Beep beep boop boop: Methods of image grounding and LLM-based robotic manipulation

Juan Luis Lopez
Faculty of Science and Engineering
University of Groningen
Groningen, Netherlands
j.l.lopez.gonzalez@student.rug.nl

Thijs Lukkien
Faculty of Science and Engineering
University of Groningen
Groningen, Netherlands
t.lukkien.1@student.rug.nl

Abstract—**Machine learning implementations in the field of robotics is complicated. The main difficulty therein lies in grounding. Grounding refers to linking abstract information to the real world. Previous literature has used models in an increasingly more modular and specialized manner to solve this issue, rather than utilizing more broadly applicable models for multiple tasks at once. We aim to investigate two methods of grounding concerning multi-modal input and language-based robotic manipulation. To do this, we utilized three pipelines. The first pipeline was used as a baseline and was implemented using ground-truth segmentation, followed by CLIP classification, GR-ConvNet grasp prediction, and BLOOM to make robotic API calls. The second pipeline used SAM to create automatic segmentation maps instead of using the ground truth from the simulation. The third pipeline uses BLOOM to make robotic API calls, Kosmos-V2 for grounding text to the image, SAM for segmenting the object via bounding box, and GR-ConvNet for grasp prediction. Results show that all pipelines are sensitive to failure due to a cascading error effect, and there is no clear difference in performance between pipelines.**

I. Introduction

Autonomous robotics is a broad field of study, dealing with a wide range of difficulties regarding data collection, multi-task learning, and system adaptability.

According to Soori, Arezoo, and Dastres [1], one of the main difficulties in Artificial Intelligence (AI) robotics is the availability of robot-complete (RC) data.

Now, due to the power of Foundation Models (FMs), abundantly available language can be used to overcome the difficulties of general knowledge, world knowledge, reasoning, and robot instructions.

However, that is under the assumption that these abilities can be accurately grounded in the robot's sensory inputs. This has been one of the major topics of recent research.

Recent works on autonomous robotic grippers have investigated how Foundation Models (FMs), more specifically LLMs, can be used to overcome grounding challenges. These papers will be shortly discussed in Section II.

There is a trend throughout these papers in the sense that ML models are being applied in an increasingly modular fashion. That is, tasks are subdivided over increasingly many models to create simple subtasks with a low risk of error. This way, pipeline error propagation is minimized.

On the other hand, multiple sources [2] [3] argue that we are currently undergoing a paradigm shift toward Multimodal Language Models (MLMs) that are more generally applicable. Such methods can perform multiple pipeline functions at once, resulting in shorter pipelines and therefore a lower risk of pipeline error propagation.

We will refer to the approach of applying specialized models (often LLMs) to specific parts of a pipeline as the 'modular approach', and the approach of applying more broadly applicable models (in this case MLMs) to perform multiple pipeline functions as the 'modal approach'.

In this paper, we aim to conduct an exploratory investigation into the difference between modular and modal approaches in two pipelines, and compare them to a pipeline using ground truth.

II. Related Work

In 2022, LLMs were implemented to utilize modules in a robotic control pipeline [4]. Since then, research has investigated the possible roles of FMs in robotic control pipelines. In the previous example [4], it was used to produce module-calling code snippets. Later, FMs were used to flexibly interpret sensory input to generate a task-planning sequence [5]. Then, FMs were starting to get combined with other types of NNs in order to utilize the general knowledge and interpreting capabilities of FMs, while retaining the exact output capabilities of NNs for precise robot control [6]. Around the same time, LLMs started to get a central role in language interpretation and task planning, most notably performing well on long-horizon tasks [7]. By now, LLMs are often used to interpret user-input queries, not as limited to specific vocabulary as traditional word-to-method mapping methods. LLMs are also used for (long-horizon) task planning, and different types of FMs are used to link sensory input to into robot-usable knowledge and, reversely, ground robot-usable knowledge to sensory input. The first papers used one model in their pipeline [4], [5], while later research implemented multiple models (often around 3) in a modular fashion [6], [7]. The current state of the art implements a combination of 4 FMs and one classical NN in order to create a pipeline capable of performing impressive benchmark tests such as complex tool use such as launching of items with catapults into specified containers [8]. Two of these papers created new benchmarks [6], [8], as they improved upon the previous iteration of pipeline methods regarding FM use for robotic gripper control.

III. Methodology

In this section, we will discuss our experimental setup, as well as the design decisions behind our three pipelines.

We compare 3 different pipelines. The first and second pipelines are similar modular approaches, differing in the image segmentation method. The third one is a more modal approach, with different steps. This will be explained in more detail later in this section.

A. Environment

The experimental setup was simulated in a PyBullet environment [9] of a Universal Robot (UR5) with a two-fingered Robotiq 2F-140 gripper using an RGB-D camera in an eye-to-hand configuration to perceive its environment. The robot is mounted on a table and it is capable of receiving and executing queries formed by natural language instructions. For example, the robot can be queried: ”Put the pear on the left side”, and the robot would be expected to pick up the pear, wherever it is on the table, and put it on the left side of it. An example of the environment can be seen in Figure 6. The camera produces input pictures of 224x224 pixels.

B. Data

In the simulation, we used the YCB dataset [10]. This dataset contains a number of household items, from which we used 16 in the simulation. For each of these items, the dataset contains a 3D model, a label, textures, a measure of weight, and a measure of rigidity. In our experiment, only the models and labels were used. A list of the 16 objects that were used in this experiment can be found in the Table I. A list of 10 possible placing locations was used and can be found in Table II.

C. Models

There is a variety of models that are used in this experiment for each of the pipelines. The models used are all pre-trained and were not fine-tuned through retraining. The models we used are:

1) CLIP: Multimodal Language Model capable of associating images with text. It is tasked with classifying segmented images. [11]

2) SAM: Performs image segmentation. It can be automatic, based on bounding boxes, or based on selected points given as input. [12]

3) BLOOM: General LLM, similar to GPT-3. Tasked with interpreting the user input query and producing a grasp-function-calling code snippet for the appropriate object grasps. [13]

4) KOSMOS-V2: Multi-modal language model with grounding capabilities. Tasked with generating bounding boxes for the queried objects. [14]

5) GR-Convnet: CNN that predicts grasp poses based on RGB-D images. [15]

D. Baseline

As mentioned earlier, the baseline method uses a ground truth segmentation map. This pipeline is used to explore and show some of the challenges of the environment. We can then later use this to interpret the other pipelines' performance. The complete default pipeline can be seen in Figure 1.

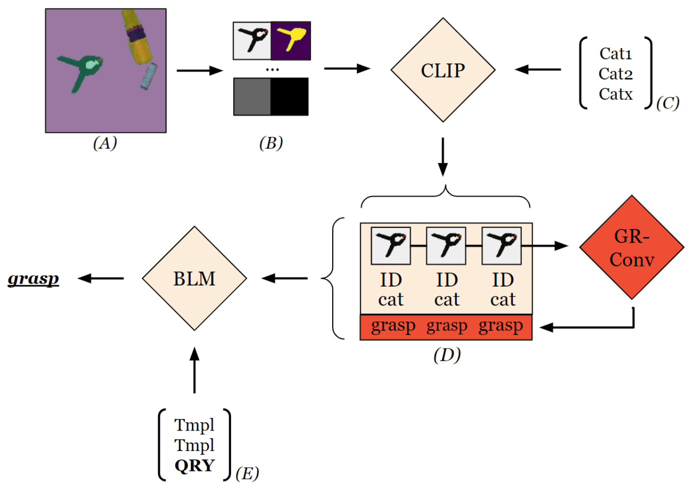

Fig. 1: The baseline pipeline. Ground truth segmentation maps are used as input to create set of n single-object images *(B)*. The single-object images *(B)*, as well as all possible category names *(C)* are fed to CLIP, which classifies each single-object image (top section of *D*). The single-object images and their depth information are fed into the GR-ConvNet in order to predict possible grasps per object (lower section of *D*). A template of possible function-calling strings and the user input query *(E)*, combined with the available object dataset *(E)* are fed to BLOOM. BLOOM then generates a function-calling string with the single-object image and grasp of whichever object best fit the input string *(E)*.

E. SAM-CLIP pipeline

Now that there is a baseline method, we can compare a similar, non-ground version of that pipeline with it. To do this, we will replicate the pipeline and implement the Segment Anything Model (SAM) to provide the segmentation map. As SAM is an MLM, it can accept different types of inputs. In this pipeline, we will use the ‘automatic segmentation’ method, in which the model accepts only an input image, and returns a set of segmentation masks. These masks are then filtered based on size such that any mask with a size larger than 2000 or smaller than 200 is filtered out. This removes backgrounds and island from the pipeline. The rest of the pipeline is identical to the baseline. The complete SAM-CLIP pipeline can be seen in Figure 2.

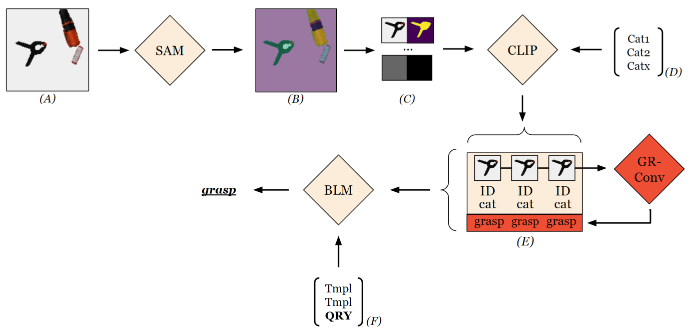

Fig. 2: The SAM-CLIP pipeline. A 224x224 pixel image is fed into the system *(A)*. SAM is then used to create a set of *n* segmentation maps *(B)*. From this point, the pipeline is identical to the baseline, as shown in Figure 1.

F. KOSMOS-SAM pipeline

So far, we have implemented a pipeline to establish ground truth performance in this environment, and a modular version of said pipeline. Now, we can create a modal approach to the current task and compare it with the previous two pipelines. As can be seen in Figure 4, this method works with the input query from the start, which means that only one bounding box and segmentation map will be produced. During the rest of the pipeline, this is a lot more efficient than the previous two approaches, where all masks and grasps were estimated and classified before being linked to the input query. To make this pipeline work, we first modified the initial prompt for BLOOM as seen in Figure 5. This prompt structure generates a set of 3 functions that are executed, once a query instruction is fed. The first function grounds a text input with the image using Kosmos, returning a bounding-box coordinates. The second function, uses the SAM with the bounding-box generated by Kosmos as input, to create a segmentation mask of the grounded object. This mask is then used on the third function, calling to pick an object based only on a segmentation mask and a target location, which internally uses GR-Convnet to predict grasp poses and moves the grasped object to the desired location.

```
objects = ["scissors", "pear", "tray"]
# put the pear to the left side.
robot.pick_and_place("pear", "left_side")
objects = ["banana", "foam_brick", "tray"]
# move the fruit to the tray.
robot.pick_and_place("banana", "tray")
```

Fig. 3: BLOOM prompt example for using SAM-CLIP and Baseline pipelines.

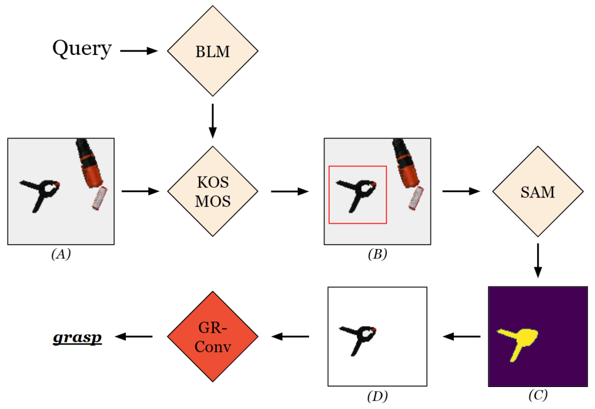

Fig. 4: The KOSMOS-SAM pipeline. A user input query is fed into BLOOM to generate a string specifying which object the user wishes to grasp. This is then fed into Kosmos with the input image *(A)* in order to produce a bounding box *(B)*. The bounding-box and image *(B)* are fed into SAM, which generates a segmentation mask *(C)*. This mask is applied to the input *(A)* to create a single-object image and depth image. These are then fed to GR-ConvNet to predict the optimal grasps. The optimal grasp is then performed.

```
# put the scissors in the tray.
bbox = robot.ground_object("scissors")
mask = robot.segmentSAM(bbox)
robot.pick_and_place_2(mask, "tray")
# put the pear to the top side.
bbox = robot.ground_object("pear")
mask = robot.segmentSAM(bbox)
robot.pick_and_place_2(mask, "top_side")
```

Fig. 5: BLOOM prompt example for using KOSMOS-SAM pipeline.

G. Experimental Setup

Each pipeline was tested throughout 20 runs. For each run, 4 quasi-randomly selected objects were picked from the list of 16 objects. The objects were placed randomly on the simulation table. One query was generated per object. Example queries can be found in the Table III. These queries were random combinations of the 4 objects and all 10 possible placing locations. With 4 objects per run, and 20 runs per experiment, each pipeline was tested using 80 objects and queries. An example run setup can be seen in Figure 6. In order to have a direct comparison between methods, a seed was utilized to ensure the same random object instances across.

IV. Results and Discussion

In our research, we created three pipelines to investigate the difference between modular and modal approaches to an object-grasping pipeline. The main results will be discussed in Section IV-A and specific cases will be discussed in Section IV-B

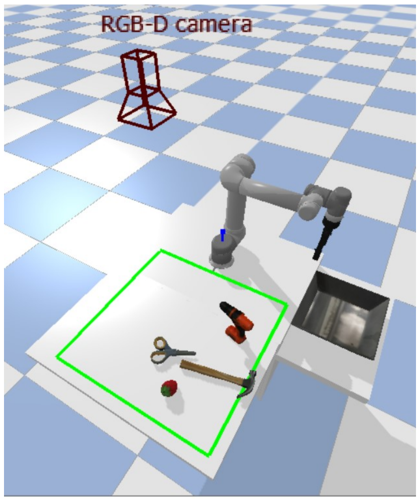

Fig. 6: Pybullet environment with 4 random objects instantiated, waiting to receive a natural language query to execute.

A. Quantitative Evaluation

For the evaluation, we have used three evaluation metrics; grasping success rate, percentage of correct grasped items, and the success percentage of the whole input query. The grasping success rate is measured by the percentage of items that were successfully lifted into the air in at least one of four attempts. The percentage of correct grasped items is calculated as 'from all successful grasps, what percentage of lifted objects were the same object as asked in the query?'. Lastly, the whole-task success rate is measured by the percentage of queries that have been completed from start to finish, with 4 allowed grasping attempts.

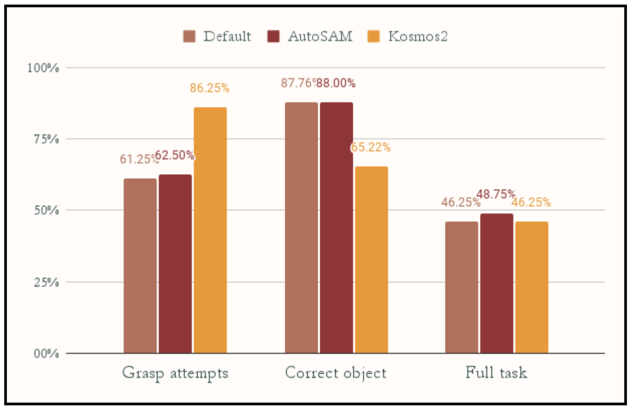

Fig. 7: Success rate per task per pipeline. "Grasp attempts" refers to successfully lifting an object gripping an object and lifting it up. "Correct object" refers to whether the correct items have been grasped, rather than another object category that was not asked for in the query. "Full task" refers to completing the complete input query.

One would think that the pipeline utilizing the ground truth segmentation from the simulation environment would have better performance than the rest. Results show otherwise, the second pipeline having slightly better grasp success.

From Figure 7, it can be seen that the Kosmos pipeline produced successful grasps much more consistently than the other two methods. However, it also grasped the incorrect objects (relatively) more often. This results in a similar final success rate across all three pipelines. This will be further discussed in the qualitative evaluation.

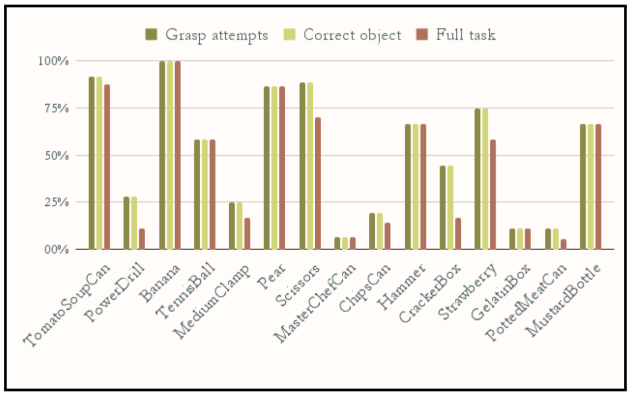

Fig. 8: Task success rate per object, averaged over all three pipelines.

In Figure 8 it can be seen how the full task success rates per object over all three pipelines. There is a lot of variance between the results of the different objects. Banana was the object that was the most easily queried with success. Overall, if the grasp attempt was successful, it will most likely reflect that the full task was successful.

B. Qualitative Evaluation

We feel as though the results are a bit mixed, but reflect the shortcomings of the different pipelines. A failure in the pipeline might introduce a cascading effect, in which the whole task is affected if a single point fails. The first failure we identified is the classification of objects using CLIP on the first and second pipelines. In Figure 9, we see that a clamp is misclassified as a hammer, which seems to happen rather consistently throughout different runs of the experiment.

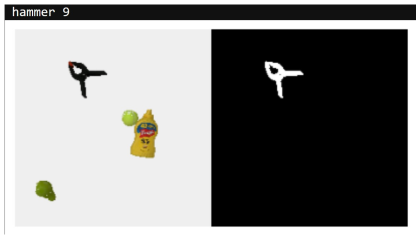

Fig. 9: Example of CLIP classifying a clamp as a hammer.

For the second pipeline, the object segmentation is done automatically. This should introduce more risk of failure since object segmentation is a complicated problem. For example, in Figure 10, we see the automatic segmentation performed with SAM. There are a lot of parameters to tune, but despite, our efforts to filter segmentation masks below a certain threshold area, the output includes masks that are very small (dark green dot in the can). Additionally, the scissors as a whole were not segmented and subsequently ignored from classifying them with CLIP.

One possible reason that explains similar results from both the first and second pipeline, is that if an object is difficult to segment correctly, then it may be difficult to classify or grasp correctly as well, due to its inherent complex shape. However, this is merely an intuition from working with the simulation, and further investigation is required.

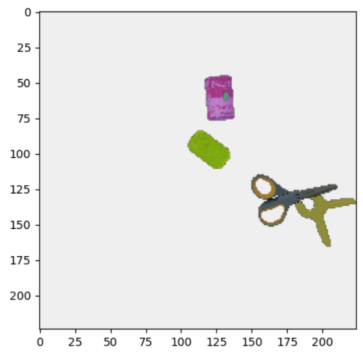

Fig. 10: Segmentation masks generated by SAM using the automatic mode. Scissors were not segmented and some segmentation masks are very small.

The third pipeline introduces another problem with object segmentation. Since it uses SAM with a bounding box prompt, instead of the automatic segmentation, it is prone to segment objects inside the bounding box area, as seen in Figure 11. The bounding box is generated using the grounding capabilities of KOSMOS, and works well overall, though it may sometimes include additional objects, as seen in Figure 12. This results in a larger set of possible grasps, increasing the chances that there is an excellent grasp (in terms of grasp quality). This could increase the chances of a successful grasp, as can be seen in the first set of columns in Figure 7 while simultaneously decreasing the percentage of correct object grasps, as can be seen in the second set of columns in Figure 7.

One advantage of the KOSMOS pipeline was its runtime performance. Performing bounding box inference and segmentation is computationally much more efficient than the automatic segmentation function from SAM. This would make it more usable in a practical application than the default and SAM-CLIP pipelines.

The reason we chose to use KOSMOS as well, was because of its capabilities as an LLM. We aimed to replace BLOOM by using a similar prompting technique to generate robotic API calls. Unfortunately, we were not able to create responses that were useful to call such API within the allocated time of this project. We think that this model might need fine-tuning to generate the API calls and ground the query object directly

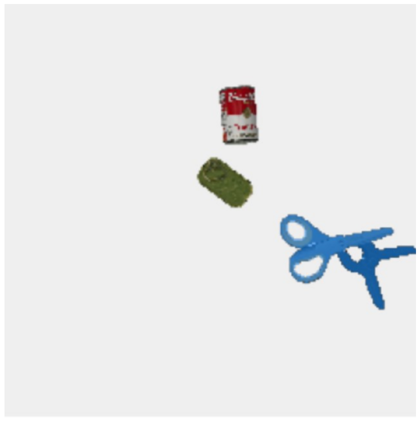

Fig. 11: Bounding box segmentation using SAM.

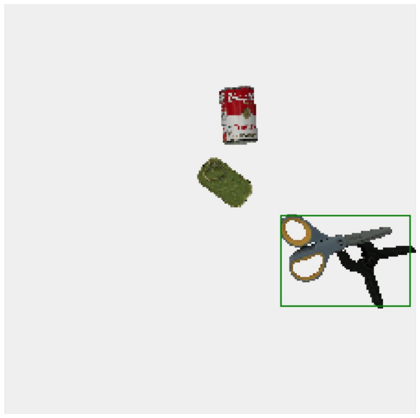

Fig. 12: Bounding box generated by KOSMOS with the prompt to ground ”scissors”.

with it. Doing this could further add to the degree of modality of this pipeline.

One major flaw of our approach is the fact that the pipelines are quite different from one another. This inhibits us from drawing strong conclusions about the efficiency of modularity versus modality. For future research, we suggest creating more similar pipelines in order to create a more solid A/B comparison.

Due to the scope of this project, we were unable to perform any ablation experiments. This, combined with the previous argument, makes it difficult to assess whether our findings stem from the difference in experiments rather than the difference in model characters. For future studies, we recommend performing a set of ablation experiments in order to optimize model choice and performance. This way, the pipelines being compared are less at risk of confounding variables such as poorly optimized or fitting models.

V. Conclusion

Foundation models have become a major part of autonomous robotics, fulfilling specialized parts in the pipeline. In this paper, we aimed to conduct an exploratory investigation into the difference between modular and modal approaches in

LLM-based robotic manipulation. Previous literature has argued that we are currently undergoing a paradigm shift towards Multimodal Language Models and we hypothesized that using such models would be beneficial. The main argument for this was that these models could bridge multiple aspects of the pipeline, hence reducing the risk of a cascading error.

The results showed no clear evidence of the superiority of one method over the other, as both the modular and modal approaches performed similarly to our baseline method. In the qualitative evaluation, we discussed the possible bottlenecks of each pipeline, yet this did not

Our main suggestion for future research is to create two similar pipelines for the two approaches. This way, one could compare the difference between modality versus modularity more clearly, as the confounding variables from the rest of the pipelines have less impact on the final results.

Finally, ablation experiments could further reduce the effect of model choice and optimization, further increasing the focus on the performance difference between modality and modularity.

We appreciate the base environment provided to us, work which was essential for this experiment. We have uploaded the code to the group9 branch from the GitHub repository linked in the appendix. Additionally, we make our Google Colab environment available to set up the environment and run the experiments.

VI. Author's Contribution

Contributions were divided equally on the report.

VII. References

[1] M. Soori, B. Arezoo, and R. Dastres, "Artificial intelligence, machine learning and deep learning in advanced robotics, a review," *Cognitive Robotics*, vol. 3, pp. 54–70, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667241323000113

[2] R. Bommasani, D. A. Hudson, E. Adeli, R. B. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. S. Chatterji, A. S. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. D. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. S. Krass, R. Krishna, R. Kuditipudi, and et al., "On the opportunities and risks of foundation models," *CoRR*, vol. abs/2108.07258, 2021. [Online]. Available: https://arxiv.org/abs/2108.07258

[3] C. Li, Z. Gan, Z. Yang, J. Yang, L. Li, L. Wang, and J. Gao, "Multimodal foundation models: From specialists to general-purpose assistants," 2023.

[4] 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," 2022.

[5] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, "PaLM-E: An Embodied Multimodal Language Model." arXiv, 2023.

[6] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, B. Ichter, T. Xiao, P. Xu, A. Zeng, T. Zhang, N. Heess, D. Sadigh, J. Tan, Y. Tassa, and F. Xia, "Language to rewards for robotic skill synthesis," 2023.

[7] Z. Wu, Z. Wang, X. Xu, J. Lu, and H. Yan, "Embodied Task Planning with Large Language Models," 2023.

[8] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," 2023.

[9] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[10] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[11] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *CoRR*, vol. abs/2103.00020, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[12] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023.

[13] B. Workshop, :, T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, J. Tow, A. M. Rush, S. Biderman, A. Webson, P. S. Ammanamanchi, T. Wang, B. Sagot, N. Muennighoff, A. V. del Moral, O. Ruwase, R. Bawden, S. Bekman, A. McMillan-Major, I. Beltagy, H. Nguyen, L. Saulnier, S. Tan, P. O. Suarez, V. Sanh, H. Laurençon, Y. Jernite, J. Launay, M. Mitchell, C. Raffel, A. Gokaslan, A. Simhi, A. Soroa, A. F. Aji, A. Alfassy, A. Rogers, A. K. Nitzav, C. Xu, C. Mou, C. Emezue, C. Klamm, C. Leong, D. van Strien, D. I. Adelani, D. Radev, E. G. Ponferrada, E. Levkovizh, E. Kim, E. B. Natan, F. D. Toni, G. Dupont, G. Kruszewski, G. Pistilli, H. Elsahar, H. Benyamina, H. Tran, I. Yu, I. Abdulmumin, I. Johnson, I. Gonzalez-Dios, J. de la Rosa, J. Chim, J. Dodge, J. Zhu, J. Chang, J. Frohberg, J. Tobing, J. Bhattacharjee, K. Almubarak, K. Chen, K. Lo, L. V. Werra, L. Weber, L. Phan, L. B. allal, L. Tanguy, M. Dey, M. R. Muñoz, M. Masoud, M. Grandury, M. Šaško, M. Huang, M. Coavoux, M. Singh, M. T.-J. Jiang, M. C. Vu, M. A. Jauhar, M. Ghaleb, N. Subramani, N. Kassner, N. Khamis, O. Nguyen, O. Espejel, O. de Gibert, P. Villegas, P. Henderson, P. Colombo, P. Amuok, Q. Lhoest, R. Harliman, R. Bommasani, R. L. López, R. Ribeiro, S. Osei, S. Pyysalo, S. Nagel, S. Bose, S. H. Muhammad, S. Sharma, S. Longpre, S. Nikpoor, S. Silberberg, S. Pai, S. Zink, T. T. Torrent, T. Schick, T. Thrush, V. Danchev, V. Nikoulina, V. Laippala, V. Lepercq, V. Prabhu, Z. Alyafeai, Z. Talat, A. Raja, B. Heinzerling, C. Si, D. E. Taşar, E. Salesky, S. J. Mielke, W. Y. Lee, A. Sharma, A. Santilli, A. Chaffin, A. Stiegler, D. Datta, E. Szczechla, G. Chhablani, H. Wang, H. Pandey, H. Strobelt, J. A. Fries, J. Rozen, L. Gao, L. Sutawika, M. S. Bari, M. S. Al-shaibani, M. Manica, N. Nayak, R. Teehan, S. Albanie, S. Shen, S. Ben-David, S. H. Bach, T. Kim, T. Bers, T. Fevry, T. Neeraj, U. Thakker, V. Raunak, X. Tang, Z.-X. Yong, Z. Sun, S. Brody, Y. Uri, H. Tojarieh, A. Roberts, H. W. Chung, J. Tae, J. Phang, O. Press, C. Li, D. Narayanan, H. Bourfoune, J. Casper, J. Rasley, M. Ryabinin, M. Mishra, M. Zhang, M. Shoeybi, M. Peyrounette, N. Patry, N. Tazi, O. Sanseviero, P. von Platen, P. Cornette, P. F. Lavallée, R. Lacroix, S. Rajbhandari, S. Gandhi, S. Smith, S. Requena, S. Patil, T. Dettmers, A. Baruwa, A. Singh, A. Cheveleva, A.-L. Ligozat, A. Subramonian, A. Névéol, C. Lovering, D. Garrette, D. Tunuguntla, E. Reiter, E. Taktasheva, E. Voloshina, E. Bogdanov, G. I. Winata, H. Schoelkopf, J.-C. Kalo, J. Novikova, J. Z. Forde, J. Clive, J. Kasai, K. Kawamura, L. Hazan, M. Carpuat, M. Clinciu, N. Kim, N. Cheng, O. Serikov, O. Antverg, O. van der Wal, R. Zhang, R. Zhang, S. Gehrmann, S. Mirkin, S. Pais, T. Shavrina, T. Scialom, T. Yun, T. Limisiewicz, V. Rieser, V. Protasov, V. Mikhailov, Y. Pruksachatkun, Y. Belinkov, Z. Bamberger, Z. Kasner, A. Rueda, A. Pestana, A. Feizpour, A. Khan, A. Faranak, A. Santos, A. Hevia, A. Unldreaj, A. Aghagol, A. Abdollahi, A. Tammour, A. HajiHosseini, B. Behroozi, B. Ajibade, B. Saxena, C. M. Ferrandis, D. McDuff, D. Contractor, D. Lansky, D. David, D. Kiela, D. A. Nguyen, E. Tan, E. Baylor, E. Ozoani, F. Mirza, F. Ononiwu, H. Rezanejad, H. Jones, I. Bhattacharya, I. Solaiman, I. Sedenko, I. Nejadgholi, J. Passmore, J. Seltzer, J. B. Sanz, L. Dutra, M. Samagaio, M. Elbadri, M. Mieskes, M. Gerchick, M. Akinlolu, M. McKenna, M. Qiu, M. Ghauri, M. Burynok, N. Abrar, N. Rajani, N. Elkott, N. Fahmy, O. Samuel, R. An, R. Kromann, R. Hao, S. Alizadeh, S. Shubber, S. Wang, S. Roy, S. Viguier, T. Le, T. Oyebade, T. Le, Y. Yang, Z. Nguyen, A. R. Kashyap, A. Palasciano, A. Callahan, A. Shukla, A. Miranda-Escalada, A. Singh,

B. Beilharz, B. Wang, C. Brito, C. Zhou, C. Jain, C. Xu, C. Fourrier, D. L. Periñán, D. Molano, D. Yu, E. Manjavacas, F. Barth, F. Fuhrimann, G. Altay, G. Bayrak, G. Burns, H. U. Vrabec, I. Bello, I. Dash, J. Kang, J. Giorgi, J. Golde, J. D. Posada, K. R. Sivaraman, L. Bulchandani, L. Liu, L. Shinzato, M. H. de Bykhovetz, M. Takeuchi, M. Pàmies, M. A. Castillo, M. Nezhurina, M. Sänger, M. Samwald, M. Cullan, M. Weinberg, M. D. Wolf, M. Mihaljcic, M. Liu, M. Freidank, M. Kang, N. Seelam, N. Dahlberg, N. M. Broad, N. Muellner, P. Fung, P. Haller, R. Chandrasekhar, R. Eisenberg, R. Martin, R. Canalli, R. Su, R. Su, S. Cahyawijaya, S. Garda, S. S. Deshmukh, S. Mishra, S. Kiblawi, S. Ott, S. Sang-aroonsiri, S. Kumar, S. Schweter, S. Bharati, T. Laud, T. Gigant, T. Kainuma, W. Kusa, Y. Labrak, Y. S. Bajaj, Y. Venkatraman, Y. Xu, Y. Xu, Y. Xu, Z. Tan, Z. Xie, Z. Ye, M. Bras, Y. Belkada, and T. Wolf, "Bloom: A 176b-parameter open-access multilingual language model," 2023.

[14] Z. Peng, W. Wang, L. Dong, Y. Hao, S. Huang, S. Ma, and F. Wei, "Kosmos-2: Grounding multimodal large language models to the world," 2023.

[15] S. Kumra, S. Joshi, and F. Sahin, "Antipodal robotic grasping using generative residual convolutional neural network," *CoRR*, vol. abs/1909.04810, 2019. [Online]. Available: http://arxiv.org/abs/1909.04810

VIII. Appendix

Tomato Soup Can	Power Drill	Banana	Tennis Ball
Medium Clamp	Pear	Scissors	Master Chef Can
Chips Can	Hammer	Cracker Box	Strawberry
Gelatin Box	Potted Meat Can	Mustard Bottle	

TABLE I: List of simulated objects.

Top left corner	Top side	Top right corner
Left side	Middle	Right side
Bottom left	Bottom side	Bottom right corner
Tray		

TABLE II: List of possible locations.

Put the pear in the middle
put the power drill in the tray
put the gelatine box in the left side
put the scissors in the bottom right corner

TABLE III: Example queries.

Link to google colab notebook: LINK.
Link to github repo: LINK.