PolyFold: A Generalizable Framework for Language-Conditioned Bimanual Cloth Folding

Haozhe Du, Kechun Xu, Rong Xiong, Yue Wang

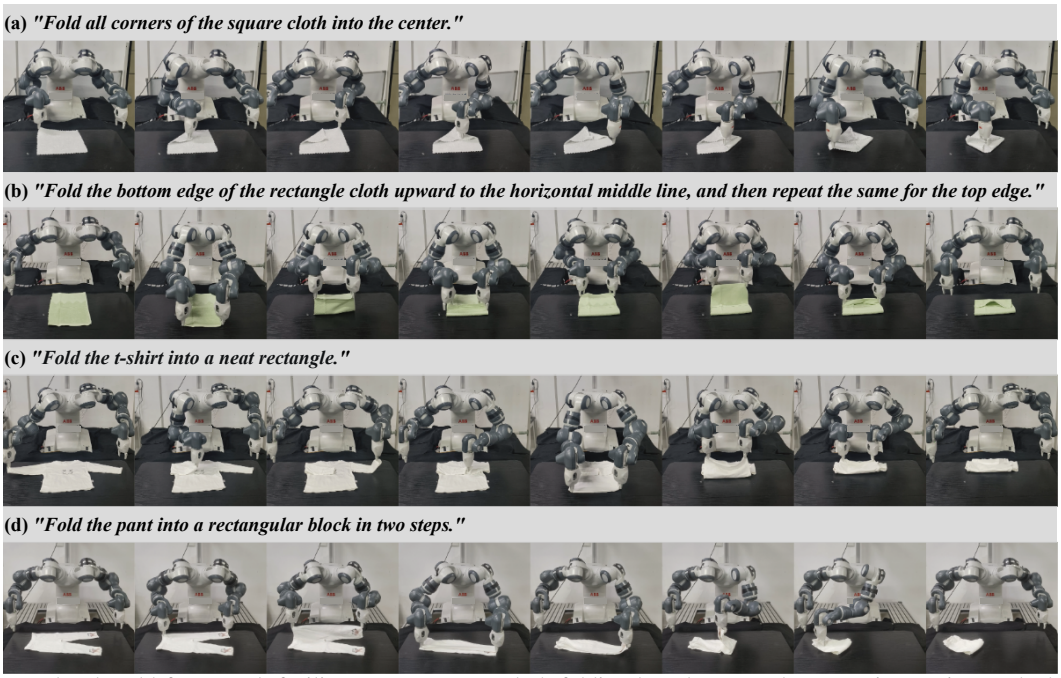


Fig. 1: Our proposed PolyFold framework facilitates autonomous cloth folding based on user language instructions, whether explicit or ambiguous. PolyFold demonstrates zero-shot generalization across various types and shapes of cloth, as well as different cloth folding tasks.

***Abstract*—Cloth folding stands as an intricate subject in robot manipulation, requiring robots to fold diverse fabrics into different configurations according to human intentions. Previous work in this area falls into three primary categories: imitation learning, reinforcement learning, and geometric model-based planning methods. While each paradigm has its merits, they generally lack inherent multi-step reasoning ability and struggle to generalize to novel cloth appearances and tasks. To tackle these problems, our key insight is to incorporate the common sense reasoning and generalization abilities of Large Language Models (LLMs) into cloth manipulation, while addressing the limitations of LLMs in manipulating deformable objects, which requires an effective grounding module and rational planning hierarchy. To this end, we present PolyFold, a novel language-conditioned bimanual cloth folding framework that leverages the parameterized polygon model as an effective abstraction and grounding module for cloth representation. Moreover, PolyFold enables LLMs to infer an intermediate-level action—specifically, the symmetrical fold line—while delegating the pick-and-place calculations to a fold-line-guided downstream policy, which is learned through self-supervision using random data. Experiments on 70 cloth folding tasks and 4 cloth types show that PolyFold excels in zero-shot generalization and inherent multi-step reasoning capability, while also operating in a sample-efficient expert-demonstration-free manner, surpassing previous SOTA vision-conditioned and language-conditioned methods. Our method can also be directly deployed in real-world scenarios. Videos, code and appendix are available at our project webpage: https://sites.google.com/view/polyfold.**

***Note to Practitioners*—This paper is motivated by the need for a deformable object manipulation algorithm, particularly for cloth, with robust reasoning and generalization capabilities to handle diverse unseen objects and tasks. Such an algorithm holds significant promise for service robots in assisting with daily household organization. Existing robotic cloth folding methods often rely on predefined subgoals for action inference and their adaptability to new fabrics and new tasks is quite limited. To address these limitations, we propose PolyFold, a novel framework that integrates Large Language Models (LLMs) into the cloth-folding process while overcoming their challenges in handling deformable objects in two aspects. Firstly, PolyFold introduces a parameterized polygon model to abstract and represent cloth geometry and then it enables LLMs to identify intermediate-level actions—specifically symmetrical fold lines—while leaving detailed pick-and-place execution to a downstream policy trained via self-supervised learning. Experiments in simulation and the real world demonstrate PolyFold's strong generalization to unseen tasks and fabrics, its capacity for multi-step reasoning, and its independence from expert demonstrations.**

***Index Terms*—Deformable object manipulation, Large Language Models, parameterized polygon model, self-supervised learning**

All authors are with the State Key Laboratory of Industrial Control and Technology and the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, P.R. China. Rong Xiong and Yue Wang are the corresponding authors: rxiong@zju.edu.cn and ywang24@zju.edu.cn.

I. Introduction

CLOTH folding, as a representative task in deformable object manipulation, presents a formidable challenge in

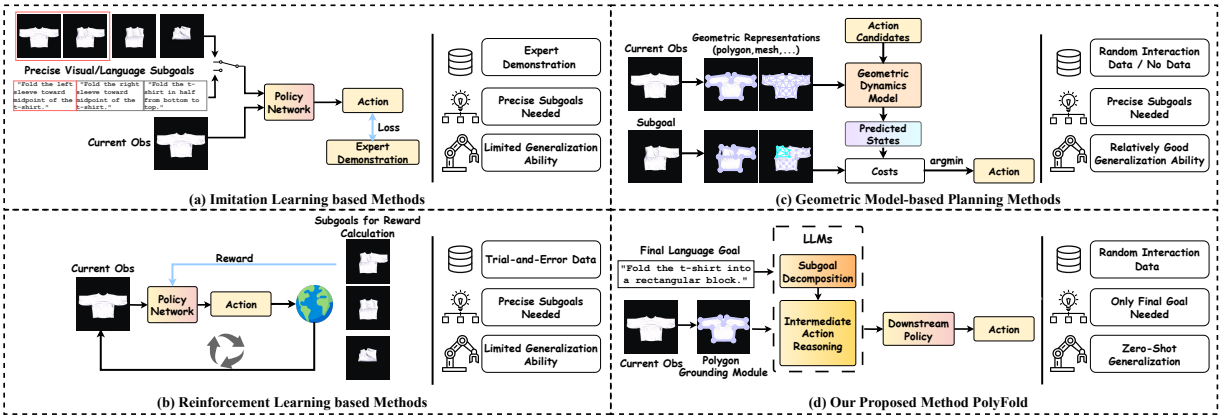


Fig. 2: Comparison of current cloth folding methods: (a) imitation learning based methods, (b) reinforcement learning based methods, (c) geometric model-based planning methods, and (d) our proposed method PolyFold.

robotics. It focuses on the automation of the folding process for various types of fabrics, tailored to meet specific configurations [1], [2]. Classical paradigms of previous works can be seen in Fig. 2. Imitation learning based methods [3]–[8] typically learn from expert demonstration and utilize a sequence of precise subgoal images or language instructions to plan current cloth folding action. These methods call for precise subgoals to support the multi-step pick-and-place actions for cloth folding, which is almost infeasible in applications, especially in open scenarios. The generalization capabilities of these methods are quite limited, making it challenging for them to adapt to previously unseen cloth shapes and unseen cloth folding tasks. Reinforcement learning based approaches [9]–[12] learn a policy from trial-and-error data gathered through environmental interaction, where the reward signal for policy updates is usually generated by comparing the current state to a subgoal. However, these methods exhibit poor generalization to unseen tasks and struggle to learn directly from the final goal, largely due to the highly complex dynamics of cloth. Geometry-based planning approaches [13]–[15] begin by extracting a geometric representation of the cloth, such as meshes [13], [14], or graphs [15]. These methods then employ a dynamics model—either learned from data or predefined by rules—to perform model-based planning. The objective is to select an action that brings the predicted geometric state of the cloth closest to the subgoal. Such methods still rely on single-step subgoals for planning, but they achieve reasonable generalization due to the usage of a relatively versatile dynamics model.

Recent development of Large Language Models (LLMs) and Vision Language Models (VLMs) [16] has endowed robot agents with exceptional common sense reasoning capability and generalization ability to perform long-horizon robot manipulation tasks [17]–[20], in a zero-shot or few-shot way and without huge reliance on expert data. However, they usually focus on rigid object tabletop manipulation, leaving deformable object manipulation relatively unexplored. Recently there have been a few works leveraging LLMs/VLMs for subgoal-guided [21] or keypoint-based [22], [23] cloth folding, as well as cloth unfolding [21], [24] and learning reward models for cloth manipulation [25]. But no method has yet achieved expert-demonstration-free, zero-shot generalizable cloth manipulation based solely on the final language goal, which is the focus of our work.

Here we focus on incorporating LLMs into cloth manipulation tasks to enhance the method's reasoning and generalization capabilities while reducing dependence on carefully collected data. However, the challenges exist in both the input and output of the LLMs: (1) Perceptual grounding methods commonly used in rigid object manipulation such as open-vocabulary detectors [26], [27] and segmentation tools [28], may not adequately address the challenges of object understanding in deformable object manipulation. (2) LLMs struggle to fully comprehend the deformable properties of cloth and the coordination relationship required for dual-arm manipulation, which hinders their ability to determine appropriate bimanual pick-and-place actions directly.

In this paper we propose PolyFold, a language-conditioned bimanual manipulation framework, which aims to perform cloth folding tasks with multi-step reasoning and strong generalization abilities, while eliminating the need for extensive expert demonstrations during training. To tackle the challenge (1) above, we propose to leverage a geometric-based representation. We use parameterized polygon models [29], [30] as structured abstractions for cloth, which extract semantically meaningful keypoints and their connection relationships. Polygon models can effectively capture the internal structure of cloth, and support real-time updating across the folding process. Compared with a detected bounding box or a segmented mask, the polygon model serves as a more effective grounding module since it provides appropriately abstracted information to enable LLMs to understand the state of the clothing. We also enhance it with a semantic- and registration-aware fitting process to ensure robust grounding against ambiguous initial poses and unexpected displacements during execution (detailed in Appendix). For challenge (2), we design a hierarchical planning framework to efficiently predict multi-step dual-arm pick-and-place actions. We first utilize an LLM to decompose the language instruction into subtasks. Given the subtask and the current polygon model abstraction, another LLM is utilized to deduce an intermediate action representation, i.e. the symmetrical fold line. Using the fold line as a condition and guidance, we propose a downstream bimanual pick-and-place policy to accomplish

one step of cloth folding. The policy comprises two parts: a trainable spatial action map for folding result prediction and an optimization module for calculating grasp points with the highest affordance. This policy can be trained using self-supervision derived from an ideal folded polygon model, which also facilitates achieving strong performance by training exclusively with random interaction data in simulation.

We conduct experiments both in simulation and on a real-world ABB YuMi robot, evaluating 70 unseen cloth-folding tasks and four types of unseen cloth. The results demonstrate the strong zero-shot generalization and inherent multi-step reasoning capabilities of our method, which significantly outperforms previous SOTA vision- and language-conditioned approaches. The key contributions of our paper are:

(1) We propose PolyFold, a language-conditioned bimanual cloth folding framework, which performs exceptionally well with zero-shot generalization and inherent multi-step reasoning abilities, and without relying on expert demonstrations.

(2) We introduce the parameterized polygon model as an effective abstraction and grounding module for cloth representation, enabling LLMs to comprehend and reason about cloth states.

(3) We design a hierarchical planning framework that leverages LLMs for subgoal decomposition and intermediate action representation inference, culminating in a downstream bimanual pick-and-place policy.

(4) Extensive experiments in simulation and in the real world demonstrate that PolyFold can zero-shot generalize to different unseen cloth shapes and manipulation tasks, surpassing previous SOTA cloth folding algorithms by a significant margin.

II. Related Works

A. Cloth Manipulation

Early works on cloth manipulation [29]–[31] focus on designing geometric representations like polygon models or meshes, and then executing scripted cloth folding sequences. Recent works [13]–[15] integrate geometric representations with dynamics models to plan optimal actions. The majority of these works focus on cloth smoothing and flattening [14], [15]. For instance, VCD [14] first learns to build a mesh representation of the cloth from partial point cloud inputs and employs a graph neural network to learn the cloth's dynamics from this mesh, enabling model-based planning for the cloth flattening operation. In the context of cloth folding, Wang et al. [13] introduce a method that simplifies the complex cloth mesh according to the current observation and a target state. This simplified mesh is used as the action space for a planner, which leverages the model from the SoftGym [32] simulator to complete tasks like folding a square cloth diagonally. Despite their ability to learn without expert data or solely from random interactions, these approaches still rely on precise subgoals to plan effectively. Moreover, they exhibit limited generalization to complex garments like t-shirts and pants, as well as to challenging multi-step manipulation scenarios.

Imitation learning based methods [3]–[5], [8], [33], [34] accomplish cloth manipulation by learning from expert demonstrations and utilize provided accurate subgoals to plan robot actions. A representative work is Foldsformer [3], which employs Vision Transformer [35] to encode the current observation and the subgoal image, and then fuses their information in the feature space using a spatio-temporal attention mechanism [36] to plan the next action. However, this subgoal-conditioned pattern reveals that these methods lack multi-step reasoning abilities and can only infer single-step actions based on the nearest subgoal. Moreover, expert data collection is time-consuming and labor-intensive; the generalization ability to novel tasks, unseen cloth shapes, and real-world scenarios remains considerably limited.

Reinforcement learning (RL) based methods have also been applied to cloth manipulation. Lee et al. [37] propose Learning-to-Fold, which utilizes the deep Q-learning algorithm [38] to learn the cloth folding task. Taking image observation and a target state as input, it employs a convolutional network to predict the Q-values for potential grasping points, thereby generating manipulation actions. QDP [11] employs Sequential Reinforcement Learning [39] to learn Q-value maps for pick-and-place actions in cloth manipulation, along with parameters associated with the action trajectory. Furthermore, due to the complex deformable properties and high-dimensional action space inherent in cloth manipulation, some works [9], [10], [33] take advantage of expert demonstrations for robot policy pretraining to enhance sample efficiency. Reinforcement learning based methods still require precise subgoals to learn manipulation actions effectively. Furthermore, the generalization capability of these approaches is highly limited. Existing work focuses on learning specific tasks, such as diagonal cloth folding, and fails to generalize effectively to unseen manipulation tasks.

B. Spatial Action Maps

Spatial action maps [40] are widely employed in cloth manipulation [41]–[43], which efficiently recover per-pixel action values for robotic manipulation. Flingbot [41] employs a single spatial action map to learn bimanual fling actions for cloth unfolding, predicting the center position of dual-arm grasp points and using predefined rules to compute the complete fling action. However, in intricate and fine-grained bimanual cloth folding tasks, we need to consider more action variables while ensuring that actions of the dual arms can meet the requirements of affordance and cooperate well. Moreover, identifying suitable self-supervision signals for folding tasks remains a challenge. To address these issues, we propose a method that combines spatial action map learning with affordance score optimization to obtain bimanual cloth folding actions. What's more, we utilize the ideal folded polygon model contour as a guiding signal for efficient self-supervision.

C. Language-conditioned Robot Manipulation

Language instructions offer flexible and accessible means of providing task-related information, which have led recent research efforts [7], [44], [45] to concentrate on grounding language in robot manipulation tasks. In the domain of deformable object manipulation, Deng et al. [8] propose a

language-conditioned approach that integrates language instructions, images, and graphs, learning from expert demonstrations to accomplish cloth folding.

The advent of Large Language Models (LLMs) and Vision Language Models (VLMs) [16] has highlighted their robust common sense reasoning ability. Recent works [17]–[20], [46] leverage LLMs for planning feasible actions in various rigid object manipulation tasks. For deformable object manipulation, GPT-Fabric [21] utilizes VLMs to perform single-arm cloth folding tasks, but with reliance on visual subgoals. RL-VLM-F [25] employs a Vision Foundation Model to provide efficient feedback for reinforcement learning in cloth folding. Similar to our motivation, CLASP [22] also focuses on developing suitable representations for cloth manipulation and uses LLMs for task planning. CLASP trains a masked autoencoder (MAE) to learn a keypoint-based representation for cloth and utilizes LLMs for high-level cloth folding action planning. In contrast, our PolyFold employs a parameterized polygon model that maintains graph structure with multi-layered nodes and edges, offering richer geometric and semantic information than pure keypoints without requiring additional training. This grounding module enables a deeper understanding of deformable fabric. Furthermore, unlike CLASP's heuristic-based low-level pick-and-place planning, our method introduces a more flexible hierarchical planning framework. In our planning framework, LLMs are responsible only for inferring subgoals and intermediate action representation—fold lines—while self-supervised learning based downstream policy completes the computation of the pick-and-place actions, which is more generalizable and efficient.

III. Preliminary: Parameterized Polygon Model

The parameterized polygon model proposed by Miller et al. [29] is an effective graph-based method for cloth representation. It defines the polygon shape through a set of parameters and optimizes the shape to make it best aligned with visual observation, which also allows continuous updates to the structure during cloth folding task execution. It comprises two parts: skeletal parameters $\mathcal{X}$ and polygon generator $\mathcal{P}$. Skeletal parameters are the minimal set of parameters necessary for polygon representation, which are composed of K two-dimensional skeletal or interior points $\mathcal{X}^P \in \mathbb{R}^{2\times K}$ and M one-dimensional scalar parameters $\mathcal{X}^S \in \mathbb{R}^M$. Therefore $\mathcal{X} = \left[vec(\mathcal{X}^P), \mathcal{X}^S\right]^T$ and the dimension is $2K + M$. The polygon generator $\mathcal{P}$ takes the skeletal parameters as input and produces the polygon model (vertices, edges), denoted as $\mathcal{P}(\mathcal{X}) = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of polygon vertices and adjacent vertices are connected to form the edges $\mathcal{E}$ of the polygon model.

1) Initial parameterized polygon model: Fig. 3(a) shows initial parameterized polygon models of square/rectangle cloth, t-shirt, and pant, which represent flattened cloth without folds. A set of skeletal parameters $\mathcal{X}_0$ of the initial polygon model is drawn in red color. The polygon generator $\mathcal{P}_0(\mathcal{X}_0)$ provides the vertices $\mathcal{V}_0$ drawn in blue color and edges $\mathcal{E}_0$ drawn in gray color.

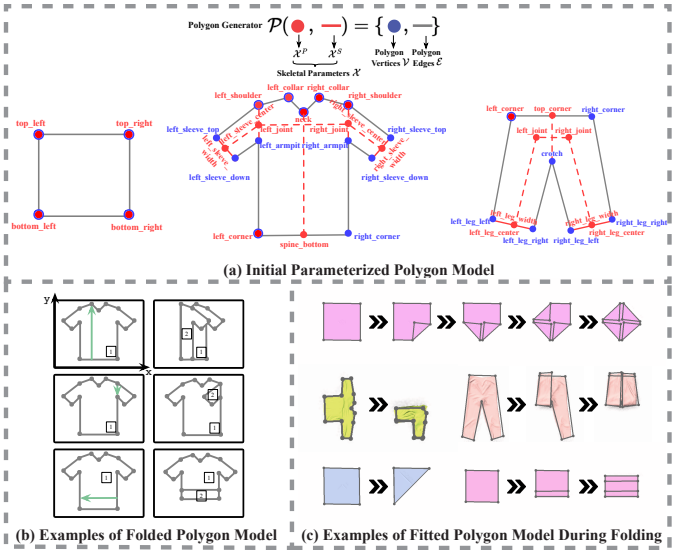


Fig. 3: Parameterized polygon model. (a) Initial parameterized polygon models of square/rectangle, t-shirt, and pant [29]. (b) Folded polygon model. (c) Fitted polygon model during the cloth folding process based on RGB image observation.

2) Folded polygon model: In the polygon model, every folding action is represented by a directional symmetrical fold line vector $\mathbf{F_t} = \overrightarrow{f_{t1}f_{t2}}$ for folding step t, where f_{t1} is the start point of the fold line vector and f_{t2} is the end point. As shown in Fig. 3(b), the polygon generator $\mathcal{P}_t$ operates as follows: the section of the polygon situated on the left side of the fold line is lifted vertically, and placed in the mirrored position of its original position with the fold line as the axis of symmetry. For the folded model, the parameters of fold line $\mathbf{F_t}$ are added to the skeletal parameters: $\mathcal{X}_{t+1} = [\mathcal{X}_t, vec([f_{t1}, f_{t2}])]^T$. The polygon generator also contains layer information of different segments, with the layer numbers increasing for upper segments [29]. For instance, in the second row of Fig. 3(b), the layer number of the folded right sleeve is greater than the rest of the garment because it lies on top after folding.

3) Polygon model fitting: From polygon generator $\mathcal{P}_t(\mathcal{X}_t) = \{\mathcal{V}_t, \mathcal{E}_t\}$ we can extract the contour $\mathcal{C}_t$ of the polygon, as a function of $\mathcal{X}_t$. For folding step t, we optimize skeletal parameters $\mathcal{X}_t$ to make the polygon model contour $\mathcal{C}_t(\mathcal{X}_t)$ best aligned with the actual contour of the cloth $\tilde{\mathcal{C}}_t$ extracted from current RGB image. This alignment is achieved through energy optimization, as described by [29]. The energy function is defined as the Chamfer Distance between sampled points of $\mathcal{C}_t$ and $\tilde{\mathcal{C}}_t$.

$$
\begin{aligned}
E(\mathcal{X}_t) &= \frac{1}{2}\mathcal{D}(\Lambda(\mathcal{C}_t(\mathcal{X}_t)), \Lambda(\tilde{\mathcal{C}}_t)) + \frac{1}{2}\mathcal{D}(\Lambda(\tilde{\mathcal{C}}_t), \Lambda(\mathcal{C}_t(\mathcal{X}_t))) \\
&= \frac{1}{2}\mathcal{D}(\mathcal{C}_t^s, \tilde{\mathcal{C}}_t^s) + \frac{1}{2}\mathcal{D}(\tilde{\mathcal{C}}_t^s, \mathcal{C}_t^s) \\
\mathcal{D}(X, Y) &= \frac{1}{|X|}\sum_{x\in X}\min_{y\in Y}\|x - y\|
\end{aligned}
$$

where Λ represents the sampling process that selects a fixed number of points $\mathcal{C}_t^s$ and $\tilde{\mathcal{C}}_t^s$ uniformly along the polygon model contour $\mathcal{C}_t$ or the actual cloth contour $\tilde{\mathcal{C}}_t$ respectively.

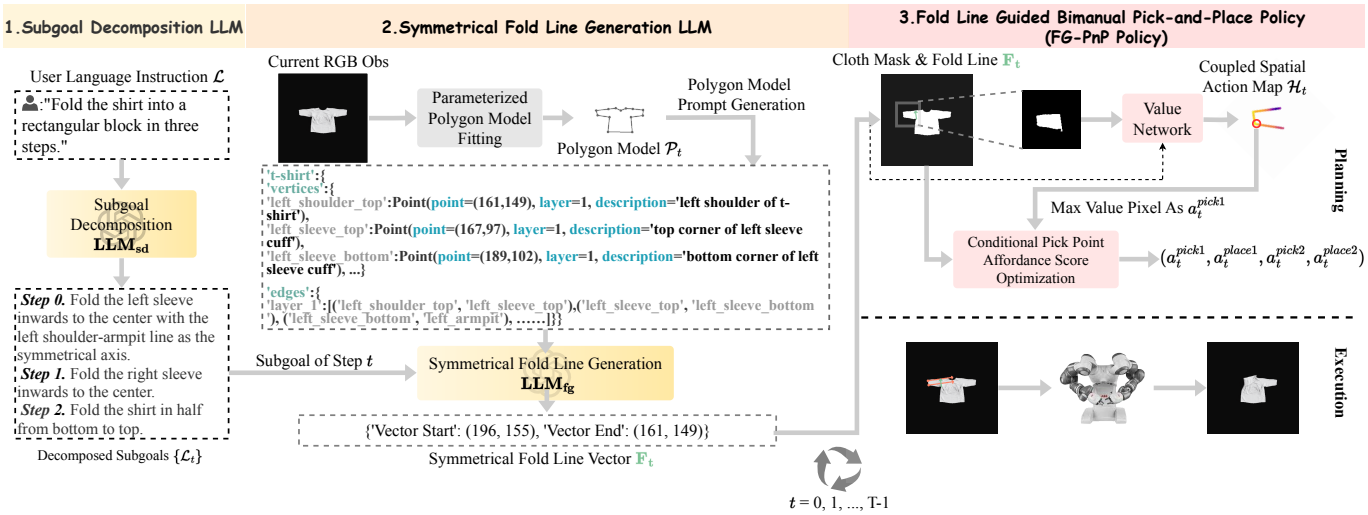


Fig. 4: PolyFold architecture overview. (1) Given the user's final goal instruction $\mathcal{L}$, we first decompose it into subgoals using Large Language Model $\mathbf{LLM_{sd}}$. (2) For each subgoal step from $t = 0$ to $t = T - 1$, current RGB observation is used to perform parameterized polygon model fitting. The resulting polygon model $\mathcal{P}_t$ and current subgoal $\mathcal{L}_t$ are fed into the symmetrical fold line generation LLM $\mathbf{LLM_{fg}}$. (3) Using the inferred symmetrical fold line $\mathbf{F_t}$ as guidance, we learn a policy (FG-PnP policy) to generate a bimanual pick-and-place action. After robot execution, the current observation is updated, and the algorithm proceeds to the next subgoal.

During the energy optimization, soft constraints are introduced to define a permissible parameter range that maintains specific cloth shapes. The structural penalty function $\Gamma(\mathcal{X}_t)$ imposes penalties on parameters that violate these soft constraints and otherwise yields zero. For more details of these, refer to [29] and our Appendix.

Hence, the final energy function to be minimized is expressed as:

$$\min_{\mathcal{X}_t} E_{\text{final}}(\mathcal{X}_t) = \frac{E(\mathcal{X}_t)}{E_{\max}} + \Gamma(\mathcal{X}_t)$$

where $E_{\max}$ serves as a normalization constant. Using black-box optimization to optimize $\min_{\mathcal{X}_t} E_{\text{final}}(\mathcal{X}_t)$, the optimal set of skeletal parameters $\mathcal{X}_t$ is obtained, and the polygon model can be generated using $\mathcal{P}_t(\mathcal{X}_t)$. Fig. 3(c) shows different fitted polygon models during the cloth folding procedure.

IV. Methodology

A. System Overview

We consider that the cloth is initially laid flattened on a horizontal surface. Given a user language instruction $\mathcal{L}$, our method PolyFold learns to generate a sequence of bimanual pick-and-place actions $\{a_t = (a_t^{pick1}, a_t^{place1}, a_t^{pick2}, a_t^{place2})\}$ $(t = 0, 1, ..., T-1)$, where each t represents a complete folding step, in order to fulfill the requirements of the language instruction $\mathcal{L}$. To elaborate further, as depicted in Fig. 4, PolyFold can be broken down into three main components: First, the language instruction $\mathcal{L}$ is fed into the subgoal decomposition LLM module, which decomposes it into a set of divided language subgoals $\{\mathcal{L}_t\}$. Subsequently, the process iterates from $t = 0$ to $t = T-1$. At each folding step t, PolyFold conducts parameterized polygon model fitting to optimize parameters $\mathcal{X}_t$ for aligning the polygon model contour $\mathcal{C}_t(\mathcal{X}_t)$ with actual cloth contour $\tilde{\mathcal{C}}_t$ and obtain current polygon model $\mathcal{P}_t(\mathcal{X}_t)$. Then the symmetrical fold line generation LLM takes the current polygon representation $\mathcal{P}_t$ and the language subgoal $\mathcal{L}_t$ as input and outputs the required symmetrical fold line $\mathbf{F_t}$ to achieve this subgoal. Given the fold line, the downstream fold line guided bimanual pick-and-place policy (FG-PnP policy) generates bimanual actions a_t through a spatial action map and conditional pick point affordance score optimization. After execution, the parameters of fold line $\mathbf{F_t}$ are integrated into the parameterized polygon model parameters $\mathcal{X}_{t+1}$. This process iterates until the final folding step $T-1$ is reached.

B. Subgoal Decomposition LLM

PolyFold utilizes dual-layered Large Language Models to facilitate the analysis of cloth folding tasks from high-level task comprehension to low-level action planning. The upper layer, denoted as the subgoal decomposition LLM ($\mathbf{LLM_{sd}}$), processes the user language instruction $\mathcal{L}$ and also the initial polygon model template $\mathcal{P}_0$ as input, yielding a sequence of decomposed language subgoals $\{\mathcal{L}_t\}$, each of which is intended to be executed in one step of bimanual pick-and-place action. The subgoal decomposition LLM ($\mathbf{LLM_{sd}}$) is designed to produce language subgoals in a structured format, as follows: *fold* ⟨*which part of the cloth*⟩ ⟨*direction, optional*⟩ *to* ⟨*which part of the cloth*⟩ ⟨*symmetrical axis, optional*⟩. Therefore it is denoted as:

$$\{\mathcal{L}_t\} = \mathbf{LLM_{sd}}(\mathcal{L}, \mathcal{P}_0) \qquad (t = 0, 1, 2, ..., T-1)$$

C. Symmetrical Fold Line Generation LLM

1) Parameterized polygon model fitting and grounding: For each step t, PolyFold starts with utilizing black-box optimization to derive an optimal set of skeletal parameters $\mathcal{X}_t$ that aligns current polygon model contour $\mathcal{C}_t(\mathcal{X}_t)$ with current observed cloth contour $\tilde{\mathcal{C}}_t$. The obtained polygon model $\mathcal{P}_t(\mathcal{X}_t)$ is utilized as an effective representation for current cloth state. To help the LLM better understand the polygon model structure, it

is then transformed into a structured text-based representation, specifically in the form of a Python dictionary, as shown in Fig. 4. This dictionary comprises two distinct entries for *vertex* and *edge*. The *vertex* dictionary uses the *vertex_name* as keys, with corresponding values being instances of the *Point* class that include coordinates, a detailed description of the vertex, and the layer information. The *edge* dictionary maintains the connectivity relationships for each layer of the polygon model. The polygonal model representation aids the Large Language Model (LLM) in comprehending the current cloth structure, thereby facilitating the LLM's ability to reason about actions that can fulfill the language subgoal at current time step.

2) Symmetrical fold line as an intuitive and efficient intermediate action representation: Given the challenges in LLM's ability to comprehend the deformable nature and complex shapes of various fabrics, as well as the complexities of dual-arm coordination, allowing the LLM to directly deduce bimanual pick-and-place actions may result in undesirable cloth folding actions, as demonstrated in ablation experiments in Table IV. As illustrated in Section III, the fold line is employed in the parameterized polygon model to represent folding action via a symmetry relationship, where one segment of the polygon split by the fold line is folded onto another. This symmetry-based representation is easier for LLMs to reason about than precise bimanual pick-and-place actions. Moreover, in the planning of bimanual pick-and-place action a_t, once the fold line is determined, the number of variables that need to be calculated can be reduced from 8 ((x, y) *pixels of* $a_t^{pick1}, a_t^{place1}, a_t^{pick2}, a_t^{place2}$) to 4 ($(x, y)$ *pixels of* a_t^{pick1}, a_t^{pick2}) as the place points are symmetrical to the pick points relative to the fold line. The fold line also constrains the search space for pick points to the segment of the image located on one side of the fold line due to its definition, further enhancing action planning efficiency. Therefore we adopt the symmetrical fold line as an efficient intermediate action representation for LLMs to reason about.

3) Symmetrical fold line generation LLM: Given current subgoal language instruction $\mathcal{L}_t$ and current polygon model $\mathcal{P}_t(\mathcal{X}_t)$, the fold line vector $\mathbf{F_t}$ is generated using the symmetrical fold line generation LLM ($\mathbf{LLM_{fg}}$). This process employs chain-of-thought reasoning [47] with a few user-provided examples, instructing the LLM to determine, based on the characteristics and complexity of the current task and current polygon model, whether to directly infer the fold line from the geometric shape or to first reason about a suitable unimanual pick-and-place action and subsequently infer the fold line as the perpendicular bisector of the line segment connecting the pick and place pixels. This process can be represented as follows:

$$\mathbf{F_t} = \mathbf{LLM_{fg}}(\mathcal{L}_t, \mathcal{P}_t(\mathcal{X}_t))$$

D. Fold Line Guided Bimanual Pick-and-Place Policy (FG-PnP Policy)

Given the symmetrical fold line $\mathbf{F_t}$ generated by $\mathbf{LLM_{fg}}$, we propose a fold line guided bimanual pick-and-place policy (FG-PnP policy). Due to the complex dynamics of deformable objects, we follow a previous method [40] to learn a spatial action map for action value prediction. Specifically here we leverage the spatial action map for folding result prediction to determine suitable pick-and-place points. However, bimanual cloth folding poses additional complexities. Naively learning the actions of two grippers using two individual spatial action maps would lead to poor learning outcomes because it ignores garment manipulation affordance and the collaborative relationship between the two arms. Another approach is to acquire dual-arm actions in a carefully designed cascaded way. Previously Flingbot [41] first learns a spatial action map for the center position of dual-arm grasp positions and then uses predefined rules to calculate the entire fling action. Inspired by this, we adopt a more advanced method: a spatial action map predicts the folding result, from which a pick position a_t^{pick1} is obtained. Subsequently, we use conditional affordance score optimization to acquire the second gripper's pick point a_t^{pick2} conditioned on a_t^{pick1} and the fold line $\mathbf{F_t}$. This ensures the selection of bimanual operational points with the highest affordance and optimal folding outcomes. It is worth noting that for a piece of cloth, we assume that all meaningful pick points for folding lie on the garment's contour. Therefore, our FG-PnP policy is designed to exclusively search for and select pick points from the pixels corresponding to the cloth's detected contour, significantly constraining the action space and improving efficiency.

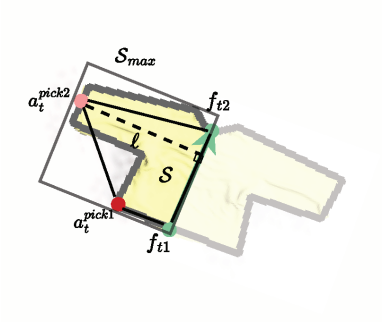


Fig. 5: Illustration of the objective function for conditional pick point affordance score optimization, which contains the quadrilateral area $\mathcal{S}(f_{t1}, f_{t2}, a_t^{pick1}, a_t^{pick2})$ and the perpendicular distance $\ell(a_t^{pick2}, \mathbf{F_t})$.

1) Conditional pick point affordance score optimization: Here we first introduce how to determine the pick position of the second gripper a_t^{pick2}, conditioned on the given fold line $\mathbf{F_t} = \overrightarrow{f_{t1}f_{t2}}$, the first gripper's pick position a_t^{pick1}, and current contour of the cloth $\tilde{\mathcal{C}}_t$. In light of the affordance in deformable cloth manipulation, it is crucial to prioritize the stability of bimanual folding actions to minimize internal deformation in the folded cloth. We utilize two simple yet effective key metrics to assess this: the quadrilateral area $\mathcal{S}$ and the perpendicular distance ℓ from the grasp point to the fold line. The quadrilateral area $\mathcal{S}(f_{t1}, f_{t2}, a_t^{pick1}, a_t^{pick2})$ denotes the area of the quadrilateral formed by the starting point f_{t1} and ending point f_{t2} of the fold line, and also two pick points a_t^{pick1}, a_t^{pick2}. The perpendicular distance $\ell(a_t^{pick2}, \mathbf{F_t})$ represents the distance between the second pick point and its projection onto the fold line. Maximizing the area and

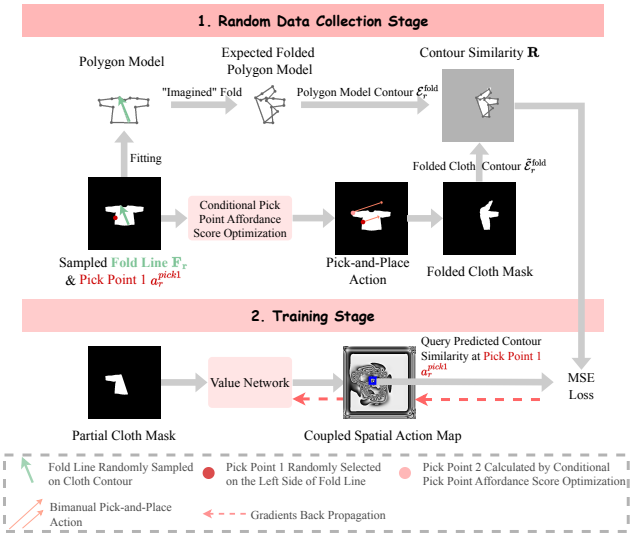


Fig. 6: Training fold line guided pick-and-place policy (FG-PnP policy). (1) Random data collection stage: In each collection step, fold line and robot actions are randomly sampled on the cloth contour and we record the contour similarity between the expected folded model and the resulting folded cloth, which serves as the ground truth similarity $\mathbf{R}$. (2) Training stage: With the partial cloth mask as input, the output spatial action map is supervised by ground truth similarity through MSELoss.

perpendicular distance can optimize the operational space of dual-arm manipulation, minimizing deformation of the fabric during manipulation and achieving the folding outcome that aligns with what the fold line desires. The conditional pick point affordance score optimization process can be detailed as follows and also shown in Fig. 5:

$$\max_{a_t^{pick2}} \frac{\mathcal{S}(f_{t1}, f_{t2}, a_t^{pick1}, a_t^{pick2})}{\mathcal{S}_{\max}} + \gamma \cdot \frac{\ell(a_t^{pick2}, \mathbf{F_t})}{\sqrt{\mathcal{S}_{\max}}}$$
$$\text{s.t.} \quad a_t^{pick2} \in \tilde{\mathcal{C}}_t \quad \text{and} \quad \overrightarrow{f_{t1}a_t^{pick2}} \times \overrightarrow{f_{t1}f_{t2}} > 0$$

where $\mathcal{S}_{max}$ is the area of the bounding box encompassing the partial cloth mask $\tilde{o}_t$, which is utilized to normalize $\mathcal{S}$ and ℓ. Due to the definition of the fold line in Section III, a_t^{pick2} is only selected on the cloth contour located on the left side of the symmetrical fold line $\mathbf{F_t}$. Similar to polygon fitting in Section III, the process is also achieved through black-box optimization.

2) Coupled spatial action map: As shown in Fig. 4, given current top-down RGB observation, we employ a value network to learn a spatial action map. Each pixel in this map represents the predicted folding outcome of a bimanual action, derived from the current pixel and conditional pick point affordance score optimization. Since a single action map predicts the outcome of a coupled bimanual action, where one arm's action depends on the other's, we term this map the *coupled spatial action map*. According to the definition of the fold line in Section III (2), only the part on the left side of the fold line is grasped and folded over to the right. To minimize the influence of irrelevant information, we crop the segmented mask, retaining only the portion on the left of the fold line, denoted as $\tilde{o}_t$. Using this partial mask as input enhances the generalization capacity of our value network. The value network is a 2D conditional UNet [48]. With the positional encoding [49] of the start point f_{t1} and end point f_{t2} as condition, the value network takes the partial mask $\tilde{o}_t$ as input and outputs a spatial action map $\mathcal{H}_t$. The first pick point a_t^{pick1} is then selected by identifying the pixel on the partial cloth contour $\tilde{o}_t$ that has the highest value in the action map. This is formulated as:

$$a_t^{pick1} = \underset{p \in \text{contour}(\tilde{o}_t)}{\text{argmax}} \ \mathcal{H}_t(p)$$

where contour($\tilde{o}_t$) denotes the set of pixel coordinates that constitute the contour of the partial cloth mask.

3) Learning coupled spatial action map with random interaction data and self-supervision: In this learning process, we aim to completely eliminate expert demonstrations and human annotations, relying solely on randomly collected data and self-supervised signals to learn a coupled spatial action map with strong predictive capabilities. As shown in Fig. 6, training data is collected through random interactions in simulation. In each collection step, a symmetrical fold line $\mathbf{F_r} = \overrightarrow{f_{r1}f_{r2}}$ is first arbitrarily chosen along the cloth's contour $\tilde{\mathcal{C}}_r$, and one pick point a_r^{pick1} is randomly selected on the left side of $\mathbf{F_r}$ on $\tilde{\mathcal{C}}_r$. Then the second pick point a_r^{pick2} is obtained by conditional pick point affordance score optimization. Following the execution of the pick-and-place actions, the resulting folded cloth contour is denoted as $\tilde{\mathcal{C}}_r^{\text{fold}}$. Here we reuse the polygon model as an approximate ground truth to evaluate the folding result by contour similarity $\mathbf{R}$, which compares the resulting folded cloth contour $\tilde{\mathcal{C}}_r^{\text{fold}}$ with the expected folded polygon model contour $\mathcal{C}_r^{\text{fold}}$:

$$\mathbf{R} = \beta - \alpha \cdot [\frac{1}{2}\mathcal{D}(\Lambda(\tilde{\mathcal{C}}_r^{\text{fold}}), \Lambda(\mathcal{C}_r^{\text{fold}})) + \frac{1}{2}\mathcal{D}(\Lambda(\mathcal{C}_r^{\text{fold}}), \Lambda(\tilde{\mathcal{C}}_r^{\text{fold}}))]$$

where α and β are constants for scaling the contour similarity to a relatively reasonable range. A higher $\mathbf{R}$ reflects that the folding result is better. As shown in Fig. 6 we train the value network to predict the contour similarity $\mathbf{R}$ as the folding outcome, utilizing MSE Loss with ground truth contour similarity as supervision, or more precisely, self-supervision, because the fitting and folding process of the polygon model do not require manual labeling and can be fully autonomously completed. Specifically, the MSE loss is computed only over the pixels that lie on the cloth contour. While the convolutional nature of our value network produces a dense 2D spatial action map, supervision is applied exclusively to the contour, as this is the only region of interest for selecting pick points.

4) Place point calculation and single-arm action switching: Given a_t^{pick1}, a_t^{pick2}, the place points a_t^{place1}, a_t^{place2} are calculated as the mirrored points with respect to the fold line $\mathbf{F_t}$. If the two pick points are too close to each other, the bimanual action is switched to single-arm action: $a_t^{pick} = \frac{a_t^{pick1}+a_t^{pick2}}{2}, a_t^{place} = \frac{a_t^{place1}+a_t^{place2}}{2}$.

5) Implementation details: The training data contains 10000 random folding actions and it is only collected on 100 t-shirts. Results in Section V prove that it can generalize to different cloth types. The dimension of the RGB image is

TABLE I: Evaluated cloth folding tasks.

Cloth Type	Task Type	single-step task	multi-step task
Square	S-Corner-Folding	4	7
	S-Triangle-Folding	4	8
Rectangle	R-Edge-to-Middle-Folding	4	4
	R-Edge-to-Opposite-Folding	4	8
T-shirt	T-Sleeve-Folding	4	8
	T-Half-Folding	3	0
	T-Block-Folding	0	2
Pant	P-Half-Folding	4	2
	P-Block-Folding	0	4
Total		**27**	**43**

400×400. Hyperparameters are selected as: $\alpha = 1500$, $\beta = 1$, $\gamma = 0.04$. The network is trained on a machine with 2×RTX 2080 Ti GPUs, which takes 39 hours.

V. Experimental Results

A. Simulation Experiment Setup

All simulation experiments are conducted in the SoftGym simulator [32]. In this simulation environment, the bimanual manipulators are represented as simplified spherical grippers. Consequently, inter-manipulator collisions are generally not a concern, with the primary exception of scenarios where target pick points are in very close proximity. Our FG-PnP policy explicitly handles such cases by reducing the bimanual action to a unimanual one, as detailed in Section IV-D.

In simulation, we use cloth meshes from CLOTH3D [50], with unsuitable or unrealistic objects excluded. The evaluation includes 100 square cloths, 100 rectangular cloths, 100 t-shirts, and 100 pants. Importantly, the t-shirts used for evaluation differ from those in Section IV-D for policy training, ensuring that all evaluation cloths are unseen for our method, presenting significant challenges for generalization.

We define 9 types of folding tasks for 4 cloth types, named in the format *cloth_type-folding_type*, resulting in a total of 70 evaluated tasks, each comprising multiple sub-tasks. For square cloth, tasks include *S-Corner-Folding* and *S-Triangle-Folding*. For rectangle cloth, we define *R-Edge-to-Middle-Folding* and *R-Edge-to-Opposite-Folding*. T-shirt tasks include *T-Sleeve-Folding*, *T-Half-Folding*, and *T-Block-Folding*, while pant tasks consist of *P-Half-Folding* and *P-Block-Folding*. Details of the generated tasks are presented in TABLE I. Tasks are categorized as single- or multi-step and are evaluated using 5 variations of language instructions expressing the same meaning differently. Further text and image descriptions are provided in the Appendix.

B. Baselines

We compare the performance of our proposed PolyFold against different types of baseline methods, with implementation details provided in the Appendix:

1) Imitation learning based methods:

- *Foldsformer* [3]: Current state-of-the-art vision-conditioned cloth folding method. It uses a sequence of visual subgoals to determine pick-and-place actions. For comparison, we also implement a dual-arm variant of Foldsformer, referred to as *Foldsformer-2Arm*.
- *Language-Deformable* [8] (abbreviated as *LangDef*): Current state-of-the-art language-conditioned cloth manipulation approach. It takes in different user-provided language instructions for different folding sub-steps. We also implement a dual-arm version of LangDef for comparison, annotated as *LangDef-2Arm*.

2) Reinforcement learning based methods:

- *Learning-To-Fold* [37] (abbreviated as *LTF*): Deep Q-learning (DQN) [38] based method for cloth folding tasks. It utilizes a fully convolutional neural network and learns from a sparse reward function to create a goal-conditioned pick-and-place policy.
- *Proximal Policy Optimization (PPO)* [51]: We implement a PPO-based reinforcement learning method that takes the RGBD image and down-sampled particle states of the cloth as input and leverages a dense distance-based reward for cloth folding task learning.

3) Geometric model-based planning methods:

- *Wang et al.* [13]: A simplified mesh based method for model-based planning. It uses a reduced action space based on the simplified mesh for CEM-based [52] efficient planning.
- *VCD* [14]: A graph neural network based method that learns a mesh-based cloth dynamics model from random interaction data.

C. Evaluation metrics

Following previous works [3], [8], we use the *Mean Particle Position Error (MPPE)* metric to measure the similarity between the executed fold and the ground truth from the oracle demonstrator, as cloth is modeled as particles in the simulation. Additionally, the *Mean Intersection over Union (mIoU)* metric is employed to evaluate the overlap between the folded cloth mask and the ideal mask from the oracle. Finally, the *success rate* is calculated based on certain thresholds of MPPE and mIoU metrics, which are explained in detail in the Appendix.

D. Simulation Experiment Results

1) Generalization ability: Generalization involves both object-level and task-level evaluation. For object-level generalization, we assess the performance of our method and baselines on clothing shapes unseen during training. For task-level generalization, we evaluate performance on entirely new tasks. Sub-tasks within each task type in TABLE I are evenly divided into seen and unseen categories, with further details provided in the Appendix. Two experiments are designed to evaluate generalization ability: *unseen cloth + seen task* and *unseen cloth + unseen task*, with the latter posing greater difficulty.

For imitation learning based baselines, expert demonstrations of only the seen tasks are used for training, while reinforcement learning baselines train their policies in the simulation restricted to these same tasks. As PolyFold and two geometric model-based baselines Wang et al. [13] and VCD [14] do not use task-specific data for training, all tasks are considered unseen. Moreover, all evaluated cloth objects

TABLE II: Simulation evaluation of our proposed PolyFold and baselines. Each method is tested on four types of cloth (square, rectangle, t-shirt, and pant) and evaluated using three metrics MPPE (mm), mIoU, success rate (%).

Method	Square			Rectangle			T-shirt			Pant		
	MPPE ↓	mIoU ↑	success% ↑	MPPE ↓	mIoU ↑	success% ↑	MPPE ↓	mIoU ↑	success% ↑	MPPE ↓	mIoU ↑	success% ↑
Unseen Cloth + Seen Task †												
Foldsformer [3]	12.59	0.915	95.3	72.67	0.555	35.1	29.60	0.839	65.5	59.43	0.693	43.6
Foldsformer-2Arm	**10.03**	**0.923**	**98.0**	**15.28**	**0.911**	**98.3**	**26.11**	0.835	**71.3**	**51.60**	**0.812**	**68.7**
LangDef [8]	12.81	0.913	94.9	71.03	0.563	34.7	28.42	**0.845**	67.8	59.59	0.700	46.3
LangDef-2Arm	18.21	0.886	91.1	19.72	0.869	90.2	52.95	0.717	32.0	75.52	0.730	36.1
LTF [37]	19.25	0.873	89.5	54.47	0.719	49.7	40.41	0.772	54.0	79.88	0.694	33.6
PPO [51]	19.58	0.882	90.9	67.31	0.612	36.4	43.69	0.745	47.5	89.87	0.663	27.2
Unseen Cloth + Unseen Task												
Foldsformer [3]	92.39	0.596	25.3	150.77	0.420	0.0	54.95	0.735	54.2	236.82	0.340	14.6
Foldsformer-2Arm	118.23	0.486	3.7	113.29	0.523	6.7	50.35	0.718	40.0	111.79	0.612	15.6
LangDef [8]	125.34	0.515	13.5	147.15	0.425	0.0	52.59	0.750	58.2	235.02	0.334	17.3
LangDef-2Arm	148.25	0.447	5.5	163.90	0.375	0.8	89.77	0.584	14.7	276.52	0.231	4.0
LTF [37]	50.01	0.767	56.5	74.06	0.656	26.2	55.61	0.648	20.4	330.12	0.193	3.2
PPO [51]	52.80	0.717	44.1	87.63	0.604	10.2	61.63	0.635	18.9	230.51	0.309	4.6
Wang et al. [13]	21.04	0.858	84.6	30.79	0.820	73.4	55.35	0.655	22.7	196.86	0.449	9.2
VCD [14]	48.92	0.735	50.2	79.15	0.651	22.6	56.34	0.651	22.2	99.26	0.619	20.4
PolyFold (Ours)	**19.58**	**0.872**	**94.0**	**23.30**	**0.862**	**90.8**	**19.74**	**0.894**	**88.3**	**49.66**	**0.830**	**79.6**

† In this unseen cloth + seen task setting, only imitation learning and reinforcement learning based baselines are evaluated to compare their generalization ability to unseen cloth, as well as a cross-comparison with the performances under the unseen cloth + unseen task setting. Since all evaluated cloth and tasks are unseen for our method PolyFold, as well as Wang et al. [13] and VCD [14], they do not appear in this setting and only appear in the unseen cloth + unseen task setting.

are also unseen. Therefore PolyFold, Wang et al. [13] and VCD [14] are evaluated exclusively in the *unseen cloth + unseen task* scenario, while other baselines are tested in both. To mitigate the complexity for all baselines, we provide accurate visual or language subgoals based on their original settings. In contrast, PolyFold relies solely on the language description of the final goal, making its evaluation more challenging.

From TABLE II, it is evident that in the *unseen cloth + unseen task* evaluation setting, PolyFold significantly outperforms all baselines across nine types of cloth folding tasks in terms of mean particle position error, mIoU, and success rate metrics. Although the method proposed by Wang et al. [13] performs relatively well on manipulation tasks involving simple objects like squares and rectangles, it heavily relies on a simplified mesh for planning. For complex tasks with intricate garments such as T-shirts and pants, this mesh simplification approach often fails, leading to planning failures. VCD [14] exhibits relatively poor performance because it is trained exclusively on data from random interactions in simulation. Such random interactions are often insufficient to capture the complex dynamics of multi-step folding tasks, which causes the model to struggle. Imitation learning and reinforcement learning based baselines also perform poorly in this challenging scenario, highlighting their inability to generalize when faced with clothing shapes and folding tasks not seen in training. Visualization of representative task evaluations in Fig. 7 further illustrates that under the *unseen cloth + unseen task* setting, PolyFold produces folding results that align closely with the given language instructions. In contrast, most baselines either replicate folds from their training data or infer only partial actions, failing to generalize effectively despite being provided with accurate per-step subgoals. Moreover, as our method is trained solely on t-shirt data, with no exposure to square, rectangle, or pant cloth instances during training, PolyFold demonstrates exceptional zero-shot generalization capability.

In the simpler *unseen cloth + seen task* setting, baselines perform exceptionally well on square folding tasks due to the standardized nature of square shapes, which only vary in size. However, their performance declines as the complexity of the clothing shapes increases in rectangle, t-shirt, and pant folding tasks. Among the six baselines, Foldsformer-2Arm achieves the best overall performance. Notably, our proposed PolyFold in the more challenging *unseen cloth + unseen task* setting surpasses the baselines' performance on t-shirt and pant folding tasks, even when the baselines operate in the simpler *unseen cloth + seen task* scenario. This highlights PolyFold's superior generalization capability.

2) Multi-step reasoning ability: In this experiment, each baseline is provided only with the final goal, either as a goal image or a language instruction, to verify whether the methods can infer the entire sequence of actions based solely on the final goal, rather than a series of precise subgoals. All methods are evaluated on the 21 multi-step tasks that are seen in the training process for imitation learning and reinforcement learning based methods. The results in TABLE III show that when provided only with a final goal, success rates of baseline methods are very low, indicating that baselines lack sufficient inherent multi-step reasoning abilities when given only the final goal. This limits the application scenarios of these methods, as precise subgoals for each step must be provided to complete a successful cloth folding task. In contrast, our method demonstrates strong multi-step reasoning and planning capability, with the aid of Large Language Models and our carefully designed hierarchical architecture.

3) Sample efficiency and scalability with random interaction data: Our method, VCD [14] and imitation learning based baselines Foldsformer(-2Arm) [3] and LangDef(-2Arm) [8] rely on supervised-learning-based policies; however, our method and VCD are trained solely with random interaction data, while imitation learning based baselines are trained with expert demonstration data. Here we evaluate how the performance changes while the amount of training data changes. We train the FG-PnP policy of PolyFold with 2500, 5000, 10000 sets of training data, where 10000 is the full dataset size as described in Section IV-D. VCD is trained on four types of cloth (square, rectangle, t-shirt and pant). We collect the training data by applying random folding actions

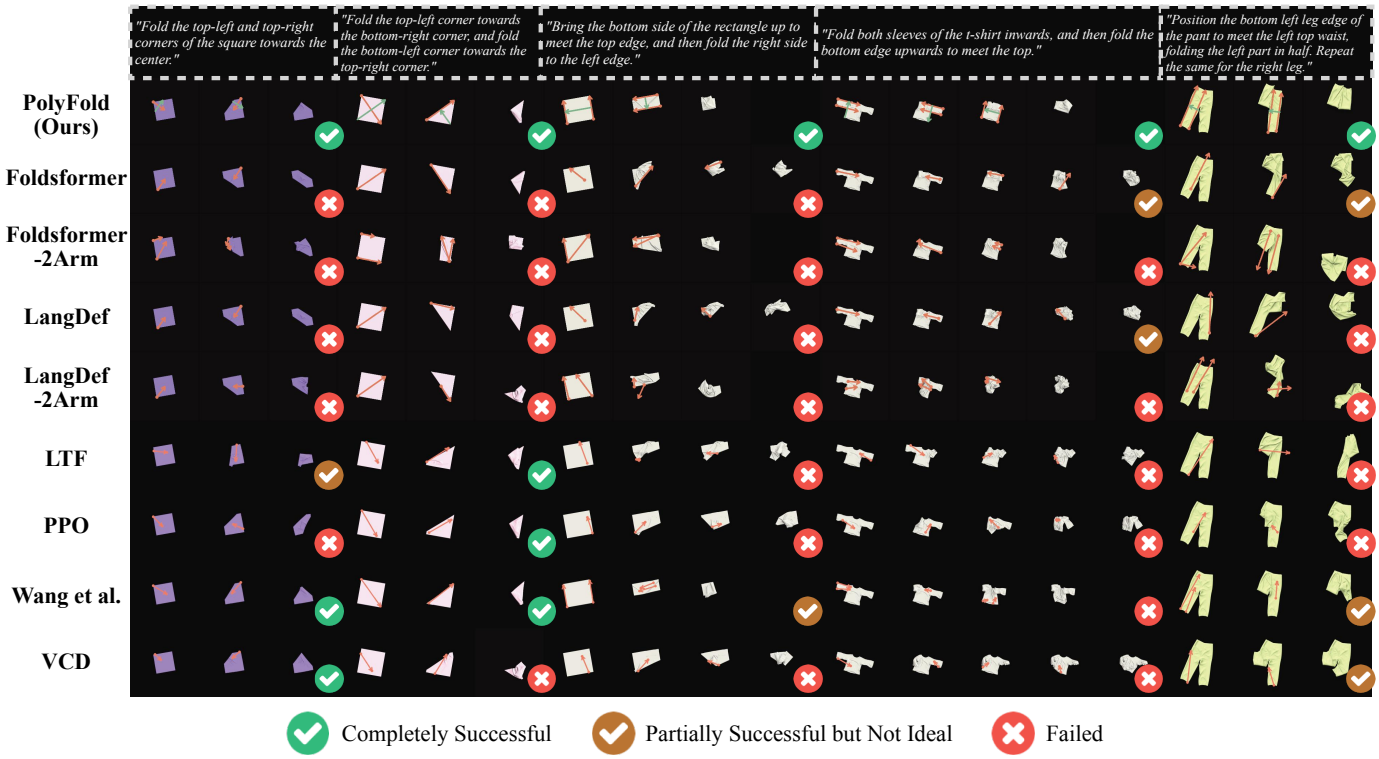


Fig. 7: Visualization of task execution in the SoftGym simulation comparing our method, PolyFold, with eight baseline models under the *unseen cloth + unseen task* setting. The orange arrow indicates the pick-and-place action for one robot arm, while the green arrow represents the symmetrical fold line deduced by PolyFold's LLM. For brevity, only PolyFold's language instructions are displayed.

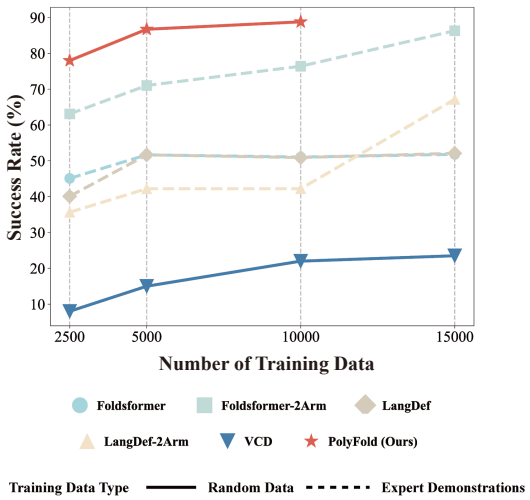


Fig. 8: Variation in method success rate with varying training data sizes and different training data types. All methods are evaluated on the challenging multi-step tasks.

and recording the cloth mesh state before and after each action. For each cloth type, we construct datasets of varying sizes—2,500, 5,000, 10,000, and 15,000 data pairs—where 15,000 constitutes the full-sized dataset. We train four imitation learning based baselines with 2500, 5000, 10000, 15000 sets of expert demonstration data, where 15000 is the full dataset size.

From Fig. 8 we find that as the amount of training data increases, the success rate of all methods becomes higher. Although both our method and VCD are trained on random data, VCD consistently achieves low success rates. In contrast, our method incorporates two key improvements to learn effectively. First, we use the fold line to enforce symmetry between the pick and place actions, which enhances both the effectiveness and efficiency of action sampling. Second, instead of learning the dynamics of the entire mesh, our method learns a spatial action map for manipulation actions under the supervision of a polygon model. These factors combine to reduce the learning difficulty, enabling our model to extract meaningful information from random data. What's more, our method trained with only 2500 random interaction samples already surpasses most of the results from the baselines trained with expert demonstrations. This demonstrates the high efficiency of our approach, which is due to the utilization of LLMs, allowing the policy to focus exclusively on downstream planning tasks, specifically the calculation of pick-and-place points, by leveraging fold lines as useful prior knowledge rather than requiring the model to understand the entire task from scratch.

4) Spatial action map and training process of value network: To provide deeper insight into what our FG-PnP policy learns, we visualize the coupled spatial action maps for four representative tasks. Fig. 9 illustrates the full pipeline from the input partial mask to the final folding result. The extracted Spatial Action Map column displays the predicted value distribution along the contour of the input mask. Warmer colors on this map correspond to higher predicted values

TABLE III: Experiment results of multi-step reasoning. Each method is provided with only the final goal instead of precise subgoals.

Method	MPPE ↓	mIoU ↑	success% ↑
Foldsformer [3]	120.52	0.526	18.5
Foldsformer-2Arm	63.38	0.702	50.3
LangDef [8]	118.46	0.536	19.5
LangDef-2Arm	148.17	0.420	9.2
LTF [37]	152.51	0.436	11.1
PPO [51]	130.71	0.508	13.1
Wang et al. [13]	83.70	0.609	28.5
VCD [14]	100.83	0.565	20.8
PolyFold (Ours)	**29.65**	**0.847**	**88.4**

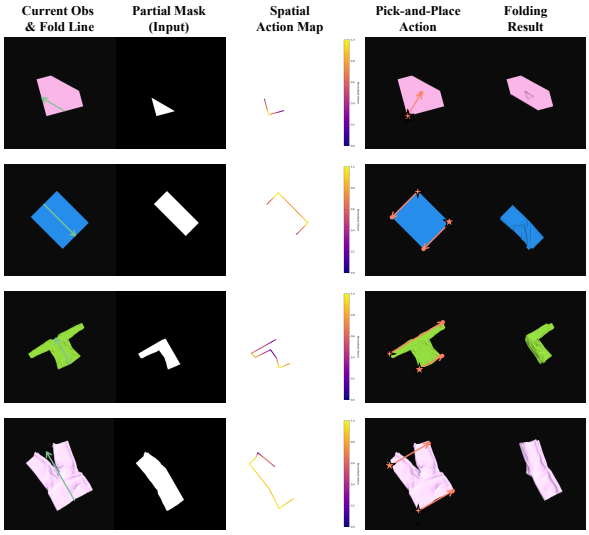


Fig. 9: Visualization of spatial action maps. First, after getting the fold line (green line of the first column), the mask of the region to be folded is segmented by it. The value network outputs a dense spatial action map which is constrained to the cloth contour. Finally, the point with the highest value on this contour is selected (denoted by a five-pointed star icon), and this point is used to generate another pick action (denoted by a four-pointed star icon) through conditional affordance score optimization process. Place points are calculated with respect to the fold line.

for a potential first pick point a_t^{pick1}. These visualizations reveal that the value network, despite being trained solely on random data, learns meaningful and intuitive values for bimanual folding: For the task of folding a single corner of a square, the action map correctly assigns the highest value to the corner vertex. The values decrease nearly symmetrically along the two adjacent edges. To fold the rectangle in half, the model correctly identifies that picking near the two corners of the folding edge provides the most stable grasp. The resulting value map is appropriately symmetric. For the t-shirt task, the policy identifies high-value regions near the bottom corner of the hem and along the sleeve. Importantly, it correctly assigns low values to the region near the armpit, which is an intuitively poor choice for a pick point in this scenario, aligning well with human common sense. When folding the pant in half, the model learns that the optimal pick points are located at the upper and lower ends of the outer leg seam. Conversely, it assigns very low values to the inner seam, correctly identifying it as an undesirable region to grasp for this type of fold.

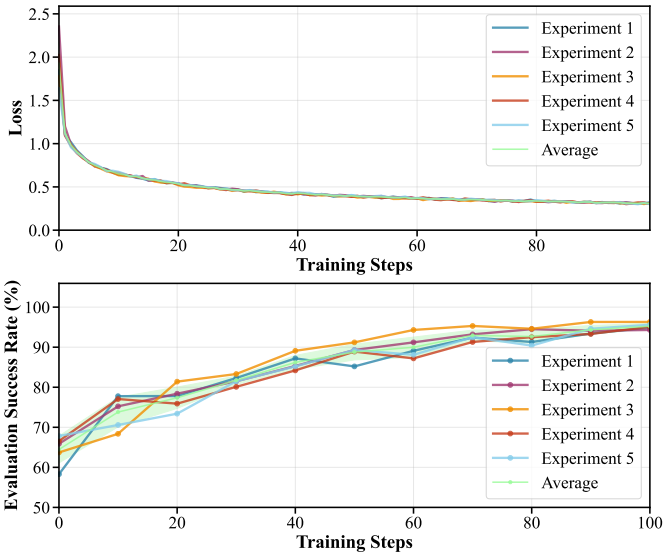
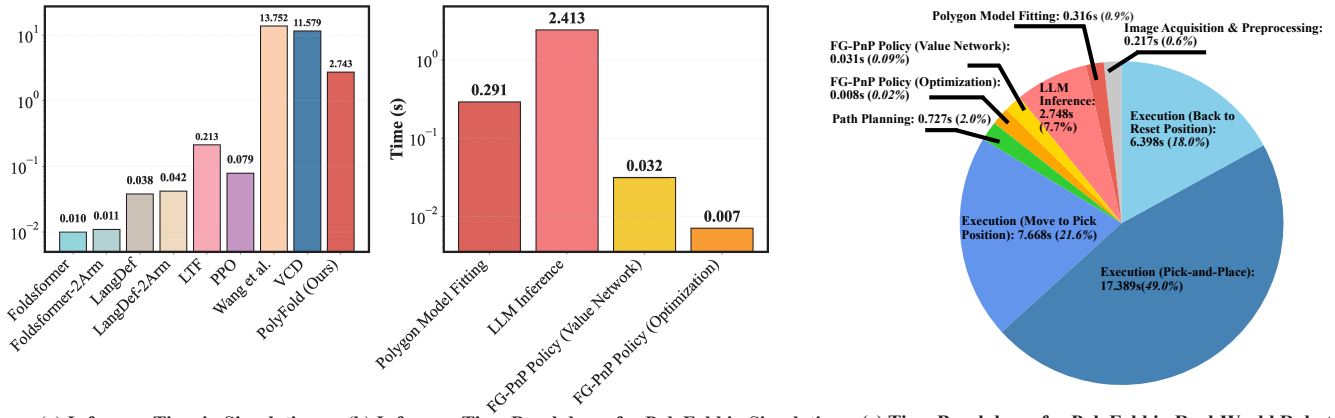


Fig. 10: Value network training loss and evaluation success rate for the FG-PnP Policy across five runs.

In summary, these visualizations demonstrate that our value network successfully learns a generalizable understanding of folding affordances. The learned spatial action maps are not arbitrary; they exhibit logical properties such as symmetry and align strongly with human intuition regarding optimal grasping points for a diverse set of garment shapes and folding tasks. This confirms the effectiveness of our self-supervised learning approach.

We also conduct an experiment to validate the training convergence and stability of our model. To do this, we train the value network in FG-PnP Policy 5 times on the same dataset, each with a different random seed. The training process lasts for 100 epochs and we evaluate the performance of FG-PnP Policy on all 70 tasks in TABLE I every 10 epochs. To ensure a fair comparison and eliminate performance variations from the LLM, we used the ground-truth fold line as direct input. The resulting loss curves and evaluation success rate curves are shown in Fig. 10. As shown in the bottom plot, all five runs exhibit a consistent upward trend, with the success rate steadily increasing and beginning to plateau after approximately 80 training steps, indicating robust convergence towards a high-performance policy. Similarly, the top plot shows the training loss, which consistently decreases and flattens out across all runs, confirming the stable convergence of the value network. Crucially, the green narrow shaded area in both plots, which represents the standard deviation across the five runs, highlights the low variance of the training process. These results demonstrate that our training methodology is highly stable, robust to different random initializations, and reliably produces a policy with consistent performance. This ensures that the quantitative results reported in our main experimental sections are representative and reproducible.

5) Time analysis: We compare the inference time of our proposed PolyFold and eight baselines in Fig. 11(a). From these results, we can observe that our method is slower than the end-to-end approaches based on imitation learning and



(a) Inference Time in Simulation **(b) Inference Time Breakdown for PolyFold in Simulation** **(c) Time Breakdown for PolyFold in Real-World Robot**

Fig. 11: Time analysis for our method PolyFold and baselines. (a) Inference time comparison for PolyFold and eight baselines. The comparison is focused solely on inference time, spanning from the input of an observation to the output of an action by the algorithm. (b) Inference time breakdown for PolyFold in simulation. (c) Time breakdown for PolyFold in real-world robot for one pick-and-place action. For better visualization, (a) and (b) use a log scale. In (c), extremely small proportions are magnified only for better visibility. All time calculations are conducted on the same machine with an Intel i5-12600KF CPU and an NVIDIA RTX 3070Ti GPU.

TABLE IV: Ablation studies on the key components of PolyFold: the grounding module, system architecture, and downstream policy.

Method	Square			Rectangle			T-shirt			Pant		
	MPPE ↓	mIoU ↑	success% ↑	MPPE ↓	mIoU ↑	success% ↑	MPPE ↓	mIoU ↑	success% ↑	MPPE ↓	mIoU ↑	success% ↑
Ablation on Grounding Module												
Keypoints	14.11	0.895	93.4	23.67	0.871	89.6	98.99	0.581	36.5	77.17	0.730	57.0
Image	29.96	0.822	78.8	70.20	0.705	55.9	51.90	0.742	48.2	175.57	0.503	18.5
Keypoints + Image	9.55	**0.925**	**99.2**	28.61	0.854	89.9	38.06	0.810	62.4	83.86	0.716	60.4
Polygon Model + Image	**9.29**	0.919	95.5	**21.23**	**0.883**	**94.9**	23.62	0.862	83.8	**45.98**	**0.821**	**84.7**
Polygon Model	16.27	0.892	95.3	22.60	0.872	90.0	**22.78**	**0.883**	**86.6**	50.04	0.820	79.0
Ablation on Architectures vs. End-to-End												
Zero-Shot VLM	30.57	0.809	76.4	55.35	0.723	47.2	56.10	0.697	28.7	192.79	0.470	17.6
Direct-1Arm	10.48	**0.909**	**97.6**	38.36	0.822	59.6	32.34	0.831	71.8	109.83	0.662	24.6
Direct-2Arm	**10.38**	0.909	97.2	48.35	0.815	62.8	25.37	0.852	76.9	81.65	0.720	46.0
Hierarchical	16.27	0.892	95.3	**22.60**	**0.872**	**90.0**	**22.78**	**0.883**	**86.6**	**50.04**	**0.820**	**79.0**
Ablation on FG-PnP Policy												
2sam	16.02	0.891	92.3	46.45	0.785	55.2	18.96	0.893	87.4	67.25	0.763	51.4
1sam	26.08	0.844	86.4	68.57	0.699	21.3	29.31	0.845	76.6	87.36	0.704	40.7
polygon vertex (random)	7.24	0.933	99.4	17.48	0.897	96.4	30.15	0.830	72.6	46.23	0.824	83.1
polygon vertex (learned)	**7.10**	**0.939**	**99.6**	**14.75**	**0.920**	**98.7**	23.78	0.873	84.3	37.54	0.847	90.1
csam+caso	10.47	0.918	98.6	16.05	0.913	98.0	**16.26**	**0.906**	**93.3**	**31.25**	**0.871**	**92.8**

reinforcement learning. In contrast, geometric model-based planning methods Wang et al. [13] and VCD [14] are much slower than our approach, owing to their requirement for extensive sampling. From Fig. 11(b) we can find that the runtime of our method is primarily consumed by LLM API calls, which represent its main bottleneck. In the real-world execution process, we can perform reasoning in parallel with the reset of robot arms, making full use of time and reducing the noticeable delay.

E. Ablation Study

We conduct a series of comprehensive ablation studies to validate the key design choices of our PolyFold framework and to answer the following critical questions: (1) The Grounding Module: What is the most effective representation for grounding LLMs in cloth manipulation tasks? (2) The System Architecture: Is our proposed hierarchical architecture superior to more direct, end-to-end alternatives? (3) The Downstream Policy: What is the optimal design for the downstream FG-PnP policy? (4) The Choice of LLM: How does the performance of our framework vary with the choice of different underlying large language models?

To address (1), we compare PolyFold with four ablation methods representing different grounding approaches in fabric manipulation, denoted as: *Keypoints*, *Image*, *Keypoints + Image*, and *Polygon Model + Image*. *Keypoints* utilizes the method proposed in ClothFunnels [42] to detect semantic keypoints in the outermost contour of the cloth, but this detection model is only useful in the flattened stage. Therefore during folding, we utilize contour detection to segment cloth contour points as anonymous keypoints (which lack exact semantic meaning and topological structure of the cloth) for LLMs. *Image* uses raw RGB image of cloth as the input, with the Large Language Models upgraded to Vision Language Models (VLMs) to process visual input. *Keypoints + Image* combines both detected keypoints and current image as inputs, while *Polygon Model + Image* uses parameterized polygon model and the image as joint multi-modal inputs. Table IV shows that *Image* directly using raw images performs poorly in almost all tasks. In simpler and easily understandable task settings like square and rectangle folding, performance differences among methods other than *Image* are minimal. However, for complex shapes like t-shirts and pants, most tasks of *Keypoints* fail at a rate higher than 50% due to its

lack of semantic and structural information during multi-step folding. While combining keypoints and images in VLMs improves performance, a significant gap remains compared to our polygon model. Furthermore, the result of *Polygon Model + Image* shows a slight improvement compared to pure *Polygon Model* due to image assistance; however, introducing images in VLMs causes considerable additional token/money costs. Therefore, considering the trade-off, we opt for the polygon model alone, as the performance of parameterized polygon model alone is already good enough.

To provide a comprehensive justification for our architectural choices over end-to-end alternatives (question 2), we conduct a systematic study presented in the "Ablation on Architectures vs. End-to-End" section of TABLE IV. This analysis dissects the end-to-end paradigm by investigating its performance from two perspectives: a pure pixel-to-action approach and a more direct planning approach on a structured representation. First, we evaluate a *Zero-Shot VLM* baseline, which represents the purest form of an end-to-end method. It uses a powerful Vision-Language Model (gpt-4o) to generate bimanual pick-and-place coordinates directly from the raw RGB image and the high-level language instruction, bypassing our entire PolyFold framework. The results show that while it achieves a moderate success rate on simple squares, its performance collapses on more complex garments. This confirms that current VLMs lack the fine-grained spatial understanding required for precise manipulation directly from pixels, strongly validating the necessity of a structured intermediate representation for successful grounding. Next, we investigate the *Direct-1Arm* and *Direct-2Arm* variants. These baselines test a quasi-end-to-end planning approach. They still utilize our structured polygon model as input but replace the intermediate fold line generation step. Instead, the LLM is prompted to directly infer the unimanual (*Direct-1Arm*) or bimanual (*Direct-2Arm*) pick-and-place coordinates. While *Direct-2Arm* performs exceptionally well on the simple square folding task, its performance significantly deteriorates on more complex garments compared to our full method. This demonstrates that even when provided with a perfect world representation, LLMs' ability to directly generate precise, low-level action coordinates is still limited, highlighting the benefits of decoupling high-level planning from low-level policy execution. In contrast, our full *Hierarchical* approach demonstrates consistently high performance across all garment types. Collectively, these experiments provide a strong, data-driven justification for our architectural design. They systematically demonstrate that both a structured grounding module and a hierarchical planning process are critical for achieving robust and generalizable performance, outperforming various end-to-end alternatives in complex, multi-step cloth manipulation.

For question (3), we conduct a comprehensive ablation study to validate the key design choices within our downstream FG-PnP policy. This analysis investigates two fundamental aspects: (i) the effectiveness of our hybrid architecture, which combines a coupled spatial action map (csam) with conditional pick-point affordance score optimization (caso), against pure learning-based alternatives; and (ii) the necessity of learning a dense affordance map over the entire contour versus simpler, polygon vertex-based selection heuristics. To address the first aspect, we compare our full *csam+caso* with two pure-learning variants: learning two spatial action maps (*2sam*) for bimanual actions and learning one spatial action map (*1sam*) for single-arm actions. To address the second aspect, we introduce two additional baselines that operate exclusively on the discrete set of polygon vertices. The first, *polygon vertex (random)*, serves as a non-learning heuristic that randomly selects a valid vertex as the first pick point. The second, *polygon vertex (learned)*, implemented through training the value network to predict values only for the vertices and selecting the one with the maximum value, in a similar self-supervised learning way like ours. To isolate the impact of incorrect fold lines inferred by the LLMs, we use fold lines annotated by the oracle demonstrator. By comparing our full policy against these four distinct variants, we aim to demonstrate that our approach of learning a dense affordance map over the continuous contour, synergized with our caso module, provides a more robust, general, and effective solution for complex bimanual cloth manipulation. The results, presented in TABLE IV, offer several key insights into the policy design. Firstly, the underperformance of both *1sam* and *2sam* confirms the challenges of pure-learning approaches and highlights the benefits of our hybrid csam+caso architecture. The *polygon vertex (random)* baseline, while simple, achieves a surprisingly high success rate on regular shapes like squares and rectangles, confirming that vertices often serve as strong heuristic pick points.

The most illuminating comparison, however, is between our full method and *polygon vertex (learned)* baseline. While this vertex-only policy is highly effective for regular shapes, achieving near-perfect success rates on squares and rectangles, our dense map approach proves its superior generality on more complex garments. Our method significantly outperforms this policy on t-shirts and pants. This performance gap is not arbitrary and highlights the fundamental advantages of our approach. From a task generality perspective, the superior performance on t-shirts and pants stems from our policy's ability to identify optimal grasping points anywhere along the contour, not just at the vertices, which is crucial for handling their more intricate shapes and folding requirements. From a system robustness perspective, our dense map approach is more resilient to potential inaccuracies in the upstream polygon fitting. The vertex-only policy is brittle to any misalignment, whereas our policy learns affordances directly from the visual contour, allowing it to compensate for slight imperfections. Finally, from a learning process perspective, the dense supervision from hundreds of contour points allows the network to learn a more robust and generalizable feature representation of 'grasping affordance', compared to the sparse signal from just a few vertices. Therefore, while a vertex-based policy is a strong heuristic for simple cases, our approach of learning a dense map represents a more principled and powerful solution.

To address point (4), we conduct a comparative analysis of 8 Large Language Models (gpt-3.5, gpt-4o, gpt-5, claude-3-opus, claude-4.1-opus, gemini-1.5-pro, gemini-2.5-flash and gemini-2.5-pro) across several key aspects. To ensure a fair

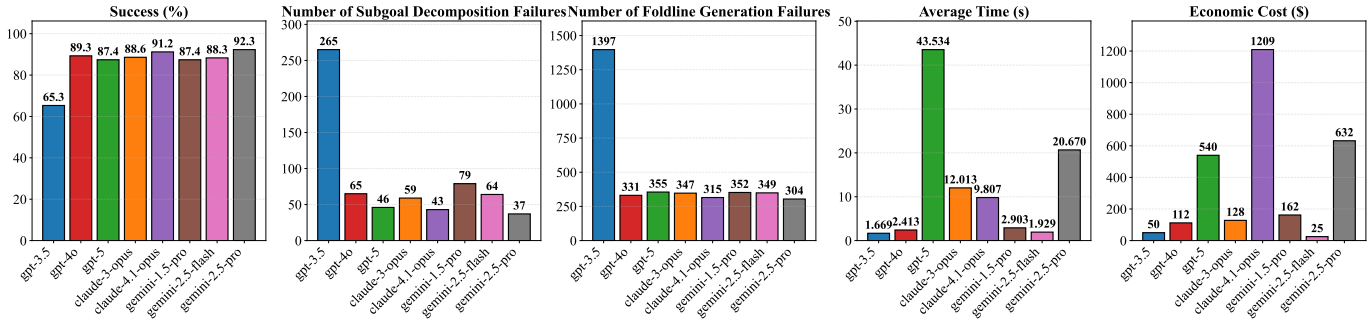


Fig. 12: Performance comparison of different Large Language Models.

(a) Real World Experiment Setup (b) Cloth Instances Used in Experiments

Fig. 13: (a) Real-world experiment setup. (b) Cloth instances used in real-world experiments.

comparison, we utilize the unified interface of LangChain and leverage OpenRouter as a unified LLM service provider. As shown in Fig. 12, for each LLM, we measure its performance across 70 tasks in TABLE I using key metrics including: success rate, number of failures in subgoal decomposition, number of failures in fold line generation, average inference time, and economic cost. The number of failures in subtask decomposition reflects the task decomposition capability of LLMs. On this metric, gemini-2.5-pro achieves the best performance, whereas gpt-3.5 performs substantially worse than all other models. The number of failures in fold line generation serves as an indicator of mathematical reasoning ability, which is critical for inferring pixel-level action representations. In this regard, gemini-2.5-pro again demonstrates the strongest performance, followed by claude-4.1-opus and gpt-4o, while gpt-3.5 remains the weakest. In terms of overall success rate, aside from the earlier-generation gpt-3.5, the performance differences among other LLMs are small, all achieving a success rate above 87%, with gemini-2.5-pro still demonstrating the best performance. However, success rate is only one aspect of evaluating LLM performance. For inference time, gpt-5, gemini-2.5-pro, and the claude series require considerably more time due to the incorporation of deep thinking reasoning mode. This is impractical for robotic manipulation tasks. Moreover, the economic costs associated with different LLMs also vary. Considering these trade-offs, LLMs like gpt-4o and gemini-2.5-flash strike an effective balance between accuracy, speed, and economic cost, which represent the most suitable choice of LLMs for the cloth folding task. In this work, we choose gpt-4o as the LLM model.

TABLE V: Real-world experiment results of PolyFold.

Metrics / Cloth Type	Square	Rectangle	T-shirt	Pant	Average
mIoU	0.793	0.795	0.916	0.816	0.827
success%	82.6	82.5	94.1	85.0	85.7

F. Real-World Experiments

We utilize the ABB YuMi robot as our real-world experimental platform, with an Intel RealSense D435i camera positioned overhead to provide a top-down view, as shown in Fig. 13. In real-world experiments, we utilize 12 different cloth instances that vary in shapes, textures, sizes, and materials, categorized as square, rectangle, t-shirt, and pant. Those cloth instances are evaluated on the 70 tasks in Table I. Similar to simulation experiments, we calculate the *mIoU* and *success rate* metrics compared with the oracle demonstrations.

As shown in Fig. 1 and Table V, experiments on four different cloth types across different tasks perform well, with most results approaching those of the oracle demonstrations. This demonstrates that our proposed PolyFold framework can be seamlessly deployed in real-world experiments without any need for fine-tuning or adaptation, despite the variations in cloth shapes, textures, sizes, and materials. Videos of real-world experiments are available on our webpage https://sites.google.com/view/polyfold.

We also perform an interesting experiment using ambiguous user language instructions to guide cloth folding tasks, as illustrated in Fig. 14. Instead of explicitly directing the robot on specific actions, we provide instructions based on geometric features, task objectives, and general task characteristics. Leveraging the common sense reasoning capabilities of LLMs, our framework effectively completes these tasks on a real-world robot. For example, when given the instruction to *organize the t-shirt by folding it into a rectangular block in three steps*, the robot infers that it should first fold the left and right sleeves inward, followed by folding the bottom edge upward. Using the downstream FG-PnP policy, the robot determines a sequence of appropriate pick-and-place actions. After execution, the results align with user expectations, showcasing the framework's ability to interpret ambiguous instructions and translate them into precise actions.

As shown in Fig. 11(c), we conduct a timing analysis of the real-world experiments, covering the entire process from

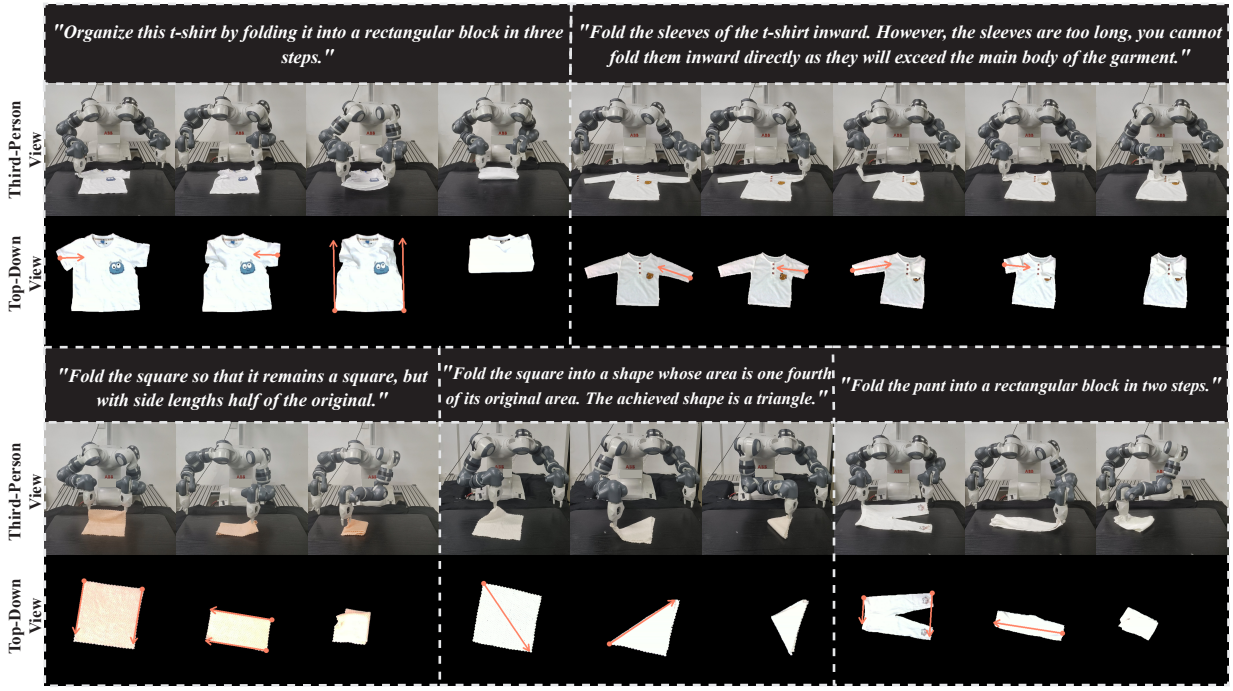


Fig. 14: Real-world task evaluation with only ambiguous user instructions as input. A third-person front view and a top-down view of the ABB YuMi robot execution process are shown and the orange arrow represents the pick-and-place action.

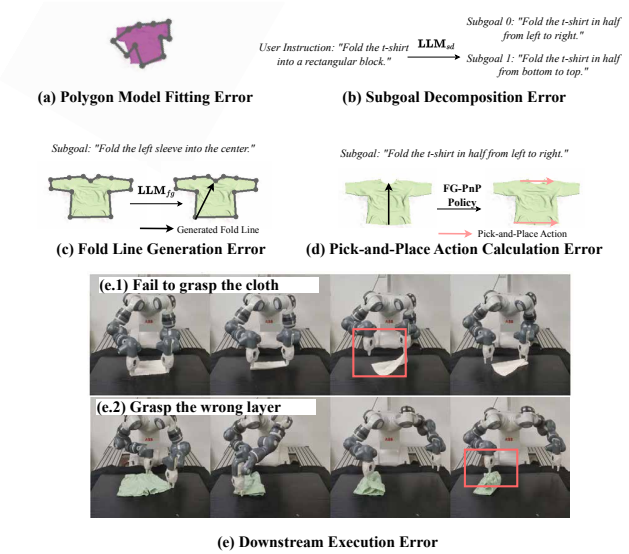


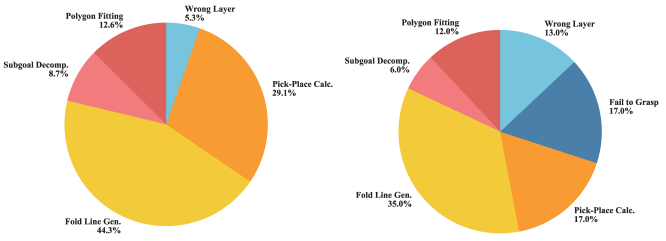
Fig. 15: Demonstration of failure cases in our method PolyFold.

image acquisition and processing to returning to the reset position and completing execution. The timing components are plotted in a pie chart, arranged counterclockwise in sequential order. This analysis reveals that the inference time of PolyFold in the real world is nearly identical to that in simulation. However, the dominant contributor to the total execution time is the robot's action execution. This is a direct consequence of our quasi-static assumption for the pick-and-place folding operation, which requires the robot to operate at a slow speed.

VI. Failure Case Analysis

In Fig. 15, we analyze the potential failure modes of PolyFold. As failures can stem from multiple cascading issues, we attribute any given failure to the first error that arises in the sequence. The first potential issue is a failure in fitting the parameterized polygon model. A poor polygon representation of the task-related part makes it difficult for the LLM to reason about accurate subsequent actions. Second, the subgoal decomposition can fail if the language command is too vague or complex, or if the LLM is not sufficiently capable. Third, even with a correct subgoal, the LLM might still produce an inaccurate fold line. Fourth, given a perfect fold line, the downstream FG-PnP Policy could generate a suboptimal dual-arm action, causing the task to fail. Finally, execution failures can occur at the low-level control stage, primarily due to the thinness of the cloth leading to failed grasps or slipping, or due to the difficulty in distinguishing between cloth layers from imprecise depth perception. Additionally, our policy lacks the ability to recover from failures, which often leads to the termination of the entire task if failures like these occur.

In Fig. 16, we analyze the distribution of PolyFold's failure cases in both simulation and real-world experiments. The analysis shows that the most frequent cause of failure is the LLM-based fold line generation, followed by the computation of pick-and-place actions. A key difference emerges in execution errors: in simulation, failures such as missed grasps or slippage are absent due to the default setting of a perfect gripper attachment, although failures from grasping the wrong layer can still occur. In the real-world experiments, however, execution errors are far more prominent, accounting for 30% of all failures. These are cases where a correct action was planned, but an issue arose during its actual execution process.



(a) Breakdown of Failure Cases in Simulation (b) Breakdown of Failure Cases in Real-World

Fig. 16: Breakdown of failure cases for our method PolyFold in both simulation and real-world experiments.

VII. CONCLUSIONS

In this paper we introduce PolyFold, a novel language-conditioned bimanual cloth folding framework that features strong zero-shot generalization, inherent multi-step reasoning capability, and expert-demonstration-free policy learning. The essence of PolyFold lies in two parts: (1) the introduction of a parameterized polygon model as an efficient abstraction and grounding module for LLMs to understand cloth states; (2) an appropriate planning hierarchy that leverages LLMs for subtask decomposition and intermediate action reasoning, leaving the downstream policy to decide bimanual pick-and-place actions. We believe PolyFold points out an efficient way for generalizable deformable object manipulation.

REFERENCES

[1] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, p. eabd8803, 2021.

[2] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, J. Kober, X. Li *et al.*, "Challenges and outlook in robotic manipulation of deformable objects," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 67–77, 2022.

[3] K. Mo, C. Xia, X. Wang, Y. Deng, X. Gao, and B. Liang, "Foldsformer: Learning sequential multi-step cloth manipulation with space-time attention," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 760–767, 2023.

[4] T. Weng, S. M. Bajracharya, Y. Wang, K. Agrawal, and D. Held, "Fabricflownet: Bimanual cloth manipulation with a flow-based policy," in *Conference on Robot Learning*. PMLR, 2022, pp. 192–202.

[5] D. Seita, A. Ganapathi, R. Hoque, M. Hwang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali *et al.*, "Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9651–9658.

[6] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. K. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "Visuospatial foresight for physical sequential fabric manipulation," *Autonomous Robots*, vol. 46, no. 1, pp. 175–199, 2022.

[7] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 894–906.

[8] Y. Deng, K. Mo, C. Xia, and X. Wang, "Learning language-conditioned deformable object manipulation with graph dynamics," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 7508–7514.

[9] R. Jangir, G. Alenya, and C. Torras, "Dynamic cloth manipulation with deep reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4630–4636.

[10] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 734–743.

[11] D. Blanco-Mulero, G. Alcan, F. J. Abu-Dakka, and V. Kyrki, "Qdp: Learning to sequentially optimise quasi-static and dynamic manipulation primitives for robotic cloth manipulation," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 984–991.

[12] L. Yang, Y. Li, and L. Chen, "Clothppo: a proximal policy optimization enhancing framework for robotic cloth manipulation with observation-aligned action spaces," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024, pp. 6895–6903.

[13] S. Wang, R. Papallas, M. Leonetti, and M. Dogar, "Goal-conditioned action space reduction for deformable object manipulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3823–3830.

[14] X. Lin, Y. Wang, Z. Huang, and D. Held, "Learning visible connectivity dynamics for cloth smoothing," in *Conference on Robot Learning*. PMLR, 2022, pp. 256–266.

[15] X. Ma, D. Hsu, and W. S. Lee, "Learning latent graph dynamics for visual manipulation of deformable objects," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8266–8273.

[16] OpenAI, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[17] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9493–9500.

[18] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian *et al.*, "Do as I can, not as I say: Grounding language in robotic affordances," in *Conference on Robot Learning*. PMLR, 2023, pp. 287–318.

[19] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," in *Conference on Robot Learning*. PMLR, 2023, pp. 540–562.

[20] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," in *Conference on Robot Learning*. PMLR, 2023, pp. 3766–3777.

[21] V. Raval, E. Zhao, H. Zhang, S. Nikolaidis, and D. Seita, "GPT-fabric: Smoothing and folding fabric by leveraging pre-trained foundation models," in *International Symposium of Robotics Research (ISRR)*, 2024.

[22] Y. Deng and D. Hsu, "General-purpose clothes manipulation with semantic keypoints," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 13 181–13 187.

[23] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2025, pp. 4573–4602.

[24] T. Fu, C. Li, J. Liu, F. Li, C. Wang, and R. Song, "Flingflow: LLM-driven dynamic strategies for efficient cloth flattening," *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8714–8721, 2024.

[25] Y. Wang, Z. Sun, J. Zhang, Z. Xian, E. Biyik, D. Held, and Z. Erickson, "Rl-vlm-f: Reinforcement learning from vision language foundation model feedback," in *International Conference on Machine Learning*. PMLR, 2024, pp. 51 484–51 501.

[26] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," *arXiv preprint arXiv:2104.13921*, 2021.

[27] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen *et al.*, "Simple open-vocabulary object detection," in *European conference on computer vision*. Springer, 2022, pp. 728–755.

[28] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4015–4026.

[29] S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, "A geometric approach to robotic laundry folding," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 249–267, 2012.

[30] A. Doumanoglou, J. Stria, G. Peleka, I. Mariolis, V. Petrik, A. Kargakos, L. Wagner, V. Hlaváč, T.-K. Kim, and S. Malassiotis, "Folding clothes autonomously: A complete pipeline," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1461–1478, 2016.

[31] C. Bersch, B. Pitzer, and S. Kammel, "Bimanual robotic cloth manipulation for laundry folding," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1413–1419.

[32] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 432–448.
[33] G. Salhotra, I.-C. A. Liu, M. Dominguez-Kuhne, and G. S. Sukhatme, "Learning deformable object manipulation from expert demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8775–8782, 2022.
[34] H. Xue, Y. Li, W. Xu, H. Li, D. Zheng, and C. Lu, "Unifolding: Towards sample-efficient, scalable, and generalizable robotic garment folding," in *Conference on Robot Learning*. PMLR, 2023, pp. 3321–3341.
[35] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[36] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 813–824.
[37] R. Lee, D. Ward, V. Dasagi, A. Cosgun, J. Leitner, and P. Corke, "Learning arbitrary-goal fabric folding with one hour of real robot experience," in *Conference on Robot Learning*. PMLR, 2021, pp. 2317–2327.
[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[39] L. Metz, J. Ibarz, N. Jaitly, and J. Davidson, "Discrete sequential prediction of continuous actions for deep rl," *arXiv preprint arXiv:1705.05035*, 2017.
[40] J. Wu, X. Sun, A. Zeng, S. Song, J. Lee, S. Rusinkiewicz, and T. Funkhouser, "Spatial action maps for mobile manipulation," *arXiv preprint arXiv:2004.09141*, 2020.
[41] H. Ha and S. Song, "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding," in *Conference on Robot Learning*. PMLR, 2022, pp. 24–33.
[42] 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)*. IEEE, 2023, pp. 5872–5879.
[43] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng, "Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4568–4575.
[44] K. Xu, S. Zhao, Z. Zhou, Z. Li, H. Pi, Y. Zhu, Y. Wang, and R. Xiong, "A joint modeling of vision-language-action for target-oriented grasping in clutter," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 597–11 604.
[45] E. Stengel-Eskin, A. Hundt, Z. He, A. Murali, N. Gopalan, M. Gombolay, and G. Hager, "Guiding multi-step rearrangement tasks with natural language instructions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1486–1501.
[46] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman *et al.*, "Grounded decoding: Guiding text generation with grounded models for embodied agents," *Advances in Neural Information Processing Systems*, vol. 36, pp. 59 636–59 661, 2023.
[47] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.
[48] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.
[49] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *Computer Vision – ECCV 2020*, vol. 12346, 2020, pp. 405–421.
[50] H. Bertiche, M. Madadi, and S. Escalera, "Cloth3d: clothed 3d humans," in *European Conference on Computer Vision*. Springer, 2020, pp. 344–359.
[51] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[52] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and computing in applied probability*, vol. 1, no. 2, pp. 127–190, 1999.

Haozhe Du received his B.Eng. degree in Control Science and Engineering (with a minor in Mechatronic Engineering) from Zhejiang University in 2022, and his M.Eng. degree in 2025. His research interests include embodied artificial intelligence and deformable object manipulation.

Kechun Xu received her B.Eng. in Control Science and Engineering from Zhejiang University, Hangzhou, China, in 2021.
She is currently working toward her Ph.D. degree at the State Key Laboratory of Industrial Control Technology and Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China. Her research interests include manipulation and robot learning.

Rong Xiong received her Ph.D. in Control Science and Engineering from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China in 2009. She is currently a Professor in the Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China. Her latest research interests include motion planning and SLAM.

Yue Wang received his Ph.D. degree in Control Science and Engineering from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China in 2016. He is currently a Professor in the Department of Control Science and Engineering, Zhejiang University, Hangzhou, P.R. China. His latest research interests include mobile robotics and robot perception.