Object-Centric Agentic Robot Policies

Anonymous Author(s)
Affiliation
Address
email

Abstract

Executing open-ended natural language queries in previously unseen environments is a core problem in robotics. While recent advances in imitation learning and vision-language modeling have enabled promising end-to-end policies, these models struggle when faced with complex instructions and new scenes. Their short input context also limits their ability to solve tasks over larger spatial horizons. In this work, we introduce OCARP, a modular agentic robot policy that executes user queries by using a library of tools on a dynamic inventory of objects. The agent builds the inventory by grounding query-relevant objects using a rich 3D map representation that includes open-vocabulary descriptors and 3D affordances. By combining the flexible reasoning abilities of an agent with a general spatial representation, OCARP can execute complex open-vocabulary queries in a zero-shot manner. We showcase how OCARP can be deployed in both tabletop and mobile settings due to the underlying scalable map representation.

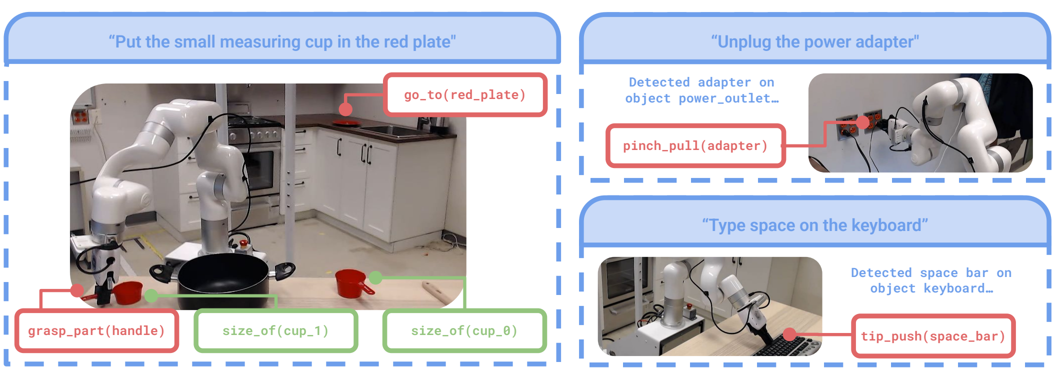

Figure 1: **OCARP** implements a language-conditioned robot policy. Leveraging foundation models for open-vocabulary perception and affordance detection, we design a general object-centric map representation that supports a compact and expressive set of tools for an LLM agent to fulfill tabletop and mobile manipulation queries expressed through natural language. The agent can handle open-vocabulary queries (e.g., `grab the panda plushie`) and reason about various relational spatial concepts (e.g., `larger/smaller`, `nearest/farthest`) using the map. Actions are carried out through interactions with relevant object parts (e.g., `handle`, `button`) and their associated affordances (e.g., `grasp_part`, `tip_push`).

1 Introduction

Generalist robot policies aim to translate complex open-ended language queries and sensor data into robot actions. Modular robotics systems historically struggled to achieve such capabilities given the complexity of explicitly modeling the relationships between language, vision and planning. Core limitations include: (i) restrictive assumptions on spatial modules, such as predefined object classes

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

or the lack of fine-grained affordances, and (ii) the inability to break down language commands into actionable steps. In this work, we address these issues with **O**bject-**C**entric **A**gentic **R**obot **P**olicies (OCARP), a modular agentic robot policy that executes user queries by applying a library of tools to a dynamic inventory of objects. The OCARP decision-making module is an LLM agent accessing a library of object retrieval, spatial understanding, and skill tools to process user queries. Underpinning these tools is a rich 3D map representation leveraging recent advancements in open-vocabulary mapping and affordance detection. The map can describe a number of interactions on any object and can naturally be extended to support reasoning over larger spatial horizons, such as rooms. In summary, our key contributions include:

1. A language-conditioned robot policy that can reason about spatial relationships and affordances in tabletop and mobile settings.
2. An interface combining flexible agentic tool calling with the general queryable capabilities of open-vocabulary maps.
3. An empirical comparison with end-to-end policies on real-world tabletop problems and additional results on mobile manipulation problems.

2 Related Work

Open-Vocabulary Mapping. Conventional semantic maps approximate object semantics using a predefined closed-set of object classes, which constrains downstream planning and scene understanding. Open-vocabulary maps remove this constraint by replacing class labels with multimodal features from foundation models such as CLIP [8, 19, 30, 10, 23, 24]. For mapping, features are typically extracted from vision sensors, grounded to a map element (point, voxel, object) and aggregated across views. At inference time, they are compared with query features that are generally extracted from a natural language query, enabling highly specific queries about objects or their properties. Open-vocabulary maps are a key module in recent queryable systems for navigation [9] and mobile manipulation [18].

Affordance Detection. Affordance detection is the task of segmenting functional elements of objects, such as handles, buttons, or knobs, that enable specific interactions [31, 20, 6, 22]. Affordances have the potential to greatly simplify the implementation of robot skills by providing direct cues about the geometry that a robot can manipulate. SceneFun3D [5] introduced a large-scale dataset with labeled affordance annotations for indoor scenes. The dataset consists of high-resolution laser scan point clouds, aligned 2D images, and associated affordance labels in the form of 3D segments and functional categories. In addition, SceneFun3D defined the task of *task-driven affordance grounding*, where the goal is to segment affordances given a task, specified in natural language. This problem is challenging for existing open-vocabulary 3D segmentation methods, which typically fail without additional fine-tuning [5]. More recent works have explored affordance detection in the context of open-vocabulary and scene-level reasoning. OpenFunGraph [36] augments scene graphs generated by ConceptGraphs with affordance information by leveraging vision-language models to generate descriptions of functional elements. Fun3DU [4] approaches affordance segmentation directly from text queries: it decomposes the input query into relevant object and part descriptions, retrieves images of the corresponding object, and uses Molmo to point at the target region for the affordance, which is subsequently segmented and lifted to 3D.

LLM Agents for Robotics. Controlling robots via language is a long-standing problem in robotics [29]. Taking advantage of the coding abilities of Large Language Models (LLMs), recent work has framed language-conditioned robot policies as a translation problem between natural language and code, effectively mapping user queries to robot API calls or LLM-generated function calls [17]. This has been followed by more reactive **agentic** approaches where LLMs interleave text generation and API calls instead of coding entire plans upfront [34, 26], enabling a tighter feedback loop with the API and existing frameworks such as ROS [25]. Our work is part of this trend: we design a general map representation that supports set of tools for the agent to execute manipulation queries without making strong perception assumptions. The ability of our framework to ground objects given a natural language query and agentic reasoning is similar to [33].

End-to-End Learning. An alternative to controlling a robots via a predefined API or tools is to directly learn a function mapping vision inputs and language commands to robot actions. Such end-to-end policies have been transformed by the emergence of Vision-Language Models (VLMs) that enable the transfer of web-scale knowledge to robot control. Examples include CLIPort [27],

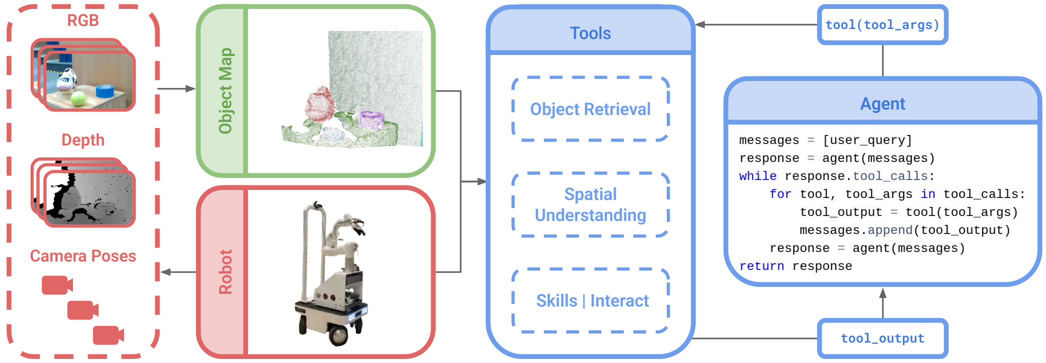

Figure 2: **Framework.** OCARP is an agentic framework for language-conditioned tabletop and mobile manipulation. **(Right)** An LLM agent breaks down the user query into a sequence of tool calls. Available tools include (i) open-vocabulary object retrieval, (ii) spatial understanding tools to measure sizes, distances and other spatial predicates, and (iii) a general interact tool that manages skill tools informed by fine-grained 3D affordances. Tools may return basic data types (e.g., `float` for a distance) or a symbolic state encoding the currently held object (if any) and an inventory of the query-relevant objects that the agent has grounded so far. **(Left)** To achieve a truly flexible language-guided policy, generalist decision-making should be paired with a general spatial representation. We build an open-vocabulary object map following [8, 19] and encode object semantics as language-aligned CLIP. We further detect affordances with foundation models and represent them using 3D point clouds, part descriptions and corresponding skills. The map plays a central role in the tool implementations and naturally allows to support reasoning and planning beyond the tabletop setting.

which leverages the general semantics of CLIP for imitation learning, and the more recent Vision-Language-Action models (VLAs) [37, 12, 2]. VLAs leverage pretrained VLMs and imitation learning to map vision and language inputs to a shared representation that can be decoded into robot actions. While enabling capable policies with zero-shot potential, VLAs still face challenges in terms of generalization to new environments and complex language understanding, often requiring some amount of fine-tuning on specific problems to perform well [13].

3 Method

OCARP implements a language-conditioned robot manipulation policy. The inputs consist of a natural language query and RGB-D frames from a wrist camera. This section contains high-level definitions of map data structures and tools, and the specific implementation details are presented in Section 4. To simplify exposition, we focus on the tabletop setting first and discuss the mobile setting in Section 3.4.

3.1 Object Map

When the agent needs to perceive the environment, we build an explicit 3D representation of the workspace. Specifically, the `ObjectMap` is a list of `Objects` which itself includes a list of `Affordances`:

```
struct Affordance:
    point_cloud: PointCloud
    part: str
    skill: Skill

struct Object:
    point_cloud: PointCloud
    rgb_crops: List[RGBImage]
    depth_crops: List[DepthImage]
    features: Vector
    affordances: List[Affordance]
```

`Object` encapsulates key perceptual elements that will be required in tool implementations. It describes different facets of an object such as

- **Geometry**: the `point_cloud` in scene frame.
- **Semantics**: the `features` are extracted and aggregated from chosen local object views in `rgb_crops` and form a shared vision-language feature space (e.g., CLIP [24]) to enable comparisons with open-ended text queries. We also store the `depth_crops` corresponding to `rgb_crops`.
- **Affordances**: affordances described in terms of their geometry (`point_cloud`), a natural language description of the relevant object part (`part`) and a `skill` that corresponds to an available skill tool (Section 3.3).

3.2 LLM Agent

The core decision-making module is an LLM agent which parses the user query and calls a sequence of tools (Figure 2). The OCARP agent is not directly exposed to sensor data or the `ObjectMap`. Instead, it perceives the environment through a symbolic state representation:

```
struct State:
    held_object: ObjectKey | None
    inventory: List[ObjectKey]
```

that is returned when calling certain tools (Section 3.3) and describes the currently held object (`held_object`) and objects that the agent can reason on or act on using other tools (`inventory`). An `ObjectKey` is a simple unique string identifier that is initially generated by the `object_retrieval` tool (Section 3.3) based on the retrieval query and the number of relevant objects (e.g., `red_ball_0`).

3.3 Tools

Tools are functions that can be called by the agent. All tools have access to the `ObjectMap` and the current `State` to implement their functionalities. The agent adds objects to its `State` inventory using the **object retrieval** tool, can reason about them using **spatial understanding** tools, and can act on them with the **interact** tool, which as access to some additonal **skill** tools. We show illustrative agent traces in Figure 3 as examples.

Object Retrieval. The object retrieval tool allows the agent to retrieve objects from the `ObjectMap` using an open-vocabulary text query:

```
function object_retrieval(
    query: str
) -> (current_state: State)
```

and adds the relevant `ObjectKeys` to `current_state.inventory`.

The OCARP agent is prompted to be specific when looking for objects and will break down user queries into multiple retrieval calls.

Spatial Understanding. Spatial understanding tools have the signature

```
function spatial(
    objects: List[ObjectKey]
) -> (output: float | bool)
```

and allow the agent to measure specific quantities (e.g., distances, sizes) or verify pairwise spatial predicates ("is on", "is left of") in the current inventory.

Skill and Interact Tools. Skill tools implement actions on inventory objects or the currently `held_object`:

```
function skill(
    obj: ObjectKey
) -> (current_state: State)
```

Skills are responsible for providing an updated `State`. For example, a successful pick will move the relevant `ObjectKey` from `inventory` to `held_object`.

Skill tools are not directly available to the main agent. Instead we expose a more general

```
function interact(
    obj: ObjectKey, action: str
) -> (current_state: State)
```

where `action` is a text description of the action the agent aims to perform. Internally, `interact` considers the requested `action` and the available object views to pick an appropriate skill tool.

Tool Preconditions and Feedback. Some tools require preconditions to be met, such as `held_object = None` when trying to pick an object or `obj in current_state.inventory` when calling spatial and skill tools. Such conditions are explicitly detailed in the tool prompts and verified in the implementations. Moreover, we wrap the output of each tool in a parsable data structure

```
struct ToolOutput:
    success: bool
    feedback_msg: str
    output: State | float | bool
```

to provide explicit feedback to the agent. `feedback_msg` can detail reasons for failures, such as unfulfilled preconditions or planning failures. The agent can leverage the feedback to reattempt some tool calls, increasing the robustness of the overall policy.

Remapping. Tools govern when the `ObjectMap` is recomputed. Calls to skill tools may trigger remapping when the location of objects becomes unknown (e.g., after failing a grasp) whereas consecutive calls to object retrieval and spatial understanding tool will reuse the same `ObjectMap`. Whenever a tool internally raises a map update, we clear the agent inventory and recompute the map on the next call to `object_retrieval` by positioning the arm at a default home pose and computing an `ObjectMap` based on the current RGB-D.

3.4 Mobile OCARP

Mobile OCARP follows the previously introduced framework with the following modifications to account for the mobile base:

- `ObjectMap` aggregates information (Section 4.1) from multiple posed RGB-D keyframes taken across a room. We do not target exploration in this work and assume the keyframes are precollected.
- The agent can navigate to inventory objects using a dedicated `go_to(obj: ObjectKey)` tool.
- Following a successful `go_to` call, we build a separate local `ObjectMap` from the current camera frame and retrieve the sought object using the same query that was used to ground the object in the main `ObjectMap`. The agent is then free to apply tabletop skill tools to this local object representation. This "redetection" renders the system more robust to possible localization and mapping errors in the main `ObjectMap`.
- We do not allow tools to update `ObjectMap` (**Remapping**), although in principle this could be achieved by revisiting keyframe locations.

4 Implementation Details

4.1 Object Map

Open-Vocabulary Object Map. We build the `ObjectMap` following recent work in open-vocabulary object-centric map representations [8, 19]. Whenever we recompute the map in the tabletop setting, we position the arm at home pose to have a good overview of the workspace and process the RGB-D wrist frame. We first run MobileSAM (in grid-sampling mode) [14, 35] to extract segmentation masks and convert them to bounding boxes. For each mask, we crop a local RGB image and embed it with CLIP to describe semantics. We also backproject the masks using the camera depth to obtain 3D point clouds.

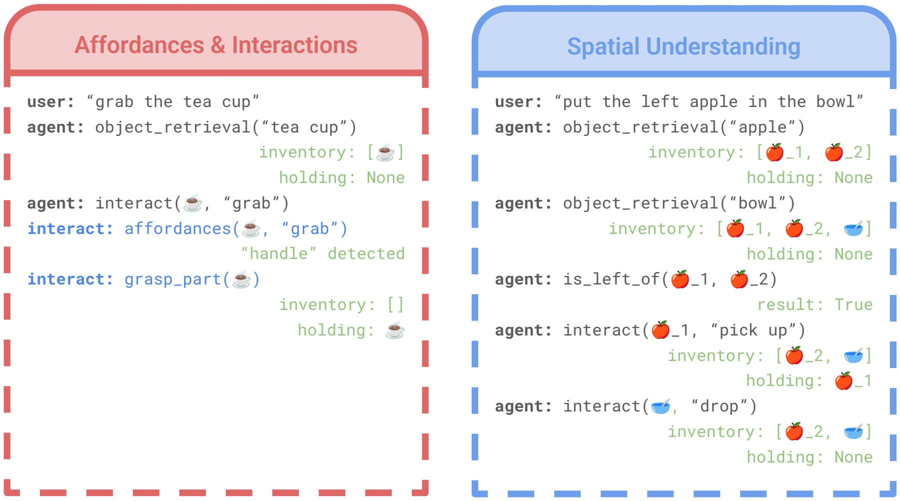

Figure 3: **Illustrative Traces.** The OCARP agent parses the user query to identify objects of interest and ground them in the map using language and the `object_retrieval` tool. This process updates a symbolic inventory with relevant objects that can be forwarded to spatial understanding and and the interact tool. **(Left)** The agent finds a tea cup and calls the interact tool, which chooses to grasp a handle on the cup. **(Right)** The agent finds two apples and a bowl before disambiguating which apple should be picked using the `left_of` tool. We omit the interact trace in this example for brevity. We use emoticons for illustrative purposes: the agent actually sees string symbols (e.g., `apple_0`).

Object Merging. This initial step yields a first set of `Objects` (with empty affordances). We then follow the merging strategy of [8] to merge different `Objects` with similar geometries (point cloud overlap) and semantics (CLIP similarities). This merging step is useful even when processing a single frame to mitigate the over-segmentation of complex objects by SAM.

When merging `Objects`, we accumulate and downsample their `point_cloud`, maintain a running average of the `features` and combine the `rgb_crops` and `depth_crops`. We sort the crops by segment area (number of pixels in the segment), with a penalty if the segment touches the image border to favor larger crops where the object is central.

Mobile OCARP. In the mobile setting, we incrementally build the `ObjectMap` and merge objects across keyframes, identical to [8].

Affordance Detection. To detect `Affordances` for an `Object` in the `ObjectMap`, we design a 2-step pipeline leveraging VLMs. First, we use GPT 4.1 mini to predict a a list of `skills` and corresponding `parts` given the best `rgb_crop` of the `Object` and the current user query. Each predicted (`part`, `skill`) pair defines a distinct `Affordance`. Second, a VLM (Gemini 2.5) prompted with the `part` and `rgb_crop` produces a bounding box for the image crop, which is used to get a 2D mask using an image segmentation model (SAM 2.1). The mask is then lifted into 3D with the corresponding `depth_crop`, yielding the `point_cloud` representation of the `Affordance`.

4.2 Agent

Agent. We implement the agent using LangChain [1] with a Gemini backend [28]. LangChain internally includes some agentic prompting and instructions.

4.3 Tools

Object Retrieval. We implement `object_retrieval` as a search for the top k similar objects in `ObjectMap` based on the similarity between the `query` CLIP features and the object `features` in the map. We then use a VLM to confirm whether each of the top k objects are relevant to the `query` or not using the best object views (`object.rgb_crops[0]`). In contrast to using a pure CLIP-based retrieval approach, this VLM classification step allows to return a variable number of relevant object without tuning a specific CLIP similarity threshold. In our experiments, we use $k = 3$ and Gemini as the VLM classifier.

Spatial Tools. We expose the following spatial understanding tools:

- `distance_to(obj) -> float`: Distance between the robot and the object centroid.
- `distance_between(obj1, obj2) -> float`: Distance between the centroids of `obj1` and `obj2`.
- `is_left_of(obj1, obj2) -> bool`: checks if `obj1` is left of `obj2`. We also expose the analogous `is_right_of`.
- `size_of(obj) -> float`: returns the size of `obj`, approximated as the axis-aligned bounding box volume.

Spatial understanding tools are implemented as basic operations on the object point clouds and their centroids.

Skill and Interact Tools. We implement all `skills` using classical motion planning. Specifically, we expose the general object skills:

- `grasp(obj)`: grasp `obj` anywhere.
- `place(obj)`: place `held_object` on `obj`.
- `drop(obj)`: drop `held_object` on `obj`.

as well as some skills inspired by the SceneFun3D affordances [5]:

- `grasp_part(obj)`: grasp a specific object part.
- `tip_push(obj)`: push on a specific object part with the tip of the gripper.
- `pinch_pull(obj)`: pinch the part with the gripper and pull.
- `hook_pull(obj)`: hook the part from above and pull.

Skill implementations generate scripted end-effector pose goals for the motion planner using a combination of operations on the object point cloud (e.g., finding the normal or a point above it) and AnyGrasp [7]. In the case of implementing skills on affordances, we derive goals using the point cloud stored in the relevant `Affordance`. We always run AnyGrasp on a context around the object and only keep the grasps that are near the object point cloud (for `grasp`) or the specific affordance point cloud (for `grasp_part`). In the case of `pinch_pull` and `hook_pull`, we use a predefined end-effector rotation to grasp the affordance centroid and infer a horizontal pulling axis using the point cloud normal.

`Skills` are available as subroutines in the `interact` tool, which calls the previously described affordance detection pipeline on the received `obj` and `action` description to infer the appropriate `skill`. This design allows skill selection to be informed by vision and influenced by the existence of a part (e.g., not all cups have handles) and their shapes (e.g., to try `pinch_pull` or `hook_pull`).

Go To Implementation. While pre-collecting keyframes in the mobile setting, we also build a 2D occupancy map using the navigation stack. Given a call to `go_to(obj)`, a target pose is selected on a radius around the object's centroid such that (i) the robot faces the centroid and (ii) its footprint remains within free space. We tune the radius to ensure that the object is accessible to the manipulator.

4.4 Robot

Hardware. Our mobile manipulator consists of a UFactory XArm6 mounted on an Agilex Ranger Mini 2.0, similar to the build proposed in [32]. We mounted an Intel Realsense D435i on the arm wrist and an Intel Realsense T265 tracking camera on the base.

Software. The robot software is integrated with ROS 2. We use MoveIt 2 [3] for the arm motion planning (RRT-Connect Planner [15]) and Nav2 [21] (Theta Star Planner) for the mobile base. SLAM is handled by RTABMap [16] using the wrist RGB-D camera and odometry estimates from the T265 and the base.

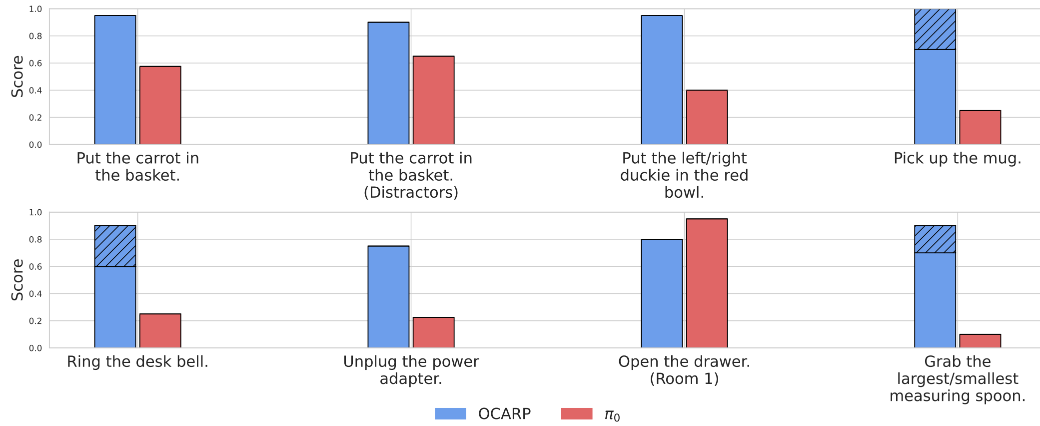

Figure 4: **Tabletop Manipulation Results.** We compare OCARP to a leading VLA [2] on 8 tabletop manipulation queries involving a variety of objects, geometric concepts (left/right, largest/smallest) and affordances. **(Score)** We report the average success rate over 10 attempts, giving partial points for intermediate steps, namely 0.25 when the gripper comes close to the relevant object without completing the grasp/place and 0.50 when the correct object is actually grasped but subsequent steps fail. **(Hatch Pattern)** We use a hatch pattern for episodes where methods use a reasonable interaction on the wrong object part or use the correct object part in an unnatural way based on human judgment. This distinction is helpful to study the part-level understanding of the different methods.

Figure 5: **Examples. (Left)** An example of a "hatched" success in Figure 4. While the handle was correctly segmented and OCARP successfully picked the mug, the grasp does not align with human expectations. **(Right)** When testing the `type space on the keyboard` query (to appear in the final manuscript), we found Gemini to be influenced by the expected location of the space bar instead of the actual location.

5 Experiments

5.1 Tabletop Manipulation

Baselines. We compare OCARP with the VLA model π_0 [2] fine-tuned on the DROID dataset [11]. Specifically, we use the `pi0-FAST-DROID` checkpoint. We chose to use a Franka Arm and the DROID setup for π_0 to minimize the risk of any distribution shift with the XArm6 (not in DROID). While π_0 does not leverage depth data, it does use two third-person cameras in addition to the wrist camera, as opposed to OCARP. Overall, we expect both arms to be equally capable on the considered tasks.

Results. We report preliminary results for 8 manipulation queries in Figure 4. OCARP outperforms π_0 on 7 of the 8 queries, showing advanced language understanding and manipulation skills. Our method also generally grabs part in the intended way (e.g., the handle on the mug) although some unnatural grasps occur (Figure 5, left). We find that most OCARP failures are owed to incorrect grasp predictions or errors in the `ObjectMap`, such as two objects being incorrectly merged together. π_0 on the other hand performs well on the drawer opening task but fails on comparatively simpler tasks, such as picking up the mug. The VLA often targets the correct object but fails to grasp it, hovering around it instead.

5.2 Mobile Manipulation

Results. We assess the performance of mobile OCARP in Figure 6. We find that the agent successfully interleaves navigation and manipulation to solve room-level problems. The double pick

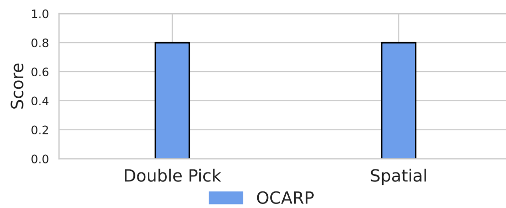

Figure 6: **Mobile Manipulation Results.** We run the mobile version of OCARP on problems staged in a medium-sized room. For each query, we teleop the robot to scan the room before launching the policy. We report the average success rate over 10 unique queries in each category. All problems include irrelevant objects in the global `ObjectMap` to stress test object retrieval. **(Double Pick)** OCARP is tasked with putting two objects in a target container. The queries cover a large variety of objects including headphones, a screwdriver, a panda plushie and a variety of toy food items. We give a score of 0.5 per object in the container and no other partial points. **(Spatial)** OCARP must disambiguate a mobile pick and place query using spatial reasoning, e.g. `"Place the egg that is near the tomato in the pan"`. We also test relationships such as left/right, nearest/farthest and smallest/largest.

queries illustrate how the agent shows some degree of planning proficiency while the spatial queries demonstrate the role of the `ObjectMap` in understanding referential queries.

5.3 Ongoing Experiments

We only presented some preliminary results in this work and are planning to benchmark OCARP against a broader range of modular and VLA methods. We also hope to test a higher number of queries and provide a quantitative assessment of our affordance segmentation pipeline.

5.4 Analysis

Strengths. We took inspiration from recent work in affordance detection [5] to design our affordances and skills and find that combining them with an LLM agent spans a useful set of language-guided manipulation behaviors. Given OCARP's reliance on vision foundation models with strong generalization capabilities, we expect reported performance to carry over to a wide range of objects. Moreover, the `ObjectMap` ensures robust spatial reasoning and, when combined with search-based motion planning, allows the agent to solve problems across extended and varied spatial layouts.

Limitations. We identify three key constraints to the OCARP behaviors. First, the OCARP agent reasons over discrete object symbols with limited state information, limiting how well it can handle situations involving granular or deformable objects. Second, while 3D affordances offer a good approximation of what should be done with an object, they do not encode how to exactly interact with a part, how to orient an object and how to place it. Finally, affordance-based skill implementations do not offer a clear avenue for solving more complex queries such as `clean the counter` or `fold the shirt`, something that can be achieved through imitation learning and VLAs (generalization notwithstanding). Nevertheless, our results show that OCARP can solve a broad range of queries over a diversity of objects, without requiring human demonstrations.

References

[1] Langchain. `https://github.com/langchain-ai/langchain`, 2022.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. *IEEE robotics & automation magazine*, 19(1):18–19, 2012.

[4] Jaime Corsetti, Francesco Giuliari, Alice Fasoli, Davide Boscaini, and Fabio Poiesi. Functionality understanding and segmentation in 3d scenes. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24550–24559, 2024. URL `https://api.semanticscholar.org/CorpusID:274233938`.

[5] Alexandros Delitzas, Ayca Takmaz, Federico Tombari, Robert Sumner, Marc Pollefeys, and Francis Engelmann. Scenefun3d: Fine-grained functionality and affordance understanding in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14531–14542, 2024.

[6] Thanh-Toan Do, Anh Nguyen, and Ian Reid. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5882–5889, 2018. doi: 10.1109/ICRA.2018.8460902.

[7] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 39(5):3929–3945, 2023.

[8] Qiao Gu, Ali Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5021–5028. IEEE, 2024.

[9] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. *arXiv preprint arXiv:2210.05714*, 2022.

[10] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Alaa Maalouf, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023.

[11] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[12] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[13] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[14] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[15] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[16] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[17] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

[18] Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

[19] Shiyang Lu, Haonan Chang, Eric Pu Jing, Abdeslam Boularias, and Kostas Bekris. Ovir-3d: Open-vocabulary 3d instance retrieval without training on 3d data. In *Conference on Robot Learning*, pages 1610–1620. PMLR, 2023.

[20] Timo Luddecke and Florentin Worgotter. Learning to segment affordances. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[21] Steven Macenski, Francisco Martin, Ruffin White, and Jonatan Ginés Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[22] Tushar Nagarajan and Kristen Grauman. Learning affordance landscapes for interaction exploration in 3d environments. *ArXiv*, abs/2008.09241, 2020. URL `https://api.semanticscholar.org/CorpusID:221246369`.

[23] Liam Paull, Sacha Morin, Dominic Maggio, Martin B¨uchner, Cesar Cadena, Abhinav Valada, and Luca Carlone. *Towards Open-World Spatial AI*. Cambridge University Press.

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[25] Rob Royce, Marcel Kaufmann, Jonathan Becktor, Sangwoo Moon, Kalind Carpenter, Kai Pak, Amanda Towler, Rohan Thakker, and Shehryar Khattak. Enabling novel mission operations and interactions with rosa: The robot operating system agent. In *2025 IEEE Aerospace Conference*, pages 1–16. IEEE, 2025.

[26] Sahar Salimpour, Lei Fu, Farhad Keramat, Leonardo Militano, Giovanni Toffetti, Harry Edelman, and Jorge Peña Queralta. Towards embodied agentic ai: Review and classification of llm-and vlm-driven robot autonomy and interaction. *arXiv preprint arXiv:2508.05294*, 2025.

[27] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pages 894–906. PMLR, 2022.

[28] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[29] Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):25–55, 2020.

[30] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024.

[31] Patrick Henry Winston, Boris Katz, Thomas O. Binford, and Michael R. Lowry. Learning physical descriptions from functional definitions, examples, and precedents. In *AAAI Conference on Artificial Intelligence*, 1983. URL `https://api.semanticscholar.org/CorpusID:7856739`.

[32] Haoyu Xiong, Russell Mendonca, Kenneth Shaw, and Deepak Pathak. Adaptive mobile manipulation for articulated objects in the open world. *arXiv preprint arXiv:2401.14403*, 2024.

[33] Jianing Yang, Xuweiyi Chen, Shengyi Qian, Nikhil Madaan, Madhavan Iyengar, David F Fouhey, and Joyce Chai. Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7694–7701. IEEE, 2024.

[34] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[35] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.

[36] Chenyangguang Zhang, Alexandros Delitzas, Fangjinhua Wang, Ruida Zhang, Xiangyang Ji, Marc Pollefeys, and Francis Engelmann. Open-vocabulary functional 3d scene graphs for real-world indoor spaces. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19401–19413, 2025. URL `https://api.semanticscholar.org/CorpusID:277314071`.

[37] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: OCARP achieves the claims and the abstract and we support the claims with preliminary results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We identify key limitations at the end of Section 5.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: We do not provide theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide enough details to reproduce the overall methodology and will release code upon acceptance of the final paper.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide enough details to reproduce the overall methodology and will release code upon acceptance of the final paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [NA]

   Justification: We do not train models in this work.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We provide preliminary experimental results only.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [NA]

   Justification: We do not train models in this work and mainly use APIs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: As far as we can tell, we conform to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: While our work is not anticipated to have direct societal impacts, we recognize the broader risks and challenges associated with automation and potential job displacement.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

Justification: The pretrained models we use are already publicly available.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all packages and software that contributed to our stack.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We are not releasing new assets at this time.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We did not crowdsource data or used human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: None of our experiments required an IRB approval.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We use an LLM agent throughout this work and provide details on our use.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.