Indoor Envision System for the Visually Impaired

MPhil Thesis

by

Mohammed Noman Mohammed Ridha

Department of Electronic and Electrical Engineering
University of Strathclyde, Glasgow

November 3, 2019

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

The copyright of this thesis belongs to the author under the terms of the United Kingdom Copyright Acts as qualified by University of Strathclyde Regulation 3.50. Due acknowledgment must always be made of the use of any material contained in, or derived from, this thesis.

Signed:

Date:

Acknowledgments

I wish to thank my supervisors, Dr. Vladimir Stankovic and Dr. Ayman Tawfik, for their guidance and support in the research pursing and their motivating feedback through the research period. Most of all, I wish to thank my parents and family, Noman Abbas and Hana Al-Khaffaf for all of their support throughout my MPhil studies.

Also I wish to give notable thanks to Mr. Auda Hazeem and Ms. Mayyada Diab for the opportunity at Nattiq Technologies. This opportunity gave me huge help in the research constructing and work. I wish to thank everyone at Nattiq Technologies that helped me during the research period.

I want to give extraordinary and marvellous special thanks to Mr. Wessam Shehieb, my best friend, and fellow researcher, for his motivation and feedback support all the time. Lastly, I wish to give unique thanks to the one person who inspired me to continue my research studies.

Abstract

This thesis presents a comprehensive assistive technology system, designed and implemented, to support the challenge faced by Visually Impaired (VI) individuals when they enter a new indoor environment. To design such a system, existing similar systems were first reviewed with their features and limitations. The identified limitations of the existing systems motivated the proposed work.

The proposed system consists of a compact unit with a camera and accompanying sensors providing the Visually Impaired individual with a convenient and cost-effective solution, unlike the conventional method of using a white cane. The features are developed to ease visualization of the indoor environment for the Visually Impaired individuals and to make them independent of any external assistance in the visualization of the room. The proposed system was compared with similar existing systems. The evaluation was done by comparing accuracy results between the proposed system and existing systems. The proposed system was discussed with the people having visual impairments and was well appreciated. With a new overall system design and implementation, this thesis contributes to the fields of Electronic and Electrical Engineering, Digital Image Processing and Signal Processing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, a smart system that helps the Visually Impaired individuals in their daily life is presented. The proposed system allows the Visually Impaired individuals to acquire some information about the objects presented in a new visited indoor environment. In this chapter the motivation for research is explained, the research background and literature survey are given, the research contributions are listed and the outline of the reminder of the thesis is detailed.

1.1 Research Motivation

Quality of life for a person is directly related to his/her understanding of their surroundings. Visually Impaired individuals can not easily visualize the surrounding space, and need to touch and identify everything as they move and interact, which creates a big gap in how they perceive their surroundings. According to World Health Organization (WHO), approximately 1.3 billion people live with some form of vision impairment while 36 million people are completely blind [1].

Visually Impaired individuals do not prefer the use of cane for moving and interacting, and many even refuse to use them because they do not want to be different. This sparked a very critical issue in their daily lives. They should be able to visualize the new space they enter. This is the missing gap in many researches that focuses on the blind community. The commercial products and solutions are also focused in other directions, and they do not fill this gap. The same is for the research field.

The motivation for the work presented in this thesis stems from the interaction with Visually Impaired people.

1.2 Background and Related Work

The research started with the background search of existing systems for the Visually Impaired people. Many solutions have been proposed to address the needs of the Visually Impaired. To reach the objective stated in the research motivation, the background research starts with the most common solutions, which are for navigation purposes.

1.2.1 Navigation

Most proposed solutions are to help the Visually Impaired in navigation and obstacle detection. One of the newest systems made is the Smart Cane [2] shown in Fig. 1.1.

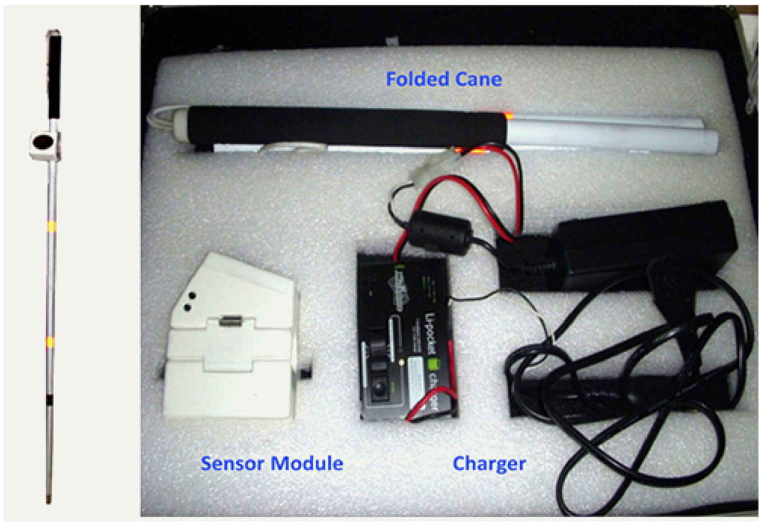


Figure 1.1: The smart cane. Image from [2]

The smart cane is a guidance stick. It comes with "smart" addition which is an Ultra-Sonic sensor that is fixed on the center-top of the cane, with a changeable angle base. It detects the objects and the obstacles in-front of it in a preset range. Then it gives a vibration or a buzzer feedback to indicate that there is an object ahead. The idea of the Smart Cane design was brought forward to have a safe independent mobility for the Visually Impaired, but there were still some limitations.

The problem with the cane is that it does not give an absolute measured distance to the objects, so it is only useful for obstacle detection and navigation. It does not give any extra information about the objects ahead. The preference and satisfaction rates of using a Smart Cane were low with the Visually Impaired people because of these limitations. As was discussed in the paper results [2], the users did not like the extra weight of the sensor on the cane. This system created a confusing environment for the users because of the ultra-sonic wide detection range, resulting in the users losing the space-distance aspects. The Smart Cane is still a viable option for many Visually Impaired individuals, but the added weight and limited information about the objects still made quite a number of people look for better solutions.

There are other guidance systems based on a completely different concept. Such as "Audio guidance system for blind" [3]. This system works with voice commands in smartphone devices. The system make use of the voice recognition services available in smartphones. It creates a very-easy-to-use interface, as the only interaction needed is a voice command. This guidance system will help the blind person to reach the destination. The idea behind such a system is by using Near-field communication (NFC) tags. These tags are placed in various locations, and they contain the information needed to start the guidance system. Fig. 1.2 shows some NFC tags.

Figure 1.2: NFC tags with a smartphone device. Image from [4]

These tags are placed on the designed path for navigation.

This enables the user to get a lot of information by the usage of smartphones or a capable device that can read the tags. One big issue of this system is that the place should already be equipped and changed to fit the new installed tags. This means it can not be placed everywhere and anywhere, making the changes to the place will put a lot of restrictions on availability. NFC/RFID tags require a distance under 10cm to be read. This guidance system can solve the problem of limited information about the objects, but the distance and location physical additions restrict this solution. Another variant of this system is made by using Radio-frequency identification (RFID) tags [5]. These tags can have a higher reading distance, but at the price of bigger reader size. In addition, the RFID readers are not implemented yet in smartphones, meaning the user will need to carry out another system unit to read the tags. Both technologies are an interesting approach. Although this type of systems can not solve the obstacles problem, as it is only an informative guide.

1.2.2 Objects characteristics identification

Other systems that meet different needs of Visually Impaired have been proposed. A system that is used to detect color of objects has been proposed [6]. This color recognition system consists of a pen-sized enclosure with various elements. Fig. 1.3 shows the design of the pen. The first main part is the color sensor at the tip, accompanied by an indicator and feedback LEDs. The second part consists of a processing unit and battery in the middle section. The last part is the output speaker and the power on-off switch.

The concept behind such a system is very simple yet effective. The processing unit reads the data from the color sensor, and then outputs the color detected through the speakers. The results of experiments made the system reach 85% of success rate. This system is very useful for the target audience, but it has many limitations. The first problem with this system is that the sensor needs to be close and directed on the object, as the color sensor reads the light data, any misalignment will create a big error rate. The second problem is that result can be achieved using a smartphone with appropriate software. This makes carrying another system for only one purpose not justified.

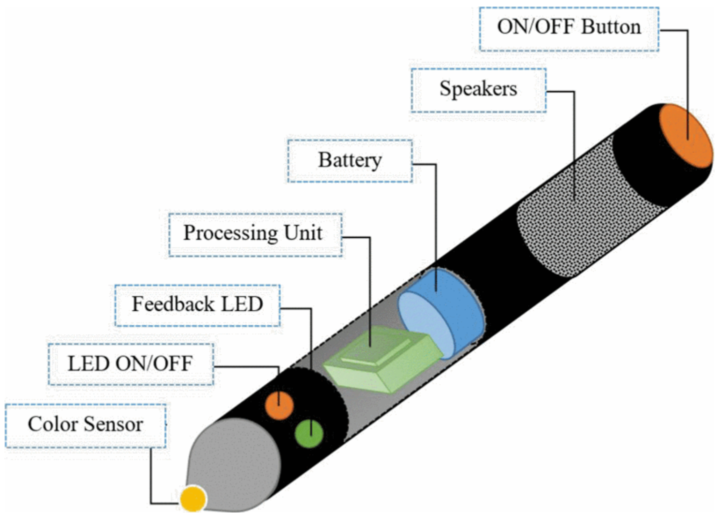


Figure 1.3: Color sensing pen structure. Image from [6]

The third problem is the system can only detect the color of the object, it can not detect any other information or details about the object.

Another system that uses color sensing but in different application is currency identification [7]. Currency notes are different from place to place. Many countries have tried to implement a new currency note that the visually impaired can use. Fig. 1.4 shows the recognition dots on different values of the currency to indicate the value. The problem with this method is that these dots get worn out from usage and become obsolete. This is where the currency identification system comes. It uses a color sensor that detects the currency colors and identify it from the saved color database. This system had a high success rate of 84.72%.

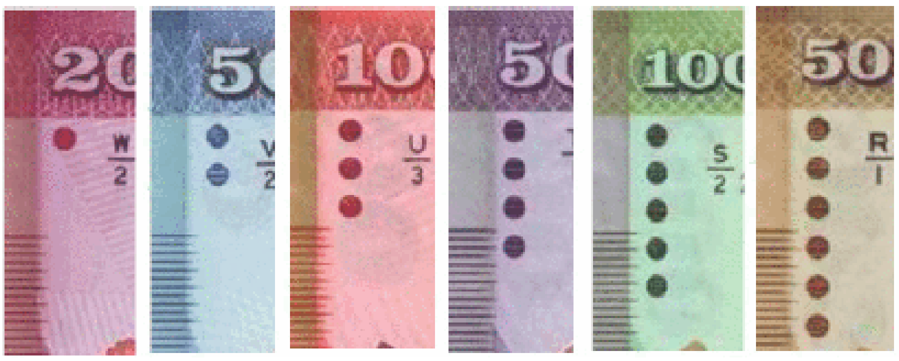


Figure 1.4: Currency recognition dots. Image from [7]

This system has many limitations. First is that each currency uses different color coding, meaning that for each currency you need to add a separate database. Second is the color detection depends on the light available, this means the success rate will change depending on the environment. Lastly, the use of a smartphone software method is still a better approach than using a standalone device. Both color detection systems have their uses, but the limitations discussed above, made them an unpopular choice for the visually impaired.

1.2.3 Area identification

Another recent approach that enables visulasition of the environment is LIDAR (Light Detection and Ranging). LIDAR can be used for real-time object detection [8]. LIDAR is a surveying method that measures distance to a target by illuminating the target with pulsed laser light and measuring the reflected pulses with a sensor. Differences in laser return times and wavelengths can then be used to make digital 3-D representations of the target. The idea behind such a system is by using light reflection across multiple layers. These light data can then be filtered and classified, then drawn out to get a depth map of the area. Fig. 1.5 shows a sample area output of LIDAR systems.

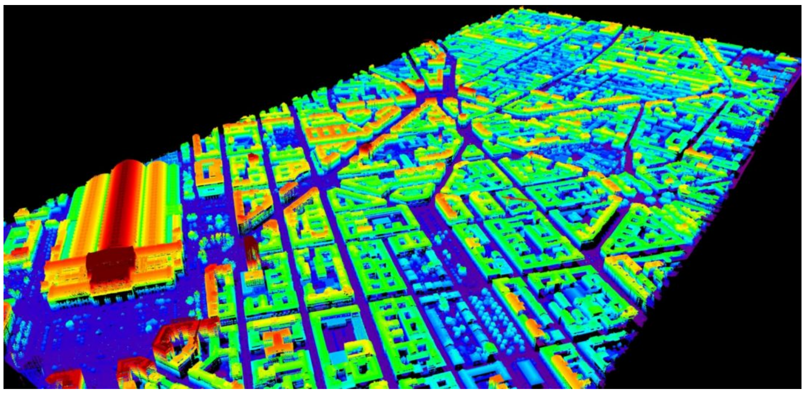


Figure 1.5: LIDAR image output. Image from [9]

This depth map can be used to give users a prescriptive of their environment. This system still has some limitations, the first is that to cover a wide range, the system needs to be mobile to be able to draw the whole area. The second problem is that the process computation load will be high as the amount of light points is huge.

The third problem is that this system can only detect the shape and position of the objects, but other information like the object description and features will require another system.

Another method to map the area around the user is by detecting the plane areas [10]. The detection of walk-able plane areas is done by using an RGB-D camera. An RGB-D image is simply a combination of an RGB image and its corresponding depth image. A depth image is an image channel in which each pixel relates to a distance between the image plane and the corresponding object in the RGB image.

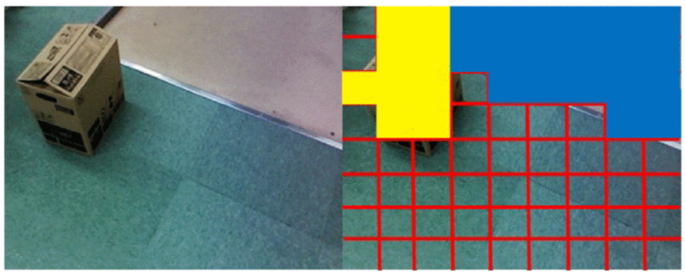


Figure 1.6: Detection of walk-able plane areas. Image from [10]

This system can detect the plane area after processing the depth image. Fig. 1.6 shows how the system can detect the plane area and reject any other parts, like the obstacles (in this case a box) and the sidewalk (as it is lower than the floor). This system is a good approach to detect the area and the obstacles. The use of an RGB-D camera means that the system gets both the depth and standard color image of the area. This system still does not get more information about the object, but it is a step toward the objective.

1.2.4 Microsoft Azure Kinect

One popular depth RGB camera is Microsoft Kinect shown in Fig. 1.7. Microsoft Kinect has been extensively used lately for navigation, object and area detection purposes. First, a quick general overview of the Kinect technology will be given and then some relevant work based on Kinect camera will be discussed. The focus will be on the newest model, the Azure Kinect [11]. The Azure Kinect uses Microsoft Azure cloud platform [12].

Figure 1.7: Microsoft Kinect: 3D-Depth camera and object tracking. Image from [13]

Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers. Azure cloud services have many fields including the compute field. The compute field have sub-categories. The Face library is used for face recognition, while the emotion library is used for emotion recognition. The video library is used to analyze videos. The computer vision supports the object detection and recognition library. The object detection library will be used for comparison with the proposed system of this thesis. The biggest limitation of Azure Kinect is that it relies on constant network connection to the cloud server (online status). This means that any interference or weak coverage will make the services redundant (unusable).

1.2.5 Microsoft Kinect-based systems

A blind navigation system based on the Kinect is proposed in [14]. The system flow is shown in Fig. 1.8.

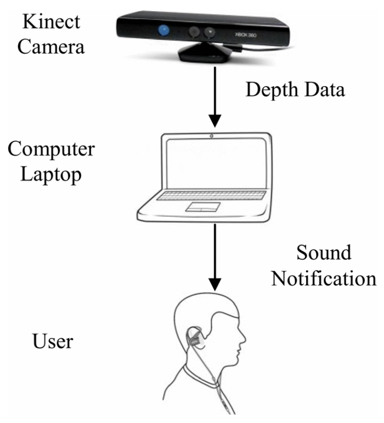


Figure 1.8: Blind navigation system based on Microsoft Kinect. Image from [14]

The system uses the depth data from the Kinect, processes it on the computer, and then outputs the data to the user as audio. This system is similar to the other obstacle detection systems. The main advantage of such a system is using the powerful Kinect libraries. The Kinect depth libraries reduce the workload on the system. When the system recognizes an obstacle, it sends a voice feedback to the user. This system uses the Kinect and the popular processing function “Windows function” [15]. In signal processing, a window function is a mathematical function that is zero-valued outside of some chosen interval, normally symmetric around the middle of the interval, and usually decreasing away from the middle. The system obstacle detection is based on a novel type of windowing technique. This windowing method uses a column based window that scans the image in a recursive pattern, and computes mean or average value of each step. This computer vision algorithm yields high accuracy results. This system is very useful in navigation scenarios. The two main limitations are that this system is still indicating only the obstacles as it is not providing details about the objects, and it is based on the Kinect. The limitations of Kinect will be discussed shortly.

Another similar system is proposed in [16], but with different output method. A vibrotactile feedback system uses the same depth data method of other systems such as [14], but the main difference is that it uses a left and right vibration modules to simulate the object distance to the user. The system flow is shown in Fig. 1.9.

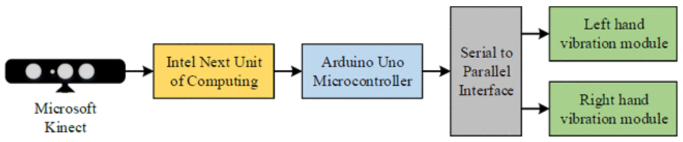

Microsoft Kinect
Intel Next Unit of Computing
Arduino Uno Microcontroller
Serial to Parallel Interface
Left hand vibration module
Right hand vibration module

Figure 1.9: Kinect-based vibrotactile feedback system. Image from [16]

This system is a merge concept between the smart cane system [2], and the Kinect navigation system [14]. This system uses k-nearest neighbors algorithm [17]. In pattern recognition, the k-nearest neighbors algorithm is a non-parametric method used for classification, the input consists of the k closest training examples in the feature space. An object is classified by a vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

With this algorithm, this system can identify fast and accurately the obstacles for real-time navigation. The idea behind the system is promising, as absence of audio feedback means that even the visually impaired with hearing impairment can still benefit from the system. In addition, there are still many users who do not like the audio output. This system still has the limitation of the reviewed other systems, including that it does not give more information about the object, and the use of the Kinect is another factor that will be discussed next.

A lot of research has been done on object detection using Kinect (see, e.g., [18], [19], [20]). The research in [21] uses the Kinect object detection libraries with modifications. The Kinect object detection uses a complex algorithm, but due to the powerful processor inside the Kinect box, it can handle operations fast. This system uses Point cloud method [22]. A point cloud is a set of data points in space. Point clouds can be used to represent volumetric data. [21] uses the Point Cloud Library [23] which is an open-source library of algorithms for point cloud processing tasks supported by Kinect.The library contains algorithms for feature extraction and segmentation. This system uses this library to achieve a high accurate and fast object detection. Fig. 1.10 shows the Kinect normal capture of objects. Fig. 1.11 shows the cut capture of removing the table while leaving the objects only on focus. This system focuses only on the short-coming part from the other systems. The actual object information and description are needed as much as the perspective. This system had an average accuracy of 83% without calibrations. The accuracy increased to 88% with calibrated process.

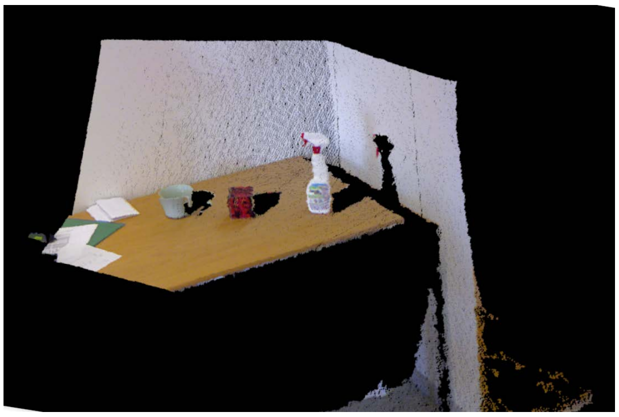


Figure 1.10: Kinect Object Detection sample. Normal capture. Image from [21]

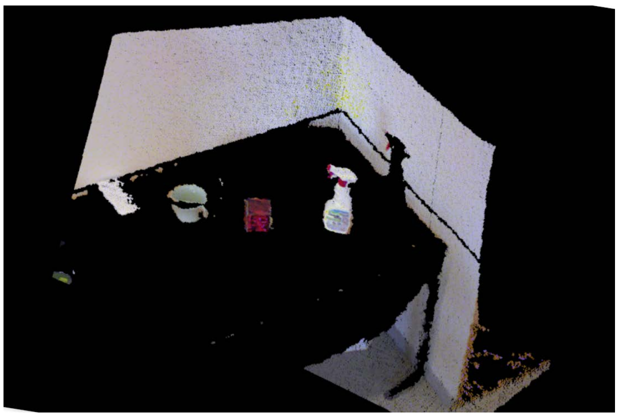


Figure 1.11: Kinect Object Detection sample. Cut capture. Image from [21]

This research concludes that the Kinect is a powerful tool, that can have the ability to get all various information about the object - starting with the object itself (name and description) then the object depth and place in the area-map. Lastly it has the ability to use the Azure cloud, increasing the system portability and reducing the training time. Despite all the above, Microsoft Kinect have many issues. It is a bulky device, with high wattage usage, and a high price. These problems are common for all systems based on Kinect.

1.2.6 Proposed system

The research reported in this thesis aims to remove the limitations of Kinect-based systems while providing similar accuracy. The task is to design a system that can detect objects and provide information about the detected objects. With minimum user interaction, the proposed system (Indoor Envision System) aims to overcome the limitations and problems of the systems based on Kinect as well as other systems reviewed above, while it will try to keep up close in the accuracy and results.

The proposed system should have the following characteristics:

1. Portable, offline (with no need for any network connection), and low power consuming system.
2. Capable to detect objects and their distance to the user.

3. Capable to acquire more information about the objects (i.e dimensions).

4. Possess a user-friendly input and output interfaces.

5. Requires no physical change on the location with minimum amount of interaction.

The proposed system will take into considerations the previous work done on other similar systems. The output and testing of the proposed system will be compared to the existing systems.

1.3 Research Contributions

The research presented in this thesis pertains to the academic fields of Electronic and Electrical Engineering, Digital Image Processing and Signal Processing. The main contribution lies in bringing together latest advances in imaging technology and machine learning into an overall system design. In particular, the following steps were performed to develop the overall system:

1. The performance comparison of Single Board Computers (SBC) and their System-on-Chip (SoC), in terms of the possible features available, the ability to integrate them into a main system and connect different peripherals and the design of a system using SBCs.

2. The image processing of raw image data from a CCD camera, object detection algorithms by the use of machine learning libraries and neural networks with mathematical tensor models.

3. The signal processing of Time-of-Flight (ToF) laser sensor to output the absolute distance from the object and the use of changeable Field-of-View (FoV) laser IR receiver to measure the approximate object height.

In summary, state-of-the-art in hardware and software design, including machine learning and signal processing tools are used to design, implement and test the proposed system and compare the results with state of the art.

1.4 Thesis Outline

This Thesis is organized into three main research parts: Chapter 2 consists of the structure of the proposed system, choosing the most suitable base system that will meet the objectives, the main processing unit and the sensor addition for the proposed system. In Chapter 3, the hardware and software integration of the proposed system main process is detailed. Chapter 4 discusses the proposed system testing setup, output results, the proposed system testing summary comparisons and visually impaired people feedback about the proposed system. Chapter 5 concludes the research and highlights possible improvements for further work.

Chapter 2

Overall Proposed System Description

In this chapter, the overall system is presented. The selection of the main components of the system and its subsystems are discussed in detail. The selection is based on the cost and availability. This chapter also details comparisons between the selected main components of the system and other available systems.

2.1 Overall System

The proposed system consists of three major parts shown in Fig. 2.1:

1. Single Board Computer: The Single Board Computer section is responsible for reading the input and producing the output to the user.
2. Object Detection: The object detection section consists of CCD camera and object detection software model.
3. Time-of-Flight sensor: The Time-of-Flight sensor section consists of a sensor that measures the distance to the object and the object height.

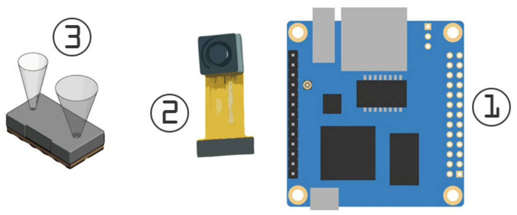


Figure 2.1: The main system components: (1) SBC (2) CCD Camera (3) ToF sensor

The system operation flowchart is shown in Fig. 2.2.

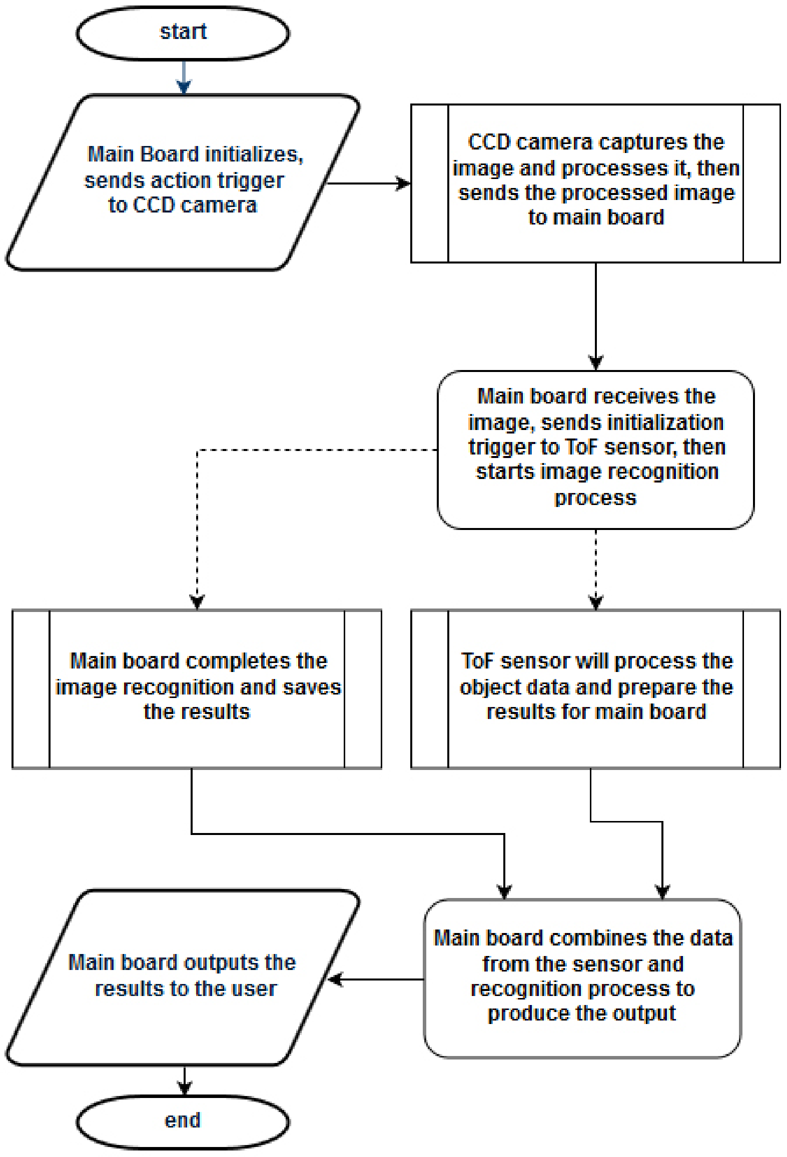

start
Main Board initializes, sends action trigger to CCD camera
CCD camera captures the image and processes it, then sends the processed image to main board
Main board receives the image, sends initialization trigger to ToF sensor, then starts image recognition process
Main board completes the image recognition and saves the results
ToF sensor will process the object data and prepare the results for main board
Main board combines the data from the sensor and recognition process to produce the output
Main board outputs the results to the user
end

Figure 2.2: The flowchart of the proposed system

The system starts with powering up, and initializing, then triggering the process that communicates with the CCD camera to get the processed image data. Then two operations are conducted at the same time; one is the Time-of-Flight sensor operation and the other is the object detection operation. The results of both operations are saved and combined to produce the output to the user.

2.2 Single Board Computer

The system requires a base processing board able to connect to all other system parts including the input and output units and it should be able to execute the high-computational image processing models. Using low-power micro-controllers as the main base board is not feasible since they have a limited processing capability. A much better choice for the main processing base unit is a single-board computer (SBC) [24]. The SBC is a complete computer unit built on a single circuit board. It includes micro-processor, memory, input-output units and it functions as general purpose computer. Many available systems are using such efficient SBCs [25] [26].

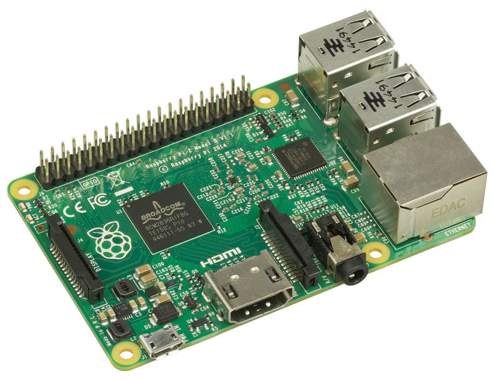


Figure 2.3: The RaspberryPi is a low cost, single-board computer that can be used to design systems. Image from [27]

RaspberryPi (RPi) [27] shown in Fig. 2.3, is often used as SBC. It has a wide range of general purpose input-output (GPIO) pins, and it supports a Camera Serial Interface (CSI), with other features and compact size that offer great system portability.

RPi is a good choice as a base processing board for the proposed system, but it has some limitations:

1. The kernel compatible with its System-on-Chip is a legacy 3.XX kernel. The new mainline Linux kernel is 4.XX [28] which contains improvements for Linux systems.

2. There is no internal storage and the system requires an external storage to work (for example a MicroSD card).

3. The consumption power for RPi is above 1W. This requires the use of a big high capacity battery to maintain decent run-time.

Further research resulted in finding a better option than the RPi. The OrangePi Zero Plus2 [29] shown in Fig. 2.4 has a better and faster CPU, with an internal storage of 8GB, smaller and lighter, consumes less overall power, has better software support and lower price rate. All the other required features are present in both boards like the CSI and GPIO needed for the other parts of the system.

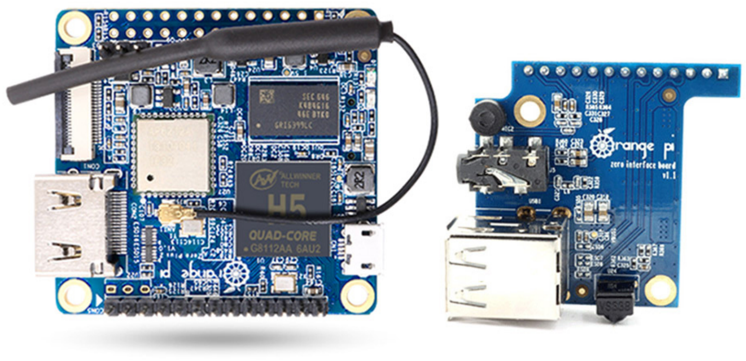


Figure 2.4: The OrangePi Zero Plus2 (left) with expansion board (right). Image from [29]

The comparison between RaspberryPi 2B and OrangePi Zero Plus2 is shown in Table 2.1. From Table 2.1, it is clear that the OrangePi Zero Plus2 outperforms Raspberry PI 2B in most aspects. The only falling point is RAM size where Raspberry Pi 2B has 1GB RAM compared to 512MB RAM of OrangePi Zero Plus2. However, since the processing speed is the most important factor in designing the proposed system, the extra RAM size has limited impact.

SBC	RaspberryPi 2B	OrangePi Zero Plus2
SoC	Broadcom BCM2836	AllWinner H5
CPU	ARMv7 (32-bit) 900MHz quad core	ARM Cortex-A53 (64-bit) 1.2GHz quad core
RAM	1GB	512MB
Internal storage	None	8GB eMMC v5.1
Camera interface	Yes	Yes
Power consumption (avg)	1.75W	700mW
Size (L x W x H mm)	86 x 51 x 21	48 x 46 x 12
Weight (g)	45	20
Software Support	Support legacy Linux Kernel 3.XX only	Support mainline Linux next kernel 4.XX
Price (MSRP)	35 USD	23.90 USD

Table 2.1: Comparison between the RaspberryPi 2B and OrangePi Zero Plus2

2.3 Object Detection

The object detection section is composed of two parts: Tensorflow and CCD camera module.

2.3.1 Tensorflow

Machine learning [30] is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using direct instructions. This means relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data. The models are usually called "training data". Machine learning algorithms are used in a wide variety of applications, including computer vision. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. Creating accurate machine learning models capable of localizing and identifying objects in a single image remains a core challenge in computer vision.

The Tensorflow Object Detection API is an open source framework built on top of Tensorflow that makes it easy to train and use object detection models. Each model is called a Tensor.

Tensorflow supports many platforms. The used platform in this project is Linux-based as the choice of the SBC fits this operating system. The initial test of Tensorflow interfaced with the chosen SBC OrangePi Zero Plus2 using the default compact models proves its compatibility and its needed functionality.

The workflow of Tensorflow Object Detection algorithm is divided into four main stages. First the data images are captured, then the important features are extracted and classified. Finally, the classifications are loaded into a model and tested. The testing data will be a different set than the trained data. Fig. 2.5 shows the workflow.

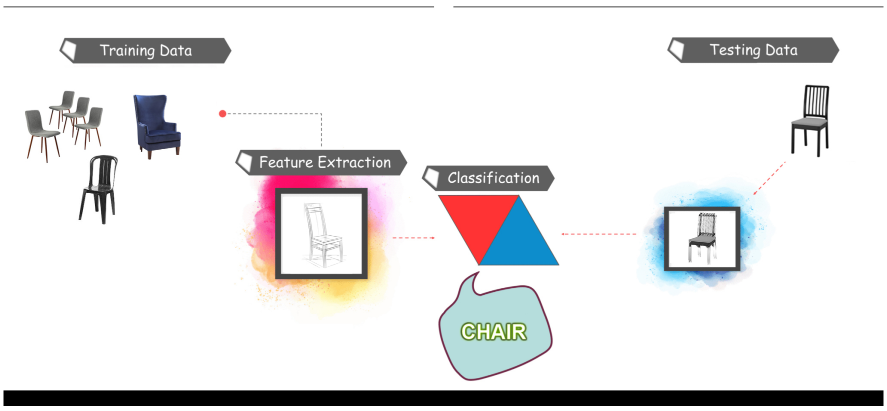


Figure 2.5: Tensorflow Object Detection workflow

Numerous object detection algorithms have been proposed. Recently, Region-Convolutional Neural Network (R-CNN) [31] was proposed to deal with the tasks of object detection, localization and classification.

R-CNN is a special type of Convolutional Neural Network (CNN) that is able to locate and detect objects in images. The architecture of R-CNN is shown in Fig. 2.6. The input image is processed to get the regions extracted. A huge number of regions are then individually warped to compute the features. Then the regions will be classified. The output is generally a set of bounding boxes that closely matches each of the detected objects, as well as a class output for each detected object.

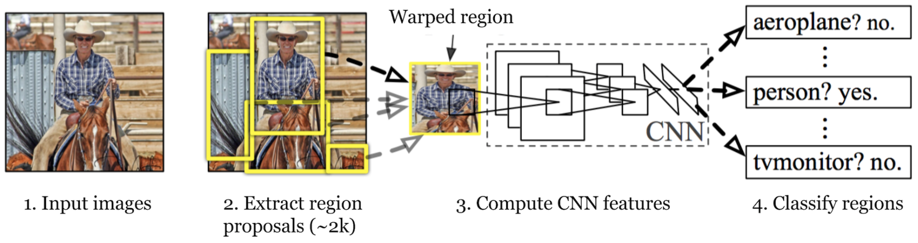


Figure 2.6: The architecture of R-CNN. Image from [31]

Some more recent improvements include Fast-R-CNN [32] and Faster-R-CNN [33]. Despite that, there were still problems with R-CNN such as:

1. Training the data is too long because of the huge number of regions.

2. Training happens in multiple phases.

3. Network is too slow at inference time.

To overcome these problems, two famous new architectures were created: YOLO (You Only Look Once) [34] and SSD (Single Shot MultiBox Detector) [35]. In SSD, Single Shot means that the tasks of object localization and classification are done in a single forward pass of the network. MultiBox [36] is the name of a technique for bounding box regression. Detector means the network is an object detector that also classifies those detected objects. Fig. 2.7 shows a comparison between R-CNN, SSD and YOLO. As shown in the figure, Faster-R-CNN have the highest raw accuracy, while SSD is faster, and YOLO is the fastest but with lower accuracy than the other two. Fig. 2.8 shows the comparison from the accuracy point-of-view only.

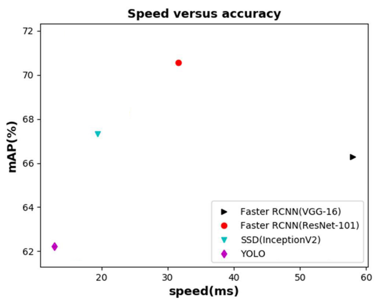


Figure 2.7: Comparison between R-CNN, SSD, and YOLO in terms of speed across accuracy. Image from [37]

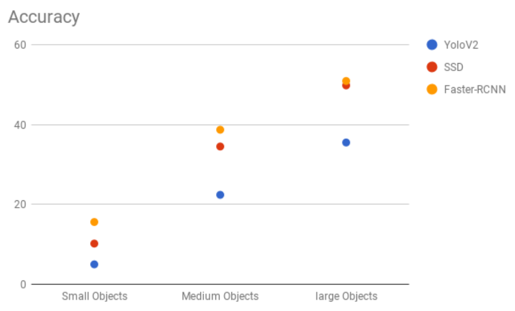


Figure 2.8: Comparison between R-CNN, SSD, and YOLO in terms of overall accuracy. Image from [38]

YOLO have worse accuracy compared to Faster-R-CNN and SSD, while SSD is very close to R-CNN in accuracy. This makes SSD the best algorithm that balances between the speed and accuracy. Due to this, SSD algorithm is used widely in object detection systems and it is the used algorithm in the proposed system. Tensorflow object detection API uses SSD algorithm and several models based on it.

In particular, the proposed system uses Common Objects in Context (COCO) [39] based SSD model. This model is used in Tensorflow applications, as it is fast, has a large database and easy to train for improvements. The integration of the object detection software model will be explained in Section 3.2.4.

2.3.2 CCD Camera Module

The camera module that is used with the proposed system is chosen to be GC2035. As shown in Fig. 2.9, the module contains a flexible cable that help in mounting the camera easily. The camera model supports capturing RGB images up to 1600x1200 pixel resolution.

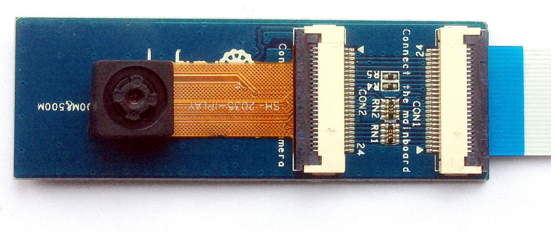


Figure 2.9: CCD camera with the extension board and cable. Image from [29]

To get an efficient use of Tensorflow object detection library, a quick comparison was made to find out the best resolution that should be used by the system. The comparison is done by fixing the camera to get the same object image but with different resolutions. The images are then processed by the standard Tensorflow model to get the output accuracy percentage. The processing time is estimated by the system for comparing purposes.

Table 2.2 shows an indoor test that uses the same CCD camera module with default Object detection settings. As shown in the table, the increase in resolution (number of pixels) increases the process time. Although this increase is not exponential it is still significant.

Camera resolution (pixel)	640x480	800x600	1280x720	1600x1200
Total pixels (in Megapixels)	0.3	0.48	0.9	1.9
Processing time (second)	40	45	70	170
Output accuracy	56%	87%	88%	90%

Table 2.2: Comparison test between different resolutions with CCD camera module

The use of low resolution leads to a poor accuracy, while the use of a high resolution gives a long processing time. From Table 2.2, the change in accuracy between the high and middle resolution is approximately under 5%. The proposed system CCD camera is chosen to use the resolution of 800x600 (0.48MP) because of faster processing time with good output accuracy.

2.4 Time-of-Flight Sensor

Time-of-Flight [40] is a method for measuring the distance between a sensor and an object, it is based on the time difference between the emission of a signal and its return to the sensor, after being reflected by an object. Various types of signals can be used with the Time-of-Flight principle, the most common being light. The usage of Infrared (IR) ToF sensor means that the system can get the absolute distance to the object.

There are many ToF sensors manufacturers available. The proposed system is using ToF sensors from STMicroelectronics.

ToF sensor model	VL53L0X	VL6180X	VL53L1X
Size (L x W x H mm)	4.4 x 2.4 x 1.0	4.8 x 2.8 x 1.0	4.9 x 2.5 x 1.56
Distance range (mm)	0 - 200	0 - 100	0 - 400
Region-of-Interest	Fixed	Fixed	Programmable
Price (MSRP)	2.26 USD	2.26 USD	3.6 USD

Table 2.3: Comparison between ToF sensors from STMicroelectronics

STMicroelectronics offers three models of ToF sensors shown in Table 2.3. As shown in Table 2.3, VL6180X does not meet the objective needed as 100mm is a very short distance. While VL53L1X is the most expensive and also a bit larger, VL53L1X has two main advantages over VL53L0X; the increase in detection range to double and the functionality of having programmable Region-of-Interest (ROI). The use of changeable ROI means that the sensor is able to detect more information than the distance, for example height of the object. Fig. 2.10 shows VL53L1X ToF sensor package.

Figure 2.10: VL53L1X ToF sensor. Image from [41]

VL53L1X default Field-of-View (FOV) is 27 degree [42]. From the datasheet, it can be reduced to 20 degree and even further to 15 degree. This change in FOV results in a change of measured distance which makes the system able to calculate the height or depth of the object referring to the programmable ROI. The ToF operation and integration with other system parts will be discussed in detail in Sections 3.1.3 , 3.2.5.

Chapter 3

Implementation of the Proposed System

In this chapter, the system hardware and software sections are detailed. The available technology is detailed. The comparisons between different technologies of each system block is shown, and this led to the selection of the system components. The main system software function and the system prototype implementation are presented and discussed in detail. The system interface and usage is explained.

3.1 System Hardware

The proposed system follows a standard hardware scheme that consists of input, processing block and output described one by one in the following sections.

3.1.1 User Interface

The system requires the user to initiate the process as shown in the system flowchart in Fig. 2.2. The goal of an input interface is to be efficient and easy to be used by Visually Impaired individuals. There are three attempts made to design the user interface.

The first attempt was to implement a capacitive touch input interface. Touch interfaces are easy to use and to implement. The touch key is shown in Fig. 3.1. However, since the goal is to make the input interface accessible to Visually Impaired individuals, only touch without any feedback is not an efficient solution. The addition of a feedback to the touch like a vibration motor or a buzzer shown in Fig. 3.1, will create an accessible touch interface. But the vibration could disturb the system.

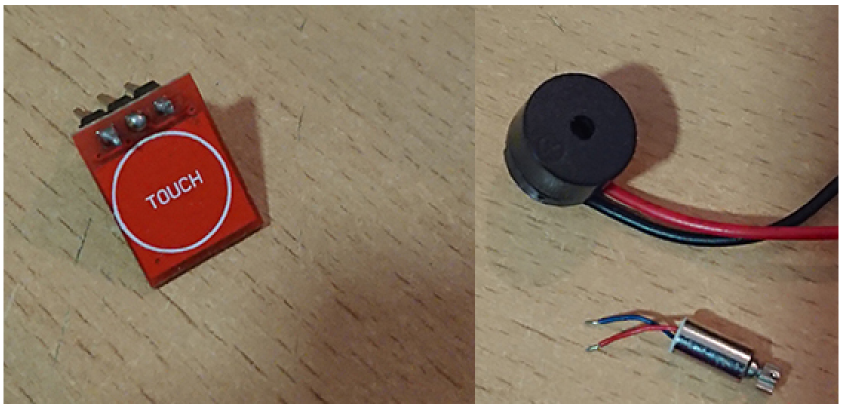


Figure 3.1: Touch key (left) vibration motor and buzzer modules (right)

The second attempt was to test a new input method, a gesture sensor that translates 1D gestures to input commands. The gesture commands are simple and easy to use. The gesture sensor shown in Fig. 3.2 has one main limitation; 10cm is the minimum distance needed to input a gesture. This relatively long distance means that it is hard to use the system while using the gesture commands, making this solution not practical.

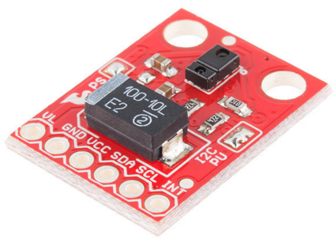


Figure 3.2: APDS-9960 [43]: 1D gesture sensor module. Image from [44]

The third attempted input method is to use a mechanical switch. A mechanical key switch is a simple switch relying on contact between two conductive materials. Mechanical switches have many advantages such as a relatively long life expectancy which means fewer problems with heavy usage. They generate an audible click that gives a positive feedback feel, and they are active only when pressed. These advantages of mechanical switches allow the Visually Impaired individuals to feel the keys and their position before using them.

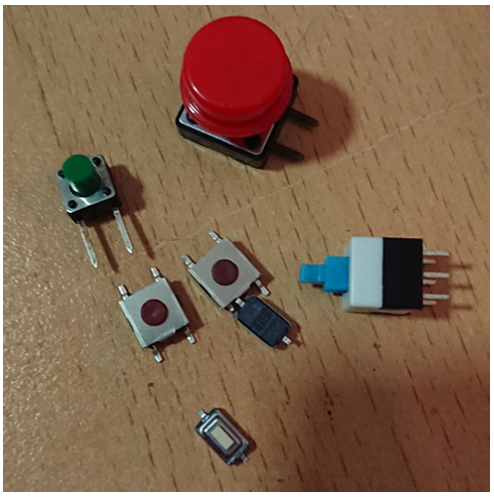


Figure 3.3: Various mechanical keys with different sizes and cap design

There are more than one type of mechanical switches. Fig. 3.3 shows variety of considered mechanical switches. In mechanical switches, poles refer to the number of circuits controlled by the switch, while throws refer to the extreme position of the switch. Because the input commands should be one for each switch, makes the use of double-poles or double-throws switches not suitable. The needed type is the simple Single-Pole Single-Throw (SPST) switch [45]. The input interface PCB is designed using the chosen mechanical switches.

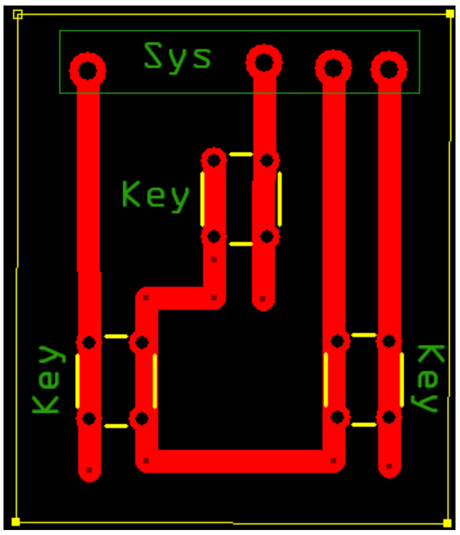


Figure 3.4: Input interface PCB design

The input PCB consists of simple 3 keys, one main key for selection and two sided keys for navigation on the interface options. The PCB prototype design is shown in Fig. 3.4. The software interface is discussed in the software section of user interface.

3.1.2 Main System Unit

The base system used is OrangePi ZeroPlus2 as chosen in section 2.2.

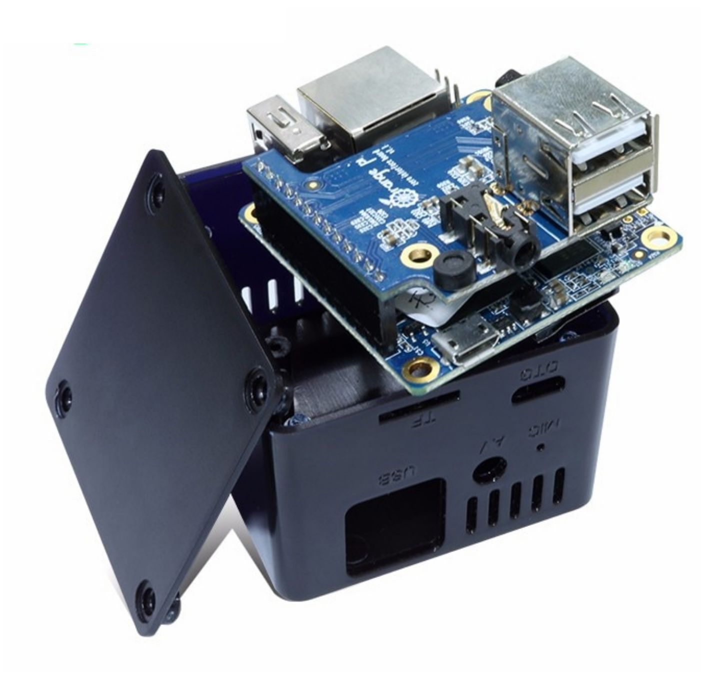


Figure 3.5: OrangePi Zero Plus2 with expansion board and cubic case. Image from [29]

OrangePi Zero Plus2 has a case made for it with all the cuts for the external ports. The system case is shown in Fig. 3.5. The proposed system is using the system cubic case with slight modifications. The cubic case has six sides:

1. Cubic left side:

   The left side contains the input interface PCB mounted. As mentioned in the user interface section 3.1.1, the input PCB consists of 3 mechanical keys. Fig. 3.6 shows the prototype PCB. The red key is the main selection key, while the two sided keys are for navigation.

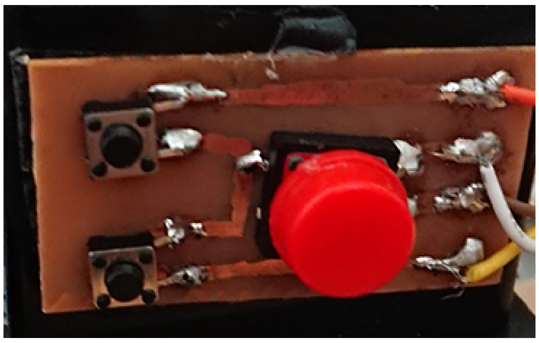


Figure 3.6: Input interface PCB

2. Cubic back side:

   The cubic back side is closed. Fig. 3.7 shows the mounting points of the system when the back cover is opened.

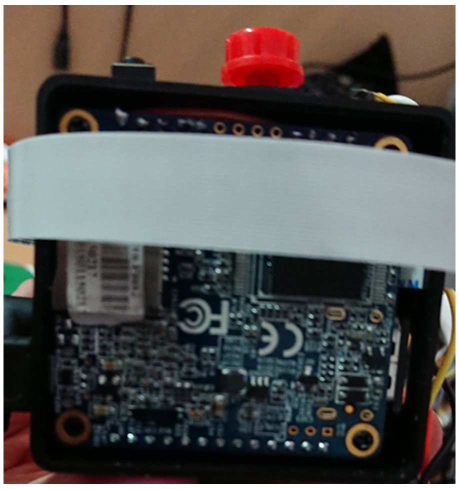


Figure 3.7: Cubic case back opened to show the system mounting points

3. Cubic upper side:

   The upper side of the cubic case is used to pass the serial debugging cables needed to interface with the SBC. The CSI cable, ToF sensor cables and the input PCB cables are also connected to the main system through this side as shown in Fig. 3.8.

4. Cubic right side:

   This side is empty and clean for a hand grip of the system.

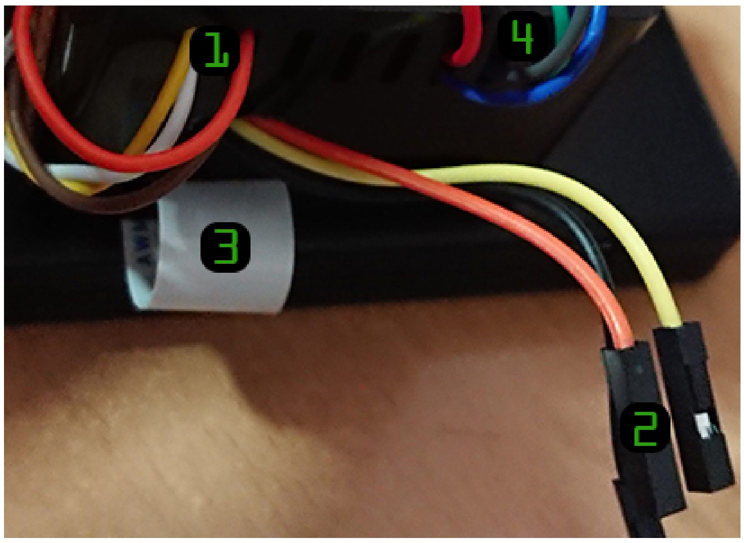


Figure 3.8: Cubic Upper side: 1-Input PCB 2-Serial Debugging 3-CSI 4-ToF sensor

5. Cubic front side:

   This side is the one facing away from the user and toward the object, it contains the CCD camera and ToF sensor. Fig. 3.9 shows the system front side.

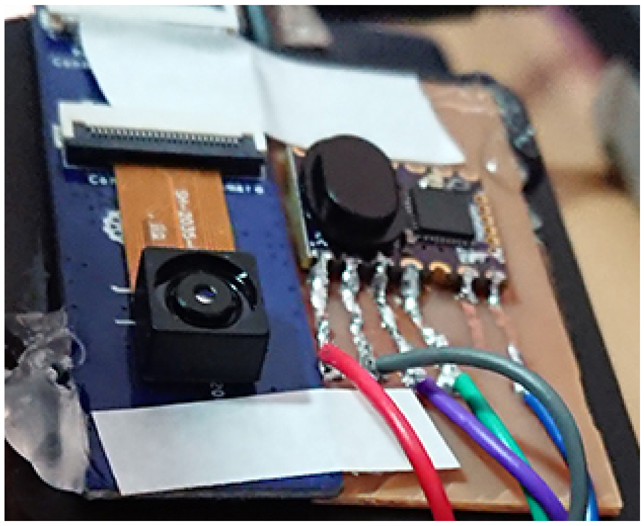


Figure 3.9: Cubic front side: Housing the CCD camera and ToF sensor

6. Cubic lower side:

   This side has the power port and audio/USB outputs ports.

3.1.3 Time-of-Flight Sensor

The Time-of-Flight sensor is fixed at the front side of the system cubic case. The initial testing for this sensor was promising. It was found that the sensor needs significant processing power to be able to use the data acquired meaningfully. The use of CPU processing power leads to a reduction in processing power for the object detection process. This reduction affects the object detection process. Fig. 3.10 shows the ToF sensor with the object detection library on the same processor with total time 48 seconds, versus side-loading the ToF process to an external processor (a micro-controller) to get a total time of 27 seconds.

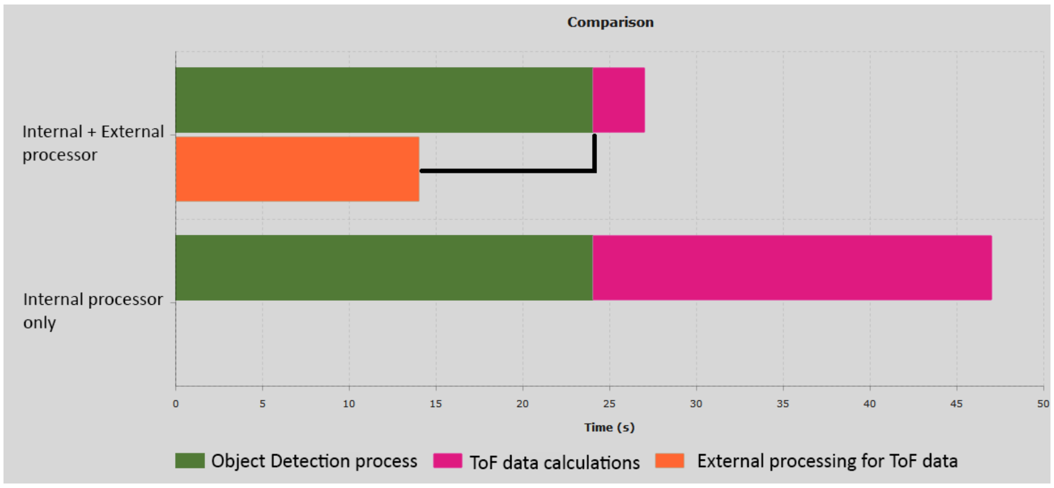


Figure 3.10: Comparison between using ext. processor vs only main processor

The addition of an external processor reduces the total time by approximately 56.25%. This led to the necessity of using an embedded external micro-controller with the ToF sensor. The next step is to set the communication method between the external processor and the internal processor. There are three common communication protocols: 1- Serial UART protocol, 2- I2C protocol, 3- SPI protocol.

The Serial Peripheral Interface (SPI) protocol is dependent on 3 or 4 wires for communication, which makes it a hard method to use. The Serial UART protocol uses 2 wires, one for Rx (receiver) and one for Tx (transmitter). The UART interface is a perfect method if the system needs to have two ways communication.

The UART port can have up to 115.2 Kbit per second (kbps). The I2C protocol also uses 2 wires, but one is for data (SDA) and another for clock (SCL). This protocol works by Master-Slave policy, and can work with a high data rate speed of 400 Kbit per second (kbps). This means the I2C is faster than UART by around 4 times. The limitation of this protocol is that it can work as one direction at a time (Master to Slave).

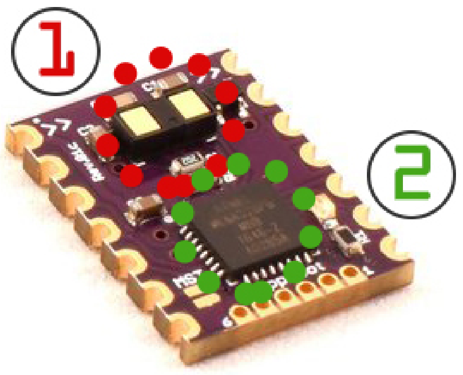


Figure 3.11: ToF sensor(1) with an embedded micro-controller(2)

Since there is no need for real-time two-way communication, it has been chosen that the main processor will communicate via the I2C protocol with the external processor. Fig. 3.11 shows the ToF sensor with its external processor.

3.1.4 Output

The output section is a critical section of the system. The most common output method for the Visually Impaired people is Braille. Braille [46] is a tactile writing system used by people who are visually impaired. Braille users can read computer screens and other electronic supports using refreshable Braille displays. Fig. 3.12 shows a standard 8-dot refreshable Braille cells. The use of Braille cells for output has three disadvantages:

1. The cost of refreshable Braille cells is high due to the patent technology.

2. The power requirements for refreshable Braille cells is 200 DCV [47].

3. The size and weight of refreshable Braille cells is more than the system with case.

All above disadvantages make the use of Braille cells a hard choice for the proposed system. Thus, the search for a more suitable alternative is needed.

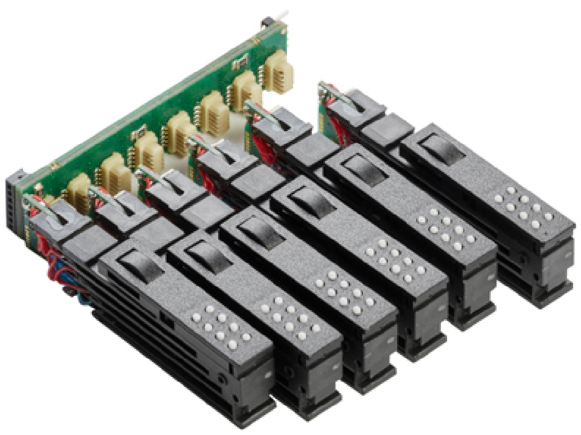


Figure 3.12: Refreshable Braille Cells: 8-dots that display Braille characters. Image from [48]

The alternative for Braille output is using audio feedback output. As many Visually Impaired individuals use voice applications and feel more comfortable with audio outputs, the proposed system output all the system results as an audible output.
The system can support a Bluetooth headset or a wired headset; the wired headset is used for testing prototype. The system is using a Text-to-Speech (TTS) [49] to convert the results to audio. The TTS configurations is discussed in the software section of output stage.

3.1.5 Power

The powering section of the system is also important, as the system does not start or function without a correct powering up module. The OrangePi Zero Plus2 does not have high power requirements as was discussed in section 2.2.

The system will use a standard Li-polymer battery with a dedicated charging/discharging module. The powering of the system is by the standard micro-USB cable attached to the system case lower side. The chosen module has a high capacity of 4000mah. As shown in Fig. 3.13 the actual output capacity is lower due to the internal loss and the conversation rate of the charging module.

The power usage of the system is around 700mW as stated in Table 2.1. With the same required voltage (5V) for the system, this battery module has approximately 5 hours of continuous use.

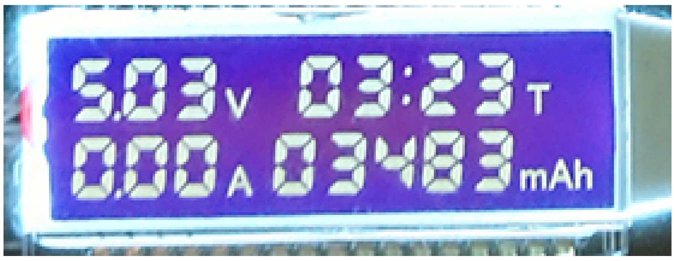


Figure 3.13: Battery module capacity at 5V

3.1.6 System Hardware Prototype

The system prototype implementation is shown in Fig. 3.14. This prototype is used with the implemented software to get the testing results.

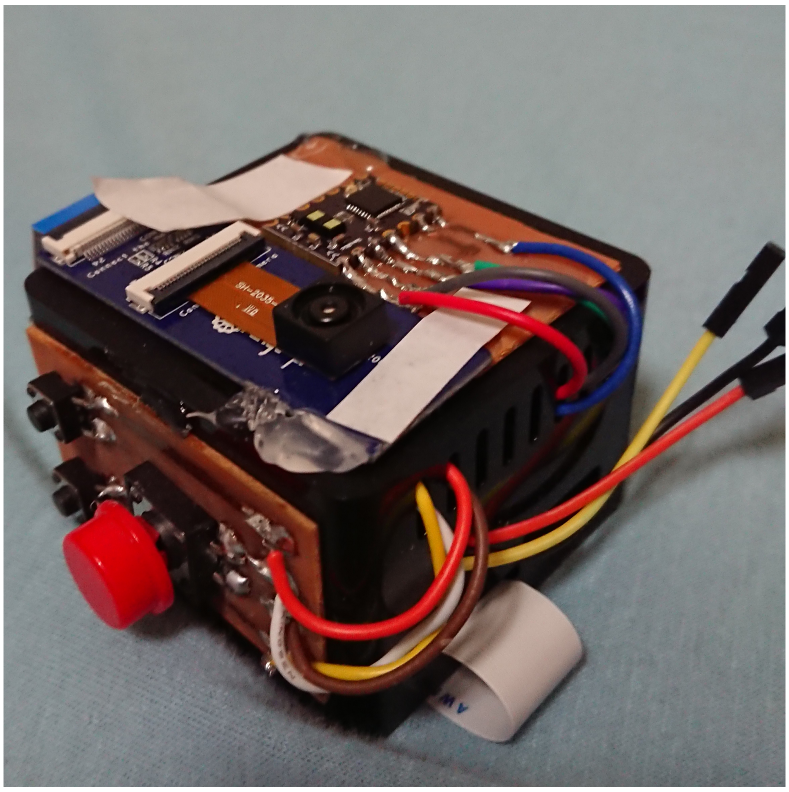


Figure 3.14: System implementation final prototype

3.2 System Software

The OrangePi ZeroPlus2 supports the mainline Linux kernel 4.xx, which is the latest updated kernel and Linux operating system at the time of testing. The used operating system is based on header-less server Lubuntu [50]. The operating system does not contain any desktop or graphical interface, and the only method to access the system is either by serial port (UART) or SSH (Secure Shell) port over network.

The main function of the system is programmed using Python [51]. Python is a high-level programming language for general-purpose programming. It consists of separate modules that each works independently or together to perform a function. Python is a programming language that has full support for Linux systems, and it has support for Tensorflow libraries and hardware interfacing libraries.

3.2.1 Main System Function

The main system function is the program code that runs on system power up. It is the sole code that must be running to control all other processes in the system. The main function must be run on startup. This can be done by using the Linux rc.local script file. The script starts after the boot up of Linux system. The other important note is that the script also disable memory killer and enable swap, to enforce the Linux system to not kill the main process for any reason. The system is locked on the main function unless it gets an exit command or a debug command.

The main system function is “main.py”. It is executed at the end of the startup script. The main function contains the user menu interface, giving the Visually Impaired user a linear command interface to use the system functions.

3.2.2 User Interface

The input interface for the proposed system is important. The Visually Impaired user needs to have a simple and easy to use interface to be able fulfill the system objective.

The system uses a linear user interface. The Visually Impaired user can navigate by the two navigation buttons on the input interface PCB.

The navigation can be done in both ways, once the user wants to select an option he clicks the selection button. The advantage of a linear user interface is that it enables users to learn the system quickly and use it efficiently. Fig. 3.15 shows the flowchart of the menu interface.

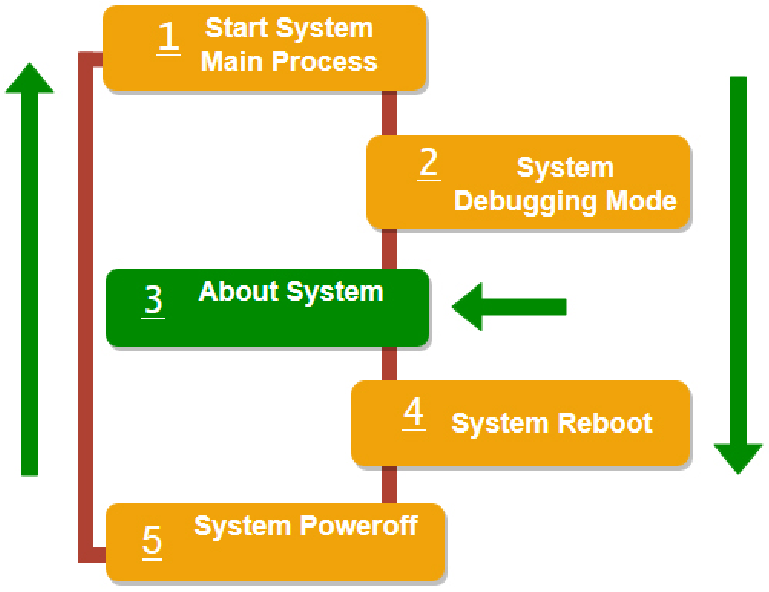


Figure 3.15: User interface menu: linear menu with bi-directional loop navigation

The interface has 5 selectable modes:

1. Start System Main Process Mode: The function part that starts all the system process to get the final results
2. System Debugging Mode: This is debugging mode for testing and calibrations. (It is only set in the menu for testing purpose and will be removed in the final design. The user does not need to do anything with the Debugging mode)
3. About System Mode: Menu to show information about the current system.
4. System Reboot Mode: restart the system.
5. System Power off Mode: power off the system.

Once the logical operation for the interface is finished, the next step is to translate it to Python code using the hardware buttons.

The OrangePi ZeroPlus2 has many GPIO pins. Fig. 3.16 shows the pinout of the board. The input interface PCB buttons is connected directly to the board pins, including the common ground pin.

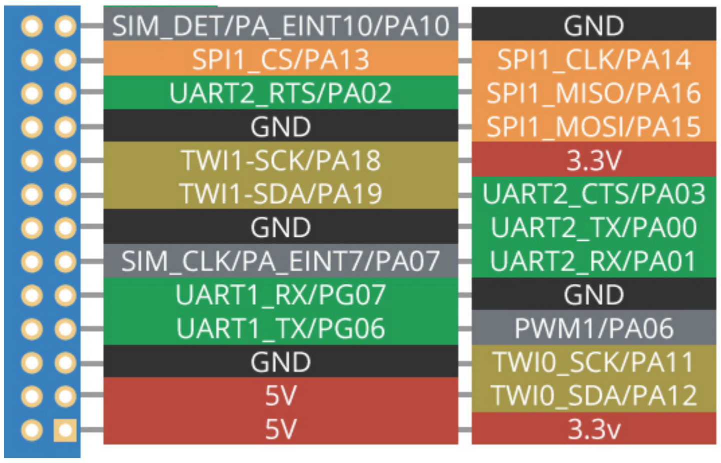


Figure 3.16: OrangePi ZeroPlus2 GPIO pinout. Image from [29]

The chosen pins for the system design are: PA14, PA15 and PA16. The reason for this choice is for the convenience of the prototype design. As these three pins are close to the GROUND pin, this creates an easy to design wires between the Input PCB and the system. The system will use only three input pins, as there are only three buttons on the Input PCB. While all three will share one connection to the GROUND pin. The Python GPIO controlling is done by the pyA20 library. The library first sets the buttons mapping, then it initializes the GPIO. It sets the direction after that; in this case it is INPUT as all three buttons are input commands. Finally, it sets the PULL-UP resistors to make the buttons ready for input reading.

The input reading is direct. If the input button is pressed then it gives a LOW signal to the board, this signal triggers the corresponding function. The navigation variable is set as “menu” while the selection variable is set as “setc”. This simple linear interface is the main function loop. The system will stay on this menu interface until it starts one of the selection functions, or gets interrupted by debugger.

3.2.3 Output

The system output method uses Text-to-Speech (TTS) as discussed in hardware output section 3.1.4.

TTS can be categorized into two main types: online cloud TTS and offline TTS. The online TTS requires an always on connection to the cloud server, which imposes a limitation on the system. Therefore, offline TTS is considered the better choice for the system. There are more than one TTS engine available. The proposed system is using eSpeakNG [52]. eSpeakNG is a compact, open source, software speech synthesizer for Linux and other platforms. It uses a format synthesis method, providing many languages in a small size. Much of the programming for eSpeakNG's language support is done using rule files with feedback from native speakers. This makes eSpeakNG a good and compatible choice to use for generating the audible messages. eSpeakNG has ready-to-use libraries for Linux systems. There are customization for the engine output; for example the sample rate of the voice, pitch, volume level and other voice options. To ensure a good audio quality, the system is using the standard settings for the English (UK) voice. Fig. 3.17 shows the effect of low sampling rate when generating the voice wave.

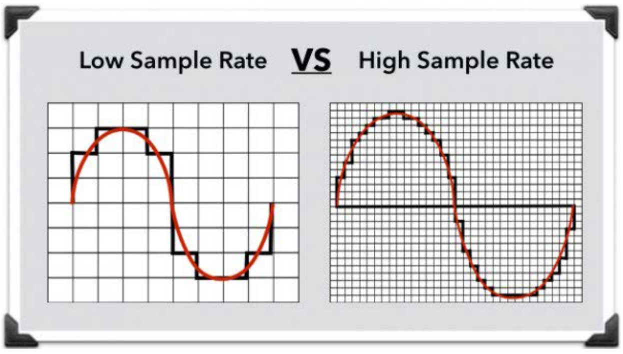


Figure 3.17: Effect of sample rate on the generated wave. Image from [53]

A higher sampling rate gives a better output. The sampling rate of the audio output is chosen to be 48kHz (DVD quality). The TTS engine directly converts the text to audio and outputs it via the chosen audio device i.e headphones. The TTS also output all the system audible messages, including the processing and debugging messages.

The TTS engine works together with the input PCB to create the full user interface. The system process command contains the object detection process and ToF sensor process with the system results output.

3.2.4 Object Detection Function

The system main process starts after the user selection. The main process starts both the object detection process "image_object_detection.py" and ToF sensor process "tof.py" simultaneously. ToF sensor process work is done by the external micro-controller. The result is then retrieved by ToF process.

The Object detection process starts with an audible message to the user, then the process will setup the required environment, imports and model preparations. Then the process loads the Tensorflow model into memory. After the model is loaded, the label map for the model is also loaded. Finally, the system indicates to the user that the loading process is finished. The next step in the process is the image capture. The process communicates with the CCD camera through the CSI interface, the system gives a notice that the image is being taken. Then the system gets the raw image in YUV420P format, resolution of 800x600 pixels (0.48MP) and save it. Then, the system informs the user that the image is saved and then it loads the image to memory. Finally, the process starts the detection process, and give results based on accuracy scores.

The score is based on the accuracy output of the object detection algorithm. The score is retrieved from object detection result. The system only considers scores higher than 70%, lower scores are ignored. The process saves the results and finish by exiting back to main function. The system speaks the results (the object detected).

3.2.5 Time-of-Flight Sensor Function

The ToF sensor process starts at the same time as object detection process. The ToF sensor process is dependent on I2C communications between the main processes and the micro-controller. The process starts with enabling the ToF sensor. The rc.local script turns off the sensor to save the power on startup. The next step is setting the registers on the micro-controller that set the ToF for standard mode.

Then the process waits for the micro-controller readings, once it returns the results the process saves it. The next part of the process is the Region-of-interest data capturing. The ToF sensor have programmable ROI, giving a change in the FOV of the sensor. After reading the data with the changed ROI, a simple mathematics calculation is done to generate the required results from the data acquired. Fig. 3.18 shows how the changeable FoV changes the distance measured. This change can be used in calculation to get an approximate height of the object. The simple equations 3.1 and 3.2 are shown below.

$$tanO = \frac{ObjectHeight}{Distance} \tag{3.1}$$

$$ObjectHeight = tanO * Distance \tag{3.2}$$

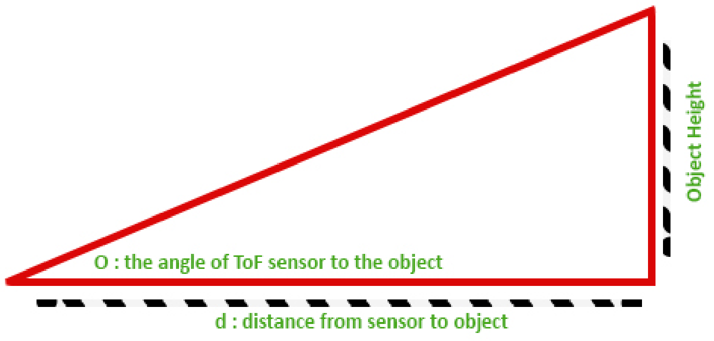

Object Height
O : the angle of ToF sensor to the object
d : distance from sensor to object

Figure 3.18: ToF sensor measurements

The ToF process will do the calculation and save the results. Any result that is out of range is discarded. Finally, the ToF process ends and power off the ToF sensor. The system main process will output the final ToF results after the object detection results. The main process ends after the results output and returns to the main menu.

3.3 System Cost Analysis

This section analysis the system components cost. The system cost is an important factor for comparisons with other systems. The cost advantage is preferable in indirect comparisons. Table 3.1 shows the cost breakdown.

System Component	Price (MSRP USD)
SBC (OrangePi Zero Plus2)	23.90
Expansion board for OPiZP2	1.79
System case	1.99
CCD camera module with cable	5.90
ToF sensor module and external MCU	3.6 + 4.3
Input PCB and output headphones	0.39 + 1.99
Battery module with battery and cable	5.14
Total cost	49.00

Table 3.1: System cost analysis: cost breakdown. Prices reported on December 2018.

As shown in Table 3.1, the total system estimated cost is 49.00 USD. This is less than the estimated 60.00 USD for the smart-cane [2]. The Azure cloud price point [54] is also higher than the proposed system. Fig. 3.19 shows a pie-chart of components percentage. The most expensive part is the SBC with 48.78% of total cost. The SBC is the core of the system, this means that close to half the system cost is in the processing unit. While the least expensive is the system expansion board at 3.65%. The input/output section is at 4.86% of the total cost; the reason is that the input PCB is custom-made as explained in section 3.1.1. The addition of an external MCU to the ToF sensor had an effect of about 8.77% increase on the total system. This addition is improving the speed of ToF process by 56.25% as explained in section 3.1.3, thus the increase is acceptable. The CCD camera module cost is close to the ToF package cost. The only flexible component is the battery module. As explained in section 3.1.5 the capacity of the used module is 4000mAh. The use of different capacity will change the price accordingly.

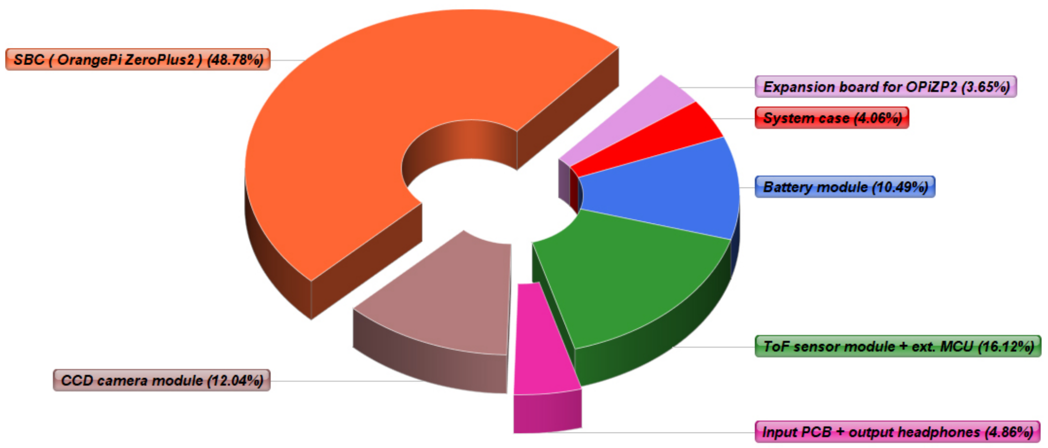


Figure 3.19: System cost analysis: components cost on the proposed system

The proposed system is cost effective; compared to other systems with similar or close functionality. The other advantage of the system components is they are all either open-source [55] or partially open-source; this creates a flexible system structure that can have room for improvements in both hardware and software.

Chapter 4

Testing and Results

This chapter will discuss the system testing. The testing will focus on the accuracy of the system, by reporting performance comparison results with both Microsoft Azure Cloud and Microsoft Kinect. The purpose of testing is to check the usability of the proposed system, that is, if the system meets the research objectives. There is also an indirect comparison with an already researched system. Various comparisons were performed to show all possible sides of the system.

4.1 Testing Setup

The testing setup for the proposed system is as follows:

1. The testing room is assumed to be well lit, with standard office size (8 x 10 ft). The layout is shown in Fig. 4.1.

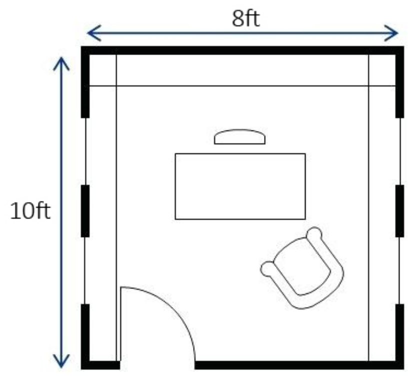


Figure 4.1: Standard room layout.

2. The testing is done with the system prototype and battery module shown in Fig. 4.2. Tensorflow model is standard “ssd_mobilenet_v1_coco” model, ToF sensor with factory default calibrations and the main system process function.

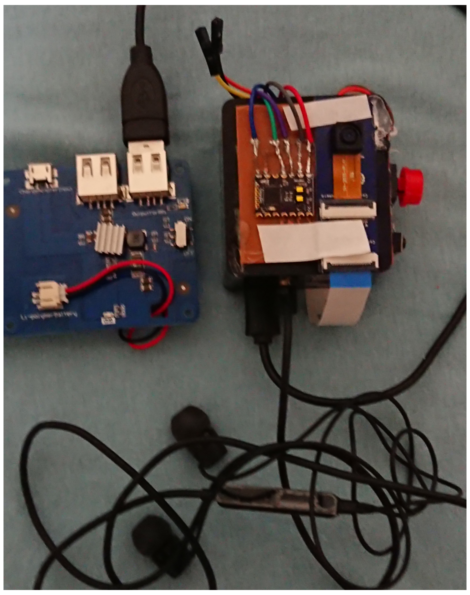


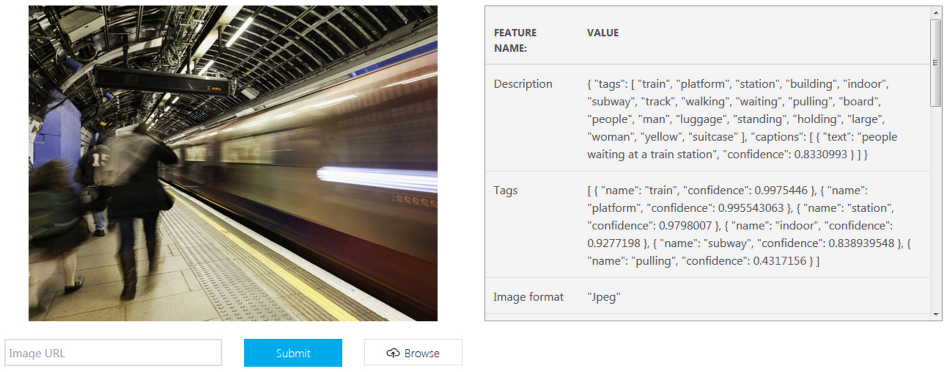
Figure 4.2: System prototype and battery module with headphones.

3. The results are saved for each test. The output is listened via headphones and printed in debug mode. The time measurements are taken with an external digital time-stop clock (in seconds) shown in Fig. 4.3. Any in-system time measurements will not be used to avoid throttling the CPU processing power.

Figure 4.3: Digital time-stop clock.

4. The object actual name is recorded as commonly known. The distance to the objects is measured perpendicularly from the system by a tape meter (in centimeters). The height of the object is measured by a ruler (in centimeters) if applicable.

5. The same test image (.jpg) is used in Microsoft Azure Cloud Vision to get comparison result. Fig. 4.4 shows the results page. The cloud time measurement is ignored due to the required network connection.



FEATURE NAME:	VALUE
Description	{ "tags": ["train", "platform", "station", "building", "indoor", "subway", "track", "walking", "waiting", "pulling", "board", "people", "man", "luggage", "standing", "holding", "large", "woman", "yellow", "suitcase"], "captions": [{ "text": "people waiting at a train station", "confidence": 0.8330993 }] }
Tags	[{ "name": "train", "confidence": 0.9975446 }, { "name": "platform", "confidence": 0.995543063 }, { "name": "station", "confidence": 0.9798007 }, { "name": "indoor", "confidence": 0.9277198 }, { "name": "subway", "confidence": 0.838939548 }, { "name": "pulling", "confidence": 0.4317156 }]
Image format	"Jpeg"

Figure 4.4: Microsoft Azure object detection cloud interface [56].

The system testing is divided into two main parts:

1. Ideal positioning testing: The ideal positioning testing is on fixed objects. The objects are placed in the center, the distance and position are calculated and fixed. The system takes pictures and processes them. This testing shows the ideal scenario results.

2. Natural positioning testing: The natural positioning testing is on objects while moving. The objects are in the room, including on the floor or on the ceiling. This testing shows the normal use of the system by Visually Impaired individuals.

The first testing part shows the system result in the ideal situation. It is used to estimate the limit that the system can reach. The second testing part is the normal test that shows the true nature of the system. Each test image result is shown in a table.

The table shows the image, object actual name, system object results, object distance from system, system calculated distance, object height, system calculated height and Microsoft Azure cloud object result. The Azure cloud test is displayed as an image inside the test table. The results show the description/tag of objects found with confidence score. The confidence score is the same as accuracy score that is explained in section 3.2.4.

4.2 Test results and comparison with Microsoft Azure

The Microsoft Azure Cloud provides a basic cloud server for object detection. Table 4.1 shows the cloud basic server specifications [57] versus the proposed system. The cloud server has a more powerful CPU, while the available RAM is 7 times of what is available in the proposed system. The main advantage of the proposed system is that it can work offline; this advantage helps to overcome any issues when there is a network interference or weak coverage. The proposed system will keep a steady performance regardless of any network loss.

System Component	Proposed System	Microsoft Azure Cloud (Basic)
CPU	Allwinner H5 - 4 cores - 1.2GHz	Intel® Xeon® E5 - 1 core - 2.4 – 3.1 GHz
RAM	512 MB	3.5 GB
Object detection model	Offline API (Tensorflow)	Online API (Azure Computer Vision)

Table 4.1: Comparison between the Proposed System and Microsoft Azure Cloud.

The testing is done with 300 test results. This section will discuss seven samples of those test results. The discussed samples of test results are divided into the two testing categories defined in section 4.1. Additional test results are shown in Appendix A.

The first test result is shown in Table 4.2. The first test is done by ideal positioning method. The object is a water bottle. The object was fixed in place for the test. The system was focused directly on the object to yield the best results (ideal positioning testing). The result of the system is “bottle”, which is an acceptable result.

The table also shows that both distance and length results are within 5% error margin. The Azure cloud result is also “bottle” with 97% confidence. This test is a direct test. Both the system and the Azure cloud got the object correctly.

The second ideal positioning test result is shown in Table 4.3. This test shows one limitation of the system. When there are multiple objects close to each other, the system gives only the first object that it detects. In this test it is a calculator, while the pen and pencil are ignored. The cloud result is also similar, with calculator being the only detected object, while it adds “indoor” or “table” with a “close up of electronics” to the detection list. The distance result is close to the measured distance, having only 2 cm as an error margin. While the length is not applicable due to the angle of the objects to the system. This test shows that both systems were able to get a correct object detection, while the Azure cloud result is wider due to the multi-object detection.

The third test is also an ideal positioning test. The result is shown in Table 4.4. This test is very interesting, as this is the test that had the longest system time. The system result is the same as the object which is a “TV remote”. The distance result is close to the measured distance. Due to the dark object in the image, the Azure cloud results were not close. The highest result is “indoor” which is a generic result. This test shows that dark objects are harder to recognize. Also, they will take longer processing time comparing to other objects.

The fourth test is a natural test shown in Table 4.5. The object in focus is the phone table, the system output is also “table”. The distance/height results are also within acceptable error margin. In this test, the Azure cloud result is far off from the object. The caption result is “a stove top oven sitting inside of a room” with confidence of 31%. The actual valid result is “wall” as it has the highest confidence of 96.6%. In both cases, the cloud result is not correct. Strangely, the Azure result also detected the trash bin with a low 10% confidence. Also, the “indoor” and “floor” results were high in confidence, but this is a very generic result and it is not acceptable.

The fifth test is a natural positioning test shown in Table 4.6. The object is a box or an obstacle. Large items that are placed in the room will be considered as obstacles.

The proposed system rejected all results because the highest detected accuracy score is below 70% as explained in section 3.2.4. The cloud result is also not correct or even close, “a close up of a glass door” with a low confidence 62%. The highest result is “art” with under 10% confidence. This test shows that not even the Azure cloud object detection can detect close obstacles. Also, the proposed system is not valid for obstacle detection, as the main focus is the detection of objects in the room. In this case another sensor or trigger is needed for indoor navigation and obstacles identification.

The sixth test is a natural test shown in Table 4.7. This test shows another limitation of the system - for some objects it detects the reflection. In this test a “window” was given because the door is close to windows design. The Azure cloud result is better here as it was able to detect it is a “close up of a door” with 88% confidence. Also, both “white” and “door” were the highest results in confidence rate. The huge trained database of the Azure cloud shows that it can distinguish better in some cases.

The seventh test is a natural positioning test. Table 4.8 shows the test result. This is another test that shows an important factor. The moving with the system creates motion blur on the images. Also, shaking or rapid movements will create the same blurred images. There is a point until the image will not be recognizable anymore. This test shows an object which is a “sofa” but the image is slightly blurred. The system was still able to get an output result, and that is “Furniture” . The Azure cloud result gave “a living room with a leather chair” with 86% confidence. Both systems gave a valid result as the descriptions may differ from person to another.

4.2.1 Testing Summary

The testing summary pie chart for Microsoft Azure Cloud service is shown in Fig. 4.5. The average detection rate was 76.67%. This average accuracy is obtained with the system setup described in Section 4.1. Other setups may have different average results. On the other hand, Fig. 4.6 shows the testing summary pie chart for the proposed system. The average detection rate of identified objects with both distance and height information is 16.67%, while the average detection rate for identified objects with only distance is at 56.67%. System total average object detection accuracy was 73.34%.

With fewer specifications, the proposed system was able to stay close to the Microsoft Azure cloud system. The average detection rate difference is lower than 10%.

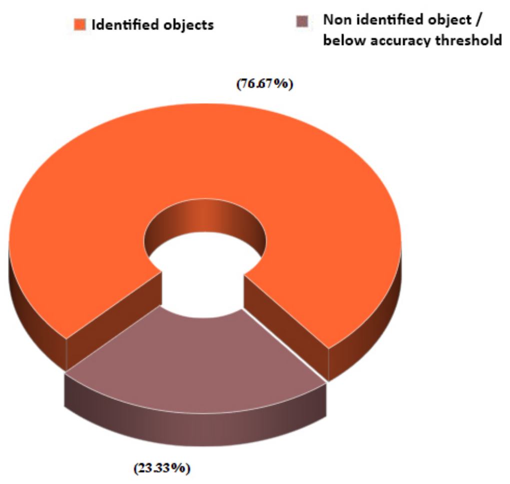


Figure 4.5: Pie Chart of testing summary - Microsoft Azure Cloud.

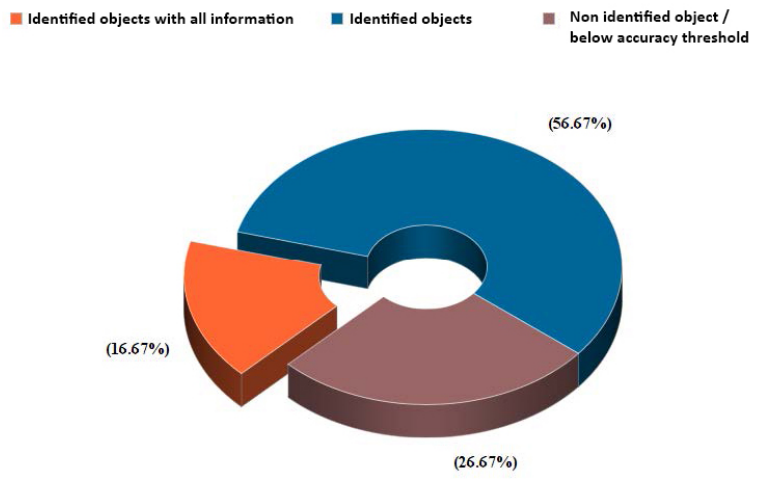


Figure 4.6: Pie Chart of testing summary - Proposed System.

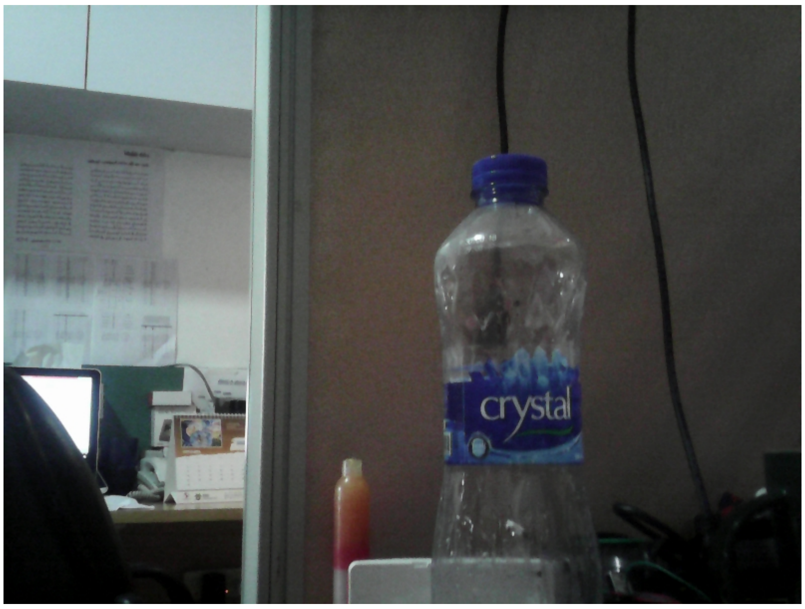
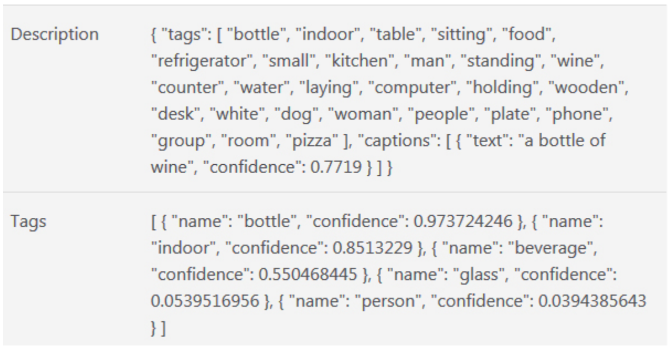
Image	
Object name	Water Bottle
System result with process time (s)	Bottles / 51.90
Object distance (cm)	25
System distance (cm)	23.9
Object Height (cm)	25
System Height (cm)	29.72
Azure Cloud result	 Description { "tags": ["bottle", "indoor", "table", "sitting", "food", "refrigerator", "small", "kitchen", "man", "standing", "wine", "counter", "water", "laying", "computer", "holding", "wooden", "desk", "white", "dog", "woman", "people", "plate", "phone", "group", "room", "pizza"], "captions": [{ "text": "a bottle of wine", "confidence": 0.7719 }] } Tags [{ "name": "bottle", "confidence": 0.973724246 }, { "name": "indoor", "confidence": 0.8513229 }, { "name": "beverage", "confidence": 0.550468445 }, { "name": "glass", "confidence": 0.0539516956 }, { "name": "person", "confidence": 0.0394385643 }]

Table 4.2: Result 1 - Ideal positioning

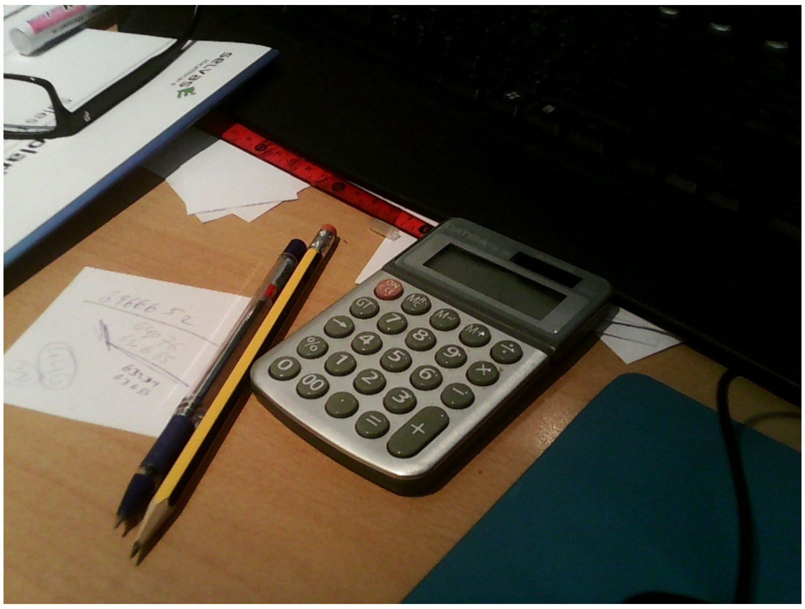
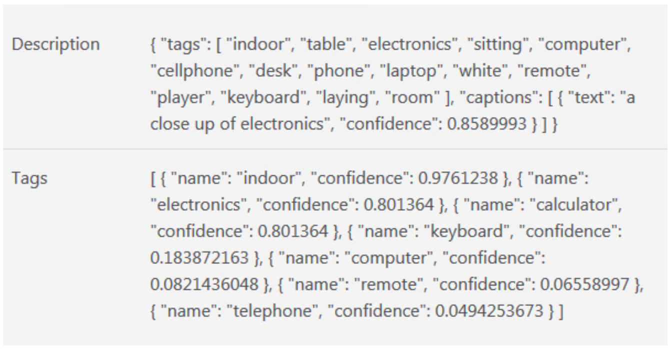
Image	
Object name	Calculator and Pen/Pencil
System result with process time (s)	Calculator / 65.08
Object distance (cm)	20
System distance (cm)	18
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description { "tags": ["indoor", "table", "electronics", "sitting", "computer", "cellphone", "desk", "phone", "laptop", "white", "remote", "player", "keyboard", "laying", "room"], "captions": [{ "text": "a close up of electronics", "confidence": 0.8589993 }] } Tags [{ "name": "indoor", "confidence": 0.9761238 }, { "name": "electronics", "confidence": 0.801364 }, { "name": "calculator", "confidence": 0.801364 }, { "name": "keyboard", "confidence": 0.183872163 }, { "name": "computer", "confidence": 0.0821436048 }, { "name": "remote", "confidence": 0.06558997 }, { "name": "telephone", "confidence": 0.0494253673 }]

Table 4.3: Result 2 - Ideal positioning

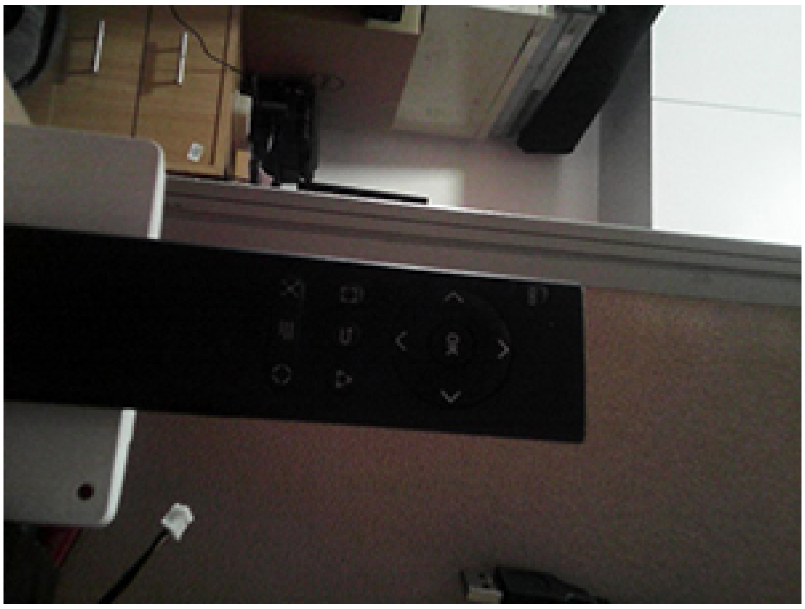
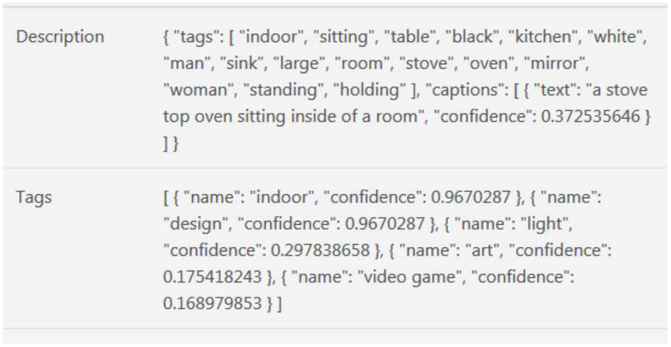
Image	
Object name	TV Remote
System result with process time (s)	TV Remote / 71.34
Object distance (cm)	17
System distance (cm)	16.8
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description { "tags": ["indoor", "sitting", "table", "black", "kitchen", "white", "man", "sink", "large", "room", "stove", "oven", "mirror", "woman", "standing", "holding"], "captions": [{ "text": "a stove top oven sitting inside of a room", "confidence": 0.372535646 }] } Tags [{ "name": "indoor", "confidence": 0.9670287 }, { "name": "design", "confidence": 0.9670287 }, { "name": "light", "confidence": 0.297838658 }, { "name": "art", "confidence": 0.175418243 }, { "name": "video game", "confidence": 0.168979853 }]

Table 4.4: Result 3 - Ideal positioning

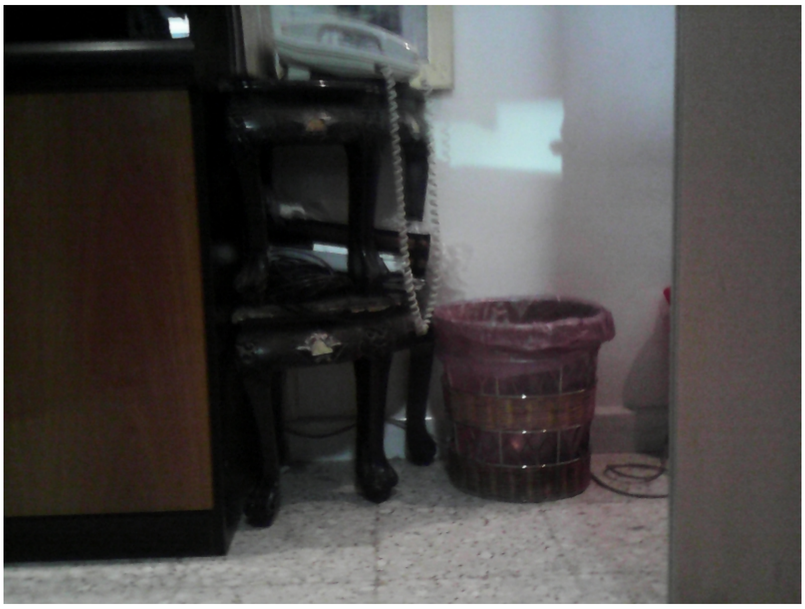
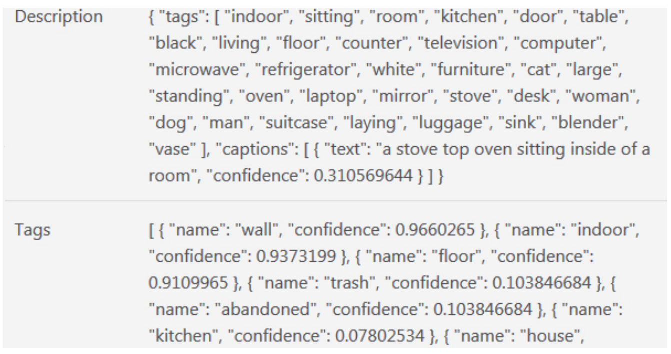
Image	
Object name	Phone table / trash can
System result with process time (s)	Table / 55.09
Object distance (cm)	30
System distance (cm)	31.1
Object Height (cm)	65
System Height (cm)	49.22
Azure Cloud result	 Description { "tags": ["indoor", "sitting", "room", "kitchen", "door", "table", "black", "living", "floor", "counter", "television", "computer", "microwave", "refrigerator", "white", "furniture", "cat", "large", "standing", "oven", "laptop", "mirror", "stove", "desk", "woman", "dog", "man", "suitcase", "laying", "luggage", "sink", "blender", "vase"], "captions": [{ "text": "a stove top oven sitting inside of a room", "confidence": 0.310569644 }] } Tags [{ "name": "wall", "confidence": 0.9660265 }, { "name": "indoor", "confidence": 0.9373199 }, { "name": "floor", "confidence": 0.9109965 }, { "name": "trash", "confidence": 0.103846684 }, { "name": "abandoned", "confidence": 0.103846684 }, { "name": "kitchen", "confidence": 0.07802534 }, { "name": "house",

Table 4.5: Result 4 - Natural positioning

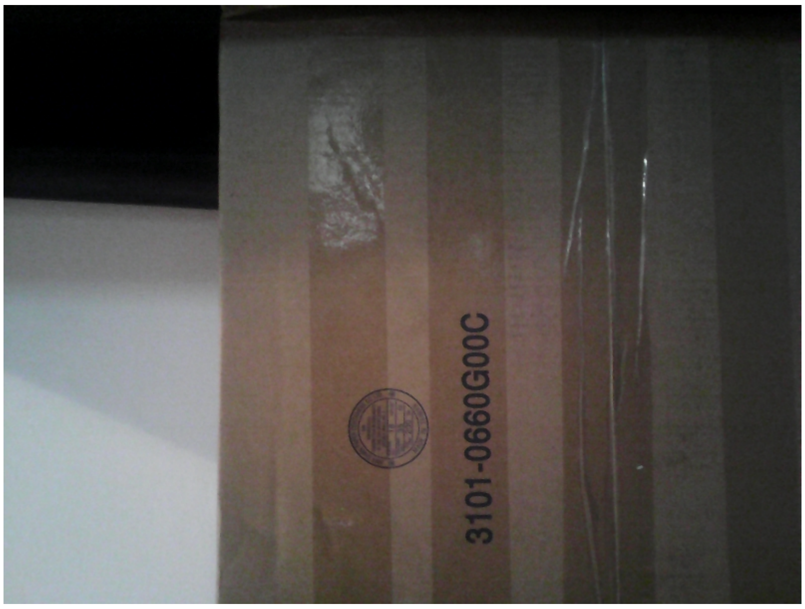
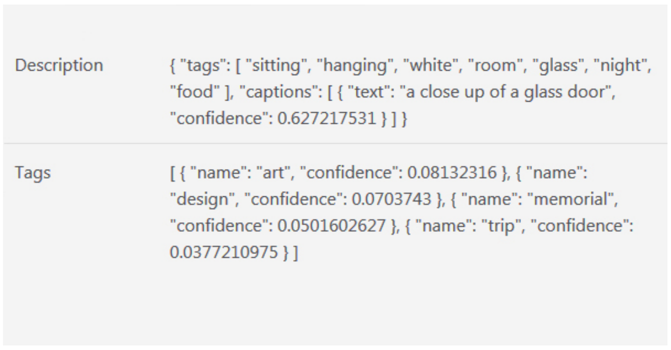
Image	
Object name	Box / Carton
System result with process time (s)	No object detected
Object distance (cm)	Not Applicable
System distance (cm)	Not Applicable
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description { "tags": ["sitting", "hanging", "white", "room", "glass", "night", "food"], "captions": [{ "text": "a close up of a glass door", "confidence": 0.627217531 }] } Tags [{ "name": "art", "confidence": 0.08132316 }, { "name": "design", "confidence": 0.0703743 }, { "name": "memorial", "confidence": 0.0501602627 }, { "name": "trip", "confidence": 0.0377210975 }]

Table 4.6: Result 5 - Natural positioning

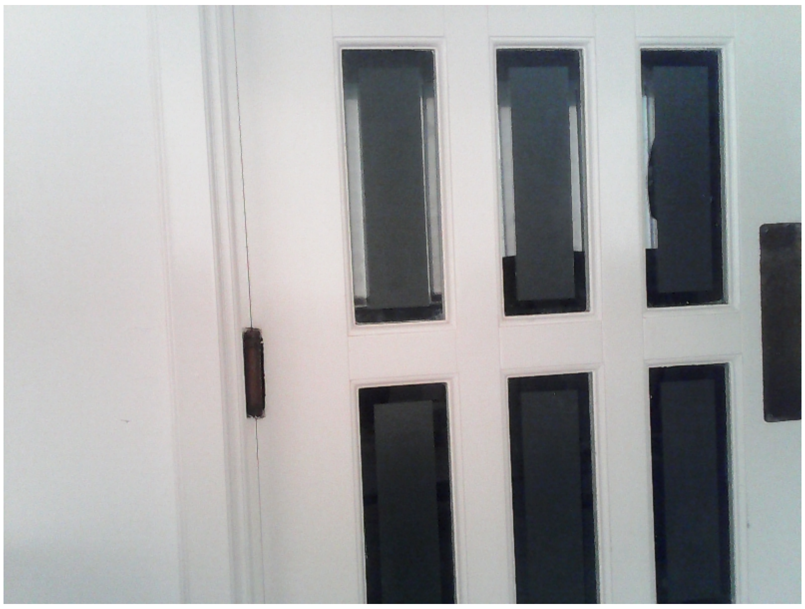
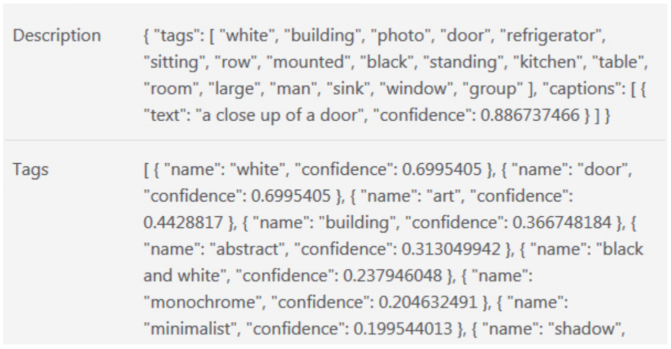
Image	
Object name	Door
System result with process time (s)	Window / 47.81
Object distance (cm)	Not Applicable
System distance (cm)	Not Applicable
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description { "tags": ["white", "building", "photo", "door", "refrigerator", "sitting", "row", "mounted", "black", "standing", "kitchen", "table", "room", "large", "man", "sink", "window", "group"], "captions": [{ "text": "a close up of a door", "confidence": 0.886737466 }] } Tags [{ "name": "white", "confidence": 0.6995405 }, { "name": "door", "confidence": 0.6995405 }, { "name": "art", "confidence": 0.4428817 }, { "name": "building", "confidence": 0.366748184 }, { "name": "abstract", "confidence": 0.313049942 }, { "name": "black and white", "confidence": 0.237946048 }, { "name": "monochrome", "confidence": 0.204632491 }, { "name": "minimalist", "confidence": 0.199544013 }, { "name": "shadow",

Table 4.7: Result 6 - Natural positioning

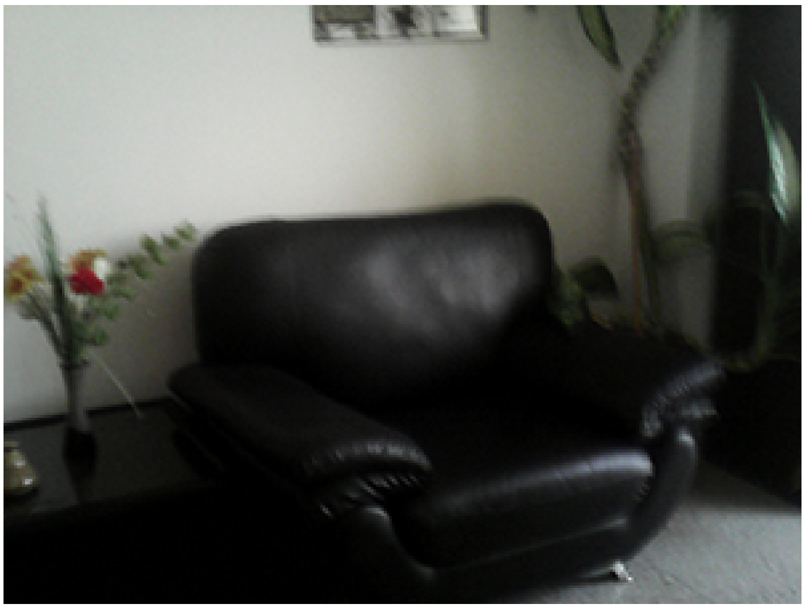
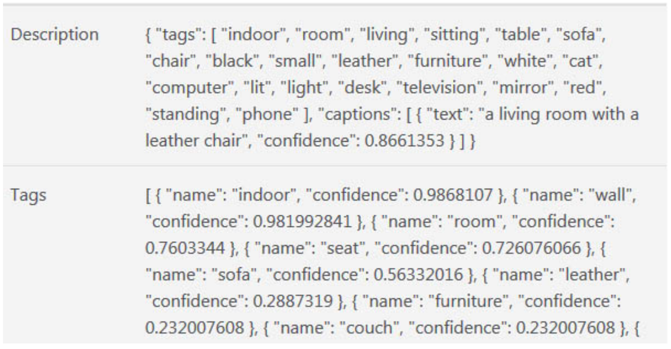
Image	
Object name	Sofa
System result with process time (s)	Furniture / 61.8
Object distance (cm)	Not Applicable
System distance (cm)	Not Applicable
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description { "tags": ["indoor", "room", "living", "sitting", "table", "sofa", "chair", "black", "small", "leather", "furniture", "white", "cat", "computer", "lit", "light", "desk", "television", "mirror", "red", "standing", "phone"], "captions": [{ "text": "a living room with a leather chair", "confidence": 0.8661353 }] } Tags [{ "name": "indoor", "confidence": 0.9868107 }, { "name": "wall", "confidence": 0.981992841 }, { "name": "room", "confidence": 0.7603344 }, { "name": "seat", "confidence": 0.726076066 }, { "name": "sofa", "confidence": 0.56332016 }, { "name": "leather", "confidence": 0.2887319 }, { "name": "furniture", "confidence": 0.232007608 }, { "name": "couch", "confidence": 0.232007608 }, {

Table 4.8: Result 7 - Natural positioning

4.3 Test results and comparison with Microsoft Kinect

The Microsoft Kinect is a powerful hardware that can be used to do many operations. The Kinect has many libraries for object detection. Despite being a powerful tool, the Kinect has many limitations that were discussed in Chapter 1. The main limitations were:

1. The Kinect is a very bulky device, compared to the proposed system and other portable systems.
2. The Kinect is not lightweight.
3. The Kinect is power hungry, it needs at least 24W power supply, comparing to the proposed system that uses less than 1W of power.
4. The Kinect price is almost four times the price of the whole proposed system.
5. The Kinect stills needs another system to utilize the software functions (i.e a windows machine)

Some research was done on object recognition with the Kinect [21]. The results of this work are shown in Fig. 4.7. The testing was done with many options, including calibration and non-calibration options. These options that are included in Fig. 4.7 can be explained as follows:

1. The testing was done with many options, including calibration and non-calibration options.
2. The testing option -v: (min validation point score) is the main option that set and defines the minimum threshold that allows the recognition of the object.
3. The testing option -m: (set unseen pixels to max range) Setting this option the borders of the objects are found even if the area around the object was unseen.
4. The testing option -o:(use orientation invariant version) This option control the orientation of the image of the object.

5. The testing option -t(reverse planar segmentation) This option uses the planar segmentation method.

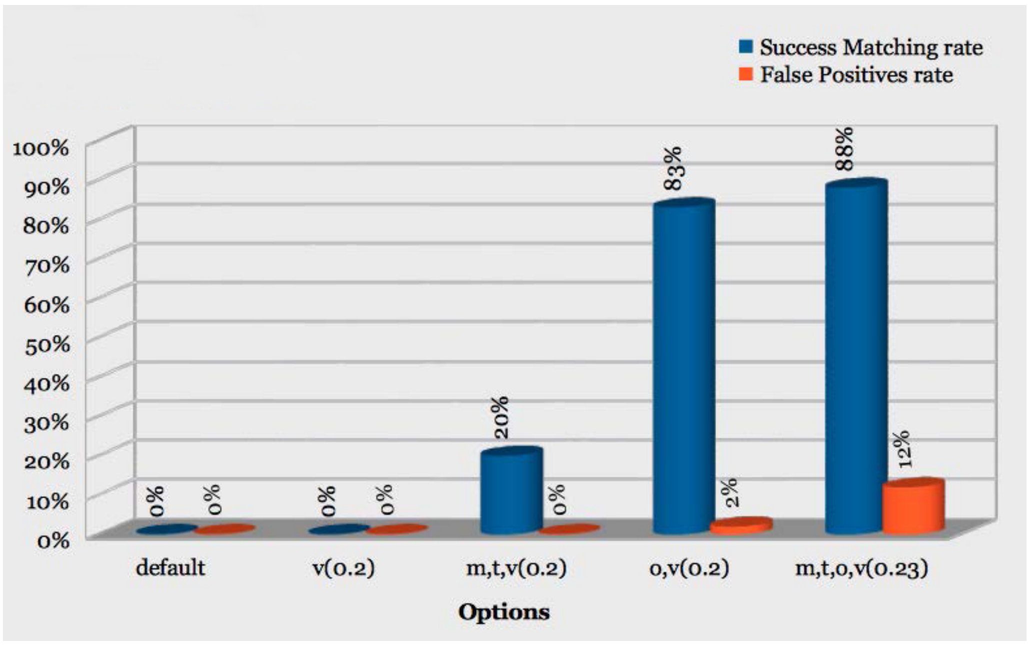


Figure 4.7: Microsoft Kinect Object Detection average results. Image from [21].

As shown in the figure, The Kinect object detection with standard calibrations was able to get a 83% of matching rate (success detection of objects), while with more calibrations, it rose up to 88% matching rate. Comparing the average results of the Kinect with the proposed system average results in Fig. 4.6, the average difference in non-calibration testing is under 10%.

This indirect comparison shows that the proposed system provide very close accuracy results to the Microsoft Kinect. All of the proposed system testing is also done with no advanced calibrations. This means that the difference can be minimized even further by improving the proposed system.

These comparisons and results shows how the proposed system was made by using state-of-art techniques, to go against more complex and advanced systems, while staying close to them in output result.

4.4 Visually Impaired People Feedback

A small survey about the proposed system was performed. The survey was conducted with 20 Visually Impaired individuals at the Emirates Association of the Visually Impaired (EAVI) [58]. The selection was random. Each part of the survey is scored from 1 to 5. 1 being the lowest score and 5 is the highest. The survey is about the following points:

1. System portability/size: This part is about the system size and portability. The smaller and portable the system the higher the score. 5 is for the optimal size, while 1 is for the worst size.

2. System functionality: This part is about the system functionality. The functionality includes the accuracy of the results. 5 is for good results, while 1 is for the unacceptable results.

3. System ease-of-use: This part is about the system ease-of-use. The system level of accessibility and how it is easy to use the functions of the system. 5 is for very simple, while 1 is for very difficult.

4. System features: This part is about the system features and customization. The system features are rated from 5 for useful available features to 1 for no useful available features.

5. System overall rating: This part is about the overall rating for the system. It is based on all and any other factors not mentioned above. 5 being the highest, 1 being the lowest.

The scored results were collected and averaged, then approximated to the nearest natural score. Fig. 4.8 shows the bar-chart of the survey results. The system portability score is high (4); due to the small size of the system. The system functionality score is average (3); because of the system average accuracy (73.34%) shown in testing summary section 4.2.1. The system ease-of-use scored high (4); this is due to the simple user interface designed for the system as shown in sections 3.1.1 and 3.2.2.

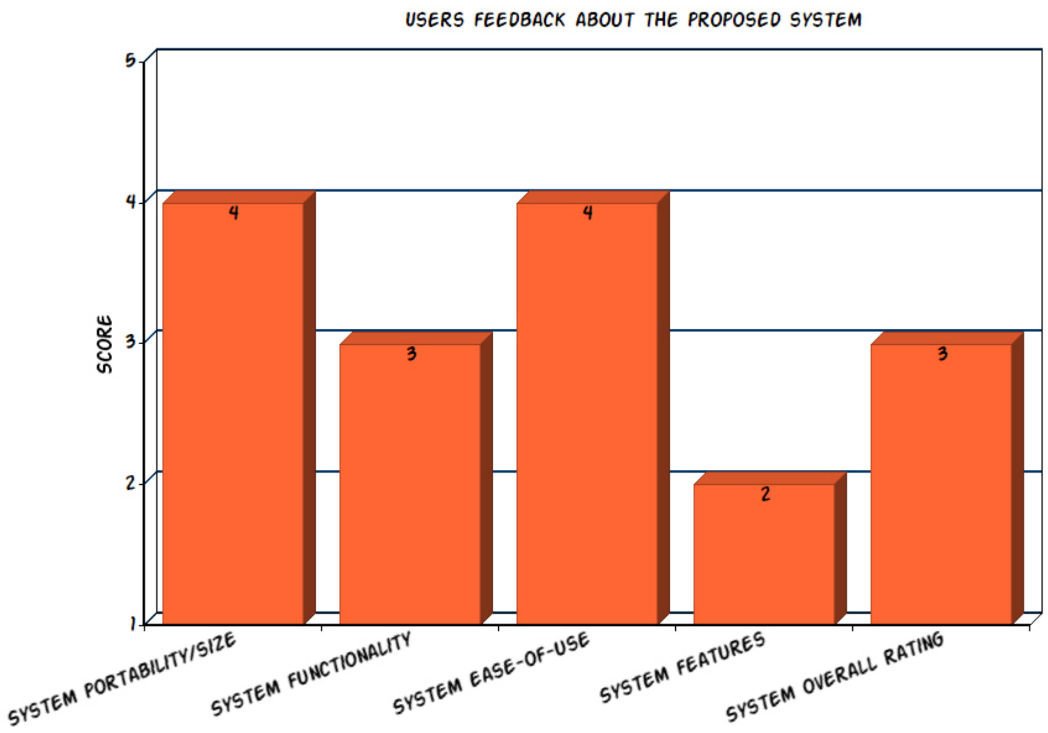


Figure 4.8: Bar-chart for VI People feedback survey about the proposed system.

The system features score is low (2); this is because the system does not offer many customization or other features than the main function. Finally, the system overall rating score is average (3).

The Visually Impaired people feedback is promising, and it shows that the proposed system design is in the correct direction to give Visually Impaired individuals more independence in their life.

Chapter 5

Conclusion and Further Work

This final chapter summaries the conclusions that can be drawn from the results presented in this thesis and provides some ideas for directing further research.

5.1 Conclusion

This thesis has introduced a proposed assistive system designed to help the Visually Impaired individuals indoors. The proposed system consists of a CCD camera with a processing unit and an accompanying Time-of-Flight sensor. The processing unit use the captured images to detect objects; with the help of Tensorflow object detection models. The Time-of-Flight sensor acquires more information about the detected objects to complete the proposed system function. Many tests have been carried out with the implementation of the proposed system. The testing results of the proposed system are promising. The proposed system has an average accuracy of 73.34%, which is comparable with other existing systems with higher specifications. Comparing to already made systems based on Microsoft Kinect, the proposed system is able to stay close in accuracy (under 10% difference) while offering none of the limitations of the Kinect. The proposed system has the advantage of being offline (no network coverage), while existing systems depends on network connections. The proposed system shows a 5% error margin in distance/length results, while other existing systems have a higher error margin. The proposed system results show a few limitations, such as when the object detection accuracy is low it rejects the result then output "no object detected".

5.2 Further Work

This final section highlights some shortcomings of the proposed system presented in this thesis and explains how the proposed system could be further improved.

The object detection accuracy can be further improved. There are two main methods; first is training the Tensorflow model, second is to use a new model algorithm. Tensorflow allows using custom models by training custom datasets [59]. The object detection library have the tools needed to export a new model or an updated model. Fig. 5.1 shows a sample of Tensorflow training process.

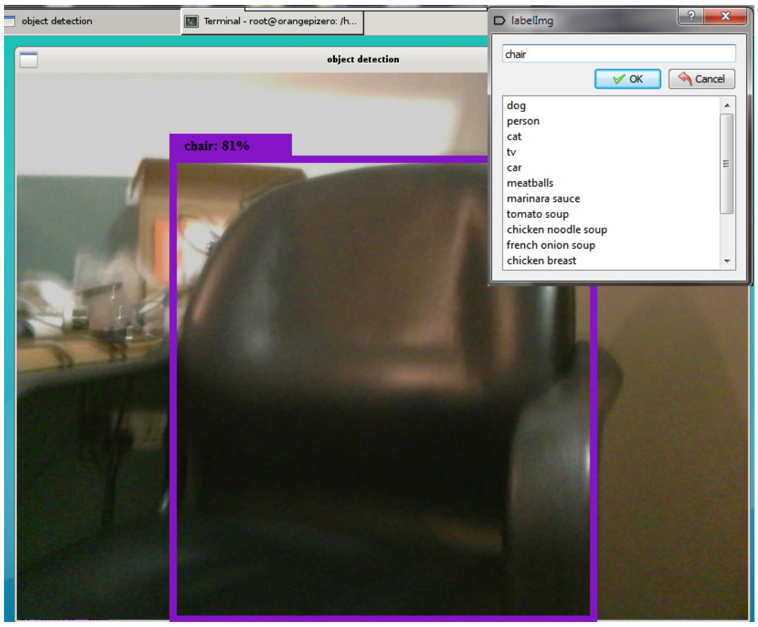


Figure 5.1: Tensorflow model training process

Tensorflow supports a wide range of tensor models. The use of a more complex model will yield higher accuracy results, while training the model will improve the accuracy on trained objects. The object detection process speed is dependent on the processing unit. The use of a more powerful SBC will hasten the process time. At time of writing, the only better SBCs have the limitation of bigger size and power consumption. The input interface can be improved upon; by using a new input method. The output TTS engine can be upgraded for better audio features. The Time-of-flight sensor can be calibrated for improved results.

The sensor used in the proposed system has three different calibration modes, which implement the standard VL53L1X API [60] calibration routines:

1. Reference SPADs calibration: Calibrates the SPAD arrays.
2. Offset calibration: Calibrates the offset of the sensor.
3. Cross-talk calibration: Calibrates the "cross-talk" that is received from a close reflected surface such as a glass cover on the sensor.

The sensor calibration will improve the results and decrease the accuracy error. Another option is to use another sensor for additional accuracy measurements. The system could implement the usage of a depth sensor, but this will increase the price point, size, and the power consumption of the system.

Bibliography

[1] "Blindness and vision impairment," blindness and vision impairment (https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment).

[2] S. Y. Kim and K. Cho, "Usability and design guidelines of smart canes for users with visual impairments," *International Journal of Design*, vol. 1, no. 7, pp. 99–110, 2013.

[3] M. Gharat, R. Patanwala, and A. Ganaparthi, "Audio guidance system for blind," in *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, vol. 1, April 2017, pp. 381–384.

[4] "guidingtech website," (https://guidingtech.com).

[5] E. Bal, "An rfid application for the disabled: Path finder," in *2007 1st Annual RFID Eurasia*, Sep. 2007, pp. 1–5.

[6] M. Samara, J. AlSadah, M. Driche, and Y. Osais, "A color recognition system for the visually impaired people," in *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, Nov 2017, pp. 1–5.

[7] S. Perera and N. Balasuriya, "Visually impaired support-system for identification of notes (vision)," in *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, Sep. 2016, pp. 96–99.

[8] M. P. Muresan, S. Nedevschi, and I. Giosan, "Real-time object detection using a sparse 4-layer lidar," in *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, Sept 2017, pp. 317–322.

[9] "spar3d website," (https://spar3d.com).

[10] K. Imai, I. Kitahara, and Y. Kameda, "Detecting walkable plane areas by using rgb-d camera and accelerometer for visually impaired people," in *2017 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, June 2017, pp. 1–4.

[11] "Azure kinect," azure Kinect (https://azure.microsoft.com/en-us/services/kinect-dk/).

[12] "Microsoft cognitive services," the Cognitive Services APIs harness the power of machine learning (https://azure.microsoft.com/en-us/services/cognitive-services/).

[13] "Microsoft website," (https://microsoft.com).

[14] A. Ali and M. A. Ali, "Blind navigation system for visually impaired using windowing-based mean on microsoft kinect camera," in *2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME)*, Oct 2017, pp. 1–4.

[15] E. W. Weisstein, *CRC concise encyclopedia of mathematics / Eric W. Weisstein.* CRC Press Boca Raton, Fla, 1999.

[16] K. Yelamarthi, B. P. DeJong, and K. Laubhan, "A kinect based vibrotactile feedback system to assist the visually impaired," in *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug 2014, pp. 635–638.

[17] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, pp. 175–185, 1992.

[18] M. S. A. Manap, R. Sahak, A. Zabidi, I. Yassin, and N. M. Tahir, "Object detection using depth information from kinect sensor," in *2015 IEEE 11th International Colloquium on Signal Processing Its Applications (CSPA)*, March 2015, pp. 160–163.

[19] A. F. Elaraby, A. Hamdy, and M. Rehan, “A kinect-based 3d object detection and recognition system with enhanced depth estimation algorithm,” in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Nov 2018, pp. 247–252.

[20] M. S. H. Abdullah, A. Zabidi, I. M. Yassin, and H. A. Hassan, “Analysis of microsoft kinect depth perception for distance detection of vehicles,” in *2015 IEEE 6th Control and System Graduate Research Colloquium (ICSGRC)*, Aug 2015, pp. 116–119.

[21] F. Saponara Iriarte Paniagua, “Object recognition using the kinect.” ser. Trita-CSC-E (Dissertation), no. 2011:115, 2011.

[22] Lihui Wang and Baozong Yuan, “Curvature and density based feature point detection for point cloud data,” in *IET 3rd International Conference on Wireless, Mobile and Multimedia Networks (ICWMNN 2010)*, Sep. 2010, pp. 377–380.

[23] “Point cloud library (pcl),” (https://pointclouds.org).

[24] A. Pajankar, *Introduction to Single Board Computers and Raspberry Pi.* Apress, 05 2017, pp. 1–25.

[25] H. Rasmussen, “A single board computer for robot control,” in *IEE Colloquium on Controllers for Robotic Applications - Concepts and Implementations*, Nov 1989, pp. 5/1–5/4.

[26] K. Akash, M. P. Kumar, N. Venkatesan, and M. Venkatesan, “A single acting syringe pump based on raspberry pi - soc,” in *2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, Dec 2015, pp. 1–3.

[27] “Raspberry pi,” a small and affordable computer that you can use to learn programming (https://www.raspberrypi.org/).

[28] “Linux kernel,” the Linux kernel (https://www.kernel.org/).

[29] "Orange pi zero plus2," orange Pi Zero Plus2 is for anyone who wants to start creating with technology – not just consuming it (http://www.orangepi.org/OrangePiZeroPlus2/).

[30] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2007.

[31] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013.

[32] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1440–1448.

[33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, 06 2015.

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

[35] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.

[36] C. Szegedy, S. E. Reed, D. Erhan, and D. Anguelov, "Scalable, high-quality object detection," *CoRR arXiv*, vol. abs/1412.1441, 2014.

[37] Y. Li, G. Tong, H. Gao, Y. Wang, L. Zhang, and H. Chen, "Pano-rsod: A dataset and benchmark for panoramic road scene object detection," 2019.

[38] "cv-tricks website," (https://cv-tricks.com).

[39] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick, "Microsoft coco: Common objects in context." Springer International Publishing, 2014, pp. 740–755.

[40] "Time-of-flight," the Time-of-Flight principle (https://www.terabee.com/time-of-flight-principle/).

[41] "Stmicroelectronics website," (https://www.st.com).

[42] "Vl53l1x datasheet," page 1 (https://www.st.com/resource/en/datasheet/vl53l1x.pdf).

[43] "Apds-9960," rgb and gesture sensor (https://www.broadcom.com/products/optical-sensors/integrated-ambient-light-and-proximity-sensors/apds-9960).

[44] "Sparkfun website," (https://www.sparkfun.com).

[45] "Spst," single pole, single throw (https://electronicsclub.info/switches.html).

[46] "What is braille?" method of writing words, music, and plain songs by means of dots, for use by the blind and arranged for them (http://www.afb.org/info/living-with-vision-loss/braille/what-is-braille/123).

[47] "Braille cells voltage requirements," 200V +-5%: absolute max. rating 215V 1.2uA typ. per dot (http://web.metec-ag.de/en/produkte-braille-module.php?p=p20).

[48] "metec website," (https://metec-ag.de).

[49] J. Baart and V. Van Heuven, "From text to speech; the mitalk system: Jonathan allen, m. sharon hunnicutt and dennis klatt (with robert c. armstrong and david pisoni): Cambridge university press, cambridge, 1987. xii+216 pp," *Lingua*, vol. 81, p. 265–270, 07 1990.

[50] "lubuntu," lubuntu is a lightweight Linux distribution based on Ubuntu (https://lubuntu.me/).

[51] D. Orban, "Introduction to computation and programming using python," *SIAM Review*, vol. 60, 06 2018.

[52] "Espeakng," software speech synthesizer (https://github.com/espeak-ng/espeak-ng).

[53] "ehomerecordingstudio website," (https://ehomerecordingstudio.com).

[54] "Microsoft azure pricing details," (https://azure.microsoft.com/en-us/pricing/details/cognitive-services/computer-vision/).

[55] A. M. S. Laurent, *Understanding Open Source and Free Software Licensing.* O'Reilly Media, 2004.

[56] "Microsoft azure cloud interface," analyze an image (https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/).

[57] "Microsoft azure hardware specifications," (https://blogs.msdn.microsoft.com/uk_faculty_connection/2016/09/13/quick-overview-of-azure-vm-specifications/).

[58] "Emirates association of the visually impaired," the Emirates Association of the Visually Impaired, was founded in (1983). It was publicized and registered by Ministry of Community Development in (UAE) in (1985). (https://www.eavi.ae/english/).

[59] T. Naga Lakshmi, N. Janaki, and R. Team In, "Deep learning based detection and recognition of objects using mobile nets and ssds," *International Journal of Development Research*, vol. 3, 07 2018.

[60] "Vl53l1x api user manual," um2356: VL53L1X API user manual (https://www.st.com/resource/en/user_manual/dm00474730.pdf).

Appendix A

System Test Results

This Appendix contains more test results. The results follows the same table format used in the system testing in section 4.2. Each test result is in a separate page. To keep the Thesis from over-sizing, only 16 results are shown here out of the total 300 (excluding the ones shown in section 4.2). The ideal positioning results are first, followed by the natural positioning results.

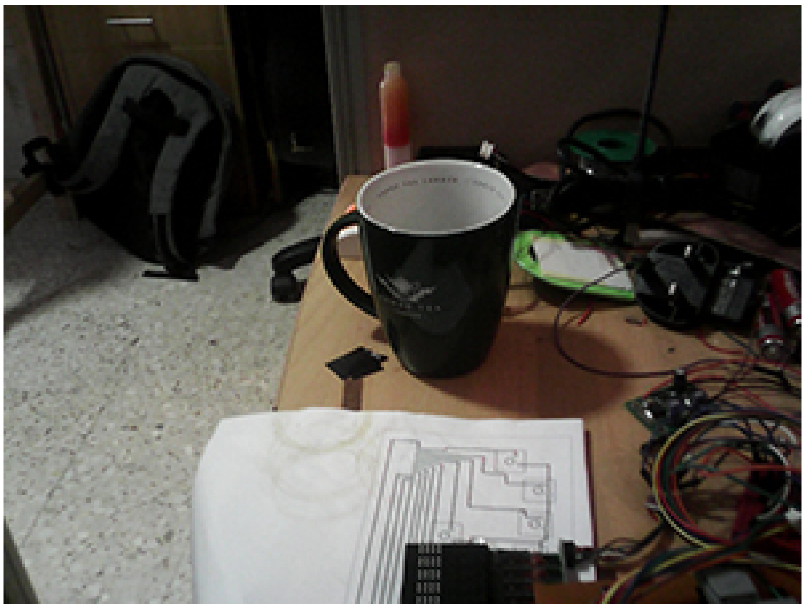
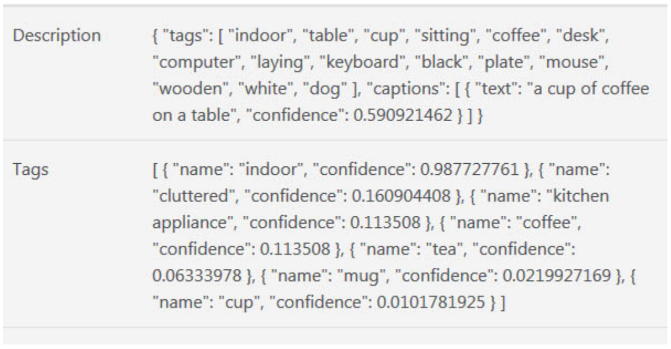
Image	
Object name	Cup/Mug/Glass
System result with process time (s)	Glass / 48.67
Object distance (cm)	35
System distance (cm)	33.7
Object Height (cm)	13
System Height (cm)	14.81
Azure Cloud result	 Description: { "tags": ["indoor", "table", "cup", "sitting", "coffee", "desk", "computer", "laying", "keyboard", "black", "plate", "mouse", "wooden", "white", "dog"], "captions": [{ "text": "a cup of coffee on a table", "confidence": 0.590921462 }] } Tags: [{ "name": "indoor", "confidence": 0.987727761 }, { "name": "cluttered", "confidence": 0.160904408 }, { "name": "kitchen appliance", "confidence": 0.113508 }, { "name": "coffee", "confidence": 0.113508 }, { "name": "tea", "confidence": 0.06333978 }, { "name": "mug", "confidence": 0.0219927169 }, { "name": "cup", "confidence": 0.0101781925 }]

Table A.1: Result 8 - Ideal positioning

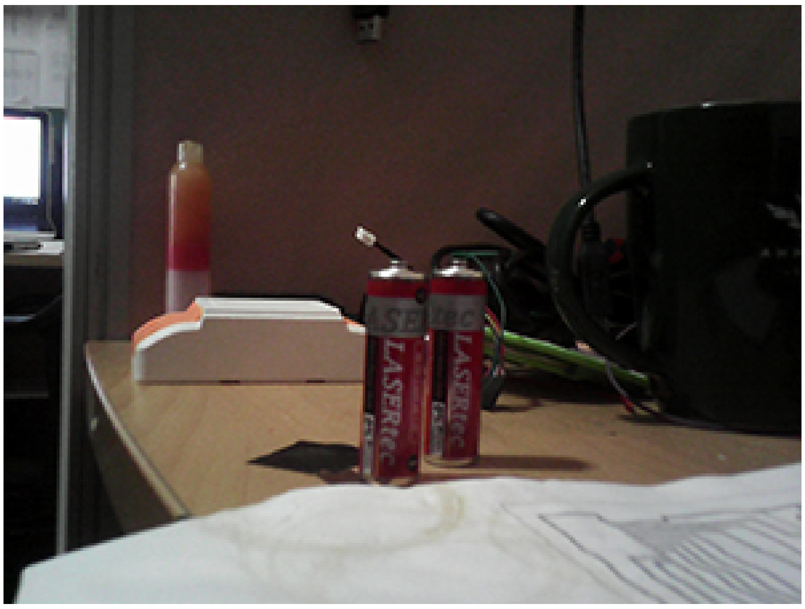
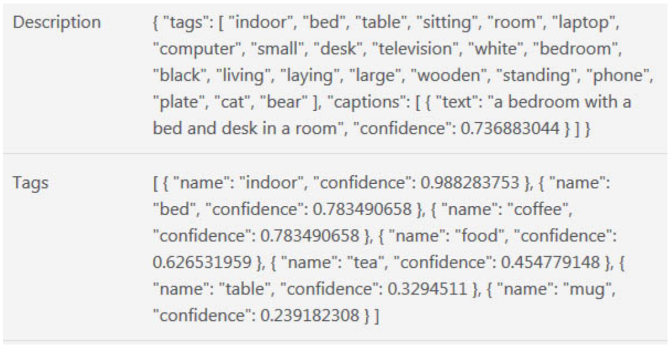
Image	
Object name	Battery
System result with process time (s)	Battery / 52.89
Object distance (cm)	15
System distance (cm)	16.2
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description: { "tags": ["indoor", "bed", "table", "sitting", "room", "laptop", "computer", "small", "desk", "television", "white", "bedroom", "black", "living", "laying", "large", "wooden", "standing", "phone", "plate", "cat", "bear"], "captions": [{ "text": "a bedroom with a bed and desk in a room", "confidence": 0.736883044 }] } Tags: [{ "name": "indoor", "confidence": 0.988283753 }, { "name": "bed", "confidence": 0.783490658 }, { "name": "coffee", "confidence": 0.783490658 }, { "name": "food", "confidence": 0.626531959 }, { "name": "tea", "confidence": 0.454779148 }, { "name": "table", "confidence": 0.3294511 }, { "name": "mug", "confidence": 0.239182308 }]

Table A.2: Result 9 - Ideal positioning

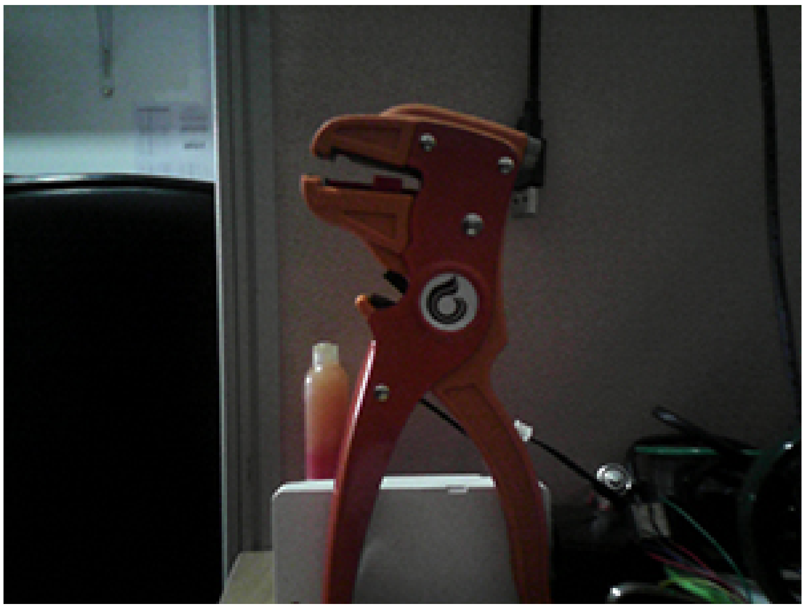
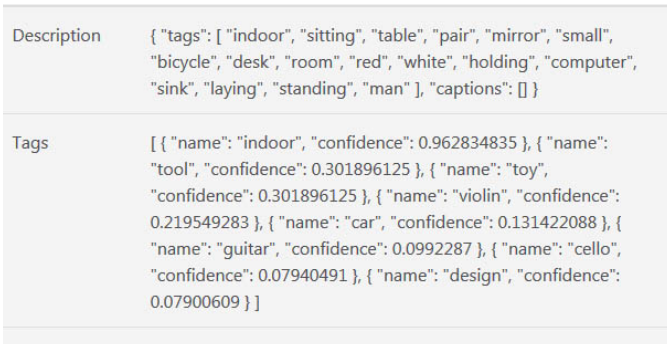
Image	
Object name	Wire Cutter/pliers
System result with process time (s)	Pliers / 58.27
Object distance (cm)	22
System distance (cm)	21.3
Object Height (cm)	20
System Height (cm)	23.27
Azure Cloud result	 Description: { "tags": ["indoor", "sitting", "table", "pair", "mirror", "small", "bicycle", "desk", "room", "red", "white", "holding", "computer", "sink", "laying", "standing", "man"], "captions": [] } Tags: [{ "name": "indoor", "confidence": 0.962834835 }, { "name": "tool", "confidence": 0.301896125 }, { "name": "toy", "confidence": 0.301896125 }, { "name": "violin", "confidence": 0.219549283 }, { "name": "car", "confidence": 0.131422088 }, { "name": "guitar", "confidence": 0.0992287 }, { "name": "cello", "confidence": 0.07940491 }, { "name": "design", "confidence": 0.07900609 }]

Table A.3: Result 10 - Ideal positioning

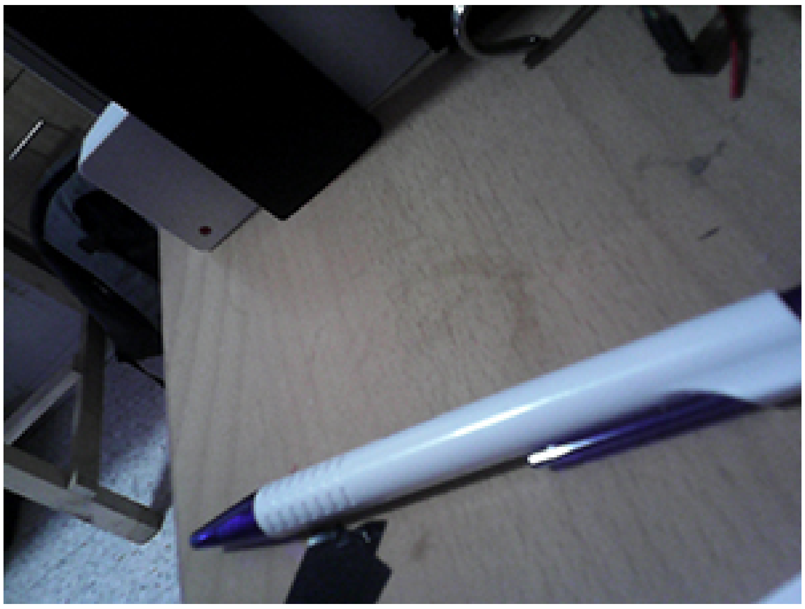
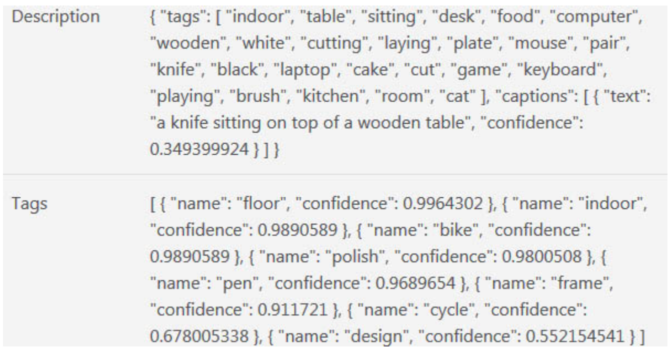
Image	
Object name	Pen
System result with process time (s)	No object detected
Object distance (cm)	Not Applicable
System distance (cm)	Not Applicable
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description { "tags": ["indoor", "table", "sitting", "desk", "food", "computer", "wooden", "white", "cutting", "laying", "plate", "mouse", "pair", "knife", "black", "laptop", "cake", "cut", "game", "keyboard", "playing", "brush", "kitchen", "room", "cat"], "captions": [{ "text": "a knife sitting on top of a wooden table", "confidence": 0.349399924 }] } Tags [{ "name": "floor", "confidence": 0.9964302 }, { "name": "indoor", "confidence": 0.9890589 }, { "name": "bike", "confidence": 0.9890589 }, { "name": "polish", "confidence": 0.9800508 }, { "name": "pen", "confidence": 0.9689654 }, { "name": "frame", "confidence": 0.911721 }, { "name": "cycle", "confidence": 0.678005338 }, { "name": "design", "confidence": 0.552154541 }]

Table A.4: Result 11 - Ideal positioning

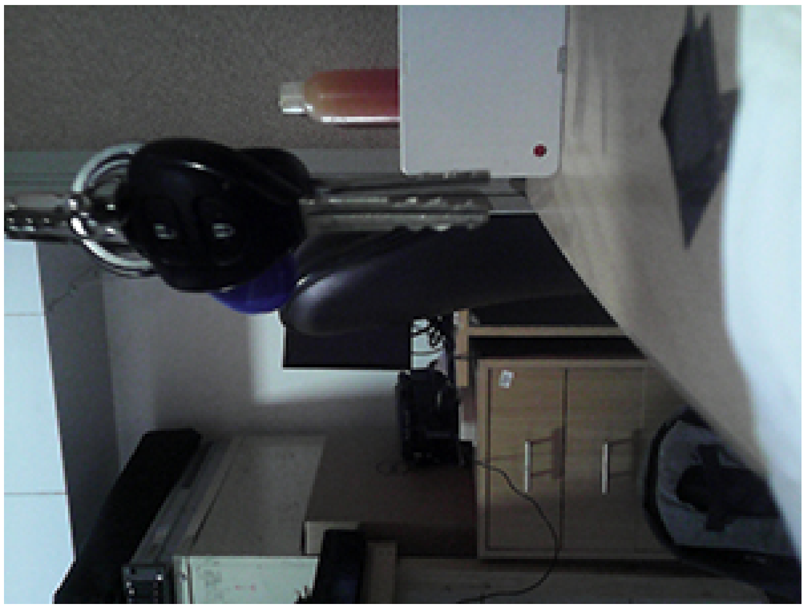
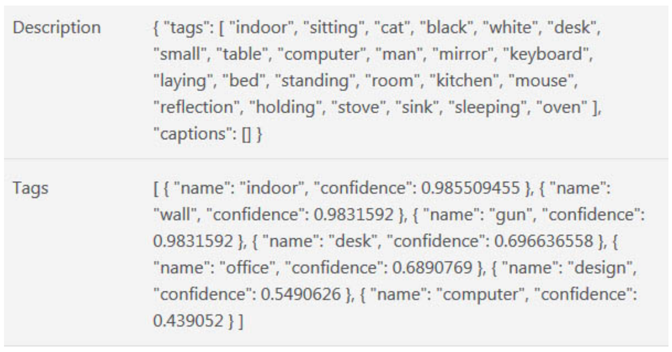
Image	
Object name	Keys
System result with process time (s)	Keys / 49.23
Object distance (cm)	20
System distance (cm)	18.9
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description: { "tags": ["indoor", "sitting", "cat", "black", "white", "desk", "small", "table", "computer", "man", "mirror", "keyboard", "laying", "bed", "standing", "room", "kitchen", "mouse", "reflection", "holding", "stove", "sink", "sleeping", "oven"], "captions": [] } Tags: [{ "name": "indoor", "confidence": 0.985509455 }, { "name": "wall", "confidence": 0.9831592 }, { "name": "gun", "confidence": 0.9831592 }, { "name": "desk", "confidence": 0.696636558 }, { "name": "office", "confidence": 0.6890769 }, { "name": "design", "confidence": 0.5490626 }, { "name": "computer", "confidence": 0.439052 }]

Table A.5: Result 12 - Ideal positioning

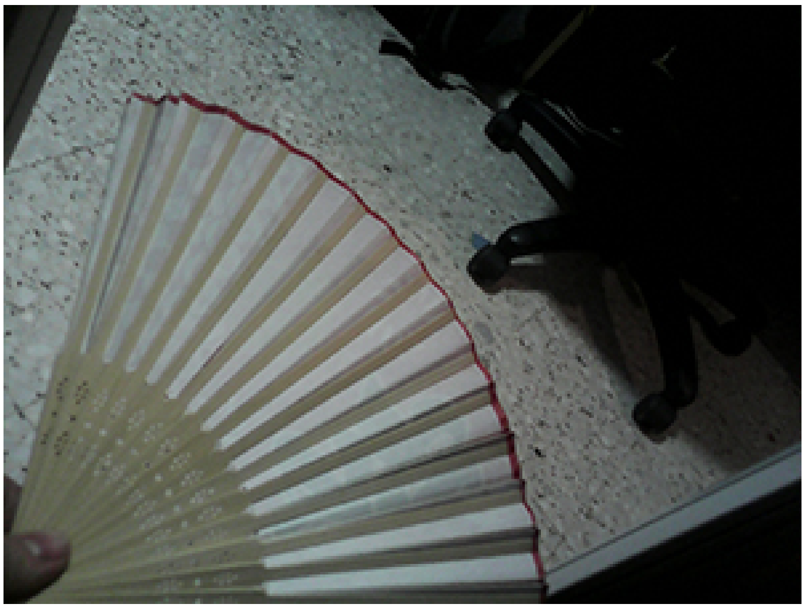
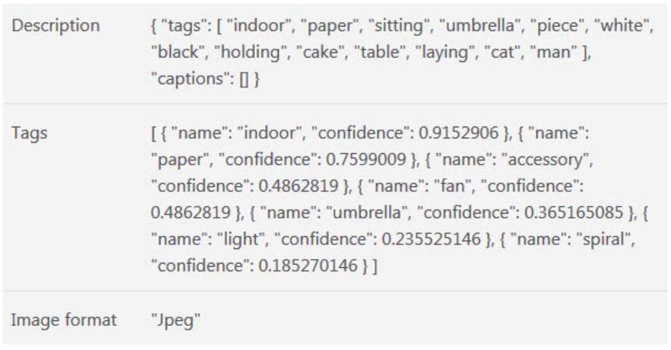
Image	
Object name	Paper fan
System result with process time (s)	Paper / 60.12
Object distance (cm)	30
System distance (cm)	24.1
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description { "tags": ["indoor", "paper", "sitting", "umbrella", "piece", "white", "black", "holding", "cake", "table", "laying", "cat", "man"], "captions": [] } Tags [{ "name": "indoor", "confidence": 0.9152906 }, { "name": "paper", "confidence": 0.7599009 }, { "name": "accessory", "confidence": 0.4862819 }, { "name": "fan", "confidence": 0.4862819 }, { "name": "umbrella", "confidence": 0.365165085 }, { "name": "light", "confidence": 0.235525146 }, { "name": "spiral", "confidence": 0.185270146 }] Image format "Jpeg"

Table A.6: Result 13 - Ideal positioning

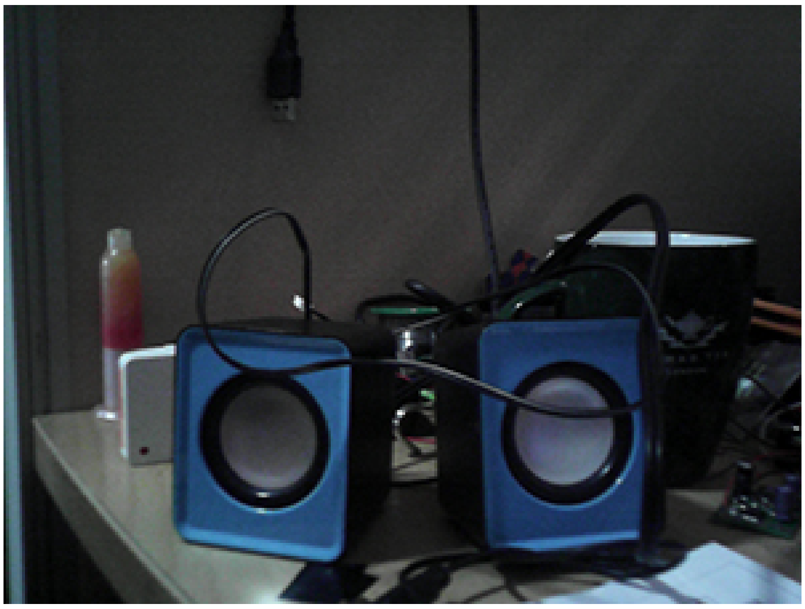
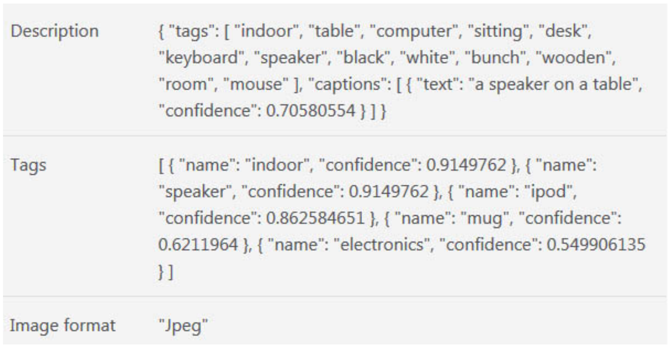
Image	
Object name	Speakers
System result with process time (s)	Speaker unit / 49.77
Object distance (cm)	15
System distance (cm)	14.2
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description: { "tags": ["indoor", "table", "computer", "sitting", "desk", "keyboard", "speaker", "black", "white", "bunch", "wooden", "room", "mouse"], "captions": [{ "text": "a speaker on a table", "confidence": 0.70580554 }] } Tags: [{ "name": "indoor", "confidence": 0.9149762 }, { "name": "speaker", "confidence": 0.9149762 }, { "name": "ipod", "confidence": 0.862584651 }, { "name": "mug", "confidence": 0.6211964 }, { "name": "electronics", "confidence": 0.549906135 }] Image format: "Jpeg"

Table A.7: Result 14 - Ideal positioning

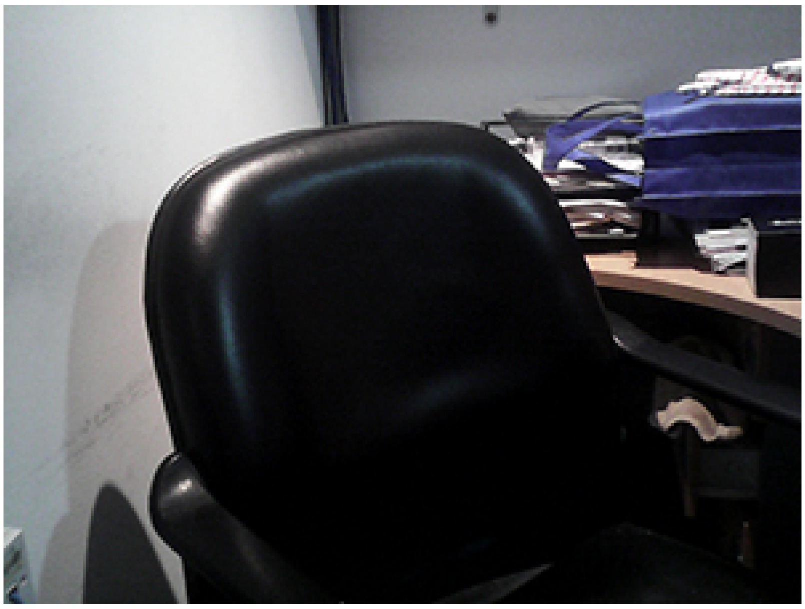
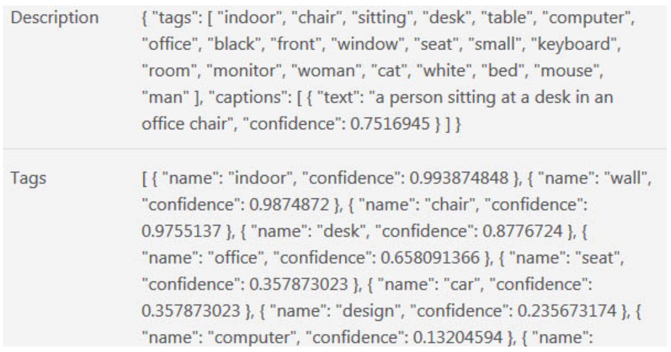
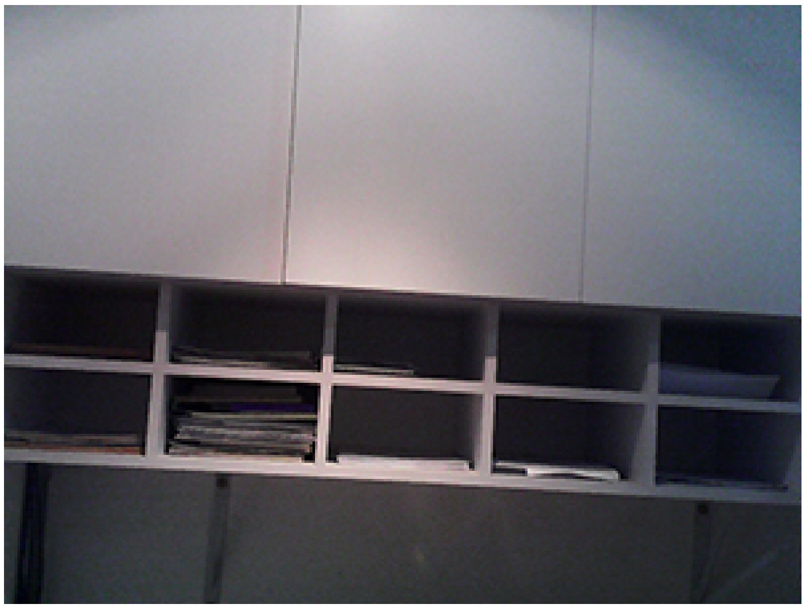
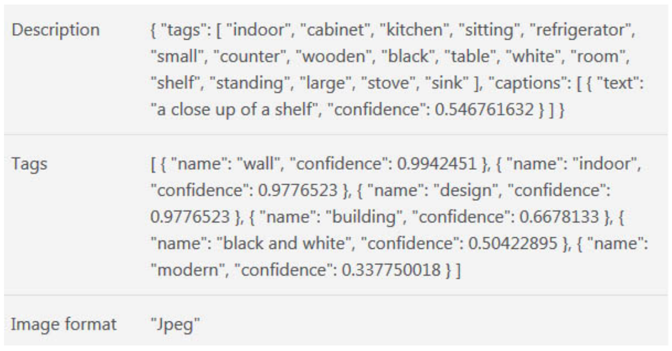
Image	
Object name	Chair
System result with process time (s)	Chair / 42.68
Object distance (cm)	50
System distance (cm)	49.7
Object Height (cm)	55
System Height (cm)	69.49
Azure Cloud result	 Description { "tags": ["indoor", "chair", "sitting", "desk", "table", "computer", "office", "black", "front", "window", "seat", "small", "keyboard", "room", "monitor", "woman", "cat", "white", "bed", "mouse", "man"], "captions": [{ "text": "a person sitting at a desk in an office chair", "confidence": 0.7516945 }] } Tags [{ "name": "indoor", "confidence": 0.993874848 }, { "name": "wall", "confidence": 0.9874872 }, { "name": "chair", "confidence": 0.9755137 }, { "name": "desk", "confidence": 0.8776724 }, { "name": "office", "confidence": 0.658091366 }, { "name": "seat", "confidence": 0.357873023 }, { "name": "car", "confidence": 0.357873023 }, { "name": "design", "confidence": 0.235673174 }, { "name": "computer", "confidence": 0.13204594 }, { "name":

Table A.8: Result 15 - Ideal positioning

Image	
Object name	shelf
System result with process time (s)	Shelf / 50.34
Object distance (cm)	65
System distance (cm)	63.7
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	

Description	{ "tags": ["indoor", "cabinet", "kitchen", "sitting", "refrigerator", "small", "counter", "wooden", "black", "table", "white", "room", "shelf", "standing", "large", "stove", "sink"], "captions": [{ "text": "a close up of a shelf", "confidence": 0.546761632 }] }
Tags	[{ "name": "wall", "confidence": 0.9942451 }, { "name": "indoor", "confidence": 0.9776523 }, { "name": "design", "confidence": 0.9776523 }, { "name": "building", "confidence": 0.6678133 }, { "name": "black and white", "confidence": 0.50422895 }, { "name": "modern", "confidence": 0.337750018 }]
Image format	"Jpeg"

Table A.9: Result 16 - Natural positioning

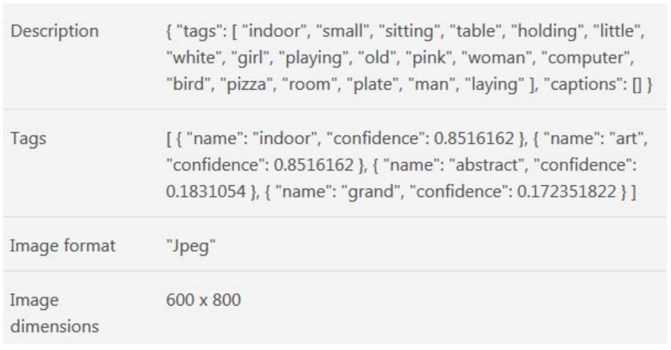
Image	
Object name	Map
System result with process time (s)	Painting / 47.19
Object distance (cm)	Not Applicable
System distance (cm)	Not Applicable
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description { "tags": ["indoor", "small", "sitting", "table", "holding", "little", "white", "girl", "playing", "old", "pink", "woman", "computer", "bird", "pizza", "room", "plate", "man", "laying"], "captions": [] } Tags [{ "name": "indoor", "confidence": 0.8516162 }, { "name": "art", "confidence": 0.8516162 }, { "name": "abstract", "confidence": 0.1831054 }, { "name": "grand", "confidence": 0.172351822 }] Image format "Jpeg" Image dimensions 600 x 800

Table A.10: Result 17 - Natural positioning

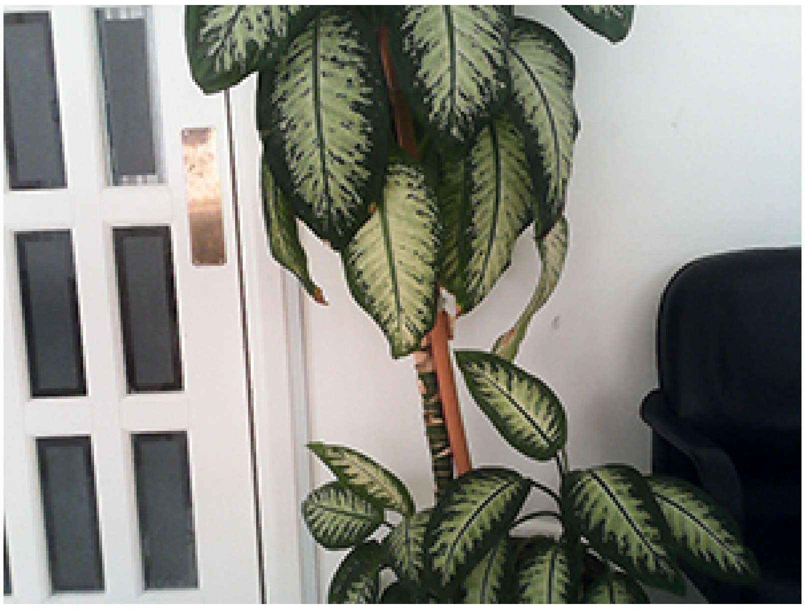
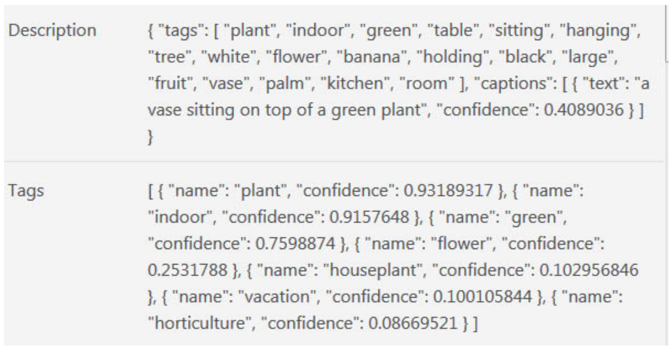
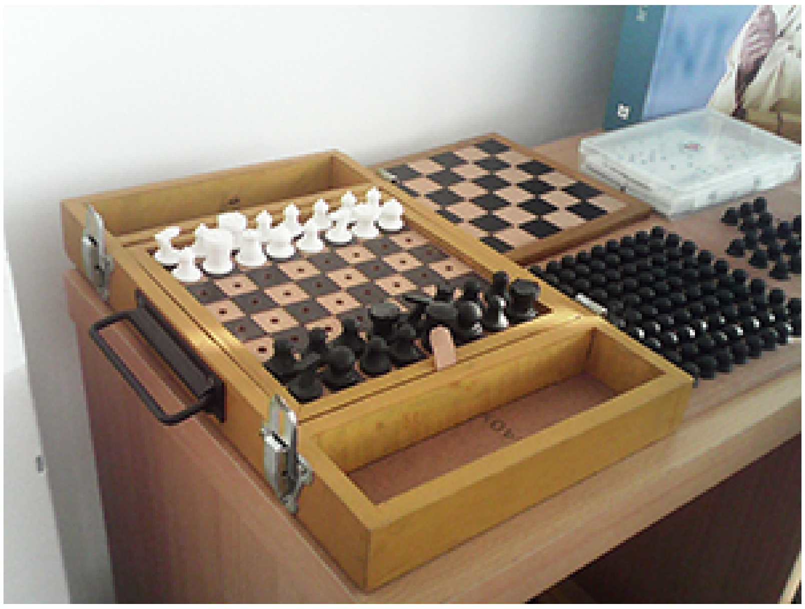
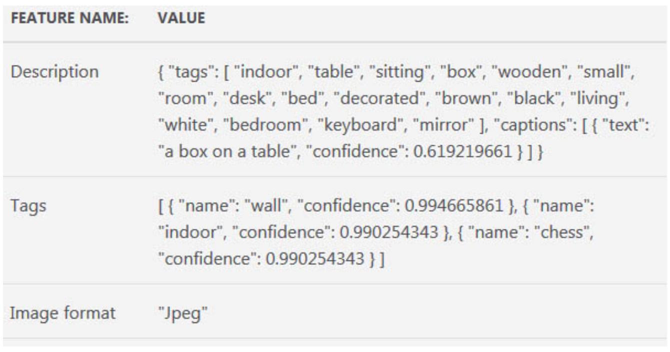
Image	
Object name	Plant/Greens/Tree
System result with process time (s)	Leaf / 62.35
Object distance (cm)	Not Applicable
System distance (cm)	Not Applicable
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description { "tags": ["plant", "indoor", "green", "table", "sitting", "hanging", "tree", "white", "flower", "banana", "holding", "black", "large", "fruit", "vase", "palm", "kitchen", "room"], "captions": [{ "text": "a vase sitting on top of a green plant", "confidence": 0.4089036 }] } Tags [{ "name": "plant", "confidence": 0.93189317 }, { "name": "indoor", "confidence": 0.9157648 }, { "name": "green", "confidence": 0.7598874 }, { "name": "flower", "confidence": 0.2531788 }, { "name": "houseplant", "confidence": 0.102956846 }, { "name": "vacation", "confidence": 0.100105844 }, { "name": "horticulture", "confidence": 0.08669521 }]

Table A.11: Result 18 - Natural positioning

Image	
Object name	Game boards
System result with process time (s)	Chess / 55.29
Object distance (cm)	Not Applicable
System distance (cm)	Not Applicable
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	

FEATURE NAME:	VALUE
Description	{ "tags": ["indoor", "table", "sitting", "box", "wooden", "small", "room", "desk", "bed", "decorated", "brown", "black", "living", "white", "bedroom", "keyboard", "mirror"], "captions": [{ "text": "a box on a table", "confidence": 0.619219661 }] }
Tags	[{ "name": "wall", "confidence": 0.994665861 }, { "name": "indoor", "confidence": 0.990254343 }, { "name": "chess", "confidence": 0.990254343 }]
Image format	"Jpeg"

Table A.12: Result 19 - Natural positioning

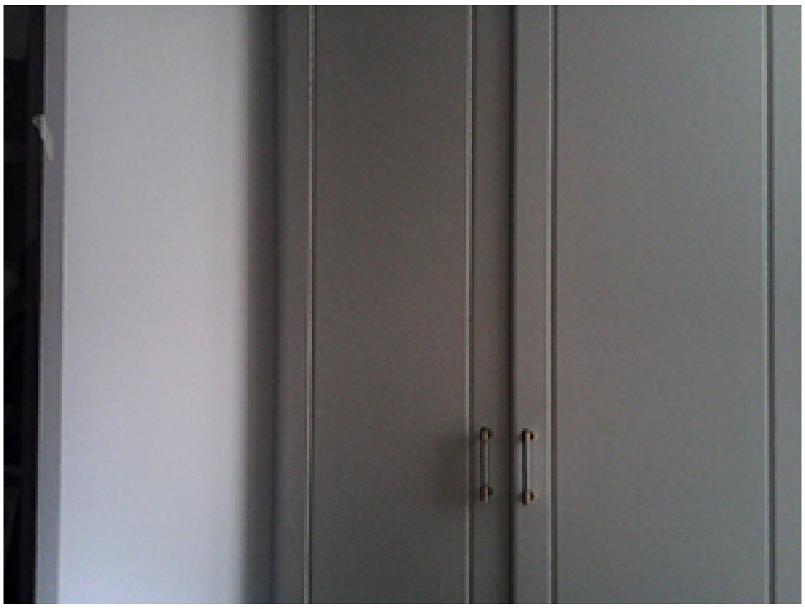
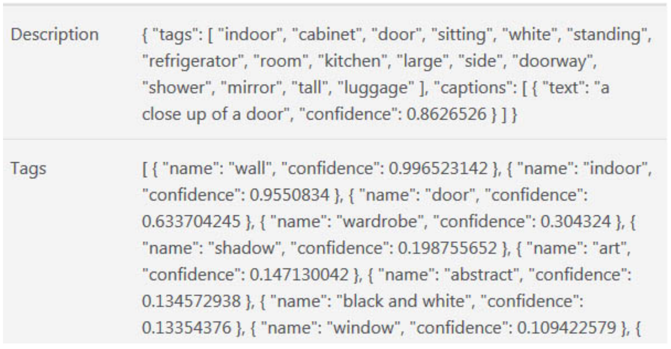
Image	
Object name	Wall Drawer
System result with process time (s)	Door / 50.82
Object distance (cm)	Not Applicable
System distance (cm)	Not Applicable
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 Description { "tags": ["indoor", "cabinet", "door", "sitting", "white", "standing", "refrigerator", "room", "kitchen", "large", "side", "doorway", "shower", "mirror", "tall", "luggage"], "captions": [{ "text": "a close up of a door", "confidence": 0.8626526 }] } Tags [{ "name": "wall", "confidence": 0.996523142 }, { "name": "indoor", "confidence": 0.9550834 }, { "name": "door", "confidence": 0.633704245 }, { "name": "wardrobe", "confidence": 0.304324 }, { "name": "shadow", "confidence": 0.198755652 }, { "name": "art", "confidence": 0.147130042 }, { "name": "abstract", "confidence": 0.134572938 }, { "name": "black and white", "confidence": 0.13354376 }, { "name": "window", "confidence": 0.109422579 }, {

Table A.13: Result 20 - Natural positioning

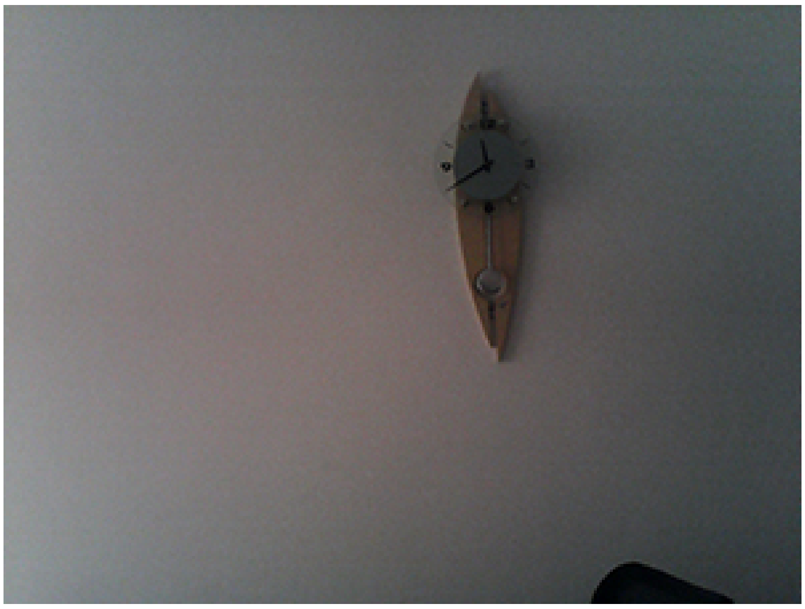
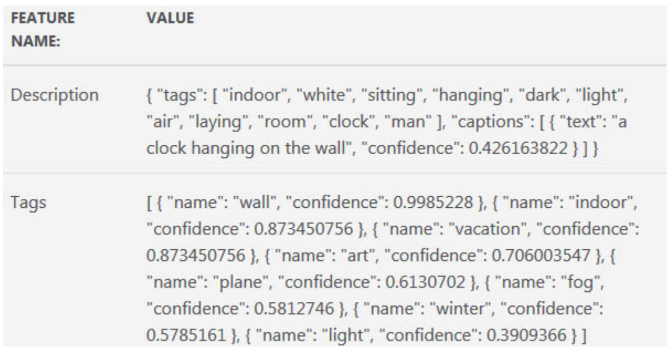
Image	
Object name	Wall Clock
System result with process time (s)	Clock / 49.94
Object distance (cm)	Not Applicable
System distance (cm)	Not Applicable
Object Height (cm)	Not Applicable
System Height (cm)	Not Applicable
Azure Cloud result	 FEATURE NAME: VALUE Description { "tags": ["indoor", "white", "sitting", "hanging", "dark", "light", "air", "laying", "room", "clock", "man"], "captions": [{ "text": "a clock hanging on the wall", "confidence": 0.426163822 }] } Tags [{ "name": "wall", "confidence": 0.9985228 }, { "name": "indoor", "confidence": 0.873450756 }, { "name": "vacation", "confidence": 0.873450756 }, { "name": "art", "confidence": 0.706003547 }, { "name": "plane", "confidence": 0.6130702 }, { "name": "fog", "confidence": 0.5812746 }, { "name": "winter", "confidence": 0.5785161 }, { "name": "light", "confidence": 0.3909366 }]

Table A.14: Result 21 - Natural positioning