CSCI 5541 Spring 2023 - Final Project Report
VLan GOGh: Vision & Language-guided Generalized Object Grasping

Nikhilanj Venkata Pelluri
University of Minnesota / Minneapolis, MN, USA
pellu003@umn.edu

1 Introduction

1.1 Motivation

Interactive robots have long been envisioned as effective assistants for humans since the early days of the JHU/APL desktop workstation (Seamone and Schmeisser, 1985). The use of natural language as an interface with an assistive robot is especially useful since language does not require any additional training or tools, and can be used readily in scenarios such as service robots, eldercare, etc. However, natural language is extremely complex, and practical applications require understanding of the semantic and syntactical structure of input sentences. Early interactive robots were cumbersome for users as they relied on a small fixed set of recognizable task or motion commands. However, with the advent of Transformer-based Large Language Models such as GPT-3 (Brown et al., 2020), Natural Language Understanding (NLU) capabilities have dramatically improved. Since these LLMs are trained on internet-scale corpora, they have also been shown to understand general world context, and have shown remarkable capabilities in tasks such as code completion (Chen et al., 2021), etc.
Our primary goal in this project is to leverage this real world context embedded in LLMs to build an interactive robotic system that can pick up and place objects based on a user's natural language instructions.
The capability of LLMs in understanding general context also brings out the issue of irrelevant output. In the case of robot assistants, and specifically robot grasping, we require the robots to understand the context of what objects are in the scene, and the objects that are "graspable". This means that the LLMs need to be "grounded" in the robot's affordances. To this end, we introduce the use of a vision-based system, that uses object detection to encode the objects in the scene into the LLM's local world context. This essentially "limits" the LLM's world model, and forces it to map the user's requests to objects in the scene.
Practical robot assistants also require require robust speech recognition capabilities, since speech is a general medium and does not require external tools such as keyboards or joysticks, making it usable by people who require advanced assistance e.g., people with motor disabilities. In this project, we use modern advanced speech recognition models such as (Radford et al., 2022) to transcribe user speech before passing it to the NLU system.

1.2 Related Work

Assistants and conversational agents with natural language understanding capabilities have a long history, and have been an active area of research, since the days of the Stanford SHRDLU (Winograd, 1971) system.
Robot manipulation using natural language has received significant research focus for several decades, primarily driven by the use cases in assistive and household service robots. However, most works typically rely on the user to give specific instructions to the robot, and focus on following these instructions accurately. We refer to (Tellex et al., 2020) for a detailed survey of robot systems that use language.
Recent works primarily rely on multimodal Transformers, combining embeddings from text and images to learn mappings in an end-to-end fashion. (Jiang et al., 2023), (Shridhar et al., 2022), (Lynch et al., 2022) are some recent works in this direction. However, this is significantly complex and requires extensive training. We instead pursue a modular approach, and make use of existing pretrained models to make the problem tractable.
Our projects extends these works to generalize to novel objects, and high-level user instructions.Instead of relying on specific low-level task instructions from the user, we utilize the reasoning capabilities of LLMs. Our work is most closely

related to (Ahn et al., 2022).

1.3 Broader Impact

We expect this system to be effectively assist people with motor disabilities in various settings such as elder-care homes, supermarkets, etc. The proposed pipeline also builds on advanced prompting techniques such as Chain-of-Thought prompting, that could be generally useful in a wide range of real world applications. The developed visual grounding model also helps build better robot agents which are grounded in the real world context.
We believe the current modular approach offers multiple avenues for further research - improving each module independently, as well as building an end-to-end model that reduces the need for multiple modules which reduces complexity and latency.

2 Approach

2.1 Overview

In this project, we adopt a modular approach, with different independent modules for each task. We make use of pre-trained models where applicable in order to speed up the development of the system. The user input to the system is through voice, which is then transcribed to text. A RGB camera provides visual input to the system. The system considers the user query in combination with the objects in the scene, and returns a single label referring to the object that is most relevant to the user's query. The position of the object is estimated and then passed on to the robot module to perform the further steps of grasping and placing the object in a fixed location. The system also provides speech feedback through a speaker to improve the user experience. The system architecture is described in Figure 1.
The entire system is implemented using the Python programming language and runs on the Ubuntu 20.04 operating system.

2.2 Speech Recognition

Speech input makes the system available to a wider range of users, and eliminates the need for specialized training or tools. In our project, we use the state-of-the-art OpenAI Whisper API (Radford et al., 2022) to perform speech transcription. Since the Whisper API does not have the capability to perform real-time streaming speech recognition, we first save the user speech to a file, which is then passed on to the API for transcription.
The API returns the transcribed text as plain text.

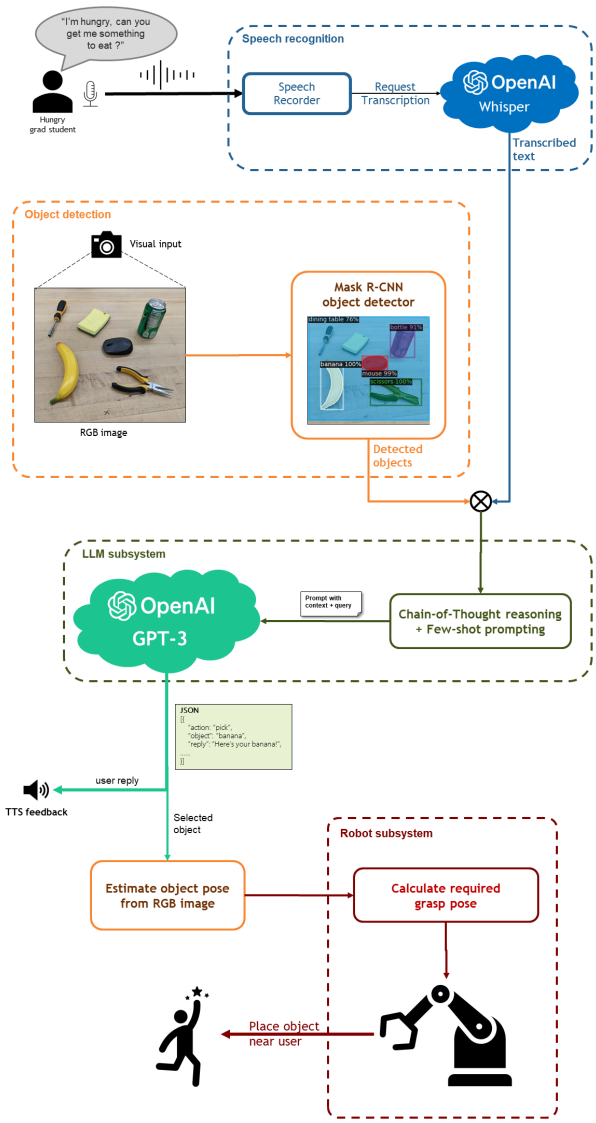

Figure 1: System Architecture.

In our project, we consider only speech in English, although the Whisper model reports state-of-the-art recognition performance on several other languages.

2.3 Visual Grounding using Object Detection

For the robot assistant to be useful, the user queries need to be grounded in the objects that are "graspable" by the robot. In the absence of this grounding, a generic user query such as "I need something to eat" is not understandable, and could refer to a wide variety of objects that are not available to the robot. To this end, we use a visual system to detect objects available in the scene, and add these objects as context for further processing.
A monocular RGB camera is fixed on the table facing the robot arm, and viewing the table. The

camera is fixed in such a way as to be able to view the entire robot workspace.
We use the Mask R-CNN object detection model (He et al., 2017) to detect objects. In our current project, we use the pre-trained model from Detectron2 (Wu et al., 2019) trained on the COCO dataset for inference. The use of an instance segmentation model instead of a simple object detection model also enables us to distinguish between multiple objects of the same class.
The labels of all detected objects are extracted and passed to the LLM subsystem, for further processing.

2.4 LLM Subsystem

In order to perform reference resolution, i.e. picking the most relevant object from the user query, we use a Large Language Model (LLM). The transcribed text from 2.2 and the object labels from 2.3 are combined in plain text into a custom prompt that is passed as input to the model. The use of a Large Language Model for reference resolution offers several benefits -

- LLMs are trained on very large scale corpora and shows state-of-the-art performance on several reasoning tasks such as question answering, commonsense reasoning, natural language inference, etc. This enables the model to better understand ambiguous or complex user queries (such as "I am thirsty") and pick objects that are more relevant to the user's request. This capability also helps LLMs stand out against traditional relevance search that depend on simple vector representations, since the traditional methods cannot adequately model complex long-form language.

- LLMs such as GPT-3 (Brown et al., 2020) shows remarkable few-shot proficiency i.e., learning using few examples instead of a large dataset, using in-context learning. This capability helps us teach the model the required task using a few manually annotated examples or "prompts", and rely on the model's few-shot learning and commonsense reasoning capabilities to performance better reference resolution.

- The GPT-3 model also shows significant improvements in reasoning when using Chain-of-Thought prompting (Wei et al., 2022). The model learns to reason sequentially about the required user query, enabling the system to respond accurately to complex user queries (such as "I am hungry, can you get me something to eat ?").

- Previous works such as (She et al., 2014) depended on complex graph-based representations for visual grounding and user expressions. We bypass these complex representations and directly teach the LLM to perform reference resolution using plain text prompting.

- Since the LLM is prompted on natural language datasets, it helps us build better conversational agents, and agents that can respond to users in "natural language" and not appear robotic. This improves the user experience, and helps reduce the friction between the assistant and users.

We use the OpenAI API with the `'text-davinci-003'` model for inference. The full prompt and the parameters for the language model are given in Appendix 5.4. The output from the GPT-3 LLM is a JSON string with the action to be taken, the object to be grasped (in case of pick action), a short reply to the user (for TTS feedback) and the reasoning (for debugging).

2.5 Robot subsystem

2.5.1 Object Pose Estimation

We consider the "pick" action for further exposition of the robot subsystem. The LLM returns the object to be picked to satisfy the user's query. From the object detection model2.3, we obtain the instance segmentation mask, which indicates the image pixels corresponding to the object. The centroid of the segmented pixels are then calculated using image moments, using the OpenCV library. We refer to (Gonzalez, 2008) for a detailed discussion on the image processing techniques involved.
Robust and generalizable object pose estimation are challenging problems and an active area of research in computer vision. In the current work, we make some important assumptions to make the problem tractable - first, all objects are assumed to be of regular polygonal shape. This enables our estimation of the centroid to match the real object's centre. Second, we assume all objects are uniformly sized, rigid, non-fragile objects. This

assumption enables us to bypass the problems of grasp force planning and instead rely on simple uniform grasping. Third, we assume all objects to be on the same horizontal plane. This helps us bypass the problem of object pose estimation and grasp planning in the vertical direction, which is more complex than the planar case.

2.5.2 Grasping and Placing

We use a 6-DOF Kinova Gen3 Lite arm, with a two-finger gripper as the robot manipulator. In our current work, we simplify the grasp approach process by assuming a uniform grasp force and a simple top-down grasp approach, which allows us to avoid the challenges associated with grasp approach pose estimation and grasp force estimation. Once we estimate the pose of the object, we execute a sequence of pre-defined vertical grasp approach poses that are relative to the object's position. Upon reaching the object grasp pose, we employ a uniform force closure to grasp the object.
After grasping the object, the object is then placed at a fixed location through a sequence of pre-defined poses, and the grasp is released.

2.6 Speech feedback

Since the user interacts with system through voice, a speech feedback in addition to the robot action provides the user with a confirmation of the task progress/completion, hence improving the user experience. This also enables the system to have a conversational dialog with the user. We use the text reply from 2.4 and convert it to speech using the pre-trained FastSpeech model (Ren et al., 2019).

3 Challenges

We encountered the following challenges when implementing the system:

- Steering/Prompting the LLM: Although advanced LLMs such as `text-davinci-003` perform well on several benchmarks, the input prompts need to be carefully crafted in order to achieve the required output. We found that the prompt was brittle and a few changes in the input prompt resulted in significant difference in results. Essentially, the model "overfits" to the format provided in the format. Furthermore, an excessively long prompt with irrelevant details would lead to misdirecting the model, whereas a very short prompt did not give the LLM enough data to accurately understand the context.

- Managing the limited token length: The `text-davinci-003` model accepts a maximum token length of 4097, i.e., 4097 tokens including the input prompt as well as output text. This acts as a limiting factor for the length of the prompt. Hence, the prompt needs to be carefully condensed so as to allow a larger output length yet containing as much information as possible to steer the LLM.

- Quirks of the LLM: We initially investigated the use of OpenAI's `'gpt-3.5-turbo'` LLM which is fine-tuned for chat conversations, in order to enable interactive dialog. However, we found it difficult to steer the LLM to output JSON responses. We found that since the model was fine-tuned for conversations, it preferred the output to be in natural language conversation, rather than the JSON output we require. This caused issues with downstream modules that depended on the output being in exact JSON format. We had to fall back to the `'text-davinci-003'` which is a less-optimized general purpose LLM.

- Engineering challenges: The envisioned system is a complex modular system with several independent modules. While this approach makes the system flexible and easily extendable, it also causes errors to propagate throughout the system, making it difficult to debug potential issues. Furthermore, since each module is a complex module with significant processing overhead, the latency with each module adds up leading to a slower system. The system requires several optimizations to keep the latency low enough to be interactive.

4 Results

4.1 Video

Some example demonstrations of the working system are available here: https://z.umn.edu/vlangogh

4.2 Observations and Limitations

We observed that the system performed well with several diverse inputs. The primary evaluation for the system was qualitative human evaluation. We

tested the system with different user queries, varied objects and varied object positions. To a large extent, the system was successfully able to understand user queries, detect objects in the scene and grasp the required objects. However, the system's reliability depends on quality of the different core modules - the object detection module, the LLM model, and the robot grasping module. We detail some errors in the existing system below.

4.2.1 Object Detection Model

We used a Mask-RCNN model trained on the COCO dataset, without further fine-tuning to detect objects in the scene. The Mask R-CNN model ranks among the state-of-the-art in object detection and instance segmentation. The model was able to robustly detect objects in a variety of positions and orientations, lighting changes, occlusions, etc. However, the model is not designed to perform zero-shot object detection, meaning that the model could not generalize well to unseen objects. This meant that direct user queries for out-of-domain objects, such as "get me the can of Sprite" could not be satisfied. A detailed discussion on the object detection model's limitations is omitted here in the interest of brevity.

4.2.2 Large Language Model

We found that the accuracy of the LLM depended significantly on the input prompt. Broadly, the LLM was able to satisfy the user request accurately even with vague or short prompts. In table 1, we highlight some sample user queries with the objects list, and the corresponding expected output against the LLM's output. In the interest of brevity, we skip the full output and highlight only the object to be picked. `None` indicates that no object in the scene can be picked.
In general, we find that the LLM outputs align with the user's requests. The LLM understands shorts vague prompts, responds to deliberate misdirection accurately, and even understands very long queries with irrelevant text.

Failure cases: We find that the LLM output is not accurate in cases where the output is vague/debatable. For example, when the user says "I feel thirsty", the LLM output is a "banana", since the the LLM assumes that a banana can, in some vague sense, quench thirst.

4.2.3 Robot subsystem

We find that the simple grasping procedure significantly reduces the robot's versatility. The enforcement of a top-down grasp limits the arm's workspace since one degree of freedom is eliminated (one joint is always enforced to be facing downward). The assumption of all objects being on the table also limits the flexibility of the system. However, relaxing these assumptions requires significant work on the grasping system, which is a major challenge in its own merit. Generalized object grasping is an active research area in robotics and general solutions are yet to be found.

5 Discussion

5.1 Replicability

The system is designed to be modular and easily replicable. The individual subsystems are easily replicable since they are standalone modules and do not require additional inputs. However, the entire system requires the use of a capable robot manipulator and an RGB camera at a minimum. Other input/output systems such as a microphone and a speaker can also be used optionally.
However, each individual module can be easily evaluated on its own merits, especially the LLM subsystem relevant to this work. The LLM subsystem requires an API key and internet access to access OpenAI's Completion API.
In our work, we set the 'temperature' of the LLM to be 0.7, leading to possibly varied output on each run. This is desirable in real world application to resolve object references when multiple equally relevant objects are present in the scene. It also promotes creative replies to the user which helps improve the user experience. However, this might lead to output not being exactly reproducible between runs.

5.2 Datasets

In our current work, we use pre-trained models where possible and do not train on our own datasets. The exact model checkpoints used for each module are listed in Appendix B

5.3 Ethics

We do not foresee major ethical risks from this work. The outputs of the LLM are grounded in the objects in the scene to reduce 'hallucinations', Furthermore, the prompt, as mentioned in appendix

User Query	Expected object	LLM output	Match
Objects list: [“banana” ,“scissors”, “mouse”]			
“hand me the banana”	banana	banana	✓
"get me the water bottle"	None	None	✓
"I feel thirsty"	None	banana	✗
"I need to cut this paper"	scissors	scissors	✓
"I want to cut my hand"	None	None	✓
Objects list: [“apple", “mouse", “bottle", “cup", “book"]			
"give me the banana"	None	None	✓
"i feel hungry. hand me something to read"	book	book	✓
"can you get me the can of sprite"	None	None	✓
"give me something other than the mouse"	<variable>	apple	✓
"I feel bored. Can you hand me something"	book	book	✓
"Today I went to the mall. It was fun. But it was very cold outside. I met my friend on the way and had some coffee with her. I then went to the bookstore and bought a hoodie for myself. I came home and slept for an hour. I feel hungry now."	apple	apple	✓
Objects list: [ ] (empty list)			
"give me the banana"	None	None	✓
"i am extremely thirsty. please give me something to drink immediately"	None	None	✓

Table 1: Sample evaluations with the user query, expected object to be picked, and the LLM’s output. The outputs are generated by setting temperature=0, to improve reproducibility.

C tries to promote shorter outputs, thus further limiting hallucinations.
We also try to prompt the LLM to reject user queries which indicate user self-harm. While this worked effectively in our limited testing, we acknowledge that this is not foolproof, and may require further work to respond appropriately to such queries.

5.4 Further Work

The system, as it currently stands, works well for a wide range of user queries in complex natural language. However, there are several areas in which the system can be improved. Some of them are listed below:

- Interactive agents: In the current work, we use a single user instruction to take decisions and perform robot actions. However, to be generally useful, we would prefer the robot to interact better with the user such as offering options, etc. With the advent of better LLMs and conversational agents, this work holds promise and would help more useful robot assistants.

- Complex multi-step tasks: In addition to interactive dialog, another step towards better assistants would be to develop the system to understand and perform complex multi-step tasks. For example, the system should be able to understand queries such as "Put the cup on the plate and hand me the plate and then the apple."

- Better LLM steering techniques: In our current work, we depend exclusively on specific prompting to steer the LLM to return the required output. However, this technique is not robust enough for general use, and creates a single point of failure. There could be further work on building better ways to steer the LLM. For example, the OpenAI API enables an optional 'logit_bias' parameter, which can be used to modulate the token probabilities. It is worth exploring if this can be used to reduce hallucinations and improve outputs with reduced inference time and cost.

- Using distilled language models: While LLMs encode a large amount of general world knowledge, for works similar to ours, a lot of the information is irrelevant. For example, we do not require information about the University of Minnesota to understand a query indicating a user's hunger (probably). We can try to use the larger model as a "teacher" to train a "student" that requires lower resources and can perform faster inference. Fine-tuning techniques such as LoRA (Hu et al., 2022) enable building faster models and are worth exploring.

- Better object detection: In our work, we used an off-the-shelf Mask R-CNN model trained on the COCO dataset. However, in recent years, there has been significant work in developing zero-shot object detection models, that can detect objects not present in the training dataset.
  Another direction in object detection could be the use of multimodal models such as CLIP (Radford et al., 2021), OWL-ViT (Minderer et al., 2022) that perform object detection directly using input natural language queries. However, these models depend on precise object definitions, and do not generalize well to vague queries. Further improvements to such models could help us perform robust object detection and reduce latency.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Rafael C Gonzalez. 2008. *Digital image processing*, 3rd ed.. edition. Pearson/Prentice Hall.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2023. VIMA: General robot manipulation with multimodal prompts.

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. 2022. Interactive language: Talking to robots in real time.

Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. 2022. Simple open-vocabulary object detection. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part X*, page 728–755, Berlin, Heidelberg. Springer-Verlag.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2019. Fastspeech: Fast, robust and controllable text to speech. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

W Seamone and G Schmeisser. 1985. Early clinical evaluation of a robot arm/worktable system for spinal-cord-injured persons. *Journal of rehabilitation research and development*, 22(1):38–57.

Lanbo She, Yu Cheng, Joyce Yue Chai, Yunyi Jia, Shaohua Yang, and Ning Xi. 2014. Teaching robots new actions through natural language instructions. *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 868–873.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. 2022. Perceiver-actor: A multi-task transformer for robotic manipulation. In *6th Annual Conference on Robot Learning*.

Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. 2020. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):25–55.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Terry Winograd. 1971. Procedures as a Representation for Data in a Computer Program for Understanding Natural Language — dspace.mit.edu. https://dspace.mit.edu/handle/1721.1/7095.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. https://github.com/facebookresearch/detectron2.

A LLM Parameters

We use the following parameters for the language model:

- Model: OpenAI `text-davinci-003`
- temperature: 0.7
- top_p: 1
- frequency_penalty: 0
- presence_penalty: 0
- logit_bias: null

Any parameters not mentioned above should be assumed the same as default/unset.

B Model Checkpoints

We use several open-source deep learning models in our work. We are grateful to the maintainers of these models for their contributions. The model checkpoints we use are listed below. Please note that these model checkpoints and weights are listed as of date of creation of this report, and may be subject to change by the model maintainers.

- Object Detection : Mask R-CNN(Resnet50-FPN 3x) from Detectron2
- Text-to-Speech: FastSpeech from Coqui TTS
- Speech Transcription: OpenAI Whisper-1
- LLM: OpenAI GPT-3.5 (text-davinci-003)

C LLM Prompt

We use the following prompt, with the detected objects and user query appended during run-time:

```
### Instructions. READ THESE INSTRUCTIONS CAREFULLY !! ###
You are a futuristic robot assistant.
You see a number of common objects which are given as a Python list
   of strings under "context".
a user will prompt you with a query in English, asking for an object,
    under "user query".
- your job is to pick the object most relevant to the user's query.
   carefully think about the reasoning step by step.
take your time to reason carefully and give the most accurate answer.
- you must return a sequence of actions as the response.
- your response must be JSON format string as defined below.
each response should be a dictionary with these items:
{"action": [ACTION], "object": [OBJECT_NAME], "user_reply": [
   REPLY_TO_USER], "reason": [CONCISE_REASONING]>}.
[ACTION] can be one of ["pick", "place", "None", "error"].
[OBJECT_NAME] muse be one of the the objects from the context. if no
   object is suggested, this must return None
```

[REPLY_TO_USER] must be a short, witty reply to the user. if the user does not mention any object in the query, you must include the object name in this reply. the reply must be warm, short and interesting. the reply must not mention the context, instead mention it as "here"
[CONCISE_REASONING] must explain your reasoning for picking a particular object concisely in less than 5 words. to choose a particular object,
you must strictly follow the below rules:
1. VERY IMPORTANT: you must return objects which are given in the context only. objects should never be from outside the context.
2. if no relevant object can be found, you must return an error in JSON format.
3. you must never ask the user to take actions that would take effort
4. if the user asks for an object which exactly matches something in the context, you must return exactly that object, without any further reasoning
6. you must suggest an object which takes the least effort for a user. for example, a banana is easier to eat than a watermelon. a glass of water is easier to drink than a cup of tea.
7. you must not respond to user queries which indicate user self-harm.

Examples
context: ['apple', 'banana', 'coffee cup', 'water bottle', 'coke can', 'peanut butter jar', 'raw egg']
1. user query: i need something to eat
reasoning: the user needs something to eat. eatable items in context are - apple, banana, peanut butter jar, raw egg. raw egg and peanut butter jar cannot be eaten directly. banana is easier to eat than apple. hence we pick a banana.
JSON output: {"action": "pick", "object": "banana", "user_reply": "<something funny>", "reason": "Banana is easy and healthy"}
2. user query: it is very cold outside. i feel thirsty
reasoning: since it is cold outside, the user would like something hot. the user is thirsty, meaning you need something to drink. the drinkable items in context are coffee cup and water bottle. we want something hot, so we pick a coffee cup.
JSON output: {"action": "pick", "object": "coffee cup", "user_reply": "Here is some coffee for you to feel better!", "reason": "coffee is warm, quenches thirst"}
3. user query: i need to turn this screw. give me something
reasoning: the user needs something to turn a screw. a screwdriver can be used to turn a screw. there is not screwdriver in the context. so the answer is None.
JSON output: {"action": "None","object": "None","user_reply": "Sorry, I cannot find anything to help turn a screw","reason": "all food items only"}
4. user query: quick ! i need to make a sandwich ! get me something !
reasoning: the user wants to make a sandwich quickly. from objects in the context, peanut butter jar and raw egg can be used in a

sandwich. peanut butter jar can help make a sandwich quickly, as raw egg needs to be cooked. so the answer is peanut butter jar.
JSON output: {"action": "pick", "object": "peanut butter jar", "user_reply": "Would you like some peanut butter for your sandwich?", "reason": "peanut butter sandwiches are popular"}
5. user query: give me the apple
reasoning: the user explicitly asks for an apple. the user must not be disobeyed, unless it is against the above rules.
JSON output: {"action": "pick", "object": "apple", "user_reply": "Your wish is my command.", "reason": "Exact match"}
IMPORTANT: verify that the output is a JSON formatted string before giving a response.
IMPORTANT: your output must not contain any apostrophes or characters which are not JSON readable
---
Real scenario
think very carefully step by step before giving the response. keep your replies short and complete the text as a JSON readable string.