LM-NavGrasp: Robotic Navigation and Grasping with Large Pre-Trained Models of Language and Vision

Venkata Harsh Suhith Muriki[1], Tianyu Wang[1], Bruce Zheng[1], and Rishab Soman[1]

***Abstract*—Human-robot interaction has been developed from multiple modals. This project investigates a combination of navigation and manipulation fields in terms of natural language and vision. We input descriptions of an object in different colors and types, and navigate the Stretch Hello-Robot to grasp the object. We simplify scope to static environment composed of furniture with various number of unknown objects on the surface. Results show a failure for the combination due to the high precision for the grasping algorithm to operate. Future work includes developing a robust grasping algorithm for better performance.**

***Index Terms*—Navigation, manipulation, path planning, experimental design**

I. Introduction

Human-Robot Interaction (HRI) has traditionally been a challenging domain due to the complexity of programming robots for precise control. The LM-NavGrasp project seeks to simplify this interaction by leveraging advancements in large pre-trained models that integrate language, vision, and action. Inspired by the LM-Nav model, our approach aims to enhance robot functionality, making it more intuitive and socially aware for users.

By incorporating large language models (LLMs), Segment Anything Model (SAM) [1], and Contrastive Language-Image Pre-training (CLIP), LM-NavGrasp enables robots to navigate environments and manipulate objects in a natural and human-centric manner. This project addresses inefficiencies in developing gesture-driven interfaces for robots, ultimately enhancing their usability in human-centric environments.

Our findings:

- Different colors cannot impede grasping accuracy, but different types of objects matter due to shadow and geometric shape for holding.
- Integration of navigation and manipulation is not possible in our model due to conflict between high requirement of distance for manipulation and navigation radar obstacle avoidance.

II. Related Work

This project is informed by numerous significant developments within the domain of human-robot interaction and robotics, with a particular emphasis on the LM-Nav framework. The LM-Nav system proficiently combines Visual Navigation (VNM) and Vision Language (VLM) models to enhance robots' ability to engage and navigate within fluctuating environments. The fundamental tenet of LM-Nav involves utilizing extensive vision-language and vision-navigation models and their general, trained knowledge, thereby enabling robots to understand their environment and maneuver effortlessly.

Building on top of the LM-Nav framework introduced by [2], our work shows the application of large pre-trained models of language, vision, and action to robot navigation in cluttered and dynamic environments. Specifically, it leverages Vision-Language Models (VLMs) for the understanding of the environment and ROS NAV to navigate to the desired goal.

Similarly, our work is inspired by the study of socially aware robot navigation through the integration of VLMs, which adapts the behavior of robots based on social cues [3]. In this way, robots could be better adapted to human-centered environments by following the social context. The work on socially adaptive navigation shares many similarities with our project of developing intuitive and efficient human-robot interactions.

Additionally, our work builds on results reported in work on goal representation for instruction following by [4]. This work demonstrated semi-supervised language interfaces capable of translating human instruction to executable command specifications. To be specific, this research is on extracting latent human intentions behind natural language instructions—for example, translating statements such as "I want to drink" into executable commands like "Bring me a bottle." The linking of the abstract linguistic expressions with practical robotic actions has given important insights into our system.

By synthesizing these research contributions, our project extends existing frameworks to create a cohesive system for natural human-robot interaction. The system embeds the integration of language processing, navigation, and manipulation capabilities, allowing robots to perform tasks more intuitively in human-centered environments.

Our work aims at building on top of those bases to create a coherent framework for natural human-robot interaction that will integrate navigation, visual perception, and manipulation in human-centric environments using the Stretch robot.

III. Method

Our approach is illustrated in Figure 1, which provides a visualization of the grasping navigation routine. The sequence begins with a human giving an instruction that indicates the desired object to grasp to the robot. Following this input, the robot begins its journey by navigating from one table to another, using its camera to identify the desired object. Once the desired object is located, the robot positions itself appropriately and the grasping mechanism is activated.

[1]Georgia Institute of Technology, Atlanta, USA {vmuriki3, twang730, bzheng44, rsoman6}@gatech.edu

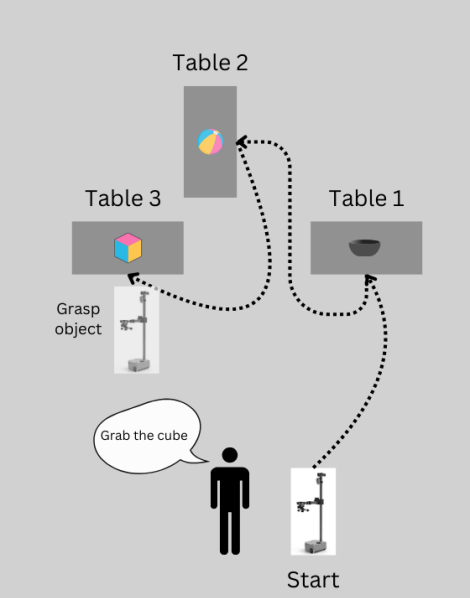

Fig. 1. Grasping Navigation Routine

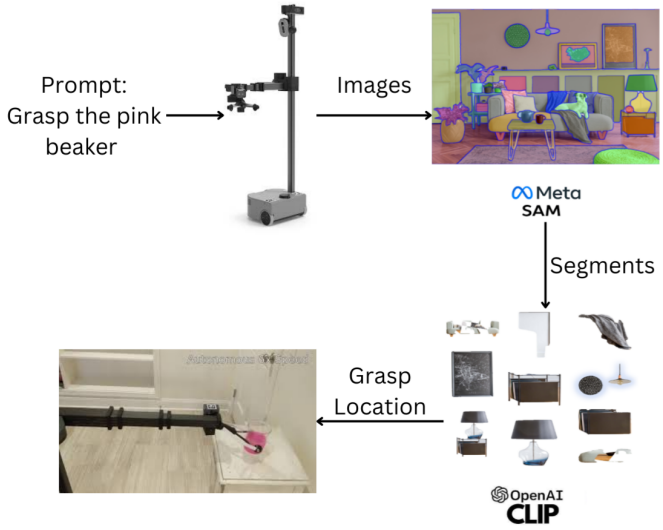

Fig. 2. Perception for Grasping

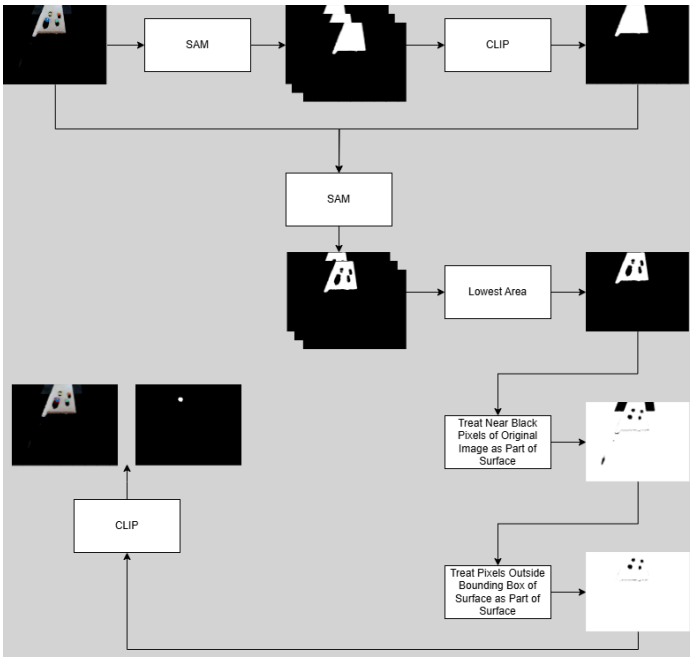

Fig. 3. Grasping Flowchart

A. Navigation

Navigation combines path planning such as A* [9] and RRT [12], and control algorithm such as PID controller. This project applies navigation stacks in Hello-Robot Stretch noetic ROS version [10], which applied DWA as local planner. The map was firstly obtained through mapping package in Stretch robot by scanning the environment. Since the environment is assumed to be static, we manually set goal locations on the map while mapping such as tables and chairs. To successfully run the grasping model, the robot has to be close enough to the table for a good grasp. However, a close distance will alert the radar to prevent robot moving. Thus, we tune the parameters of costmap's radius and ratio to smaller values so that both navigation and manipulation can work. We also remove all the objects at the bottom of the table to reduce obstacles detected by the radar. During experiments, we simplify the task by asking the robot to move straight for obstacle avoidance.

B. Grasping

Once the robot reaches the correct table, it can begin the process of grasping the desired object. A high-level overview of our grasping process is illustrated in Figure 2. The process of grasping starts with the robot capturing an image of the table using its camera. It then combines SAM [**?**] and CLIP [11] to isolate the desired object given by the user from the image, allowing it to identify the grasp location.

Figure 3 presents a more detailed diagram illustrating how the robot isolates the desired object from the image. We are utilizing a custom grasping stack built upon Hello-Robot Stretch's grasping stack which inputs a 3D RGBD Pointcloud and outputs a depth projected RGB image. This is then passed into a SAM model to get a list of object masks. We then utilize the CLIP model and apply on all the masks to get a list of probabilities to identify a wood table for an initial iteration. The mask with the highest probability for wood table is the best surface mask.

The problem with the best surface mask currently is that it does not segment the surface and the objects on it. That is, the objects are treated as part of the surface. To solve this issue, the image is again fed into the SAM model prompted with the bounding box of the best surface mask to get a list of new masks of objects on the surface of the table.

Ultimately, we want a mask for all the objects on the surface. This requires treating all pixels that are not part of an

object to be part of the surface. These pixels are those with no depth information, which appear black. In addition, all pixels outside the bounding box of the surface are not considered to be part of an object, so they are treated to be part of the surface as well.

Now that we have a mask for all the objects on the surface, we can then treat each object as a region, giving us a bounding box around each object. Each object is fed into the CLIP model again to get probabilities for each type of object. The object with the highest probability of the desired object to grasp is selected. A final mask can then be generated to isolate the desired object from everything else. The Stretch robot can then take this final mask and fit an ellipse to it and grasp the object around the ellipse's axes. One iteration of this entire process takes about 2 minutes and 15 seconds to run.

IV. Experiments

A. Navigation

Four tests were conducted for navigation and the results are shown in Table I. Here, Table 1, Table 2 and Table 3 are three different types of tables at the lab with different objects on the surface, such as a cube or a ball. We run tests to get the time taken to reach the table and if it was successful in doing that or not. Results show a huge variance for each test. Table 1 is typically stable, as the robot only needs to move straight toward the goal, with an average completion time of 21.5 seconds. In contrast, reaching Table 2 takes an average of 50 seconds, while attempts to reach Table 3 consistently result in failure.

TABLE I
Navigation Performance Across Different Tables

Test	**Time (sec)**		
	Table 1	***Table 2***	***Table 3***
1	25	61	38
2	20	44	**Failure**
3	19	44	**Failure**
4	22	41	**Failure**

This experiment shows the unstable navigation in the lab environment. After further investigation, we find that this is due to some small obstacles on the floor near the table, such as table feet because the radar cannot accurately detect them which causes bias in navigation. Another failure is caused by the bad control algorithm of ROS Nav Stack, which cannot control the robot smoothly to the goal. The robot seems to vibrate to the waypoints line, which leads to failure. In the experiment, although we calibrate the robot position and orientation with the visible marks on the ground, a small error in this alignment leads to a big error in the robot's navigation, especially when the robot compares the static map with the real-time radar input.

B. Grasping

In our grasping experiments, we evaluate how various object colors, types, and CLIP prompts influence the robot's ability to successfully grasp the correct object. We categorize the clutter level into 3 types: low clutter (1 object), moderate clutter (2 objects), and high clutter (3 objects). The metric used to assess performance is the success of the grasp (S = Successful, U = Unsuccessful). The results are presented in Tables II, III, and IV. From these, it is evident that the robot successfully grasps all objects when the clutter level is low. This result proves that our grasping perception strategy is successful when there are no other objects in the environment.

TABLE II
Color Performance Across Different Clutter Levels

Object Name	**Clutter Level**		
	Low	***Moderate***	***High***
Red Bowl	S	S	S
Green Bowl	S	S	S
Blue Bowl	S	S	S

TABLE III
Object Type Performance Across Different Clutter Levels

Object Name	**Clutter Level**		
	Low	***Moderate***	***High***
Blue Bowl	S	U	U
Blue Cube	S	S	S
Blue Ball	S	U	U

TABLE IV
Object Prompt Performance Across Different Clutter Levels

Object Name	**Clutter Level**		
	Low	***Moderate***	***High***
Green Cube	S	U	U
Green Cube with Holes	S	U	U
Green Cube with Black Outline	S	S	S

For the moderate and high cluster levels, it is important to note that the bounding box for each object is only about 15 $\times$ 15 pixels, which makes them low quality. We can see that the robot is able to differentiate objects based on color really well, which makes sense since you can still tell different colors apart even when the objects are low quality. However, for different types of objects, the robot has trouble distinguishing between a bowl and a ball. This confusion is most likely due to the objects being both circular in shape from a top-down perspective, making them difficult to tell apart in the low resolution image. For different prompts, we can see that the robot is also struggling to differentiate between green cube and green cube with holes. This is probably because of the low quality of the objects causing important details like holes to be lost. These experiments demonstrate that while our approach works well for distinguishing objects based on color, it has problems with objects that have similar shapes from a top-down view or those that require finer details for differentiation.

C. Navigation and Grasping Integration

When navigation and grasping are integrated, the resulting success rates are presented in Table V. The robot succeeded in grasping the red bowl but failed in all attempts that involved

TABLE V
NAVIGATION AND GRASPING INTEGRATION

Object Name	Table Number		
	Table 1	*Table 2*	*Table 3*
Red Bowl	S	U	U
Green Bowl	U	U	U
Green Cube	S	U	U
Blue Cube	U	U	U

more complex navigation paths due to navigation being very inconsistent. Failures in grasping the green bowl across all tables suggest that certain objects might have specific requirements or challenges that are not addressed by the current setup or strategy, possibly related to the object's color, material, or size relative to the robot's sensing and grasping mechanisms. This also highlights an integration failure, as it was successful in the previous standalone navigation and grasping experiments. For the cubes, the failures could be attributed to their smaller size and the higher precision required in both navigation to the correct location and the grasping detecting the object itself.

V. LIMITATIONS

A. Navigation

We are currently employing the ROS-Nav stack for navigation, which relies on IMU data that can be unreliable due to several factors. The accuracy of this system is heavily influenced by the robot's initial position, its speed (slower speeds generally providing more reliable results), potential skidding during movement, and the laser sensor's capability to detect obstacles that aren't present in the pre-established static map. Moreover, the navigation's path precision is consistently low, which can lead to variable distances when approaching objects for grasping tasks. This inconsistency may result in the robot's failure to accurately detect and grasp objects.

Integrating navigation with grasping tasks is challenging with this setup, unless the sensory equipment used for perception is exceptionally precise. Through our experiments and our navigation and manipulation stacks, the Hello-Robot Stretch system may not achieve such high levels of perception accuracy, and factors like human error during initialization must also be taken into account.

B. Grasping

With the current grasping framework, we send the depth projected RGB into SAM, which can be unreliable and can even fail to identify the object to be detected. One of the reasons behind this is that the depth information is very unreliable. It depends on the type of surface (cannot be too smooth), the color of the surface (cannot be white because it reflects) and the angle of the robot's head with respect to the table, which we noticed during testing. This particular framework also requires the robot to be very close to the table so that it can identify the surface, which can be troublesome because if the robot is too close to the table, navigating to the next object table fails.

Additionally, if objects are too close, due to the low resolution images, the perception models cannot properly differentiate between them, which leads to errors in the grasping coordinates. We currently hard-coded the pose of the robot head, which also leads to bad rgbd images since different types and shapes of objects work best with different poses.

VI. CONCLUSION

We conduct navigation and manipulation tasks on Hello-Robot Stretch. We apply SAM in robot perception, ROS Nav for robot navigation, and the modified stretch grasping package for grasping. Our experiment concludes that successful navigation and manipulation are achievable when the object is a "bowl" and the path planning route is simple. We found that colors of objects don't matter for the grasping, but types of objects do. We also found that due to the conflict of the navigation sensor's built-in obstacle avoidance and the grasping's high-quality image requirement, the integration is most likely not possible with the current setup.

In the future, we would like to use a dynamic map for navigation using SLAM so that it can navigate the environment in real time without getting stuck in places. We would also like to use the ROS2 NAV stack, which is reportedly more precise than the ROS NAV stack. For grasping, we would like to dynamically recognize obstacles and potential objects like chairs and tables instead of hard coding them and building a framework such that we can identify what objects and locations to go to depending on the prompt and the search object given.

REFERENCES

[1] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., & Girshick, R. (2023, April 5). Segment anything. arXiv.org. https://arxiv.org/abs/2304.02643

[2] Shah, D., Osinski, B., Ichter, B., & Levine, S. (2022). LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. arXiv preprint.*arXiv preprint*, 2022.

[3] Osinski, B., Shah, D., Ichter, B., & Levine, S. (2022). Socially aware robot navigation through scoring using vision-language models. arXiv preprint.

[4] Levine, S., Shah, D., Osinski, B., & Ichter, B. (2023). Goal representations for instruction following: A semi-supervised language interface to control. arXiv preprint.

[5] Hugging Face. (n.d.). OpenAI CLIP-VIT-large-patch14. Retrieved December 9, 2024

[6] Ichter, B., Shah, D., & Osinski, B. (2022). Visual navigation models: Learning robotic navigation. arXiv preprint.

[7] Robotics Proceedings. (2021). Learning robotic manipulation through visual planning. In Robotics: Science and Systems XV.

[8] Google Research. (n.d.). LFG-NAV Project. Retrieved December 9, 2024

[9] Foead, D., Ghifari, A., Kusuma, M. B., Hanafiah, N., & Gunawan, E. (2021). A systematic literature review of A* pathfinding. Procedia Computer Science, 179, 507-514. https://doi.org/10.1016/j.procs.2021.01.034

[10] Hello Robot Inc. (n.d.). Stretch Tutorials: ROS 1 Example 13. Retrieved December 9, 2024, from https://docs.hello-robot.com/0.2/stretch-tutorials/ros1/example_13/

[11] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021, February 26). Learning transferable visual models from Natural Language Supervision. arXiv.org. https://arxiv.org/abs/2103.00020

[12] Karaman, S., & Frazzoli, E. (2011). Sampling-based Algorithms for Optimal Motion Planning. ArXiv. https://arxiv.org/abs/1105.1186