Project Report: Scene Descriptions for the Visually Impaired
CSCI 5541 NLP

Mohit Yadav, Alex Besch, Abbas Booshehrian, Ruolei Zeng
yadav171@umn.edu, besch040@umn.edu, boosh002@umn.edu, zeng0208@umn.edu
Sentimentals

1 Introduction

This project aims to develop an assistive system for visually impaired individuals by integrating both RGB and depth data. It leverages the latest advancements in the cognitive capabilities of large language models to provide real-time navigation assistance and coherent language descriptions, enhancing the independence and safety of visually impaired users in dynamic environments.

1.1 Motivation

Visually impaired individuals face daily challenges in navigating their environments, particularly in detecting and avoiding obstacles and understanding the dynamics of objects around them[(1)]. The motivation behind this project was to empower these individuals by giving them a sense of their surroundings using the latest advancements in deep learning, specifically Vision and Language Models. Our goal was to develop a system that can provide descriptive insights to the user using the traditional RGB as well as the depth information. The depth data, when paired with time, can provide valuable spatial insights to the user.

1.2 Previous Work

As this was an application-based project, we analyzed the currently available products addressing the task we aimed to solve. We found two leading works in this space: one by Meta (Meta Smart Glasses) and another by EnvisionAI (EnvisionAI), both of which are currently available in the market. However, these glasses are limited [(2)] in the sense that they only use RGB color images as inputs and therefore lack explicit depth information. Due to this limitation, they are restricted to providing generic responses and descriptions to the user, which may not convey an understanding of distances – a crucial aspect for a visually impaired person.

We argue that having explicit depth information would be beneficial for a product marketed as an assistive device for the visually impaired, as depth perception can help a person better understand and navigate their environment[(3)]. While computer vision algorithms have matured significantly over the years, our objective goes beyond merely outputting numbers and detecting objects from a frame. Instead, we aim to produce coherent language outputs that enable the user to make better judgments while navigating. Additionally, there is the task of filtering objects based on their relevance to the specific task at hand[(4)]. To achieve this, we chose to employ a Large Language Model (LLM) as our cognitive brain to convert raw numbers from the computer vision algorithms into coherent language outputs that are easy for a human to understand.

In this work, we aim to use recent advancements in the cognitive capabilities of LLMs to assist people with visual impairment.

2 Approach

We envisioned our project as a product that could be used by a visually impaired person. Given the complexities of the real world, there are many circumstances the user may encounter. We selected a subset of these circumstances and focused on developing solutions, with the primary one being navigation in the environment, which would benefit most from depth modalities. Based on the problem, we hypothesize that LLMs have the cognitive ability to understand temporal and spatial scenes and, therefore, can act as a guide for visually impaired individuals by providing user-friendly descriptions of the environment, given the right data and prompts.

We discuss the modalities and our solution ap-

proach below:

2.1 Navigation

One of the biggest hurdles for visually impaired people is being able to move safely in dynamic environments. To address this, we designed the navigation module using a series of models and techniques to parse relevant obstacles and provide language cues to the user. The basic information flow of this module is shown in Figure 1.

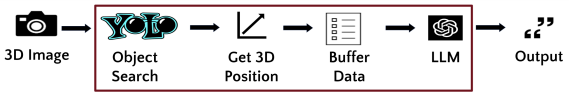

Figure 1: Navigation module pipeline. Note: models specific to the navigation module are within the maroon box.

This module required processes to run in parallel, necessitating a modular system design to complete this task. We discuss the major components of the design below.

2.1.1 Camera Input

To obtain the depth modality, we used an Intel RealSense Lidar Camera L515 capable of streaming color and depth frames of 640x480 resolution at 30fps [(5)]. The camera can be set up using the Intel RealSense SDK on Linux-based devices[(6)] (support for Mac systems is limited). Since the camera requires code to operate, we ran it in a separate thread to continuously stream frames.

Data Collection

As some of our team members were remote students, we also created a pseudo-class to mimic the actual camera behavior using pre-saved data. This allowed everyone to access the color and depth streams without requiring the actual hardware. We collected 12 videos of different environments in and around Keller Hall using the depth camera to be used for testing and evaluation of our methods.

2.1.2 Object Detection

Once we had the camera feed, we extracted information about the relevant objects in the user's view. To achieve this, we leveraged the YOLO object detector, first introduced by Redmon et al. in [(7)]. We specifically used YOLOv11 [(8)]trained on the COCO[(11)] dataset with 80 classes. The dataset includes most commonly found objects in our surroundings, which were sufficient for this project's scope. Hence, we did not need to train the system on new classes. YOLO object detectors are extremely fast, with inference times of around 10ms on consumer-grade GPUs, making them ideal for real-time systems. The YOLO model takes an RGB image as input and outputs bounding boxes and object labels for objects in the frame. Since bounding boxes are in pixel coordinates, we required correspondence between these coordinates and the real-world depth of the object, which can be assessed using the camera's depth information.

2.1.3 Object Localization in World Coordinates

Mapping pixel coordinates to world coordinates using depth information is a well-studied problem in computer vision. However, challenges arose because YOLO outputs bounding boxes rather than single pixel points, and obtaining accurate depth information was critical. Additionally, lidar sensors inherently produce noisy data, requiring filtering before use. A representative image is shown in Figure 2.

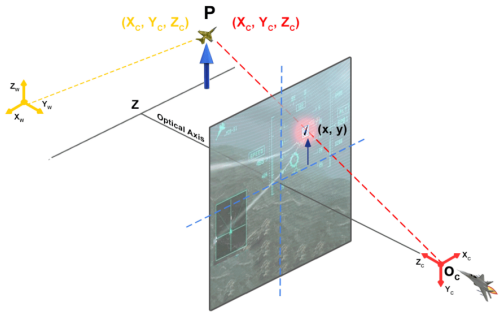

Figure 2: Projection of a point from image to world coordinates. Image Source.

To address these issues, we developed a simple filtering technique for depth information. Given a YOLO object detector's bounding box output (Figure 3), we assumed that most of the area within the bounding box is filled by the object. We then sampled 25 points randomly within the bounding box region, as shown in red. To estimate the object's depth, we calculated the median depth of these 25 points from the depth image, resulting in a robust depth estimate.

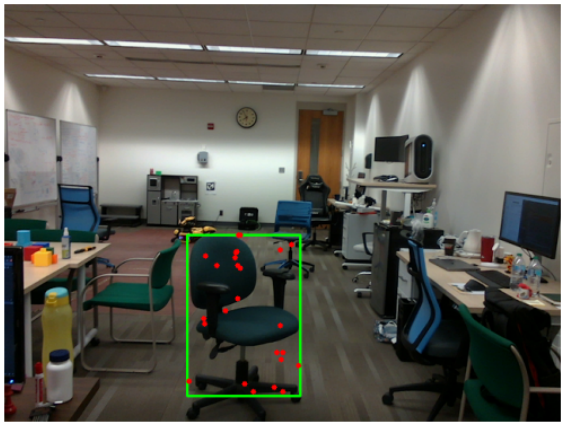

Figure 3: Randomly sampled points on an object.

Using the depth information and pixel coordinates, we estimated the object's world coordinates relative to the user. This process ran in parallel on its own thread once per second.

2.1.4 Large Language Model

From the above, we obtained a list of objects and their world locations relative to the user. However, this output, consisting mostly of numbers, was not user-friendly. A collection of objects with numerical positions is not how humans perceive the world. The previously discussed modules lacked cognitive understanding of their inputs and outputs and functioned as simple mathematical operations. To make the data more meaningful, we used a Large Language Model (LLM) as a cognitive agent to condense and filter this information into human-interpretable language.

However, we faced challenges in enabling the LLM to infer from numerical data. After discussions with our TA, Mr. James Mooney, we modified the numerical input to a format more comprehensible to the LLM. Specifically, we mapped raw numbers to descriptive terms such as "far right," "slightly left," and "in front," as shown in Figure 4. Given the camera's 70° field of view, we chose $\theta_1 = 10°$, $\theta_2 = 20°$, and $\theta_3 = 35°$. This approach resolved the issue of the LLM outputting raw numerical data to users, which was not intuitive.

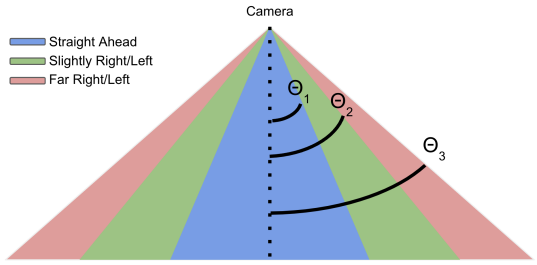

Figure 4: Labeled regions within the camera frame.

As this system was intended for dynamic environments, we provided the LLM with temporal data to understand environmental dynamics. To achieve this, we used a 15-second timeframe of camera data. To manage this efficiently without memory overhead, we implemented a double-ended queue (deque) as a buffer, achieving $O(1)$ time complexity for data addition and removal. The buffer was filled with object locations obtained using the YOLO object detector and localization function discussed earlier. A schematic showing temporal data flow is shown in Figure 5.

Using the temporal data, we employed few-shot prompting to guide the LLM to provide language outputs for the user. After experimenting, we formatted this historical data in JSON for optimal LLM performance. Additionally, to reduce repetitive responses, we included the LLM's previous five outputs as part of its input, enabling better reasoning about changes. This mitigated repetitive outputs effectively. We used OpenAI's ChatGPT-4o-mini [(12)] for inference due to its low cost and competitive performance. Sample outputs from the model are available on the project webpage: https://mohitydv09.github.io/nlpfinalprojectwebsite/.

The novelty of this work stems from the use of depth information that we haven not seen in any products available today. Two products in the same domain, Meta Ray-Ban glasses and EnvisionAI glasses, referenced earlier, only use RGB information and are not marketed as helpful in navigating the environment, but rather only as a visual information parser. We wanted to tackle the most apparent use of vision understanding for the visually impaired, i.e., navigation in the real world. In the scientific community, there are works in the space of depth understanding by LLMs, such as 3D-LLM: Injecting the 3D world into Large Language Models by Hong et al. [(13)], in which researchers have created a model to take in 3D point cloud as input and perform a diverse set of 3D-related tasks,

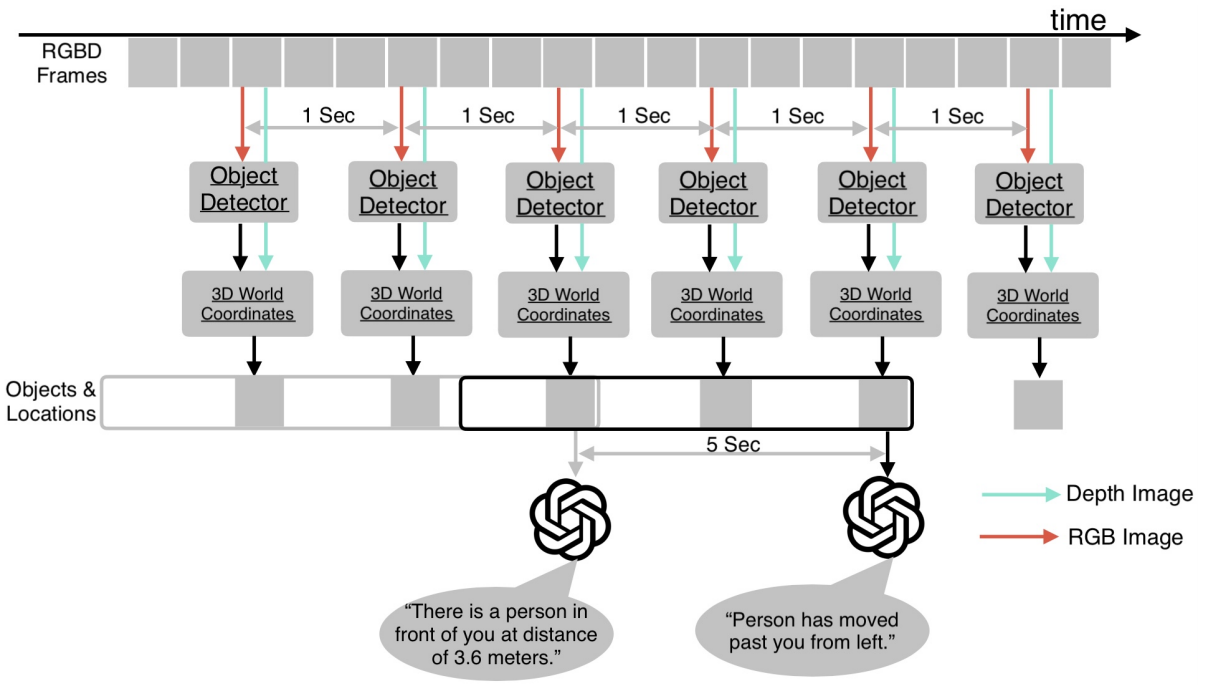

Figure 5: Schematic showing the temporal flow of data in navigation mode.

such as captioning, dense captioning, 3D question answering, task decomposition, 3D grounding, 3D-assisted dialog, navigation, and so on. However, we were unable to find any directly correlated work on the task that we were trying to solve.

2.2 Scene Description

As our project was application-based and envisioned as a product, we added this module as an add-on feature. This mode takes the current frame viewed by the user and provides a general description of the scene. For this task, we used an off-the-shelf vision-language model, specifically the BLIP model by Junnan Li et al. [(14)], available on Hugging Face.

2.3 Visual Question Answering (VQA)

Another additional application that could be helpful to the visually impaired user is to have a more detailed understanding of the environment. In addition to the $SceneDescription$, the user might be interested to know more about a specific object or area of the view. Inherently, such an application requires a question-answering environment to make sure that the specific need of the user is properly understood and responded to. For example, the the $SceneDescription$ model outputs: "you are in a classroom with a person in the middle of the class next to a whiteboard." Then the VQA model comes into play to respond to the user's follow-up questions such as: "other than the person next to the whiteboard, how many people are in front of me?" or "what color is the person's shirt?" or "is there a desk next to me?" or "what is on the desk?", etc.

To develop such capability, similar to $SceneDescription$, the BLIP model by Junnan Li et al. [(14)], available on Hugging Face was utilized. Specifically, the package used for this purpose is $blip - vqa - base$. The upside of using such a model is the efficiency of analyzing the frame and responding to the user's specific question without developing a separate engine to go back into the frame and extract the relevant information; however, the downsides are that

- the answer is very direct and summarized into a word or two or an extremely short sentence. For instance, the answer to "what color is the person's shirt?" is "black", which is non-conversational.
- the answer could be irrelevant or may require adjustment before responding to the user. For instance, the answer to "list five more major items in this room" could be "table, chair.", which is not five but two items.

We used few-shot prompting to guide the LLM to provide language output for the user. After experimenting, the final design of the conversation is described here: a) use Scene Description model to

provide a general understanding of the environment. b) feed the user's question to the BLIP VQA model. c) feed the recent user's question and VQA model response as well as the history of this conversation to the designed LLM.

As previously alluded, we used OpenAI's ChatGPT-4o-mini[9]for inference. Sample outputs from the model are available on the project webpage: https://mohitydv09.github.io/nlpfinalprojectwebsite/.

3 Application Pipeline

The primary objective of our study is to develop a tool that assists visually impaired users. To achieve this, it is essential to design a pipeline enabling users to make verbal requests and receive verbal responses from the tool. Also, we need to implement a classifier that can route the request to the corresponding developed capability. Thus, three additional features were added as a wrapper to our products including:

- Speech-to-Text: used $speech\_recognition$ Python package.
- Text-to-Speech: used $gTTS$ Python package.
- Feature Classifier: used one-shot prompting to classify the user's request between our available features. The classifier reports "Out of scope" if our app does not have a certain capability requested by the user.

4 Results and Evaluation

To evaluate our model, we opted for a human evaluation as it appeared to be the most suitable approach, given the absence of ground truth data for our work. For comparison, we used vision-language models (VLMs) as baselines, as they are the closest to our intended methodology. Specifically, we utilized BLIP[14] as our VLM. Since existing VLMs lack the ability to input depth information, they were restricted to using only the color data from the camera. Additionally, VLM prompting is limited to either a question-answering mode or an image-captioning mode, which constrains their capabilities compared to our method.

For evaluation, we tested our method and BLIP on two sample datasets we collected. Both models were run at intervals of 5 seconds, and the outputs were shown to individuals without visual impairments for assessment. To streamline the data collection process, we created a Google Form (link to Google Form), which allowed users to easily provide feedback. This approach helped us collect 22 data points for evaluating the models. We gathered data on two aspects: (1) a preference comparison between our method and the VLM baseline, based on user preferences, and (2) the perceived usefulness of the methods on a scale of 1–5.
The results of the evaluation are shown in Table 1.

Method	**Ours**	**Baseline (VLM)**
Sample 1	21/22	1/22
Sample 2	18/21	3/21
Total	39/43	4/43

Table 1: User Preference Count

Based on the user preference data we collected, we calculated the Net Preference Score (NPS) for our method, defined as:

$$\text{NPS} = \frac{\text{Votes for A} - \text{Votes for B}}{\text{Total Votes}}.$$

We achieved an NPS of 81.4% for our method across the two sample datasets.

In addition to preference data, we also asked users to rate the usefulness of the methods in describing the scene for visually impaired individuals on a scale of 1–5. The aggregated results, showing scores on the x-axis and the number of respondents assigning each score on the y-axis, are presented in Figure 6.

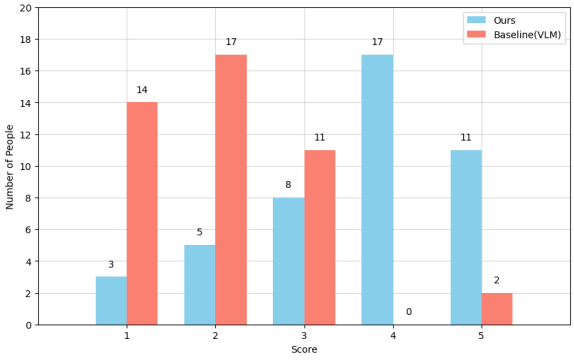

Figure 6: User-assigned scores for both models.

4.1 Evaluation Metrics

- **User Preference Comparison**: Count of user preferences between the proposed method and baseline (Table 1).

- **Net Preference Score (NPS)**: NPS of 81.4% for the proposed method.

- **Usefulness Rating**: Users rated the methods on a scale of 1–5 for describing the scene for visually impaired individuals.
  - Proposed method: mean score of 3.63
  - Baseline VLM: mean score of 2.07

4.2 Evaluation of Usefulness Scores

From the results, it is evident that our method received higher scores on the usefulness scale as well. While the baseline VLM had a mean score of 2.07, our method achieved a mean score of 3.63.

5 Limitations and Future Work

Based on the results, we can conclude that we successfully developed a system that outperforms the current method of using a VLM for the same task. However, the overall utility of the model cannot be definitively established due to the limited dataset and the absence of evaluations by individuals with visual impairments.

While the model demonstrated promise, there are several limitations in the pipeline. First, the reliance on object detectors introduces challenges, as these models are imperfect and often produce incorrect results and labels. We used a pre-trained YOLO model with only 80 classes. Although this limitation could potentially be mitigated through fine-tuning on a relevant dataset, deep learning models inherently suffer from occasional inaccurate predictions. This limitation stems from the broader field of computer vision and is beyond the scope of this project.

Another constraint was the hardware available. The Intel LiDAR camera we employed is designed for indoor use and cannot reliably produce depth information outdoors or for objects beyond 9 meters. This limitation, which became evident during testing, restricts the use case of our system to indoor environments and close objects. While this is not a limitation of our pipeline but rather of the hardware, it could be mitigated with improved equipment.

A further limitation lies in the fixed time window for making inferences with the LLM. Most large language models work by processing a stream of tokens and predicting the next word. Ensuring the LLM only produces output when needed, rather than at fixed intervals, proved challenging. Ideally, we would enable dynamic, context-aware inferences where the LLM responds only when detecting changes in the environment. Achieving this functionality would require a deep learning model capable of continuous input processing and context-dependent output generation—an area for future exploration.

Additionally, we developed three distinct modules for the product within the available time. However, we foresee the potential for numerous additional modules. Once a more extensive set of modules is developed, similar to the classifier module described in the pipeline, an LLM-based agent could be employed to infer user intent from inputs and autonomously determine which module to execute.

6 Discussion

6.1 Potential for Publication

This work was highly application-oriented, and we envisioned the system as a product, which is why we didn't focus on the potential for publishing this work. That said, there is definitely a direction in which this work could be made more research-oriented. Since we used the LLM's spatial scene understanding, there is a research opportunity in this space. We found some recent work, SceneGPT[15] by Shivam Chandhok, where the author used a similar approach to ours and enabled an LLM to understand spatial relationships between objects, leveraging a graph structure. This is very recent work, published in August 2024, which shows that research is being done in this direction.

6.2 Replicability

The work we did is replicable as long as the hardware to obtain depth data is available to the user. With LiDAR technology being adopted in various avenues, the hardware cost is also decreasing rapidly. With the right hardware, our work is easily replicable, and we have open-sourced both the code and the methodology. We would be happy to see this work contribute to assistive technologies in any way it can. We used OpenAI's ChatGPT as our LLM, which is proprietary and not available as open-source. However, the methodology is not dependent on the specific LLM used, and with some tuning, many of the openly available LLMs could be used for the same task.

6.3 Potential Impact / Ethics

Although our work showed promising results, we would not be able to claim that we have solved the challenges that visually impaired people face in their lives. This work is merely a step in the

right direction and requires additional efforts to be truly helpful to the users. It is also important to note that we did not consult any visually impaired individuals for this work; therefore, there might be gaps in our understanding of the users' needs.

During the course of this project, we also realized some of the technical challenges involved in developing our work into a product. For instance, the accuracy of depth data depends on the quality of the hardware, and LiDAR-based devices can be quite heavy for users to handle continuously. Additionally, there are computational limitations associated with deep learning models used in this pipeline, which require powerful hardware to run locally. Although this could be somewhat mitigated by using a cloud server, it introduces the potential risk of misuse of users' private data.

Lastly, with specialized hardware, there is the challenge of cost and accessibility. If the system is only affordable to a subsection of society, it could exacerbate disparities between people.

7 Conclusion

In this work, we explored the use of large language models as cognitive filters and parsers of depth and visual information for the specific use case of assisting visually impaired individuals. The results demonstrate that the concept is feasible and warrants further exploration with enhanced hardware and resources. Due to time and resource constraints, we tested the method in a limited number of scenarios. However, with additional results and fine-tuning using reward-based methods, we anticipate significant improvements in the system.

8 Individual Contributions

- Mohit Yadav: Data collection, system design, navigation module pipeline implementation, scene description module implementation, LLM integration, evaluation and result analysis, project website creation, final report writing.
- Alex Besch: Data collection, navigation module pipeline implementation, LLM Integration, evaluation and result analysis, project website, final report writing.
- Abbas Booshehrian: Project idea, Visual Question Answering module implementation, LLM Integration, Speech-to-Text integration, Text-to-speech integration, Application classifier integration, Creating demo, final report writing.
- Ruolei: Early writing of the final report, final report review.

9 Additional Materials

We created a webpage to show our work and prerecorded demos, which are available at the below link:
mohitydv09.github.io/nlpfinalprojectwebsite

The code for our project with the detailed read me file showing how to run the code can be found on our GitHub repo at the below link:

GitHub: github.com/mohitydv09/nlp-final-project

10 Acknowledgment

We extend our heartfelt gratitude to everyone who took the time to complete our evaluation form, as well as to the staff of CSCI 5541 Fall 2024, especially Prof. Dongyeop Kang (DK), for their invaluable feedback and unwavering support in helping us complete this project. A special thanks goes to Nirshal Chandra Sekar for his assistance in setting up the depth camera, and to Prof. Karthik Desingh for granting us access to the LiDAR camera.

References

[1] Z. Başgöze, J. Gualtieri, M. T. Sachs, and E. A. Cooper, "Navigational aid use by individuals with visual impairments," *PLoS One*, vol. 17, no. 3, p. e0266089, 2022

[2] "Ray-Ban Meta Glasses: Where Do The Fashionable Smart Glasses Stand On Accessibility?", *Envision Blog*, 2024.

[3] M.H. Abidi, A.N. Siddiquee, H. Alkhalefah, V. Srivastava, A comprehensive review of navigation systems for visually impaired individuals, Heliyon. 10 (2024) e31825.

[4] E. Waisberg, J. Ong, M. Masalkhi, N. Zaman, P. Sarker, A.G. Lee, A. Tavakkoli, Meta smart glasses—large language models and the future for assistive glasses for individuals with vision impairments, Eye. 38 (2024) 217–219.

[5] "Intel RealSense LiDAR Camera L515," Devie Store, 2024

[6] "Linux Distribution," Intel RealSense, 2024.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[8] "Guide to Fine-Tuning and Deploying YOLOv11 for Object Tracking," E2E Networks Blog, 2024. [Online].

[9] OpenAI, "ChatGPT" (Mar 14 version) [Large language model], https://chat.openai.com/chat

[10] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation," arXiv:2201.12086 [cs.CV], 2022.

[11] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," arXiv:1405.0312 [cs.CV], 2015.

[12] OpenAI, "ChatGPT-4: Advancing Language Model Capabilities," OpenAI, 2023. Available: https://openai.com/chatgpt.

[13] Y. Hong, H. Zhen, P. Chen, S. Zheng, Y. Du, Z. Chen, and C. Gan, "3D-LLM: Injecting the 3D World into Large Language Models," arXiv:2307.12981 [cs.CV], 2023. Available: https://arxiv.org/abs/2307.12981.

[14] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation," arXiv:2201.12086 [cs.CV], 2022. Available: https://arxiv.org/abs/2201.12086.

[15] S. Chandhok, "SceneGPT: A Language Model for 3D Scene Understanding," arXiv:2408.06926 [cs.CV], 2024. Available: https://arxiv.org/abs/2408.06926.