Baymax: HoloLens2-Based Navigation Assistant for the Visually Impaired

Nicolas Pfitzer Yifan Zhou Marco Poggensee Defne Kurtulus
ETH Zurich
{npfitzer, yifzhou, mpoggensee, dkurtulus}@student.ethz.ch

Abstract

Over 43 million people worldwide live with blindness [11], facing significant challenges in navigating unfamiliar environments. We present Baymax, a HoloLens2-based mixed reality system that enhances spatial awareness for visually impaired users through real-time scene understanding and intuitive audio feedback. Our system combines computer vision algorithms for object detection and depth estimation with natural language processing to provide contextual scene descriptions, proactive collision avoidance, and navigation instructions. The distributed architecture processes sensor data through MobileNet for object detection and employs RANSAC-based floor detection with DBSCAN clustering for obstacle avoidance. Integration with public transit APIs enables navigation with public transportation directions. Through our experiments with user studies, we evaluated both scene description and navigation functionalities in unfamiliar environments, demonstrating satisfactory usability and effectiveness. Code and demo video available here.

A. Introduction

Navigating unfamiliar environments presents significant challenges for the visually impaired population, which consists over 43 million people worldwide [11]. While various assistive technologies exist, they often struggle with real-time scene understanding and intuitive user interaction. We present *Baymax*, a HoloLens2-based navigation assistant that leverages mixed reality and computer vision to enhance spatial awareness for visually impaired users.

Our system integrates real-time object detection, depth estimation and spatial audio to provide three key functionalities: (1) contextual scene description through natural language processing, (2) multimodal collision avoidance using point cloud clustering, and (3) navigation with public transit directions. The system processes sensor data through a distributed architecture, utilizing MobileNet for efficient object detection and RANSAC-based floor detection for reliable obstacle avoidance.

We evaluate *Baymax*'s usability and effectiveness through user studies in navigation confidence and obstacle avoidance. Our main contributions include:

- A real-time scene understanding pipeline optimized for HoloLens2
- A multimodal navigation system combining spatial audio, natural language feedback and visual enhancement
- An evaluation of the system's effectiveness in real-world scenarios

B. Related Works

In both academic research and commercial development, several approaches have been explored to enhance navigation for visually impaired individuals. Bai et al. [1] demonstrated the feasibility of smart glasses for indoor navigation, utilizing computer vision techniques for obstacle detection and path planning. A notable development in dedicated assistive technology is the Dot Lumen device [6], which combines computer vision with haptic and audio feedback to provide obstacle detection and navigation assistance. However, these solutions primarily focus on providing directional guidance similar to a guide dog, offering only suggested directions and obstacle avoidance. They lack the ability to help users build a comprehensive understanding of their surroundings.

Research has demonstrated that visually impaired users require more than simple navigational directions. In a comprehensive study of blind pedestrians' needs, Brunet et al. [3] found that blind users strongly desire to form "operative mental representations" of their environment before and during navigation. Their research showed that users who could build comprehensive mental models of the space reported higher confidence and reduced anxiety during navigation. While navigation systems often focus solely on obstacle avoidance and basic directions, Brunet et al. found that blind users need "detailed information concerning various aspects of the map" including rich descriptions of street layouts, crossing points, and environmental context that allows them to "form a clearer mental representation of the

itinerary and to better anticipate difficulties" [3]. Current commercial solutions, while effective for basic obstacle avoidance, fail to provide this specific level of environmental awareness and contextual understanding that users require for confident independent navigation.

Our work builds upon these foundations while addressing their limitations through the integration of multiple feedback modalities, real-time processing, and natural language interaction. By combining advanced computer vision with conversational AI capabilities, we aim to create a system that not only guides users safely through space but also enables them to build a mental model of their environment through natural dialogue and scene descriptions.

C. App Framework

The *Baymax* system comprises four main functional components that work together to assist visually impaired users in navigating urban environments while providing them with an understanding of their unfamiliar surroundings.

C.1. User Interface

The primary interface enables hands-free control through voice commands, with a wake word "Hey Baymax" followed by specific commands. For users with partial vision, the system provides visual feedback through an interface displaying detected objects and navigation cues (Figure 2), as well as *Baymax*'s reply to the user's queries (Figure 4). The welcome page shown on Figure 1 illustrates the components of the user interface and their interactions, along with an audio tutorial.

Figure 1. **Welcome Page.** This gives the user an introduction of the commands and actions that can be performed. Saying "help" will provide the same content verbally.

C.2. Scene Description

The user can pose any question about the scene in front of them, and the app will provide concise verbal descriptions, identifying key environmental objects and their distances to the user. The user can raise follow-up questions in one conversation based on one camera frame.

C.3. Obstacle avoidance

Our system provides the user with a multimodal feedback system combining visual enhancement and spatial audio feedback. The real-time processing pipeline continuously analyzes depth sensor data to maintain accurate obstacle positions and dynamically updates the navigation path. For users with partial vision, we render detected obstacles as white cubes, while a floating blue sphere indicates the recommended safe path direction, as illustrated on Figure 2. This visual guidance is complemented by a spatial audio system that provides directional cues through binaurally rendered sounds, in the direction of the obstacle-free path.

Figure 2. **Collision Avoidance.** The white cubes show the detected obstacles. The blue sphere visualizes suggested path direction according to our obstacle avoidance algorithm D.2. The position of the blue sphere is also provided via directional audio.

C.4. Google Maps based navigation

Users can request guidance to a specific address, and the system provides through auditory feedback transit instructions based on their current location and real-time public transport data provided by the Google Maps API [9], as illustrated in Figure 4. The directions provide the user information about tram or bus lines they need to take, along with departure stop and times.

D. Implementation

The system architecture, illustrated in Figure 3, comprises two primary modules. While most user interactions are processed directly on the HoloLens device, computationally intensive tasks are offloaded to a backend server, which maintains continuous communication with the HoloLens.

Communication between the HoloLens and the PC server, including the transmission of user input and the results of algorithmic and API-based processes, is handled via HTTP requests and JSON packets. Sensor data from the HoloLens, including RGB and depth frames, is streamed to

Figure 3. **Application Pipeline.** The application consists of two primary components: a Unity application running on the HoloLens as the front end, and a Flask server hosted on a computer as the back end. User commands are sent from HoloLens to the PC server, triggering one of three modules: Scene Description, Obstacle Avoidance, or Navigation. The module processes the required video/depth data into information that is returned to the Hololens. The Navigation module also uses the current GPS location provided by a phone.

the PC using a the HoloLens2 Sensor Streaming framework *hl2ss* [4]. The PC server also receives GPS coordinates from the user's phone, used for the Google Maps feature. Most user interactions are handled through voice commands and audio feedback and all speech related communications are powered by Azure Speech Services [14].

For seamless communication while running the app, it is essential that the HoloLens, PC, and phone are connected to the same Wi-Fi network. To meet user requirements for low latency, we recommend using a reliable and high-speed Wi-Fi connection, possibly through a router.

The implementation of each feature in our app is described in the following subsections.

D.1. Scene Description

When the user activates the scene description feature, their query, along with the current RGB and depth frames from the HoloLens, is transmitted to the Flask server on the PC.

For each new frame, object detection is performed first to identify nearby objects, along with their corresponding depths. This information is used to potentially alert the user to objects in close proximity and their distances, which can be later integrated into the prompt with the image frame to the chatbot. We utilize MobileNet [17] for object detection due to its compatibility with the libraries in our implementation and its ability to achieve sufficiently fast runtimes on a CPU environment.

Figure 4. **Navigation.** Transit guidance provided by the Google Maps API to a destination address shown as captions in the app.

The depth estimation is performed before querying the Gemini API, as large language models lack the capability to accurately estimate depth from a single RGB image. Although we considered monocular depth estimation models, such as UniDepth [15], we opted to use HoloLens long-throw depth sensors due to their superior reliability and seamless integration into our pipeline.

The detected objects and their estimated depths, along with the user's query, are then passed to the Gemini API [8]. We chose to consult a chatbot for the scene description because it identifies a broader range of objects compared to our object detection model and provides responses in a user-friendly language, which is essential for our assistive application.

Finally, the chatbot's scene description is sent back to the Unity app on the HoloLens via a JSON packet and then announced to the user.

D.2. Obstacle avoidance

For obstacle avoidance, real-time depth data from the HoloLens Longthrow sensor is streamed to a back-end server along with head-tracking information. Each incoming depth frame is converted into a 3D point cloud, which is then registered to a global reference frame (GRF) using the user's head pose. To maintain near real-time performance (close to the 5 fps limit of the depth sensor), the point cloud is first downsampled using a voxelized grid approach. Performing obstacle avoidance in the GRF offers two key advantages:

- **Obstacle Tracking**: A list of previously discovered obstacles is maintained on the server and is continuously updated as the user navigates the space. By combining observations of the same obstacle from multi-

ple viewpoints, the system can refine and converge on a more accurate pose estimate. This significantly improves robustness and reliability, especially in large or cluttered environments.

- **Floor Detection**: Raw point clouds do not inherently differentiate floor points from walls, ceilings, or other obstacles. To isolate actual obstacles, the floor must be removed. In the GRF, this can be achieved by fitting a horizontal plane using RANSAC [2], which effectively filters out the floor. In the local (head-centered) reference frame, the user's head tilt would complicate this filtering, since the system would have no fixed notion of what constitutes a horizontal plane.

Figure 5. **Point Cloud Segmentation.** A visualization of the point cloud segmentation process, where the floor (shown in red) is filtered out and each obstacle is assigned a distinct color.

After the floor has been identified and removed, the remaining points in the global point cloud are segmented into clusters using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [7]. DBSCAN is particularly well-suited for this scenario because it does not require prior knowledge of the number of clusters, and it can easily separate sparse outliers from meaningful groups of points. Once the points have been grouped, each cluster is enclosed in a bounding box, creating a compact geometric representation of the obstacle. These bounding boxes are then used for collision avoidance.

A collision free heading is computed by systematically applying the following algorithm on 2D projection of the scene:

1. **Sample Angles Around the User** The algorithm imagines drawing a set of circles around the user and sampling points in all directions on each circle. Each point represents a potential heading.

2. **Check Distances to Obstacles** For every sample point, the algorithm measures how far it is from every known obstacle. If a point falls within a "danger zone" (too close to an obstacle), it's considered unsafe. If the algorithm finds a collision-free heading on a circle, it immediately stops sampling the rest of that circle's angles.

3. **Select the Safest Heading** Once the algorithm checks all circles, it picks the direction that has the greatest clearance from obstacles and immediately stores that as the best heading.

In essence, the code is systematically testing possible orientations in an expanding ring around the user, quickly zeroing in on whichever angle gives the most obstacle-free path. The algorithm is further accelerated by parallelizing clockwise and counter-clockwise searches.

Figure 6. **Heading diagram.** The diagram illustrates how the algorithm samples potential directions counter-clockwise around the user's current position (labeled "HEAD") by expanding outwards through multiple safety circles (shown in dashed blue). Each colored circle (e.g., "Sample 1.1," "Sample 2.2," "Sample 3.2") represents a candidate heading at that distance, with obstacles appearing as larger orange circles. Red samples are "invalid" because they are too close to an obstacle, while green samples offer enough clearance to pass safely. The algorithm ultimately returns the "Heading Target", which is the valid heading with the smallest absolute angle to the "Default Heading".

D.3. Google maps based navigation

For the Google Maps-based navigation feature, the user's destination address from the Unity app, along with their GPS coordinates from their phone, are sent to the PC server. The GPS coordinates are transmitted via the GPS2IP Lite [10] app, installed on user's phone.

The Google Maps Directions API [9] in Python returns relevant tram or train details to the destination — such as lines, times, and departure stops — along with GPS coordinates for the tram stops. This data is transmitted to the HoloLens app and the rail instructions are played through audio to the user.

Upon request, the app can also provide walking directions to the next tram stop, while continuously comparing user's current GPS coordinates with those of the next walking waypoint on the path to destination, to play the next in-

struction at the appropriate time. Walking instructions specify the direction to walk, along with the distance and time in meters and minutes, respectively. However, this walking navigation feature is still under development. The walking instructions provided by Google Maps are currently inadequate for visually impaired users, often offering vague directions like "Walk south-east for 10 meters" which can be confusing even for individuals without visual disabilities, thus making testing more difficult.

E. User Study

User studies were conducted to evaluate the *Baymax* app, involving 14 participants.

The aim of the user studies was to gather both qualitative and quantitative insights into the app's performance, addressing the following research questions:

- Does the app provide useful information about the surroundings for a visually impaired user?
- Does the app deliver accurate auditory and visual cues for the obstacle avoidance navigation mode?
- Is the app easy and intuitive to use, considering the user interface and spatial audio features?

Quantitative metrics were employed to assess the response time of the scene description, as well as the accuracy of the obstacles' positions and the safe navigation path generated by the obstacle avoidance mode. Qualitative feedback was collected to evaluate user experience and identify areas for improvement to determine how the app's features can be enhanced.

E.1. Experiment Settings

All user studies were performed indoors within ETH buildings. Some tests were conducted while connected to a router for a fast and reliable Wi-Fi connection. Additionally we, the developers of *Baymax*, also tested the app outdoors, and more details on these results can be found in Subsection E.2.5.

Participants had no prior experience with the HoloLens. Although we were unable to test the app with visually impaired users, we simulated visual impairment by asking participants to perform tasks with their eyes closed during the obstacle avoidance mode.

For the evaluation, participants were asked to perform different tasks for each feature of the app and then complete a survey to help us answer our research questions.

E.1.1 Tasks

Each participant was first provided with a tutorial on how to use the available voice commands, which are also outlined on the app's welcome page.

For the scene description feature, participants were instructed to make the following queries to *Baymax*:

- Ask what is in front of them.
- Read text from signs, such as auditorium names with floors, ETH store opening hours, restroom signs, cafeteria names and vending machines.
- Request navigation instructions, such as "How do I get out of the building or this hallway?"

For the obstacle avoidance mode, participants first walked through a hallway with obstacles such as chairs and walls while keeping their eyes open, allowing them to see the visual cues on the user interface. Then, they were asked to walk with their eyes closed and follow the spatial audio cues. During this test, we moved dynamically in front of them to simulate obstacles and evaluate their ability to navigate based solely on auditory feedback.

For the Google Maps-based navigation feature, users requested transit directions to a destination address, along with walking directions to the first tram stop along the route.

E.1.2 User survey

We asked 16 questions in a Google Form to the users about their experience. While some questions were omitted for brevity, the key questions included are as follows:

1. Did the scene descriptions provide you with useful information about your surroundings? (rate from 1 to 5; 1 is very bad, 5 is very good)
2. Is there any other kind of information you would have liked to get from the scene description?
3. Did the obstacle avoidance mode help you feel more confident navigating the environment? (rate from 1 to 5)
4. Were the obstacle warnings timely and accurate? (rate from 1 to 5)
5. Was the spatial audio helpful in avoiding obstacles? (rate from 1 to 5)
6. Do you think other kinds of sound effects or visual enhancements would be more helpful?
7. Is there any other kind of information you would have liked to get from Google Maps?
8. How easy was it to interact with the app using the available commands? (rate from 1 to 5)
9. Are there any other features you would have liked, to feel more comfortable while navigating blindly?
10. If you were visually impaired, would you use the app? (rate from 1 to 5)

We then gathered the ratings and feedback for evaluation and future improvements.

E.2. Results, Limitations and Future Improvements

For each feature, we evaluated both its accuracy and usefulness, gaining valuable insights into its limitations and identifying potential solutions.

E.2.1 Scene Description

The feature received positive feedback from the 12 users who tested it and was regarded as a helpful tool. 10 out of 12 participants found the length of the chatbot's scene descriptions appropriate. In terms of content, users expressed high satisfaction, giving it an average rating of 4.25 out of 5. The app successfully performed all tasks outlined in the subsection E.1.1. In some cases, it even offered navigation instructions to reach the end of the hallway.

Across 20 tests conducted by the *Baymax* developers, while connected to a router, the scene description demonstrated a mean response time of 5.5 seconds, with a notable standard deviation of 5.1 seconds. However during user studies, the Gemini API response showed delays of $\sim$1 minute. This difference in response time could be attributed to longer and more complex scene descriptions, as well as the presence of numerous Wi-Fi networks in close proximity, causing interference and reducing the reliability of the connection in the test location. A stable connection and continuous token streaming could mitigate these effects. Furthermore, some users suggested putting more emphasis on moving objects and providing the name of the current location (e.g. HG of ETH Zurich) in the scene description. While the first suggestion could be directly incorporated into the prompt passed to the chatbot, the second request can be solved by calling the Google Maps API with the user's current GPS coordinates.

E.2.2 Obstacle avoidance

This user mode received mixed ratings and feedback from the 14 participants who tested it. While some aspects were positively reviewed, others highlighted areas for improvement, particularly in the auditory cues and the behavior of the ball guiding users along the safe path. The ratings collected for the obstacle avoidance mode, regarding its accuracy and usability, are summarized in Figure 7.

Visual cues. The cubes representing obstacles were generally accurately positioned, achieving an average user rating of 3.71 out of 5. Most obstacles are successfully detected, with the exception of glass walls and stairs. While the guiding ball often avoided obstacles as intended, it could sometimes change place abruptly or appear outside the user's field of view. The ball may direct users towards obstacles located more than 2 meters ahead of them, as it updates to point toward the free path 1.5 meters ahead of the user. Once the user nears the obstacle, the ball adjusts

Figure 7. **User studies feedback.** Feedback for the obstacle avoidance feature, evaluating the accuracy and the ease of use. The x-axis represents the score, with 1 indicating a very poor rating and 5 indicating an excellent rating. The y-axis shows the count of votes corresponding to each score.

direction to continue guiding them safely. Future implementations could extend the safe path calculation to cover a greater distance. Planning algorithms such as Rapidly Exploring Random Trees (RRT) [16] or Dijkstra's Algorithm [5] could help users anticipate distant obstacles earlier in their journey. In addition, the guiding ball can be replaced with a more intuitive shape, such as an arrow, to enhance the user interface.

Auditory cues. The feedback on spatial audio varies across users, with an average rating of 3 out of 5 for ease of use. Many users reported difficulty in determining the obstacle-free path based solely on stereo audio cues when their eyes were closed. A consistent observation was that the HoloLens' spatial audio cues for right and left directions were generally easy to follow, whereas interpreting

forward and backward directions was more challenging, as the sound seemed to come from both sides of the HoloLens simultaneously. While we implemented the audio to originate from the guidance sphere's 3D spatial coordinates in our Unity application, the HoloLens speaker system faces hardware limitations in delivering precise spatial audio perception. Future enhancements to convey more precise directional information may include playing distinct sounds for each direction, providing verbal cues, or integrating the application with haptic feedback systems.

Alternative implementation. User feedback indicated that the continuous audio cues might overwhelm the user, particularly those with visual impairments who often have heightened auditory sensitivity. A more user friendly navigation mode could warn users of immediate obstacles through sound cues, instead of playing constant audio. The app could also verbally inform the user when there are no obstacles within a certain radius. As obstacles approach, the sound could increase in intensity or change frequency, depending on the proximity of the obstacle.

E.2.3 Google Maps based navigation

As mentioned in Subsection D.3, the walking instruction feature is difficult to test due to the vagueness of some directions. Consequently, we were unable to determine whether the walking instructions provided by the Google Maps API would lead users to their desired destinations. However, users were able to request transit directions to specific destinations, such as "ETH Zurich," "HB" or "Bellevue," and found the information useful for transportation.

Certain street names and locations in German could not be converted from speech to text when using the English-speaking Google Maps API. This issue could potentially be solved by setting the API's language argument to German.

To improve the navigation feature's accessibility for blind users, a directions system such as Lazarillo [13] specifically designed to compute safe routes for blind people, could be integrated into the application.

E.2.4 User Interface

The user interface was generally regarded as easy to use, earning an average rating of 3.93 out of 5 for usability by 14 users.

There were occasional issues with speech recognition in noisy environments, where commands such as "start" or "stop" were not recognized. Additionally, a poor Wi-Fi connection can lead to prolonged loading times for the scene description, leaving the user unsure about how to interact with the interface. This could be addressed by implementing a timeout that returns the user to the welcome page with an audio prompt, or by ensuring a stable Wi-Fi connection.

Additionally, the task of starting the app and server on a PC—including configuring the PC's IP address on the HoloLens app—is not feasible for blind users. The app can be launched via Microsoft's Cortana, but external assistance may still be required to initialize the server on the PC. Future work could explore implementing on-device processing directly on the HoloLens, thus eliminating the need for a separate back-end server.

E.2.5 Outdoors testing

Testing the app outdoors revealed several operational constraints. The HoloLens's camera demonstrated significant limitations in varying lighting conditions, particularly during low-light situations (e.g. after sunset), which severely impacted the system's object detection capabilities. The HoloLens's depth sensor, like most commercial depth sensors, faces well-documented challenges with long-distance measurements and strong sunlight conditions [12]. The device's narrow field of view further restricted its ability to capture comprehensive environmental information. Network connectivity posed another challenge, as the system's reliance on WiFi resulted in occasional latency issues and connection drops outdoors. Additionally, we observed performance degradation in cold weather conditions, with the device experiencing increased lag and reduced responsiveness.

Future iterations of the system would benefit from implementing robust monocular depth estimation algorithms such as UniDepth [15] alongside sensor fusion techniques to combine the strengths of both methods, particularly in challenging lighting conditions.

F. Conclusion

After testing our app, we asked participants whether they would use *Baymax* if they had visual impairments, and they gave an average rating of 3.79 out of 5.0. Since the participants in our user study did not have visual impairments, we cannot say whether a blind user would choose to navigate using solely our app. However, based on preliminary feedback, it is reasonable to suggest that our proof-of-concept app can help users gain awareness of unfamiliar surroundings and contribute to a sense of safety, especially when used alongside other assistive tools such as a cane or haptic devices. We identified several areas for improvement in our implementation and propose solutions in Section E.2 to better accommodate visually impaired users. One of the most significant takeaways from the user studies is to optimize the safe-path navigation algorithm and modify the audio cues; future iterations will explore an event-triggered approach where audio feedback is provided only for imminent collision risks, potentially reducing mental fatigue while maintaining safety.

G. Acknowledgements

We thank our supervisors Zuria Bauer, Mihai Dusmanu and Bessie Dominguez for their valuable guidance and insights throughout the development of our application.

References

[1] Jinqiang Bai, Shiguo Lian, Zhaoxiang Liu, Kai Wang, and Dijun Liu. Smart guiding glasses for visually impaired people in indoor environment. *IEEE Transactions on Consumer Electronics*, 63(3):258–266, 2017. 1

[2] Robert C Bolles and Martin A Fischler. A ransac-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI*, volume 1981, pages 637–643, 1981. 4

[3] Lucie Brunet, Françoise Darses, and Malika Auvray. Strategies and needs of blind pedestrians during urban navigation. *Le Travail Humain*, 81(2):141–171, 2018. 1, 2

[4] Juan C. Dibene and Enrique Dunn. Hololens 2 sensor streaming, 2022. 3

[5] E. W. Dijkstra. *A Note on Two Problems in Connexion with Graphs*, page 287–290. Association for Computing Machinery, New York, NY, USA, 1 edition, 2022. 6

[6] Dot Lumen. Dot lumen: Ai-powered assistive device for the visually impaired. https://www.dotlumen.com/technology, 2024. Accessed: 2024-12-31. 1

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. 4

[8] Google. Generative ai python library. https://github.com/google-gemini/generative-ai-python, 2024. Accessed: 2024-12-24. 3

[9] Google. Google maps services python client. https://github.com/googlemaps/google-maps-services-python, 2024. Accessed: 2024-12-24. 2, 4

[10] GPS2IP Lite. Gps2ip lite. https://apps.apple.com/us/app/gps2ip-lite/id1562823492, 2024. Accessed: 2024-12-24. 4

[11] A Haileamlak. The burden of visual impairment and efforts to curve it down. *Ethiopian Journal of Health Sciences*, 32(5):874, 2022. 1

[12] Achuta Kadambi, Ayush Bhandari, and Ramesh Raskar. *3D Depth Cameras in Vision: Benefits and Limitations of the Hardware*, pages 3–26. 07 2014. 7

[13] Lazarillo. Lazarillo: Navigation app for the blind. https://lazarillo.app, 2024. Accessed: 2024-12-24. 7

[14] Microsoft. *Azure Speech Services*, 2024. https://azure.microsoft.com/en-us/products/cognitive-services/speech-to-text/. Accessed: 2024-12-24. 3

[15] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation, 2024. 3, 7

[16] Rodriguez, Xinyu Tang, Jyh-Ming Lien, and N.M. Amato. An obstacle-based rapidly-exploring random tree. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 895–900, 2006. 6

[17] Debjyoti Sinha and Mohamed El-Sharkawy. Thin mobilenet: An enhanced mobilenet architecture. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0280–0285, 2019. 3