MapleGrasp: Mask-guided Feature Pooling for Language-driven Efficient Robotic Grasping Supplementary Material

Vineet Bhat Naman Patel Prashanth Krishnamurthy Ramesh Karri
Farshad Khorrami

New York University Tandon School of Engineering
Brooklyn, NY, USA
vrb9107@nyu.edu

1. Additional testing on language variation, dataset size and masked attention alternatives

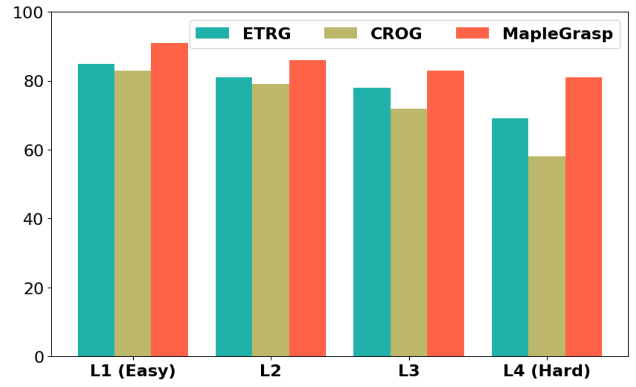


Figure 1. Comparisons across text inputs: from level 1 with just object names to level 4 with color, shape, position attributes

To assess robustness to language variation, we group test queries by object attributes. For example, "Grasp the red circular box near the top right of the image" contains three attributes. Using a popular named entity recognition model, GLiNER, we count attributes in the RefGraspNet test split and assign queries to bands: L1 EASY (one attribute) through L4 HARD (four). Fig. 1 plots Top-1 accuracy for MapleGrasp and baselines: accuracy declines with added attributes, yet MapleGrasp consistently remains ahead.

We also evaluate the training efficiency of MapleGrasp in comparison to two baseline methods. In this experiment, we vary the size of the training dataset and assess model performance after training on 20%, 40%, 60%, 80%, and 100% of the RefGraspNet corpus. The results, presented in Fig. 2, demonstrate that MapleGrasp consistently outperforms the baselines on the test set as the dataset size increases, with particularly notable improvements observed after training on just 20–40% of the data. These findings suggest that incorporating referring object masks for pooling vision-language feature maps in grasp detection enhances both training effectiveness and sample efficiency.

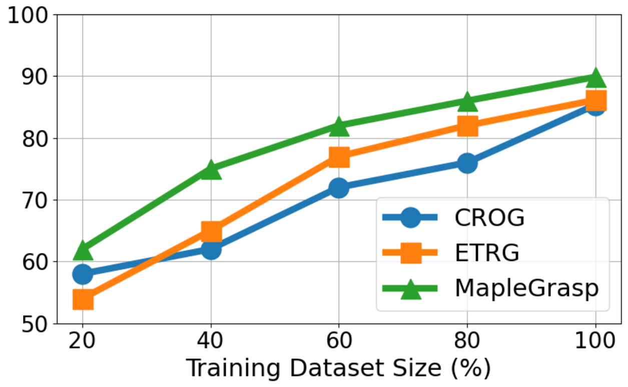


Figure 2. Training data efficiency comparisons across baselines on RefGraspNet. Scores reported as grasping success rate in %.

2. Limitations and Failure Cases

While MapleGrasp demonstrates strong overall performance, certain challenges remain. The main sources of failure arise from (i) occlusions and partial visibility of target objects, (ii) ambiguities in referring expressions, and (iii) limitations of mask-based pooling in complex spatial arrangements. We present representative examples in Table 1 and discuss them below, along with possible remedies.

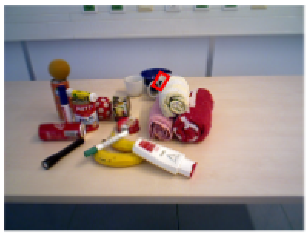
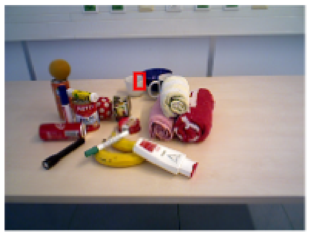
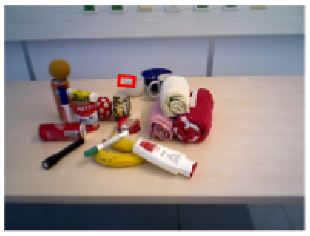
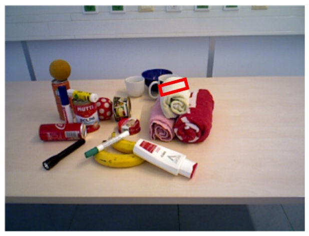
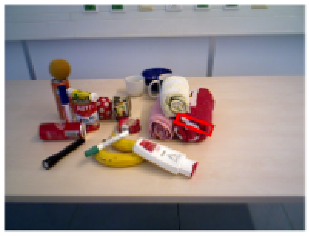
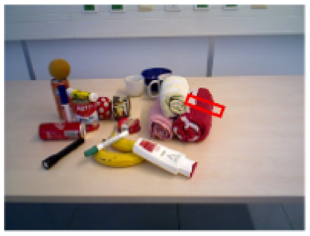
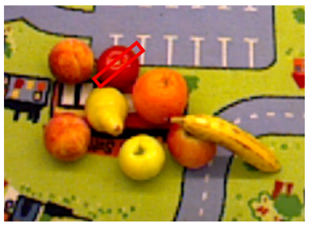
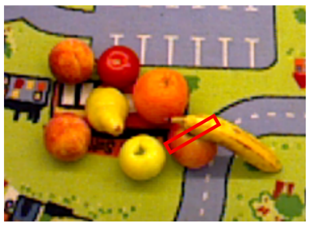
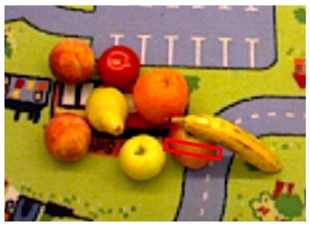
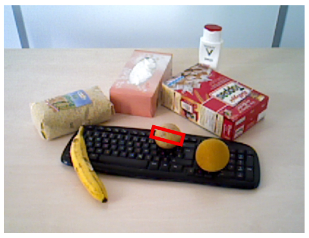
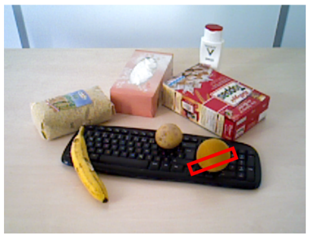
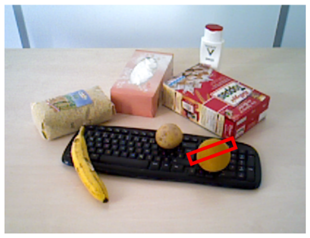
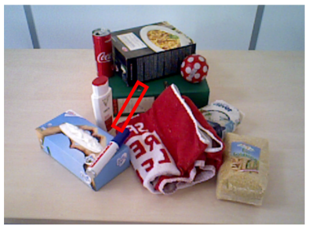
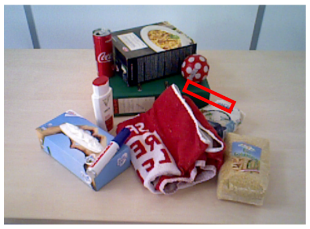
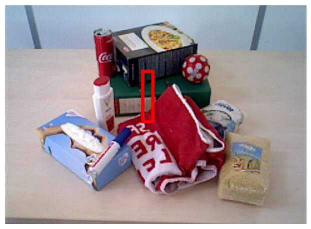
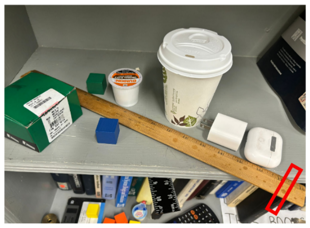
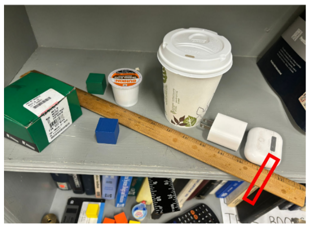
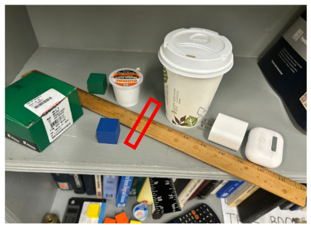
Referring Text	Ground Truth	CROG	MapleGrasp
Pass me the right mug			
Get me a towel			
Find me an apple that is easier to grasp			
Grab the yellow circular food item furthest away from the camera			
Grab the green textbook			
Pass me the wooden ruler			

Table 1. Failure cases of our approach. In the future we will include an LLM agent in our framework that can intelligently capture better images of the workspace or provide an interface to communicate with the human for a better target oriented grasping.

2.1. Occlusions and Partial Visibility

In rows 1 and 5, the target object is heavily occluded or only partially visible. This reduces the quality of the predicted mask and consequently weakens the grasp decoder's ability to generate accurate poses. For example, in row 1, MapleGrasp's predicted mask is extremely small and shifted off the true target, while in row 5 the mask points to the correct object but fails to align with the annotated grasp. Notably, even the ground-truth annotation in row 5 appears ambiguous, highlighting that ideal strategies may require multi-step retrieval—first clearing surrounding clutter before attempting a grasp. **Potential remedy:** Incorporating multi-step planned grasp estimation or reasoning about occlusions would allow the model to infer when a preparatory action is needed before executing the grasp.

2.2. Ambiguities in Referring Expressions

Rows 2–4 illustrate cases where the referring query is underspecified. In row 2, the scene contains multiple towels, but the ground truth selects the easiest-to-grasp instance. Neither CROG nor MapleGrasp learns this implicit bias, leading to failures. Row 3 highlights a query such as "apple that is easier to grasp": while humans intuitively choose the left apple (least occluded, no object leaning against it), both models lack the scene-physics understanding to reason about stability or occlusion. In row 4, two yellow circular items are close together, but one is slightly ahead in depth; existing queries are typically phrased in image-centric coordinates (e.g., "top of the image") rather than camera or world-centric coordinates, limiting spatial reasoning. **Potential remedy:** Future work should co-train models on datasets with explicit spatial and physical relationship annotations, enabling general-purpose reasoning about object affordances and scene geometry along with fine-tuning on the grasping task.

2.3. Limitations of Mask-Based Pooling

Row 6 shows a case where mask pooling constrains the model too tightly to the target object. The only feasible grasp of the ruler requires approaching from underneath, but the predicted mask excludes surrounding free space that is critical for collision-free grasping. CROG, which does not rely on mask pooling, partially attempts this strategy but still misses the ground truth. **Potential remedy:** Relaxing mask boundaries (e.g., adding a small offset margin or lowering softmax thresholds) could preserve contextual cues around the object, giving the model additional room to predict valid grasp trajectories.

2.4. Failure Cases Due to Incorrect Mask Prediction in robot trials

In our robot trials, we observed that MapleGrasp occasionally produces partial masks for unseen objects. These incomplete segmentations often lead to boundary-aligned grasp poses that are difficult to execute reliably. Since our approach relies strongly on the quality of the predicted

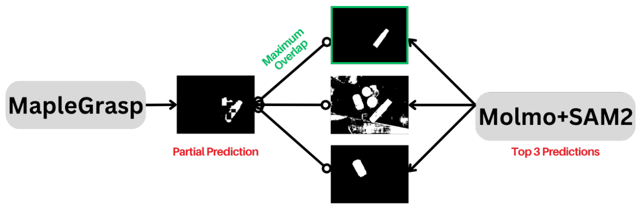


Figure 3. MapleGrasp-Ensemble uses the open-set strength of large VLMs with finer language based grounding capabilities of MapleGrasp to generate better segmentation masks

mask, stable grasping requires masks that capture the object's geometry around its center of mass, ensuring the decoder can predict grasps supported by multiple contact surfaces. We also compared against the zero-shot pipeline Molmo+SAM2+GraspNet, which typically generates complete object masks. However, this approach struggles in scenes with complex referring queries or multiple distractors. To combine the strengths of both methods, we designed an ensemble strategy illustrated in Fig. 3. Given the same input image and referring expression, we take MapleGrasp's prediction along with the top-3 masks from Molmo+SAM2. We then compute the IoU between MapleGrasp's mask and each Molmo+SAM2 candidate, selecting the candidate with the highest IoU as the final segmentation. This refined mask is subsequently used for pooling in MapleGrasp's decoder to generate the grasp pose. This ensemble approach mitigates the effect of partial masks, providing more complete object coverage and improving grasp stability on unseen objects.

3. Inverse Kinematics Control for Manipulation

Our approach for real world object manipulation transforms the 2D grasp prediction into a desired 3D pose. Specifically, we obtain the final grasp pose in the robot's base frame, $\mathbf{T}^{\text{pose}}_{\text{base}}$, by composing three transformations:

$$\mathbf{T}^{\text{pose}}_{\text{base}} = \mathbf{T}^{\text{ee}}_{\text{base}}\,\mathbf{T}^{\text{cam}}_{\text{ee}}\,\mathbf{T}^{\text{pose}}_{\text{cam}}, \tag{1}$$

where:

1. $\mathbf{T}^{\text{ee}}_{\text{base}}$ is the forward-kinematics transformation from the robot base to its end-effector,
2. $\mathbf{T}^{\text{cam}}_{\text{ee}}$ is the hand–eye calibration matrix (mapping from end-effector to camera),
3. $\mathbf{T}^{\text{pose}}_{\text{cam}}$ is the camera-to-pose transformation predicted by our grasp model.

To execute a grasp in closed loop, we generate a smooth interpolation $\mathcal{C}(t)$ in SE(3) between the initial end-effector pose $\mathbf{T}_{\text{init}}$ and the desired grasp pose $\mathbf{T}_{\text{goal}}$, as:

$$\mathcal{C}(t)\;:\;[0,\,1]\;\rightarrow\;\mathrm{SE}(3),\quad \mathcal{C}(0)=\mathbf{T}_{\text{init}},\quad \mathcal{C}(1)=\mathbf{T}_{\text{goal}}. \tag{2}$$

At each control step t, the desired pose is

$$\mathbf{T}_{\text{des}} \;=\; \mathcal{C}(t). \tag{3}$$

Let $\mathbf{T}_{\text{current}}$ denote the current pose of the end effector. We define the positional error

$$\mathbf{e}_p = \mathbf{p}_{\text{des}} - \mathbf{p}_{\text{current}}, \tag{4}$$

and compute the rotational error using the matrix logarithm of the orientation difference:

$$\mathbf{R}_{\text{err}} = \mathbf{R}_{\text{des}}\,\mathbf{R}_{\text{current}}^{\top}, \qquad \boldsymbol{\theta}_{\text{err}} = \operatorname{Log}\bigl(\mathbf{R}_{\text{err}}\bigr), \tag{5}$$

where $\mathbf{R}_{\text{des}}$ and $\mathbf{R}_{\text{current}}$ are the rotation components of $\mathbf{T}_{\text{des}}$ and $\mathbf{T}_{\text{current}}$, respectively, and $\operatorname{Log}(\cdot)$ is the matrix logarithm on $\mathrm{SO}(3)$. We form the combined error vector

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_p \\ \boldsymbol{\theta}_{\text{err}} \end{bmatrix} \;\in\; \mathbb{R}^6. \tag{6}$$

The desired end-effector velocity is then

$$\dot{\mathbf{x}}_{\text{target}} \;=\; K_p\,\mathbf{e}, \tag{7}$$

where K_p is a diagonal gain matrix. We compute the joint velocity command by projecting $\dot{\mathbf{x}}_{\text{target}}$ through the robot Jacobian pseudoinverse:

$$\dot{\mathbf{q}}_{\text{cmd}} \;=\; \mathbf{J}^{\dagger}\,\dot{\mathbf{x}}_{\text{target}}. \tag{8}$$

Finally, we update the robot's joint states with $\dot{\mathbf{q}}_{\text{cmd}}$ and advance $t \leftarrow t + \Delta t$. This process repeats until

$$\|\mathbf{T}_{\text{des}} - \mathbf{T}_{\text{current}}\| \;\le\; \epsilon, \tag{9}$$

indicating successful convergence to the target grasp pose.

4. LIBERO Task Suites

To evaluate complementary competencies of our vision–language–action policy, we draw on three ten-task suites from the LIBERO simulation environment: **Goal**, **Object**, and **Spatial**. Each suite probes a different aspect of robot reasoning.

- **Goal Suite** – long-horizon *state-achievement* problems that require composing several primitive skills (e.g., open a drawer, insert an object, close the drawer) until a final world state is satisfied.
- **Object Suite** – *semantic object grounding*. The scene layout is fixed; success hinges on recognising the correct item among distractors and placing it in a basket, testing category generalisation.
- **Spatial Suite** – *spatial-language grounding*. All tasks concern a black bowl whose initial spatial relation to other scene elements varies ("on", "next to", "between", etc.); the agent must interpret the phrase, localise the bowl, and place it precisely on a plate.

Tasks in Each Suite

Goal (10).

1. Put the wine bottle on top of the cabinet.
2. Open the top drawer and put the bowl inside.
3. Turn on the stove.
4. Put the bowl on top of the cabinet.
5. Put the bowl on the plate.
6. Put the wine bottle on the rack.
7. Put the cream cheese in the bowl.
8. Open the middle drawer of the cabinet.
9. Push the plate to the front of the stove.
10. Put the bowl on the stove.

Object (10).

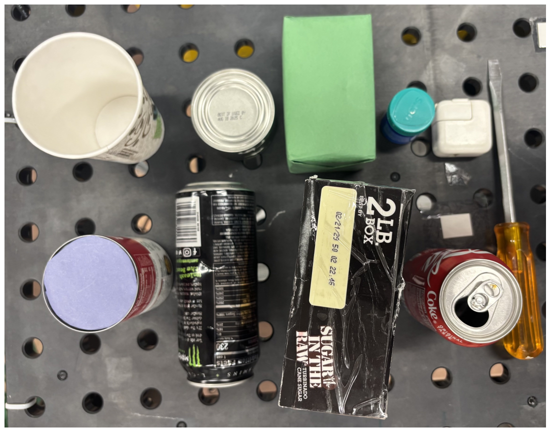
1. Pick up the alphabet soup and place it in the basket.
2. Pick up the cream cheese and place it in the basket.
3. Pick up the milk and place it in the basket.
4. Pick up the tomato sauce and place it in the basket.
5. Pick up the butter and place it in the basket.
6. Pick up the orange juice and place it in the basket.
7. Pick up the chocolate pudding and place it in the basket.
8. Pick up the BBQ sauce and place it in the basket.
9. Pick up the ketchup and place it in the basket.
10. Pick up the salad dressing and place it in the basket.

Spatial (10).

1. Pick up the black bowl from the table centre and place it on the plate.
2. Pick up the black bowl next to the cookie box and place it on the plate.
3. Pick up the black bowl next to the plate and place it on the plate.
4. Pick up the black bowl next to the ramekin and place it on the plate.
5. Pick up the black bowl on the cookie box and place it on the plate.

6. Pick up the black bowl on the ramekin and place it on the plate.
7. Pick up the black bowl on the stove and place it on the plate.
8. Pick up the black bowl on the wooden cabinet and place it on the plate.
9. Pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate.
10. Pick up the black bowl between the plate and the ramekin and place it on the plate.

5. Real world Trials


(a) Seen Objects

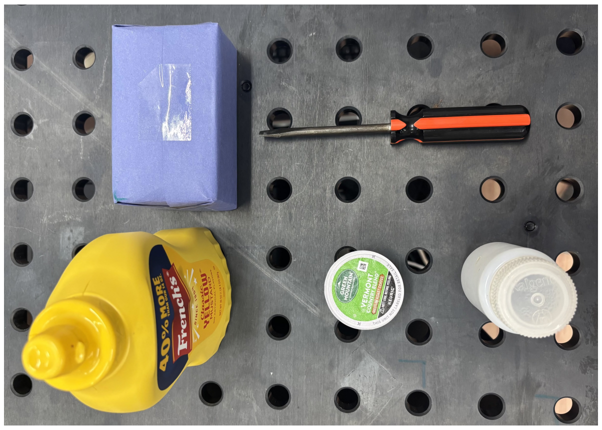

(b) Unseen Objects

Figure 4. Set of objects used in real robot trials.

We conduct comprehensive experiments in the real open world to test MapleGrasp and other baselines within the standard table-top environment with a 7 DoF Franka arm. The robot's camera is attached to the gripper which allows a top-down image capture of the workspace for referring grasp synthesis. A front-view camera prevents the objects from being placed in a random compact fashion and hence we use the camera in hand setting. We collected 15 objects for our experiments as shown in Fig. 4. 10 objects were chosen to be seen as they are very similar to some of the objects present in the OCID-VLG and RefGraspNet datasets. The experiments followed the below set of steps -

1. We first chose an object to be the target object for the experiment
2. We confirm if the object is being grasped in a single object setting by just placing the object at the center of the workspace visible within the camera view and use a referring expression. This is repeated for all the 3 baselines and MapleGrasp
3. Then we create a workspace comprising 7-10 objects, randomly placed but visible within camera view. The goal remains to grasp the chosen target object, but we test our various configurations where the object is completely visible, partially visible or graspable only from one side.
4. Once the workspace has been set, we test the task using 10 different grasping instructions using a simple LLM call to diversify the sentence. We ensure that the text query does not deviate significantly from the type of queries seen in the training dataset
5. We record the success for each of the 10 queries and repeat all steps for all the objects - recording data correctly for seen and unseen items.

HiFi-CS, CROG and MapleGrasp initially return a rectangular grasp. We project the rectangle to 3D using the depth map and point cloud representation which allows us to identify graspable points on the surface of the object. These points are then sent to a function which interfaces with the 6 DoF grasp proposals from Contact-GraspNet to choose the most appropriate 6 DoF grasp for execution. This allows for a more balanced evaluation for all baselines while ensuring the core comparisons between various approaches on visual grounding and grasp pose estimation is measured.

6. Articulated Tasks in LIBERO

We evaluate MapleGrasp on two articulated manipulation tasks from LIBERO: operating a stove knob and manipulating a drawer handle. While MapleGrasp generates 6-DoF grasp poses, a pre-defined skill controller is used after grasping the target object to complete the articulation (knob turning or drawer pulling). Examples of both tasks are shown in Tab. 2.

Importantly, MapleGrasp was trained only on grasp data from the combined LIBERO Spatial, Goal, and Object datasets, which are easier to annotate. In contrast, OpenVLA was trained end-to-end on full task demonstrations, with 50 roll-outs per task. Despite this difference, our results show that MapleGrasp achieves comparable performance without requiring demonstration data, highlighting the effectiveness of separating grasp generation from downstream skill execution.

On the stove knob task, MapleGrasp outperforms OpenVLA. OpenVLA often struggles to precisely rotate the knob, leading to oscillatory behavior around the target before termination. On the drawer task, however, OpenVLA marginally outperforms MapleGrasp, since successful execution requires fine-grained control to avoid obstacles around the drawer handle. Our heuristic controller failed in four trials due to collisions with neighboring structures, such as the top drawer's handle.

These findings suggest that while pre-defined skill controllers are sufficient for validating grasp generalization in articulated settings, they limit performance on tasks requiring adaptive, fine-grained control. We believe that incorporating imitation-learned skill modules trained on human demonstrations would extend MapleGrasp's applicability to a broader range of articulated tasks, alleviating the need for end-to-end demonstrations.

7. More comparisons in OCID-VLG and LIBERO

In this section, we present additional qualitative examples from OCID-VLG and LIBERO-based evaluations. For clarity, we illustrate rectangular grasp representations in the figures, though all RefGraspNet and LIBERO experiments are executed with full 6-DoF grasp poses. Table 4 compares predictions from CROG and MapleGrasp against the ground-truth grasps. In the first example, MapleGrasp predicts a stable grasp for a ball, while CROG's prediction is unstable. In the second, MapleGrasp correctly infers a larger grasp width for a wide object, whereas CROG underestimates the width. In the third, our use of segmentation masks helps focus on the correct region of the target object, improving accuracy. Table 3 shows two examples from LIBERO. In the first, OpenVLA fails to execute the correct grasp motion for the referred object, likely due to its reliance on precise mappings from vision–language inputs to joint-space velocities. Such mappings typically require many tele-operated demonstrations, and performance can degrade if object positions shift due to simulator randomness. In contrast, MapleGrasp generates visually grounded grasp poses that remain valid across object arrangements. These poses are executed via a precision inverse kinematics controller, enabling accurate motion generation without the need for demonstration-heavy training. Importantly, our approach requires only grasp-rectangle annotations, which are far easier to collect than tele-operated data. Once an object is grasped, identifying empty spaces or target locations could be delegated to powerful VLMs (e.g., Molmo) to produce placement motions—a promising direction for future work that would further extend MapleGrasp's capabilities.

8. Rectangular 2-D Grasp Representation

A 2-D grasp in language-driven grasping is commonly characterized by four parameters: $(x,\, y,\, w_{\text{rect}},\, \theta)$, where (x, y) is the center of the grasp in image coordinates, w_{rect} is the gripper width, and θ is the top-down rotation angle around the vertical axis. In some cases, an additional height component h_{rect} is also defined, for example, when the predicted bounding box is more rectangular than square. Fig. 5 illustrates this representation.

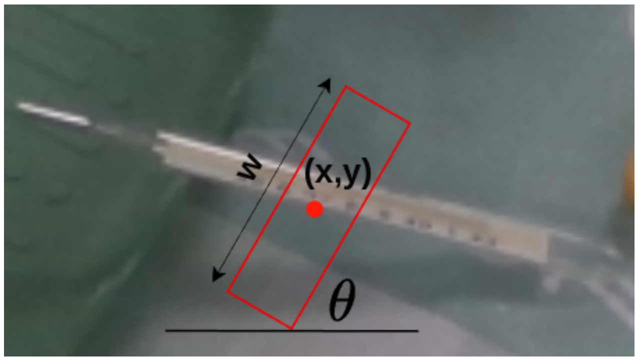


Figure 5. 2-D rectangular grasp pose. Here, (x, y) is the center, w_{rect} the gripper width, and θ the rotation angle about vertical axis.

Although not all objects can be grasped with a top-down approach, many eye-in-hand or overhead camera setups naturally favor a top-down configuration for robotic manipulation. This greatly simplifies both perception and control in tabletop or industrial scenarios.

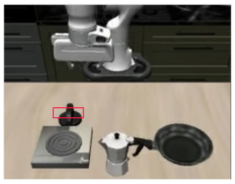
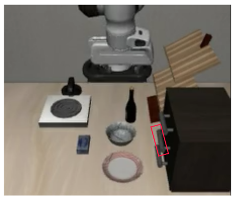
Task	MapleGrasp Prediction	Robot roll-out			Success rate with MapleGrasp	Success rate with OpenVLA
Turn on the stove using the knob			...		49 / 50	45 / 50
Open the middle drawer by grasping the handle along its axis and pulling			...		46 / 50	48 / 50

Table 2. Comparing two articulated tasks in LIBERO using modular MapleGrasp approach vs OpenVLA

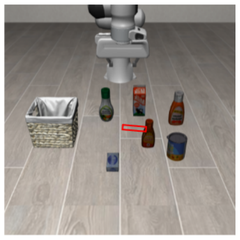
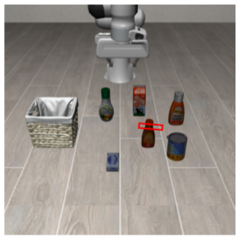
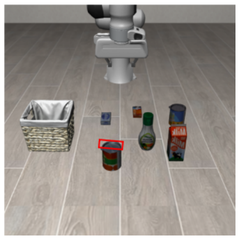
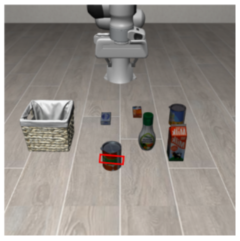
Task Instruction	OpenVLA	MapleGrasp
Grab the bbq sauce		
Pick up the alphabet soup		

Table 3. Comparisons with OpenVLA on the LIBERO benchmark. In certain rollouts, OpenVLA fails to complete the task due to inaccurate grasping as shown. Our method performs well and predicts a stable oriented grasp which is executed with our inverse kinematics controller within the environment.

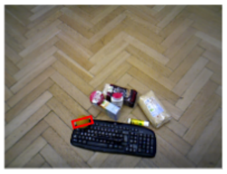
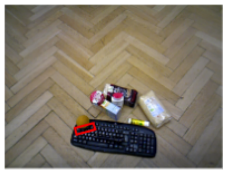
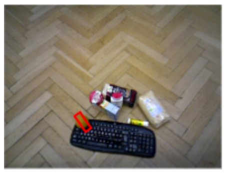
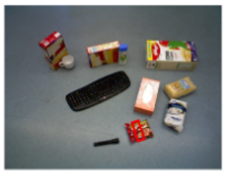
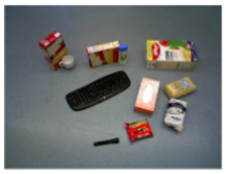
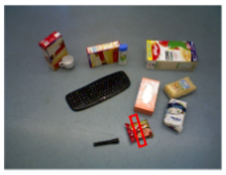
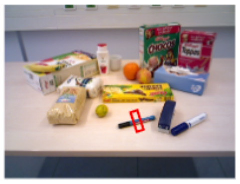
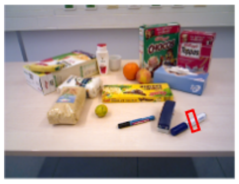
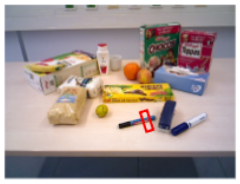
Referring Text	Ground Truth	CROG	MapleGrasp
Grab the yellow ball			
Grab the noodles			
Grasp the blue marker on the left			

Table 4. More successful examples of MapleGrasp compared to CROG. MapleGrasp learns an geometrically aware grasp feature map for each referred object, leading to accurate visual grounding and stable grasps.