A. Appendix

A.1. Hardware Configuration

We detail our hardware setup in Figure 9, which centers on a Franka Emika Panda robotic arm, a 7-DOF manipulator known for its precision and torque-controlled movements. The standard parallel gripper is equipped with UMI fingers [6], which are flexible and can conform to objects of varying shapes, providing a more adaptable and secure grasp. Two Intel RealSense D415 RGB-D cameras are utilized for perception. One camera is mounted on the gripper, providing a first-person view from the robot's perspective, while the other is positioned opposite the robot to offer a third-person view of the workspace. This setup facilitates both detailed object perception and overall scene understanding. The system is powered by a workstation equipped with an Intel Core i7 processor, 64GB of RAM, and an NVIDIA RTX 4090 GPU, ensuring real-time inference and planning.

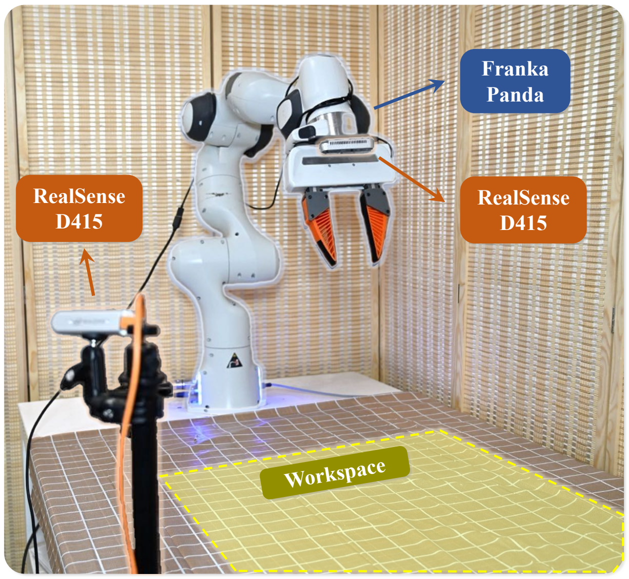


Figure 9. Hardware Configuration.

A.2. Prompt for Querying VLM

This paper uses GPT-4O from the OpenAI API as VLM. Below, we provide specific prompts and corresponding JSON schema to enable VLM to deliver structured outputs.

Task-relevant object grounding and stage partitioning. Takes task instructions and the object detection results from the VFM as inputs. Generates the task-relevant object IDs and the stage partitioning results (Section 3.1 of the main manuscript). The prompt is available for download here.

Grounding Interaction Point. Takes task instructions and object images processed by the SCAFFOLD visual prompting mechanism as inputs. Generates the IDs of task-relevant interaction points on the object (Section 3.2 of the main manuscript). The prompt is available for download here.

Captioning Interaction Direction. Takes object images with candidate interaction directions and descriptions of task-relevant object parts as inputs. Generates textual descriptions of the functionality associated with each candidate interaction direction (Section 3.2 of the main manuscript). The prompt is available for download here.

Determining Interaction Direction. Takes task instructions and textual descriptions of the functionality associated with candidate directions as inputs. Generates a ranking of the candidates according to the relevance between the directions and task instructions (Section 3.2 of the main manuscript). The prompt is available for download here.

Self-correction via RRC. Takes rendered images of the interaction results as inputs. Determines whether the interaction can be successful or if refinement is needed(Section 3.3 of the main manuscript). The prompt is available for download here.

A.3. Implementation Details of Method

Object Canonicalization. In this paper, we employs the single-view object generation model TripoSR to generate 3D object meshes $\mathcal{M}$ from single-view RGB images. However, such reconstructions inherently face ambiguities in scale, rotation, and translation. To resolve these ambiguities, we estimate the similarity transformation $\{s, \mathbf{R}, \mathbf{t}\}$ between the reconstructed mesh and the canonical object space using the observed point cloud $\mathcal{P}_{\text{obs}}$. First, the 6D object pose is estimated using Omni6DPose, and $\mathcal{P}_{\text{obs}}$ is transformed into the canonical point cloud $\mathcal{P}_{\text{can}}$. With known correspondences between $\mathcal{P}_{\text{can}}$ and the reconstructed mesh $\mathcal{M}$, the Umeyama [44] algorithm is applied to compute the similarity transformation. This transformation includes the scale factor $s \in \mathbb{R}^+$, the rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$, and the translation vector $\mathbf{t} \in \mathbb{R}^3$.

Functional Grasping. We apply special processing to the 'grasp' stage by directly generating primitives using universal grasp models [10, 45], without using RRC. Multiple grasp points on an object that meet the task requirements are obtained through [22]. For each point, a Gaussian distribution is generated and then superimposed, resulting in a continuous heatmap. This heatmap is used to post-process and filter the multiple grasp poses predicted by the grasp model, ultimately identifying the most suitable grasp pose.

Rendering Details of RRC. After obtaining the canonicalized object mesh and deriving the interaction vector and corresponding target pose as described in Section 3.2, we set the pose for all meshes and render them using Pyrender. Before rendering, we apply the method from [40] to inpaint the original areas of the rendered object in the image, preventing interference with the VLM.

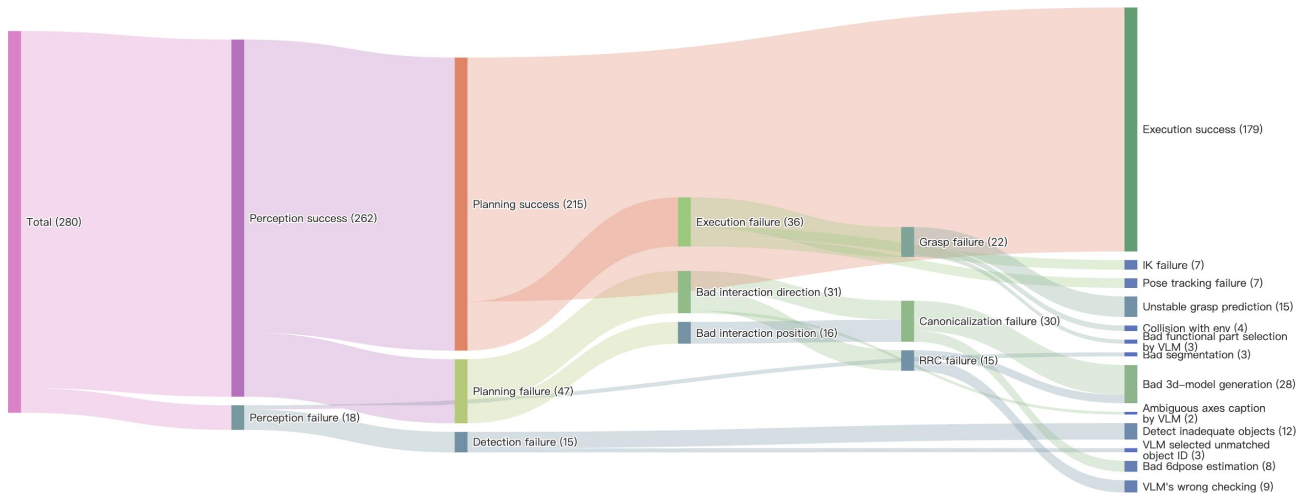


Figure 10. System error breakdown.

A.4. Action Primitives based on Constraints

We define the following atomic actions for VLM to select and complete constraint-based manipulation. (a) Grasp: Move to 8cm in front of the grasp pose, move forward and close the gripper. (b) Place: move to $v$cm in front of the target pose, move to the target pose and open the gripper. (c) Push: move to $v$cm in front of the target pose, close the gripper, and move to the target pose. (d) Pull: move to $v$cm in behind of the target pose (e) Rotate: rotate v degrees along the interaction vector. (f) Pour: move to the target point and then rotate around that point to achieve the desired orientation. v is the distance predicted by VLM. These atomic actions can be coupled with VLM to ahchive most everyday manipulation tasks.

A.5. System Error Breakdown

In this section, we conduct an empirical study by manually examining the failure cases of experiments in Table 1, calculating the likelihood of each module causing failures in the pipeline. The results are shown in Figure 10. Among the different modules, object canonicalization (3D AIGC and Pose estimation) is relatively prone to failure. Our qualitative analysis indicates that the main reason is the significant decline in the quality stability of [41] generation when small objects are positioned far from the camera. Therefore, it is recommended to use high-resolution cameras and capture images as close to the objects as possible. Additionally, the Grasping module and the VFM-based object grounding module also contribute to some errors. In contrast, the module that extracts interaction points and directions based on VLM is more stable, contributing less to failures. Lastly, factors such as unsolvable inverse kinematics (IK) for the robotic arm and collisions during execution also lead to some failure cases.

A.6. OmniManip for Demonstration Generation

We deployed OmniManip in Isaac Sim for autonomous demonstrations collection. Unlike prior methods reliant on task-specific privileged information, OmniManip collects demonstration trajectories for new tasks in a zero-shot manner, without needing task-specific details or prior object knowledge. To validate the effectiveness of OmniManip-generated data, we collected 150 trajectories per task to train behavior cloning policies [5]. Figure 11 illustrates the overall trajectory distribution for the tasks 'insert flower' and 'fit lid onto teapot', showcasing a notable diversity in the data. As shown in Table 4, the policy [5] trained using demonstration trajectories generated by OmniManip achieved an average accuracy of 86.93% across five tasks, which partially validates the quality of the demonstrations generated by OmniManip.

Task	**Success Rate**
Pick up cup on dish	95.24%
Recycle battery	91.30%
Insert pen into holder	86.36%
Fit lid onto teapot	79.16%
Insert flower into vase	82.61%
Total	86.93%

Table 4. Behavior cloning with demonstrations from OmniManip.

A.7. Comprehensive Limitation Analysis

Stability and Trade-offs in 3D-AIGC. OmniManip's closed-loop planning reduces sensitivity to various components, but 3D AIGC model quality remains crucial. Balancing stability and efficiency is key. To ensure fast inference

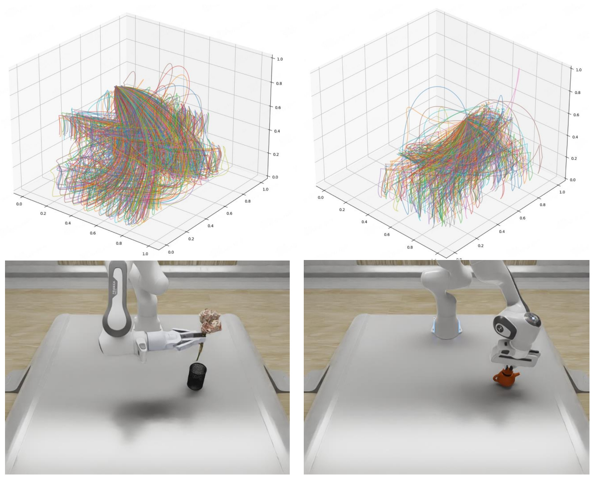


Figure 11. Trajectories visualization from OmniManip.

in real-world tests, we used [41] to generate a 3D mesh in under half a second. This efficient option can yield poor meshes with low resolution or unusual angles. For better results, users can choose higher-quality models like [29], though this increases computational time significantly.

Challenges in General Pose Estimation. General pose estimation [58] shows instability with certain rare objects, such as uncommon categories and transparent items. The estimated 6D pose at this point may not correctly align with the object's functional axis, reducing the sampling efficiency of OmniManip. However, the presence of RRC allows OmniManip to compensate to some extent.

Heavy VLM Calls. Multiple VLM calls average a cost of 11,000 tokens per plan, which represents a large computational overhead. However, most modules within the system can be executed and optimized in parallel to significantly enhance efficiency.

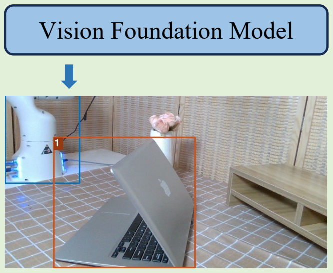
Complex Task Handling with OmniManip. It is challenging for OmniManip to handle complex tasks that are difficult to represent structurally, such as manipulating deformable objects. However, we believe that OmniManip, with its dual closed-loop features, continues to advance in the field of general object manipulation.

A.8. Inference Results Example

To provide a more detailed explanation of our approach, this section includes qualitative visualizations of three real-world tasks. These visualizations illustrate the key steps at each stage, such as identifying interaction points, providing textual descriptions of interaction directions and their corresponding task-related sequences, and showcasing the results of the self-correction phase. Each stage of our method is clearly and intuitively depicted. The tasks are: (a) Close lid of laptop (Figure 12) (b) Insert pen into holder (Figure 13) and (c) Pick up cup onto dish (Figure 14).

Instruction: Close lid of laptop

Task Stage Partitioning

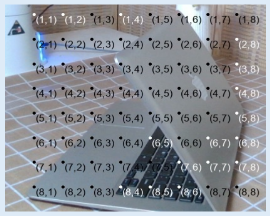


```
{
"Task Name": "Close lid of laptop",
"Think Step by Step": "To close the lid of the laptop, the gripper needs to push the top edge of the laptop lid downwards until it is fully closed.",
"objects": [
    {"object_ID": "laptop", "bbox_ID": 1},
    {"object_ID": "gripper", "bbox_ID": -1}
],
"stages": [
    {"action": "push", "active_object_ID": "gripper", "passive_object_ID": "laptop"}
]
}
"interaction_elements": [
    {"object_ID": "gripper", "element": {"action": "push", "type": "active", "interaction_part": "finger", "part_visible_and_tangible": True}},
    {"object_ID": "laptop", "element": {"action": "push", "type": "passive", "interaction_part": "lid", "part_visible_and_tangible": True}}]
}
```

Stage-1: Push

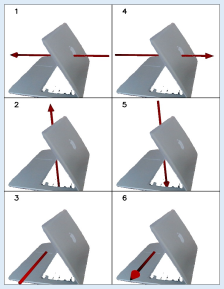
Passive Object (laptop)

Interaction point generation (visible_and_tangible)



```
{
    "Target Part": "base center",
    "Selected dot": [2,6]
}
```

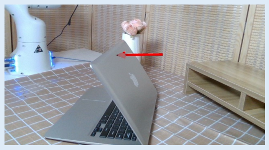
Interaction direction extraction



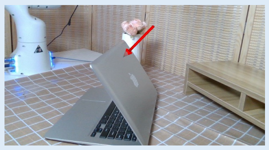
```
[
"1. The axis runs horizontally through the laptop, perpendicular to the screen, pointing outward from the top edge.",
"2. The axis is vertical, extending upward from the top edge of the laptop.",
"3. The axis runs horizontally along the keyboard, perpendicular to the screen, pointing away from the top edge.",
"4. The axis runs horizontally through the laptop, perpendicular to the screen, pointing inward towards the top edge.",
"5. The axis is vertical, extending downward from the top edge of the laptop.",
"6. The axis runs diagonally along the keyboard, pointing towards the top edge."
]

{
"expected interaction direction caption": "The axis runs horizontally through the laptop, perpendicular to the screen, pointing inward towards the top edge.",
"ranked directions": [1, 5, 4, 2, 3, 6]
}
```

RRC

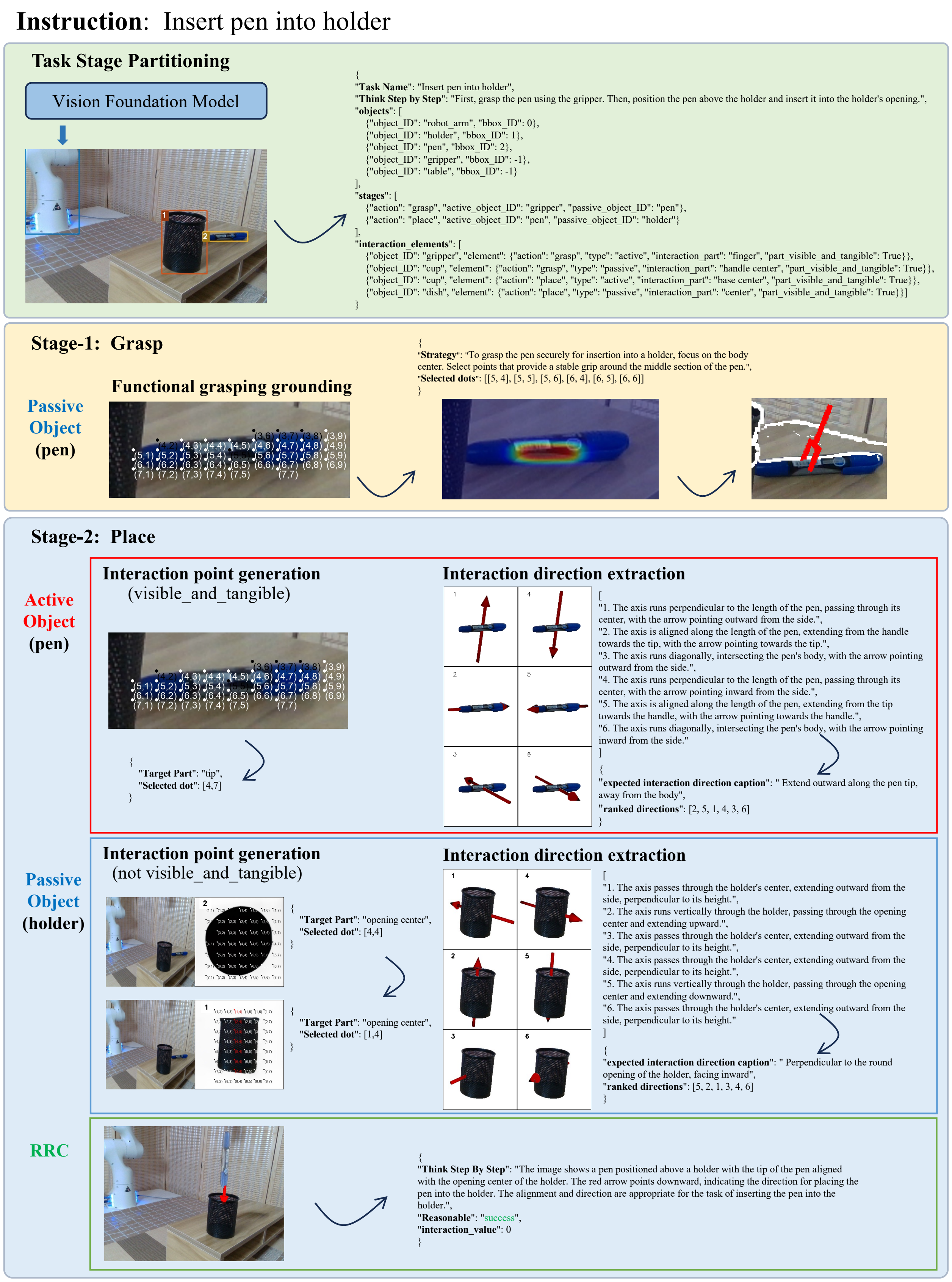


```
{
"Think Step By Step": "The image shows a laptop with its lid partially open. The red arrow is pointing horizontally towards the top edge of the laptop lid, indicating the direction to push the lid closed. The interaction part is the finger of the gripper, which should push the top edge of the laptop lid. The arrow is correctly aligned with the task of closing the lid, as it points directly towards the edge that needs to be pushed. However, the arrow is not perpendicular to the surface of the lid, which would be more effective for pushing it closed.",
"Reasonable": "refine",
"interaction_value": 0
}
```



```
{
"Think Step By Step": "The image shows a laptop with its lid partially open. The red arrow is pointing downward towards the top edge of the laptop lid, indicating the direction to push the lid closed. The arrow is aligned with the task of closing the laptop lid, as it suggests a downward motion which is appropriate for this task. The gripper's finger is positioned to interact with the top edge of the laptop lid, which is suitable for pushing it closed.",
"Reasonable": "success",
"interaction_value": 0
}
```

Figure 12. Result of task ‘Close lid of laptop’.


Instruction: Insert pen into holder
Task Stage Partitioning
Vision Foundation Model
Stage-1: Grasp
Passive Object (pen)
Functional grasping grounding
Stage-2: Place
Active Object (pen)
Interaction point generation (visible_and_tangible)
Interaction direction extraction
Passive Object (holder)
Interaction point generation (not visible_and_tangible)
Interaction direction extraction
RRC

```
{
"Task Name": "Insert pen into holder",
"Think Step by Step": "First, grasp the pen using the gripper. Then, position the pen above the holder and insert it into the holder's opening.",
"objects": [
   {"object_ID": "robot_arm", "bbox_ID": 0},
   {"object_ID": "holder", "bbox_ID": 1},
   {"object_ID": "pen", "bbox_ID": 2},
   {"object_ID": "gripper", "bbox_ID": -1},
   {"object_ID": "table", "bbox_ID": -1}
],
"stages": [
   {"action": "grasp", "active_object_ID": "gripper", "passive_object_ID": "pen"},
   {"action": "place", "active_object_ID": "pen", "passive_object_ID": "holder"}
],
"interaction_elements": [
   {"object_ID": "gripper", "element": {"action": "grasp", "type": "active", "interaction_part": "finger", "part_visible_and_tangible": True}},
   {"object_ID": "cup", "element": {"action": "grasp", "type": "passive", "interaction_part": "handle center", "part_visible_and_tangible": True}},
   {"object_ID": "cup", "element": {"action": "place", "type": "active", "interaction_part": "base center", "part_visible_and_tangible": True}},
   {"object_ID": "dish", "element": {"action": "place", "type": "passive", "interaction_part": "center", "part_visible_and_tangible": True}}]
}
```

```
{
"Strategy": "To grasp the pen securely for insertion into a holder, focus on the body
center. Select points that provide a stable grip around the middle section of the pen.",
"Selected dots": [[5, 4], [5, 5], [5, 6], [6, 4], [6, 5], [6, 6]]
}
```

```
{
"Target Part": "tip",
"Selected dot": [4,7]
}
```

```
[
"1. The axis runs perpendicular to the length of the pen, passing through its center, with the arrow pointing outward from the side.",
"2. The axis is aligned along the length of the pen, extending from the handle towards the tip, with the arrow pointing towards the tip.",
"3. The axis runs diagonally, intersecting the pen's body, with the arrow pointing outward from the side.",
"4. The axis runs perpendicular to the length of the pen, passing through its center, with the arrow pointing inward from the side.",
"5. The axis is aligned along the length of the pen, extending from the tip towards the handle, with the arrow pointing towards the handle.",
"6. The axis runs diagonally, intersecting the pen's body, with the arrow pointing inward from the side."
]
```

```
{
"expected interaction direction caption": " Extend outward along the pen tip, away from the body",
"ranked directions": [2, 5, 1, 4, 3, 6]
}
```

```
{
"Target Part": "opening center",
"Selected dot": [4,4]
}
```

```
{
"Target Part": "opening center",
"Selected dot": [1,4]
}
```

```
[
"1. The axis passes through the holder's center, extending outward from the side, perpendicular to its height.",
"2. The axis runs vertically through the holder, passing through the opening center and extending upward.",
"3. The axis passes through the holder's center, extending outward from the side, perpendicular to its height.",
"4. The axis passes through the holder's center, extending outward from the side, perpendicular to its height.",
"5. The axis runs vertically through the holder, passing through the opening center and extending downward.",
"6. The axis passes through the holder's center, extending outward from the side, perpendicular to its height."
]
```

```
{
"expected interaction direction caption": " Perpendicular to the round opening of the holder, facing inward",
"ranked directions": [5, 2, 1, 3, 4, 6]
}
```

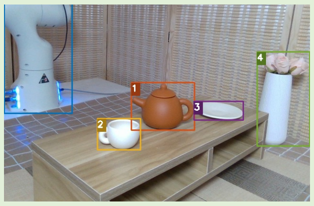
```
{
"Think Step By Step": "The image shows a pen positioned above a holder with the tip of the pen aligned
with the opening center of the holder. The red arrow points downward, indicating the direction for placing the
pen into the holder. The alignment and direction are appropriate for the task of inserting the pen into the
holder.",
"Reasonable": "success",
"interaction_value": 0
}
```

Figure 13. Result of task ‘Insert pen into holder’.

Instruction: Pick up cup onto dish

Task Stage Partitioning

Vision Foundation Model

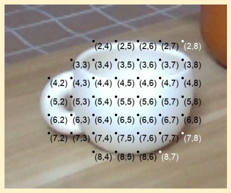


```
{
"Task Name": "Pick up cup onto dish",
"Think Step by Step": "First, grasp the cup using the gripper. Then, lift and move the cup to position it above the dish. Finally, place the
cup onto the dish.",
"objects": [
   {"object_ID": "cup", "bbox_ID": 2},
   {"object_ID": "dish", "bbox_ID": 3},
   {"object_ID": "gripper", "bbox_ID": -1},
   {"object_ID": "table", "bbox_ID": -1}
],
"stages": [
   {"action": "grasp", "active_object_ID": "gripper", "passive_object_ID": "cup"},
   {"action": "place", "active_object_ID": "cup", "passive_object_ID": "dish"}
],
"interaction_elements": [
   {"object_ID": "gripper", "element": {"action": "grasp", "type": "active", "interaction_part": "finger", "part_visible_and_tangible":
True}},
   {"object_ID": "cup", "element": {"action": "grasp", "type": "passive", "interaction_part": "handle center", "part_visible_and_tangible":
True}},
   {"object_ID": "cup", "element": {"action": "place", "type": "active", "interaction_part": "base center", "part_visible_and_tangible":
True}},
   {"object_ID": "dish", "element": {"action": "place", "type": "passive", "interaction_part": "center", "part_visible_and_tangible": True}}]
}
```

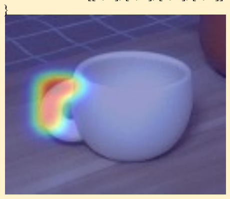
Stage-1: Grasp

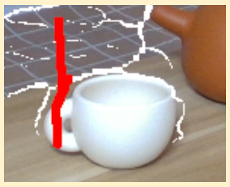
Functional grasping grounding

Passive Object (pen)



```
{
"Strategy": "To grasp the cup by its handle center, select points
that are centrally located on the handle for a stable grip.",
"Selected dots": [[4, 2], [4, 3], [5, 2], [6, 2]]
}
```

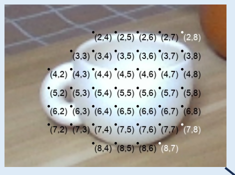




Stage-2: Place

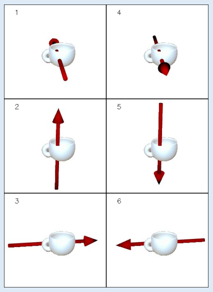
Active Object (cup)

Interaction point generation
(visible_and_tangible)



```
{
 "Target Part": "base center",
 "Selected dot": [7,5]
}
```

Interaction direction extraction

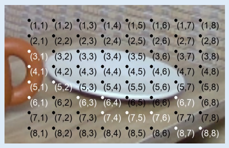


```
[
"1. The axis passes through the cup's base center, extending outward from the side,
perpendicular to the handle.",
"2. The axis runs vertically through the cup, passing through the base center and extending
upward.",
"3. The axis is horizontal, extending outward from the base center, parallel to the handle.",
"4. The axis passes through the cup's base center, extending outward from the side,
perpendicular to the handle.",
"5. The axis runs vertically through the cup, passing through the base center and extending
downward.",
"6. The axis is horizontal, extending outward from the base center, parallel to the handle."
]

{
"expected interaction direction caption": " Vertically through the cup, passing through
the base center and extending downward",
"ranked directions": [5, 2, 1, 4, 3, 6]
}
```

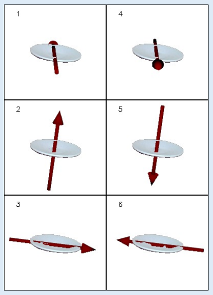
Passive Object (dish)

Interaction point generation
(visible_and_tangible)



```
{
 "Target Part": "surface",
 "Selected dot": [4,4]
}
```

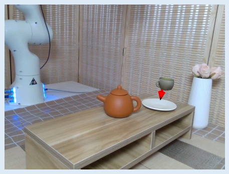
Interaction direction extraction



```
[
"1. The axis runs along the plane of the dish, extending.",
"2. The axis passes through the center of the dish, extending outward perpendicular to its
surface.",
"3. The axis runs along the plane of the dish, passing through its center.",
"4. The axis runs along the plane of the dish, passing through its center, similar to image 1
but from a different perspective.",
"5. The axis passes through the center of the dish, extending outward perpendicular to its
surface, similar to image 2 but from a different perspective.",
"6. The axis runs along the plane of the dish, passing through its center, in the opposite
direction of image 3."
]

{
"expected interaction direction caption": " Perpendicular to the round opening of the
holder, facing inward",
"ranked directions": [5, 1, 2, 4, 3, 6]
}
```

RRC



```
{
"Think Step By Step": "The image shows a cup positioned above a dish. The red arrow indicates a
downward direction, suggesting the cup will be placed onto the dish. The base center of the cup aligns with
the center of the dish, which is appropriate for the task of placing the cup on the dish. The interaction seems
reasonable as the cup is positioned correctly to be placed on the dish.",
"Reasonable": "success",
"interaction_value": 0
}
```

Figure 14. Result of task ‘Pick up cup onto dish’.