

FoldNet: Learning Generalizable Closed-Loop Policy for Garment Folding via Keypoint-Driven Asset and Demonstration Synthesis

Yuxing Chen, Bowen Xiao, and He Wang

Abstract—Due to the deformability of garments, generating a large amount of diverse and high-quality data for robotic garment manipulation tasks is highly challenging. In this paper, we present *FoldNet*, a synthetic garment dataset that includes assets for four categories of clothing as well as high-quality closed-loop folding demonstrations. We begin by constructing geometric garment templates based on keypoints and applying generative models to generate realistic texture patterns. Leveraging these garment assets, we generate folding demonstrations in simulation and train folding policies via closed-loop imitation learning. To improve robustness, we introduce *KG-Dagger*, a keypoint-based strategy for generating recovery demonstrations after failures. *KG-Dagger* significantly improves the quality of generated demonstrations and the model performance, boosting the real-world success rate by 25%. After training with 15K trajectories (about 2M image-action pairs), the model achieves a 75% success rate in the real world. Experiments in both simulation and real-world settings validate the effectiveness of our proposed dataset.

Index Terms—Bimanual manipulation, deep learning for visual perception, deep learning in grasping and manipulation.

I. INTRODUCTION

GARMENT manipulation has been widely studied in robotics [1]. Here is a difference, due to the deformable nature of garments, such tasks remain highly challenging. In recent years, data-driven learning approaches have made significant progress, with imitation learning [2] gradually emerging as the main paradigm for the acquisition of various robotic skills. Some prior works [3], [4] have demonstrated strong garment manipulation capabilities using imitation learning. However, enabling the learned policy to generalize to unseen environments and objects remains hindered by the scarcity of large-scale, diverse, high-quality demonstration data.

Learning from synthetic data has become an efficient approach for robot learning [5], [6]. Many datasets [7]–[9] and simulation environments [10], [11] are now available to generate garment manipulation data. In simulation, it is possible to flexibly modify both the environment and the properties of the garments, allowing stronger generalization capabilities. However, improving the quality of synthetic data

remains a key challenge. Current methods face two main limitations:

Limited garment assets and lack of detailed annotations. Existing datasets often contain only a small number of garment meshes and lack rich annotations. This limited data availability imposes an upper bound on generalization performance. Moreover, the lack of detailed annotations requires researchers to put in extra effort to generate high-quality demonstration data in simulation.

Limited handling of error recovery. Garment manipulation tasks are long-horizon tasks that involve complex deformable object dynamics. Compared to previously dominant open-loop approaches for garment manipulation [5], [12], closed-loop control offers the potential to retry after failures. However, if training data only contains perfect demonstrations, small errors at each step can accumulate and potentially cause the garment to enter previously unseen states, often resulting in task failure. This poses a significant challenge for learning robust policies.

To address the scarcity of garment assets, we propose a novel framework for generating garment assets. For each category of garment, we design a template whose geometry is controlled by a set of keypoints. We then apply generative models to synthesize texture maps for garments. This approach enables scalable garment mesh generation, and each mesh is accompanied by automatically generated semantic keypoint annotations for subsequent demonstration generation and policy learning.

To handle out-of-distribution states, we introduce Keypoint-Gated DAgger (*KG-Dagger*). After training the initial policy network, we run the policy and use the previously automatically annotated keypoints to detect potential failure cases. When a failure is detected, a keypoint-based strategy is invoked to perform a correction. The corrected trajectories are then added to the dataset for further policy training. The final model is end-to-end: given the current observation, the model directly outputs the action sequence without requiring any additional hyperparameters.

In summary, this work makes the following two key contributions:

- 1) We propose a garment mesh generation framework that can automatically generate highly diverse garment meshes with annotated keypoints.
- 2) We introduce *KG-Dagger*, improving the data quality and boosting the success rate of closed-loop folding in the real world from 50% to 75%.

Manuscript received: January 9, 2026; Revised January 17, 2026; Accepted: January 7, 2026. This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers comments. This work was supported by Galbot. (Yuxing Chen and Bowen Xiao contributed equally to this work.) (Corresponding author: He Wang.)

The authors are with CFCS, School of Computer Science, Peking University, Beijing 100051, China, and also with Galbot, Beijing 100010, China.

E-mail: yuxingc_20@stu.pku.edu.cn; xiaobowenbowie@stu.pku.edu.cn; hewang@pku.edu.cn

Digital Object Identifier (DOI): see top of this page.



Fig. 1. **FoldNet** is a dataset designed for robotic garment manipulation. It provides (1) a large collection of synthetic garment assets with keypoint annotations, (2) high-quality folding demonstrations, and (3) keypoint-based error recovery demonstrations. Leveraging these assets, FoldNet supports a wide range of downstream tasks, including (a) keypoint detection, (b) folding in simulation, and (c) folding in the real world.

II. RELATED WORKS

A. Garment Manipulation

Garment manipulation is a widely studied task in robotics [1]. The main challenges arise from the deformable nature of garments and their complex dynamics. Various approaches have been explored, including imitation learning [13], reinforcement learning [6], and model-based methods [14], [15].

From a task perspective, many works focus on garment folding [9], [12], [13], [16] and unfolding [5], [17]. However, most approaches for solving folding or unfolding tasks rely on modular perception and control pipelines, which exhibit several limitations. Many rely on object point clouds [12], [16] and therefore depend on accurate camera calibration and depth information to achieve robust grasping, making recovery from failed grasps challenging. In addition, they often require numerous hand-designed hyperparameters — such as lift heights during folding — which are difficult to generalize across garments of varying sizes.

In recent years, closed-loop policies [3], [4], [18], [19] trained on large-scale real-world data have demonstrated strong capabilities in garment manipulation. However, collecting these datasets requires a large amount of human labor. Moreover, real-world data cannot offer the strong generalization that synthetic data can provide. In this work, we investigate how to generate high-quality synthetic demonstrations of garment manipulation for training closed-loop models.

B. Synthetic Garment Assets

Compared with rigid-body mesh assets [20], garment assets that can be physically simulated place much higher demands on mesh quality. Existing garment mesh datasets are typically designed manually by artists [8] or generated based on predefined templates [7], [9]. Though template-based methods allow

for large-scale mesh generation at substantially lower cost compared to manual design, they face significant challenges when applying realistic texture to the mesh. Previous template-based methods either directly apply existing texture libraries to garment meshes [9], or use generative models to synthesize textures [21], [22]. However, the textures generated by the first method differ significantly from those of real garments, whereas the second method perform poorly when applied to layered garment meshes. In this work, we adopt a template-based method to generate garment geometry and introduce a pipeline that facilitates generative models in producing scalable and realistic textures.

C. Imitation Learning

Imitation learning [23], [24] has received increasing attention from the research community. A key challenge lies in collecting high-quality demonstration data. Recent studies have shown that enabling models to recover from errors leads to better performance than naive imitation learning, making it a topic of great interest [25], [26]. Our method performs imitation learning in simulation by distilling a keypoint-based policy into a vision-based model, while improving robustness by generating demonstrations that incorporate recovery from failures.

III. GARMENT MESH SYNTHESIS

Our method begins with synthesizing high-quality garment meshes. These meshes need to be suitable for physical simulation and rendering. Our pipeline for garment generation is shown in Figure 2. The main steps include: (1) creating the geometry of the garment, (2) generating the texture of the garment, (3) combining the geometry and texture, and then filtering. Detailed descriptions of these stages are provided in III-A, III-B and III-C. To show the advantages of our

TABLE I

COMPARISON WITH OTHER SYNTHETIC DATASETS. THE TABLE COLUMNS INDICATE THE NUMBER OF GARMENT MESHES, NUMBER OF GARMENT CATEGORIES, INCLUSION OF RGB TEXTURES, MULTI-LAYER MESHES (FRONT LAYER AND BACK LAYER), SEMANTIC KEYPOINTS, AND MESH RESOLUTION. ¹THE NUMBERS HERE INDICATE THAT THE GARMENTS CAN BE GENERATED, WITH THE QUANTITY REPRESENTING THE NUMBER OF MESHES THAT CAN BE GENERATED IN ONE DAY ON A SINGLE RTX 3090. ²INCLUDING THE TIME REQUIRED FOR RENDERING. ³ALTHOUGH RGB DATA IS INCLUDED, IT DOES NOT MAINTAIN CONSISTENCY WITH THE CLOTHING GEOMETRY. ⁴THE RESOLUTION IS ADJUSTABLE.

Dataset	#M	#C	RGB	ML	SK	Res
ClothesNet [8]	3.1K	11	✓	✓	✗	1 cm
Cloth3D [7]	11.3K	4	✗	✓	✗	1 cm
aRTF [9]	10K/D ^{1,2}	3	✗ ³	✗	✓	Adj. ⁴
Ours	2K/D ¹	4	✓	✓	✓	Adj. ⁴

approach, we compare the resulting asset dataset with several existing datasets in Table I.

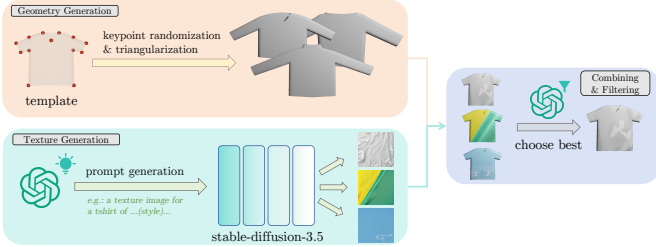


Fig. 2. **Pipeline for garment mesh synthesis.** By performing *geometry generation*, *texture generation*, *combining-and-filtering*, we can synthesize scalable, high-quality garment meshes.

A. Geometry Generation

We use a template-based approach to generate garment geometry of four types of garments — t-shirt (including long-sleeved and short-sleeved), vest (sleeveless), hoodie, and trousers. For each type of garment, the template is constructed by manually specifying a set of semantic keypoints, i.e., 2D positions (x, y) , that capture the structural characteristics of the garment. These keypoints serve a dual role: they identify semantically meaningful manipulation points on the garment and implicitly define its shape. Once the keypoint positions are determined, we connect them along the border using Bezier curves and perform triangulation within the xy-plane. Then, we heuristically define the z-coordinates and UV coordinates for the mesh vertices. During this process, keypoints are automatically annotated on the generated triangular mesh by saving the keypoint indices. With this generation method, we can generate a large variety of garment shapes with high efficiency by simply randomizing the positions of the keypoints.

B. Texture Generation

To automatically generate garment textures, we use pre-trained generative models. First, for each type of garment, we use a large language model [27] to generate a description of the texture. Then, we use this description as a prompt for a Text2Image model [28]. Repetition of this process multiple times can quickly generate a large number of texture images.



Fig. 3. **Synthetic garment meshes.** These static garment meshes can be used for subsequent physics simulation and policy learning.

C. Combining and Filtering

To enhance the consistency between texture images and garment meshes, we introduce an additional filtering step. For each garment mesh with only geometry, we combine it with different texture images and render the results. A vision language model [27] is then used to automatically select the most suitable texture as the final texture for that mesh. We present several examples of the final generated garments, as shown in Figure 3.

IV. DEMONSTRATION GENERATION

Using the generated garment assets, we design keypoint-based policies to automatically collect demonstrations in simulation. A vision-action model M_0 is then trained on these demonstrations via imitation learning. To improve data efficiency and policy robustness, we introduce KG-Dagger, a variant of DAgger. In KG-Dagger, at the i -th iteration, we use the current model M_i to generate new trajectories. During this process, a keypoint-based error recovery strategy is employed. These newly generated trajectories teach the model how to recover from errors—particularly the types of errors to which the model is most prone—thereby continually improving the model’s performance. In the final deployment, the model is trained on the entire set of trajectories. The final policy is an *end-to-end* system: it does not require explicit keypoint detection or error detection, as these capabilities are learned implicitly by the model.

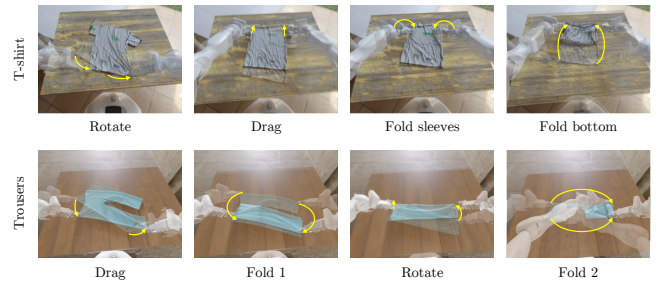


Fig. 4. **Keypoint-based demonstration generation.** The entire folding process can generally be divided into several stages, which are then executed sequentially during the demonstration generation process.

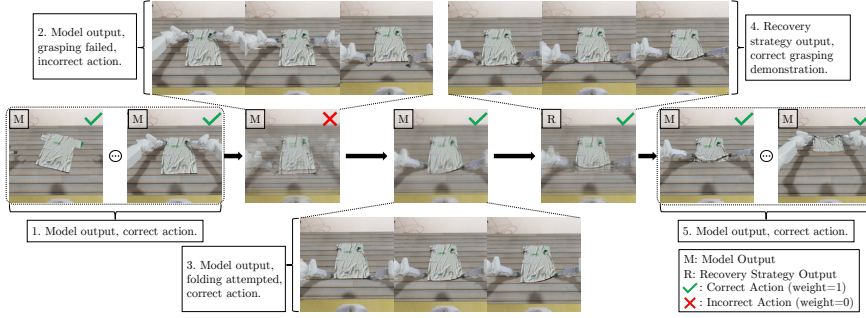


Fig. 5. **Keypoint-based recovery strategy.** The figure on the left illustrates an example of the recovery strategy. The recovery data are incorporated into the dataset and used jointly to train the end-to-end vision-action model. With these data, the model can learn how to retry when a grasp attempt fails. The figure on the right shows the pseudocode of KG-Dagger.

Algorithm 1: KG-DAGGER

Input: Keypoint-Based Policy π_K , Offline Dataset

```

1  $M_0 \leftarrow \text{Train}(D_{BC})$ 
2  $D \leftarrow D_{BC}$ 
3 for  $i \leftarrow 1$  to  $N_{iter}$  do
4   for timestep  $t \in T$  do
5     if KeypointBasedErrorDetected then
6        $a \leftarrow \pi_K(o)$ 
7     else
8        $a \leftarrow M_{i-1}(o)$ 
9      $Env.step(a)$ 
10     $D \leftarrow D \cup \{o, a\}$ 
11  $M_i \leftarrow \text{Train}(D)$ 

```

A. Keypoint-Based Demonstration Generation

For each garment category, we design a simple yet effective keypoint-based policy to fold the garment in a predefined manner. For example, in the case of a t-shirt, one possible folding strategy involves first rotating the garment, then dragging it, folding both sleeves inward, and finally folding the bottom of the shirt. Example demonstrations are shown in Figure 4. At each stage, the initial grasping points and target placement positions for the two grippers are derived from ground-truth keypoint locations, while intermediate positions are obtained through interpolation. Different folding strategies can be generated by modifying the initial and target positions at each stage. Owing to the keypoint annotations in our assets, this policy is unified across garments of the same category, independent of shape variations.

B. Error-Recovery Demonstration Generation

KG-Dagger is similar to HG-Dagger [25]: during model inference, we use a keypoint-based strategy to detect grasp failures — a step that is performed by a human expert in HG-Dagger. This monitoring process leverages the keypoints of the garment at each time step, along with the gripper state. Figure 5 illustrates in detail how the keypoint-based error recovery strategy is implemented. The entire recovery process is divided into five stages.

- In Stage 1, the model outputs correct actions, and the recovery strategy does not need to intervene.
- In Stage 2, the model outputs incorrect actions, failing to move the gripper to the correct position and thus causing a grasp failure. This stage covers the interval from the previous release of the gripper until the failed attempt when the gripper closes. These actions should not be encouraged; during training, we assign them a weight of zero.
- In Stage 3, the gripper fails to grasp the garment due to its incorrect position in the previous step. However, the model should still continue attempting: only after moving the gripper and observing that the garment does not move can the failure be detected. Therefore, the actions in this stage are still correct.
- In Stage 4, the recovery strategy takes over and retries the grasp. All actions in this stage are correct.

- In Stage 5, the error has been resolved, and the model resumes generating and executing actions.

By incorporating these error-recovery trajectories into the dataset for training, the model can learn how to retry grasping after a failure.

This KG-Dagger process is used only during the training phase and only in simulation. During testing in simulation and in the real world, we directly use the outputs of the vision-action model, without requiring KG-Dagger or keypoint detection.

C. Model Training

We choose diffusion policy [29] as our vision-action model for its compact size and good performance in modeling multi-modal behaviors and producing coherent action sequences. It is also possible to use other vision-action models, as our demonstration data do not require a specific model.

We retain only successful episodes and filter out failed ones. Here, an episode refers to the entire trajectory starting from the initial state and finally resulting in the garment being fully folded. At the end of each episode, we pad several no-op actions to indicate termination. The training loss is a modified version of the original diffusion loss [29]: $L_\theta = \sum_{i=1}^{T_a} m_i * ||\varepsilon_i^k - \varepsilon_\theta(O_t, A_t^0 + \varepsilon^k, k)_i||^2$. For the i -th action in an action chunk, we multiply the loss by a coefficient m_i . If a zero-weight action (due to a grasp failure) appears in the action chunk, then m_i for that action and all subsequent actions is set to 0; otherwise, m_i is 1. When all $m_i = 1$, the loss reduces to the original diffusion policy loss.

V. EXPERIMENTS

We design two tasks to validate the effectiveness of our method: keypoint detection and garment folding. The keypoint detection task is easier to benchmark and illustrates how closely the generated garment meshes resemble real-world garments. The garment folding task is more comprehensive and enables the evaluation of the quality of the generated demonstration data.

A. Keypoint Detection

1) Experiment Setup:

TABLE II
 QUANTITATIVE RESULTS OF KEYPOINT DETECTION ON REAL IMAGES. THE FIGURE ILLUSTRATES THE PERFORMANCE OF MODELS TRAINED USING DIFFERENT GARMENT MESH SYNTHESIS METHODS. IN THE *Average* ROW, WE HIGHLIGHT THE TOP TWO VALUES IN BOLD.

Category	mAP _{4,8} (↑)				AKD (↓)			
	Ours	w/o filter	aRTF	Paint-it	Ours	w/o filter	aRTF	Paint-it
T-Shirt	59.0	50.5	42.2	47.2	10.3	9.30	11.3	14.2
Trousers	51.7	57.0	47.4	47.8	16.9	16.4	14.1	32.0
Vest	42.5	43.5	26.0	37.3	20.0	16.7	17.3	43.7
Hoodie	35.7	32.3	31.0	29.5	19.8	20.1	18.9	35.9
Average	47.2	45.8	36.6	38.0	15.6	16.7	15.4	31.4

Environment. In this experiment, the model is given an image and the garment category and is required to predict the positions of all keypoints. We use PyFlex [11] as the physics simulator and Blender [30] for rendering. By synthesizing garment images and keypoint annotations in simulation, we train a model to predict keypoints and then directly evaluate it on real-world images without any fine-tuning on real data. We assume that the mask is known and the background is masked out. For real-world images, we use Grounded-SAM [31] for segmentation. Models trained on synthetic datasets are directly tested on this real-world dataset without fine-tuning.

Asset. For each garment category, we generate 1,500 synthetic garment instances for training using our proposed asset generation pipeline. We manually collected and annotated 480 t-shirts, 82 trousers, 96 vests, and 96 hoodies to construct a real-world test dataset.

Metric. We select **Mean Average Precision (meanAP)** and **Average Keypoint Distance (AKD)** as metrics [9]. We model keypoint detection as a classification problem, where a keypoint is considered correctly classified if the pixel-wise Euclidean distance between the detected keypoint and the ground truth is below a given threshold. The thresholds in our experiments are 4 pixels and 8 pixels, which correspond to approximately 0.5 cm and 1 cm in the real world, respectively. Under this definition, meanAP is the proportion of keypoints that are correctly classified, and the reported meanAP is the average over the two thresholds. AKD is the average pixel-wise distance between predicted and ground truth keypoints. In the ground truth images, we only annotate visible keypoints, and for both metrics, invisible keypoints are ignored during evaluation.

Data generation cost. In our data generation pipeline, creating a mesh template is very fast. Deforming a mesh using PyFlex takes approximately 6 seconds, and generating a texture image with Stable-Diffusion-3.5 requires around 20 seconds (excluding the time waiting for ChatGPT responses). Rendering a 480×720 image takes about 2 seconds. Overall, it takes roughly 30 seconds to generate a single cloth instance. All experiments are conducted on a server equipped with two Intel Xeon Platinum 8255C CPUs (48 cores, 96 threads, 2.50 GHz base frequency) and an NVIDIA RTX 3090 GPU.

Training details. The keypoint coordinates are converted into Gaussian blobs on 2D probability heatmaps, which serve as targets for pixel-wise logistic regression using a binary cross-entropy loss. We directly adopt the model architecture from [9], which is a U-Net [32]-inspired architecture, with

a pretrained MaxViT [33] nano model as the encoder. The network takes a 480×720 masked RGB image as input and outputs N heatmaps of size 256×256 , where N denotes the number of keypoints for the garment category. Keypoints are extracted from the predicted heatmaps by identifying local maxima within a 3×3 pixel window, using a probability threshold of 0.01. During training, we apply data augmentation techniques including color jittering, random rotation and translation, and random patching. The model is trained for 5 hours per category on a single NVIDIA RTX 3090 GPU.

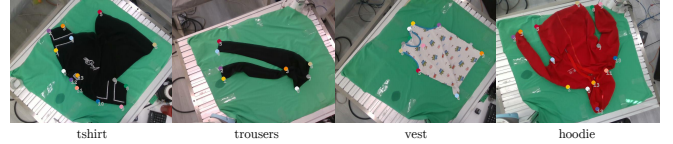


Fig. 6. **Qualitative results of keypoint detection on real images.** The figure shows the predicted output of our keypoint detection model on real images.

2) Experiment Results:

In this experiment, we address the following two questions:

How do the textures generated by our method compare with those produced by other approaches? We study this question by training the same model on datasets generated using different garment texture methods: *Ours*, *aRTF* [9], and *Paint-it* [21].

Can filtering with a VLM improve the appearance quality of the generated meshes? We include results from our pipeline both with and without the final filtering stage.



Fig. 7. **Examples of generated meshes.** Compared with other texture generation methods, our approach produces textures that are generally more plausible.

The experimental results are presented in Table II, and some keypoint detection results on real images are shown in Figure 6. The results indicate that, compared with other approaches, our framework produces garments with more realistic appearances and achieves strong performance on both

the meanAP and AKD metrics. The ablation study regarding the final filtering stage further demonstrates the effectiveness of the VLM-based filtering step. Some qualitative comparisons of the generated meshes are shown in Figure 7.

B. Folding Policy Learning

1) Experiment Setup:

Environment. We use PyFlex [11] as the physics simulator and Blender [30] for rendering. The initial garment state includes random rotation around the z-axis (vertical axis), random flipping, and randomly generated wrinkles. We use the RGB images from the robot’s head-mounted D436 camera as single-view visual input. The complete action space consists of the XYZ coordinates of both grippers, as well as the grippers’ open–close states, resulting in a total of 8 dimensions. We use inverse kinematics (IK) to compute the robot’s joint angles from the end-effector pose. During IK solving, we constrain only the grippers to remain parallel to the table, leaving the other two rotational degrees of freedom unconstrained. The table height is assumed to be known and fixed. Each demonstration trajectory has approximately 120 steps. During testing, a trajectory is considered terminated if it exceeds 300 steps or if the movement distance between consecutive actions is less than 1 mm.



Fig. 8. **Real-world assets for garment folding.** In the real-world experiments, each garment is folded twice, and the average success rate is computed.

Asset. For each garment category, we generate 1,000 training instances using our proposed asset generation pipeline. During testing, an additional set of 100 previously unseen garments is used. For the table and scene backgrounds, we randomly sample a collection of indoor assets downloaded from PolyHaven [34]. For real-world testing, we use 10 unseen garments, as shown in Figure 8.

Metric. To automatically determine folding success in simulation, we first run the keypoint-based policy on a perfectly initialized garment configuration. The resulting garment mesh is referred to as *mesh_gt*. During model evaluation, the final folded mesh, *mesh_eval*, is compared with *mesh_gt*. After aligning the two meshes by an arbitrary rigid-body rotation and translation, we compute the Euclidean distances between all pairs of corresponding vertices and define their average as the evaluation metric. Folding is considered successful if the average vertex distance is below 0.4 mm. In real-world experiments, folding success is determined by human experts.

Data generation cost. Simulation and rendering are computationally intensive for this task, with high demands on both CPU and GPU resources. We use AMD EPYC 7543 CPUs (128 cores in total) and 8 NVIDIA RTX 4090 GPUs to perform

the simulation and rendering. Generating 1,000 trajectories requires approximately one day.

Training Details. We use a CNN-based policy from Diffusion Policy [29]. The observation encoder employs ResNet50, producing observation features with a dimensionality of 512. Simultaneously, the robot’s current state (the XYZ coordinates of the left and right end-effectors and the grippers’ open–close status) is mapped to a 512-dimensional space. The observation and state features are then concatenated to form the conditional input to the diffusion policy. The model uses only the current observation and proprioception as input. Its output is an action sequence of length 16, and during inference, we execute the first four actions. Depending on the dataset size and the specific task, the total number of training steps ranges from approximately 100k to 400k. Training requires approximately one day on 8 NVIDIA RTX 4090 GPUs.

2) Experiment Results:

In this experiment, we address the following five questions:

How does the proposed KG-Dagger improve the quality of training data? We evaluate this by employing different methods for generating demonstrations and comparing the performance of models trained with the same amount of data.

Within our data generation framework, can new folding rules be designed to enable the model to learn alternative garment folding strategies? We devise different folding strategies and evaluate the success rate of each.

What is the trend of success rate with respect to the amount of training data? We compare model performance under varying amounts of training data and training meshes.

Can the model transfer to the real world? We evaluate the success rates of models trained with different data generation methods in real-world scenarios.

Can the VLA model be fine-tuned with FoldNet? We load the pretrained π_0 [3] model, a mainstream large VLA model, and fine-tune it on the FoldNet dataset. We evaluate the model on a robot unseen by the original π_0 model, conducting tests in both simulation and the real world.

a) Different demonstration generation pipelines: In Figure 9(a), we compare the performance of models trained with different demonstration generation pipelines in simulation. *Perfect* refers to using only perfect demonstrations, while *Noised* refers to demonstrations generated by adding noise to ground-truth actions [35]. This baseline also employs the keypoint-based error recovery strategy to augment the dataset but differs from KG-Dagger in that the actions are obtained by perturbing the ground-truth actions before execution, rather than using actions predicted by the network. *KG-Dagger* corresponds to the complete method described in Section IV. The numbers in parentheses indicate the total number of training trajectories used.

When the dataset includes trajectories with error corrections (*Noised*, *KG-Dagger*), there is a significant performance improvement compared to using only perfect demonstrations (*Perfect*). Figure 10 illustrates this difference: the model trained exclusively on perfect demonstrations fails to retry after a grasp failure, whereas the model trained with error recovery data succeeds. Moreover, the *KG-Dagger* method further reduces the gap between the training and testing data

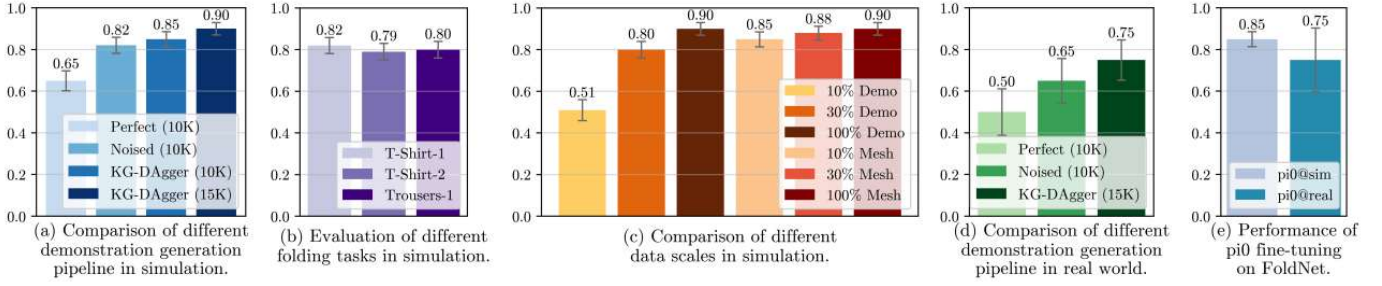


Fig. 9. **Quantitative results of garment folding.** We compare the average success rates of garment folding tasks for different models in simulation and in the real world.

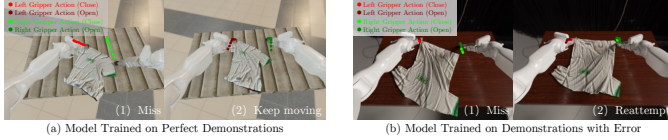


Fig. 10. **Comparison of models trained with different demonstration data generation methods in simulation.** In (a), the training data do not include recovery strategies for errors, so a failed grasp results in an out-of-distribution situation. In (b), the training data include recovery strategies, allowing the model to retry grasping after a failure.

distributions compared to the *Noised* method, leading to better performance.

b) Different tasks: Our data generation pipeline can be adapted to various folding patterns. In Figure 9(b), we use 10K trajectories generated by the *Noised* method as training data. The folding procedures for *T-Shirt-1* and *Trousers-1* are shown in Figure 4(a). The difference between *T-Shirt-2* and *T-Shirt-1* lies in the final step: instead of folding the bottom of the shirt upward, *T-Shirt-2* folds it from left to right. The experimental results show that our framework is not limited to a specific folding method.

c) Data scale: Figure 9(c) illustrates how the model’s performance varies with the amount of training data. Here, 100% usage indicates training with 1000 garments and 15K demonstrations. When varying the number of meshes or demonstrations, the quantity of the other is kept fixed.

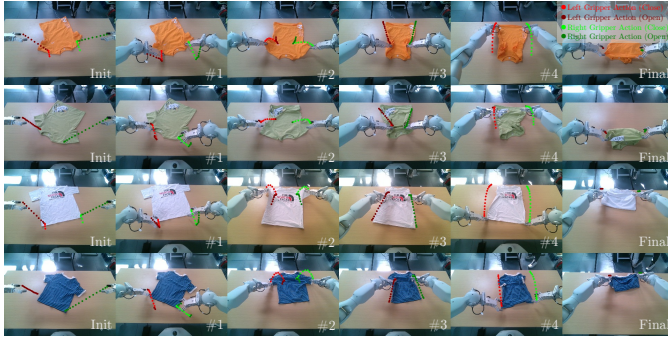


Fig. 11. **Real-world deployment.** The figure illustrates the performance of our policy in real-world scenarios.

d) Sim2real performance: Our trained model can be directly transferred from simulation to the real world. As shown in Figure 9(d), we compare the real-world performance of models trained with different demonstration gen-

eration methods. The model trained with our KG-Dagger approach outperforms those trained with other demonstration data. Representative examples of model outputs in real-world experiments are shown in Figure 11.

e) Fine-tuning VLA with FoldNet: We directly fine-tuned the π_0 model on our dataset, with the language input fixed as “Fold the T-shirt.”. The pre-trained π_0 model we used has 3 billion parameters and employs a flow matching head. The rest of the model’s inputs and outputs are consistent with those of the DP model. We use a batch size of 64 and a learning rate of $2.5e-5$, fine-tuning all parameters for 50,000 steps, which takes approximately 16 hours on 8 H100 GPUs. The experimental results are shown in Figure 9(e). The results demonstrate that even without using any real-world data, we can still train a VLA model capable of generalizing to real-world scenarios.

VI. CONCLUSIONS

In this paper, we present a synthetic dataset for garment folding. At the core of the dataset are garment keypoints, which enable both the synthesis of garment meshes and the generation of demonstration data. To further improve model performance, we incorporate keypoint-based error recovery data into the demonstration dataset. Our experiments show that models trained with this dataset can be directly transferred to real robots and unseen garments.

VII. LIMITATION

Although KG-Dagger improves the model’s ability to recover from failures, certain failure modes remain challenging. Representative examples are shown in Figure 12 and Figure 13. In particular, some unexpected situations in the real world are difficult to accurately reproduce in simulation.

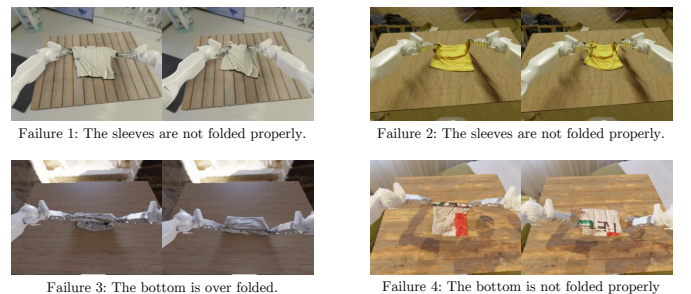


Fig. 12. **Failure mode in simulation.**

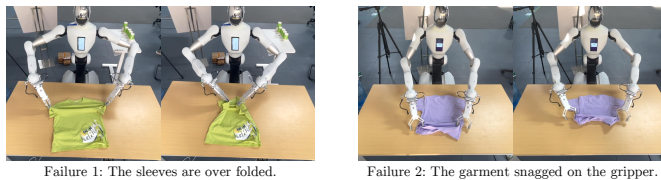


Fig. 13. Failure mode in real world.

Currently, the folding patterns are relatively simple, mainly due to limitations in the physical realism of the cloth simulation. When more complex folding methods are adopted, the realism of the simulation significantly degrades. In the future, the use of finer cloth meshes and more efficient and accurate simulators could further reduce the sim-to-real gap. Exploring the addition of rotational degrees of freedom in the action space is also a promising direction. In addition, combining synthetic and real-world data has the potential to improve success rates in the real world.

REFERENCES

- [1] A. Longhini, Y. Wang, I. Garcia-Camacho, D. Blanco-Mulero, M. Molletta, M. C. Welle, G. Alenyà, H. Yin, Z. Erickson, D. Held, J. Borràs, and D. Kragic, “Unfolding the literature: A review of robotic cloth manipulation,” *Annu. Rev. Control. Robotics Auton. Syst.*, vol. 8, no. 1, pp. 295–322, 2025.
- [2] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, “A survey of imitation learning: Algorithms, recent developments, and challenges,” *IEEE Transactions on Cybernetics*, vol. 54, no. 12, pp. 7173–7186, 2024.
- [3] Physical Intelligence Team, “ π_0 : A vision-language-action flow model for general robot control,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.24164>
- [4] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu, “Rdt-1b: a diffusion foundation model for bimanual manipulation,” *arXiv preprint arXiv:2410.07864*, 2024.
- [5] A. Canberk, C. Chi, H. Ha, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, “Cloth funnels: Canonicalized-alignment for multi-purpose garment manipulation,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5872–5879.
- [6] Y. Wang, Z. Sun, Z. Erickson, and D. Held, “One Policy to Dress Them All: Learning to Dress People with Diverse Poses and Garments,” in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.
- [7] H. Bertiche, M. Madadi, and S. Escalera, “Cloth3d: Clothed 3d humans,” in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 344–359.
- [8] B. Zhou, H. Zhou, T. Liang, Q. Yu, S. Zhao, Y. Zeng, J. Lv, S. Luo, Q. Wang, X. Yu, H. Chen, C. Lu, and L. Shao, “Clothesnet: An information-rich 3d garment model repository with simulated clothes environment,” in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 20 371–20 381.
- [9] T. Lips, V.-L. De Gussemme, and F. Wyffels, “Learning keypoints for robotic cloth manipulation using synthetic data,” *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6528–6535, 2024.
- [10] H. Lu, R. Wu, Y. Li, S. Li, Z. Zhu, C. Ning, Y. Shen, L. Luo, Y. Chen, and H. Dong, “Garmentlab: A unified simulation and benchmark for garment manipulation,” in *Advances in Neural Information Processing Systems*, vol. 37. Curran Associates, Inc., 2024, pp. 11 866–11 903.
- [11] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, “Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids,” in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019*, 2019.
- [12] H. Xue, Y. Li, W. Xu, H. Li, D. Zheng, and C. Lu, “Unifolding: Towards sample-efficient, scalable, and generalizable robotic garment folding,” in *Proceedings of The 7th Conference on Robot Learning*, 2023.
- [13] Y. Avigal, L. Berscheid, T. Asfour, T. Kröger, and K. Goldberg, “Speedfolding: Learning efficient bimanual folding of garments,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 1–8.
- [14] T. Tian, H. Li, B. Ai, X. Yuan, Z. Huang, and H. Su, “Diffusion dynamics models with generative state estimation for cloth manipulation,” *CoRR*, vol. abs/2503.11999, 2025.
- [15] H. Jiang, H.-Y. Hsu, K. Zhang, H.-N. Yu, S. Wang, and Y. Li, “Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos,” 2025.
- [16] H. Chen, J. Li, R. Wu, Y. Liu, Y. Hou, Z. Xu, J. Guo, C. Gao, Z. Wei, S. Xu, J. Huang, and L. Shao, “Metafold: Language-guided multi-category garment folding framework via trajectory generation and foundation model,” in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.
- [17] W. Chen, H. Xue, F. Zhou, Y. Fang, and C. Lu, “Deformpam: Data-efficient learning for long-horizon deformable object manipulation via preference-based action alignment,” *CoRR*, vol. abs/2410.11584, 2024.
- [18] OpenVLA Model Team, “Openvla: An open-source vision-language-action model,” 2024. [Online]. Available: <https://arxiv.org/abs/2406.09246>
- [19] Octo Model Team, “Octo: An open-source generalist robot policy,” in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.
- [20] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, E. VanderBilt, A. Kembhavi, C. Vondrick, G. Gkioxari, K. Ehsani, L. Schmidt, and A. Farhadi, “Objaverse-xl: A universe of 10m+ 3d objects,” *arXiv preprint arXiv:2307.05663*, 2023.
- [21] K. Youwang, T.-H. Oh, and G. Pons-Moll, “Paint-it: Text-to-texture synthesis via deep convolutional texture map optimization and physically-based rendering,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [22] B. Wen, W. Yang, J. Kautz, and S. Birchfield, “Foundationpose: Unified 6d pose estimation and tracking of novel objects,” in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 17 868–17 879.
- [23] J. Hejna, C. A. Bhateja, Y. Jiang, K. Pertsch, and D. Sadigh, “Remix: Optimizing data mixtures for large scale imitation learning,” in *Proceedings of The 8th Conference on Robot Learning*, 2024.
- [24] V. Myers, B. C. Zheng, O. Mees, S. Levine, and K. Fang, “Policy adaptation via language optimization: Decomposing tasks for few-shot imitation,” *CoRR*, vol. abs/2408.16228, 2024.
- [25] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, “Hg-dagger: Interactive imitation learning with human experts,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8077–8083.
- [26] J. Luo, C. Xu, J. Wu, and S. Levine, “Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning,” *Science Robotics*, vol. 10, no. 105, p. eads5033, 2025.
- [27] OpenAI, “Gpt-4o system card,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.21276>
- [28] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, “SDXL: improving latent diffusion models for high-resolution image synthesis,” in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7–11, 2024*, 2024.
- [29] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.
- [30] Blender Foundation, “Blender.” [Online]. Available: <https://www.blender.org>
- [31] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang, “Grounded SAM: assembling open-world models for diverse visual tasks,” *CoRR*, vol. abs/2401.14159, 2024.
- [32] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015, pp. 234–241.
- [33] Z. Tu, H. Talebi, H. Zhang, F. Yang, P. Milanfar, A. Bovik, and Y. Li, “Maxvit: Multi-axis vision transformer,” *ECCV*, 2022.
- [34] PolyHaven Team, “Poly haven - the public 3d asset library.” [Online]. Available: <https://polyhaven.com/>
- [35] J. Lyu, Y. Chen, T. Du, F. Zhu, H. Liu, Y. Wang, and H. Wang, “Scissorbot: Learning generalizable scissor skill for paper cutting via simulation, imitation, and sim2real,” in *Proceedings of The 8th Conference on Robot Learning*, 2024.