

OVA-Fields: Weakly Supervised Open-Vocabulary Affordance Fields for Robot Operational Part Detection

Supplementary Material

1. Training Details

In our experiments, the OVA-Fields is trained on a workstation with an NVIDIA RTX 4090 GPU. We use the Adam optimizer with an initial learning rate of $1e-4$ and a weight decay of 0.003. The visual encoder is CLIP ViT-B/32, and the text encoder is Sentence-BERT all-mpnet-base-v2. The model is trained for 70 epochs, with each epoch taking 20 seconds on average. The batch size is 11,000, and the training process is completed in about 20 minutes. We use a Multiresolution Hash Encoder (MHE) to encode 3D spatial features. The input is 3D coordinates $(x, y, z) \in \mathbb{R}^3$. The encoder has 16 grid layers, each with a feature dimension of 8. The size of hash table is 2^{24} , and the scale factor of each layer is 2.0. The encoded spatial features are combined with affordance embeddings. Affordance heatmap values are linearly transformed into high-dimensional representations consistent with the spatial features. Threshold t and scale s are set to adjust the heatmap values appropriately.

Item/Parameters	Method/Value
Optimizer	Adam
Initial Learning Rate	10^{-4}
Weight Decay	3×10^{-3}
Visual Encoder	CLIP ViT-B/32
Text Encoder	Sentence-BERT all-mpnet-base-v2
Epochs	70
Batch Size	11,000
MHE Grid layers	16
Hash Table Size	2^{24}
Num of Attention Head	8
Embedding Dimension	128
MLP Activate Funtion	ReLU
α	1.0
β	1.0

Table 1. Training details

The combined features are enhanced using an 8-head multi-head attention mechanism with an embedding dimension of 128. This mechanism captures the crucial information of functional components. The enhanced features are processed through an MLP with two hidden layers, each with a dimension of 600 and ReLU activation. The MLP produces high-dimensional feature representations. Batch normalization is embedded in the MLP to improve model stability and generalizability. The loss function consists of visual embedding loss and affordance embedding loss. A temperature parameter, initially set to $1.0/0.07$, controls the

smoothness of the similarity distribution. Both losses are assigned equal weight coefficients α and β of 1.0 during the total loss calculation. Specific training details are provided in Tab. 1.

2. More Application Results

The OVA-Fields automatically highlights actionable parts of a scene based on user commands. For example, when instructed to “take out some food from the refrigerator”, the framework correctly identifies the fridge handle. Our method consistently shows accurate part-level affordance detection for various objects, including a fridge, microwave, and knife, across tasks such as “grab the bottle” or “give me the knife”.

Why we design these diverse and challenging experiments? The goal is to test whether the OVA-Fields can handle not only straightforward commands but also more complex and ambiguous instructions. For instance, the OVA-Fields performs well with ambiguous queries, accurately identifying affordance locations even when instructions are vague, such as recognizing “the sharp object” or responding to “I want to cook the chicken”.

Additionally, the OVA-Fields can interpret polysemous queries. In a scene with multiple fruits, it successfully highlights the banana when given the command “help me grab the yellow fruit.” These experiments demonstrate our framework’s strength in managing both complex semantic and ambiguous queries, bridging the gap between affordance detection and real-world robotic interaction.

The OVA-Fields accurately detects and localizes multiple objects and tasks in multi-step queries. Examples include “Put the banana on the table into the refrigerator” and “Use the knife to cut the banana.” It identifies key functional components, such as the knife blade and refrigerator handle, to complete these tasks.

The model also handles ambiguous queries effectively. For commands such as “Give me the yellow fruit” or “give me the sharp object,” it recognizes the correct targets, such as the banana or knife blade, while ignoring unrelated details. These results show the model’s ability to handle both explicit and ambiguous queries. It demonstrates adaptability and robustness in interpreting open-ended instructions. Fig.1 visualizes the model’s performance in detecting fine-grained objects and understanding ambiguous commands. This confirms the practicality and efficiency of OVA-Fields in real-world applications. The details of queries are shown in Fig. 1, Tab. 2.



Figure 1. **Experimental results of our framework detecting affordance locations in real-world scenes based on user commands.** This demonstrates the superior affordance detection ability of the OVA-Fields across various objects and tasks. The OVA-Fields also handles ambiguous queries and multi-tasks queries effectively, accurately identifying relevant affordance locations even when user instructions are ambiguous

These results show the model’s ability to handle both explicit and ambiguous queries. It demonstrates adaptability and robustness in interpreting open-ended instructions. This confirms the practicality and efficiency of OVA-Fields in real-world applications.

3. Details of Real Robot Experiments

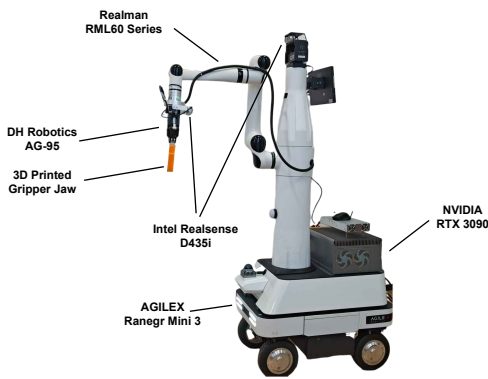


Figure 2. **Mobile robot platform used in our experiments.** This platform consists of AGILEX Ranger Mini 3 chassis, Realman RML60 robotic arm, DH Robotics AG-95 gripper, Intel Realsense D435i camera, and an NVIDIA RTX 3090 for processing. Also, we install a 3D printed gripper jaw on the gripper to make it easier to grab items.

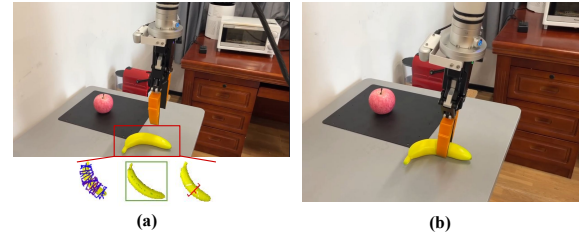


Figure 3. **Grasp pose obtain method.** (a) Gripper remains fixed at a specific height above the affordance coordinate and calculates the gripper pose (b) The robot determines the precise grasp pose.

As shown in Fig. 3, our model does not include the robotic arm’s pose data. To address this issue, we keep the gripper fixed at a specific height above the affordance coordinate. The robot begins by generating several candidate grasp poses using the AnyGrasp algorithm, which identifies potential regions for grasping based on the object’s geometry. Next, LangSAM refines this selection by segmenting the target object and filtering out unsuitable grasp poses based on their stability and feasibility. The task is considered successful once the robot completes to grasp the object items. We test various user queries in real-world environments. Starting from different positions in the scenes, the robot consistently identifies the correct objects and completes the tasks. Examples of experimental results are shown in Fig. 4. LLM-generated queries in our real robot experiments are shown at Tab. 3-Tab. 6

Scene	Query
Lab	<i>take out some food from the refrigerator</i> <i>help me to take the bottle</i> <i>give me the knife</i> <i>I want to eat banana</i> <i>roasting chicken in the oven</i> <i>I want to use the yellow pen to write something on the paper</i>
Home	<i>take out some food from the refrigerator</i> <i>help me input something on the laptop</i> <i>take the cup from the table</i> <i>where can i seat on chair</i> <i>pick up the phone from the desk</i>
ScanNet	<i>take out some food from the refrigerator</i> <i>take the bottle from the table</i> <i>give me the metal bowl</i> <i>take the bottle from the table</i> <i>give me the metal bowl</i> <i>take the bottle to me from the table</i> <i>give me the book</i> <i>bring the knife from the kitchen counter</i> <i>get a pen from the pen holder</i> <i>grab the fork from the dining table</i> <i>pick up the axe from the tool shed</i> <i>get the scissors from the table</i>
SceneFun3D	<i>I'm thirsty, pass me the teapot on the kitchen counter</i> <i>Help me open the microwave</i> <i>Please open the oven</i> <i>Open the refrigerator next to the oven</i> <i>Give me the bottle on the table</i> <i>I want to read the book on the coffee table</i>

Table 2. **Query details.** This table lists all user queries tested across different scenes: Lab, Home, and ScanNet Datasets. These queries include a variety of instructions, such as retrieving objects, interacting with specific items, and demonstrating the model’s ability to handle diverse and realistic commands in varied environments.

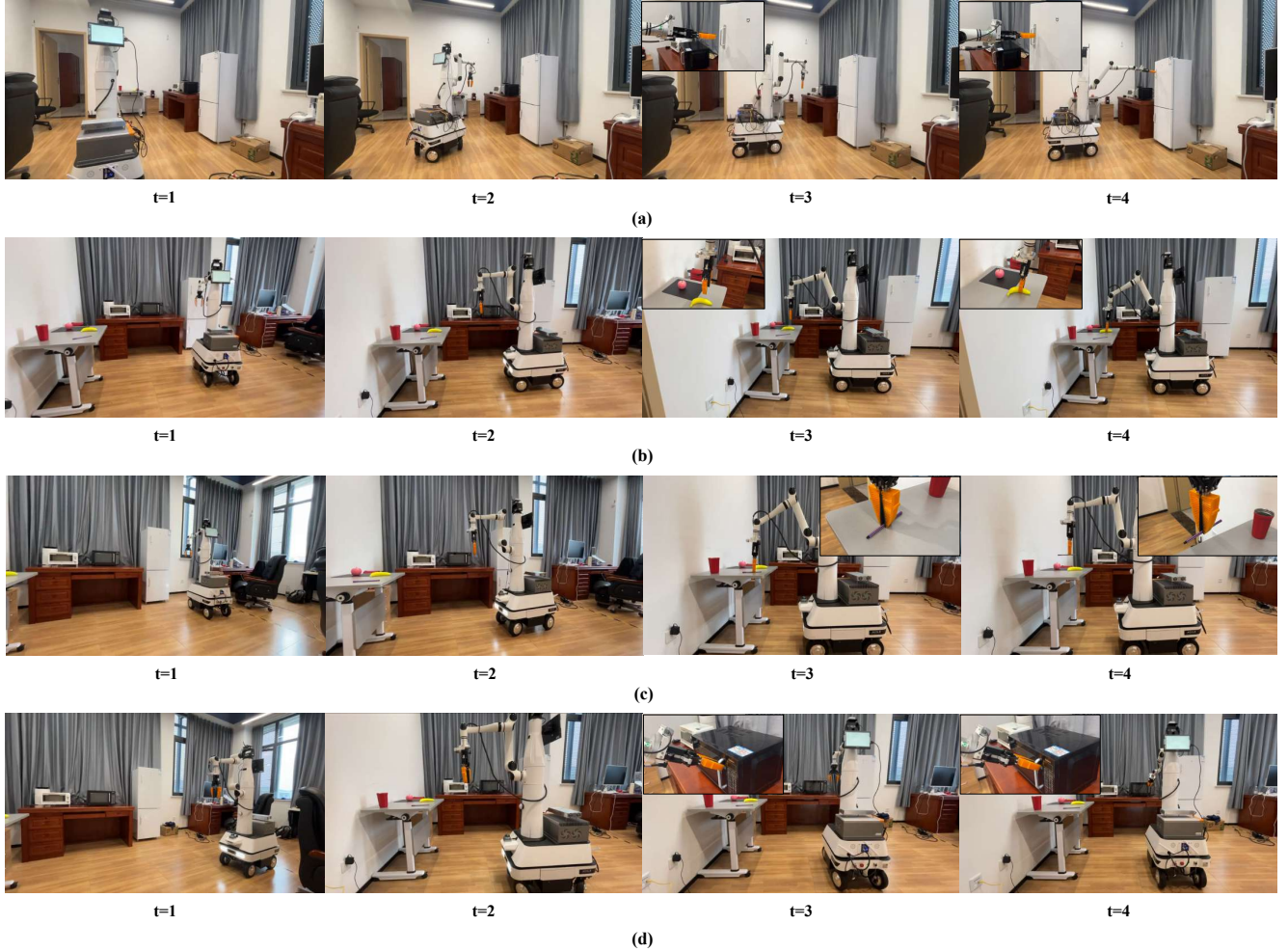


Figure 4. **Robotics application results.** (a) The sequence illustrates the robot performing the task “Take out some food from the refrigerator.” At $t=1$ and $t=2$, the robot identifies the affordance location. At $t=3$, it aligns its gripper with the identified handle and begins interaction. At $t=4$, the robot successfully opens the refrigerator, demonstrating its ability to accurately locate and interact with functional components in real-world environments. Since the model does not generate the robotic arm’s pose for opening the refrigerator, successful grasping of the handle is considered as completing the task. (b) The sequence illustrates the robot performing the task “I want to eat banana.” At $t=1$ and $t=2$, the robot identifies the affordance location. At $t=3$, the robot aligns its gripper with the banana. At $t=4$, it successfully grasps the banana, demonstrating precise affordance detection and manipulation based on user instructions. (c) The sequence illustrates the robot performing the task “I want to use the pen on the table to write something.” At $t=1$ and $t=2$, the robot identifies the affordance location. At $t=3$, the robot aligns its gripper with the pen. At $t=4$, it successfully grasps the pen, showcasing accurate affordance detection and manipulation guided by the user’s query. (d) The sequence illustrates the robot performing the task “Warm up my lunch in the microwave.” At $t=1$ and $t=2$, the robot identifies the location of the microwave handle. At $t=3$, the robot aligns its gripper with the microwave handle. At $t=4$, the gripper successfully grasps the handle. Since the model does not generate the robotic arm’s pose for opening the microwave, successful grasping of the handle is considered as completing the task.

Semantic Complexity	Query
Level 1	<i>Open the fridge</i> <i>Open the refrigerator door</i> <i>Pull the refrigerator handle</i> <i>Access the refrigerator</i> <i>Unlatch the fridge</i> <i>Swing the fridge door open</i> <i>Move the refrigerator door</i> <i>Release the fridge latch</i> <i>Begin fridge interaction</i> <i>Initiate refrigerator access</i>
Level 2	<i>Take out some food</i> <i>Grab an item from the fridge</i> <i>Remove a meal from the refrigerator</i> <i>Extract a food container</i> <i>Retrieve leftovers</i> <i>Fetch a drink from the fridge</i> <i>Collect a snack from the refrigerator</i> <i>Withdraw a vegetable</i> <i>Pick up a dairy product</i> <i>Transfer food to the counter</i>
Level 3	<i>Find a snack for my lunch</i> <i>Locate the leftover pizza in the fridge</i> <i>Search for a healthy option inside</i> <i>Identify the oldest item to remove</i> <i>Check if there's expired food to discard</i> <i>Choose a low-calorie meal from the shelves</i> <i>Detect any fruits that need refrigeration</i> <i>Prioritize perishable items first</i> <i>Avoid touching the glass jars</i> <i>Ensure the door closes automatically</i>

Table 3. Basic to complex commands for open the refrigerator operations

Semantic Complexity	Query
Level 1	<i>Get a banana</i> <i>Hand me the banana</i> <i>Pick up the banana</i> <i>Locate the banana</i> <i>Move the banana here</i> <i>Select a yellow fruit</i> <i>Grasp the banana</i> <i>Bring the banana</i> <i>Identify the banana</i> <i>Transfer the banana</i>
Level 2	<i>Peel the banana and place it on the plate</i> <i>Pick the ripe banana from the fruit bowl</i> <i>Avoid the green bananas; choose a yellow one</i> <i>Check for bruises before grabbing</i> <i>Separate the banana from other fruits</i> <i>Measure the banana's weight first</i> <i>Align the gripper with the stem</i> <i>Ensure no other fruits are touched</i> <i>Verify the banana's position</i> <i>Rotate the banana for better grip</i>
Level 3	<i>I'm on a diet; find the smallest banana</i> <i>The banana is behind the apples; navigate carefully</i> <i>Prepare the banana for a smoothie by cutting it</i> <i>Check if the banana is organic or imported</i> <i>If the banana is too soft, discard it</i> <i>Compare sizes and pick the healthiest option</i> <i>Avoid the bunch; take only one banana</i> <i>Ensure the banana is not touching dirty surfaces</i> <i>Prioritize bananas with brown spots for immediate use</i> <i>Estimate calories before serving</i>

Table 4. Basic to complex commands for grab the banana's operations

Semantic Complexity	Query
Level 1	<i>Open the microwave</i> <i>Turn on the oven</i> <i>Activate the stove</i> <i>Access the cooking appliance</i> <i>Locate the microwave handle</i> <i>Press the oven start button</i> <i>Rotate the stove knob</i> <i>Close the microwave door</i> <i>Check appliance power status</i> <i>Initiate heating mode</i>
Level 2	<i>Place the chicken in the microwave</i> <i>Set the oven to 180°C for cooking</i> <i>Heat the chicken for 10 minutes</i> <i>Ensure the chicken is fully covered</i> <i>Adjust the stove flame to medium</i> <i>Verify the chicken's position inside</i> <i>Avoid touching hot surfaces</i> <i>Monitor cooking progress</i> <i>Rotate the chicken halfway</i> <i>Confirm temperature safety</i>
Level 3	<i>Defrost the frozen chicken in the microwave first</i> <i>Marinate the chicken before placing it in the oven</i> <i>Check internal temperature with a probe</i> <i>If smoke is detected, pause and ventilate</i> <i>Compare cooking time across appliances</i> <i>Prioritize energy-efficient heating methods</i> <i>Ensure no cross-contamination with utensils</i> <i>Adjust seasoning based on weight</i> <i>Schedule delayed cooking for dinner</i> <i>Handle raw chicken with gloves</i>

Table 5. Basic to complex commands for open the microwave operations

Semantic Complexity	Query
Level 1	<i>Find a pen</i> <i>Bring the pen here</i> <i>Locate a writing tool</i> <i>Pick up the pen</i> <i>Move the pen to the desk</i> <i>Identify any available pen</i> <i>Check pen functionality</i> <i>Select a blue pen</i> <i>Avoid broken pens</i> <i>Transfer the pen</i>
Level 2	<i>Ensure the pen has ink</i> <i>Test the pen on paper first</i> <i>Remove the pen cap carefully</i> <i>Align the pen tip for writing</i> <i>Check grip comfort</i> <i>Verify no leaks</i> <i>Compare pen types (ballpoint vs. gel)</i> <i>Adjust pen angle for better control</i> <i>Store the pen upright</i> <i>Avoid pressing too hard</i>
Level 3	<i>Sign a document neatly with the pen</i> <i>Write a birthday card in cursive</i> <i>Draw a straight line without a ruler</i> <i>Replace the ink cartridge if empty</i> <i>Use the pen to solve a math equation</i> <i>Avoid smudging the fresh ink</i> <i>Estimate remaining ink volume</i> <i>Customize pen grip for left-handed use</i> <i>Simulate calligraphy strokes</i> <i>Check pen compatibility with paper type</i>

Table 6. Basic to complex commands for grab the pen operations