Supplemental Material

A. Prompt Design

A.1. Annotate Prompt

Control a robot to perform manipulation tasks based on an image of a single object with marked keypoints and a text instruction. The goal is to list the possible uses of the object and all key primitives in each stage.
Object Analysis
- Determine how many important parts of the object based on the usage and image. For example: - ”Important parts of the Teapot”:
    - Body: The main container for holding the tea or water.
    - Lid: A cover for the opening at the top of the teapot to prevent spillage and retain heat.
    - Handle: A grip for holding and pouring the teapot, often designed to insulate from heat.
    - Spout: A narrow outlet for pouring the liquid from the teapot.
    - Base: The bottom support of the teapot that ensures stability and contact with the surface.

Possible Uses
- According to the important parts above, determine how many usages of the object in image. For example:
  - ”Possible Uses of the Teapot”:
    - Pouring tea from the teapot.
    - Filling the teapot with water.
    - Place the fallen teapot upright.

Task Stages
- Break down each use into stages. For example:
  - ”pouring tea from teapot”:
    - 3 stages: ”grasp teapot”, ”align teapot with cup opening”, and ”pour liquid”
  - ”put red block on top of blue block”:
    - 2 stages: ”grasp red block”, ”drop the red block on top of blue block”
  - ”reorient bouquet and drop it upright into vase”:
    - 3 stages: ”grasp bouquet”, ”upright bouquet”, and ”align the reoriented bouquet with vase”
  - ”open microwave door to heat food”:
    - 6 stages: ”grasp microwave door handle”, ”pull door open along hinge axis”, ”grasp food”, ”place food container on turntable”, ”close door along hinge axis” and ”press start button”

Key Primitive Definitions
- List all candidate key primitives involved in each stage categorized as:
    - *Main*: The reference point and principal orientation axis of the object, serving as the basis for defining its spatial configuration. (e.g. Center of the teapot body + vertical axis.)
    - *Anchor*: A specific point and axis used as a reference for defining the object’s pose during movement or ensuring keypoint constraints are met at the end of a substage (e.g., Tip of the spout + pouring direction.).
    - *Grasp*: The position and orientation of the end-effector when securely holding the object (e.g., Center of the handle + approach direction for grasping).
    - *Actuation*: The position and orientation of the end-effector required to trigger mechanical operations, such as pressing, rotating, or toggling components (e.g., Center of the heating button + pressing direction).
    - *Hinge*: The position and orientation of the end-effector used to manipulate articulated objects, typically around a rotational axis (e.g., Center of the lid hinge + rotation axis direction).

- *Actuation*: The position and orientation of the end-effector required to trigger mechanical operations, such as pressing, rotating, or toggling components (e.g., Center of the heating button + pressing direction).
- *Hinge*: The position and orientation of the end-effector used to manipulate articulated objects, typically around a rotational axis (e.g., Center of the lid hinge + rotation axis direction).

Key Primitive Example
- Define key primitives for each stage. For example:
key_primitives = [
{Type: Main, Pos: [x, y, z], Orientation: [dx, dy, dz], Stage: "Pour Liquid", Description: "Global reference for maintaining proper pouring orientation"},
{Type: Grasp, Pos: [x, y, z], Orientation: [dx, dy, dz], Stage: "Grasp Teapot", Description: "Grasping the teapot handle for secure hold"},
{Type: Anchor, Pos: [x, y, z], Orientation: [dx, dy, dz], Stage: "Align Teapot with Cup Opening", Description: "Reference point and axis for positioning the teapot relative to the cup"}
# Add more key primitive if needed]

Note:
- You do not need to consider collision avoidance. Focus on what is necessary to complete the task.
- *List all possible options thoroughly* when there are multiple reasonable candidate primitives.
- If the object has multiple possible uses, try to cover as many usage scenarios as possible, rather than just the most obvious operation.
- When you annotate a point, you MUST label its corresponding orientation; vice versa.
- Prioritize vectors based on the object's skeleton or intrinsic orientation, and then consider vectors representing orientations that are not directly visible in the image.
- Do not contain the specific id of key points in the your output.
- The initial orientation may affect grasping points, annotate all available primitives. For example, if handles are unusable, consider edges for grasping.
- Do not use '...' for omission in your output, list all usages and possibles.
- If the task involves multiple sub-operations, ensure that all stages are clearly labeled and avoid omitting any possible intermediate steps (such as rotation, alignment, etc.)
- List all possible uses of the object, ensuring diversity and avoiding repetitive scenarios by considering:
    - Interaction with typical objects (how the object interacts with other common objects in manipulation).
    - Position or orientation adjustment (how the object's position or orientation is adjusted for a specific purpose).
    - Functional Demonstration (describe tasks that showcase the object's functionality, such as pressing buttons, turning knobs, or activating features to achieve a specific outcome.).
- Pay attention to identifying the **Actuation Point** of any special buttons or knobs. For example, use the button to open the lid before heating a liquid (or filling with water).
    - There are two sub-steps(locate the button then press the button) and two key primitives(button center area and press orientation).
    - If there are multiple buttons, please make sure to distinguish between them.(eg. the heating button or the lid opening button)
- Special consideration for Actuation (execution) points: Differentiate between various types of control objects such as buttons, knobs, sliders, and other mechanical interfaces.
- In this stage, you don't need to fill in the specific value in Pos and Orientation, just let it be [x, y, z] and [dx, dy, dz]

Structure of python code block as follows:
List all important parts of the object based on the usage and image

A.2. Alignment Prompt

You are provided with semantic annotations of an object in the image and need to map these annotations to the corresponding primitives detected in the image. Your goal is to generate a JSON annotation result that aligns the semantic labels with the detected keypoints based on the following rules:

The Definition of Principal Orientation
- The principal orientation (X, Y, Z) are marked in different colors in the image. Ensure that you match the following colors to the corresponding axes:
    - X Axis: Red, +x: [1, 0, 0]; -x: [-1, 0, 0]
    - Y Axis: Green, +y: [0, 1, 0]; -y: [0, -1, 0]
    - Z Axis: Blue, +z: [0, 0, 1]; -z: [0, 0, -1]

Key Primitive Mapping
- If a key position is correctly identified, include the corresponding point in the ”points” attribute.
- Append a probability value to each point in the format {”pos_ID”: n, ”pos_Probability”: p}.

For example:

{”pos_ID”: 1, ”pos_Probability”: 0.75},
{”pos_ID”: 2, ”pos_Probability”: 0.85}.
- If multiple points represent the same position, list them all.
- If a key orientation is correctly identified, represent it with following two ways:
    - Using Principal Orientation Labels for standard orientations (e.g., horizontal/vertical): [0, 0, 1], [1, 0, 0]. For example, [0, 0, 1] means orientation towards the positive z-axis.
    - Using Point-based orientation for object-specific orientations (e.g., spout orientation): [x, y] (x, y is the ID of the keypoints, and the orientation is from point X to point)
- Append a probability value to the orientation in the format {”ori_ID”: [x, y], ”ori_Probability”: p} or {”ori_ID”: [0, 0, 1], ”ori_Probability”: p}. For example: {”ori_ID”: [1, 2], ”ori_Probability”: 0.7}, {”ori_ID”: [0, 0, 1], ”ori_Probability”: 1.0}. If there are multiple pairs for the same orientation, list them all.

- Probability values indicate the confidence level of the point/orientation being correctly identified:
    $>$ 0.5: The point/orientation is worth considering but may need verification.
    $>$ 0.8: The point/orientation is highly accurate and can be trusted.
- Handling Missing or Erroneous Annotations:
    - Use ”None” if the keypoint/orientation is not visible in the viewpoints.
    - Use ”Error” if the keypoint/orientation is visible in the viewpoints, but no points annotated. (Re-detect is required)

Note:
- Exclude Undetected Points: Do not include points that are not present in the image.
- Output Format: Only output the final JSON result without any additional text.
- The initial orientation may affect grasping points, annotate all available positions. For example, if handles are unusable, consider edges for grasping.
- Identify and list as many effective points as possible.
- The point numbers are continuous and consistent across different viewpoints. Match the correct points and vectors by aligning the point numbers across different perspectives.
- ”ori_ID”: [x, y] (towards the target) and ”ori_ID”: [y, x] (away from the target).
- Approach Orientation represents the end-effector movement towards the Grasp Point or Actuation Point. Ensure orientation (e.g., ”ori_ID”: [x, y] or ”ori_ID”: [0, 0, -1]) point towards the target, not away from it. - For example, USE ”ori_ID”: [x, y] (towards the handle/neck), NOT ”ori_ID”: [y, x] (away from the handle/neck), to avoid moving the end-effector away from the manipulation point.
- In this stage, you don’t need to fill in the specific value in Pos and Orientation, just let it be [x, y, z] and [dx, dy, dz]

if handles are unusable, consider edges for grasping.
- Identify and list as many effective points as possible.
- The point numbers are continuous and consistent across different viewpoints. Match the correct points and vectors by aligning the point numbers across different perspectives.
- ”ori_ID”: [x, y] (towards the target) and ”ori_ID”: [y, x] (away from the target).
- Approach Orientation represents the end-effector movement towards the Grasp Point or Actuation Point. Ensure orientation (e.g., ”ori_ID”: [x, y] or ”ori_ID”: [0, 0, -1]) point towards the target, not away from it. - For example, USE ”ori_ID”: [x, y] (towards the handle/neck), NOT ”ori_ID”: [y, x] (away from the handle/neck), to avoid moving the end-effector away from the manipulation point.
- In this stage, you don’t need to fill in the specific value in Pos and Orientation, just let it be [x, y, z] and [dx, dy, dz]

Output Format
- Organize the annotations by type and format them in the required JSON structure:
JSON format your output for task description and key primitives

```
{
    ”Grasp”: [
        {
            ”Stage”: ”Grasp Teapot”,
            ”pos_ID”: n,
            ”pos_Probability”: p_1,
            ”ori_ID”: [x, y],
            ”ori_Probability”: p_2
            ”Pos”: [x, y, z],
            ”Orientation”: [dx, dy, dz],
            ”Description”: ”Grasping the teapot handle for secure hold”
        },
    ],
    ”Anchor”: [
        {
            ”Stage”: ”Align Teapot with Cup Opening”,
            ”pos_ID”: n,
            ”pos_Probability”: p_1,
            ”ori_ID”: [0, 0, 1],
            ”ori_Probability”: p_2,
            ”Pos”: [x, y, z],
            ”Orientation”: [dx, dy, dz],
            ”Description”: ”Reference point and axis for positioning the teapot relative to cup”
        },
    ]
    ”Hinge”: [
        {
            ”Stage”: ”Open Lid”,
            ”pos_ID”: n,
            ”pos_Probability”: p_1,
            ”ori_ID”: [0, 0, 1],
            ”ori_Probability”: p_2
            ”Pos”: [x, y, z],
            ”Orientation”: [dx, dy, dz],
            ”Description”: ”Rotation center and axis for opening the lid”
        }
    ]
}
```

B. Implementation Details

B.1. Manipulation Task Evaluation

To validate the effectiveness of PASG in robotic manipulation, we conduct comprehensive evaluations using the RoboTwin[38] simulation platform, open-source environment designed to emulate realistic robotic manipulation scenarios. RoboTwin provides standardized benchmarks that ensure both reproducibility and practical relevance.

Task Setting We evaluate PASG's performance across six representative manipulation tasks, including tasks requiring dual-arm coordination and single-arm manipulations, as well as interactions within cluttered environments. Specifically, the tasks are: (1) Block Hammer Beat, (2) Container Place, (3) Dual Bottles Pick, (4) Empty Cup Place, (5) Pick Apple, and (6) Messy Shoe Place.(see Fig. 6 for visual illustrations and difficulty categories).

To provide a better understanding of these tasks, we briefly describe each one below, accompanied by an illustrative figure showcasing the robot's interactions. The tasks are categorized by difficulty levels: hard, medium, and easy, highlighting the range of challenges faced by PASG.

- **Messy Shoe Place:** The robot needs to grasp a shoe with one hand. The main difficulty lies in the fact that the shoe's width is nearly identical to the width of the gripper, resulting in a low tolerance for error.
- **Dual Bottles Pick:** This task requires dual-arm coordination to simultaneously pick up two bottles and maintain balance while lifting them. It is a challenging task due to the necessity for precise synchronization between the two arms.
- **Block Hammer Beat:** The robot uses a hammer to hit a block at a predefined position. This task evaluates fine motor skills and the ability to manipulate tools effectively.
- **Container Place:** In this task, the robot picks up a container and places it accurately at a designated location. It is relatively easy because of the simple geometry of the object.
- **Empty Cup Place:** The robot must handle a fragile and lightweight object (a cup) and place it gently. The challenge is to maintain stability while carefully avoiding excessive force.
- **Pick Apple:** The robot is required to grasp a round and slippery object (an apple) from a cluttered environment. This is a medium-difficulty task that demands dexterity and precise perception.

Automated Labeling Consistency with Human Usage
The two images in Figure 7 illustrate PASG's ability to autonomously label tools in a way that aligns with human usage conventions. In the case of the bottle, the labeled grasping points are distributed evenly around its circular body, which ensures stability and effectiveness in gripping. For the hammer, the annotated functional points are concentrated on the hammerhead, which is the intended area for striking. These results highlight PASG's comprehensiveness and its capability to capture both the functional and practical aspects of tools, closely mirroring human understanding of their usage.

C. Additional Experiments

C.1. Reliability of VLM-Based Semantic Alignment

To assess the robustness of our VLM-based annotation pipeline, we conducted a detailed hallucination analysis over 844 annotations generated by GPT-4o across 100 diverse objects. We categorize potential annotation errors into four types: (1) invalid label indices, (2) semantically unreasonable suggestions, (3) missing functional labels, and (4) output format violations.

Metric	Invalid Label Index	Unreasonable Interaction	Missing Functional Labels	Format Violation	Total
Error Rate	0.40% ± 0.56%	6.67% ± 0.46%	0.80% ± 0.01%	0.00% ± 0.00%	844

Table 4. Error types in semantic annotations across 100 sampled objects (844 outputs).

As shown in Table 4, GPT-4o maintains a low overall hallucination rate, with most errors falling into the category of semantically unreasonable but still structurally valid suggestions. Notably, there were no format violations, validating its strong instruction-following reliability for structured annotation generation.

C.2. Expanded Human Verification of Annotations

To further validate the annotation quality, we expanded our human verification from 50 to 200 objects and introduced more granular metrics. Each object was independently verified by three annotators.

Metric	Accuracy				Completeness		Diversity
	Keypoint	Direction	Consistency	Object-level*	Positional	Functional	Usage
Success Rate	98.70% ± 0.45%	98.39% ± 0.59%	92.69% ± 1.15%	72.28% ± 2.45%	92.52% ± 3.43%	97.00% ± 0.04%	97.00% ± 0.04%

Table 5. Expanded human verification results on 200 objects. Object-level accuracy considers a failure in any sub-metric as incorrect.

These results confirm the annotation pipeline's effectiveness and robustness, supporting the integration of PASG-generated annotations in real manipulation settings.

D. Author Contribution

Code Implementation Zhihao Zhu implemented the foundational PASG framework and the automated pipeline. SiYu Pan built upon this with extensive optimizations, including prompt optimization and updates to the recognition backbone, helping to refine the automated pipeline. Yifan Zheng completed the benchmark design, model deployment, and testing code, as well as fine-tuning models. SiYu Pan and Zhihao Zhu implemented the adaptation of PASG

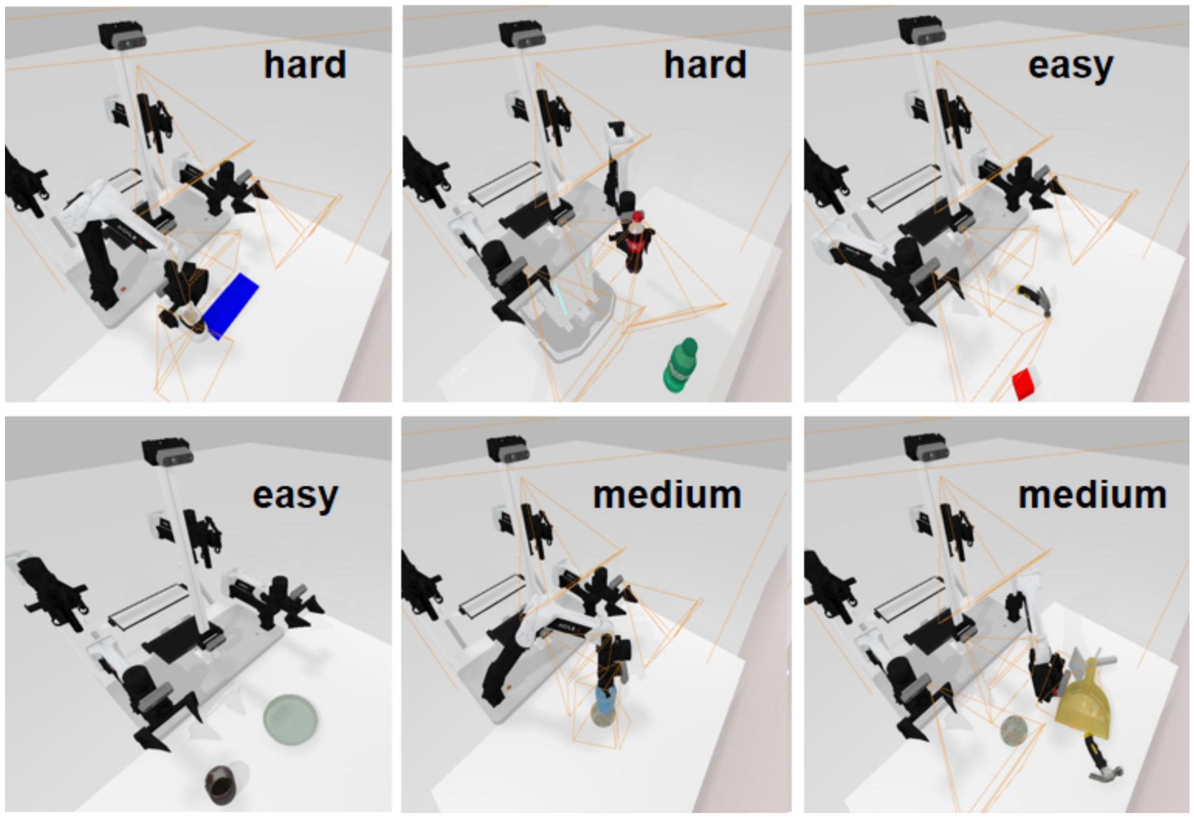

Figure 6. Illustrations of the six representative manipulation tasks evaluated in PASG. From left to right and top to bottom, the tasks are: Messy Shoe Place, Dual Bottles Pick, Block Hammer Beat, Container Place, Empty Cup Place, and Pick Apple. The tasks are categorized by difficulty levels: the first two tasks are hard, the middle two are easy, and the final two are medium.

in simulation scenarios and constructed the testing pipeline code. The code of Robotwin, SoM, Robocasa and LLaMA-Factory accelerated the implementation.

Paper writing Zhihao Zhu and Yao Mu finished introduction and methodology sections of the paper. Yifan Zheng and Zhihao Zhu wrote the experiments section. Zhihao Zhu provided all the visualizations shown in the paper. Yifan Zheng and Zhihao Zhu added results and analysis for their corresponding parts. Yao Mu carefully reviewed and revised the paper and gave feedback. Other authors help proofread and provide feedbacks.

Experiments Yifan Zheng led the design and construction of the benchmark, and completed experiments on model fine-tuning and performance evaluation. Zhihao Zhu and SiYu Pan co-led the performance evaluation experiments in simulation scenarios, and also conducted experiments on analysis and visualization.

Yao Mu is the main advisor of this project.

E. Limitation

While advantageous, PASG also has limitations. First, semantic annotations may inherit hallucinations from GPT-4o, despite our multi-round iterative refinement strategy. This process incurs significant computational costs, motivating future exploration of lightweight verification mechanisms (e.g., human-in-the-loop validation or physics-guided pruning) to enhance efficiency. Second, our framework mainly focuses on rigid objects, whereas deformable or articulated objects (e.g., hinged tools) also exhibit critical interaction primitives. Extending PASG requires redefining and adapting primitive categories to accommodate dynamic shape variations. This remains an open challenge but aligns with our ongoing efforts to broaden applicability.

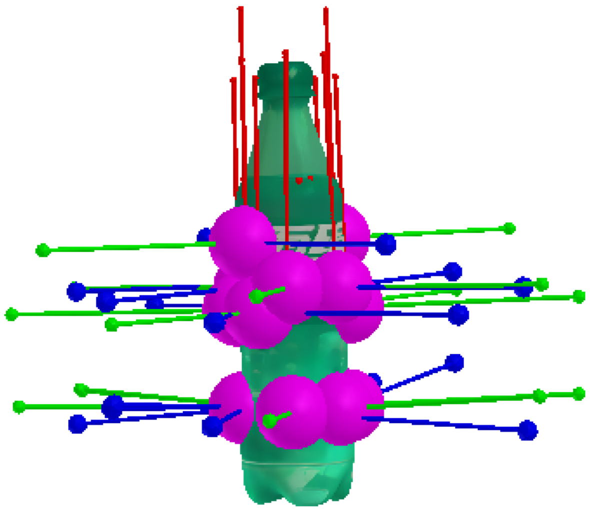
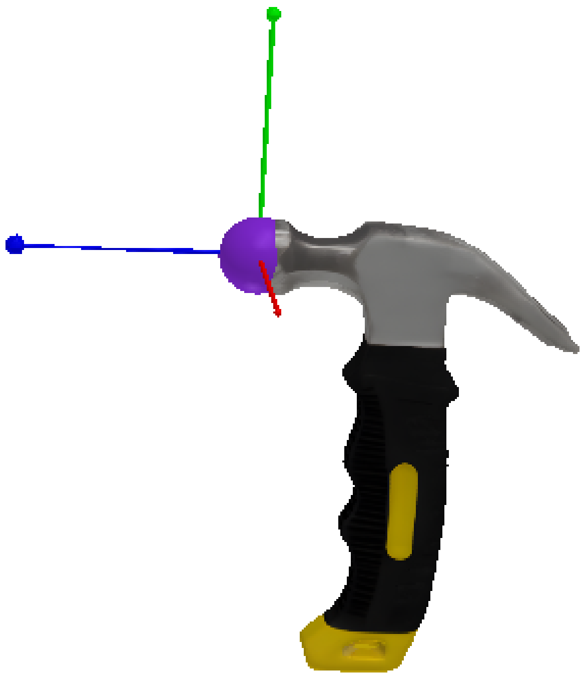

Figure 7. (Left) Automatically labeled grasping points on a bottle. The grasp points are distributed around the bottle's circular body, aligning with human intuition for stable and effective grasping. (Right) Automatically labeled functional points on a hammer. The points indicate the striking area at the hammerhead, which corresponds to its intended use for hitting objects. These annotations demonstrate PASG's capability to identify functional and practical areas of tools autonomously.