steps so that a relatively clean trajectory can be obtained. For the rest of the diffusion steps, the following gradient guidance is applied:

$$\nabla_{\mathbf{x}_i} \log p_{\alpha_i}(\mathbf{x}_i|y) = \nabla_{\mathbf{x}_i} \log p_{\alpha_i}(\mathbf{x}_i) + \gamma \nabla_{\mathbf{x}_i} \log p(y|\mathbf{x}_i). \quad (5)$$

where y is the cost of the trajectory, $\nabla_{\mathbf{x}_i} \log p(y|\mathbf{x}_i)$ corresponds to the first-order gradients obtained from differentiable physics simulators, and γ is the scale of the gradient guidance. In our experiment, we use $\gamma = 1e-4$.

D.4 Diffusion-ES

Diffusion-ES, concurrent research [47], also optimises a trajectory using gradient-free guided sampling with a truncated diffusion process. While the prior work chooses the last trajectory of the optimisation process as output, we observe that it is often worse than the trajectories found in the middle of optimisation iterations. Thus, we report the score of the best trajectory found during the trajectory optimisation process. The hyperparameters used in Diffusion-ES is reported in Table 6. Following the original work [47], initially, the diffusion mutation steps start from 5 and the mutation step is linearly decayed to 1 over 200 search steps.

Table 6: Diffusion ES Hyperparameters

Parameter	Value
Mutation diffusion start steps	5
Mutation diffusion final steps	1
Population	5
Optimisation steps	200

¹https://github.com/UM-ARM-Lab/pytorch_mppi/tree/master

E Gradient-Free Guided Sampling for Trajectory Optimisation

Algorithm 2 is a complete version of gradient-free guided sampling for trajectory optimisation in *D-Cubed*. To determine the best sequence of skill latent representations $\mathbf{z}_{best}^{1:T_{skill}}$, each skill sequence in a batch of B skill trajectories is evaluated in simulation, and the skill sequence that minimises a cost is selected as the best sequence (Line 5).

Algorithm 2 Gradient-Free Guided Sampling for Trajectory Optimisation in Reverse Diffusion Process

```

1: Require: denoising model,  $G_\theta$ ; target state of deformable objects,  $\mathbf{s}_{target}$ ,  $T_{skill} = \frac{T}{H}$ 
2: Initialise:  $C_{best} = \infty$ ,  $\mu_{best} = \text{None}$ 
3:  $\{\mathbf{z}_N^1, \dots, \mathbf{z}_N^{T_{skill}}\}^{|B|} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$   $\triangleright$  Sample  $B$  initial sequences of latent representations
4: for  $i = N, N-1, \dots, 1$  do
5:    $\mathbf{z}_{best}^{1:T_{skill}} \leftarrow \text{FINDBESTLATENTS}(\{\mathbf{z}_i^{1:T_s}\}^{|B|})$   $\triangleright$  Choose the best sequence of skill latents
6:    $\mu_i \leftarrow \mu_\theta(\mathbf{z}_{best}^{1:T_{skill}})$   $\triangleright$  Predict a mean of a Gaussian distribution (see Eq. 4)
7:    $\text{cost} = \text{evaluate}(q_\theta^{dec}(\mathbf{a}^{1:T}|\mu_i))$   $\triangleright$  Evaluate the predicted mean
8:   if  $\text{cost} < C_{best}$  then
9:      $\mu_{best} \leftarrow \mu_i$ ,  $C_{best} \leftarrow \text{cost}$ 
10:   $\{\mathbf{z}_{i-1}^1, \dots, \mathbf{z}_{i-1}^{T_{skill}}\}^{|B|} \sim \mathcal{N}(\mu_{best}, \sigma_{i-1}^2 \mathbf{I})$   $\triangleright$  Sample a batch  $B$  of sequences of skill latents
11: return  $p_\psi^{dec}(\mathbf{a}^{1:T}|\mu_{best})$ 
12: function  $\text{FINDBESTLATENTS}(\{\mathbf{z}^1, \dots, \mathbf{z}^{T_{skill}}\}^{|B|})$ 
13:    $\text{cost}_{best}, \mathbf{z}_{best} = \infty, \text{None}$ 
14:   for  $\mathbf{z}^{1:T_{skill}} = \{\mathbf{z}^1, \dots, \mathbf{z}^{T_{skill}}\}^{|B|}$  do  $\triangleright$  Evaluate each sequence of latent representations in
the batch
15:      $\text{cost}_j = \text{evaluate}(p_\psi^{dec}(\mathbf{a}^{1:T}|\mathbf{z}^{1:T_{skill}}))$ 
16:     if  $\text{cost}_j < \text{cost}_{best}$  then
17:        $\text{cost}_{best} \leftarrow \text{cost}_j$ ,  $\mathbf{z}_{best} \leftarrow \mathbf{z}^{1:T_{skill}}$ 
return  $\mathbf{z}_{best}$ 

```

F Task Details

F.1 Cost Function

The cost function used for trajectory optimisation is defined by Sinkhorn Divergence. Following the prior work [14], the *geomloss* library is used to define the cost function:

```

559 1 from geomloss import SamplesLoss
560 2 OT_LOSS = SamplesLoss(loss="sinkhorn", p=1, blur=0.0001)

```

F.2 Tasks

For single-hand task, such as *Folding* and *Wrap*, the action dimension is 26 (20 for actuators including finger joints and wrist, and 6 for the base). For in-hand manipulation tasks (*Flip*), a single hand with a fixed base is assumed, resulting in an action dimension of 20. In dual-hand environments, the action dimension is 52, allowing for a movable base for both hands. Since a VAE is trained to encode a single-arm action trajectory in the play dataset, an LDM generates a single-arm skill trajectory. Thus, to handle dual-hand tasks using *D-Cubed*, the LDM generates a trajectory for each arm. In the following, we describe the details of each task.

Folding: The initial position of the robot hand is above the dough, and the hand must fold the dough in four different directions: front, back, left, and right.

Wrap: The robot hand first picks up the plasticine ball and places it onto the dough shaped like a rope. Then, it pinches the side of the rope to wrap the ball inside it.

Flip: The robotic hand tosses the dough in the air to reshape and reposition it.

574 **Bun:** The two robotic hands deftly pinch and push the dough to form a bun-shaped object.

575 **Rope:** The right-hand grasps the rope on the right, lifts it, and places it above the left rope. Then,
576 the left-hand bends the left rope.

577 **Dumpling:** To wrap a dumpling, the right hand first grasps the right side of the dough. While
578 holding the dough with the right hand, the left hand lifts the left side of the dough. Finally, the two
579 hands bring the two sides of the dough together and form it into a dumpling shape.