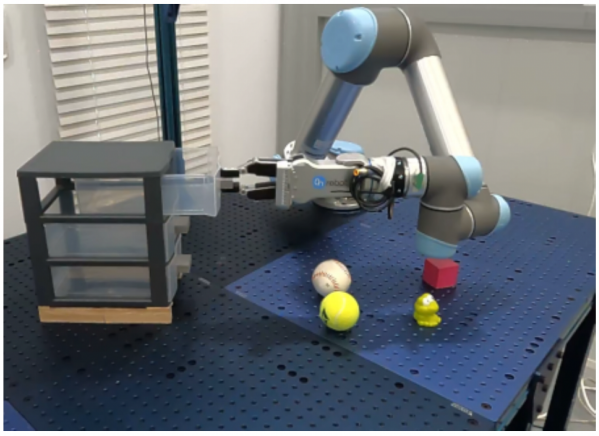


"open the top drawer"

Towards Generalizable Vision-Language Robotic Manipulation: A Benchmark and LLM-guided 3D Policy

Ricardo Garcia, *Shizhe Chen*,
Cordelia Schmid

(October 2024)

Robotics research paper analysis and testing

Martin Jolif, Tancrède Martinez

December 2024

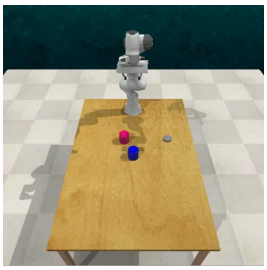
1 Introduction

1.1 Objective of the research paper

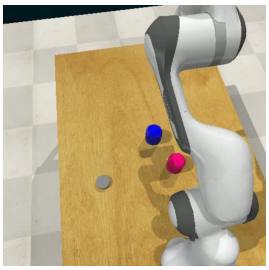
Generalization capabilities are part of the feature the human brain tends to do easily, however, such capabilities are quite challenging to reproduce by robot. That is why this article tackles the issues of task generalization.
To give a little bit of context, the authors are interested in the realization of numerous tasks and some complex unseen variant of those tasks by a robotic arm, described by a text instruction. For example, we give the robot the text instruction "open the top drawer" and he has to complete this task in an environment by "seeing" the environment through cameras.
The paper 'Towards Generalizable Vision-Language Robotic Manipulation: A Benchmark and LLM-guided 3D Policy' [8] addresses the challenge of task generalization using language-vision robotic policies.
To do so, the authors introduce the following :
- A benchmark to evaluate the ability of generalization of models: GemBench
- A language-conditioned point cloud transformer model 3D-LOTUS, achieving state-of-the-art results
- A version of 3D-LOTUS focused on task generalization using task planning with foundation models: 3D-LOTUS++, establishing state-of-the-art results on existing benchmarks and GemBench, and also work reliably on a real robot.

1.2 Robot

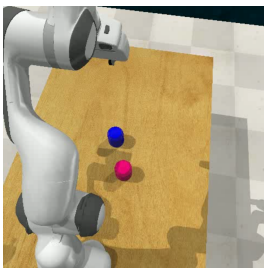
The robot consists of an articulated arm with 6 degrees of freedom (6 rotating joints) and a gripper as an end-effector. Moreover, there are in the simulation phase 4 RGB-D cameras around the robot (just 3 in the real world experiment). They are disposed in front, at the left and right shoulder and at the gripper of the robot (see figure 1):



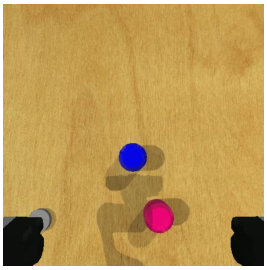
(a) Front camera



(b) Left shoulder camera



(c) Right shoulder camera



(d) Gripper camera

Figure 1: Cameras position

The real world robot used for the real experiment is an UR5 arm equipped with a Robotiq 85 gripper (see image from the cover page).

1.3 Our methodology

Our approach to studying this article was to try to reproduce the results of the article for the 3D LOTUS model, as well as testing his capabilities on our own created tasks.
Unfortunately, we did not reproduced the results of 3D-LOTUS ++ due to computational costs.
Additionally, we analyzed the general limitation of those models and proposed some ideas for new features to tackle those limitations.

2 Architecture and benchmark

2.1 Gembench: a generalizable vision-language robotic manipulation benchmark

The paper studied [8] introduces a benchmark divided into four levels built on RLBench [4]. This benchmark consists of 16 simple tasks (and 31 variations) such as pick, press, close, open, stack, etc., and their combination. The level corresponds to increasingly difficult generalization tasks allowing one to better understand the limitations of a given model:

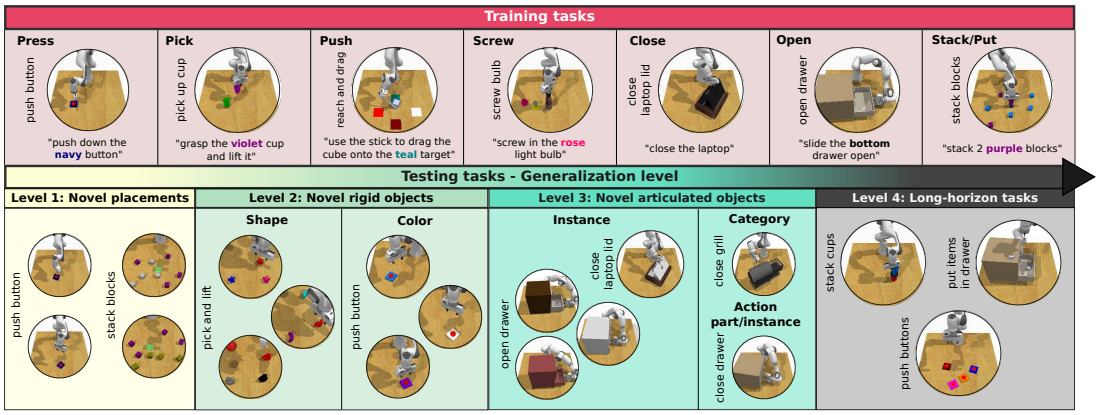


Figure 2: GemBench overview [8]

Level 1: Consists of the same tasks as in the training set but with random placement of objects and/or adding similar objects.

Level 2: Add 15 unseen tasks (28 variations) using two categories of generalization :
- Novel object color composition: Swap the color of different objects (with associations unseen in the training set)
- Novel object shapes : Uses tasks learned for some shapes (such as picking) with another shape.

Level 3: Consists of tasks regarding articulated objects divided into three categories:
- Novel action-part compositions : Test the ability to use intermediary action with fully trained sequences of actions (for example, if trained for open bottom drawer and put item in middle shelf then we test open middle drawer)
- Novel instances : Generalization of number of repetitions of an action.
- Novel categories : Generalize tasks that require multiple features ("pick the blue slice of apple" and "pick the blue slice of watermelon")

Level 4: Test tasks requiring multiple actions to be done (put items in the locked box) or that contain multiple actions in their definitions (pick an apple and a banana and place them in the bag).

This benchmark is more complete than the existing ones about tasks generalization, as we can see in figure 3.

TABLE I: Comparison of benchmarks for vision-and-language robotic manipulation. *Multi-skill:* covering multiple action primitives beyond pick and place. *atc-obj:* tasks involve interactions with articulated objects. *Test generalization level to attr-obj:* unseen size, color or texture of the object, *act-obj:* unseen action object combination, *inst* or *cate*: same action for unseen object instance or category respectively and *long-horizon:* unseen combination of multiple seen actions and objects.

Benchmark	Simulator	Physics	# Train task(var)	# Test task(var)	Multi-skills	atc-obj	Test generalization level attr-obj	act-obj	inst	cate	long-horizon
RLBench-74Task [17]	RLBench	✓	74 (74)	74 (74)	✓	✓	✗	✗	✗	✗	✗
RLBench-18Task [18]	RLBench	✓	18 (249)	18 (249)	✓	✓	✗	✗	✗	✗	✗
VLMBench [19]	RLBench	✓	8 (233)	8 (374)	✓	✓	✓	✗	✓	✗	✗
ALFRED [20]	AI2-THOR	✗	7 (21,023)	7 (1,529)	✓	✓	✓	✗	✗	✗	✓
Calvin [21]	PyBullet	✓	34	34 (1000)	✓	✓	✓	✗	✗	✗	✓
Ravens [22]	PyBullet	✓	10	10	✗	✗	✓	✗	✓	✓	✗
Arnold [23]	Isaac Sim	✓	8 (3571)	8 (800)	✓	✓	✓	✓	✓	✗	✗
VIMA-Bench [24]	Ravens	✓	13	17	✗	✗	✓	✗	✓	✓	✓
Colosseum [25]	RLBench	✓	20 (280)	20 (20,371)	✓	✓	✓	✗	✗	✗	✗
GemBench (Ours)	RLBench	✓	16 (31)	44 (92)	✓	✓	✓	✓	✓	✓	✓

Figure 3: [8].

2.2 3D-Lotus architecture

The goal is to learn a policy $\pi(a_t|O_t, L)$ for robotic manipulation where L is the text instruction, and $a_t \in \mathbb{A}, O_t \in \mathbb{O}$ are the action and observation at time step t. The observation space $\mathbb{O}$ is composed of the aligned RGB-D images from the K cameras and the robot state. The action space $\mathbb{A}$ is composed of the gripper's position $a_t^p \in \mathbb{R}^3$, the Euler angles $a_t^r \in \mathbb{R}^3$ and the open state of the gripper $a_t^o \in \{0, 1\}$ indicating if the gripper is open or closed. As we know the position of the end-effector of the robot, inverse kinematics is used to move from a_{t-1} to $a_t = (a_t^p, a_t^r, a_t^o)$.

As in PolarNet [2], the multi-view RGB-D images are projected into a unified point cloud V only covering objects and robot gripper (points outside the robot workspace and points from the robotic arm are removed). Each point is associated with a feature vector containing its RGB color and its relative height to the table in addition to its coordinates: this is the point tokens (see figure 4). Moreover, the text instruction is encoded with the CLIP text encoder, this gives the text tokens (see figure 4).

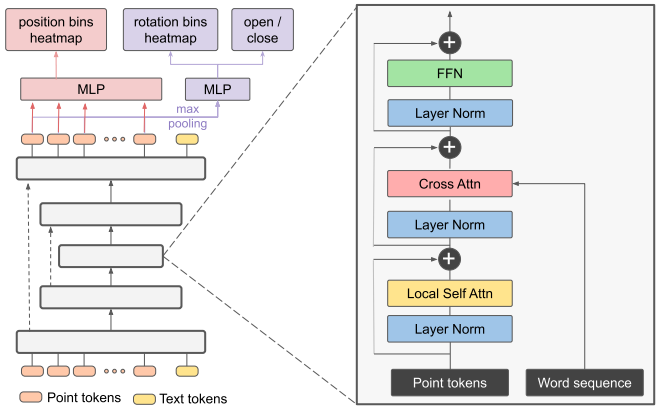


Figure 4: 3D-LOTUS architecture [8]

Then, the point cloud and text tokens are encoded through Point Transformer V3 (PTV3 [10]). The task sequence embedding is obtained with cross-attention (a cross-attention layer is

added after each self-attention layer of PTV3) or weighted average (we use this average on each normalization layer of PTV3). We then get an embedding of each point in space conditioned by the language instruction denoted v_i^e where $v_i \in V$.

Finally, the next action should be predicted. They predict it as a classification approach instead of a regression approach. To do so, bins are created over the point cloud, and a heatmap is predicted with the embeddings v_i^e on these bins through an MLP. We select the bin with the highest score to determine axis positions independently, this gives us a_t^p. For the rotation prediction the same method is used as before by discretizing the Euler angles: we obtain a_t^r. The open state of the gripper remains a simple binary classification problem (a_t^o). The cross-entropy-loss is used to train both position, rotation and open state classifiers.

2.2.1 3D-LOTUS++ architecture

The general idea behind 3D-LOTUS++ is to use foundation models like LLMs and Vision-Language-Models (VLM) and use their abilities to generalize to unseen scenarios due to their training on massive data. 3D-LOTUS++ can be decomposed into three parts (see figure 5):

- task planning with LLM
- object grounding with VLM
- motion control with a modified version of 3D-LOTUS

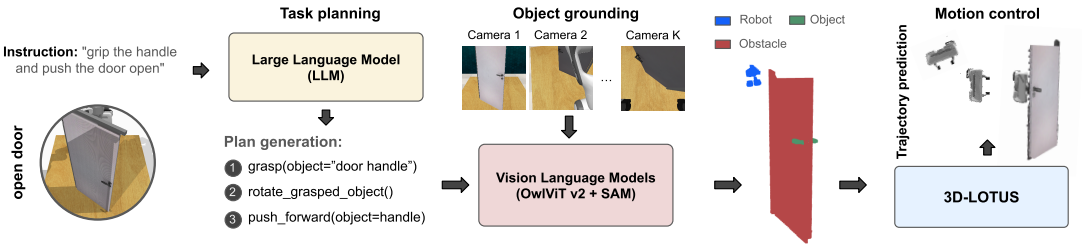


Figure 5: Overview of 3D-LOTUS++ framework [8]

Task planning: use a LLaMa3-8B (prompts) to decompose the instruction L into a sequence of steps $l_1, ..., l_T$ where l_i is a primitive action which corresponds to the following *grasp(object)*, *move_grasped_object(target)*, *push_down(object)*, *push_forward(object, target)*, *release()* and *rotate_grasped_object()*.

Object grounding with VLM: use state-of-the-art OWLv2 VLM [6] to obtain bounding box of surrounding objects and their associated semantic embedding. Then, we segment objects within each bounding box with Segment Anything Model (SAM) [5]. Then we can obtain the 3D point cloud for each bounding box using segmentation information for each camera. To merge observation from different cameras, we identify objects that have similar semantic embeddings and point cloud distances below a certain threshold. At that moment, we have a 3D point-cloud in which each object has an associated semantic embedding. To match an object with his text description, we measure the cosine similarity between the object text embedding and the semantic embedding of the points in the point clouds. The highest similarity is selected as the match: we obtain an annotated point cloud as in the second last step in figure 5.

Motion Control : We compute the motion using point-cloud input features from the Object grounding with VLM step and use it sequentially to do all the actions given by the LLM step by step.

3 Methodology analysis and critics

3.1 Critical analysis of the methodology

The approach used by the author is really interesting, proposing multiple visions of the same problem.
As for the benchmark, the described level of generalization seems natural with an evident progression that corresponds well with the designed model (one could argue that the designed benchmark is what makes the model 3D-LOTUS ++ really shine in terms of novelty, especially for level 3 and 4, even though 3D-LOTUS already has state of the art results on RLBench 18 benchmark).
For the 3D-LOTUS architecture, the authors use their own designed Polarnet point cloud transformer [2], relying on their previously great result using this vision tool, adding a CLIP encoder to embbed the text-based instructions making this a simple and effective model.
The 3D LOTUS model is a way to 'validate' the 3D LOTUS ++, even if the last perform does not perform as well on other benchmark than Gembench.
The 3D-LOTUS ++ is based on some ideas that have proved their validity and are quite natural for human to do such as the decomposition of instructions into simple tasks by LLM in [3]. The combination of Foundation Models enforce the generalization capabilities; however, we think that another model between 3D LOTUS and 3D LOTUS ++ (called something like 3D LOTUS +) using only LLM and 3D LOTUS could have been interesting to compare results with 3D LOTUS ++ and 3D LOTUS.
Finally, even if the model focuses on 'tasks generalization' some aspect of generalization are clearly not spoken about in the article like doing similar task in different environment condition.

3.1.1 About testing

As discussed in the section above, the capabilities of generalization of 3D-LOTUS ++, as designed, can only be tested by the gembench, as for, the author of [8] does not even provide the results of test with 3D LOTUS ++ on RLBench-18Task [9].
Another interesting benchmark that 3D-LOTUS ++ could have is Colosseum [7] which test different generalization capabilities such as lightning changes and camera angles defects and more generally environment perturbations. We saw in our own test that 3D-LOTUS seemed less capable of color identification with floating balls (the ball close to the table were likely more chosen than the right colored ball if it was floating).
Finally, some results of 3D-LOTUS ++ are hard to interpret, whereas 3D-LOTUS performs well in most of low level tasks and at least decently in the other, 3D-LOTUS ++ fails almost certainly for some tasks such as PutInCupboard described here:

Task: Grab the specified object and put it in the cupboard above. The scene always contains 9 YCB objects that are randomly placed on the tabletop.
New/Modified: Yes, modified the object names to include object category and therefore the instructions.
Objects: 9 YCB objects, and 1 cupboard (that hovers in the air like magic).

Such divergence from the original 3D-LOTUS model seems quite surprising and we did not find any further explanation by ourselves.

Method	Avg.	Close Fridge+0	Close Jar+15	Close Jar+16	CloseLaptop Lid+0	Close Microwave+0	LightBulb In+17	LightBulb In+19	Open Box+0	Open Door+0	Open Drawer+0
Hiveformer [17]	$60.3_{\pm 1.5}$	$96_{\pm 4.2}$	$64_{\pm 13.9}$	$92_{\pm 2.7}$	$90_{\pm 3.5}$	$88_{\pm 7.6}$	$12_{\pm 4.5}$	$13_{\pm 6.7}$	$4_{\pm 4.2}$	$53_{\pm 15.2}$	$15_{\pm 12.2}$
PolarNet [2]	$77.6_{\pm 0.9}$	$99_{\pm 2.2}$	$99_{\pm 2.2}$	$99_{\pm 2.2}$	$95_{\pm 3.5}$	$98_{\pm 2.7}$	$72_{\pm 12.5}$	$71_{\pm 6.5}$	$32_{\pm 11.5}$	$69_{\pm 8.9}$	$61_{\pm 12.4}$
3D diffuser actor [35]	$91.9_{\pm 0.8}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{99}_{\pm 2.2}$	$\mathbf{100}_{\pm 0.0}$	$85_{\pm 5.0}$	$88_{\pm 2.7}$	$11_{\pm 2.2}$	$96_{\pm 4.2}$	$82_{\pm 9.1}$
RVT-2 [37]	$89.0_{\pm 0.8}$	$77_{\pm 11.0}$	$97_{\pm 4.5}$	$98_{\pm 2.7}$	$77_{\pm 13.0}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{93}_{\pm 5.7}$	$\mathbf{91}_{\pm 8.2}$	$7_{\pm 4.5}$	$\mathbf{98}_{\pm 4.5}$	$\mathbf{93}_{\pm 5.7}$
3D-LOTUS (ours)	$\mathbf{94.3}_{\pm 3.5}$	$96_{\pm 3.7}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$98_{\pm 2.5}$	$98_{\pm 4.0}$	$84_{\pm 7.4}$	$85_{\pm 9.5}$	$\mathbf{99}_{\pm 2.0}$	$77_{\pm 2.5}$	$83_{\pm 8.7}$
3D-LOTUS++ (ours)	$68.7_{\pm 0.6}$	$95_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$99_{\pm 2.0}$	$28_{\pm 2.5}$	$87_{\pm 5.1}$	$55_{\pm 10.5}$	$45_{\pm 8.9}$	$55_{\pm 8.9}$	$79_{\pm 9.7}$	$68_{\pm 12.5}$

Method	Open Drawer+2	Pick& Lift+0	Pick& Lift+2	Pick& Lift+7	PickUp Cup+8	PickUp Cup+9	PickUp Cup+11	Push Button+0	Push Button+3	Push Button+4	PutIn Cupboard+0
Hiveformer [17]	$59_{\pm 7.4}$	$86_{\pm 4.2}$	$92_{\pm 6.7}$	$93_{\pm 2.7}$	$83_{\pm 7.6}$	$69_{\pm 12.9}$	$61_{\pm 19.8}$	$84_{\pm 11.9}$	$68_{\pm 6.7}$	$87_{\pm 7.6}$	$34_{\pm 8.2}$
PolarNet [2]	$90_{\pm 7.1}$	$92_{\pm 9.1}$	$84_{\pm 7.4}$	$88_{\pm 5.7}$	$82_{\pm 7.6}$	$79_{\pm 4.2}$	$72_{\pm 10.4}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$99_{\pm 2.2}$	$52_{\pm 7.6}$
3D diffuser actor [35]	$\mathbf{97}_{\pm 4.5}$	$\mathbf{99}_{\pm 2.2}$	$99_{\pm 2.2}$	$99_{\pm 2.2}$	$96_{\pm 2.2}$	$97_{\pm 4.5}$	$98_{\pm 2.7}$	$98_{\pm 2.7}$	$96_{\pm 4.2}$	$98_{\pm 2.7}$	$85_{\pm 5.0}$
RVT-2 [37]	$94_{\pm 4.2}$	$\mathbf{99}_{\pm 2.2}$	$98_{\pm 2.7}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{99}_{\pm 2.2}$	$\mathbf{99}_{\pm 2.2}$	$\mathbf{99}_{\pm 2.2}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$88_{\pm 8.4}$
3D-LOTUS (ours)	$93_{\pm 6.0}$	$\mathbf{99}_{\pm 2.0}$	$\mathbf{100}_{\pm 0.0}$	$99_{\pm 2.0}$	$97_{\pm 4.0}$	$96_{\pm 3.7}$	$94_{\pm 4.9}$	$99_{\pm 2.0}$	$99_{\pm 2.0}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{89}_{\pm 5.8}$
3D-LOTUS++ (ours)	$75_{\pm 4.5}$	$97_{\pm 6.0}$	$94_{\pm 3.7}$	$93_{\pm 5.1}$	$86_{\pm 8.0}$	$88_{\pm 6.8}$	$91_{\pm 4.9}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$1_{\pm 2.0}$

Method	PutIn Cupboard+3	PutMoney InSafe+0	PutMoney InSafe+1	Reach& Drag+14	Reach& Drag+18	Slide Block+0	Slide Block+1	Stack Blocks+30	Stack Blocks+36	Stack Blocks+39
Hiveformer [17]	$74_{\pm 6.5}$	$85_{\pm 3.5}$	$88_{\pm 2.7}$	$37_{\pm 5.7}$	$32_{\pm 7.6}$	$99_{\pm 2.2}$	$91_{\pm 12.4}$	$6_{\pm 5.5}$	$7_{\pm 4.5}$	$6_{\pm 4.2}$
PolarNet [2]	$\mathbf{88}_{\pm 4.5}$	$93_{\pm 4.5}$	$95_{\pm 5.0}$	$99_{\pm 2.2}$	$99_{\pm 2.2}$	$\mathbf{100}_{\pm 0.0}$	$0_{\pm 0.0}$	$34_{\pm 10.8}$	$30_{\pm 9.4}$	$36_{\pm 12.9}$
3D diffuser actor [35]	$82_{\pm 11.5}$	$\mathbf{95}_{\pm 5.0}$	$98_{\pm 2.7}$	$\mathbf{100}_{\pm 0.0}$	$99_{\pm 2.2}$	$\mathbf{100}_{\pm 0.0}$	$89_{\pm 4.2}$	$88_{\pm 7.6}$	$85_{\pm 6.1}$	$89_{\pm 5.5}$
RVT-2 [37]	$80_{\pm 6.1}$	$93_{\pm 8.4}$	$96_{\pm 8.5}$	$85_{\pm 10.0}$	$94_{\pm 2.2}$	$\mathbf{100}_{\pm 0.0}$	$37_{\pm 6.7}$	$88_{\pm 5.7}$	$\mathbf{93}_{\pm 2.7}$	$88_{\pm 11.5}$
3D-LOTUS (ours)	$72_{\pm 11.2}$	$94_{\pm 3.7}$	$\mathbf{99}_{\pm 2.0}$	$99_{\pm 2.0}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{100}_{\pm 0.0}$	$\mathbf{94}_{\pm 5.8}$	$91_{\pm 6.6}$	$\mathbf{90}_{\pm 4.5}$
3D-LOTUS++ (ours)	$2_{\pm 2.5}$	$22_{\pm 6.8}$	$16_{\pm 4.9}$	$94_{\pm 3.7}$	$62_{\pm 8.7}$	$\mathbf{100}_{\pm 0.0}$	$65_{\pm 5.5}$	$86_{\pm 5.8}$	$20_{\pm 4.5}$	$28_{\pm 13.6}$

Figure 6: Performance on GemBench Level 1

3.2 Model limitation

One of the main limitation of the model is that its computation are made statically, hence the trajectory is fully computed after camera's images treatment. Knowing so, the inherent capabilities of the model to work with simple moving object is inexistent. We proposed some architecture to resolves this in the last part of our report.

In addition, even if the model 3D-LOTUS ++ achieves state-of-the art results for level 3 and 4, it is necessary to notices that the model achieves state-of-the-art on only 3 tasks for level 3, which is less than most of his competitors, but achieves overall better results. As for level 4, as we can see below, only button and tower related task find some success, which is really a novelty in itself, since no other compared model can do so, but leaves a lot of room for improvement in other level 4 tasks.

Method	Avg.	Close Door+0	Close Box+0	Close Fridge2+0	CloseLaptop Lid2+0	Close Microwave2+0	Open Door2+0	Open Box2+0
Hiveformer	$35.1_{\pm 1.7}$	$0_{\pm 0.0}$	$1_{\pm 2.2}$	$34_{\pm 9.6}$	$52_{\pm 9.1}$	$15_{\pm 7.1}$	$32_{\pm 11.5}$	$5_{\pm 3.5}$
PolarNet	$38.5_{\pm 1.7}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$78_{\pm 5.7}$	$26_{\pm 8.2}$	$74_{\pm 6.5}$	$33_{\pm 6.7}$	$23_{\pm 8.4}$
3D diffuser actor	$37.0_{\pm 2.2}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$\mathbf{97}_{\pm 2.7}$	$23_{\pm 6.7}$	$88_{\pm 7.6}$	$\mathbf{86}_{\pm 7.4}$	$\mathbf{67}_{\pm 9.8}$
RVT-2	$36.0_{\pm 2.2}$	$1_{\pm 2.2}$	$2_{\pm 2.7}$	$72_{\pm 6.7}$	$42_{\pm 14.0}$	$71_{\pm 8.9}$	$79_{\pm 6.5}$	$5_{\pm 6.1}$
3D-LOTUS (ours)	$38.1_{\pm 1.1}$	$0_{\pm 0.0}$	$\mathbf{58}_{\pm 8.1}$	$36_{\pm 9.7}$	$\mathbf{54}_{\pm 10.7}$	$85_{\pm 7.1}$	$42_{\pm 6.8}$	$11_{\pm 6.6}$
3D-LOTUS++ (ours)	$\mathbf{41.5}_{\pm 1.8}$	$\mathbf{1}_{\pm 2.0}$	$29_{\pm 8.6}$	$93_{\pm 2.5}$	$50_{\pm 9.5}$	$\mathbf{99}_{\pm 2.0}$	$52_{\pm 10.3}$	$16_{\pm 8.0}$

Method	Open Drawer2+0	Open Drawer3+0	OpenDrawer Long+0	OpenDrawer Long+1	OpenDrawer Long+2	OpenDrawer Long+3	Toilet SeatUp+0	Open Fridge+0
Hiveformer	$59_{\pm 11.9}$	$39_{\pm 11.9}$	$78_{\pm 8.4}$	$82_{\pm 4.5}$	$49_{\pm 4.2}$	$57_{\pm 11.5}$	$6_{\pm 4.2}$	$0_{\pm 0.0}$
PolarNet	$\mathbf{91}_{\pm 4.2}$	$29_{\pm 8.2}$	$84_{\pm 11.9}$	$\mathbf{88}_{\pm 5.7}$	$\mathbf{63}_{\pm 8.4}$	$37_{\pm 7.6}$	$2_{\pm 2.7}$	$4_{\pm 2.2}$
3D diffuser actor	$19_{\pm 8.2}$	$1_{\pm 2.2}$	$15_{\pm 5.0}$	$35_{\pm 13.7}$	$26_{\pm 9.6}$	$\mathbf{79}_{\pm 12.9}$	$0_{\pm 0.0}$	$\mathbf{7}_{\pm 5.7}$
RVT-2	$81_{\pm 11.9}$	$0_{\pm 0.0}$	$\mathbf{84}_{\pm 8.2}$	$39_{\pm 10.8}$	$11_{\pm 8.9}$	$75_{\pm 6.1}$	$7_{\pm 5.7}$	$0_{\pm 0.0}$
3D-LOTUS (ours)	$90_{\pm 3.2}$	$22_{\pm 8.1}$	$56_{\pm 13.9}$	$33_{\pm 11.2}$	$17_{\pm 8.1}$	$75_{\pm 6.3}$	$0_{\pm 0.0}$	$4_{\pm 5.8}$
3D-LOTUS++ (ours)	$70_{\pm 5.5}$	$\mathbf{41}_{\pm 4.9}$	$72_{\pm 4.0}$	$52_{\pm 10.8}$	$23_{\pm 8.1}$	$78_{\pm 5.1}$	$\mathbf{8}_{\pm 5.1}$	$0_{\pm 0.0}$

Method	OpenLaptop Lid+0	Open Microwave+0	PutMoney InSafe+2	Open Drawer+1	Close Drawer+0	Close Grill+0
Hiveformer	$\mathbf{100}_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$83_{\pm 5.7}$	$44_{\pm 10.8}$
PolarNet	$\mathbf{100}_{\pm 0.0}$	$0_{\pm 0.0}$	$1_{\pm 2.2}$	$4_{\pm 4.2}$	$29_{\pm 11.9}$	$42_{\pm 11.5}$
3D diffuser actor	$\mathbf{100}_{\pm 0.0}$	$0_{\pm 0.0}$	$2_{\pm 4.5}$	$0_{\pm 0.0}$	$66_{\pm 7.4}$	$\mathbf{65}_{\pm 13.7}$
RVT-2	$93_{\pm 5.7}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$\mathbf{6}_{\pm 2.2}$	$78_{\pm 8.4}$	$9_{\pm 4.2}$
3D-LOTUS (ours)	$\mathbf{100}_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$\mathbf{87}_{\pm 8.1}$	$29_{\pm 6.6}$
3D-LOTUS++ (ours)	$86_{\pm 6.6}$	$0_{\pm 0.0}$	$\mathbf{13}_{\pm 8.1}$	$0_{\pm 0.0}$	$69_{\pm 5.8}$	$19_{\pm 13.9}$

Figure 7: Performance on GemBench Level 3

Method	Avg.	Push Buttons4+1	Push Buttons4+2	Push Buttons4+3	TakeShoes OutOfBox+0	PutItems InDrawer+0	PutItems InDrawer+2
Hiveformer	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$
PolarNet	$0.1_{\pm 0.2}$	$1_{\pm 2.2}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$
3D diffuser actor	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$
RVT-2	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$
3D-LOTUS (ours)	$0.3_{\pm 0.3}$	$3_{\pm 4.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$
3D-LOTUS++ (ours)	$\mathbf{17.4}_{\pm 0.4}$	$\mathbf{76}_{\pm 7.4}$	$\mathbf{49}_{\pm 8.6}$	$\mathbf{37}_{\pm 8.1}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$

Method	PutItems InDrawer+4	Tower4+1	Tower4+3	Stack Cups+0	Stack Cups+3	PutAllGroceries InCupboard+0
Hiveformer	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$
PolarNet	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$
3D diffuser actor	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$
RVT-2	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$
3D-LOTUS (ours)	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$
3D-LOTUS++ (ours)	$0_{\pm 0.0}$	$\mathbf{17}_{\pm 10.8}$	$\mathbf{30}_{\pm 13.4}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$	$0_{\pm 0.0}$

Figure 8: Performance on GemBench Level 4

4 Our experiments

For the experiments part, our work is based on the GitHub provided by the authors, where they have released the code of their models and their checkpoints obtained after the training.

After several problems, we manage to install all the dependencies necessary to run the evaluation of the model. To do so, we created an Amazon EC2 instance on AWS with an Ubuntu 20.04 operating system and a T4 GPU. We first tried with the default CentOS operating system provided by Amazon but it didn't work because the *open3d* library didn't work on this operating system, so we moved on to the Ubuntu 20.04 operating system. After that another error appeared, this time, it was related to the *flash attention* library: the version given in the *requirements.txt* file was not adapted for a T4 GPU, therefore we created a fork of the official repository to change the version of the library (the authors used an A100 GPU). Moreover, we had to change some name functions from this library because they have changed between the two versions. Finally, we had to change some path names adapted to our config and remove slurms commands (which is probably related to the fact that the authors used a HPC resources from GENCI-IDRIS).

Once we have done all of this which was not so simple as expected we decided to reproduce some results given in the article by evaluating the GemBench Level 2 tasks with the checkpoints of 3D-LOTUS given by the authors. We can see it in the figure 9 where the "MVA students" line corresponds to our evaluation and the "ours" line corresponds to the results of the authors given in the paper.

We can see that our results are similar to the one of the authors on average which is coherent because we used their checkpoints.

After doing that, we decided to create our own task to better understand how the authors created their benchmark. As GemBench [8] is based on RLBench [4] work. We followed a tutorial provided on the RLBench repository to create our own task (for that we created a fork of the RLBench repository). The steps to reproduce our experiments: create and evaluate a new task with 3D-LOTUS are the following:

- Create or modify an existing task following the tutorial of the official RLBench repository here. Once you have saved your task *.py* and *.ttm* files in the right folder you can continue.
- Go in 'dependencies/RLBench/tools' and run the following command in the terminal:

```
python dataset_generator.py  --tasks reach_target_turorial1
```

Method	Avg.	Push Button+13	Push Button+15	Push Button+17	Pick& Lift+14	Pick& Lift+16	Pick& Lift+18	PickUp Cup+10	PickUp Cup+12	PickUp Cup+13
Hiveformer	26.1±1.4	97±2.7	85±10.0	88±2.7	21±6.5	9±4.2	8±6.7	30±7.1	22±13.5	26±10.6
PolarNet	37.1±1.4	100±0.0	**100**±0.0	85±7.9	3±4.5	1±2.2	0±0.0	48±11.0	46±8.9	16±6.5
3D diffuser actor	43.4±2.8	87±13.0	81±6.5	60±9.4	9±4.2	18±9.1	0±0.0	84±5.5	60±11.7	62±13.0
RVT-2	51.0±2.3	**100**±0.0	**100**±0.0	**100**±0.0	47±7.6	29±9.6	8±4.5	81±8.2	59±9.6	72±9.7
3D-LOTUS (MVA students)	49.5	100	100	100	0	25	30	85	75	70
3D-LOTUS (ours)	49.9±2.2	99±2.0	**100**±0.0	**100**±0.0	3±2.5	18±8.7	33±9.3	**89**±3.7	78±8.7	57±7.5
3D-LOTUS++ (ours)	**64.5**±0.9	99±2.0	**100**±0.0	99±2.0	**94**±3.7	**96**±3.7	**95**±3.2	79±4.9	**89**±9.7	**84**±10.2

Method	Stack Blocks+24	Stack Blocks+27	Stack Blocks+33	Slide Block+2	Slide Block+3	Close Jar+3	Close Jar+4	LightBulb In+1	LightBulb In+2	Lamp On+0
Hiveformer	0±0.0	4±4.2	0±0.0	0±0.0	0±0.0	0±0.0	0±0.0	4±4.2	0±0.0	7±4.5
PolarNet	1±2.2	2±2.7	6±8.2	0±0.0	0±0.0	20±10.6	82±5.7	22±11.5	17±8.4	**14**±10.8
3D diffuser actor	**66**±13.9	82±2.7	50±14.6	0±0.0	0±0.0	23±16.8	82±5.7	51±17.8	**60**±10.0	7±7.6
RVT-2	18±4.5	56±16.7	45±13.7	0±0.0	1±2.2	7±7.6	77±5.7	**68**±14.4	6±6.5	0±0.0
3D-LOTUS (MVA students)	15	35	70	0	0	75	85	25	20	0
3D-LOTUS (ours)	13±8.1	40±9.5	**69**±5.8	0±0.0	0±0.0	71±5.8	90±4.5	24±4.9	41±8.6	0±0.0
3D-LOTUS++ (ours)	22±9.3	**83**±7.5	59±3.7	**27**±9.8	**5**±3.2	**98**±2.5	**96**±3.7	56±9.7	43±7.5	2±2.0

Method	Reach& Drag+5	Reach& Drag+7	PutCube InSafe+0	Pick&Lift Cylinder+0	Pick&Lift Star+0	Pick&Lift Moon+0	Pick&Lift Toy+0	PutIn Cupboard+7	PutIn Cupboard+8	Reach Target+3
Hiveformer	1±2.2	0±0.0	4±2.2	78±5.7	73±7.6	88±2.7	87±4.5	0±0.0	0±0.0	
PolarNet	61±8.2	10±6.1	**40**±14.1	93±6.7	88±8.4	93±6.7	90±3.5	0±0.0	0±0.0	
3D diffuser actor	0±0.0	64±6.5	3±2.7	**99**±2.2	43±17.9	91±9.6	30±9.4	0±0.0	**3**±4.5	
RVT-2	91±2.2	**89**±6.5	6±5.5	98±2.7	**98**±4.5	**94**±4.2	78±8.4	0±0.0	0±0.0	
3D-LOTUS (MVA students)	100	20	25	65	80	85	100	0	0	43% (3/7)
3D-LOTUS (ours)	**95**±4.5	18±10.8	25±5.5	88±8.7	69±6.6	80±8.4	**96**±3.7	0±0.0	0±0.0	
3D-LOTUS++ (ours)	94±2.0	64±12.4	37±5.1	91±2.0	94±3.7	29±6.6	71±2.0	**1**±2.0	0±0.0	

Figure 9: Performance on GemBench Level 2 tasks (ReachTarget+3 is a new task proposed by the MVA students)

replace reach_target_tutorial1 with your task name.

- It should have created a folder '/tmp/rlbench_data/' with the initial configurations for each episode for your task (.pkl files). Move this in:

  'data/gembench/test_dataset/microsteps/seed200' for example.

- Now go to 'dependancies/RLBench' and do again:

```
pip install .
```

- Go to 'assets/taskvars_instructions_new.json' and modify the file by adding your task name and the text instructions like this in my case:

```
"reach_target_tutorial1+3": [
     "reach the green target",
     "reach the green thing"
]
```

- Then always in 'assets', create a .json file with your task name: for example I created a 'taskvars_new_task.json' file like this:

```
[
    "reach_target_tutorial1+3"
]
```

- Now you have to encode the text instructions of your task. For that use the 'genrobo3d/clip_text_encoder.py' file by replacing with your instruction. It should have created a 'embeddings.npy' file.

- Now go to ‘data/experiments/gembench/3dlotus/v1/logs/‘ and modify the ‘training_config.yaml‘file. Replace line 166 in the VAL_DATASET part with the path of your new ‘embeddings.npy‘ file.
- Now you should be able to evaluate your task! For that go back to ‘robot-3dlotus‘ and run the following command in the terminal:

```
python genrobo3d/evaluation/eval_simple_policy_server.py
--expr_dir /robot-3dlotus/data/experiments/gembench/3dlotus/v1/
--ckpt_step 150000 --taskvar_file assets/taskvars_new_task.json
--seed 200 --num_demos 20
--microstep_data_dir data/gembench/test_dataset/microsteps/seed200
--record_video --video_dir videos
```

  You can remove the last parameters ‘–record_video –video_dir videos‘ to not record the video, it will be faster!

Except the first step which is explained through the tutorial, all of the other steps are not clearly explained, therefore we had to explore the files in the github to guess the files to change to evaluate only our new task, especially the embedding of the text instructions which was hidden in the training part. Therefore, we created a small tutorial for installation and the steps to evaluate a new task on 3D-LOTUS.

To explain in more details the task we created, there are three spheres levitating above the table: one is green this is the target, the two others have random colors and are distractors. We give the following text instruction to 3D-LOTUS model: "Reach the green target".

Figure 10: reach_target_tutorial1+3 task created by MVA students

We finally evaluated our new task with the 3D-LOTUS model, defining the success rate to 1 if the gripper reaches the correct green target and to 0 otherwise. In this context, we evaluate our new task on seven different initial episode configurations: for each initial episode, the position of the three spheres is chosen randomly in a box above the table, the colors of the

two distractor spheres are chosen randomly too. Finally, we obtained a success rate of 3/7 (see figure 9)!

Moreover, during the evaluation phase, we recorded the video of all the runs of each task to be able to see and better understand the weaknesses of the model when it fails and the strength when it succeeds. In the end, we obtained a folder organized as follows:

```
- videos
    - task_name+variation_number (for example close_jar_peract+3)
        - 0_SR{success_rate}
            - global.avi
            - left.avi
            - right.avi
            - wrist.avi
        - ...
        - 19_SR{success_rate}
    - ...
with success_rate = 1.0 or 0.0 depending on the succes of the run.
There are 20 initial configurations for each task, except ours with only 10.
```

You can download it here (warning it is very heavy, almost 6 Go).

5 Improvement and perspectives

5.1 Improvement suggested

The paper offers a lot of interesting ideas with great features such as a low training time models (3D LOTUS) and a way to 'generalize' it using foundation models in a very adaptable fashion (3D LOTUS ++ can achieves better results in the future with new foundation model).
However, as we discussed previously, some limitation are inherent to those models especially the pre-computation of trajection and the inability to 'discover' new object while performing his actions and to adapt to moving object.
As for the task-decomposition by LLM part, it seems unlikely to be really improved and, as discussed and tested in the research paper is clearly not the bottleneck of the model. However, we should at this first step of task planning use a LLM with a vision module to help the LLM better plan the task with some context of the initial scene.
The idea to fully use VLMs with SAM [5] is really interesting for a sharp division of the observed space and we would like to use VLMs even more :
Hence the 3D LOTUS ++ model computes a trajectory $(x_t)_{t \in \text{elementaryInstructionSet}}$ that is updated after each elementary task given by the LLM, we thought of the idea to decompose even more this into smaller step, to capture new camera information during motion and be resilient to hidden object, in the following way :
For each x_t, we follow x_t as long as we do not overcome an obstacle (that we segmented using VLM and SAM at the beginning of current instruction) and if we do so we can recompute x_t through current camera information.
So following this procedure, if an object is hidden behind an obstacle, it should be discovered and integrated to the trajectory immediately.
It is more complicated for moving object, however the described procedure allows to notice moving object as we update more frequently vision information and if embedding coordinates of an object does not match anymore with the previous coordinates (relative to robot motion) of the same object according to the semantic embedding. Then interpolating between different time step inside of x_t could give somehow an idea of the motion of the associated object for simple object motion.
Finally, since VLMs are the current bottleneck for 3D-LOTUS ++, we can hope that their results will improves making 3D-LOTUS ++ performance better.

Another improvement, comparing GemBench to other benchmark like [7] would be to offer more variations of tasks for better evaluation of models.

5.2 Perspectives

As for now, GemBench as not been used as a benchmark outside of this research paper due to his recent publication, however this benchmark offers many challenging tasks to overcome and his one of the few benchmark purposing long-horizon task.
The integration of foundation models into robotics is a promising idea that should be developed in the near future as we can see with many of recent articles such as [3] [1] or [8].

References

[1] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, and R. Julian et al. Do as i can, not as i say: Grounding language in robotic affordances. 2023.

[2] Shizhe Chen, Ricardo Garcia, Cordelia Schmid, and Ivan Laptev. Polarnet: 3d point clouds for language-guided robotic manipulation. In *Conference on Robotic Learning (CoRL)*, 2023.

[3] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. 2022.

[4] James, Stephen, Ma, Zicong, Rovick Arrojo, David, Davison, and Andrew J. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

[5] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[6] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection, 2024.

[7] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation, 2024.

[8] Cordelia Schmid Ricardo Garcia, Shizhe Chen. Towards generalizable vision-language robotic manipulation : A benchmark and llm-guided 3d policy. 2024.

[9] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. 2023.

[10] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger, 2024.