OpenD: Accurately Open the Door and Drawer

Yizhou Zhao[1], Qiaozi Gao[2], Govind Thattai[2], Gaurav S. Sukhatme[2,3]

***Abstract*— We focuses on a seemingly simple but instructive task: opening a cabinet by hand. To deploy this task in a virtual environment, we present OPEND, an open-source benchmark to simulate the door and drawer opening task for a cabinet in a photo-realistic and physics-reliable setting. We train baseline models based on a universal multi-step planner to understand semantic instructions and spatial relations. The evaluation of models lies in the zero-shot performance on opening novel cabinets from language instructions. Results show that decision planning from a multi-step planner for different types of hands performs drastically differently, suggesting that there is significant room for developing innovative models that learn to solve this everyday task with our benchmark.**

I. INTRODUCTION

In robotics, it requires much effort to set up an object manipulation task in a real scene. Recently, involving the development of simulation engines, the Embodied AI research pushes intelligent robotic systems to reality closer than ever before. Thanks to manipulation benchmarks like VLMBench [1], ManipulaTHOR [2], ManiSkill [3], new models and algorithms are emerging to help robotics research in object manipulation, aiming to overcome the domain gap from virtual to physical spaces.

However, most existing benchmarks in the simulation engine attempt to cover a wide range of tasks but simplify the process of manipulation. The simplification is usually achieved by changing the object size, de-emphasizing object collision, or abstracting hand grasping [4]. Instead, our work concentrates on only one task (*opening a cabinet*) but tries to bring as much realism, scalability, and interactivity w.r.t. physical reality and task complexity.

To enhance the quality of simulation scenes with realism, we apply NVIDIA OMNIVERSE(citation?),an easily extensible platform for 3D design and simulation, to obtain real-time and true-to-reality simulation. It enables a physics-realistic task setting and helps to achieve photo-realistic rendering. To achieve better scalability, we bring the large-scale SAPIEN PartNet-Mobility dataset [5] for the 3D cabinet asset and consider task randomization from multiple aspects. Finally, we think of the control for four common types of hand models, in order to improve fine-grained manipulative interactions.

We present OPEND, a challenge for hand manipulation skill learning over articulated cabinets from visual and language inputs. OPEND has three main features: First, to restore the natural scene as much as possible. OPEND borrows high-topology objects with various geometry and high-quality room layouts as shown in Figure 1. It currently includes hundreds of automatically processed cabinets and several manually designed rooms. Second, OPEND focuses on four typical hand models for the manipulation task, posing challenges in designing a universal approach to drive and control these hands by understanding their different structures and components. As an ongoing project, we are including more hands into the challenge. Third, to establish baseline performance, we design two powerful baseline models based on Faster R-CNN [6] and CLIPort [7].

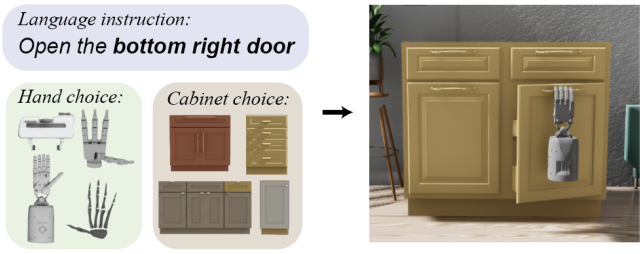

Fig. 1. Problem setting. In a simulated scene, the task is to open a cabinet door or drawer by hand corresponding to the language instruction and the camera image. OPEND provides 174 different cabinets, 372 pieces of language instructions, and 4 types of hands.

We evaluate our approach on the unseen dataset: the model needs to drive the hand to open the correct cabinet door or drawer according to the language description. In the zero-shot one-trial setting, the model only has once chance without replanning, and the cabinet is never revealed during training or validation. Experiments show that the model trained on synthetic data vary the task success rate from 14.6% to 30.3% based on different hand types and baselines. Even though the model applies an end-to-end framework that fuses language understanding, spatial reasoning, and hand manipulation, it lacks long-term planning capabilities and error correction mechanisms. Because of this, we hope that OPEND would encourage future exploration in this area.

In summary, OPEND's primary contributions are:

- A new simulation environment for benchmarking different hand-like robotics in opening the cabinet,
- A generic framework to control different kinds of robots used to accomplish the same task,
- A baseline to solve opening tasks for different hands based on visual inputs and language instructions.

[1]Y. Zhao is with the Department of Statistics, University of California, Los Angeles yizhouzhao@g.ucla.edu

[2]Q. Gao, G. Thattai, and G.S. Sukhatme are with Amazon Alexa AI {A, B, C}@amazon.com

[3]G. Sukhatme is with the Viterbi School of Engineering, University of Southern California gaurav@usc.edu

TABLE I

COMPARISON WITH OTHER SIMULATION FRAMEWORKS.

Benchmark	continous state	various cabinet type	realistic cabinet size	6-DOF grasping	various robot/hand type	photo-realistic/real background	multi-task
ManipulaTHOR [2]	✗	✓	✓	✓	✗	✓	✓
VLMBench [1]	✓	✗	✗	✓	✗	✗	✓
ManiSkill [3]	✓	✓	✗	✓	✗	✗	✓
Calvin [8]	✓	✗	✓	✓	✗	✗	✓
Robotmimic [9]	✓	✗	✓	✓	✗	✓	✓
Ours	✓	✓	✓	✓	✓	✓	✗

II. RELATED WORK

Simulation Environment. A large number of works simulate indoor household activities for training and evaluating AI agents [10]–[13]. Most of these simulators emulate high-level instructions and post effects of agent behaviors that use simplified state and action representations. Besides, some works use simplified abstract discrete action space [14]. Abstract discrete action space can reduce the task's difficulty. However, models trained in the setting without any awareness of the low-level geometry and dynamics of the objects would undermine their possibility of transferring to real-world application. For example, grasping is often simplified by attaching a nearby item to the gripper [2], [13]–[15]. By contrast, we control our hands with continuous control for each one of the joints and friction-based grasping powered by state-of-the-art physics engines (PhysX5.0).

Manipulation Task. Mastering manipulation skill for a robot usually requires an understanding of vision, language, and robotics. Recently, this field has attracted much attention across disciplines. Beyond applying imitation learning or reinforcement learning to train the robot to grasp and manipulate objects [3], [8], much recent work proposes end-to-end networks that can learn skillful controls that require precise spatial reasoning or language understanding [7]. With the help of an environment that offers high-performance physics simulation, training an agent for manipulation tasks can be achieved efficiently.

However, learning a model from image and language input in the simulation environment with continuous states is still considered challenging [4], [16], as the simulation engine needs to provide rendering and simulation results constantly. Therefore, for manipulation benchmarks, compromise often occurs by giving the model full knowledge of the environment and object for easier training [16], reducing the difficulty of grabbing items [14], and providing a limited number of items without varying the material and background for simpler evaluation [1], [8].

Nevertheless, our work simulates continuous states for articulated objects and uses the RGB images and language instructions as model inputs. Without applying abstract grasping or changing the object size, we train our hands (agents) to open the cabinet by hybridizing the end-to-end network and the motion planner. The advantage of the neural network lies in its powerful spatial reasoning capability, and the motion planner helps stable performance based on spatial reasoning.

Language Conditioned Manipulation. Relating human language to robot actions has been of interest in recent research [8], [14], [17]. Natural language presents specification, providing an intuitive way to refer to abstract concepts concerning spatial, temporal, and causal relationships.

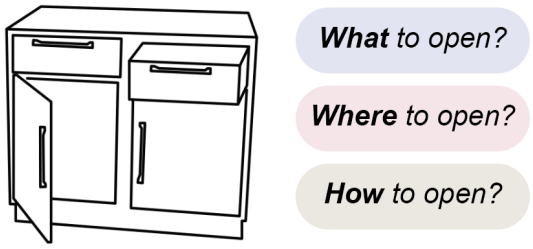

Fig. 2. A cabinet of multiple drawers and doors. Language instructions are necessary to address the ambiguity. A valid task instruction must specify what (drawer or door) and where to open. A baseline model solves how to open.

We focus on the language that describes the object type and spatial relationships in our task. Because of multiple interactive parts on the cabinet, language is crucial to address the ambiguity. Table II comparison the settings between our OPEND and other famous challenges.

III. SIMULATION ENVIRONMENT

This section introduces how we build the simulator to support different types of simulated hands, hundreds of drawers and cabinets, and various scenes with randomized material and backgrounds.

Engine. We choose the platform OMNIVERSE as the backend to design the challenge of the OPEND. Rigid body, soft body, articulated body, and fluid can be efficiently and reliably simulated in OMNIVERSE. For its Python scripting environment, it is easy to bring open-source and third-party Python libraries into it to help the research. Besides, it provides a powerful rendering ability with the ray tracing technology.

Asset. The asset for building the simulation for opening cabinets and drawers are from SAPIEN [5]. As a collection of rigid bodies, robots, and articulated objects, the SAPIEN dataset provides hundreds pieces of storage furniture as cabinets and drawers for our challenge. The furniture is well-furbished with detailed rendering material.

To build photo-realistic simulation background, we manually build several rooms as synthetic indoor scenes with

randomized lights, floor materials, wall types, and decorations.

Hand. A robotic hand is a type of mechanical hand, usually programmable, with similar functions to a human hand. OPEND offers the controller for four of the most representative robot hands. The first three, *Franka gripper*, *Allegro hand*, and *Shadow hand*, are commercially available robot hands. The last one, *Skeletal hand*, is modeled based on the biological structure of the human hand.

Figure 3 sketches how hands are modeled and rigged in OMNIVERSE. There are three types of joints in hand modeling: the prismatic joint allows two bodies to slide along a common axis; the revolute joint allows two bodies to rotate along a common axis; the D6 joint for the hands enables body parts to rotate on the y and z axis while locking the rotation on the x axis.

Table II lists each hand's joint components and degrees of freedom to control. On top of that, we need an additional six degrees of freedom to control the position and rotation of each hand, since these hands are detached from the robot body.

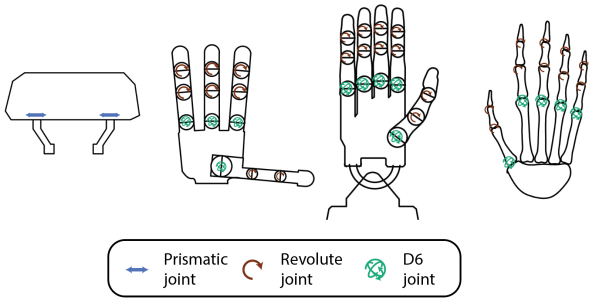

Fig. 3. Hand sketches. From left to right: Franka gripper, Allegro hand, Shadow hand, and Skeleton hand.

TABLE II
HAND JOINT COMPONENTS AND DEGREES OF FREEDOM.

	Prismatic joint	Revolute joint	D6 joint	DoF
Franka gripper	2	0	0	2
Allegro hand	0	8	4	16
Shadow hand	0	10	5	20
Skeleton hand	0	10	5	20

IV. THE TASK

Opening a cabinet can involve pulling out a cabinet drawer or rotating a cabinet door. A piece of storage furniture may contain both drawers or doors. Especially when one piece of furniture contains multiple drawers and doors, we must involve language instructions to resolve ambiguities.

We define the door as one driven by a prismatic joint and the cabinet as the one driven by a revolute joint.

Task definition. Given a cabinet **door** on a piece of furniture at a certain location, the task is to rotate the door to at least a certain degree δ (in percentage to its maximum opening limit).

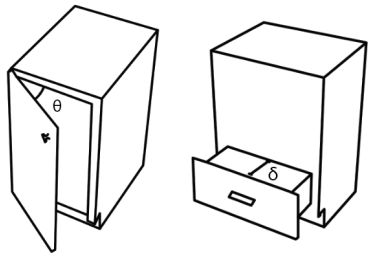

Fig. 4. Opening condition for the door and door.

Given a **drawer** on a piece of furniture at a certain location, the task is to pull out the drawer to at least a certain distance θ (in percentage to its maximum opening limit).

In our task, we set the percentage to be 20 for both doors and drawers. The maximum opening limit for doors are generally 180 degree.

Language instruction. The language instruction in our task should specify what (door or drawer) and where to open it. After deciding what to open according to the joint type, we apply Algorithm 1 to generate spatial descriptions for the multiple positions of drawers and doors on one cabinet.

Algorithm 1 Generate language instructions for a cabinet

Require: Obtain the total number of drawers and doors $n \geq 0$, and their positions $\{p_i \mid p_i = (x_i, y_i, z_i)\}_{i=1,2,...,n}$
for $i = 1, 2, 3, ..., n$ **do**
  if $i \geq 2$ **then**
    Compare y_i with $\{y_1, ..., y_{i-1}\}$;
    Update description for $\{p_1, ..., p_i\}$ with {left, second left, middle, second right, and right}
    Compare z_i with $\{z_1, ..., z_{i-1}\}$;
    Update description for $\{p_1, ..., p_i\}$ with {top, second top, middle, second bottom, and bottom};
  end if
  if find the same description for two positions **then**
    return invalid cabinet.
  end if
end for
return descriptions for $\{p_1, ..., p_n\}$

The key idea of the algorithm is to perform iterative comparison between the target positions vertically and horizontally. However, this algorithm may fail to generate valid descriptions if the cabinet has too many doors and cabinets. Figure 5 shows some examples from the language generation algorithm.

Task Statistics. The SAPIEN dataset [5] contain a total number of 346 different pieces of the storage furniture. After filtering out the model with mesh or URDF format [18] errors, we collect 198 cabinets for our work. After getting rid of the cabinets with invalid language descriptions, we obtain 174 unique cabinets and a total number of 372 doors and drawers along with descriptions. We split a part of cabinets with valid language descriptions as the testing dataset. Table III shows the detail of the train-test split.

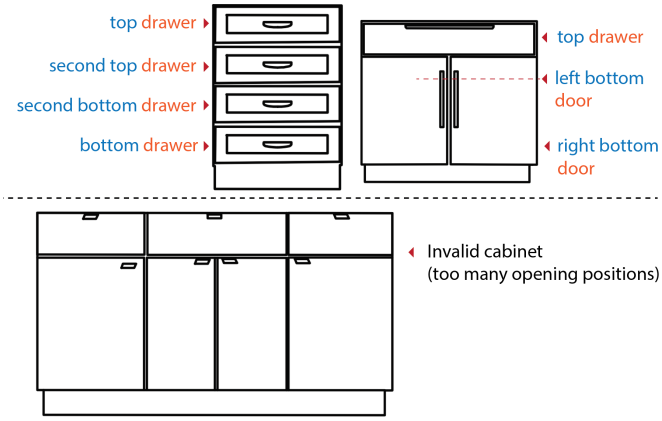

Fig. 5. Language description examples.On the top, language descriptions are validly generated. On the bottom, the parser fails because there are too many doors and drawers to describe.

TABLE III
DATA SPLIT FOR TRAINING AND TESTING

	Count	Train	Test
Cabinet drawer	167	138	29
Cabinet door	205	145	60
Cabinet	174	135	39

V. MODEL

How to open the cabinet by hand? We wanted to solve this task realistically and naturally.

For us humans, we use visual perception to locate the cabinets. At the same time, we can understand the semantics to determine which cabinet doors and drawers need to be opened. Afterward, we open the target by controlling the hand and the movement of the knuckles.

A. The multi-step planner

Similarly, for the robot, we use a multi-step planner to solve the task (See Figure 6). We would analyze our model to illustrate why opening cabinets is a difficult task, especially when we introduce faithful physics in 3D space.

Initial state. Given the inputs of RGB+D images of the cabinet and a sentence as language instruction, the robot hand needs to open the correct door or drawer of the cabinet.

Locate handle.: a **handle solver** is a model to identify and localize the correct handle of the door and drawer to open. The difficulty lies in this step in accurately recognizing the grasp position.

Approach handle.: after localizing the handle, we drive the robot hand in front of the handle. The error occurs when the depth prediction is not accurate. We simplify this part by assuming that the camera can sense the depth.

Close finger.: a **grasp planner** defines the movement of the finger joints to grasp to handle. Figure 7 shows various types of handles we have in the simulation environment. A grasp planner needs to be strong enough to help the fingers hold many kinds of handles.

Pull open.: we push out the robot hand straightly to pull open the target. If the grasp is incorrect, the door will likely loosen at this step.

Final state.: a task checker built in the OPENED determines the task's success based on the acquisition of articulation information on the cabinet.

B. Handle solver

The handle solver's task is to find the cabinet's correct handle. It needs to know the location and orientation of the handle to open the cabinet. The problem with this approach is that we can only provide an RGB image and a piece of sentence since we only have information from the camera and the language instruction.

To tackle this challenge, we apply two compelling models which use powerful deep neural networks.

The first idea, **Faster R-CNN** [6], leverages the region proposal network to perform efficient object detection. From the image input, the Faster R-CNN model predicts the bounding boxes of all handles in the image. Then, we apply Algorithm 1 to generate language instructions for predicted handles. Finally, we compare the generated language instructions with the language instruction input to identify the desired predicted handle.

The second idea, **CLIPort** [7], is a framework that combines learning generalizable semantic representations for vision and understanding necessary spatial information for fine-grained manipulation. Initially, the CLIPort as a language-conditioned imitation-learning agent learns broad semantic knowledge (what) and the spatial precision (where) to transport. However, since we already know that the handle on the drawer and door is what to open, we only apply half of the CLIPort to learn where to grasp according to the predicted affordance map.

C. Grasp planner

After attending to a local region to decide where to grasp, a grasp planner drives the hand to close the fingers or the gripper.

The second crucial part of the model is designing a heuristic search algorithm that can be used on robot actions to perform the grasp action. The idea behind this approach is that we are not interested in finding the exact position of each joint at each time but instead in finding the most likely final state of the joints. After that, we can see the intermediate joint actions by interpolating the initial and final joint positions.

To do so, we need to define what an ideal final position is: a state that has a high probability of being executed to open the cabinet after the pull-open step (Figure 6 step (5)). Due to page limitations, we put how to do an empirical search to obtain the grasp planner for each hand in the appendix.

VI. EXPERIMENT

We perform experiments in the simulation engine to answer the following questions: 1) How accurate is the language-conditioned prediction for the handle from the

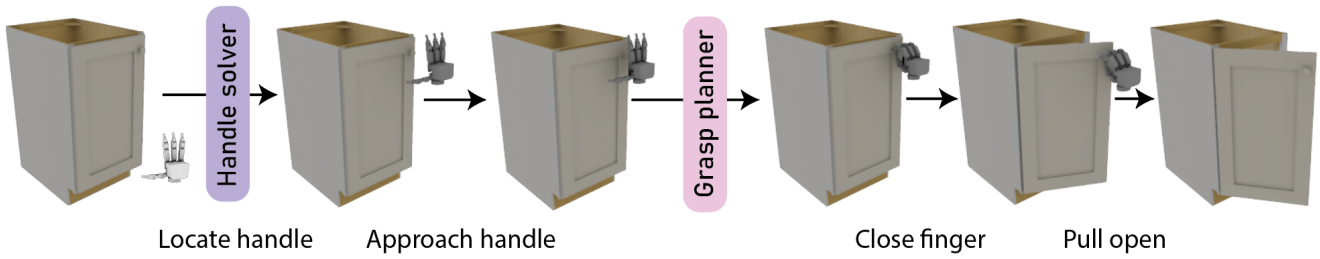

Fig. 6. Multi-step planner to open the cabinet. We apply such a planner to all hands. All the hands share the same handle solver to locate the handle position, while they differ in the grasp planner.

Fig. 7. Various handles in the task

architecture of the Faster R-CNN or CLIPort? 2) How is the performance from a heuristic search for the grasp action for different hands? 3) How generalizable is our multi-step planner when we put together the best handler solver we trained from training and the best grasp planner from searching?

A. Training the Faster R-CNN and CLIPort

To prepare the training data from the networks, we first determine the training images and the ground truth bounding boxes of the handles. We introduce randomization to the training data to eliminate inherent biases and encourage a generalized model. The position of the cabinet randomly shifts one plane by x axis and y axis. Along with the handle's original texture, the handle's material is randomized by giving a random RGB input. This process results in 1720 training images with bounding boxes. We apply 80% of then for training and 20% for validation.

To train a Faster R-CNN model, we first load the model pre-trained on COCO (train2017) dataset [19] and modify the scoring head to get a one-class prediction. Then, we fine-tune the model on our training dataset based on regression loss from bounding box prediction.

$$\mathscr{L}_{\text{reg}} = \sum_{i\in\{x,y,w,h\}} (t_i - d_i(\mathbf{p}))^2 + \lambda \|\mathbf{w}\|^2 \tag{1}$$

Figure 6 (top right) shows the learning curves of different versions of the Faster R-CNN. We also compares different versions of the Faster R-CNN with different backends: ResNet-50, MobileNet-V3, and MobileNet for low-resolution inputs.

To train a CLIPort agent for attention map prediction, we regard the center of the bounding box as the attention target (one-hot map) and fine-tune the CLIPort with ResNet-18 backend according to the binary cross-entropy loss.

$$CE = -\sum_i t_i \log\left(f(s)_i\right) \tag{2}$$

Figure 6 (bottom right) shows the learning curves of CLIPort agents with different numbers of vision-language fusion layers. The vision part of the CLIPort uses a pre-trained ResNet backend, and the language part uses the a pre-trained CLIP model. We only fine-tune the vision-language fusion part of the CLIPort.

B. Searching the grasp planner

The grasp planner works by allowing to define movement to close fingers for different types of hands. Since we hand many unique cabinets, in this baseline, we need to define one planner for each hand that can successfully grasp the handle and opens the target as possible.

TABLE IV

GRASPING SUCCESS RATE FOR DIFFERENT HANDS

	Drawer	Door	Overall
Franka gripper	91.6%	58.5%	73.4%
Allegro hand	79.6%	66.2%	72.8%
Shadow hand	92.3%	24.4%	54.8%
Skeleton hand	50.3%	51.2%	50.8%

Figure 9 plots the final state for each hand after grasp searching. Table IV illustrates the corresponding grasping success rate for the four types of hands. We can also consider the task success rate if the ground-truth bounding box location is given.

Comparisons between different grasp planners on all cabinets show that the most straightforward structure's Franka gripper achieves the highest success rate. We can also see that opening a cabinet door is generally harder than pulling open a drawer. As the hand structure becomes more complex, the overall success rate drops. It can be explained by the fact that the complex structure of the hand potentially causes an unexpected collision.

C. Put together

Then, we combine the grasp planner and handle solver to obtain the experimental results shown in Table VI-C. We have received two sets of experimental results using

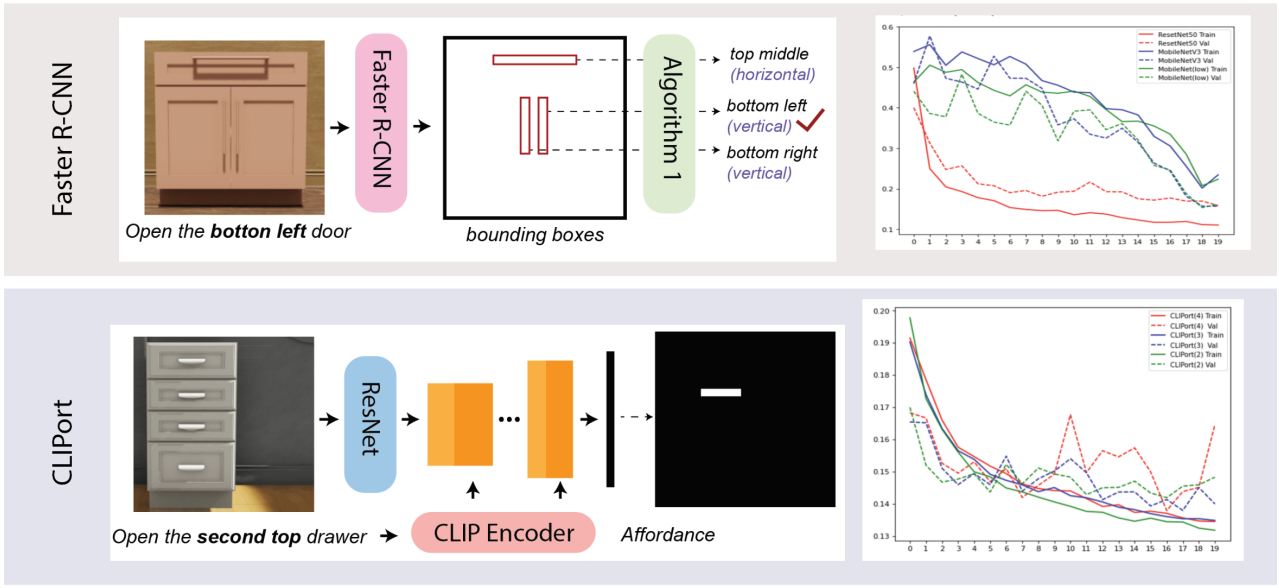

Fig. 8. Training the handle solver.

Fig. 9. Final grasping state deduced by the grasp planner

two different handle solvers but the same grasp planner. Comparing these two sets of experimental results, we find that Allegro Hand has the best testing performance (about 30% success rate) when using our model. Its finger shape and relatively simple structure allow it to perform well in opening doors and drawers.

Another important finding is that opening doors is generally more difficult than opening drawers. For example, because of the small area that Franka gripper can cover, precise identification of the grasp position is critical to completing the task. However, some knobs of the door are smaller than the handle of the drawer, making opening the door a more difficult task.

Although the more complex Shadow hand and Skeleton hand are more similar to the real human hand, they also require more precise control. In our model, the performance of these two is not as good as that of the Franka hand and Allegro hand, which are relatively simple in structure. Especially in grasping the handle, just knowing the position and horizontal and vertical direction of the hand seems insufficient for accurate control, which inspires our future work to make them more potential in complex environments.

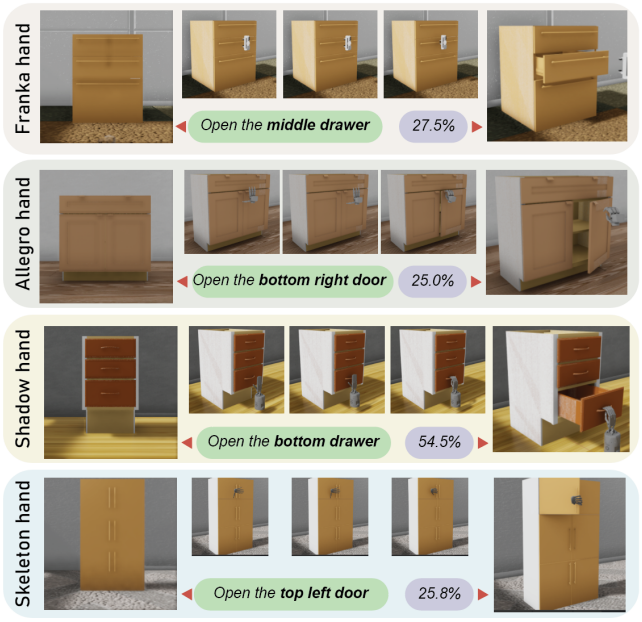

Fig. 10. Caption

VII. Discussion

In this section, we briefly discuss the potential as well as the flaws of our model to throw light on future research.

A. Integrating with the robot arm

In a typical robot task, the placement of the robot largely determines whether the task can be completed. Especially for robots with fixed bases, the robot's initial position is critical. However, in our design, we can use a reverse way of thinking to weaken the impact of this issue. Since our task is deployed

	Faster R-CNN + Language parser						CLIPort					
	Drawer		Door		Overall		Drawer		Door		Overall	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Franka gripper	19.6%	20.7%	23.4%	13.3%	21.6%	15.7%	43.5%	35.9%	19.3%	13.3%	31.1%	21.3%
Allegro hand	36.2%	48.3%	33.8%	21.7%	35.0%	30.3%	59.7%	50.0%	30.8%	20.0%	45.0%	30.0%
Shadow hand	47.1%	44.8%	18.6%	8.3%	24.7%	20.2%	17.3%	27.6%	30.8%	14.8%	17.6%	14.6%
Skeleton hand	15.9%	10.3%	24.8%	16.7%	20.5%	14.6%	21.0%	17.2%	26.9%	18.3%	24.0%	18.0%

in a simulation environment, we may first determine the rotation of the hand position and then derive the possible positions of the base by the inverse kinetic method.

Inspired by the Joint-space inverse differential kinematics (IK) motion controller [20], we can convert task-space errors (from the calculation of the position and rotation of the hand) into joint space errors and then apply proportional and derivative (PD) gains to generate joint torques. Specifically, after we determine the analytical Jacobian matrix of the robot, we apply the Levenberg-Marquardt algorithm [21] to determine whether the root position as well as initial hand position p_0 rotation r_0 of a robot allows the end robot hand to reach position $\hat{p}$ and rotation $\hat{r}$.

Algorithm 2 Levenberg-Marquardt algorithm

Require: Obtain the analytical Jacobian J, initial hand (end-effector) position and rotation (p_0, r_0) and target hand position and rotation $(\hat{p}, \hat{r})$;
while True **do**
  Get hand current position and rotation (p_t, r_t)
  Get current robot joint states (DoF states) d_t;
  Update current analytical Jacobian matrix for the end-effector J_{ee}
  Calcualte current position error $error_p(p_t, \hat{p})$ and orientaion error $error_r(r_t, \hat{r})$
  Solve damped least squares:

$$\delta = J_{ee}^T (J_{ee} J_{ee}^T + \lambda I)^{-1} (error_p, error_r)^T$$

  Update the joint state to be $d_{t+1} = d_t + \delta$ and thus the hand state (p_t, r_t);
  if $error_p < \varepsilon$ and $error_r < \varepsilon$ **then**
    return initial hand position (p, r) is valid.
  end if
  if reach time or iteration limit **then**
    return initial hand position (p, r) is invalid.
  end if
end while

B. Limitations

Readers may have noticed that the task of opening a cabinet becomes exceptionally complex when we introduce sufficiently enough random factors. In our study, we only covered a few essential random factors: object type, object location, and robot (hand) type, leaving a lot of unexplored random factors (see Figure **??**) that ought to be discussed.

Light intensity. We first realized that the light intensity substantially affected the experimental results. We have tried to weaken the light intensity by 30% to see what changes in the model results. The affordance map obtained by the CLIPort model deviates significantly from the actual one. The Faster R-CNN model also has difficulty inferring the correct number of bounding boxes in the images. As a result, all robot hands perform very poorly, and only a few tasks can be completed, even on the training dataset.

Depth sensing. Another critical assumption behind our OPEND benchmark is that a perfect depth sensing camera exists to tell the distance between the cabinet and the camera. Few of our tasks can be accomplished without the correct depth-sensing technique. Traditionally, depth sensing would involve matching point pairs between aligned images from two different sensors and then using the resulting difference maps to acquire the object's depth in the environment. However, there have been many advancements in depth sensing which have occurred in parallel with improvements in sensor hardware [22], computer vision and machine learning [23].

C. Human performance

We obtained a human evaluation of 10 randomly sampled cabinet doors and cabinet drawer directives from the dataset. The experiment involved lab experts who randomly received initial Franka gripper positions and completed 100 trajectories each for opening the drawer and the door. The task each uses a keyboard-and-mouse or gamepad controller. Before the experiment, the participants were allowed to familiarize themselves with the controller and the task. The participants obtained a high success rate of 92% in opening the drawer and a 100% success rate in opening the cabinet. The failure cases are mainly due to the loss of patience when grabbing the handle. This indicates that the directives in OPEND are well-aligned with the demonstrations and shows that there is still plenty of room to improve the model.

VIII. CONCLUSION

We introduced OPEND, a benchmark for learning to open a cabinet drawer or door from language instructions and vision inputs. To recreate the realistic physics and scenery as much as possible, we used the most challenging settings as far as we know. In particular, we stop making any simplifications to the hand grasping actions.

OPEND brings us closer to the community goal of language-driven robots that can perform accurate and stable interactions. The photo-realistic environment background and dynamic required in OPEND narrows the gap between virtual simulation and reality.

We use OPEND to evaluate the state-of-the-art model involved in our framework. Experiments show the effectiveness of the model in handling our tasks. While the model is relatively competent at accomplishing some key steps, the overall task success rates across all hands show that there still is much room for improvement. Particularly when we consider multiple aspects of randomness w.r.t. as lighting, material, object location, and background, the difficulty of this single task becomes unbearable.

Therefore, we have decided not only to open source the dataset and training procedure but also to present the whole process of how the data is collected. In our OPEND benchmark, researchers can discover that we define and build task scenes, model and drive robotic hands, and implement multi-angle randomization. Besides, they may modify our settings to make the task even closer to real robotics or enlarge the dataset size to raise the model's performance and robustness.

We also believe that better models can be approachable by models that exploit hierarchical control, modular training, structured reasoning, and systematic planning [14]. We are encouraged by the possibilities and challenges that the OPEND benchmark introduces to the community.

REFERENCES

[1] K. Zheng, X. Chen, O. C. Jenkins, and X. E. Wang, "Vlmbench: A compositional benchmark for vision-and-language manipulation," *arXiv preprint arXiv:2206.08522*, 2022.

[2] K. Ehsani, W. Han, A. Herrasti, E. VanderBilt, L. Weihs, E. Kolve, A. Kembhavi, and R. Mottaghi, "Manipulathor: A framework for visual object manipulation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4497–4506, 2021.

[3] T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su, "Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations," *arXiv preprint arXiv:2107.14483*, 2021.

[4] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan, "A survey of embodied ai: From simulators to research tasks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

[5] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, *et al.*, "Sapien: A simulated part-based interactive environment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11097–11107, 2020.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[7] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on Robot Learning*, pp. 894–906, PMLR, 2022.

[8] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters*, 2022.

[9] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *arXiv preprint arXiv:2108.03298*, 2021.

[10] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, 2017.

[11] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018.

[12] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. Vainio, C. Gokmen, G. Dharan, T. Jain, *et al.*, "Igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," *arXiv preprint arXiv:2108.03272*, 2021.

[13] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[14] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10740–10749, 2020.

[15] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, K. Liu, *et al.*, "Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments," in *Conference on Robot Learning*, pp. 477–490, PMLR, 2022.

[16] O. Mees, L. Hermann, and W. Burgard, "What matters in language conditioned robotic imitation learning," *arXiv preprint arXiv:2204.06252*, 2022.

[17] C. Lynch and P. Sermanet, "Language conditioned imitation learning over unstructured data," *arXiv preprint arXiv:2005.07648*, 2020.

[18] R. T. Arrazate, "Development of a urdf file for simulation and programming of a delta robot using ros," *Santiago de Querétaro*, 2017.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[20] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, *et al.*, "Factory: Fast contact for robotic assembly," *arXiv preprint arXiv:2205.03532*, 2022.

[21] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis*, pp. 105–116, Springer, 1978.

[22] A. M. Pinto, P. Costa, A. P. Moreira, L. F. Rocha, G. Veiga, and E. Moreira, "Evaluation of depth sensors for robotic applications," in *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, pp. 139–143, IEEE, 2015.

[23] L. Shao, J. Han, P. Kohli, and Z. Zhang, *Computer vision and machine learning with RGB-D sensors*, vol. 20. Springer, 2014.