## APPENDIX A
## POLYFOLD IMPLEMENTATION DETAILS

### A. Parameterized Polygon Model

**Classic Polygon Model Fitting Pipeline.** The polygon model fitting process adjusts the current parameters of the polygon model to best match the contour obtained from RGB observations through black-box optimization. As illustrated in the original paper [29], the strategy of the optimization process is to relax the constraints on parameters to be optimized gradually, which can be divided into at most four stages: initialization, orientation optimization, symmetry optimization and asymmetry optimization. In this process, the constraints on the parameters are gradually reduced. During the initialization stage, principal component analysis (PCA) is used to obtain a coarse set of parameters, including rotation angle, scale, and translation factor, to roughly align the parameterized polygon model with the observed cloth contour. In the orientation optimization stage, only the rotation of the polygon model is adjusted to better fit the real contour. In the symmetry optimization phase, parameters are optimized while maintaining the symmetrical structure of the polygon model (e.g. in the case of a t-shirt, the *left_shoulder_top* and *right_shoulder_top* points are symmetrical relative to the vertical centerline). Regardless of changes in point coordinates, the symmetry must be preserved. During the asymmetry optimization stage, all parameters can be adjusted freely, without symmetry constraints. For fitting square, rectangle, or pants objects, the process is simpler, and the symmetry optimization stage can be omitted. For the fitting process of folded model, as polygon model of previous step offers a good initial value, only asymmetry optimization is used for efficient polygon model fitting. Fig. 17 illustrates the fitting process of different types of flattened clothes, as well as the fitting process of corresponding folded clothes after each step of folding, where the gray points and lines represent the vertices and edges of parameterized polygon model.

For structural penalty, as depicted in Section III, in order to prevent the polygon model from excessive and unrealistic deformations during the optimization process, soft constraints are introduced. The structural penalty function $\Gamma(\mathcal{X}_t)$ imposes penalties on parameters that violate these soft constraints $\Phi$ and otherwise yields zero. For example, optimization related to t-shirt objects, one soft constraint is that the horizontal length (distance from the left sleeve to the right sleeve) should be less than 5 times the vertical length (distance from the bottom edge to the top collar) to make it form a reasonable shape. For detailed soft constraints and structural penalties, readers are advised to refer to the original paper [29] and its official code implementation[1].

While the original parameterized polygon model provides a strong foundation, its fitting process relies on a multi-stage optimization that can be brittle in challenging, real-world scenarios. We identify two key bottlenecks and propose corresponding enhancements to improve the robustness of our grounding module.

[1]https://github.com/rll/visual_feedback/blob/master/clothing_models

**Semantic-Aware Initialization for Initial Fitting.** The original method utilizes Principal Component Analysis (PCA) on the observed cloth contour to determine the initial orientation for fitting. However, PCA is inherently direction-agnostic. For garments with only a single axis of symmetry (e.g., t-shirts and pants, which are symmetric left-to-right but not top-to-bottom), this leads to a 180-degree ambiguity. Although orientation optimization can sometimes recover the correct fit, in most cases the subsequent optimization converges to a local minimum, resulting in failure. As shown in Fig. 18, when the absolute angle between the cloth and the polygon model exceeds 90°, this situation may tend to occur, leading to an incorrectly fitted polygon model. A natural idea is to employ two polygon model templates with a 180° rotation difference for parallel fitting. However, as illustrated in Fig. 18(d), the incorrectly fitted model can still exhibit a high contour similarity, making it difficult to directly determine which result is correct.

To resolve this ambiguity, we introduce a semantic-aware initialization step. Before running the fitting optimization, we first pass the input image to a lightweight, pretrained keypoint detector for clothes [42]. While this detector is not precise enough for full state estimation, it reliably provides crucial semantic information, such as the approximate locations of the *left sleeve* and *right sleeve*. By comparing the relative positions of these detected semantic keypoints, we can unambiguously determine the correct orientation of the garment's principal axis. This semantically-informed orientation is then used to initialize the fitting process. To be more specific, we can divide the points into two sets, the left set $P_L$ and the right set $P_R$. For t-shirt, $P_L$ contains *left sleeve*, *left bottom*, *left collar*; for pant, $P_L$ contains *left top waist* and *left leg*. The right point set contains corresponding right-sided points. Then we calculate the mean vector from the left set to the right set:

$$v = \frac{1}{m}\sum_{i=1}^{m} P_{R,i} - P_{L,i} \tag{9}$$

where $m$ is the number of points in one point set. Then we can apply a counter-clockwise rotation of 90 degrees to the mean vector:

$$v_{semantic} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot v \tag{10}$$

Then we can calculate the dot product between $v_{semantic}$ and principal axis $v_{pca}$. If it is below zero, then we revert the direction of the principal axis $v_{pca} = -v_{pca}$, otherwise the direction of the principal axis remains unchanged.

**Registration-based Initialization for Folded Models.** The original fitting process for folded models assumes that the cloth's pose (position and orientation) remains relatively static between folding steps. However, during real-world execution, interactions with the gripper can cause the cloth to be inadvertently dragged, leading to relatively large, unexpected translations and rotations. In such cases, using the previous step's parameters as the initial guess for the current step's fitting is no longer valid and leads to failure, as shown in Fig. 19.

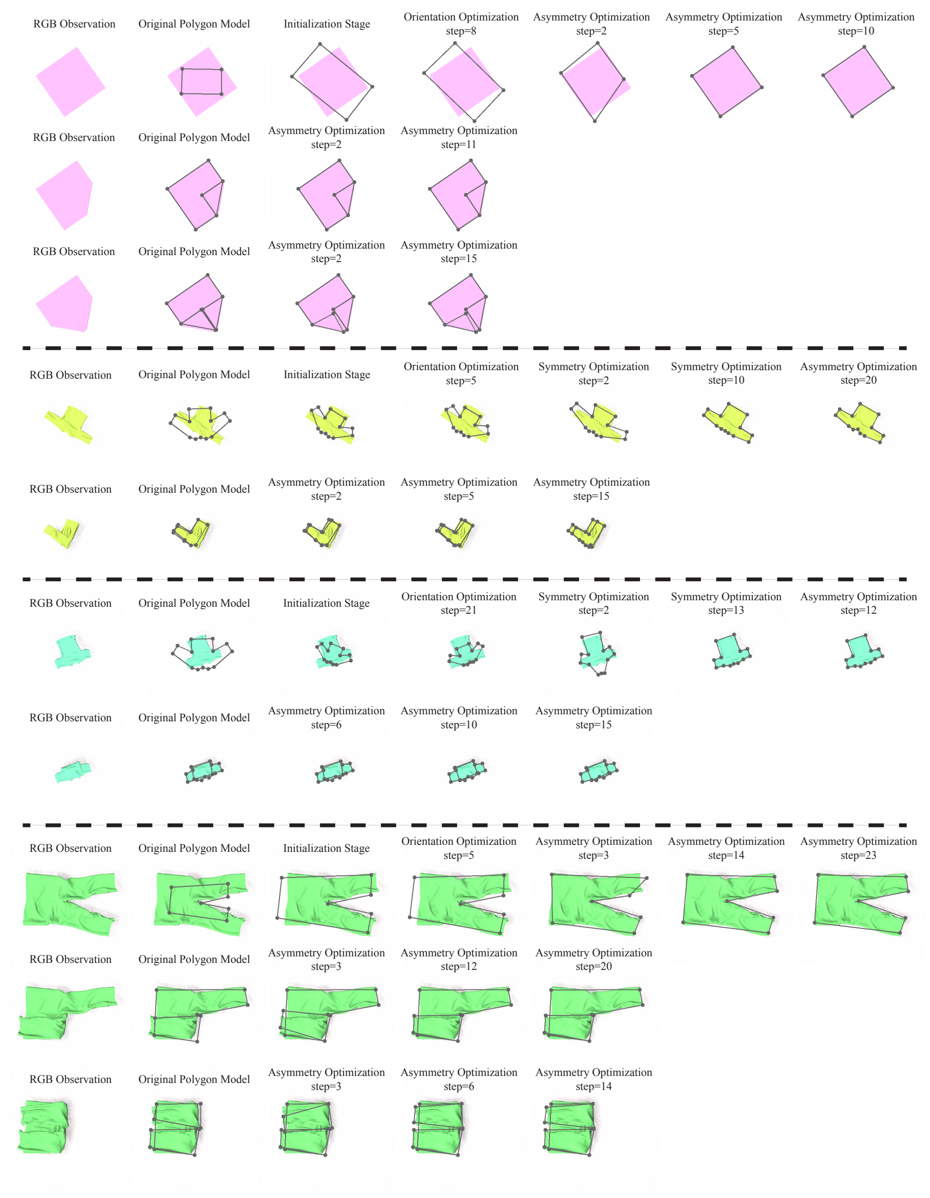

RGB Observation
Original Polygon Model
Initialization Stage
Orientation Optimization step=8
Asymmetry Optimization step=2
Asymmetry Optimization step=5
Asymmetry Optimization step=10
RGB Observation
Original Polygon Model
Asymmetry Optimization step=2
Asymmetry Optimization step=11
RGB Observation
Original Polygon Model
Asymmetry Optimization step=2
Asymmetry Optimization step=15
RGB Observation
Original Polygon Model
Initialization Stage
Orientation Optimization step=5
Symmetry Optimization step=2
Symmetry Optimization step=10
Asymmetry Optimization step=20
RGB Observation
Original Polygon Model
Asymmetry Optimization step=2
Asymmetry Optimization step=5
Asymmetry Optimization step=15
RGB Observation
Original Polygon Model
Initialization Stage
Orientation Optimization step=21
Symmetry Optimization step=2
Symmetry Optimization step=13
Asymmetry Optimization step=12
RGB Observation
Original Polygon Model
Asymmetry Optimization step=6
Asymmetry Optimization step=10
Asymmetry Optimization step=15
RGB Observation
Original Polygon Model
Initialization Stage
Orientation Optimization step=5
Asymmetry Optimization step=3
Asymmetry Optimization step=14
Asymmetry Optimization step=23
RGB Observation
Original Polygon Model
Asymmetry Optimization step=3
Asymmetry Optimization step=12
Asymmetry Optimization step=20
RGB Observation
Original Polygon Model
Asymmetry Optimization step=3
Asymmetry Optimization step=6
Asymmetry Optimization step=14

Fig. 17: Polygon model fitting pipeline of flattened and folded clothes, optionally including the initialization, orientation-optimization, symmetry-optimization, and asymmetry-optimization stages.

To address this, we incorporate a registration module for initializing the fit of folded models. After a fold is executed, we update the polygon parameters to obtain an initial folded polygon model and its corresponding mask. This is done through applying binary thresholding to separate the polygon outline from the background, detecting all contours and selecting the largest one, and then filling the detected contour to create a solid mask. The cloth mask is obtained through GrabCut algorithm with a predefined, fixed bounding box fed into it. Then we uniformly sample 1000 points inside the mask and perform 2D point-set registration using ICP algorithm. This registration step solves for the 2D rigid transformation (rotation and translation) that best aligns the two sampled point sets. The resulting transformation is then applied to the polygon model from the previous step to generate a much more accurate initial guess for the current pose.

**Failure Cases Analysis.** We categorize polygon model failures into three primary types, as illustrated in Fig. 20.

The first type, inaccurate model fitting, can occur during both the initial and subsequent folded model fitting stages. Our simple optimization objective, based on the Chamfer distance of the contour points, can sometimes converge to a local minimum, resulting in an inaccurate fit. However, the impact of such fitting errors is highly dependent on the task context. If the inaccuracy occurs in a task-relevant region—for instance, a poor fit on the left sleeve for a left-sleeve-folding task (as shown in Fig. 20(a), top)—it often leads to a planning failure. If the fitting error is minor or located in a region irrelevant to the current manipulation subgoal (Fig. 20(a), bottom), the system can often still succeed, demonstrating a degree of robustness.

Another significant failure mode arises from robot execution

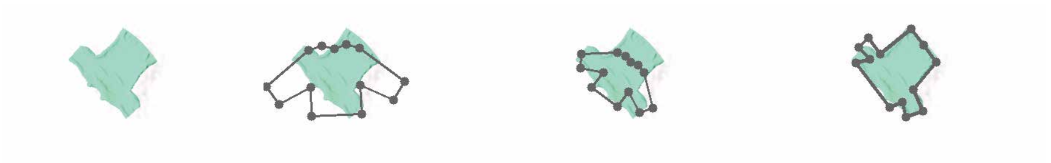

(a) Initial Cloth
(b) Initial Polygon Model
(c) Polygon Model After PCA
(d) Wrong Fitted Polygon Model

Fig. 18: Illustration of the PCA ambiguity that can lead to an incorrectly fitted polygon model.

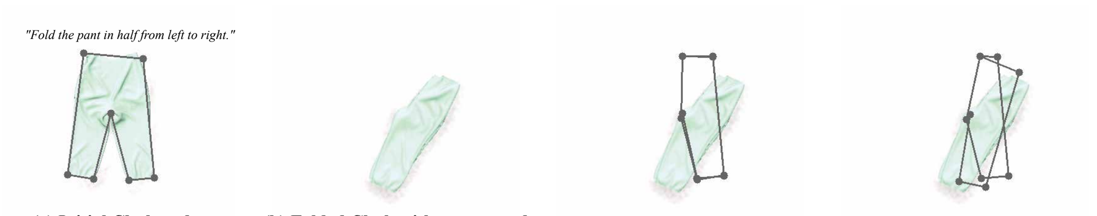

"Fold the pant in half from left to right."
(a) Initial Cloth and Fitted Polygon Model
(b) Folded Cloth with unexpected translation and rotation
(c) Initial Folded Polygon Model
(d) Fitted Folded Polygon Model

Fig. 19: Illustration of an incorrectly fitted folded polygon model caused by unexpected rotation and translation during execution.

errors that lead to unexpected cloth states. For example, during a multi-step task like folding pants first left-to-right and then bottom-to-top, the gripper might only grasp the top layer in the second step instead of all layers (as depicted in Fig. 20(b)). This creates a chaotic physical state that deviates significantly from the expected folded model. Consequently, the subsequent polygon model fitting process fails, as it cannot find a valid fit for the disordered configuration. This results in a complete breakdown of the grounding module, making it impossible for the system to recover.

Finally, our framework's scope is inherently limited by its template-based design, where each polygon model is tailored for a specific garment category. When the system is presented with an out-of-distribution object for which no template exists—such as attempting to fit a t-shirt model to a hoodie (Fig. 20(c))—the polygon abstraction breaks down entirely due to the topological mismatch. This highlights that extending the system to handle novel garment topologies is an important direction for future research.

## B. Prompt Templates for Hierarchical LLMs

Here we show the prompt templates for subgoal decomposition LLM in Section IV-B and symmetrical fold line generation LLM in Section IV-C, respectively in Fig. 22 and Fig. 23. The components of the two prompt templates are similar, which mainly contain (1) system information indicating the role of the LLM and what the LLM should do, (2) definition and detailed illustration of parameterized polygon model, (3) input/output format and task definition, and (4) few examples. In subgoal decomposition Large Language Model, only one example is provided, teaching the LLM to decompose the goal of *folding both sleeves of the t-shirt into the center*. In symmetrical fold line generation LLM, we utilize two examples: *fold the top left corner of the square to the center* and *fold the left sleeve of the t-shirt inwards to the center*, to teach the LLM about the concept of fold line in cloth folding and how to reason about it.

**Algorithm 1** PolyFold Algorithm

```
Input: Language instruction $\mathcal{L}$; RGB observation $\{o_t\}$
Task: Fold the cloth based on the language instruction $\mathcal{L}$.
    Annotation:
    $\mathbf{LLM_{sd}}$: subgoal decomposition LLM
    $\mathbf{LLM_{fg}}$: symmetrical fold line generation LLM
    UNet: network utilized for coupled spatial action map calculation
    caso: conditional pick point affordance score optimization
    FitPolygon($o$, $\mathcal{P}$): Optimize the parameters of input polygon $\mathcal{P}$ to fit current observation $o$
    FoldPolygon($\mathcal{P}$, F): Update the polygon model $\mathcal{P}$ to fold $\mathcal{P}$ according to the given fold line vector F
    symm($p$, F): calculate the symmetric point of a given point $p$ with respect to symmetry axis F
1:  Initialize parameterized polygon model $\mathcal{P}_0$.
2:  $\{\mathcal{L}_t\} \leftarrow \mathbf{LLM_{sd}}(\mathcal{L}, \mathcal{P}_0)$
3:  $T \leftarrow \mathbf{length}(\{\mathcal{L}_t\})$
4:  for $t \leftarrow 0$ to $T-1$ do
5:      Get current RGB observation $o_t$
6:      $\mathcal{P}_t \leftarrow \mathbf{FitPolygon}(o_t, \mathcal{P}_t)$.
7:      $\mathbf{F_t} \leftarrow \mathbf{LLM_{fg}}(\mathcal{L}_t, \mathcal{P}_t)$
8:      Crop RGB observation $o_t$ to leave only the part that is on the left side of the fold line $\mathbf{F_t}$ and segment the cloth mask, denoted as $\tilde{o}_t$.
9:      $a_t^{pick1} \leftarrow \arg\max \mathbf{UNet}(\tilde{o}_t, \mathbf{F_t})$
10:     Extract cloth contour $\tilde{\mathcal{C}}_t$ from partial mask $\tilde{o}_t$.
11:     $a_t^{pick2} \leftarrow \mathbf{caso}(\tilde{\mathcal{C}}_t, \mathbf{F_t}, a_t^{pick1})$
12:     $a_t^{place1} \leftarrow \mathbf{symm}(a_t^{pick1}, \mathbf{F_t})$
13:     $a_t^{place2} \leftarrow \mathbf{symm}(a_t^{pick2}, \mathbf{F_t})$
14:     The robot executes bimanual action.
15:     $\mathcal{P}_{t+1} \leftarrow \mathbf{FoldPolygon}(\mathcal{P}_t, \mathbf{F_t})$
16: end for
```

## C. Pseudocode of PolyFold

In this section, we present the pseudocode for our proposed cloth-folding framework, PolyFold in Algorithm 1, providing

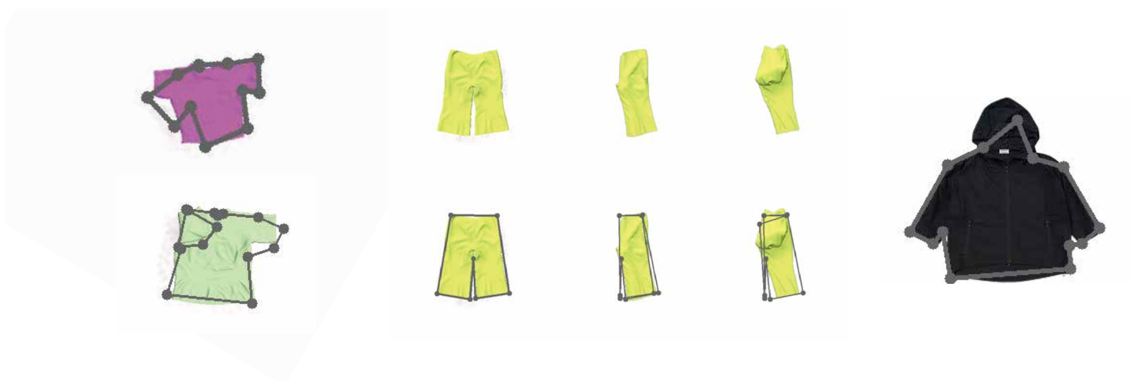

(a) Inaccurate Polygon Model Fitting (b) Downstream Execution Error (c) Out-of-Distribution Cloth

Fig. 20: Potential failure cases of polygon models.

a detailed description of the algorithm.

## Appendix B
## Simulation Experiment Setup

### A. Evaluated Tasks

As described in Section V-A and Table I, we generate 70 tasks for the simulation evaluation of our proposed method, baselines and ablation studies. A detailed illustration of these tasks is provided in Table VI and Table VII. A visualization of evaluated tasks is also provided in Fig. 21. The evaluated tasks include four types of cloth and a wide variety of cloth folding tasks, which can be used to effectively test the validity, robustness, and generalization of each method.

### B. Evaluation Metrics

We employ the same metrics as in previous works [3], [8]. As the simulation platform SoftGym [32] is a particle-based simulation, the first metric is mean particle position error (MPPE), which assesses the particle position error of current achieved folded cloth and those in scripted oracle demonstration. MPPE is a relatively strict metric as it evaluates the distance of each particle and its corresponding one in oracle demonstration. Another metric, the mean Intersection over Union (mIoU), compares the mask of the folded cloth with that in the oracle demonstration. mIoU is a relatively lenient metric as it only assesses the contour and the area covered by the cloth, ignoring the details of the interior structure. Referring to these two criteria, we also use the success rate to assess the quality of task completion. When MPPE is below a certain threshold $\kappa_1$ and mIoU is above a certain threshold $\kappa_2$, we consider the task to be successful. For square, rectangle and t-shirt tasks, we choose $\kappa_1 = 20$mm, $\kappa_2 = 0.8$ for single-step tasks and $\kappa_1 = 50$mm, $\kappa_2 = 0.7$ for multi-step tasks. For pant tasks, as pant meshes imported into the simulator exhibit increased thickness relative to other meshes and this distribution leads to a larger variation in the distances between the pant mesh and the oracle mesh, as particles tend to deviate more from their corresponding positions in the oracle mesh compared to square, rectangle and t-shirt meshes. Therefore in pant tasks, we choose $\kappa_1 = 35$mm, $\kappa_2 = 0.8$ for single-step tasks and $\kappa_1 = 65$mm, $\kappa_2 = 0.7$ for multi-step tasks.

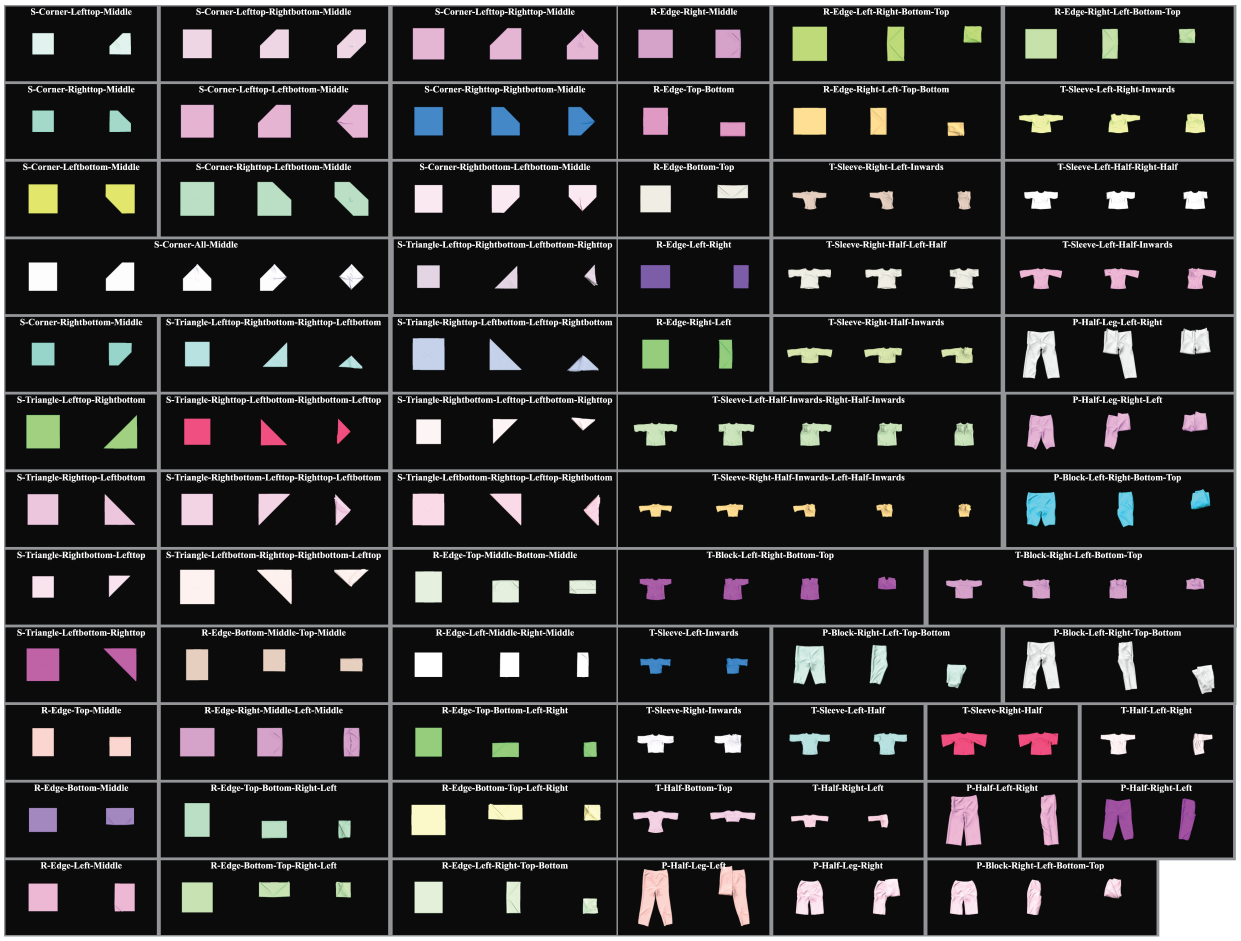
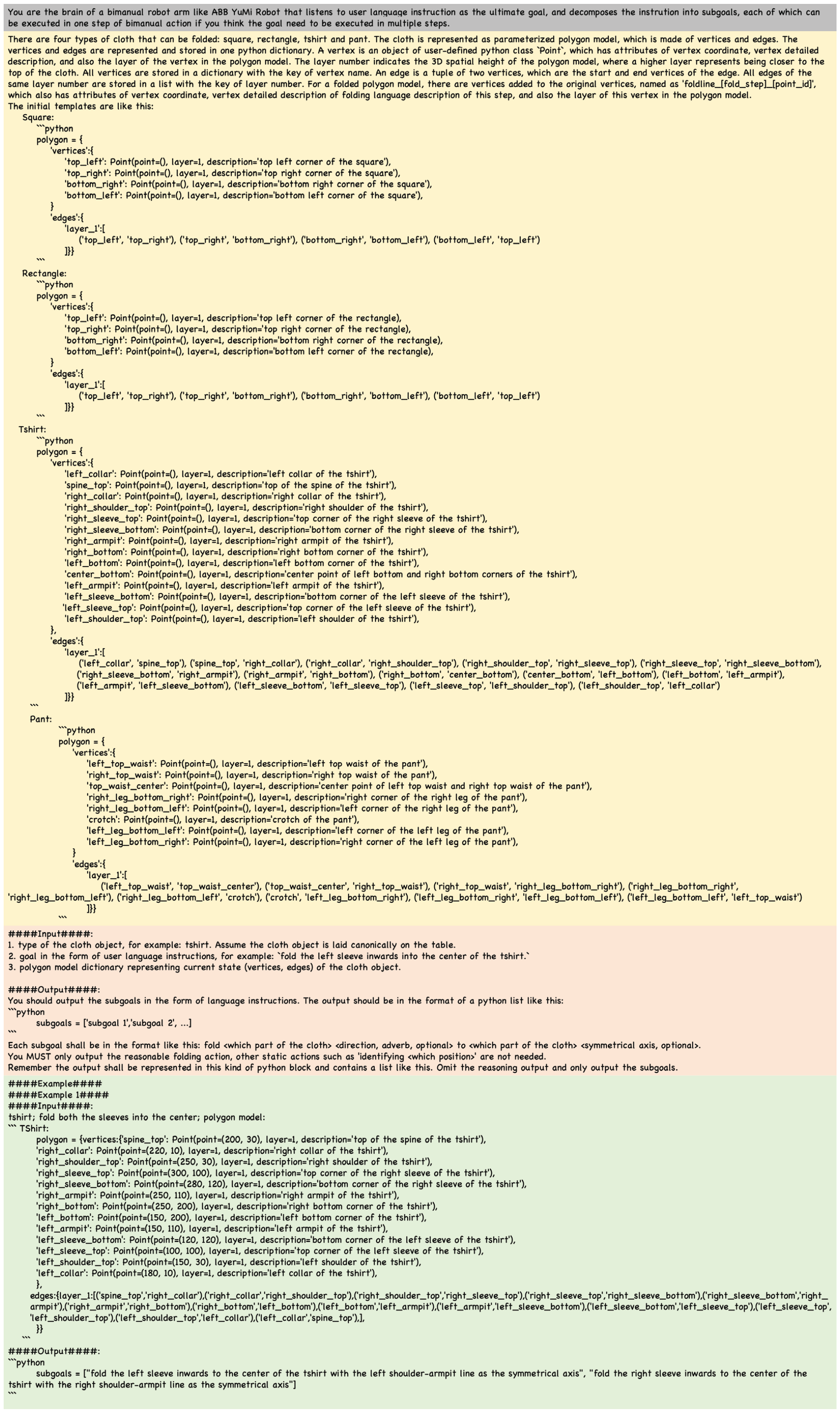

S-Corner-Lefttop-Middle
S-Corner-Lefttop-Rightbottom-Middle
S-Corner-Lefttop-Righttop-Middle
R-Edge-Right-Middle
R-Edge-Left-Right-Bottom-Top
R-Edge-Right-Left-Bottom-Top
S-Corner-Righttop-Middle
S-Corner-Lefttop-Leftbottom-Middle
S-Corner-Righttop-Rightbottom-Middle
R-Edge-Top-Bottom
R-Edge-Right-Left-Top-Bottom
T-Sleeve-Left-Right-Inwards
S-Corner-Leftbottom-Middle
S-Corner-Righttop-Leftbottom-Middle
S-Corner-Rightbottom-Leftbottom-Middle
R-Edge-Bottom-Top
T-Sleeve-Right-Left-Inwards
T-Sleeve-Left-Half-Right-Half
S-Corner-All-Middle
S-Triangle-Lefttop-Rightbottom-Leftbottom-Righttop
R-Edge-Left-Right
T-Sleeve-Right-Half-Left-Half
T-Sleeve-Left-Half-Inwards
S-Corner-Rightbottom-Middle
S-Triangle-Lefttop-Rightbottom-Righttop-Leftbottom
S-Triangle-Righttop-Leftbottom-Lefttop-Rightbottom
R-Edge-Right-Left
T-Sleeve-Right-Half-Inwards
P-Half-Leg-Left-Right
S-Triangle-Lefttop-Rightbottom
S-Triangle-Righttop-Leftbottom-Rightbottom-Lefttop
S-Triangle-Rightbottom-Lefttop-Leftbottom-Righttop
T-Sleeve-Left-Half-Inwards-Right-Half-Inwards
P-Half-Leg-Right-Left
S-Triangle-Righttop-Leftbottom
S-Triangle-Rightbottom-Lefttop-Righttop-Leftbottom
S-Triangle-Leftbottom-Righttop-Lefttop-Rightbottom
T-Sleeve-Right-Half-Inwards-Left-Half-Inwards
P-Block-Left-Right-Bottom-Top
S-Triangle-Rightbottom-Lefttop
S-Triangle-Leftbottom-Righttop-Rightbottom-Lefttop
R-Edge-Top-Middle-Bottom-Middle
T-Block-Left-Right-Bottom-Top
T-Block-Right-Left-Bottom-Top
S-Triangle-Leftbottom-Righttop
R-Edge-Bottom-Middle-Top-Middle
R-Edge-Left-Middle-Right-Middle
T-Sleeve-Left-Inwards
P-Block-Right-Left-Top-Bottom
P-Block-Left-Right-Top-Bottom
R-Edge-Top-Middle
R-Edge-Right-Middle-Left-Middle
R-Edge-Top-Bottom-Left-Right
T-Sleeve-Right-Inwards
T-Sleeve-Left-Half
T-Sleeve-Right-Half
T-Half-Left-Right
R-Edge-Bottom-Middle
R-Edge-Top-Bottom-Right-Left
R-Edge-Bottom-Top-Left-Right
T-Half-Bottom-Top
T-Half-Right-Left
P-Half-Left-Right
P-Half-Right-Left
R-Edge-Left-Middle
R-Edge-Bottom-Top-Right-Left
R-Edge-Left-Right-Top-Bottom
P-Half-Leg-Left
P-Half-Leg-Right
P-Block-Right-Left-Bottom-Top

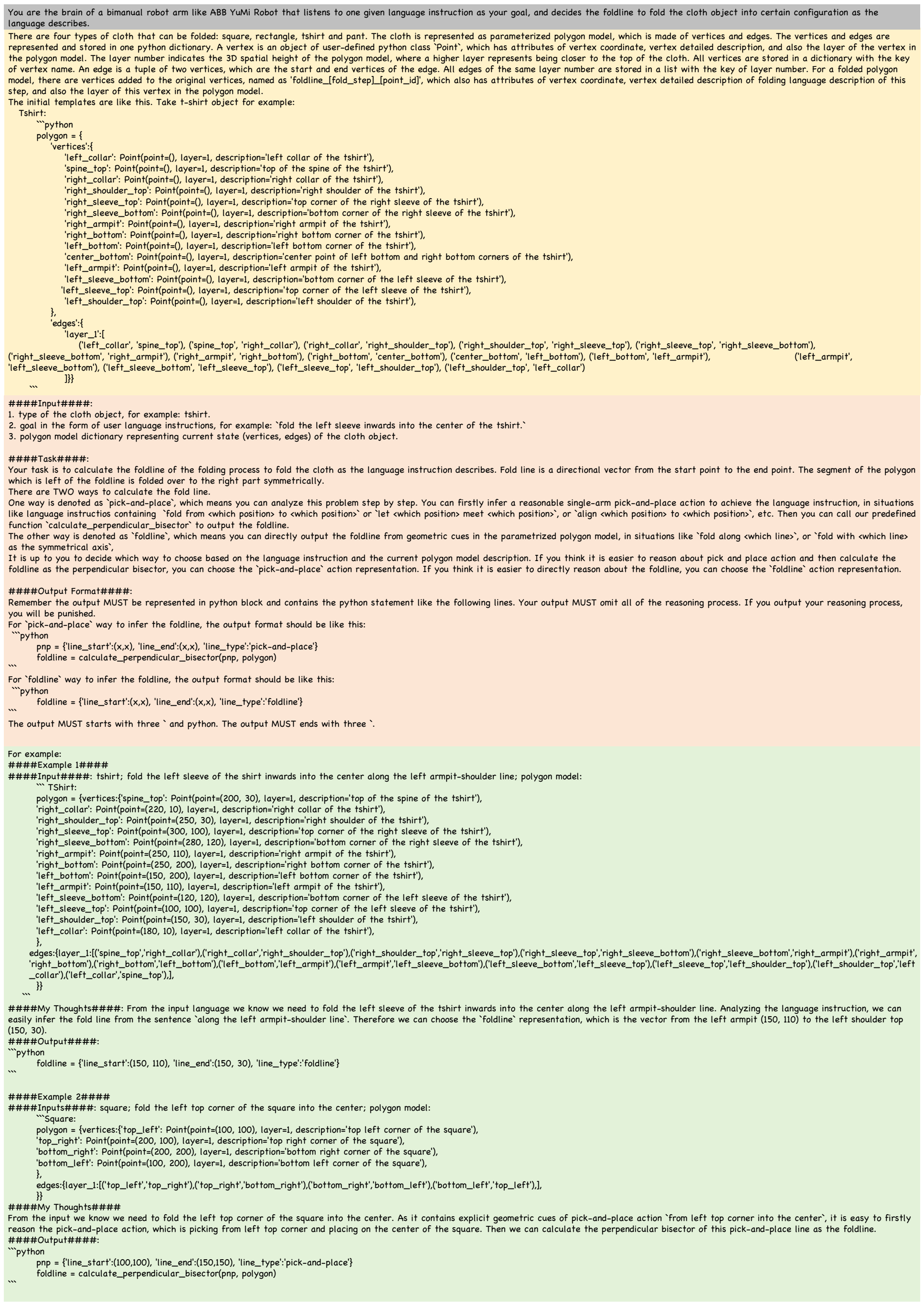
Fig. 21: 70 evaluated tasks in SoftGym simulator.



~~~
You are the brain of a bimanual robot arm like ABB YuMi Robot that listens to user language instruction as the ultimate goal, and decomposes the instrution into subgoals, each of which can
be executed in one step of bimanual action if you think the goal need to be executed in multiple steps.
There are four types of cloth that can be folded: square, rectangle, tshirt and pant. The cloth is represented as parameterized polygon model, which is made of vertices and edges. The
vertices and edges are represented and stored in one python dictionary. A vertex is an object of user-defined python class `Point`, which has attributes of vertex coordinate, vertex detailed
description, and also the layer of the vertex in the polygon model. The layer number indicates the 3D spatial height of the polygon model, where a higher layer represents being closer to the
top of the cloth. All vertices are stored in a dictionary with the key of vertex name. An edge is a tuple of two vertices, which are the start and end vertices of the edge. All edges of the
same layer number are stored in a list with the key of layer number. For a folded polygon model, there are vertices added to the original vertices, named as 'foldline_[fold_step]_[point_id]',
which also has attributes of vertex coordinate, vertex detailed description of folding language description of this step, and also the layer of this vertex in the polygon model.
The initial templates are like this:
    Square:
        ```python
        polygon = {
            'vertices':{
                'top_left': Point(point=(), layer=1, description='top left corner of the square'),
                'top_right': Point(point=(), layer=1, description='top right corner of the square'),
                'bottom_right': Point(point=(), layer=1, description='bottom right corner of the square'),
                'bottom_left': Point(point=(), layer=1, description='bottom left corner of the square'),
            }
            'edges':{
                'layer_1':[
                    ('top_left', 'top_right'), ('top_right', 'bottom_right'), ('bottom_right', 'bottom_left'), ('bottom_left', 'top_left')
                ]}}
        ```
    Rectangle:
        ```python
        polygon = {
            'vertices':{
                'top_left': Point(point=(), layer=1, description='top left corner of the rectangle),
                'top_right': Point(point=(), layer=1, description='top right corner of the rectangle),
                'bottom_right': Point(point=(), layer=1, description='bottom right corner of the rectangle),
                'bottom_left': Point(point=(), layer=1, description='bottom left corner of the rectangle),
            }
            'edges':{
                'layer_1':[
                    ('top_left', 'top_right'), ('top_right', 'bottom_right'), ('bottom_right', 'bottom_left'), ('bottom_left', 'top_left')
                ]}}
        ```
   Tshirt:
        ```python
        polygon = {
            'vertices':{
                'left_collar': Point(point=(), layer=1, description='left collar of the tshirt'),
                'spine_top': Point(point=(), layer=1, description='top of the spine of the tshirt'),
                'right_collar': Point(point=(), layer=1, description='right collar of the tshirt'),
                'right_shoulder_top': Point(point=(), layer=1, description='right shoulder of the tshirt'),
                'right_sleeve_top': Point(point=(), layer=1, description='top corner of the right sleeve of the tshirt'),
                'right_sleeve_bottom': Point(point=(), layer=1, description='bottom corner of the right sleeve of the tshirt'),
                'right_armpit': Point(point=(), layer=1, description='right armpit of the tshirt'),
                'right_bottom': Point(point=(), layer=1, description='right bottom corner of the tshirt'),
                'left_bottom': Point(point=(), layer=1, description='left bottom corner of the tshirt'),
                'center_bottom': Point(point=(), layer=1, description='center point of left bottom and right bottom corners of the tshirt'),
                'left_armpit': Point(point=(), layer=1, description='left armpit of the tshirt'),
                'left_sleeve_bottom': Point(point=(), layer=1, description='bottom corner of the left sleeve of the tshirt'),
               'left_sleeve_top': Point(point=(), layer=1, description='top corner of the left sleeve of the tshirt'),
                'left_shoulder_top': Point(point=(), layer=1, description='left shoulder of the tshirt'),
            },
            'edges':{
                'layer_1':[
                    ('left_collar', 'spine_top'), ('spine_top', 'right_collar'), ('right_collar', 'right_shoulder_top'), ('right_shoulder_top', 'right_sleeve_top'), ('right_sleeve_top', 'right_sleeve_bottom'),
                   ('right_sleeve_bottom', 'right_armpit'), ('right_armpit', 'right_bottom'), ('right_bottom', 'center_bottom'), ('center_bottom', 'left_bottom'), ('left_bottom', 'left_armpit'),
                   ('left_armpit', 'left_sleeve_bottom'), ('left_sleeve_bottom', 'left_sleeve_top'), ('left_sleeve_top', 'left_shoulder_top'), ('left_shoulder_top', 'left_collar')
                ]}}
      ```
      Pant:
            ```python
            polygon = {
                'vertices':{
                    'left_top_waist': Point(point=(), layer=1, description='left top waist of the pant'),
                    'right_top_waist': Point(point=(), layer=1, description='right top waist of the pant'),
                    'top_waist_center': Point(point=(), layer=1, description='center point of left top waist and right top waist of the pant'),
                    'right_leg_bottom_right': Point(point=(), layer=1, description='right corner of the right leg of the pant'),
                    'right_leg_bottom_left': Point(point=(), layer=1, description='left corner of the right leg of the pant'),
                    'crotch': Point(point=(), layer=1, description='crotch of the pant'),
                    'left_leg_bottom_left': Point(point=(), layer=1, description='left corner of the left leg of the pant'),
                    'left_leg_bottom_right': Point(point=(), layer=1, description='right corner of the left leg of the pant'),
                }
                'edges':{
                    'layer_1':[
                        ('left_top_waist', 'top_waist_center'), ('top_waist_center', 'right_top_waist'), ('right_top_waist', 'right_leg_bottom_right'), ('right_leg_bottom_right',
'right_leg_bottom_left'), ('right_leg_bottom_left', 'crotch'), ('crotch', 'left_leg_bottom_right'), ('left_leg_bottom_right', 'left_leg_bottom_left'), ('left_leg_bottom_left', 'left_top_waist')
                    ]}}
            ```
####Input####:
1. type of the cloth object, for example: tshirt. Assume the cloth object is laid canonically on the table.
2. goal in the form of user language instructions, for example: `fold the left sleeve inwards into the center of the tshirt.`
3. polygon model dictionary representing current state (vertices, edges) of the cloth object.

####Output####:
You should output the subgoals in the form of language instructions. The output should be in the format of a python list like this:
```python
        subgoals = ['subgoal 1','subgoal 2', ...]
```
Each subgoal shall be in the format like this: fold <which part of the cloth> <direction, adverb, optional> to <which part of the cloth> <symmetrical axis, optional>.
You MUST only output the reasonable folding action, other static actions such as 'identifying <which position>' are not needed.
Remember the output shall be represented in this kind of python block and contains a list like this. Omit the reasoning output and only output the subgoals.
####Example####
####Example 1####
####Input####:
tshirt; fold both the sleeves into the center; polygon model:
``` TShirt:
        polygon = {vertices:{'spine_top': Point(point=(200, 30), layer=1, description='top of the spine of the tshirt'),
        'right_collar': Point(point=(220, 10), layer=1, description='right collar of the tshirt'),
        'right_shoulder_top': Point(point=(250, 30), layer=1, description='right shoulder of the tshirt'),
        'right_sleeve_top': Point(point=(300, 100), layer=1, description='top corner of the right sleeve of the tshirt'),
        'right_sleeve_bottom': Point(point=(280, 120), layer=1, description='bottom corner of the right sleeve of the tshirt'),
        'right_armpit': Point(point=(250, 110), layer=1, description='right armpit of the tshirt'),
        'right_bottom': Point(point=(250, 200), layer=1, description='right bottom corner of the tshirt'),
        'left_bottom': Point(point=(150, 200), layer=1, description='left bottom corner of the tshirt'),
        'left_armpit': Point(point=(150, 110), layer=1, description='left armpit of the tshirt'),
        'left_sleeve_bottom': Point(point=(120, 120), layer=1, description='bottom corner of the left sleeve of the tshirt'),
        'left_sleeve_top': Point(point=(100, 100), layer=1, description='top corner of the left sleeve of the tshirt'),
        'left_shoulder_top': Point(point=(150, 30), layer=1, description='left shoulder of the tshirt'),
        'left_collar': Point(point=(180, 10), layer=1, description='left collar of the tshirt'),
        },
       edges:{layer_1:[('spine_top','right_collar'),('right_collar','right_shoulder_top'),('right_shoulder_top','right_sleeve_top'),('right_sleeve_top','right_sleeve_bottom'),('right_sleeve_bottom','right_
       armpit'),('right_armpit','right_bottom'),('right_bottom','left_bottom'),('left_bottom','left_armpit'),('left_armpit','left_sleeve_bottom'),('left_sleeve_bottom','left_sleeve_top'),('left_sleeve_top',
       'left_shoulder_top'),('left_shoulder_top','left_collar'),('left_collar','spine_top'),],
        }}
   ```
####Output####:
```python
        subgoals = ["fold the left sleeve inwards to the center of the tshirt with the left shoulder-armpit line as the symmetrical axis", "fold the right sleeve inwards to the center of the
tshirt with the right shoulder-armpit line as the symmetrical axis"]
```
~~~

Fig. 22: Prompt for subgoal decomposition LLM $\mathbf{LLM_{sd}}$.



~~~
You are the brain of a bimanual robot arm like ABB YuMi Robot that listens to one given language instruction as your goal, and decides the foldline to fold the cloth object into certain configuration as the language describes.
There are four types of cloth that can be folded: square, rectangle, tshirt and pant. The cloth is represented as parameterized polygon model, which is made of vertices and edges. The vertices and edges are represented and stored in one python dictionary. A vertex is an object of user-defined python class `Point`, which has attributes of vertex coordinate, vertex detailed description, and also the layer of the vertex in the polygon model. The layer number indicates the 3D spatial height of the polygon model, where a higher layer represents being closer to the top of the cloth. All vertices are stored in a dictionary with the key of vertex name. An edge is a tuple of two vertices, which are the start and end vertices of the edge. All edges of the same layer number are stored in a list with the key of layer number. For a folded polygon model, there are vertices added to the original vertices, named as 'foldline_[fold_step]_[point_id]', which also has attributes of vertex coordinate, vertex detailed description of folding language description of this step, and also the layer of this vertex in the polygon model.
The initial templates are like this. Take t-shirt object for example:
   Tshirt:
        ```python
        polygon = {
            'vertices':{
                'left_collar': Point(point=(), layer=1, description='left collar of the tshirt'),
                'spine_top': Point(point=(), layer=1, description='top of the spine of the tshirt'),
                'right_collar': Point(point=(), layer=1, description='right collar of the tshirt'),
                'right_shoulder_top': Point(point=(), layer=1, description='right shoulder of the tshirt'),
                'right_sleeve_top': Point(point=(), layer=1, description='top corner of the right sleeve of the tshirt'),
                'right_sleeve_bottom': Point(point=(), layer=1, description='bottom corner of the right sleeve of the tshirt'),
                'right_armpit': Point(point=(), layer=1, description='right armpit of the tshirt'),
                'right_bottom': Point(point=(), layer=1, description='right bottom corner of the tshirt'),
                'left_bottom': Point(point=(), layer=1, description='left bottom corner of the tshirt'),
                'center_bottom': Point(point=(), layer=1, description='center point of left bottom and right bottom corners of the tshirt'),
                'left_armpit': Point(point=(), layer=1, description='left armpit of the tshirt'),
                'left_sleeve_bottom': Point(point=(), layer=1, description='bottom corner of the left sleeve of the tshirt'),
               'left_sleeve_top': Point(point=(), layer=1, description='top corner of the left sleeve of the tshirt'),
                'left_shoulder_top': Point(point=(), layer=1, description='left shoulder of the tshirt'),
            },
            'edges':{
                'layer_1':[
                    ('left_collar', 'spine_top'), ('spine_top', 'right_collar'), ('right_collar', 'right_shoulder_top'), ('right_shoulder_top', 'right_sleeve_top'), ('right_sleeve_top', 'right_sleeve_bottom'),
('right_sleeve_bottom', 'right_armpit'), ('right_armpit', 'right_bottom'), ('right_bottom', 'center_bottom'), ('center_bottom', 'left_bottom'), ('left_bottom', 'left_armpit'),                              ('left_armpit',
'left_sleeve_bottom'), ('left_sleeve_bottom', 'left_sleeve_top'), ('left_sleeve_top', 'left_shoulder_top'), ('left_shoulder_top', 'left_collar')
                ]}}
      ```

####Input####:
1. type of the cloth object, for example: tshirt.
2. goal in the form of user language instructions, for example: `fold the left sleeve inwards into the center of the tshirt.`
3. polygon model dictionary representing current state (vertices, edges) of the cloth object.

####Task####:
Your task is to calculate the foldline of the folding process to fold the cloth as the language instruction describes. Fold line is a directional vector from the start point to the end point. The segment of the polygon which is left of the foldline is folded over to the right part symmetrically.
There are TWO ways to calculate the fold line.
One way is denoted as `pick-and-place`, which means you can analyze this problem step by step. You can firstly infer a reasonable single-arm pick-and-place action to achieve the language instruction, in situations like language instructios containing  `fold from <which position> to <which position>` or `let <which position> meet <which position>`, or `align <which position> to <which position>`, etc. Then you can call our predefined function `calculate_perpendicular_bisector` to output the foldline.
The other way is denoted as `foldline`, which means you can directly output the foldline from geometric cues in the parametrized polygon model, in situations like `fold along <which line>`, or `fold with <which line> as the symmetrical axis`,
It is up to you to decide which way to choose based on the language instruction and the current polygon model description. If you think it is easier to reason about pick and place action and then calculate the foldline as the perpendicular bisector, you can choose the `pick-and-place` action representation. If you think it is easier to directly reason about the foldline, you can choose the `foldline` action representation.

####Output Format####:
Remember the output MUST be represented in python block and contains the python statement like the following lines. Your output MUST omit all of the reasoning process. If you output your reasoning process, you will be punished.
For `pick-and-place` way to infer the foldline, the output format should be like this:
 ```python
        pnp = {'line_start':(x,x), 'line_end':(x,x), 'line_type':'pick-and-place'}
        foldline = calculate_perpendicular_bisector(pnp, polygon)
```
For `foldline` way to infer the foldline, the output format should be like this:
 ```python
        foldline = {'line_start':(x,x), 'line_end':(x,x), 'line_type':'foldline'}
```
The output MUST starts with three ` and python. The output MUST ends with three `.

For example:
####Example 1####
####Input####: tshirt; fold the left sleeve of the shirt inwards into the center along the left armpit-shoulder line; polygon model:
        ``` TShirt:
        polygon = {vertices:{'spine_top': Point(point=(200, 30), layer=1, description='top of the spine of the tshirt'),
        'right_collar': Point(point=(220, 10), layer=1, description='right collar of the tshirt'),
        'right_shoulder_top': Point(point=(250, 30), layer=1, description='right shoulder of the tshirt'),
        'right_sleeve_top': Point(point=(300, 100), layer=1, description='top corner of the right sleeve of the tshirt'),
        'right_sleeve_bottom': Point(point=(280, 120), layer=1, description='bottom corner of the right sleeve of the tshirt'),
        'right_armpit': Point(point=(250, 110), layer=1, description='right armpit of the tshirt'),
        'right_bottom': Point(point=(250, 200), layer=1, description='right bottom corner of the tshirt'),
        'left_bottom': Point(point=(150, 200), layer=1, description='left bottom corner of the tshirt'),
        'left_armpit': Point(point=(150, 110), layer=1, description='left armpit of the tshirt'),
        'left_sleeve_bottom': Point(point=(120, 120), layer=1, description='bottom corner of the left sleeve of the tshirt'),
        'left_sleeve_top': Point(point=(100, 100), layer=1, description='top corner of the left sleeve of the tshirt'),
        'left_shoulder_top': Point(point=(150, 30), layer=1, description='left shoulder of the tshirt'),
        'left_collar': Point(point=(180, 10), layer=1, description='left collar of the tshirt'),
        },
      edges:{layer_1:[('spine_top','right_collar'),('right_collar','right_shoulder_top'),('right_shoulder_top','right_sleeve_top'),('right_sleeve_top','right_sleeve_bottom'),('right_sleeve_bottom','right_armpit'),('right_armpit',
      'right_bottom'),('right_bottom','left_bottom'),('left_bottom','left_armpit'),('left_armpit','left_sleeve_bottom'),('left_sleeve_bottom','left_sleeve_top'),('left_sleeve_top','left_shoulder_top'),('left_shoulder_top','left
      _collar'),('left_collar','spine_top'),],
        }}
     ```
####My Thoughts####: From the input language we know we need to fold the left sleeve of the tshirt inwards into the center along the left armpit-shoulder line. Analyzing the language instruction, we can easily infer the fold line from the sentence `along the left armpit-shoulder line`. Therefore we can choose the `foldline` representation, which is the vector from the left armpit (150, 110) to the left shoulder top (150, 30).
####Output####:
```python
        foldline = {'line_start':(150, 110), 'line_end':(150, 30), 'line_type':'foldline'}
```

####Example 2####
####Inputs####: square; fold the left top corner of the square into the center; polygon model:
        ```Square:
        polygon = {vertices:{'top_left': Point(point=(100, 100), layer=1, description='top left corner of the square'),
        'top_right': Point(point=(200, 100), layer=1, description='top right corner of the square'),
        'bottom_right': Point(point=(200, 200), layer=1, description='bottom right corner of the square'),
        'bottom_left': Point(point=(100, 200), layer=1, description='bottom left corner of the square'),
        },
        edges:{layer_1:[('top_left','top_right'),('top_right','bottom_right'),('bottom_right','bottom_left'),('bottom_left','top_left'),],
        }}
####My Thoughts####
From the input we know we need to fold the left top corner of the square into the center. As it contains explicit geometric cues of pick-and-place action `from left top corner into the center`, it is easy to firstly reason the pick-and-place action, which is picking from left top corner and placing on the center of the square. Then we can calculate the perpendicular bisector of this pick-and-place line as the foldline.
####Output####:
```python
        pnp = {'line_start':(100,100), 'line_end':(150,150), 'line_type':'pick-and-place'}
        foldline = calculate_perpendicular_bisector(pnp, polygon)
```
~~~

Fig. 23: Prompt for symmetrical fold line generation LLM $\mathbf{LLM_{fg}}$.

TABLE VI: **Details of evaluated cloth folding tasks (single-step tasks). Task name annotated with _ _ _ _ belongs to seen tasks for baseline training, while the rest are unseen tasks for baselines. All tasks are unseen tasks for our proposed PolyFold.**

| Task Type | Task Numbers | Sub-Task Name | Sub-Task Description |
|---|---|---|---|
| S-Corner-Folding | 4 | S-Corner-Lefttop-Middle | Fold the left top corner of the square cloth into the center. |
| | | S-Corner-Righttop-Middle | Fold the right top corner of the square cloth into the center. |
| | | S-Corner-Leftbottom-Middle | Fold the left bottom corner of the square cloth into the center. |
| | | S-Corner-Rightbottom-Middle | Fold the right bottom corner of the square cloth into the center. |
| S-Triangle-Folding | 4 | S-Triangle-Lefttop-Rightbottom | Fold the left top corner of the square cloth to the right bottom corner. |
| | | S-Triangle-Righttop-Leftbottom | Fold the right top corner of the square cloth to the left bottom corner. |
| | | S-Triangle-Rightbottom-Lefttop | Fold the right bottom corner of the square cloth to the left top corner. |
| | | S-Triangle-Leftbottom-Righttop | Fold the left bottom corner of the square cloth to the right top corner. |
| R-Edge-to-Middle-Folding | 4 | R-Edge-Top-Middle | Fold the top edge of the rectangular cloth to the middle. |
| | | R-Edge-Bottom-Middle | Fold the bottom edge of the rectangular cloth to the middle. |
| | | R-Edge-Left-Middle | Fold the left edge of the rectangular cloth to the middle. |
| | | R-Edge-Right-Middle | Fold the right edge of the rectangular cloth to the middle. |
| R-Edge-to-Opposite-Folding | 4 | R-Edge-Top-Bottom | Fold the top edge of the rectangular cloth to the bottom edge. |
| | | R-Edge-Bottom-Top | Fold the bottom edge of the rectangular cloth to the top edge. |
| | | R-Edge-Left-Right | Fold the left edge of the rectangular cloth to the right edge. |
| | | R-Edge-Right-Left | Fold the right edge of the rectangular cloth to the left edge. |
| T-Sleeve-Folding | 4 | T-Sleeve-Left-Inwards | Fold the left sleeve of the t-shirt into the center along the left armpit-shoulder line. |
| | | T-Sleeve-Right-Inwards | Fold the right sleeve of the t-shirt into the center along the right armpit-shoulder line. |
| | | T-Sleeve-Left-Half | Fold the left sleeve of the t-shirt in half. |
| | | T-Sleeve-Right-Half | Fold the right sleeve of the t-shirt in half. |
| T-Half-Folding | 3 | T-Half-Left-Right | Fold the t-shirt in half horizontally from left to right. |
| | | T-Half-Right-Left | Fold the t-shirt in half horizontally from right to left. |
| | | T-Half-Bottom-Top | Fold the t-shirt in half vertically from bottom to top. |
| P-Half-Folding | 4 | P-Half-Left-Right | Fold the pant in half horizontally from left to right. |
| | | P-Half-Right-Left | Fold the pant in half horizontally from right to left. |
| | | P-Half-Leg-Left | Fold the left leg of the pant in half from bottom to top. |
| | | P-Half-Leg-Right | Fold the right leg of the pant in half from bottom to top. |
| Total | 27 | | |

TABLE VII: **Details of evaluated cloth folding tasks (multi-step tasks). Task name annotated with _ _ _ belongs to seen tasks for baseline training, while the rest are unseen tasks for baselines. All tasks are unseen tasks for our proposed PolyFold.**

| Task Type | Task Numbers | Sub-Task Name | Sub-Task Description |
|---|---|---|---|
| S-Corner-Folding | 7 | S-Corner-Lefttop-Righttop-Middle | Fold the left top and right top corners of the square cloth into the center. |
| | | S-Corner-Lefttop-Rightbottom-Middle | Fold the left top and right bottom corners of the square cloth into the center. |
| | | S-Corner-Lefttop-Leftbottom-Middle | Fold the left top and left bottom corners of the square cloth into the center. |
| | | S-Corner-Righttop-Rightbottom-Middle | Fold the right top and right bottom corners of the square cloth into the center. |
| | | S-Corner-Righttop-Leftbottom-Middle | Fold the right top and left bottom corners of the square cloth into the center. |
| | | S-Corner-Rightbottom-Leftbottom-Middle | Fold the right bottom and left bottom corners of the square cloth into the center. |
| | | S-Corner-All-Middle | Fold all four corners of the square cloth into the center. |
| S-Triangle-Folding | 8 | S-Triangle-Lefttop-Rightbottom-Leftbottom-Righttop | Fold the left top corner of the square cloth to the right bottom corner and then fold the left bottom corner to the right top corner. |
| | | S-Triangle-Lefttop-Rightbottom-Righttop-Leftbottom | Fold the left top corner of the square cloth to the right bottom corner and then fold the right top corner to the left bottom corner. |
| | | S-Triangle-Righttop-Leftbottom-Lefttop-Rightbottom | Fold the right top corner of the square cloth to the left bottom corner and then fold the left top corner to the right bottom corner. |
| | | S-Triangle-Righttop-Leftbottom-Rightbottom-Lefttop | Fold the right top corner of the square cloth to the left bottom corner and then fold the right bottom corner to the left top corner. |
| | | S-Triangle-Rightbottom-Lefttop-Leftbottom-Righttop | Fold the right bottom corner of the square cloth to the left top corner and then fold the left bottom corner to the right top corner. |
| | | S-Triangle-Rightbottom-Lefttop-Righttop-Leftbottom | Fold the right bottom corner of the square cloth to the left top corner and then fold the right top corner to the left bottom corner. |
| | | S-Triangle-Leftbottom-Righttop-Lefttop-Rightbottom | Fold the left bottom corner of the square cloth to the right top corner and then fold the left top corner to the right bottom corner. |
| | | S-Triangle-Leftbottom-Righttop-Rightbottom-Lefttop | Fold the left bottom corner of the square cloth to the right top corner and then fold the right bottom corner to the left top corner. |
| R-Edge-to-Middle-Folding | 4 | R-Edge-Top-Middle-Bottom-Middle | Fold the top edge of the rectangular cloth to the middle and fold the bottom edge to the middle. |
| | | R-Edge-Bottom-Middle-Top-Middle | Fold the bottom edge of the rectangular cloth to the middle and fold the top edge to the middle. |
| | | R-Edge-Left-Middle-Right-Middle | Fold the left edge of the rectangular cloth to the middle and fold the right edge to the middle. |
| | | R-Edge-Right-Middle-Left-Middle | Fold the right edge of the rectangular cloth to the middle and fold the left edge to the middle. |
| R-Edge-to-Opposite-Folding | 8 | R-Edge-Top-Bottom-Left-Right | Fold the top edge of the rectangular cloth to bottom edge, then fold the left edge to right edge. |
| | | R-Edge-Top-Bottom-Right-Left | Fold the top edge of the rectangular cloth to bottom edge, then fold the right edge to left edge. |
| | | R-Edge-Bottom-Top-Left-Right | Fold the bottom edge of the rectangular cloth to top edge, then fold the left edge to right edge. |
| | | R-Edge-Bottom-Top-Right-Left | Fold the bottom edge of the rectangular cloth to the top edge and then fold the right edge to the left edge. |
| | | R-Edge-Left-Right-Top-Bottom | Fold the left edge of the rectangular cloth to the right edge and then fold the top edge to the bottom edge. |
| | | R-Edge-Left-Right-Bottom-Top | Fold the left edge of the rectangular cloth to the right edge and then fold the bottom edge to the top edge. |

| | | | |
|---|---|---|---|
| | | R-Edge-Right-Left-Top-Bottom | Fold the right edge of the rectangular cloth to the left edge and then fold the top edge to the bottom edge. |
| | | R-Edge-Right-Left-Bottom-Top | Fold the right edge of the rectangular cloth to the left edge and then fold the bottom edge to the top edge. |
| T-Sleeve-Folding | 8 | T-Sleeve-Left-Right-Inwards | Fold the left sleeve of the t-shirt into the center along the left armpit-shoulder line and then fold the right sleeve inwards along the right armpit-shoulder line. |
| | | T-Sleeve-Right-Left-Inwards | Fold the right sleeve of the t-shirt into the center along the right armpit-shoulder line and then fold the left sleeve inwards along the left armpit-shoulder line. |
| | | T-Sleeve-Left-Half-Right-Half | Fold the left sleeve of the t-shirt in half and then fold the right sleeve in half. |
| | | T-Sleeve-Right-Half-Left-Half | Fold the right sleeve of the t-shirt in half and then fold the left sleeve in half. |
| | | T-Sleeve-Left-Half-Inwards | Fold the left sleeve of the t-shirt in half and then fold the left sleeve inwards along the left armpit-shoulder line. |
| | | T-Sleeve-Right-Half-Inwards | Fold the right sleeve of the t-shirt in half and then fold the right sleeve inwards along the right armpit-shoulder line. |
| | | T-Sleeve-Left-Half-Inwards-Right-Half-Inwards | Fold the left sleeve of the t-shirt in half, fold the left sleeve inwards, fold the right sleeve in half and then fold the right sleeve inwards. |
| | | T-Sleeve-Right-Half-Inwards-Left-Half-Inwards | Fold the right sleeve of the t-shirt in half, fold the right sleeve inwards, fold the left sleeve in half and then fold the left sleeve inwards. |
| T-Block-Folding | 2 | T-Block-Left-Right-Bottom-Top | Fold the left sleeve of the t-shirt inwards, fold the right sleeve inwards, and then fold the t-shirt in half vertically from bottom to top. |
| | | T-Block-Right-Left-Bottom-Top | Fold the right sleeve of the t-shirt inwards, fold the left sleeve inwards, and then fold the t-shirt in half vertically from bottom to top. |
| P-Half-Folding | 2 | P-Half-Leg-Left-Right | Fold the left leg of the pant in half vertically from bottom to top and then fold the right leg in half vertically from bottom to top. |
| | | P-Half-Leg-Right-Left | Fold the right leg of the pant in half vertically from bottom to top and then fold the left leg in half vertically from bottom to top. |
| P-Block-Folding | 4 | P-Block-Left-Right-Bottom-Top | Fold the pant in half horizontally from left to right and then fold it in half vertically from bottom to top. |
| | | P-Block-Right-Left-Bottom-Top | Fold the pant in half horizontally from right to left and then fold it in half vertically from bottom to top. |
| | | P-Block-Left-Right-Top-Bottom | Fold the pant in half horizontally from left to right and then fold it in half vertically from top to bottom. |
| | | P-Block-Right-Left-Top-Bottom | Fold the pant in half horizontally from right to left and then fold it in half vertically from top to bottom. |
| Total | 43 | | |