A Vision Based System for Assisting Blind People at Indoor and Outdoor Exploration

Raluca Didona Brehar[a] and Sand Elena-Andreea
Computer Science Department, Technical University of Cluj-Napoca, Romania
-

Keywords: Object Detection, Smart Navigation, Computer Vision, Embedded Systems.

Abstract: An approach that combines hardware processing facilities with artificial intelligence and computer vision is proposed in this paper resulting in a prototype for assisting blind people at indoor and outdoor exploration in terms of object detection and recognition, color recognition, obstacle avoidance and smart navigation. Several test scenarios have been experimented and prove the efficiency of the proposed approach. The targeted test scenarios comprise shopping assistance, obstacle avoidance and directions for safe navigation.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

According to the most recent studies (Pesudovs, K., Lansingh, V.C., Kempen, J.H., 2024), visually impaired people represent a significant portion of the population, specifically, 295 million individuals are affected by this condition, of which 17 million are blind.

Among the problems faced by visually impaired people, the most important are: finding an object, perception of the surrounding environment, identifying a color, using public transportation, navigation in the external environment and avoiding obstacles. The approach proposed in this paper is engineered to help blind people in various situations encountered in everyday life, giving them the opportunity to carry out their activities without having to ask for the help of other people. To help visually impaired individuals achieve a higher degree of independence, a system that perceives and processes the surrounding environment in real-time and has the capability to accurately and precisely recognize and locate existing objects or potential obstacles is needed. To ensure this, the proposed solution combines embedded components with state of the art computer vision models. The proposed approach develops five main modes of operation all having an interactive and friendly mode of use: (i) smart navigation; (ii) object detection and recognition; (iii) color identification to assist in outfit selection; (iv) groceries detection and recognition; (v) obstacle avoidance. Through voice commands, the user can specify their needs, and the necessary instructions are provided via audio output.

The main contributions of the proposed solution reside in:

- Development of an embedded hardware system with strong processing capabilities for computer vision assistance of blind people.
- Design and implementation of object detection, recognition, smart navigation based on state of the art deep learning models.
- Integration of hardware and software solutions for typical test scenarios in blind people assistance.
- Real time processing.

2 RELATED WORK

This section provides an overview of existing research and developments related to systems tailored for visually impaired users that focus on object detection, groceries detection and obstacle avoidance.

In the work presented by (Kabir et al., 2023), the authors compare the You Only Look Once (YOLO) models (Redmon et al., 2016), versions 4 and 7, and the SSD Mobilenet-V2 model (Sandler et al., 2019). After the tests were conducted, they claim that YOLO provides better accuracy, while SSD provides fast and efficient detection on embedded devices.

The approach proposed by (Kumar and Jain, 2021) describe a system that provides travel instructions for blind people using the YOLOv3 object de-

[a] https://orcid.org/0000-0003-0978-7826

Brehar, R. and Elena-Andreea, S.
A Vision Based System for Assisting Blind People at Indoor and Outdoor Exploration.
DOI: 10.5220/0012973100003822
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2024) - Volume 2*, pages 54-65
ISBN: 978-989-758-717-7; ISSN: 2184-2809
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

tection model and ultrasonic sensors to signal obstacles close to the user. The authors modified the model YOLOv3 to be integrated into a portable system and trained it using a dataset containing images classified into about 25 different categories such as straight roads, intersections, pedestrians or vehicles. The system provides real-time audio feedback through headphones. The authors claim that the system offers 96.14% accuracy with the help of sensors ultrasonics, while the modified YOLOv3 model provides 81% accuracy.

In (Baskar et al., 2021) the implementation of a system that helps blind people to avoid obstacles that may appear in the indoor environment is described. To develop the system, they use a Raspberry Pi 3 camera, an infrared sensor and ultrasonic sensors to detect and identify obstacles in real time. The authors claim that the infrared sensor detects objects in movement, the ultrasonic sensor detects static objects, and the camera is used to detect predefined objects.

The article (Rodríguez et al., 2007) presents an obstacle avoidance system for assisting the visually impaired people, using a stereo camera that generates a disparity map that allows obstacle detection in different scenarios. Once the depth map from the stereo camera is obtained, the algorithm creates a grid using polar coordinates, the size of which covers a distance of four and a half meters. The grid is divided into three zones to represent the distance to obstacles (4.5m, 3m, and 1.5m), and each zone is divided into 30° portions. Next, the algorithm determines the pixels that belong to the ground, using the RANSAC algorithm, in order not to determine the area of interest for the user. Once the pixels that belong to the ground are determined, the rest are analyzed to find if they intersect the grid, that is, if they are part of a possible obstacle. Finally, the algorithm alerts the user to the presence of an obstacle by sending a beeping sound into the headphones, controlling the frequency of the sound based on the distance to the obstacle. The test results demonstrate the performance of the system, both in the outdoor and indoor environments.

The paper (K et al., 2014), presents an application that helps blind people to do their shopping in a supermarket. The system uses a camera to capture images from the supermarket which are then processed to extract the text in order to identify the products. The system also extracts the price of the products, and at the user's command, adds that price to the purchase calculation. This bill amount is then sent as voice output to the user.

A system to help visually impaired individuals shop independently in supermarkets is described by (Devipriya et al., 2018). The system addresses three primary concerns: locating products, identifying products, and automating the billing process. Product identifier helps locate the desired product using RFID technology. A buzzer rings when the product is found. Smart glove assists in identifying the product by providing audio details through a headset when the product is scanned and smart trolley automates the billing process. When a product is placed in the trolley, it is automatically scanned and added to the bill. The total bill is displayed on an LCD screen and can be updated in real-time. The system was tested with three products: Masala, Milk, and Apples. The Product Identifier successfully detected and matched products with voice inputs. The Smart Glove provided audio feedback about the products, and the Smart Trolley automated the billing process, displaying and updating the total bill.

3 PROPOSED APPROACH

The original aspect of the proposed approach consists in the development of an embedded system based on state of the art computer vision perception models meant to help visually impaired individuals to improve their every day life activities.

The software architecture of the system is described in the figure 1. To ensure the performance of the system, it was decided to implement it on NVIDIA Jetson Nano, a mini-computer with a specialized hardware architecture for the development of artificial intelligence and deep learning applications in embedded devices. To analyze the environment, two cameras were connected: Raspberry Pi Module 2 and Intel Realsense D435 that communicate with the board through the UART and CSI interfaces. 1.

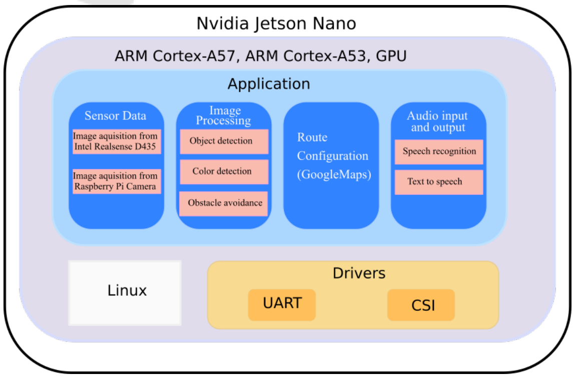


Figure 1: Software architecture.

The current solution consists of 5 main modes of operation, which presents an interactive and friendly mode of use. Using voice commands recognized with the help of the Speech Recognition library, the user can specify their needs, and the necessary instructions are provided via audio output using the Text-To-Speech library.

The module for navigation with Google Maps contains specific functions to provide the user with a summary of the trip based on the directions received from Google Maps. Depending on the distance to be covered, the user can choose if they want directions involving public transport, or if they want to walk the route without them.

The object detection module is based on the SSD MobileNet-v2 (Sandler et al., 2019) detection model which is used by the component which identifies a specific object and the component that identifies all surrounding objects. An algorithm that controls the frequency of audio transmission of information was also implemented.

The color detection module contains the specific functionality for identifying a specific color or recognizing the predominant color in the frame. This functionality can be used for clothes shopping or other color identification scenarios.

The store product detection module is based on the SSD MobileNet-v2(Sandler et al., 2019) model, retrained on a set of store product images. This mode of operation notifies the user upon the section of the store in which they are located.

The outdoor navigation module contains the functions for implementing the obstacle avoidance algorithm and also a function that locates a person and provides the user with information about that person's position.

3.1 Hardware System

The hardware architecture of the system is described in the Figure 2. To ensure a high system performance, it was implemented on the NVIDIA Jetson Nano, a mini-computer that supports high-resolution sensors and allows multiple neural networks to run in parallel for applications such as image classification, object detection, segmentation, and speech processing. To analyze the environment, the following two cameras were connected: Raspberry Pi Module 2 and Intel Realsense D435. The Raspberry Pi Camera Module 2 is equipped with an 8-megapixel Sony IMX218 sensor, allowing the capture of high-resolution images, which is crucial for the functionality that provides information about the color of objects. The Intel RealSense D435 stereo camera constructs a depth map, from which information about the distances of obstacles in the scene can be extracted. To receive voice commands from the user and provide audio output, a headset with a microphone were connected.

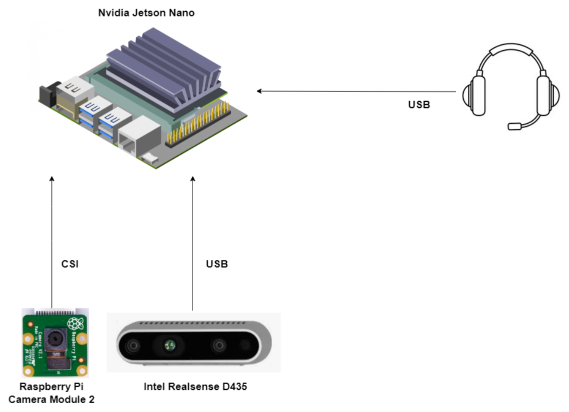


Figure 2: Hardware architecture.

3.2 Smart Navigation

The algorithm for smart navigation uses specific functions from Google Maps that provide the summary of a route. These functions need the start address and the destination address. Thus, to obtain them, the user is interrogated using the function speech_to_text(), and his answers are stored in the variables origin_address and destination_address. Because it is assumed that a short walking route is desired, the directions() function from Google Maps is called with the 'walking' parameter which returns the details of the route, and then the total distance in kilometers is calculated. If this distance is greater than 3 kilometers, the user is asked if they want to use public transport. If the answer is yes, the directions() function is called with the 'transit' parameter, thus obtaining the summary of the route that contains public transportation for the trip. After receiving the instructions from the Google Maps API, the HTML elements are stripped to get the instruction text in plain format. Depending on how the user wants to travel the route, there are two functions that generate the detailed descriptions of the directions: generate_walking_directions() and generate_public_transport_directions(). Finally, through the function text_to_speech (), the corresponding route summary is presented to the user.

3.3 Object Detection

The object detection module can identify an object specified by the user or it can identify all surrounding objects. For object detection, two models were compared in order to choose the one that satisfies both the accuracy and processing speed requirements. The two models are YOLO v3 (Redmon et al., 2016) and SSD Mobilenet-v2 (Sandler et al., 2019).

Because real-time detection is important for system development, the speed of the two models was

Table 1: Processing time.

Model	Frames processed per second
SSD MobileNet-v2	17.3
YOLOv3	0.2

compared and the results can be seen in the table 1.

Since model accuracy is also important for system development, the detection confidence levels of the two models were also compared. The results can be seen in the table 2.

Table 2: Comparison of Detection Accuracy for SSD Mobilenet-v2 and YOLOv3.

Object	SSD Mobilenet-v2	YOLOv3
Dinning table	0.94	0.99
Laptop	1.00	0.99
Chair	0.99	0.99
Potted plant	0.96	0.98
Bottle	0.97	0.99
Remote	0.90	0.84
Handbag	0.96	0.90
Backpack	0.94	-
Cup	0.90	0.97
Bowl	0.99	0.99
Book	0.93	0.98
Cell phone	0.89	0.99

Based on the results, since there is only a small difference in the confidence level of detections between the two models, the decision was made to choose the SSD MobileNet-v2 model for the system's development, justified by its processing speed.

3.4 Color Recognition

This mode of operation provides two functionalities: finding a color specified by users and identifying the predominant color in front of the user. A list of 18 colors, considered to be more common, and their RGB combination. Among these colors are red, blue, green, pink, yellow and many others.

3.4.1 Find Specific Color

In order to find a specific color a conversion from RGB to HSV is made and the lower and upper limit is HSV space are determined for each color. The range of each color was calibrated and established by testing on a set of images with different colors. A dictionary containing specific Saturation and Value adjustments based on the color name was also used. Colors such as ”white”, ”silver”, ”gray”, ”black” have special adjustments due to their unique characteristics (for example, white has very high saturation and brightness). Also, the image on which color recognition is operated is converted from the BGR color space, to HSV. Next, a binary mask is created for all pixels which are in the specific color range of HSV values specified by the previously determined lower and upper bounds.

After obtaining this mask, to determine the position where the color was in the image, the center of mass of the region is calculated, determining its coordinates (centroid_x and centroid_y). Next, the position of the ”centroid_x” coordinate is checked in relation to the width of the image, divided into three equal sections, and based on this position, the function returns one of the following values: ”center”, ”left” or ”right”.

3.4.2 Find Dominant Color

The specific function for identifying the predominant color in the frame, iterates over all colors combinations, and for each color determines the lower and upper limit in the HSV space and creates a color limits dictionary. Then for each color we compute the area covered by the color in the given image (which pixels are in the upper and lower range of the color). The color with the largest area is considered the predominant color.

3.5 Groceries Detection

To implement this functionality, the SSD Mobilenet-v2 model was retrained for groceries detection on an extension of the Freiburg Groceries Dataset (Jund et al., 2016). The initial dataset contains a total of 4947 images for 25 common food product classes found in supermarkets. Each class has at least 97 images, and many of these contain multiple object instances. The dataset extension (Aleksandrov, 2020) that was used for retraining extends the original dataset by adding labeled bounding boxes for each of the available images. Annotations are in PascalVOC format.

3.6 Exploration System

In order to implement the obstacle avoidance algorithm three perception methods have been employed and compared: (i) uses three HC-SR04 ultrasonic sensors (ii) uses the monocular vision camera Raspberry Pi v2 and (iii) uses the Intel® RealSense ™ D435 depth camera.

3.6.1 Using the HC-SR04 Ultrasonic Sensor

To implement the obstacle avoidance algorithm, we used three ultrasonic sensors, designed to detect obstacles around the user. The algorithm checks the data from the middle sensor, which is responsible for detecting obstacles in the user's path, and if it detects an object closer than 150 cm, it is considered an obstacle. To avoid the obstacle, the data from the left and right sensors is checked. If they detect objects further than 150cm, it means that the obstacle in the user's path can be avoided by the left or right side. Thus, the values from the two sensors are compared, and the sensor that detects a more distant obstacle highlights a clear path that can be used to avoid the obstacle.

After testing the algorithm, it was observed that a disadvantage of this sensor is its difficulty in detecting small objects that reflect an insufficient ultrasonic signal for detection or objects that absorb acoustic signals, such as those made from soft materials. Consequently, the farther an object is and the smaller it appears in the sensor's field of view, the harder it is to detect. Also, if an object's position is oriented such that the ultrasonic signal is deflected rather than reflected back to the sensor, the calculated distance will be incorrect. Another aspect to consider is that the positioning of this array of sensors on the user's body is very important because it is limited by the size of the field of view, which is about 30°. If it is positioned around the legs, it only detects objects on the ground, if it is positioned around the abdomen, it only detects large objects in the user's path, and if it is positioned around the head, it only detects obstacles from above.

3.6.2 Using Monocular Vision

A second approach employed the usage of a monocular vision Raspberry Pi v2 camera. To estimate the distance to the objects in the scene, required for the obstacle avoidance algorithm, four pre-trained models that generate the depth map of the captured frames from monocular images were tested. In order to choose a model to implement the algorithm for obstacle avoidance, both the accuracy of the four models and the processing speed were compared, evaluating their results on the same set of images.

These four models are:

- Depth Net (CS Kumar et al., 2018)
- Fast Depth (Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne, 2019)
- MiDaS (Ranftl et al., 2022), (Ranftl et al., 2021)
- Depth Anything (Yang et al., 2024)

Part of the results generated by the four models after testing them on a set of images can be seen in the Table 3 and Table 4. Map colors allow interpretation of depth differences. Objects that are closer have a lighter color and those that are further away have a darker color.

Table 3: Depth maps generated by Depth Net and Fast Depth.

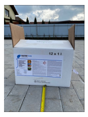
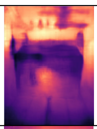
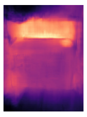
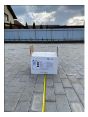
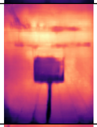
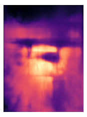
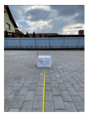
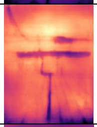
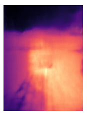
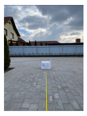
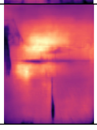
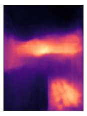
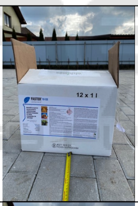
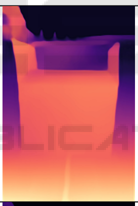
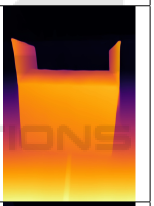
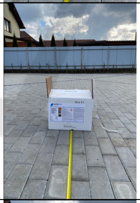
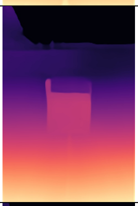
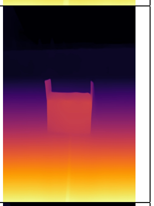
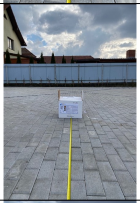
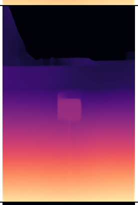
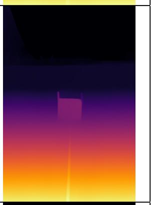
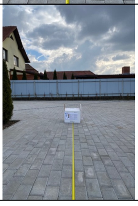
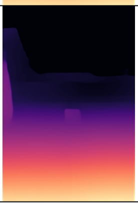
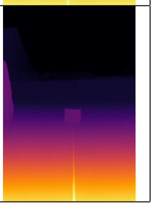
Distance	Image	DepthNet	FastDepth
50 cm			
110 cm			
210 cm			
310 cm			

Table 4: Depth maps generated by MiDaS and Depth Anything.

Dist.	RGB Img.	MiDaS	Depth A.
50			
110			
210			
310			

As can be seen, the Depth Net and Fast Depth models fail to generate a detailed depth map that dif-

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

ferentiates the objects in the scene. The depth of the objects cannot be interpreted from the results of the two models. In contrast, the MiDaS and Depth Anything models achieve some promising results, generating dense and detailed maps that differentiate objects in the scene based on their depth.

However, to use one of these models in the development of the obstacle avoidance algorithm for blind people, it is necessary to obtain the real-time distance estimation data. In this sense, we compared the processing speed of these four models. We measured the time required to process a single image for each model and the reuslts can be seen in table 5. As

Table 5: Processing time per frame.

Model	Time (min)
Depth Net	0.55
Fast Depth	1.6
MiDaS	3.33
Depth Anything	11

can be observed, the Depth Net model achieves the best processing speed, but due to the unfavorable results obtained in generating depth maps, it cannot be used within the algorithm. Although the MiDaS and Depth Anything models produced promising results for depth map generation, their processing speed is too slow to provide real-time results necessary for algorithm development.

Therefore, none of the four tested models succeed in generating a detailed map that contains the necessary information while simultaneously providing data in real-time.

3.6.3 Using Stereo-Vision

In the article (Saputra et al., 2014), the authors describe an algorithm that uses a self-adaptive threshold for the separation of objects in the scene, using a depth camera. For the development of the obstacle avoidance algorithm within this system, the algorithm described by (Saputra et al., 2014) was implemented, adapting it for the system components. The steps of the algorithm are represented in Figure 3.

The first processing step consists in acquiring the depth map in grayscale format from the camera sensor. This map, according to the camera specifications, can contain values between 280mm and 10000mm, but the relevant values for this algorithm can be reduced to the range [300,4000]. The image is divided into 3 equal areas (left, center, right), these representing the possible directions of movement for the user as shown in 4. The following steps will be carried out for each area separately, analyzing and detecting

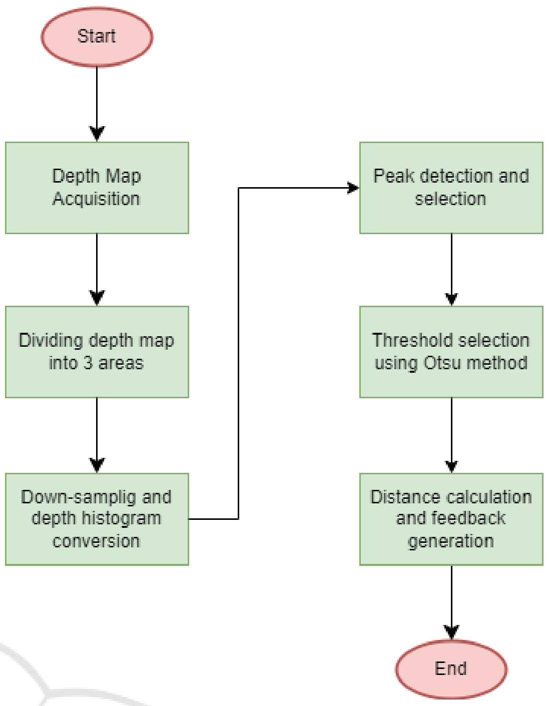

Start
Depth Map Acquisition
Dividing depth map into 3 areas
Down-samplig and depth histogram conversion
Peak detection and selection
Threshold selection using Otsu method
Distance calculation and feedback generation
End

Figure 3: The obstacle avoidance algorithm.

obstacles that may appear in the user's path.

To speed up the calculation and remove unnecessary information, down sampling is performed, which reduces the size of the depth map. For each group of 4 pixels, only one is considered. Next, the depth histogram is created for each area of the image. Thus, the values in the interval [0.4000] will be divided into 100 groups, the interval between each being 40 mm, and for each group the number of belonging pixels is calculated.

Peaks (local maxima) are identified using the contrast function. The contrast function compares the value of each point in the histogram with the values of neighboring points, a sudden change in depth being considered the edge of an object. The contrast of a point i in the histogram is calculated by adding the values around point i, in a range defined by the parameter n, and subtracting the sum of the values from a larger range as shown in equation 1.

$$\text{contrast}(i,n) = \sum_{k=i-n}^{i+n} p_k - \sum_{k=i-2n}^{i-n-1} p_k - \sum_{k=i+n+1}^{i+2n} p_k \quad (1)$$

After calculating the contrast value around each point in the histogram, local maxima are identified based on the following criteria:

- The minimum contrast value should be 50, suggesting that the point is significantly distinguishable from its neighborhood

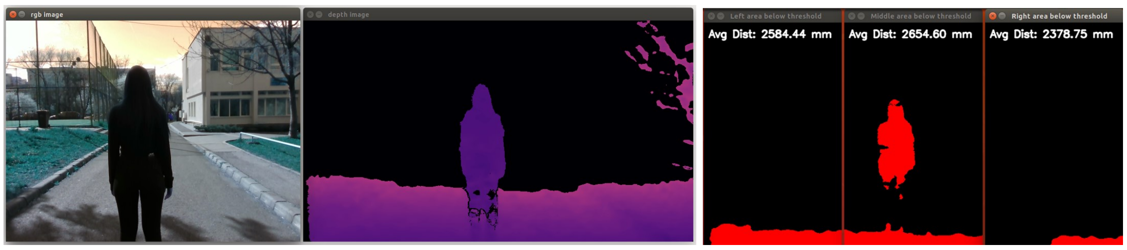


Figure 4: Input Image (left), Depth (middle) and Filtered depth and image division in three areas (left, center, right).

- The minimum distance between two successive peaks must be at least 4 units in order to distinguish it from a neighboring obstacle.

After identifying all local maxima by applying the above criteria, two maxima are selected that have the closest values to the camera, being considered the closest obstacles.

After identifying the two local maxima, the Otsu method is used which generates a threshold to separate the nearest obstacle from other obstacles or from the image background. The Otsu method looks at all possible threshold values and calculates the between-class variance for each. The classes are defined by pixels that have values below the threshold, assumed to be part of the obstacle, and those with values above the threshold, considered to be part of the background or other obstacle. The inter-class variance measures how well a certain threshold separates the two groups of pixels. The value that maximizes the variance between classes is chosen as the optimal threshold.

After determining the optimal threshold for each area, the distance to the nearest obstacle is calculated as the average of the pixels with values lower than the determined threshold. After determining the distance to the nearest obstacle for each area of the image, audio feedback is provided to avoid it.

4 EVALUATION AND RESULTS

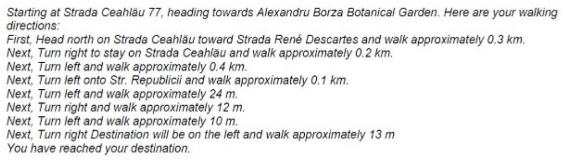
The modularity of the system facilitated its testing, thus, each mode of operation was tested individually, tracking the accuracy of results and processing speed.

4.1 Smart Navigation

This mode of operation was tested in the following two cases:

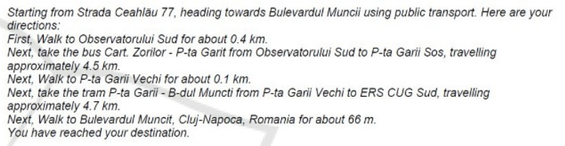
1. the distance of the route is less than 3 km
2. the distance of the route is greater than 3 km

The result obtained in the case where the distance is less than 3 km can be seen in the figure 5, and the one in which the distance is greater than 3 km can be seen in figure 6.



Starting at Strada Ceahlău 77, heading towards Alexandru Borza Botanical Garden. Here are your walking directions:
First, Head north on Strada Ceahlău toward Strada René Descartes and walk approximately 0.3 km.
Next, Turn right to stay on Strada Ceahlău and walk approximately 0.2 km.
Next, Turn left and walk approximately 0.4 km.
Next, Turn left onto Str. Republicii and walk approximately 0.1 km.
Next, Turn left and walk approximately 24 m.
Next, Turn right and walk approximately 12 m.
Next, Turn left and walk approximately 10 m.
Next, Turn right Destination will be on the left and walk approximately 13 m
You have reached your destination.

Figure 5: Summary of the route shorter than 3 km.



Starting from Strada Ceahlău 77, heading towards Bulevardul Muncii using public transport. Here are your directions:
First, Walk to Observatorului Sud for about 0.4 km.
Next, take the bus Cart. Zorilor - P-ta Garit from Observatorului Sud to P-ta Garii Sos, travelling approximately 4.5 km.
Next, Walk to P-ta Garii Vechi for about 0.1 km.
Next, take the tram P-ta Garii - B-dul Muncti from P-ta Garii Vechi to ERS CUG Sud, travelling approximately 4.7 km.
Next, Walk to Bulevardul Muncit, Cluj-Napoca, Romania for about 66 m.
You have reached your destination.

Figure 6: Summary of the route greater than 3 km.

4.2 Object Detection

A series of tests were conducted to assess the accuracy of results in detecting individual objects.

A set of images was prepared , where 10 objects were photographed from various angles and the model was tested using this set. The results showed that the SSD model's outcomes are influenced by the objects' positions, with their shapes changing as the photographing angle changes. The best and worst results obtained in detecting objects using the SSD model are highlighted in Table 6. These results are influenced by the position and orientation of the objects in the image.

Since providing real-time results is one of the goals of this system, multiple tests were performed to verify this aspect. The Mobilenet-v2 SSD model manages to process 17 frames per second, and in the table 7, maximum time required to process a frame and the minimum time can be observed. The variation in processing time is influenced by the number of objects detected in a frame.

4.3 Color Detection

In order to determine the performance of the system in color detection, several tests were carried out to high-

Table 6: Accuracy variation.

Object	Highest Accuracy	Lowest Accuracy
Dinning table	0.94	0.51
Laptop	1.00	0.55
Chair	0.99	0.50
Potted plant	0.96	0.59
Bottle	0.99	0.51
Remote	0.90	0.53
Handbag	0.96	0.59
Backpack	0.94	0.52
Cup	0.99	0.50
Bowl	0.99	0.50
Book	0.93	0.52
Cell phone	0.96	0.51

Table 7: Time variation for processing a frame.

Information	Value (s)
Best time	0.083
Worst time	0.357

light the accuracy of the recognition, but also the time required for the recognition.

Thus, a set of 50 images with clothing items photographed using Raspberry Pi V2 camera was prepared. The tests were conducted to determine how much variations in light affect color recognition. Thus, the image set contains colors photographed under different lighting conditions. The results of this test are presented in table 8. It can be seen how, the color of the same article of clothing is recognized differently when the brightness differs.

At the same time, the accuracy of the system to determine the position of a certain color in the image was also tested. Thus, a set of images was prepared in which several clothing items of different colors are present and the position of a specific color in each image was determined. The results are presented in table 9.

Also, because providing a real-time response is a goal of this system, the time it takes the system to identify the predominant color in a frame was analyzed. It was observed that the speed is influenced by the color in the frame, as the algorithm scans the entire color dictionary to check the extent to which a color is present in the image. The best and worst times obtained by the algorithm are shown in Table 10.

4.4 Groceries Detection

To analyze the results obtained by the retrained SSD model to recognize groceries, numerous tests were performed. The accuracy of the results and factors influencing the results were analyzed.

Table 8: Colors variation.

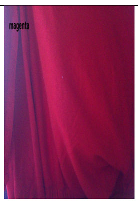
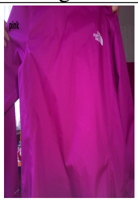
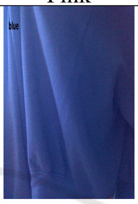
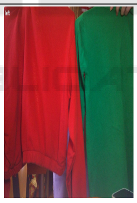
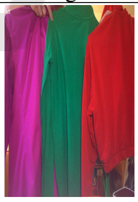
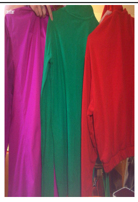
Color	Result 1	Result 2	Result 3
Red	Red	 Magenta	Red
Pink	Pink	 Pink	Magneta
Blue	 Blue	Silver	Blue

Table 9: Color localization.

Color	Test 1	Test 2	Test 3
Red	Right	 Left	Center
Pink	 Left	Right	Center
Green	 Center	Right	Left

Thus, a set of 50 images was used for testing the model's ability to correctly recognize the categories. The set depicts items from 20 categories photographed on store shelves. The image set contains

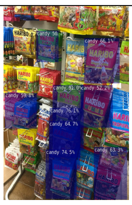
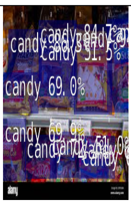
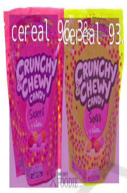
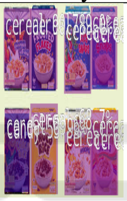
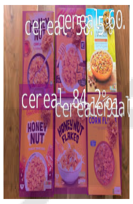
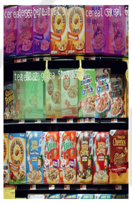
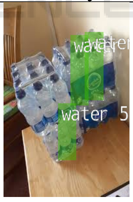
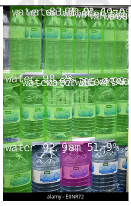
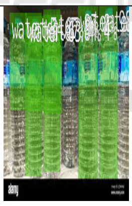
SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Table 10: Time variation for recognizing the dominant color.

Information	Value (s)
Best time	0.11
Worst time	0.28

products from the same category, which have different packaging, photographed from various distances to observe how product size and packaging influence detection. Some of the obtained results are presented in the table 11.

Table 11: Groceries detection.

Class	Result 1	Result 2	Result 3
Candy	 Candy	 Candy	 Cereal
Cereal	 Cereal / Candy	 Cereal	 Cereal / Tea
Water	 Water	 Water	 Water

The model has trouble differentiating products with similar packaging, such as water and juice, cereal and tea box, chocolate and spice packets, coffee and chocolate, or tuna, corn or bean cans that are very shaped similar. As a result of the test, out of 50 images, the model managed to correctly detect at least one object in 20 images.

The model was also tested on a set of 5 videos to analyze its behavior. In each video, the model managed to correctly detect at least once the category captured in frames. The worst case was when, out of 20 frames processed per second, the model did not detect the correct category at all, and the best case was when, out of 20 frames processed per second, the model correctly detected the category at least once in 15 frames. The model has problems distinguishing categories with similar packaging or shapes, and also encounters difficulties in detecting products in videos where they are far away, because the products in the training dataset contain relatively close images, so the objects appear large in the image.

Detection speed is equally important to the performance of this system, so the results obtained by the retrained SSD model were analyzed. The model manages to process 20 frames per second, maintaining its performance. The best time, but also the worst time that the model needs to process a frame can be seen in the table 12. This is influenced by the number of objects present in the frame. The results obtained are good, meeting both the accuracy and speed requirements.

Table 12: Time variation for groceries detection.

Information	Value (s)
Best time	0.04
Worst time	0.60

4.5 Exploration System

Because the results provided by the obstacle avoidance algorithm are very important for user safety, several tests were performed to analyze the accuracy of the results.

The first set of tests examines the variation in distance calculated by the algorithm, with the aim of determining its average error to analyze whether it poses a danger. Table 13 shows the variations in the results provided by the algorithm compared to the actual distances.

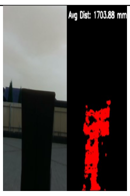
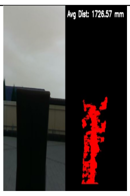
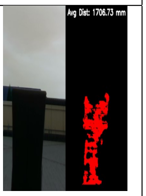
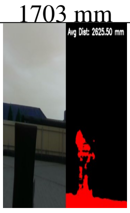
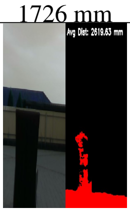
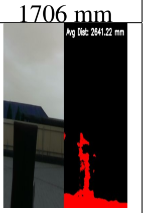
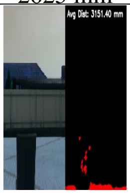
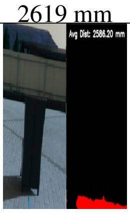
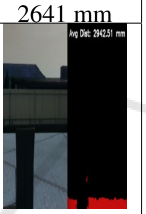
As seen in the table, at distance less than 2600mm, the average distance calculation error is less than 50 mm. However, since the algorithm does not differentiate well between the object and the floor, at a distance greater than 2500 mm, both will be detected as obstacles, and this will reduce the accuracy of the calculation, the average error increasing up to 500 mm. However, because the error only occurs for far distance, it does not affect the algorithm's ability to provide avoidance guidance, effectively detecting obstacles closer to the user.

The accuracy of the algorithm and its effectiveness in guiding the user to avoid obstacles in time were tested on a series of tests with real scenarios. One of the more complicated routes in which the algorithm demonstrated its efficiency is illustrated in figure 7.

The results obtained when meeting each obstacle will be shown next.

In the figure 8, the first obstacle is represented. There is a chair in the user's path that the algorithm detects in time when it is within 2 meters of the user

Table 13: The variation of the distance calculated by the algorithm.

Real d.	Result 1	Result 2	Result 3
1700	 1703 mm	 1726 mm	 1706 mm
2600	 2625 mm	 2619 mm	 2641 mm
3000	 3151 mm	 2586 mm	 2942 mm

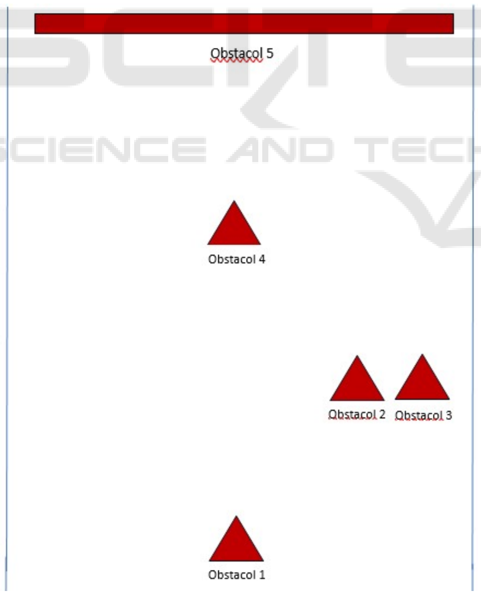


Figure 7: Representation of the obstacle route.

and tells the user that it can be avoided by walking to the right.

In Figure 9, obstacles 2 and 3 can be seen. The user encounters a bicycle and a person in his path, and the only way he can avoid these obstacles is the left side, and the algorithm manages to guide him correctly. It can also be seen in this figure that the viewing angle of the depth camera is greater than that of the RGB camera.

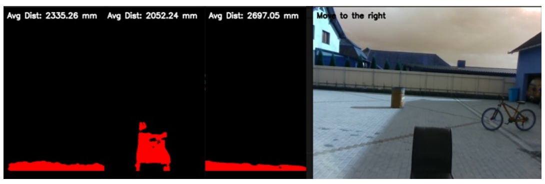


Figure 8: Obstacle 1: a chair is detected and the system directs the user to avoid it by walking to the right.

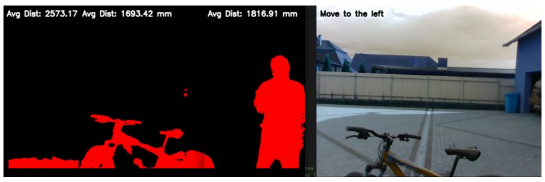


Figure 9: Obstacle 2 and obstacle 3: a bicycle and a person are encountered in the path of the user. The system guides the user to walk left in order to avoid the obstacles.

Obstacle 4, which the algorithm detects in time, can be seen in the figure 10. Upon encountering this obstacle in the user's path, the algorithm tells the user that it can be avoided by walking to the right.

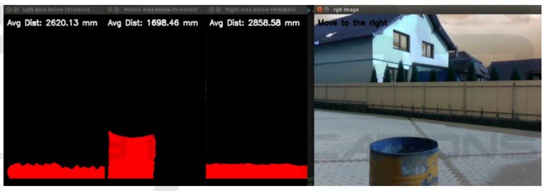


Figure 10: Obstacle 4: the system detects an object that can be avoided, and it alerts the user to walk right.

The last obstacle can be seen in the figure 11. This is represented by a closed gate that the user cannot avoid, and the algorithm correctly tells the user to stop.

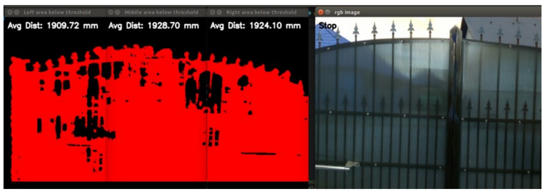


Figure 11: Obstacle 5: a closed gate, the system emits a stop command.

The last case, the one in which there is no obstacle in the way of the user, is represented in figure 12. In this case, the algorithm tells the user that they can go straight.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

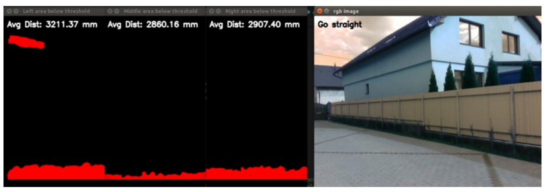


Figure 12: No obstacles in the way of the user, the system tels the user they can go straight.

5 CONCLUSIONS

The implemented system assists visually impaired individuals in perceiving their surroundings through the following three modes of operation: object detection, color detection, and product detection in stores. Thus, the system can identify an object or a color specified by the user or analyze the environment, providing the user with information about the present objects or the predominant color. When the user is in a store, the system can detect the section of products located on the shelves in front of them, helping them to navigate more easily and find the necessary products. Additionally, since navigating without assistance is a real challenge for visually impaired individuals, the system provides support for obstacle avoidance in both outdoor and indoor environments and can also offer a summary of the route the user wishes to take, including information about streets and public transportation.

To test the system, real scenarios were used, which simulated different situations in which visually impaired people might encounter problems, and all these tests obtained encouraging results.

In the future, improvements may be made to the presented system. The most important future development involves making the system portable by powering the Nvidia development board with a battery and adding a Wi-Fi module to not depend on the Ethernet cable for Internet connection. This connection is required for the Google Text To Speech Python library to work.

Another further development involves the addition of a GPS module to provide the route summary mode with information about the user's location in real time. In this way, the system can guide the user in real time to move to the desired destination.

Also, for better system accuracy, it is necessary to retrain the SSD model for in-store product recognition using a larger image set containing a larger variety of product packaging.

Last but not least, in the future the algorithm that calculates the distance to the nearest obstacle can be improved to differentiate the floor from objects. This would improve the accuracy of the calculation, providing more security to users.

We also envision testing the system as a whole with volunteers in real usage conditions.

ACKNOWLEDGEMENTS

The research work related to training, fine tuning and testing some of the deep learning models involved by this paper has been partially supported by the CLOUDUT Project, cofunded by the European Fund of Regional Development through the Competitiveness Operational Programme 2014-2020, contract no. 235/2020.

REFERENCES

Aleksandrov, A. (2020). Groceries object detection dataset. https://github.com/aleksandar-aleksandrov/groceries-object-detection-dataset.

Baskar, V. V., Ghosh, I., Karthikeyan, S., Hemalatha, R. J., and Thamizhvani, T. R. (2021). An indoor obstacle detector to assist the visually impaired person on real-time with a navigator. In *2021 International Conference on Computational Performance Evaluation (ComPE)*, pages 136–141.

CS Kumar, A., Bhandarkar, S. M., and Prasad, M. (2018). Depthnet: A recurrent neural network architecture for monocular depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Devipriya, D., Sri, V. S., and Mamatha, I. (2018). Smart store assistor for visually impaired. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1038–1045.

Jund, P., Abdo, N., Eitel, A., and Burgard, W. (2016). The freiburg groceries dataset. abs/1611.05799.

K, A. F., R, A., S, H., A, S., and R, V. (2014). Development of shopping assistant using extraction of text images for visually impaired. In *2014 Sixth International Conference on Advanced Computing (ICoAC)*, pages 66–71.

Kabir, M. S., Karishma Naaz, S., Kabir, M. T., and Hussain, M. S. (2023). Blind assistance: Object detection with voice feedback. In *2023 26th International Conference on Computer and Information Technology (ICCIT)*, pages 1–5.

Kumar, N. and Jain, A. (2021). Smart navigation detection using deep-learning for visually impaired person. In *2021 IEEE 2nd International Conference On Electrical Power and Energy Systems (ICEPES)*, pages 1–5.

Pesudovs, K., Lansingh, V.C., Kempen, J.H. (2024). Global estimates on the number of people blind or visually impaired by cataract: a meta-analysis from 2000 to 2020. *1*. Accessed: 2024-06-12.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Ranftl, R., Bochkovskiy, A., and Koltun, V. (2021). Vision transformers for dense prediction. *ICCV*.

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. (2022). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3).

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

Rodríguez, A., Bergasa, L., Fernández Alcantarilla, P., Yebes, J., and Cela, A. (2007). Obstacle avoidance system for assisting visually impaired people. *1*.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2019). Mobilenetv2: Inverted residuals and linear bottlenecks.

Saputra, M. R. U., Widyawan, and Santosa, P. I. (2014). Obstacle avoidance for visually impaired using auto-adaptive thresholding on kinect's depth image. In *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*, pages 337–342.

Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne (2019). FastDepth: Fast Monocular Depth Estimation on Embedded Systems. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., and Zhao, H. (2024). Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS