GAZIANTEP UNIVERSITY

Visual Object Recognition for People Having Visual Impairment

EEE 499 GRADUATION PROJECT

IN

ELECTRICAL & ELECTRONICS ENGINEERING

Students

Arda ÜNAL

Halil İbrahim ŞİMŞEK

Hüseyin KILIÇ

Metehan KOZAN

Supervisor

Dr. Taner İNCE

JANUARY 2020

ABSTRACT

People with visual impairment face various problems in their daily life as the modern assistive devices are often not meeting the consumer requirements in term of price and level of assistance. This report is a new design of assistive smart glasses for visually impaired people. The objective is to assist in guidance the user when (s)he came across with an obstacle using the advantage of wearable design format. The project works on a raspberry pi which is integrated with a stereo camera module, and an object recognition is made to accomplish the objective of the project.

Keywords: Smart Glasses, Object Recognition, Stereo Camera Distance Measurement

ÖZET

Görme engelli insanlar günümüz modern cihazlarının yardımına rağmen çeşitli sorunlarla karşı karşıya kalmaktadırlar çünkü bu modern cihazlar fiyat performans açısından gereksinimleri karşılayamamaktadır. Bu rapor görme engelliler için yapılmış yeni bir akıllı bir gözlük tasarımıdır. Amaç giyilebilir gözlük formatının avantajlarını kullanarak bir engelle karşılaşıldığında kullanıcıya rehberlik etmektir. Proje bir stereo kamera modülü ile entegre edilmiş Raspberry Pi ile çalışıyor ve nesne tanıma işlevi içeriyor.

Anahtar Kelimeler: Akıllı Gözlük, Nesne Tanıma, Stereo Kamera Mesafe Ölçümü

ACKNOWLEDGEMENT

First, we would like to thank our supervisor Dr. Taner İNCE for his patience and guidance throughout the project.

In addition to this, we would like to thank all our teachers who contributed to us throughout our education life.

At last, we would like to thank the stuff of İkizgen Mühendislik in Teknopark for their contribution to our project.

TABLE OF CONTENTS

Page

ABSTRACT........ii

ÖZET........iii

ACKNOWLEDGEMENT........iv

LIST OF FIGURES........vii

1. INTRODUCTION........1

2. BACKGROUND SYSTEM........2

2.1 Artificial Intelligence........2

2.1.1 How Artificial Intelligence Works........3

2.2 Object Recognition........3

2.2.1 Object Detection........3

2.3 Python........4

2.4 Raspberry Pi........5

2.5 Distance Measurement with Stereo Camera........5

3. SOFTWARE........6

3.1 Object Recognition with HAAR Cascade........6

3.2 Distance Measurement........9

3.3 Text to Speech Conversion........13

3.4 Python........13

3.4.1 Advantages of Python........14

3.4.2 NumPy in Python........15

3.4.3 OpenCV in Python........15

3.5 The Code........16

4. HARDWARE........21

4.1 Raspberry Pi 3 Model B........21

4.2 Stereo Camera........22

5. DESIGN OF THE GLASSES........24

6. CONCLUSION……………………………………………………………………………….…25

REFERENCES...26

APPENDICES

A) COST ANALYSIS..27

B) GANTT CHART…………………………………………………………………………....27

LIST OF FIGURES

Page

Figure 2.1 Object Recognition………………………………...4

Figure 2.2 Object Detection……………………………………………………………………………4

Figure 3.1 HAAR Cascade Features………………………………………………………….…..6

Figure 3.2 Integral Images………..7

Figure 3.3 Cascade Classifier Train Scheme......……………………………………………………8

Figure 3.4 The Picture of Object Taken From Two Cameras.…………………………………..………..10

Figure 3.5 The Picture of The Object Taken with Left Camera……………………………….….…...10

Figure 3.6 The Picture of The Object Taken with Right Camera…………………………….……......11

Figure 4.1 Raspberry Pi 3 Model B……………………………………………………………..22

Figure 4.2 Synchronized Dual Lens Stereo USB Camera………………………………...…………..23

Figure 5.1 Top View of the Glasses……………………………………………………………….….24

Figure 5.2 Front View of the Glasses……………………………………………………………....24

1. INTRODUCTION

Visual impairment can limit people's ability to perform everyday tasks and can affect their quality of life and ability to interact with the surrounding world. Blindness, the most severe form of visual impairment, can reduce people's ability to perform daily tasks, and move about unaided. The number of visually impaired people is growing over the past decades. As reported by the world health organization (WHO), about 285 million people worldwide are estimated to be visually impaired. However, until now many schools and jobs cannot accommodate them mainly due to lack of assistive technologies and economic barriers. As a result, 90 % of them still live in low level of income. The biggest challenge for a blind person, especially the one with the complete loss of vision, is to navigate around places. Commercial places can be made easily accessible for the blinds with tactile tiles. But, unfortunately, this is not done in most of the places. This creates a big problem for blind people who might want to visit the place. Even when the new aids or technologies become available, they are either too expensive ($3000and above), or affordable ($200) but with single or limited task functions only. Among all assistive devices, wearable devices are found to be the most useful because they are hand free or require minimum use of hands. The most popular types head mounted device. Their main advantage is that the device points naturally at the viewing direction, thus eliminates the need of additional direction instructions, unlike other devices. This report presents a new design of smart glasses that can provide assistance in warning the user for the obstacles on their way before any collision. The design uses a raspberry pi single board computer, a stereo camera inserted into 3D printed glasses, a battery pack, and an earphone to convey information to the user. Report contains all the background information to be needed and the detailed analysis of software, hardware, and design of the project.

2. PROJECT BACKGROUND

2.1 Artificial Intelligence

Artificial Intelligence is a branch of computer science concerned with the study and creation of computer systems that exhibit some form of intelligence systems that learn new concepts and tasks systems that can reason and draw useful conclusions about the world around us, systems that can understand a natural language or perceive and comprehend a visual scene, and systems that perform other types of feats that require human types of intelligence.

Artificial Intelligence is not the study and creation of conventional computer systems. Even though one can argue that all programs exhibit some degree of intelligence, an Artificial Intelligence program will go beyond this in demonstrating a high level of intelligence to a degree that equals or exceeds the intelligence required of a human in performing some task. Artificial Intelligence is not the study of the mind, nor of the body, nor of languages, as customarily found in the fields of psychology, physiology, cognitive science, or linguistics. To be sure, there is some overlap between these fields and Artificial Intelligence. All seek a better understanding of the human's intelligence and sensing processes. But in Artificial Intelligence the goal is to develop working computer systems that are truly capable of performing tasks that require high levels of intelligence. The programs are not necessarily meant to imitate human senses and thought processes. Indeed, in performing some tasks differently, they may actually exceed human abilities. The important point is that the systems all be capable of performing intelligent tasks effectively and efficiently.

Finally, a better understanding of Artificial Intelligence is gained by looking at the component areas of study that make up the whole. These include such topics as robotics, memory organization, knowledge representation, storage and recall, learning models, inference techniques, commonsense reasoning, dealing with uncertainty in reasoning and decision making, understanding natural language, pattern recognition and machine vision methods, search and matching, speech recognition and synthesis, and a variety of Artificial Intelligence tools. [1]

2.1.1 How Artificial Intelligence Works

At the heart of our framework is the observation that robotics and current practice in Artificial Intelligence are continuing what other automation technologies have done in the past using machines and computers to substitute for human labor in a widening range of tasks and industrial processes.

Production in most industries requires the simultaneous completion of a range of tasks. For example, textile production requires production of fiber, production of yarn from fiber (e.g., by spinning), production of the relevant fabric from the yarn (e.g., by weaving or knitting), preliminary treatment (e.g., cleaning of the fabric, scouring, mercerizing and bleaching), dyeing and printing, finishing, as well as various auxiliary tasks including design, planning, marketing, transport, and retail [2]. Each one of these tasks can be performed by a combination of human labor and machines. At the dawn of the British Industrial Revolution, most of these tasks were heavily labor-intensive (some of them were merely performed). Many of the early innovations of that era were aimed at automating spinning and weaving by substituting mechanized processes for the labor of skilled artisans (Mantoux, 1928).

2.2 Object Recognition

Object recognition is a virtual vision type for identifying objects in images or videos. Object recognition is an important outcome of deep learning and machine learning algorithms. When people look at a photograph or watch a video, they can identify objects and visual details readily. Purpose of object recognition is to teach a computer to what comes to mind naturally for example to gain a level of understanding of what an image contains. This method of virtual vision works as seen in Figure 2.1

2.2.1 Object Detection

Actually, object recognition and object detection are similar virtual vision type for identifying objects, but just different is implementation. Object detection is the process of finding instances of objects in images. In the case of deep learning, object detection is a subset of object recognition, where the object is not only identified but also located in an image. This allows for multiple objects to be identified and located within the same image. This method of virtual vision works as seen in Figure 2.2

Fig. 2.1 Object Recognition

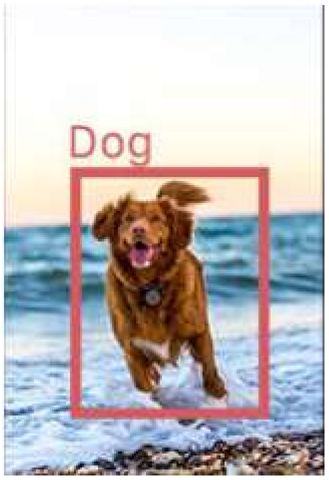


Fig. 2.2 Object Detection

2.3 Python

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third-party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications [3].

2.4. Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

2.5 Distance Measuring Based on Stereoscopic Pictures

Stereoscopy is a technique used for recording and representing stereoscopic (3D) images. It can create an illusion of depth using two pictures taken at slightly different positions. There are two possible way of taking stereoscopic pictures: by using special two-lens stereo cameras or systems with two single-lens cameras joined together. Stereoscopic pictures allow us to calculate the distance from the camera(s) to the chosen object within the picture. The distance is calculated from differences between the pictures and additional technical data like focal length and distance between the cameras. The certain object is selected on the left picture, while the same object on the right picture is automatically detected by means of optimization algorithm which searches for minimum difference between both pictures. The calculation of object's position can be calculated by doing some geometrical derivations. The accuracy of the position depends on picture resolution, optical distortions and distance between the cameras. The results showed that the calculated distance to the subject is relatively accurate.

Methods for measuring the distance to some objects can be divided into active and passive ones. The active methods are measuring the distance by sending some signals to the object [4, 5] (e.g. laser beam, radio signals, ultra-sound, etc.) while passive ones only receive information about object's position (usually by light). Among passive ones, the most popular are those relying on stereoscopic measuring method. The main characteristic of the method is to use two cameras. The object's distance can be calculated from relative difference of object's position on both cameras [6]. [7]

3. SOFTWARE

3.1. Object Recognition with HAAR Cascade

It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

The algorithm has four stages:

1- Haar Feature Selection
2- Creating Integral Images
3- Adaboost Training
4- Cascading Classifiers

It is well known for being able to detect faces and body parts in an image but can be trained to identify almost any object.

Let’s take face detection as an example. Initially, the algorithm needs a lot of positive images of faces and negative images without faces to train the classifier. Then we need to extract features from it.

First step is to collect the Haar Features. A Haar feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.

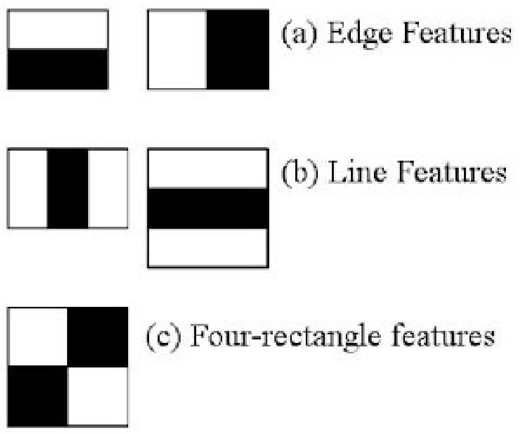


Fig. 3.1 HAAR Cascade Features

Integral images are used to make this super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider in Figure 3.2. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant.

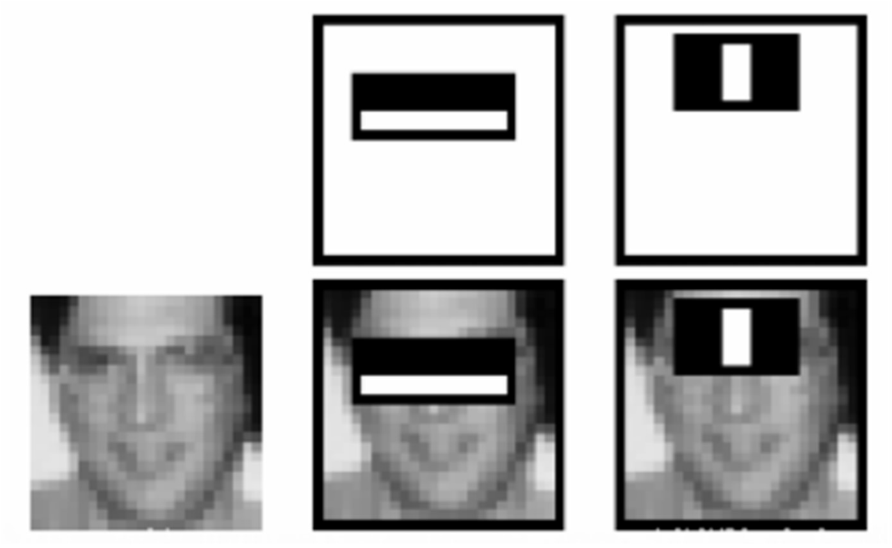


Fig. 3.2 Integral Images

So how do we select the best features out of 160000+ features? This is accomplished using a concept called Adaboost which both selects the best features and trains the classifiers that use them. This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers. The process is as follows.

During the detection phase, a window of the target size is moved over the input image, and for each subsection of the image and Haar features are calculated. You can see this in action in the video below. This difference is then compared to a learned threshold that separates non-objects from objects. Because each Haar feature is only a "weak classifier" (its detection quality is slightly better than random guessing) a large number of Haar features are necessary to describe an object with sufficient accuracy and are therefore organized into cascade classifiers to form a strong classifier [8].

Cascade Classifier

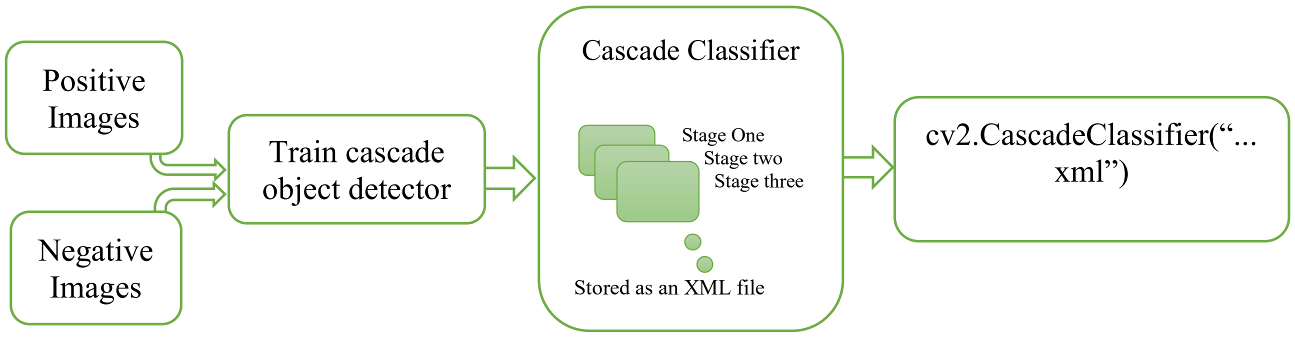


Fig. 3.3 Cascade Classifier Train Scheme

The cascade classifier consists of a collection of stages, where each stage is an ensemble of weak learners. The weak learners are simple classifiers called decision stumps. Each stage is trained using a technique called boosting. Boosting provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. Positive indicates that an object was found and negative indicates no objects were found. If the label is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as positive. This training work as seen in figure 3.3.

The stages are designed to reject negative samples as fast as possible. The assumption is that the vast majority of windows do not contain the object of interest. Conversely, true positives are rare and worth taking the time to verify.

- A true positive occurs when a positive sample is correctly classified.

- A false positive occurs when a negative sample is mistakenly classified as positive.
- A false negative occurs when a positive sample is mistakenly classified as negative.

To work well, each stage in the cascade must have a low false negative rate. If a stage incorrectly labels an object as negative, the classification stops, and you cannot correct the mistake. However, each stage can have a high false positive rate. Even if the detector incorrectly labels a nonobject as positive, you can correct the mistake in subsequent stages. Adding more stages reduces the overall false positive rate, but it also reduces the overall true positive rate.

Cascade classifier training requires a set of positive samples and a set of negative images. You must provide a set of positive images with regions of interest specified to be used as positive samples. You can use the Image Labeler to label objects of interest with bounding boxes. The Image Labeler outputs a table to use for positive samples. You also must provide a set of negative images from which the function generates negative samples automatically. To achieve acceptable detector accuracy, set the number of stages, feature type, and other function parameters. [9]

3.2. Distance Measurement

In our project, we use stereoscopic technique for measuring distances. There is a stereo camera on a board. While the object recognizes with camera, object is framed by square automatically. After object framed by square, we can find out its left corner coordinates, its width and height respectively. With using this information, we can detect the object's coordinates from left and right camera. We create a new algorithm because of our cameras has one frame in the computer. It means, our cameras detected by computer as one camera. Since our camera has one frame for two cameras, we just use a basic algorithm to find out its distance from the camera. Method of measuring distance shown below:

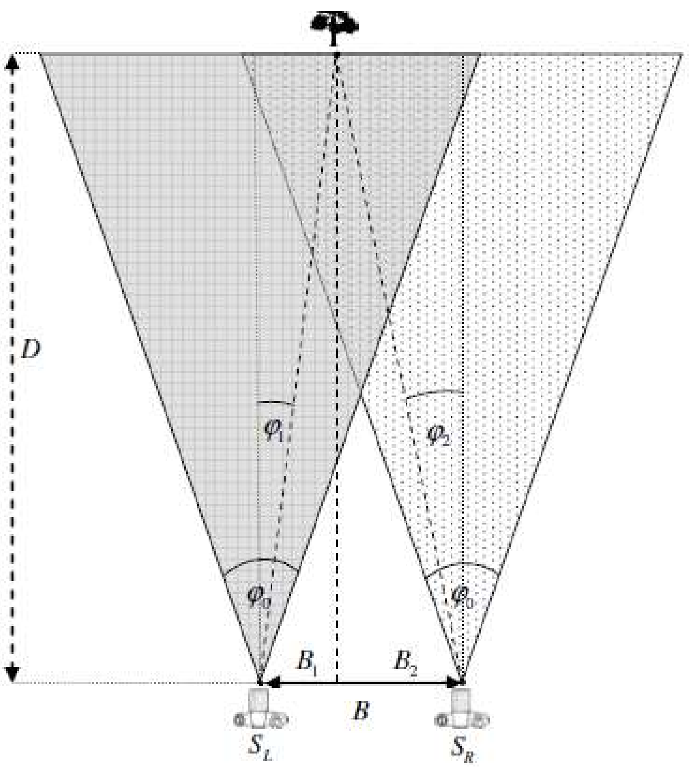


Fig. 3.4 The Picture of Object Taken from Two Cameras

$$B = B_1 + B_2 = D\tan\varphi_1 + D\tan\varphi_2 \qquad D = \frac{B}{\tan\varphi_1 + \tan\varphi_2}$$

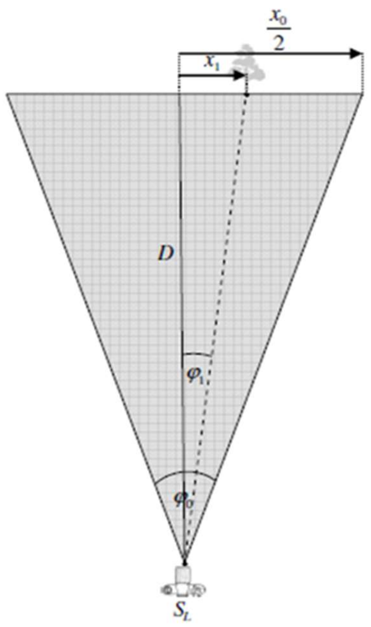


Figure 3.5 The Picture of The Object Taken with Left Camera

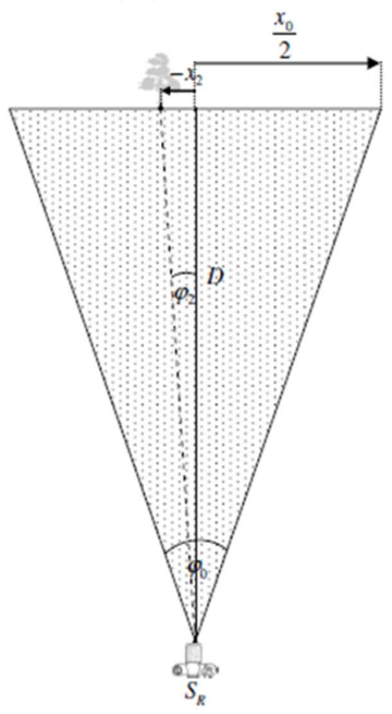


Figure 3.6 The Picture of The Object Taken with Right Camera

Using figure 3.5 and figure 3.6 and basic geometry, we find:

$$\frac{x_1}{\frac{x_0}{2}} = \frac{\tan\varphi_1}{\tan\left(\frac{\varphi_0}{2}\right)} \qquad \frac{-x_2}{\frac{x_0}{2}} = \frac{\tan\varphi_2}{\tan\left(\frac{\varphi_0}{2}\right)}$$

Distance can be calculated as follows:

$$D = \frac{Bx_0}{2\tan\left(\frac{\varphi_0}{2}\right)(x_1 - x_2)}$$

We developed a real time distance measuring method which is based on this fundamental method. When an object detects by object recognition algorithm, there is an array of coordinates of all detected objects. Array consists of arrays of individual recognized object. There are examples of code which is give coordinates of recognized object:

```
yuzler = yuz_casc.detectMultiScale(kare,1.1,5)
```

In our codes if detected objects' number is smaller than 3, we use this algorithm shown below:

```
if len(yuzler) < 3 and len(yuzler) > 1:
    first_array = yuzler[0]
    second_array = yuzler[1]
    if first_array[0] < 320:
        xbir = first_array[0] + first_array[2]/2 - t/2
        xiki = second_array[0] + second_array[2]/2 - t/2 - 320
    else:
        xbir = first_array[0] + first_array[2]/2 - t/2 - 320
        xiki = second_array[0] + second_array[2] / 2 - t / 2
    distance = abs( (b * t) / (2 * math.tan(pi * angle / 360) * (xbir - xiki)))
    mesafe_yuz = distance
```

If the recognized object's number is higher than 3, this code requires an improvement. If there are more than 3 objects in the frame, array consist of arrays of all object. In these arrays, we have x and y coordinates of left corner of square which is belong to left camera frame, but there are also another coordinates of square which is belong to right camera frame, so we try to solve this problem to using their x coordinates to find which coordinates belong same object. After we find object's x coordinates in the frame, we use this algorithm for measuring distance and find out of coordinates of different objects shown below:

```
elif len(yuzler)>2:
    k=0
    j=0
    x1=0
    x2 = 0
    for k in range(len(yuzler)):
        for j in range(len(yuzler)):
            if abs(yuzler[k][0]-yuzler[j][0])<300 and abs(yuzler[k][0]-yuzler[j][0])>280:
                x1 = yuzler[k][0]
                if x1 != x2:
                    print("x1 = ", yuzler[k][0])
                    print("x2 = ", yuzler[j][0])
                    x2 = yuzler[j][0]
                    if x1 < 320:
```

```
        xbir = x1 + yuzler[k][2]/2 - t/2
        xiki = x2 + yuzler[j][2]/2 - t/2 - 320
    else:
        xbir = x1 + yuzler[k][2]/2 - t/2 - 320
        xiki = x2 + yuzler[j][2] / 2 - t / 2
    distance = abs( (b * t) / (2 * math.tan(pi * angle / 360) * (xbir - xiki)))
    if mesafe_yuz > distance:
        mesafe_yuz = distance
```

3.3. Text to Speech Conversion

```
import vlc
import time
import os

folder ="/home/oem/Desktop/sesler/"
file_list = os.listdir(folder)
def ses(name):
    for file_name in file_list:
        if file_name == name + ".mpeg":
            player = vlc.MediaPlayer(folder + name + ".mpeg")
            player.play()
            time.sleep(2)
```

As shown above, the code for text to speech conversion is using some libraries to perform the operation. Those libraries are vlc which is to get a sound output and time for adjusting the duration of the sound. In the aftercoming parts of the code there is matching process between the name the of the object which is detected by object detection code and the names of sound files that we had put into a file after recording the pronunciations of the objects and naming them as the object names. When two names are equal the pronunciation of the object is played.

3.4. Python

Throughout the project Python has been our main programming language, and in this section advantages of Python and the libraries that are used the most for the project (i.e. NumPy, OpenCV) will be mentioned. Lastly, the code that is developed for the project will be present.

3.4.1 Advantages of Python

The diverse application of the Python language is a result of the combination of features which give this language an edge over others. Some of the benefits of programming in Python include:

1. Presence of Third-Party Modules:

The Python Package Index (PyPI) contains numerous third-party modules that make Python capable of interacting with most of the other languages and platforms.

2. Extensive Support Libraries:

Python provides a large standard library which includes areas like internet protocols, string operations, web services tools and operating system interfaces. Many high use programming tasks have already been scripted into the standard library which reduces length of code to be written significantly.

3. Open Source and Community Development:

Python language is developed under an OSI-approved open source license, which makes it free to use and distribute, including for commercial purposes.

Further, its development is driven by the community which collaborates for its code through hosting conferences and mailing lists and provides for its numerous modules.

4. Learning Ease and Support Available:

Python offers excellent readability and uncluttered simple-to-learn syntax which helps beginners to utilize this programming language. The code style guidelines, PEP 8, provide a set of rules to facilitate the formatting of code. Additionally, the wide base of users and active developers has resulted in a rich internet resource bank to encourage development and the continued adoption of the language.

5. User-friendly Data Structures:

Python has built-in list and dictionary data structures which can be used to construct fast runtime data structures. Further, Python also provides the option of dynamic high-level data typing which reduces the length of support code that is needed.

6. Productivity and Speed:

Python has clean object-oriented design, provides enhanced process control capabilities, and possesses strong integration and text processing capabilities and its own unit testing framework, all of which contribute to the increase in its speed and productivity. Python is considered a viable option for building complex multi-protocol network applications [10].

3.4.2 NumPy in Python

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases [11].

3.4.3 OpenCV in Python

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general-purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of NumPy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy such as SciPy and Matplotlib [12].

3.5 The Code

Here are all the codes that is written for this project all in one place

```
from cv2 import cv2
import numpy as np
import math
import vlc
import time
import os
#def distance():

b = 7 #cm
angle = 85 #degree
t = 320 #pixel
pi = 3.1415926

folder ="/home/oem/Desktop/sesler/"
file_list = os.listdir(folder)
def ses(name):
    for file_name in file_list:
        if file_name == name + ".mpeg":
            player = vlc.MediaPlayer(folder + name + ".mpeg")
            player.play()
            time.sleep(2)

kamera = cv2.VideoCapture(2)

while True:
    ret,kare = kamera.read()
    #gri_kare = cv2.cvtColor(kare,cv2.COLOR_BGR2GRAY)
    yuz_casc = cv2.CascadeClassifier("yuz.xml")
```

```
car_casc = cv2.CascadeClassifier("cars.xml")
tree_casc = cv2.CascadeClassifier("tree.xml")
sira_casc = cv2.CascadeClassifier("sira.xml")
basamak_casc = cv2.CascadeClassifier("basamak.xml")

yuzler = yuz_casc.detectMultiScale(kare,1.1,5)
car = car_casc.detectMultiScale(kare,1.1,5)
tree = tree_casc.detectMultiScale(kare,1.1,5)
sira = sira_casc.detectMultiScale(kare,1.1,5)
basamak = basamak_casc.detectMultiScale(kare,1.1,5)

mesafe_basamak = 500
mesafe_yuz = 500
mesafe_sira = 500
mesafe_tree = 500
mesafe_car = 500

for (x,y,w,h) in yuzler:
    cv2.rectangle(kare,(x,y),(x+w,y+h),(255,0,0),2)
#for (x,y,w,h) in car:
#    cv2.rectangle(kare, (x, y), (x + w, y + h), (255, 0, 0), 2)
cv2.imshow("Yuz Tanıma",kare)

#Yuzler
if len(yuzler) < 3 and len(yuzler) > 1:
    first_array = yuzler[0]
    second_array = yuzler[1]
    if first_array[0] < 320:
        xbir = first_array[0] + first_array[2]/2 - t/2
        xiki = second_array[0] + second_array[2]/2 - t/2 - 320
    else:
        xbir = first_array[0] + first_array[2]/2 - t/2 - 320
        xiki = second_array[0] + second_array[2] / 2 - t / 2
    distance = abs( (b * t) / (2 * math.tan(pi * angle / 360) * (xbir - xiki)))
    mesafe_yuz = distance

elif len(yuzler)>2:
    k=0
    j=0
    x1=0
    x2 = 0
    for k in range(len(yuzler)):
        for j in range(len(yuzler)):
            if abs(yuzler[k][0]-yuzler[j][0])<300 and abs(yuzler[k][0]-yuzler[j][0])>280:
                x1 = yuzler[k][0]
                if x1 != x2:
                    print("x1 = ", yuzler[k][0])
                    print("x2 = ", yuzler[j][0])
                    x2 = yuzler[j][0]
                    if x1 < 320:
                        xbir = x1 + yuzler[k][2]/2 - t/2
                        xiki = x2 + yuzler[j][2]/2 - t/2 - 320
                    else:
                        xbir = x1 + yuzler[k][2]/2 - t/2 - 320
                        xiki = x2 + yuzler[j][2] / 2 - t / 2
                    distance = abs( (b * t) / (2 * math.tan(pi * angle / 360) * (xbir - xiki)))
                    if mesafe_yuz > distance:
                        mesafe_yuz = distance
#Araba
if len(car) < 3 and len(car) > 1:
    first_array = car[0]
    second_array = car[1]
    if first_array[0] < 320:
        xbir = first_array[0] + first_array[2]/2 - t/2
        xiki = second_array[0] + second_array[2]/2 - t/2 - 320
    else:
        xbir = first_array[0] + first_array[2]/2 - t/2 - 320
        xiki = second_array[0] + second_array[2] / 2 - t / 2
    distance = abs( (b * t) / (2 * math.tan(pi * angle / 360) * (xbir - xiki)))
    mesafe_car = distance
elif len(car)>2:
    k=0
    j=0
    x1 = 0
    x2 = 0
    for k in range(len(car)):
        for j in range(len(car)):
            if abs(car[k][0]-car[j][0])<300 and abs(car[k][0]-car[j][0])>280:
                x1 = car[k][0]
                if x1 != x2:
                    print("x1 = ", car[k][0])
                    print("x2 = ", car[j][0])
                    x2 = car[j][0]
                    if x1 < 320:
                        xbir = x1 + car[k][2]/2 - t/2
                        xiki = x2 + car[j][2]/2 - t/2 - 320
                    else:
                        xbir = x1 + car[k][2]/2 - t/2 - 320
                        xiki = x2 + car[j][2] / 2 - t / 2
                    distance = abs( (b * t) / (2 * math.tan(pi * angle / 360) * (xbir - xiki)))
                    if mesafe_car > distance:
                        mesafe_car = distance
#Agac
if len(tree) < 3 and len(tree) > 1:
        first_array = tree[0]
        second_array = tree[1]
        if first_array[0] < 320:
            xbir = first_array[0] + first_array[2]/2 - t/2
            xiki = second_array[0] + second_array[2]/2 - t/2 - 320

    elif len(tree)>2:
        k=0
        j=0
        x1=0
        x2 = 0
        for k in range(len(tree)):
            for j in range(len(tree)):
                if abs(tree[k][0]-tree[j][0])<300 and abs(tree[k][0]-tree[j][0])>280:
                    x1 = tree[k][0]
                    if x1 != x2:
                        print("x1 = ", tree[k][0])
                        print("x2 = ", tree[j][0])
                        x2 = tree[j][0]
                        x1 < 320:
                            xbir = x1 + tree[k][2]/2 - t/2
                            xiki = x2 + tree[j][2]/2 - t/2 - 320
    #Sira
    if len(sira) < 3 and len(sira) > 1:
        first_array = sira[0]
        second_array = sira[1]
        if first_array[0] < 320:
            xbir = first_array[0] + first_array[2]/2 - t/2
            xiki = second_array[0] + second_array[2]/2 - t/2 - 320
        else:
            xbir = first_array[0] + first_array[2]/2 - t/2 - 320
            xiki = second_array[0] + second_array[2] / 2 - t / 2
        distance = abs( (b * t) / (2 * math.tan(pi * angle / 360) * (xbir - xiki)))
        mesafe_sira = distance
    elif len(sira)>2:
        k=0
        j=0
        x1=0
        x2 = 0
        for k in range(len(sira)):
            for j in range(len(sira)):
                if abs(sira[k][0]-sira[j][0])<300 and abs(sira[k][0]-sira[j][0])>280:
                    x1 = sira[k][0]
                    if x1 != x2:
                        print("x1 = ", sira[k][0])
                        print("x2 = ", sira[j][0])
                        x2 = sira[j][0]
                        if x1 < 320:
                            xbir = x1 + sira[k][2]/2 - t/2
                            xiki = x2 + sira[j][2]/2 - t/2 - 320

    #Closest distance
    if mesafe_yuz < mesafe_tree and mesafe_yuz < mesafe_sira and mesafe_yuz < mesafe_car and
mesafe_yuz < mesafe_basamak:
        print(mesafe_yuz)
        ses("yuz")
    if mesafe_tree < mesafe_yuz and mesafe_tree < mesafe_sira and mesafe_tree < mesafe_car
and mesafe_tree < mesafe_basamak:
        print(mesafe_tree)
        ses("tree")
    if mesafe_sira < mesafe_tree and mesafe_sira < mesafe_yuz and mesafe_sira < mesafe_car and
mesafe_sira < mesafe_basamak:
        print(mesafe_sira)
        ses("sira")
    if mesafe_car < mesafe_tree and mesafe_car < mesafe_sira and mesafe_car < mesafe_yuz and
mesafe_car < mesafe_basamak:
        print(mesafe_car)
        ses("car")
    if mesafe_basamak < mesafe_tree and mesafe_basamak < mesafe_sira and mesafe_basamak <
mesafe_car and mesafe_basamak < mesafe_yuz:
        print(mesafe_basamak)
        ses("basamak")

    if cv2.waitKey(1000) & 0xFF == ord("q"):
        break
kamera.release()
cv2.destroyAllWindows()
```

4. HARDWARE

The project’s hardware consists of two parts which are a Raspberry Pi 3 Model B and a stereo camera module.

4.1 Raspberry Pi 3 Model B

Raspberry Pi 3 Model B is the first module of the third-generation Raspberry Pi with Bluetooth 4.1 and Wi-Fi. It is the most famous powerful pocket SBC (single board computer) around the world. You can use this credit card sized computer to process documents, control robots, play large games, and watch HD videos...

Such a petite body has such huge power, mainly thanks to its powerful Quad Core Broadcom BCM2837 64bit CPU. Which upgraded the processor speed from the Raspberry Pi 2 900Mhz to Pi 3 1.2Ghz.

Another exciting upgrade is the Pi 3 B adds an onboard Bluetooth module and an onboard Wi-Fi module. The powerful wireless connectivity allows the Raspberry Pi to have a wider range of applications without the need for a wireless expansion module.

Meanwhile, Raspberry Pi 3B strengthens the power management system. Now the power source supplied by the Micro USB has increased to 2.5A. Which means it can drive more peripherals, such as a mouse, a keyboard, a speaker and more.

On top of all that, Raspberry Pi 3B's 40Pin GPIO Header is fully compatible with Raspberry Pi 2. You can use all the existing Raspberry Pi 2 accessories and kits with the new Raspberry Pi 3 Model B. [13]

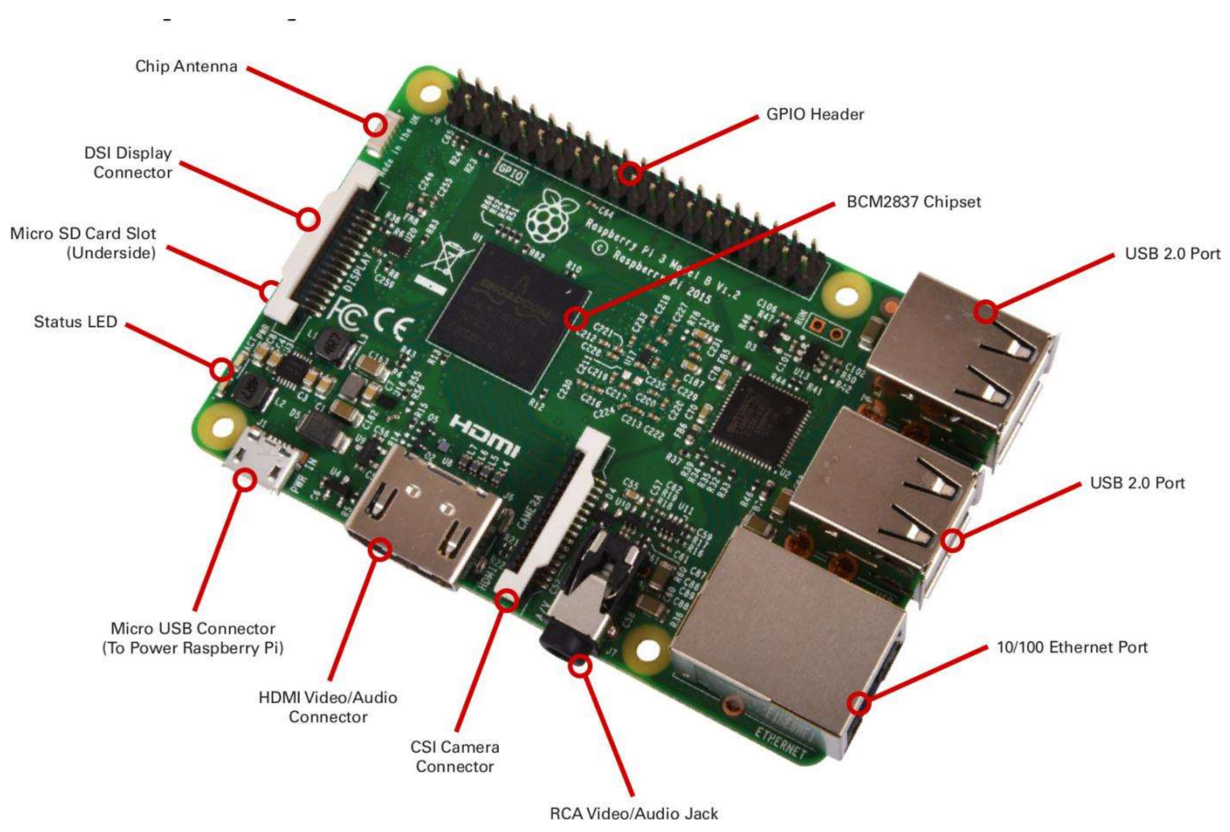


Fig. 4.1 Raspberry Pi 3 Model B

4.2 Stereo Camera

Specifications:

- 1.3MP dual lens usb camera module with 1/3 CMOS OV9715 image sensor, designed for easy integration and vision acceleration. The sensor module output can be implemented after a quick installation. The data output is transmitted through USB 2.0
- 1.3MP HD 960P stereo 3D VR dual lens usb camera with dual lens frame synchronization provide accuracy at less than one millisecond. Object identification, motion tracking, mapping, and AR & VR applications can integrate seamlessly, suitable for stereo 3D VR application like people counting system, stereoscopic computer vision
- 1.3MP 960 dual lens camera module with high frame rate can get MJPEG:2560X960@ 60fps/2560X720@60fps /1280X480@60fps /640X240@60fps.perfectly synchronized stereo images

- 1.3MP Web camera utilizing the disparity between the two lenses through algorithms, Allow MACHINE to see the world as we do. Passive stereo vision designed for environment versatility and featuring long-range capabilities and high-resolution depth perception. Small outline 80*16.5mm for embedded application [14]

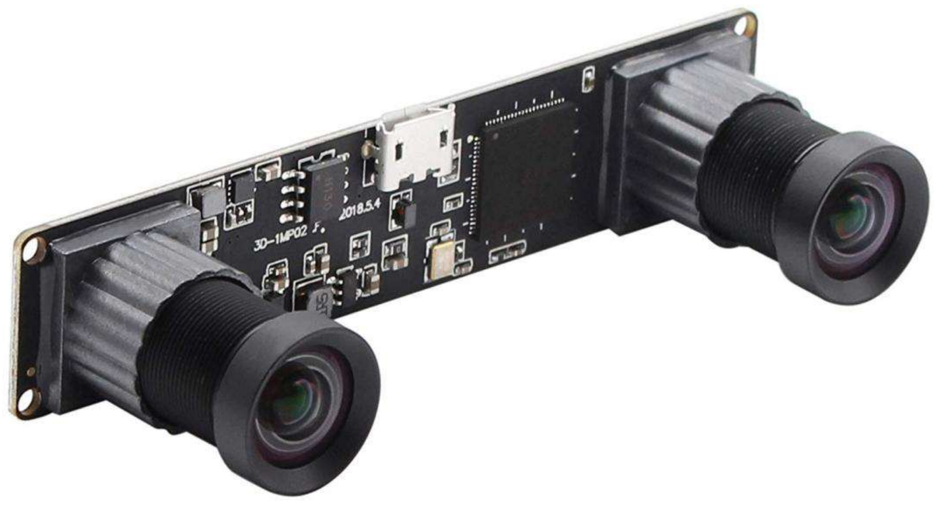


Fig. 4.2 Synchronized Dual Lens Stereo USB Camera

5. DESIGN OF THE GLASSES

Some views of the design of the glasses are given below.

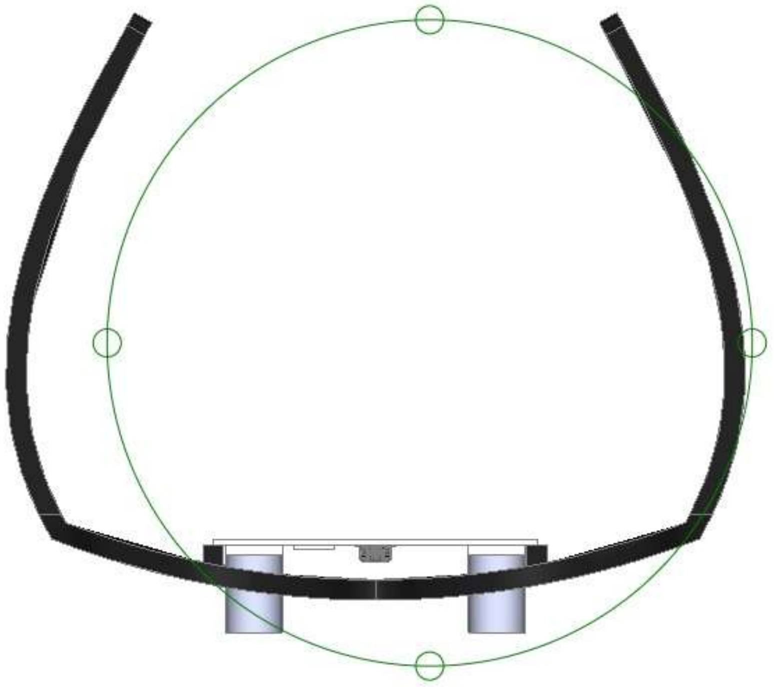


Fig. 5.1 Top View of the Glasses

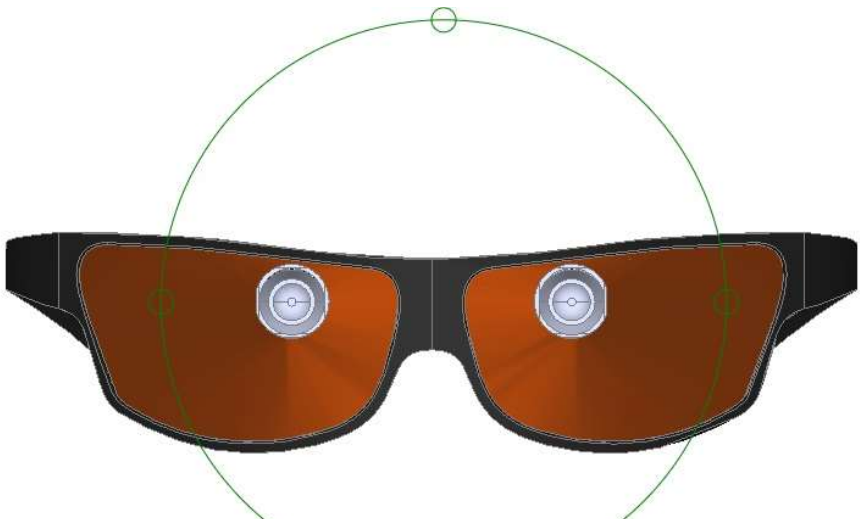


Fig. 5.2 Front View of the Glasses

6. CONCLUSION

We designed a pair of mobile wearable glasses for visually impaired people. The system uses a Raspberry Pi 4 Model B, a stereo camera, 3D printed glasses and a power bank to power up the system. The system makes object recognition and distance measurement with the methods HAAR Cascade and stereoscopic distance measurement, respectively. The objective of the project is to get any visually impaired person to anywhere they want to go.

The object recognition codes identify the objects that are considered to be an obstacle to the visually impaired person. With the help of our distance measurement codes that we developed distance to the object is also measured through the images taken from stereo camera. Both the name of the object and distance to it delivered to the user with a pair of headphones so that users can adjust themselves according to the objects that are defined.

The project can be improved with adding more functionality to the project to serve a better assistant for a visually impaired person, but with this project we wanted set a starting point to make an impact in the lives of those visually impaired people.

REFERENCES

1. INTRODUCTION TO ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS by Dan W. Patterson

2. http://textileguide.chemsec.org/find/get-familiar-with-your-textile-production-processes/

3. 2001-2016 Python Software Foundation

4. H. Walcher, Position sensing – Angle and distance measurement for engineers, Second edition, Butterworth-Heinemann Ltd., 1994.

5. Navigation 2, Radio Navigation, Revised edition, Oxford Aviation Training, 2004.

6. J. Carnicelli, Stereo vision: measuring object distance using pixel offset, http://www.alexandria.nu/ai/blog/entry.asp?E=32, (29.11.2007).

7. Jernej Mrovlje and Damir Vrancic, Distance measuring based on stereoscopic pictures, Izola, Slovenia, 2008.

8. Docs, OpenCV. "Face Detection Using Haar Cascades." *OpenCV: Face Detection Using Haar Cascades*, 4 Aug. 2017, Face Detection using Haar Cascades

9. Mathworks, Mathworks. "Train a Cascade Object Detector" *Mathworks*, 2017, Train a Cascade Object Detector

10. https://www.invensis.net/blog/it/benefits-of-python-over-other-programming-languages/

11. https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/

12. https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html

13. https://www.seeedstudio.com/Raspberry-Pi-3-Model-B-p-2625.html

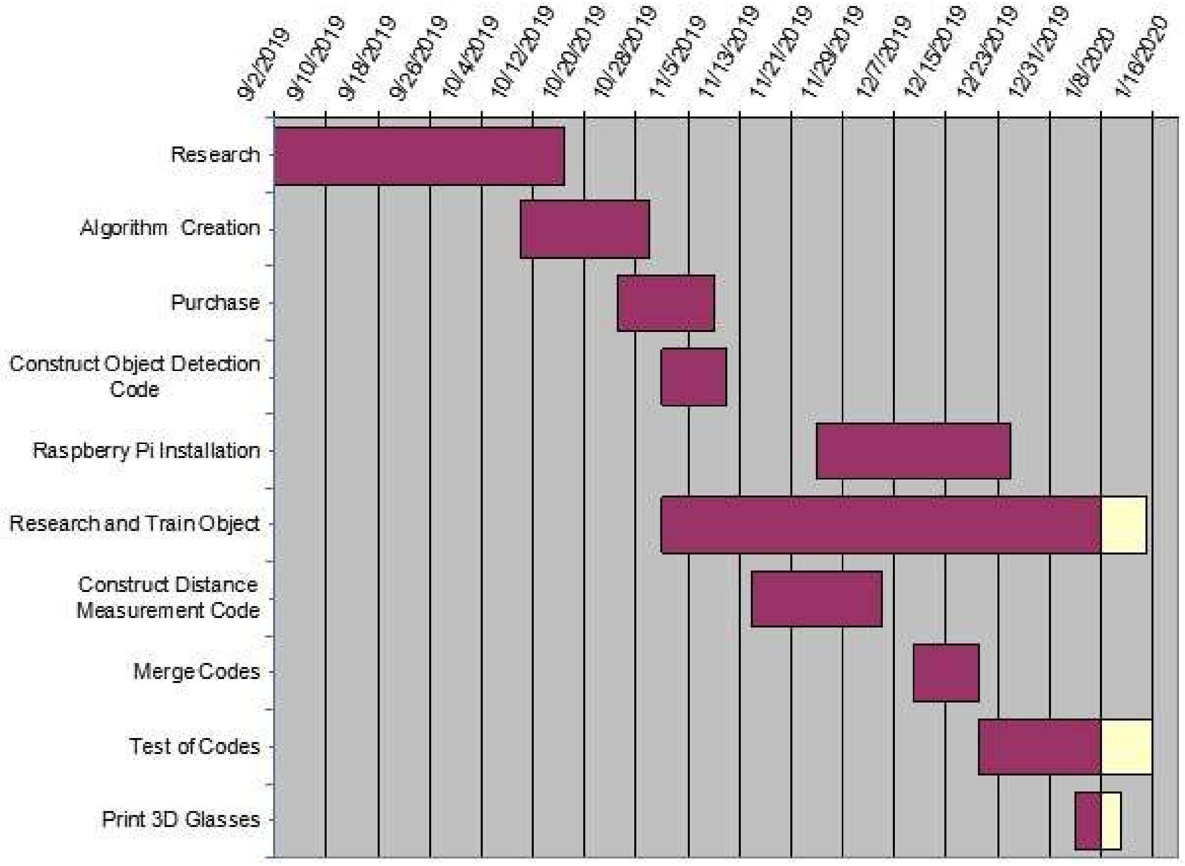
14.https://www.amazon.com/gp/product/B07THGYSBC/ref=ox_sc_act_title_1?smid=A3VIUP WY8LOI3F&th=1

APPENDICES

A) COST ANALYSIS

Count	Item	Cost ₺ / €	
1	Stereo Camera	620 ₺	95 €
1	Raspberry Pi 3 Model B	Donation	

B) GANTT CHART


9/2/2019
9/10/2019
9/18/2019
9/26/2019
10/4/2019
10/12/2019
10/20/2019
10/28/2019
11/5/2019
11/13/2019
11/21/2019
11/29/2019
12/7/2019
12/15/2019
12/23/2019
12/31/2019
1/8/2020
1/16/2020
Research
Algorithm Creation
Purchase
Construct Object Detection Code
Raspberry Pi Installation
Research and Train Object
Construct Distance Measurement Code
Merge Codes
Test of Codes
Print 3D Glasses