Discovering Object Attributes by Prompting Large Language Models with Perception-Action APIs

Angelos Mavrogiannis, Dehao Yuan, Yiannis Aloimonos

Abstract— There has been a lot of interest in grounding natural language to physical entities through visual context. While Vision Language Models (VLMs) can ground linguistic instructions to visual sensory information, they struggle with grounding non-visual attributes, like the weight of an object. Our key insight is that non-visual attribute detection can be effectively achieved by active perception guided by visual reasoning. To this end, we present a perception-action programming API that consists of VLMs and Large Language Models (LLMs) as backbones, together with a set of robot control functions. When prompted with this API and a natural language query, an LLM generates a program to actively identify attributes given an input image. Offline testing on the Odd-One-Out (O^3) dataset demonstrates that our framework outperforms vanilla VLMs in detecting attributes like relative object location, size, and weight. Online testing in realistic household scenes on AI2-THOR and a real robot demonstration on a DJI RoboMaster EP robot highlight the efficacy of our approach.

I. INTRODUCTION

Robots may need to understand natural language and execute verbal instructions from novice users to be valuable assistants in a household. Connecting natural language instructions to the physical world often requires robots to detect object attributes in order to discriminate between candidate objects. This is a challenging problem as instructions might be linguistically ambiguous, for example "Can you please get me the second mug from the right on that shelf?". Identifying attributes can also be required implicitly to determine the state or affordance [12] of an object in order to verify the feasibility of an action. These attributes might not be directly perceivable through vision sensors, for example "Is this lightweight enough to pick up?".

In this paper, we focus on identifying object attributes in a programmatic fashion. Attributes usually appear in the form of descriptive adjectives, some of which might not be adequately represented in the training sets of data-driven perceptual models. Furthermore, while attributes might not necessarily characterize an object in an absolute scale, they are often applicable based on context [32]. We argue that attribute detection is highly contextual - an analogy of J. R. Firth's famous quotation "You shall know a word by the company it keeps" [11] applies to attributes. More specifically, characterizing an object as big, tall, or heavy can sometimes depend on the other objects and their respective attribute

The authors are with the Department of Computer Science, University of Maryland, College Park, 8125 Paint Branch Dr, College Park, MD 20742, USA. {angelosm, dhyuan, jyaloimo}@umd.edu

Our code, dataset, and robot demonstration video can be found here: https://github.com/angmavrogiannis/Embodied-Attribute-Detection.
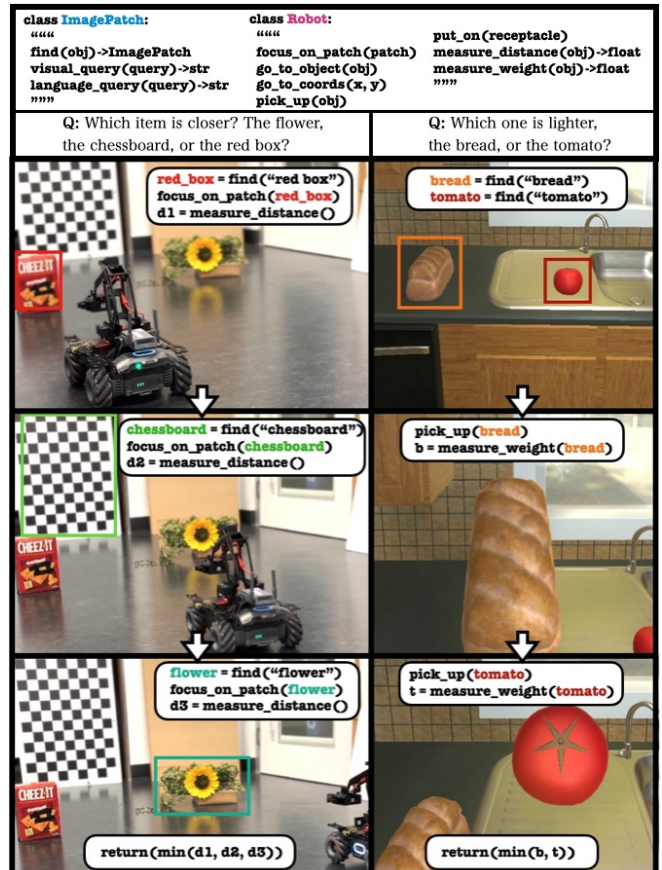

Fig. 1: Demonstration of our perception-action API solving a minimum distance query on a real robot (*left*) and a minimum weight query in simulation (*right*). The LLM receives a perception-action API and a natural language query as input (*top*). It then generates code that invokes API functions leveraging on-board sensors (camera, distance sensor, force/torque sensor) to actively identify these attributes.

values in the current environmental context. Ambiguity can also arise due to occlusion or partial observability [45]. These problems might not occur when studying attribute detection in static images, but can be common in a household environment where a robot is tasked with executing user instructions. In these cases, erroneous attribute detection can be detrimental, producing and executing an action plan involving an incorrect object, or misinterpreting the affordance of an object and failing to even execute the action.

Existing attribute detectors [3, 5, 7, 14, 33, 40, 45] are mainly obtained by either supervised training [35] or contrastive pre-training [34]. While attribute detection is an

active area of research, it is often studied separately from embodied reasoning. To bridge this gap, we model attribute detection as visual reasoning with programs. This provides us with a powerful representation for reasoning in the presence of embodied agents and allows us to utilize the space of plans and movements via robot actions as programs [36].

Summarizing these ideas, our main observation is that modern real-world robotic systems relying on visually-driven attribute detection using VLMs in isolation can be myopic in language grounding. Our key insight is that combining different VLMs as visual reasoning functions with a robot control API can benefit from the code synthesis and commonsense reasoning capabilities of LLMs to actively reason about attribute detection in the form of computer programs. We prompt an LLM with an attribute detection API on a dataset that we curate, consisting of embodiment-crucial **location**-, **size**-, and **weight**-related attributes and construct a perception-action API for active attribute detection. Our key contributions can be summarized as follows:

- We highlight some of the drawbacks of using VLMs for attribute detection in isolation and the complementary reasoning capabilities that emerge by reasoning in the form of LLM-generated visual programs.
- We construct a perception-action API by integrating visual reasoning with robot control functions and demonstrate its benefits by invoking active perception behaviors towards solving attribute detection queries.
- We release an end-to-end framework that integrates this perception-action API on a real robotic platform using visual servoing-based control.

II. RELATED WORK

Attribute Detection: Attribute detection has been a fundamental problem in the computer vision community with early work [9, 10, 27, 35] on learning visual attribute classifiers to describe unseen objects. There has been work on identifying relative attributes [8, 32] but these approaches require prior training and can be limited to a certain domain ([8] demonstrate relative attributes of shoes and facial characteristics), while we show a general method that works in a zero-shot fashion. Recent approaches have applied Open-Vocabulary object Detection (OVD) task [5, 7] to attribute detection. The goal in OVD is to detect unseen classes of objects defined at inference time in the form of textual queries. Bravo et al. [5] showed that the performance of various VLMs in zero-shot attribute detection is still low compared to OVD. However, most of these approaches focus on visually-perceivable attributes in disembodied settings. On the other hand, we focus on embodiment-crucial attributes such as the weight of an object, leveraging the physical reasoning capabilities of VLMs [43]. Our simulations in AI2-THOR and our robot demonstration shows that our end-to-end framework can invoke active perception behaviors to reason about object attributes, inspired by prior work [40, 45].

LLMs as Embodied Agents: LLMs have shown remarkable performance in translating natural language instructions to robot actions that are admissible in a given environment [17–20, 30, 31, 42]. This has been achieved predominantly by a few-shot prompting scheme where the LLM receives a set of examples of sample tasks and action plans as input, and generates an action plan for an unseen task at inference time. While some of these works rely on the inherent commonsense reasoning capabilities of LLMs to map natural language to actions [18–20], a line of work expressing action plans as programs [17, 30, 36, 42] has been using programming language constructs to elicit more profound reasoning capabilities such as action precondition checking through conditional and assertion statements [36], reasoning about task execution using control flow tools [42] and recursively defining undefined functions [30], or simply invoking VLMs in task execution functions [17]. However, these approaches focus on action planning using OVD methods in isolation [13, 22, 24, 34] and might not be able to handle attribute detection in challenging scenarios. Similar to [4], our approach combines the expressiveness of an intermediate programmatic representation and the complementary reasoning capabilites of LLMs and VLMs [44] to reason about attribute detection under an LLM-prompting scheme.

Visual Reasoning with Programs: Generating and executing programs for vision applications originated from Neural Module Networks (NMNs) [2, 16, 21], on the basis of the idea that complex vision tasks are fundamentally compositional. Motivated by this idea, NMNs decompose a task into trainable modules that learn specific perceptual functions. However, these models produce domain-limited programs, rely on hand-tuned parsers [2] or are difficult to optimize [16, 21]. To overcome these shortcomings, a recent line of work has proposed a formulation of generating visual programs to deal with image-based natural language queries through in-context learning with an LLM. The programs consist of pseudocode instructions [15] or executable python code [37–39] and intermediate variables that map to computer vision models, image processing subroutines, or LLMs. These intermediate variables are consumable downstream and illustrate a step-by-step reasoning process towards the task at hand, which is primarily related to language grounding or Visual Question Answering (VQA). Our approach invokes active perception robot behaviors guided by visual programming towards attribute detection.

III. METHOD

A. Problem Statement

Consider a robot equipped with a set of sensors $\mathcal{S}$ in a scene with a set of objects $\mathcal{O}$. The robot is tasked with executing a natural language instruction $inst = f(a, g, o, img)$, where a is a high-level action, g is an object attribute, o is an object, and img is an input image. Our goal is to determine whether an object exists with this attribute, expressed by the predicate $g(o)$, and localize it in img by obtaining its bounding box coordinates $\mathcal{X} = \{x_{min}, y_{min}, x_{max}, y_{max}\}$. If $\exists o$, such that $g(o)$ holds, a visual navigation policy $\pi(a(\mathcal{X}))$ is deployed, which allows the robot to leverage its sensors S, navigate and manipulate object o given its 2D bounding box coordinates $\mathcal{X}$ towards task completion.
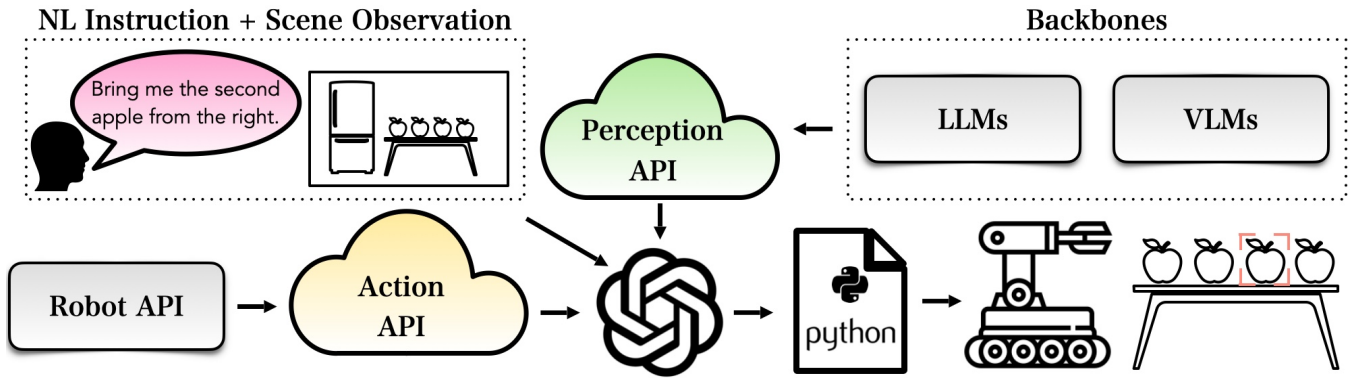
Fig. 2: We describe our end-to-end framework for embodied attribute detection. The LLM receives as input a perception API with LLMs and VLMs as backbones, an action API based on a Robot Control API, a natural language (NL) instruction from a user, and a visual scene observation. It then produces a python program that combines LLM and VLM function calls with robot actions to actively reason about attribute detection.

We adopt a generalized definition of an attribute, viewing it as an abstract property of an object that does not necessarily map to a visual representation, including primarily descriptive adjectives related to the size (*big*) or weight (*heavy*) of an object, but also prepositional phrases indicating spatial relationships (*the second object from the left*).

B. Prompt-based Attribute Detection

We adopt the methodology of Surís et al. [39] into constructing a Python API for attribute detection. The API consists of a main `ImagePatch` class that is instantiated by an input image img. `find` is the fundamental function of the API that uses an OVD model (MM-Grounding-DINO [46] or GLIP [29]) to locate an object o and return the detection-resulting cropped patch $\mathcal{X}$ from the image given a chunk of natural language $inst$. While we do not explicitly define a function for spatial reasoning, we provide in-context examples encoded in the docstring of the function. The examples perform pixelwise math given bounding box coordinates $\mathcal{X}$ of detected objects $\mathcal{O}$ that are returned from calls to `find` in order to reason about relative object locations on the image frame. `visual_query` calls a pre-trained VLM (BLIP-2 [28]) to provide a textual answer to a visual query given an image. `language_query` recursively calls the LLM with a textual query such as the visually-extracted return value from `visual_query`. This API can be viewed as an internal dialogue between VLMs and LLMs, similar to the idea of Socratic Models [44], but enhanced by structural programming tools hosted on a pythonic platform. In the following subsections we describe the complementary reasoning capabilities that emerge from calling these functions.

C. Programmatic Reasoning

Reasoning in the form of programs inherits the expressiveness of programming languages through control flow tools, data structures, and built-in methods. LLM-generated programs invoke loops to iterate over detected object patches and conditional statements to determine whether an object exists ($\exists o$) in the input image img and whether it possesses an attribute g, grounding the predicate $g(o)$. Python lists are used to store instances of image patches dynamically with the `append` function. Other built-in functions such as `sort` and the `lambda` function utilize horizontal and vertical coordinates of the detected bounding boxes and their centroids within simple mathematical operations and leverage basic geometrical notions (e.g. computing the area of an image patch) to reason about the size or relative position of detected entities. The generated code is interpretable and mainly consists of elementary arithmetic in the image frame. The commonsense reasoning component of the LLM proves to be essential in mapping complex language queries to these computations, as well as adapting attribute interpretation to image-specific contexts, as shown in Sec. IV-E.

D. Vision-Informed Language Reasoning

Combining LLMs and VLMs in the input API prompt unlocks complementary reasoning capabilities through information passing between different model calls in the the context of an LLM-generated program. This model interplay can be particularly efficient when dealing with non-visually perceivable attributes, such as estimating the weight of an object. In this case, `visual_query` can serve as a zero-shot object recognition function, subsequently passing information to the input of the `language_query`, which deduces factual knowledge on the weight of the recognized object, as explained in Sec IV-B. Similarly, when a task instruction involves an attribute that does not typically describe an object in an absolute scale, recognizing adjacent objects in addition to the object at hand establishes context. Then, posing a textual query with this visually-acquired information might reduce ambiguity and provide the correct grounding thanks to the domain knowledge of LLMs. We demonstrate an empirical evaluation of such use cases in Sec. IV.

E. Embodied Attribute Detection

Attribute detection in embodied settings often requires active perception. To this end, we formulate an action-perception API by integrating the attribute detection API with a high-level robot control API (See Fig. 2). The robot control API is implemented as a `Robot` Python class that consists of sensors as member variables and methods that map to simple navigation and pick-and-place actions. The

robot can navigate to an object with the `go_to_object` function which implements a visual navigation policy by calling `go_to_coords` with an image patch $\mathcal{X}$ as a parameter. `pick_up` and `put_on` implement picking and placing actions. Assuming the lack of an on-board RGB-D camera or a depth estimation model, a robot could employ additional sensors to measure the distance to an object in order to reason about scene geometry or depth. This can be achieved by the `measure_distance` function which calls `focus_on_patch`, a function that aligns the geometric center of the image frame to an object patch and can then retrieve the distance sensor measurement to compute the distance from the camera to that object. A demonstration on a real robot is shown in Fig. 1 (*left*). Similarly, `measure_weight` can measure the weight of an object grasped by the robot, under the precondition that the robot first navigates and picks it up. These preconditions are encoded in the example use of the function in a docstring. We integrate this perception-action API into an AI2-THOR simulated environment and a real robot and demonstrate its benefits in Sec. IV-E.

IV. EVALUATION

We design a set of experiments to showcase some of the drawbacks of using OVD or VQA models for attribute grounding in isolation and we highlight the complementary commonsense reasoning that emerges by visual reasoning with LLM-generated programs including actions.

A. Spatial Reasoning

We evaluate the spatial reasoning capabilities of the attribute detection API by comparing its ability to ground linguistically complex spatial queries with an open-vocabulary object detector [29]. We manually craft a dataset that consists of 200 challenging spatial queries based on the Odd-One-Out (O^3) Dataset [26]. Every image in this dataset includes multiple instances of an object or similar objects with an instance being slightly different to stand out. We leverage the multiple instances of an object to invoke reasoning that requires differentiating between objects based on their relative attributes rather than obvious qualitative differences between entirely different objects. Therefore, instead of focusing on relative attributes that localize the object with respect to another object of different type in the image [23, 39] (e.g. "*the car to the left of the tree*"), our 100 **location** queries require commonsense reasoning in the form of counting and establishing the relative order of an arranged set of objects, such as "*second umbrella from the left at the second to last row*" or "*the window in the middle at the bottom*". Our 100 **size** queries utilize descriptive size-related adjectives (long, wide, short, large etc.) in their superlative and absolute form, such as "*the tallest item*" or "*the wide line*", respectively. We test the same queries on both forms and expect that the superlative form will outperform the absolute, forcing a specific object to stand out by emphasizing its attribute. We anticipate that the attribute detection API will outperform OVD by incorporating pixelwise mathematical operations and expressive python utility functions.
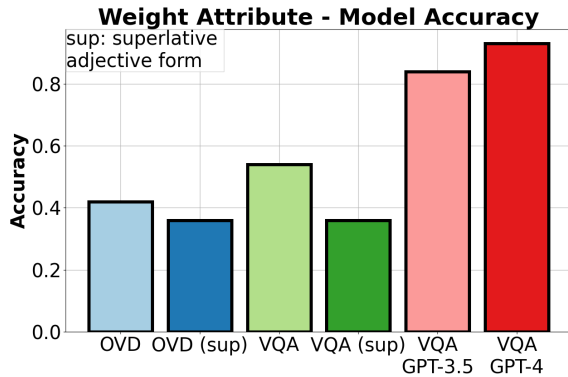

Fig. 3: The accuracy of OVD (GLIP), VQA (BLIP-2), and VQA+GPT in determining the heaviest object in an image.

B. Non-visually Perceivable Attributes

To evaluate more profound reasoning capabilities, we focus on the **weight** of an object as a representative sample of non-visually perceivable attributes, as it is crucial for executing essential manipulation tasks. In the absence of a dataset with a suitable schema for our use case, we prompt GPT-4 to generate sets of objects of different weight, along with the ground truth label of the heaviest object. To simplify the task, we design the prompt so that the object weight distribution is monotonic and clearly distinguishable by a human observer: *Generate* 100 *triplets of objects where each object is significantly heavier than the other (for example: feather, dog, car)*. After acquiring the generated textual data, we utilize it to extract relevant images from the web and arrange them to form an image dataset where each data sample is an image that includes three objects of monotonically decreasing weight. Assuming some rudimentary commonsense reasoning functionality in VLMs [6], we expect that they are capable of identifying obvious differences in weight and hence dealing with examples that are intuitive to humans, such as selecting the heaviest object between a handbag, a kangaroo, and a bus (See Fig. 5). We compare the performance of OVD: `find("a heavy object")`, VQA: `visual_query("Out of these items, which one is the heaviest?")`, and vision-informed language reasoning (VQA+GPT) that is invoked by our prompt API: `visual_query("What are the items in this image?")` $\rightarrow$ `language_query("Out of these items, which one is more likely to be the heaviest one?")`.

C. Evaluation in Embodied Settings

To evaluate our perception-action API in embodied settings, we adapt it to a simulated AI2-THOR [25] household environment. We assume that the robot comes with a proximity sensor and a force/torque sensor mounted on the wrist of the gripper, capable of measuring the weight of an object. To replicate the behavior of these sensors, we query the simulator for the distance between an object centered on the frame captured by the on-board robot camera, and build a queryable dictionary that maps an object to an approximate weight when the robot is holding that object. We measure the accuracy (%) of our perception-action API
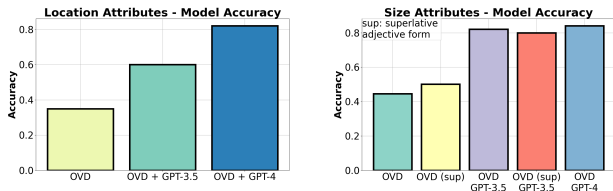
Fig. 4: We compare the accuracy of OVD-only (GLIP) with (OVD+GPT) on our **location** (*left*) and **size** (*right*) datasets.

in estimating the relative distances of objects from the robot camera, and identifying the most lightweight object. Our baselines are OVD, VQA, GPT-4o, and the attribute detection API (VQA/OVD+GPT-4). We use the following prompt templates: "*Out of the {objects}, which one is closer to me?*", "*Out of the {objects}, which one is the most lightweight?*". We anticipate that the LLM-generated programs from the perception-action API are capable of actively interacting with the environment to identify object attributes leveraging sensor-powered visual reasoning functions and robot actions.

D. Hypotheses

We formalize these insights into the following hypotheses:

H1: OVD+GPT outperforms OVD- and VQA-only baselines in **location**- and **size**-related queries.

H2: VLMs possess the rudimentary reasoning capability to tackle evident **weight** estimation queries.

H3: The superlative form of a descriptive adjective yields a better grounding performance than the absolute form.

H4: Our perception-action API solves attribute detection queries by actively interacting with the environment.

To evaluate these hypotheses, we measure the grounding accuracy by comparing the bounding boxes returned by OVD and OVD+GPT. For OVD, we report results from GLIP [29] since it outperforms MM-Grounding-DINO [46] on our data. In the case of the weight attribute, we additionally consider the textual output of VQA+GPT. We demonstrate results from GPT-3.5, GPT-4, and GPT-4o in the embodied settings.

E. Results & Discussion

Based on the results in Fig. 3, Fig. 4, and Table I we make the following observations regarding our hypotheses.

H1: Our first hypothesis is confirmed. We find that OVD+GPT significantly outperforms OVD (See Fig. 4) in **location** queries by 134% and in **size** queries by 67%. Qualitative examples are shown in Fig. 6. In example *a*, OVD and OVD+GPT correctly identify the apple as the small fruit. Example *b* shows a common mutual failure case where the bounding box for one paper clip mistakenly includes all of them. We also observe failure cases due to occlusion. Examples *c*, *d* show cases where OVD+GPT outperforms OVD. In these examples, all instances of the object in the query are localized with the `find` function, and then the coordinates of the bounding boxes and their centroids are used to compute relative distances and areas and sort them, if needed. In terms of the **size**-related queries, GPT-4 can adapt the generated code to the context of an image and interpret what dimension *long* or *short* corresponds to based
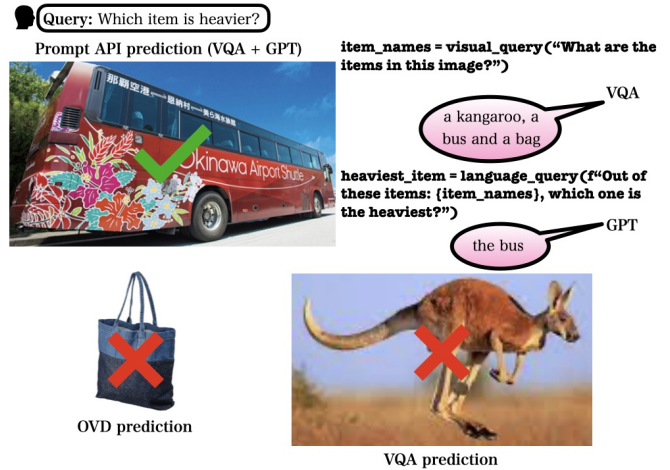

Fig. 5: An example where OVD and VQA fail to identify the heaviest object in the image (✗), but the API prompt-generated code (VQA+GPT) returns the correct answer (✓).
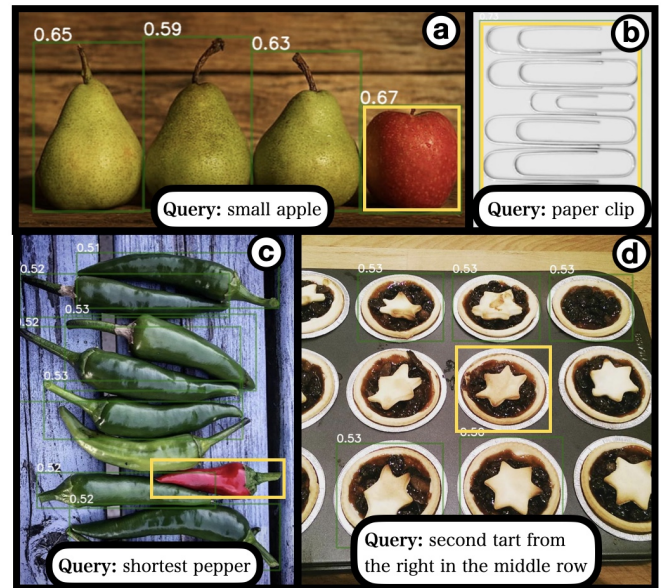

Fig. 6: OVD (numbers in white denote the OVD confidence score) and OVD+GPT predictions are shown with green and yellow bounding boxes, respectively. *a* shows an example of agreement between OVD and OVD+GPT, *b* a mutual failure case, and *c*, *d* show cases where OVD+GPT exhibits superior performance compared to OVD.

on the orientation of an object in an image. On the other hand, GPT-3.5 is more rigid and tends to tie certain adjectives to hardcoded dimensions following common norms. For example, in Fig. 6 (*c*), it cannot connect the adjective *short* to the red pepper, since it is horizontally aligned. Such minor details explain the slight discrepancy in performance, which is still superior than vanilla OVD when identifying attributes with an attribute detection API. We would need to add targeted examples in the prompt API covering all failure cases to induce equal accuracy from both models. Finally, in example *d*, OVD+GPT-4 understands that the

Method	Task	
	Weight	Distance
OVD	0.14	0.64
VQA	0.64	0.56
Attribute Detection API	0.90	0.22
GPT-4o	0.88	0.70
Perception-Action API	**0.96**	**0.94**

TABLE I: Performance of our perception-action API in 50 weight and 50 distance estimation queries in simulated AI2-THOR [25] household environments against baselines.

arrangement of the tarts is forming a row and column pattern. On the contrary, OVD fails to recognize this pattern, and OVD+GPT-3.5 exhibits a context-agnostic interpretation of rows, dividing the image into parts based on the image height, which yields incorrect results.

H2: Our second hypothesis is only partially confirmed. Based on Fig. 3, the combination of VQA, followed by a call to an LLM, significantly outperforms OVD-only (+121%) and VQA-only (+72%) solutions although we expected that all the models would be able to handle simple comparative attribute detection tasks. We believe that the step-by-step reasoning process followed by the prompt API leverages the strength of each model separately. On the contrary, burdening a model with additional reasoning tasks upon the ones that it was naturally tasked with in the first place (zero-shot language-conditioned object detection for GLIP, and zero-shot object recognition for BLIP-2) might be the reason we are missing out on its full task-specific potential. Another potential reason for this discrepancy in performance is that the pre-training objective of these models might not be aligned to our specific use case, which is identifying non-visually perceivable attributes.

H3: Our third hypothesis is rejected, mainly because the absolute form of an adjective yields better performance in the weight estimation task (See Fig. 3). In Fig. 3, Fig. 4, *sup* stands for superlative adjective form in the prompt. GPT-4 demonstrates the same performance for both forms, therefore we report a joint measurement for this model in Fig. 4.

H4: Confirming our hypothesis, our perception-action API solves both tasks and outperforms all baselines based on the results shown in Table I. In distance estimation, the robot first identifies an object patch with `find`, and then leverages its distance sensor by focusing on the detected image patch with `focus_on_patch` and calling `measure_distance` to get the measurement. In weight estimation, after locating the patch with `find`, the robot navigates (`go_to_object`) and proceeds to `pick_up` every object and measure its weight using the force/torque sensor by calling `measure_weight`. At every measurement, the generated programs compare the currently measured value with a previously stored minimum and update it if the current value is lower, finally yielding the minimum distance or weight. In some cases, `find` cannot locate the object patches leading to failures. We believe that this occurs because of the image quality of the simulated objects, which are not as realistic as the ones in real-world images and the OVD model that `find` uses struggles to locate them. In distance estimation, the generated code by the

attribute detection API (OVD+GPT-4) incorrectly hardcodes the distance from the object to the robot camera to the distance of the object from the geometric center of the image frame, leading to a very low accuracy.

F. End-to-End Framework - Robot Demonstration

We integrate our perception-action API into a real robot by implementing a wrapper over a robot-specific API. We deploy the combined end-to-end framework on DJI® RoboMaster™ EP [1], an affordable ground robot with holonomic movement and pick-and-place capabilities. A demonstration of our framework in action is shown in Fig. 1. Picking-and-placing an object first requires navigating in front of it with the appropriate orientation (`go_to_object`). To this end, we leverage sensory information to design a control policy for implementing the `go_to_object` function to navigate to a detected object. The control policy is further divided into two sub-policies: i) a visual servoing-based control policy for the lateral movement that aligns the center of the patch of the detected object to the center of the image frame captured by the on-board robot camera (`focus_on_patch`), ii) a control policy for the longitudinal movement that steers the robot towards a proximal position to the object at hand based on an infrared distance sensor. The target distance from the gripper to an object is a pre-computed functional gap, or in other words an experimentally determined *sweet spot* for picking and placing. Each of these policies is separately handled by a hand-tuned Proportional-Integral-Derivative (PID) controller. The robot is connected to a (*local*) computer via wifi connection and communicates with a (*remote*) computing cluster through a client-server architecture running on an SSH tunnel. To reduce latency due to the computational load of deploying a VLM on the cluster, we only run OVD on the first frame captured by the robot camera, and then track the corresponding position(s) with the Kanade–Lucas–Tomasi (KLT) feature tracker[1] [41].

V. LIMITATIONS & FUTURE WORK

Sensor Integration: Our experiments provide some insights on failure cases and emerging reasoning capabilities of VLMs in attribute detection. We demonstrate the applicability of our action-perception API on a robot in simulation and in the real world. In the future we plan to leverage the compositionality of our API and extend its sensing capabilities by incorporating more sensors (e.g. IMU, temperature sensor) via wrapper functions, supporting the discovery of additional attributes through active perception.

Error Propagation across Model Calls: In Sec. IV-E we showed how the attribute API (VQA+GPT) outperforms calling an OVD or VQA model in isolation. However, if the first call yields an incorrect result, any downstream calls consume erroneous parameters and hence lead to an incorrect final result. In the future we plan to develop mechanisms that leverage additional feedback from the environment to catch such exceptions before errors propagate downstream.

[1] We follow the implementation in https://github.com/ZheyuanXie/KLT-Feature-Tracking.git.

REFERENCES

[1] DJI RoboMaster EP. https://www.dji.com/robomaster-ep. Accessed: 2024-02-01.

[2] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016.

[3] L. Bianchi, F. Carrara, N. Messina, C. Gennaro, and F. Falchi. The devil is in the fine-grained details: Evaluating open-vocabulary object detectors for fine-grained understanding. *arXiv preprint arXiv:2311.17518*, 2023.

[4] H. Biggie, A. N. Mopidevi, D. Woods, and C. Heckman. Tell me where to go: A composable framework for context-aware embodied robot navigation. In *7th Annual Conference on Robot Learning*, 2023.

[5] M. A. Bravo, S. Mittal, S. Ging, and T. Brox. Open-vocabulary attribute detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7041–7050, 2023.

[6] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[7] K. Chen, X. Jiang, Y. Hu, X. Tang, Y. Gao, J. Chen, and W. Xie. Ovarnet: Towards open-vocabulary object attribute recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23518–23527, 2023.

[8] S. Chen and K. Grauman. Compare and contrast: Learning prominent visual differences. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1267–1276, 2018.

[9] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *2009 IEEE conference on computer vision and pattern recognition*, pages 1778–1785. IEEE, 2009.

[10] V. Ferrari and A. Zisserman. Learning visual attributes. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[11] J. Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, pages 10–32, 1957.

[12] J. J. Gibson. *The ecological approach to visual perception: classic edition*. Psychology press, 2014.

[13] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Open-vocabulary object detection via vision and language knowledge distillation. *International Conference on Learning Representations*, 2022.

[14] X. Guo, K. Chen, J. Rao, Y. Zhang, B. Sun, and J. Yang. Lowa: Localize objects in the wild with attributes. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

[15] T. Gupta and A. Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962, 2023.

[16] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 804–813, 2017.

[17] S. Huang, Z. Jiang, H. Dong, Y. Qiao, P. Gao, and H. Li. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *arXiv preprint arXiv:2305.11176*, 2023.

[18] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022.

[19] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[20] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. T. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K.-H. Lee, Y. Kuang, S. Jesmonth, N. J. Joshi, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 205, pages 287–318, 2023.

[21] J. Johnson, B. Hariharan, L. Van Der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE international conference on computer vision*, pages 2989–2998, 2017.

[22] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, and N. Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021.

[23] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014.

[24] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[25] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, et al. Ai2-THOR: An interactive 3d environment for visual AI. *arXiv preprint arXiv:1712.05474*,

2017.

[26] I. Kotseruba, C. Wloka, A. Rasouli, and J. K. Tsotsos. Do Saliency Models Detect Odd-One-Out Targets? New Datasets and Evaluations. In *British Machine Vision Conference (BMVC)*, 2019.

[27] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE conference on computer vision and pattern recognition*, pages 951–958. IEEE, 2009.

[28] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[29] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975, 2022.

[30] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[31] A. Mavrogiannis, C. Mavrogiannis, and Y. Aloimonos. Cook2ltl: Translating cooking recipes to ltl formulae using large language models, 2023.

[32] D. Parikh and K. Grauman. Relative attributes. In *2011 International Conference on Computer Vision*, pages 503–510. IEEE, 2011.

[33] K. Pham, K. Kafle, Z. Lin, Z. Ding, S. Cohen, Q. Tran, and A. Shrivastava. Learning to predict visual attributes in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13018–13028, 2021.

[34] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[35] O. Russakovsky and L. Fei-Fei. Attribute learning in large-scale datasets. In *Trends and Topics in Computer Vision: ECCV 2010 Workshops, Heraklion, Crete, Greece, September 10-11, 2010, Revised Selected Papers, Part I 11*, pages 1–14. Springer, 2012.

[36] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[37] A. Stanić, S. Caelles, and M. Tschannen. Towards truly zero-shot compositional visual reasoning with llms as programmers. *arXiv preprint arXiv:2401.01974*, 2024.

[38] S. Subramanian, M. Narasimhan, K. Khangaonkar, K. Yang, A. Nagrani, C. Schmid, A. Zeng, T. Darrell, and D. Klein. Modular visual question answering via code generation. *arXiv preprint arXiv:2306.05392*, 2023.

[39] D. Surís, S. Menon, and C. Vondrick. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*, 2023.

[40] G. Tatiya, J. Francis, H.-H. Wu, Y. Bisk, and J. Sinapov. Mosaic: Learning unified multi-sensory object property representations for robot perception. *arXiv preprint arXiv:2309.08508*, 2023.

[41] C. Tomasi and T. Kanade. Detection and tracking of point. *Int J Comput Vis*, 9(137-154):3, 1991.

[42] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor. Chatgpt for robotics: Design principles and model abilities. *Microsoft Auton. Syst. Robot. Res*, 2:20, 2023.

[43] Y. Wang, J. Duan, D. Fox, and S. Srinivasa. NEWTON: Are large language models capable of physical reasoning? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9743–9758, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp. 652. URL https://aclanthology.org/2023. findings-emnlp.652.

[44] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

[45] X. Zhang, S. Amiri, J. Sinapov, J. Thomason, P. Stone, and S. Zhang. Multimodal embodied attribute learning by robots for object-centric action policies. *Autonomous Robots*, pages 1–24, 2023.

[46] X. Zhao, Y. Chen, S. Xu, X. Li, X. Wang, Y. Li, and H. Huang. An open and comprehensive pipeline for unified object grounding and detection. *arXiv preprint arXiv:2401.02361*, 2024.