Palestine Polytechnic University

College of IT and Computer Engineering

Department of Computer Engineering

Graduation Project

Real-time Stage Tracking Camera using Raspberry Pi

Project Team

Suhair Shareef, Leyan Zahdeh, Beesan Atawna

Supervisor

Dr. Ayman Wazwaz

2021 - 2022

Acknowledgment

In the name of "Allah", the most beneficent and merciful who gave us strength, knowledge and helped us to get through this project. To the people that have inspired, supported, and molded us into the people that we are today, our families, friends and our supervisor. We would've never been able to reach this achievement without your support, care and encouragement.

We would like to express our sincere appreciation to our graduation project supervisors Dr. Ayman Wazwaz for his guidance, continuous encouragement, and support throughout the semester.

Furthermore, we have to show our gratitude to Eng. Wael Takrouri for being there to answer our questions and provide us with valuable suggestions and tips which helped us in choosing some system components. Moreover, it is our duty to thank our families for their generous encouragement and continuous support throughout our life. For our friends, Ghaid, Bara' and, Ruba. We are truly grateful for all your support throughout the whole education stage.

At last, we would like to thank all the people who helped, supported, and encouraged us to successfully finish the graduation project.

Abstract

Object tracking and detection has become a large field with a wide range of algorithms being used as a result in several computer vision applications. Because of the number of cameras used to cover a large area, these applications are constrained by the cost of each node, the power consumption, the robustness of the tracking, the processing time, and the ease of deployment of the system. To meet these challenges, the use of low-power and low-cost embedded vision platforms to achieve reliable tracking becomes essential in the field of cameras.

In this project, we propose a financially cheap and fast object detection and tracking model. The tracking is implemented in C++ with OpenCV on a Raspberry Pi 4 and the Raspberry Pi Camera Module as camera feed. To be able to detect and follow an object a pan and tilt system is built to be able to follow the object using tilting and panning motion. Two different detection algorithms have been used, object detection the default one that detects the whole object, and the second one when the object starting to get closer to the camera the system starts face detection.

The system created is able to detect and track a moving object and keeping it in the center of the image.

Contents

List of Figures

Acronyms

API Application Programming Interface.

CSI Camera Serial Interface.

GPIO General Purpose Input/Output.

IDE Integrated Development Environment.

OpenCV Open Computer Vision.

PWM Pulse Width Modulation.

ROI Region of Interest.

SoC System on a Chip.

USB Universal Serial Bus.

Glossary

Centroids A centroid is the imaginary or real location representing the center of the object.

Object tracking The task of taking the center of a detected moving object, calculate the difference between the center of the camera and that object, and control a motor to move according to the difference.

Chapter 1

Introduction

1.1 Overview

Object tracking and detection camera is a camera that has the ability to track certain objects using different object detection models. If we're talking about people, the camera would simply be moving according to your movement.

1.2 Motivation

Tracking cameras powered by artificial intelligence would reduce the costs of having a camera crew, reduce time in setting up your station and go right into execution. Most tracking cameras nowadays aren't affordable and might not have all the features you're looking for. This project will help people who are looking for affordable and smart cameras to work with on stage.

1.3 Objectives

In this project we aim to achieve the following objectives:

- Stage human tracking using object tracking and recognition in real-time.
- Deploying an object detection machine learning model on the raspberry pi.
- Live video streaming using a streaming API on Raspberry Pi.

1.4 Project Description

- Our project detects human movement using object detection machine learning model. The camera will detect the person's movement and move along depending on his movement (left/right/up/down).
- A live video is streamed online using streaming technology.

1.5 Problem Statement

- A stage tracking system usually would be manually installed by a number of crew members to control cameras from different angles, which could be time-consuming and costly.
- For real-time video processing, object detection must not only be able to classify and locate important objects, but it must also be incredibly fast at prediction time.
- Since the camera is a static object, it acquires an external source (human) to control it.

1.6 List of Requirements

- The system's ability to detect a human movement and track him based on his movement.
- The system must send the voice and the video recording at the exact time.
- Availability of a permanent internet connection.

1.7 Expected Results

We expect this project to be able to do the following:

- The Raspberry Pi camera will be able to detect a human.
- The Raspberry Pi will be able to to track human movement and move in their direction.

- Video streaming using a streaming API on Raspberry Pi.

1.8 Constrains

In the currently implemented project, we should note that:

- This system tracks only 1 human.
- Good lighting towards the presenter is important in order for the camera to detect and track his movement.

1.9 Block Diagram

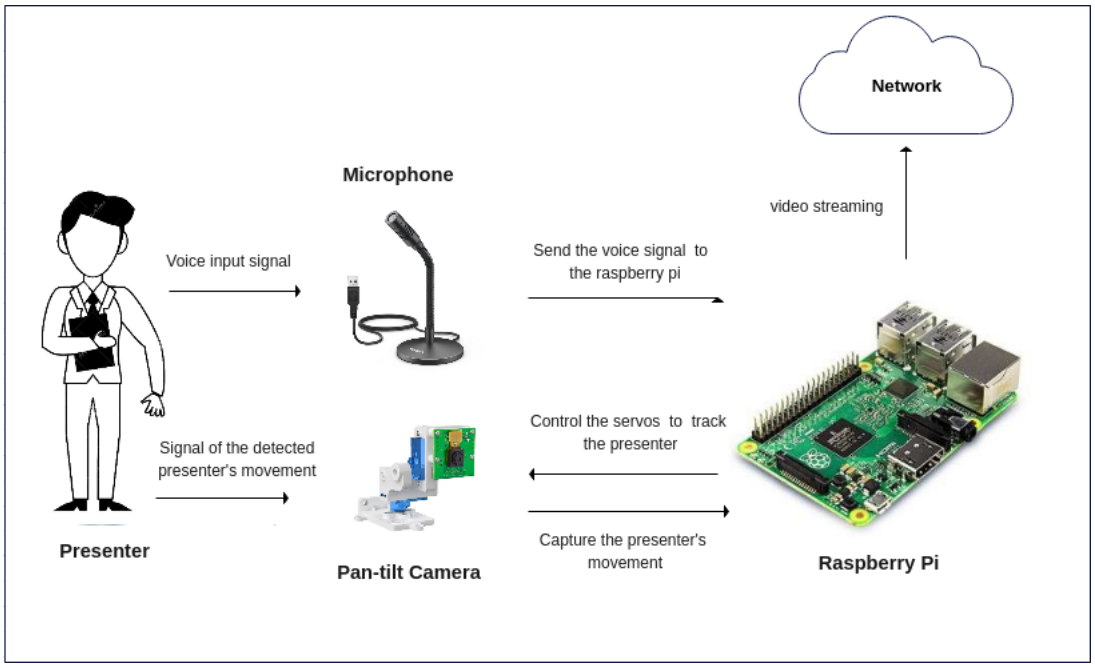


Figure 1.1: Basic Block Diagram.

The Figure 1.1 shows a simplified block diagram of our system. It clarifies how the user interacts with the system and how the system responds. The system runs on a microcontroller where the computations and machine learning models exist. The microcontroller receives data from the environment . The camera will move using the motors, which takes its input according to the computations of the machine learning model on the microcontroller.

1.10 Report Outline

This report is organized as follows: Chapter 2 introduces some literature review including available tracking cameras and related projects. It also goes briefly over the theoretical background of the project, hardware, and software components. Chapter 3 discusses the conceptual design of the system, block diagrams, pseudo-code, and detailed hardware connections. Chapter 4 discusses the hardware and software implementations of the project. Chapter 5 shows the testing which is made until we reach the final system design. Finally, Chapter 6 presents a summary and the future work.

Chapter 2

Background

2.1 Overview

This chapter mainly provides a quick background about the important components in our project. First, we will give a brief about the theoretical background, we will explain general terms like object detection, face detection, and object tracking. Secondly, we will present a literature review to show how previous studies achieved such objectives. Also, a short description of the hardware and software parts that are used in the system is also introduced.

2.2 Theoretical Background

In this section, we will provide some fundamental information about the main concepts on which our system is based.

2.2.1 Object detection

Object detection is the process of detecting and defining objects of a certain known class in an image or a video. The goal of object detection is to develop computational models that provide the most fundamental information needed by computer vision applications: "What objects are where?".When humans look at images or video, we can recognize and locate objects of interest within a matter of moments.

To be able to have a functional tracking system it requires an object detection system to be able to detect the object that will be tracked. Depending on the type of

object detection is being used the system uses a single frame or a sequence of frames to spot the object [1].

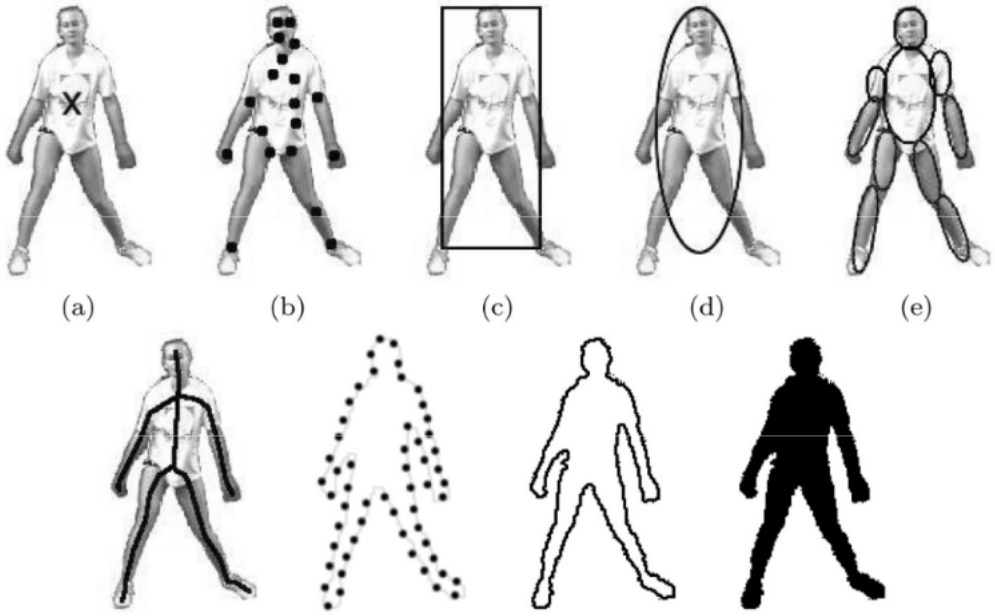


Figure 2.1: Object representations. (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) control points on object contour, (h) complete object contour, (i) object silhouette [1].

The object detection system uses a representation of an object for the recognition based on the shape and appearance of the object. The different kinds of object representation can be seen in Figure 2.1 and are:

- Points.
- Primitive geometric shapes.
- Contours.
- Articulated shape models.
- Skeletal models.

The appearance representation are:

- Probability densities of object appearance.
- Templates.

- Active appearance models.
- Multiview appearance models.

Choosing the object representation is mostly connected with application and the tracking algorithm since they are strongly related.

2.2.2 Face detection

Face detection is an AI-based computer technology that can identify and locate the presence of human faces in digital photos and videos. It can be regarded as a special case of object-class detection, where the task is to find the locations and specify the sizes of all the objects that belong to a given class – in this case, faces – within a specific image.

Due to the advancements in face detection technology, it is now possible to detect faces in an image or video, regardless of head pose, lighting conditions, and skin color.
Face detection applications use algorithms that determine whether images are positive images (i.e. images with a face) or negative images (i.e. images without a face). To be able to do this accurately, the algorithms must be trained on huge datasets containing hundreds of thousands of face images and non-face images [11].

Once trained, the algorithms are able to answer two questions in response to input in the form of an image:

- Are there any faces in this image?
- If yes, where are they?

If a face or faces are present in an image, the algorithms will answer these questions by placing a bounding box around the detected face(s), as illustrated in Figure 2.2:

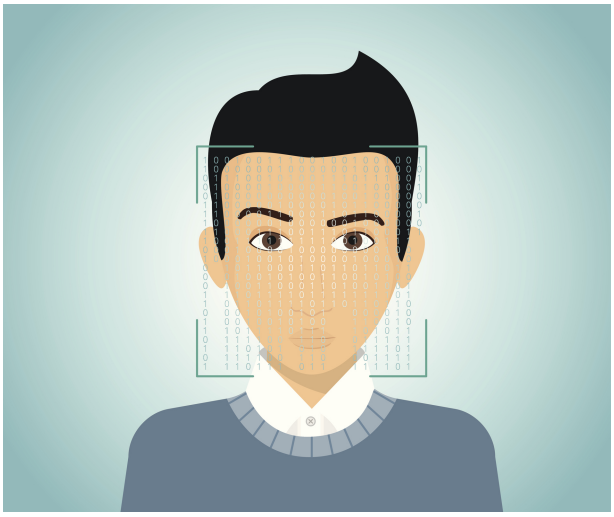


Figure 2.2: Face detection.

2.3 Literature Review

Below, we will introduce some of the previously proposed products and projects that are similar to this project.

2.3.1 Real time object tracking on Raspberry Pi 2

Description:

In this project the tracking is implemented in C++ with Opencv on a raspberry Pi 2 and the Raspberry Pi camera as shown in Figure 2.3. The model tracks moving objects using two different tracking algorithms and comparing the results between both of the two algorithms. There was three different experiments on different aspects of the tracking, speed, partial occlusions and illumination. The two object being tracked, a tennis ball and a book cover, the trackers was initialized with a Region of Interest (ROI) in each experiment as its starting point [2].

Features:

- The object detection system uses a representation of an object for the recognition based on the shape and appearance of the object.
- The tracking is done in real time.
- Use color-based recognition for tracking with another feature because color-based recognition is sensitive to changes.

- The measurements of successful tracking is tracked by using localization error. The localization error is determined by the Euclidean distance between the "manually segmented ground truth" center point and the center point of the tracker.

Limitations:

- The controller for the servo was sometimes a limiting factor by being too slow.
- If there are similar colors in the image it's a real problem to track a specific object. Since in all of the experiments the environment was mostly white and black.
- The KLT algorithm uses points for tracking and the tracker drops point after a while of tracking. which gets occluded or shifts to much frame to frame.
- The representation of the object limits the transformation the object being tracked in the image can go through without not losing it.

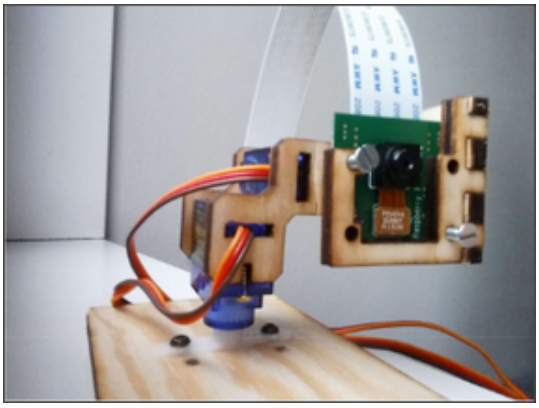


Figure 2.3: First version of camera mounts [2]

2.3.2 Fast and Economical Object Tracking using Raspberry Pi 3.0

Description:

This project uses KCF tracker algorithm using Opencv version 4.0 and Raspberry Pi 3. When the program is executed the user has to use the mouse and select the object to be tracked then the algorithm will start tracking the selected object. When the object is moved away from the line of sight of the camera, it remembers the object and when it appears it detects the object again [12].

Features:

- This system can track both video and live feed input.
- The tracking is done in real time.
- The frames are optimized per second to boost the tracker efficiency.

Limitations:

- It's not implemented for object detection since it limits the object to be tracked.
- This tracking is designed for tracking individual objects.

2.3.3 TR530 30X auto tracking camera

Description:

The TR530 auto-tracking camera as shown in Figure 2.4 creates an engaging video for streaming, sharing, and recording during lectures, demonstrations, video conferences and speeches. It tracks the target everywhere with Wide Area Tracking. Also, it provides flexibility to the presenter to leave the stage and interact with a crowd of students [3].

Features:

- Wide area tracking and capturing every interaction.
- Track the target in any environment.
- Features a second camera to offer a panoramic view.
- Segment tracking creates 4 content zones that allow for immediate recognition and tracking of your target as they move between content zones.

Limitations:

- It only offers 120° FOV (field of view), whereas our project provides 180° FOV.

Figure 2.4: TR530 30X auto tracking camera [3]

2.3.4 TensorFlow Enabled Deep Learning Model Optimization for enhanced Realtime Person Detection Using Raspberry Pi operating at the Edge

Description:

In this research, Quantization effects are assessed for a real-time Edge-based person detection use case that is based on the use of a Raspberry Pi. TensorFlow architectures are presented that enable the use of real-time person detection on the Raspberry Pi. The model quantization is performed, the performance of quantized models is analyzed. In this work on person detection, the input will be captured by a pin-point camera device which will be mounted on an edge processor. The real-time streaming and processing of image is necessary at the edge device to initiate the detection process on the edge device. The Edge computing device used in this work is a Raspberry Pi 4 with 4GB RAM shown in the Figure 2.5 [4].

Features:

- Response time and having access to the system even when the internet is down.
- Uses edge computing as the process of transferring the computation and communication resources to the edge of networks, from the cloud, to reduce the latency, enabling faster responses for end users.
- Uses some cheap cameras that provide an interface to fetch images, a local server that searches for those local cameras and processes their images using machine learning and computer vision, then sends the processed data to the cloud to

monitor and treat the cameras as a sensor by knowing the content of the image and it can be used for detection.

Limitations:

- It's only for object detection, it doesn't track objects.

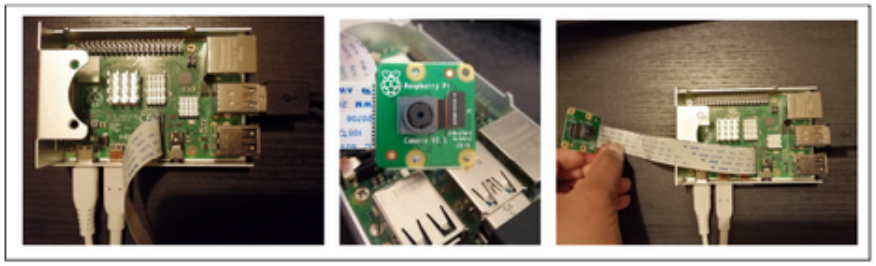


Figure 2.5: Pi camera and Raspberry Pi [4].

2.4 Components

This section contains a short description of the hardware and software components used in our system. It also contains a brief justification for certain choices.

2.4.1 Hardware

In this section, we will present the main hardware components of the project and some detailed descriptions of them.

Raspberry Pi 4 (4GB RAM)

The Raspberry Pi is a low cost, small weight and credit-card sized computer, [5]. It is essentially a system-on-a-chip (SoC) with connection ports. It has a 32-bit ARM processor which delivers a range of processing speed from 700 - 1000 MHz and Videocore 4 GPU as shown in Figure 2.6.

Raspberry Pi 4 key features include a high-performance 64-bit quad-core processor, dual-display output via two Micro HDMI ports, up to 4K resolution, hardware video decoding at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability [5].

Compared to Raspberry Pi 3, Raspberry Pi 4 has a wide range of RAM sizes, whereas

Raspberry Pi 3 comes with only 2GB. It also sports a faster 1.5GHz clock speed processor. We selected it over a tablet or a phone due to the size, price and complexity.

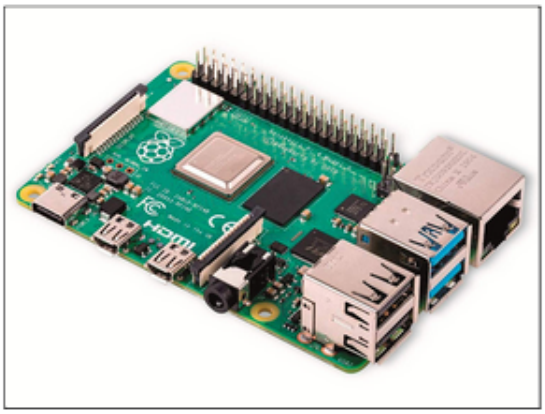


Figure 2.6: Raspberry Pi 4 [5]

Raspberry Pi Camera

A high-quality 5-megapixel image sensor custom-designed add-on board for Raspberry Pi supporting 1080p and 720p video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSi interface, designed especially for interfacing to cameras as shown in Figure 2.7 [6].

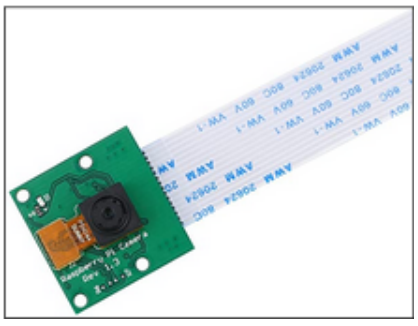


Figure 2.7: Raspberry Pi Camera [6]

2 Axis Servo Pan Tilt

2 Axis Pan Tilt Brackets for Raspberry Pi Camera which is based on 2 axis pan and tilt mechanism for mounting the wired camera. Panning, rolling and tilting are achieved by controlling Servo motors using PPM pulses as shown in Figure 2.8 [7].

2 Servo Motors

A rotary actuator or linear actuator that allows for precise control of angular or linear

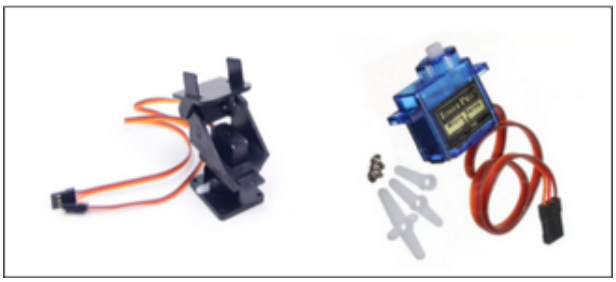


Figure 2.8: Pan-Tilt & Servo Motor [7].

position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors as shown in Figure 2.8 [13].

5V 3A Switch Power Adapter

This is the power adapter for Raspberry Pi 4 Model B, with a USB type C interface. And there is an ON/OFF switch button on the cable. It provides an input range of (100V 240V, AC, 50/60 Hz) and an output of 5V 3A, DC [14].

USB Microphone

The USB Microphone for dictation and recording can be used for moderated sales meetings, public address functions and courtroom, laboratory and surgical suite environments proving assurance that the spoken word will be heard clearly with distortion-free clarity, It's shown in Figure 2.9 [8].

Figure 2.9: USB Microphone [8].

2.4.2 Software

Raspbian

Although Raspberry Pi can utilize different types of Linux distributions for its default operating system (OS), Raspbian is the recommended one. It is a free operating system based on Debian, optimized for the Raspberry Pi hardware. It comes with over 35,000 packages; precompiled software bundled in a good format for easy installation on Raspberry Pi hardware, [15].

A free operating system based on Debian optimized for the Raspberry Pi hardware. Coming with over 35,000 packages, pre-compiled software. This operating system provides significantly faster performance for applications that make heavy use of floating-point arithmetic operations [16].

TensorFlow Lite

An end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

Since we're running our application at the edge, we're choosing a simplified version of TensorFlow called TensorFlow Lite that provides several advantages over TensorFlow's protocol buffer model format such as reduced size (small code footprint) and faster inference (data is directly accessed without an extra parsing/unpacking step) that enables TensorFlow Lite to execute efficiently on devices with limited compute and memory resources [17].

MobileNetV1-SSD model

The ssd-mobilenet-v1-coco model is a Single-Shot multibox Detection (SSD) network intended to perform object detection. The difference between this model and the mobilenet-ssd is that there the mobilenet-ssd can only detect face, the ssd-mobilenet-v1-coco model can detect objects[18].

The model can be used in the following demos provided by the Open Model Zoo to show its capabilities:

- Object Detection C++ Demo.
- Object Detection Python Demo.
- Pedestrian Tracker C++ Demo.
- Single Human Pose Estimation Demo.

OpenCV

Opencv is a free library with hundreds of computer vision api that can be used in image processing to improve a real-time application. Object detection, camera calibration, 3D reconstruction, and an interface to video processing are some of the data processing functions in opencv. In this project, python language will be used as the programming language with the required libraries from opencv to build the hand gesture recognition system.

Chapter 3

Design

3.1 Overview

This chapter discusses the overall design of the system and the way its components are integrated, showing the block diagram and schematic diagram for the design, in addition to some details about the algorithms we are going to use.

3.2 Detailed Design

The system design shown in Figure 3.1 illustrates a general overview of the system's main components and the connections between them.

- The system is set up in the middle of the stage with a certain distance far from the speaker on origin point (motors are in 90-degree status).
- The pan-tilt will cover the whole stage area, so the speaker is free to move at a normal speed.
- The camera and microphone are connected to the Raspberry pi as input devices, while the servos of the pan-tilt work as output depending on the presenter's movement.
- Video streaming will be started manually and it'll be accessible online using streaming API.
- The camera will detect the presenter and move along depending on their movement.

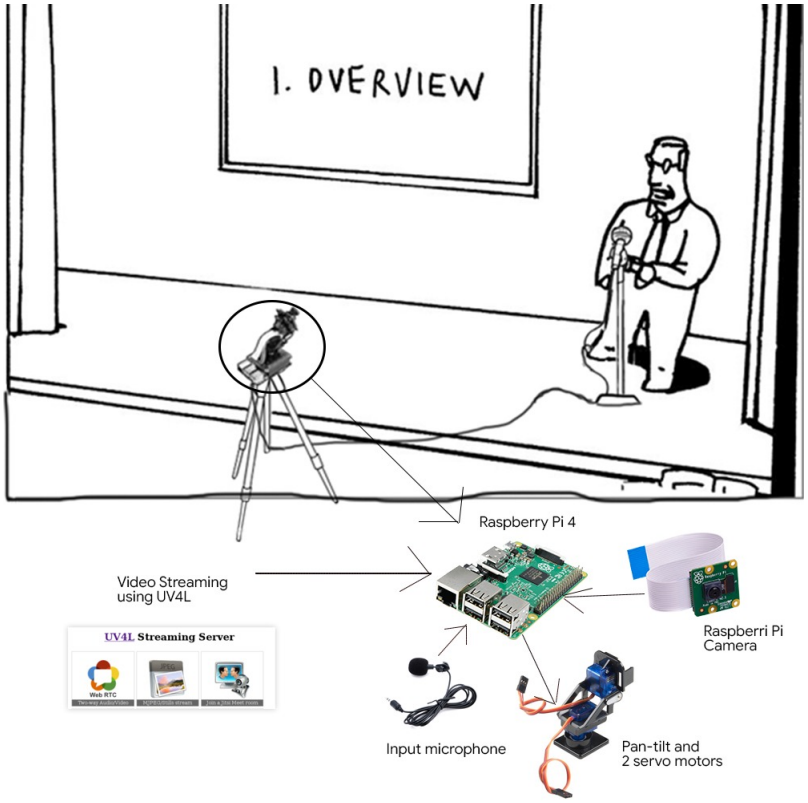


Figure 3.1: Environment and System Representation.

3.3 Block Diagrams and Flowcharts

The system diagram in figure 3.2 shows how the frames coming from the camera are transmitted in one way to the Raspberry Pi, which will determine whether the motors are going to move or not. The system starts looking for an object to detect. While the user is presenting, the system will stream the video online using streaming API.
While the system is running, there will be several uses to it:

- The system will start detecting the presenter, it'll run the rpi-object-detection package and start looking for humans in the camera view.
- While the presenter is in view, when running the tracking script, the pan-tilt will move according to the presenter's movement.
- If there's no presenter around, the camera will look around for one every 5 seconds.
- If the presenter come closer to the camera, the system will start face detection.

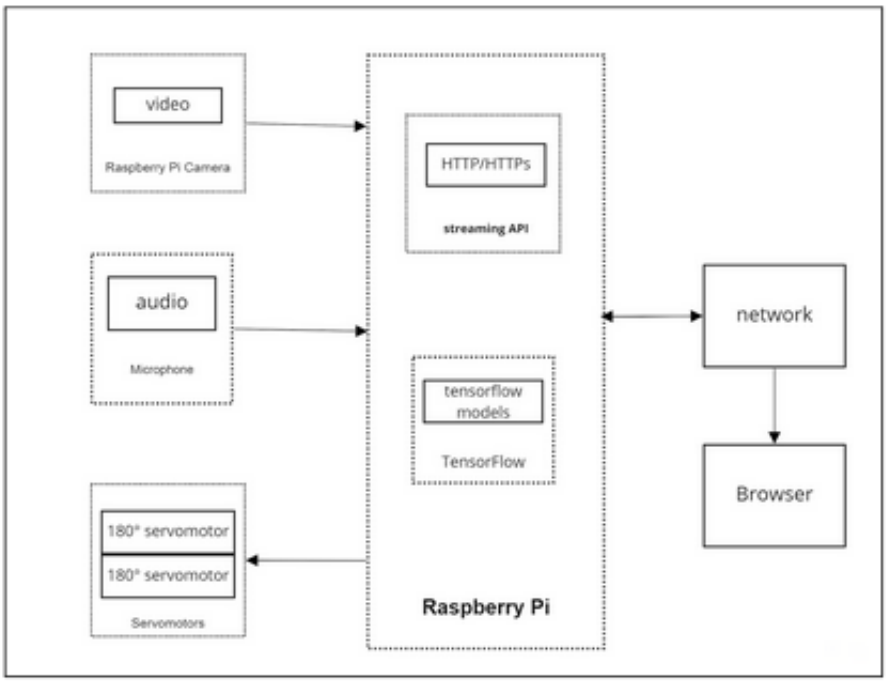


Figure 3.2: System Block Diagram.

3.4 Video and voice streaming

After running uv4l it will automatically load the plug-in and run the server. We can access the homepage at the following address: http://raspberrypi:8080 (where raspberry pi has to be replaced with the actual hostname or IP of the RaspberryPi in our network). And also we do not necessarily need a browser to see the video streams. we can use a VLC media player (open source cross-platform multimedia player and framework that plays most multimedia files as well as DVDs, Audio CDs, VCDs, and various streaming protocols) From within VLC, it's possible to see, transcode and record into a file – at the same time – almost any kind of stream format over HTTP/HTTPS [19].

3.5 Pseudo-Code and Algorithms

The algorithm shown below explains how the system will be initialized and triggered at the Raspberry Pi.

```
1: // Setup Part

2: START the machine learning model
3: INITIALIZE the stream
4: INITIALIZE the servos motor's pins and degree to the origin

5: // Loop Part
6: START
7: WHILE no interrupt

8:     IF the object detected == "person"
9:         IF face found in the picture  center of the face in the borders of the person
10:            center = center of the face

11:        ELSE
12:            center = center of the object
13:    ELSE IF  faces found in the picture
14:            center = center of the the face with highest score of accuracy
15:    ELSE
16:            search for objects every 5 seconds

17:    difference = || camera's center - centroids ||
18:    IF difference > threshold
19:        servo motor's degree ± = difference / 75

20: END WHILE
21: END
```

3.6 Schematic Diagram

In figure 3.3, the schematic diagram represents the components of the system and their connection with the Raspberry Pi with the output pins of the servos.

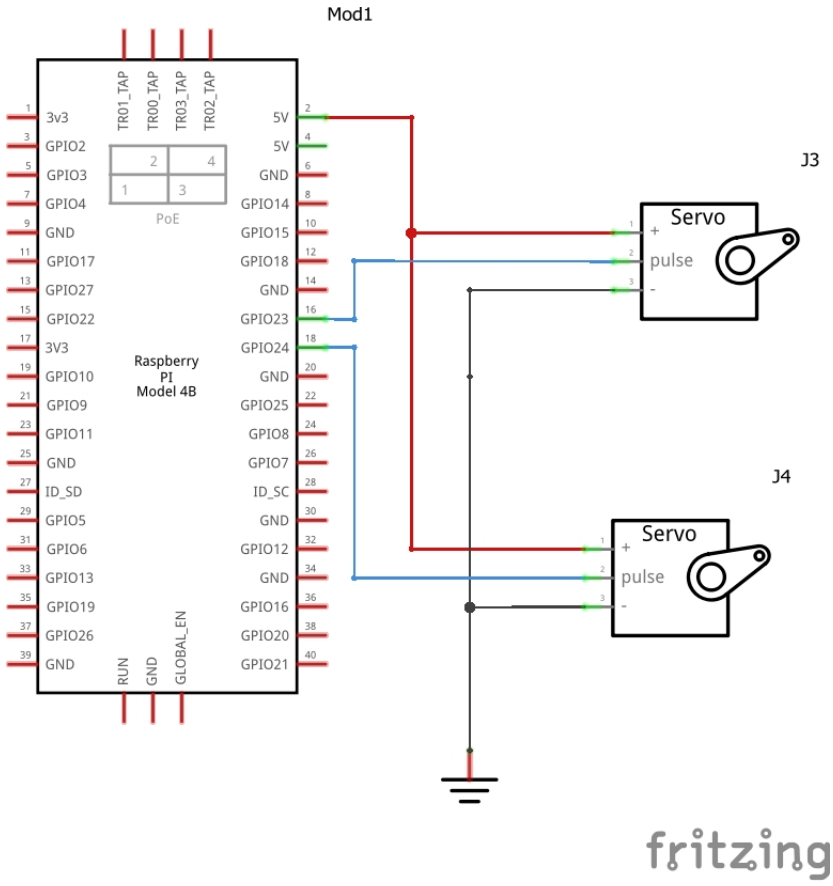


Figure 3.3: Pan-tilt servos and Raspberry Pi schematic diagram.

3.7 Hardware Setup

At the stage, the presenter wears the microphone that is connected to the Raspberry Pi. The pan-tilt connected to the Raspberry Pi and the Raspberry Pi camera is set in the middle of the stage away from the origin by two meters so that the camera can easily navigate between the left and right movement of the presenter while tracking.

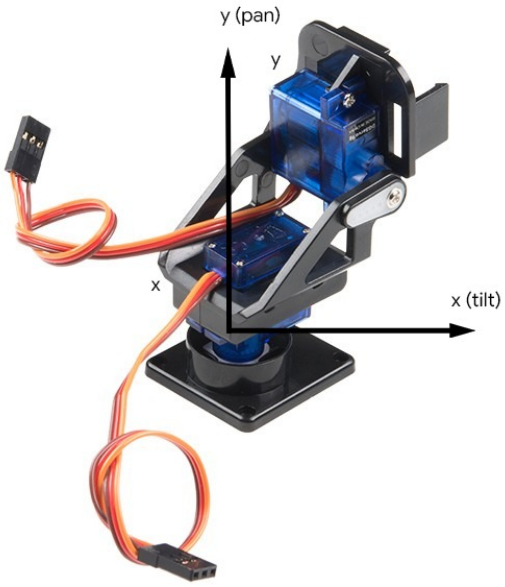


Figure 3.4: Pan-tilt rotation angles. [9]

As shown in figure 3.4, when the servo motor at y moves, the upper stand moves at y direction causing the pan movement. When the servo motor at x moves, the lower stand moves along the x axis causing the camera to move up and down causing tilt movement.

Chapter 4

Implementation

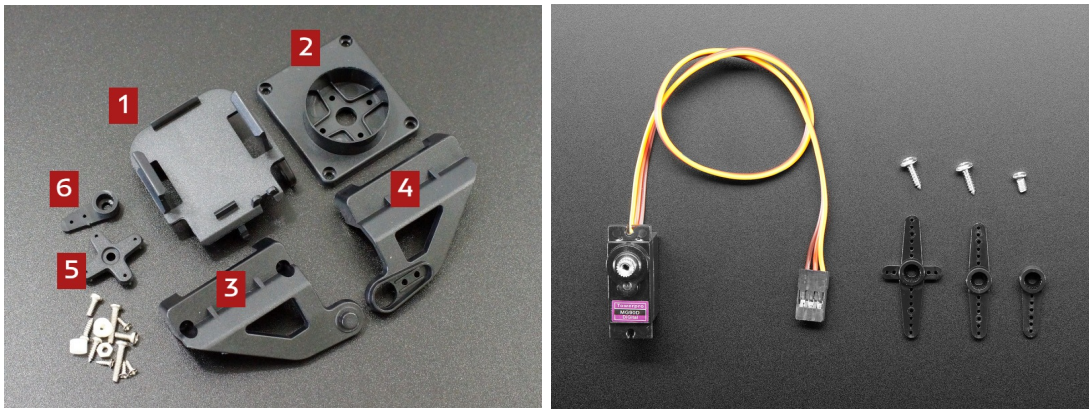
4.1 Overview

This chapter describes the implementation part of our project with more details. It dives deep into the different hardware components of the system and its software with all of its modules.

4.2 Hardware Implementation

In the previous chapter we viewed the schematic diagram of the system which shows how the components interconnect with each other. In this section we will present how we assembled these components and arranged them.

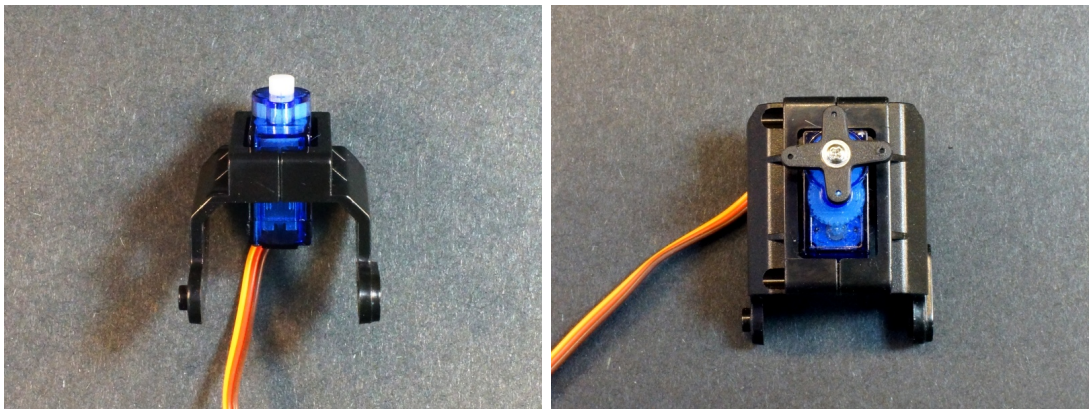
Our hardware assembly is pretty straightforward, it included the pan-tilt kit and the two servos. Alongside the servos pin connections to the raspberry pi, and the pi camera.



(a) 2-Axis Pan and Tilt Mount Kit (b) Micro Servo - MG90D

Figure 4.1: Pan-tilt kit and two servos.

The first step was to assemble the pan-tilt parts. Starting from the top part, we attached the pieces 3 and 4 in figure 4.1 (a) that are used to surround one of the servos with 2 of the longer self tapping screws. The figure 4.2 (a)two servos represents the result.



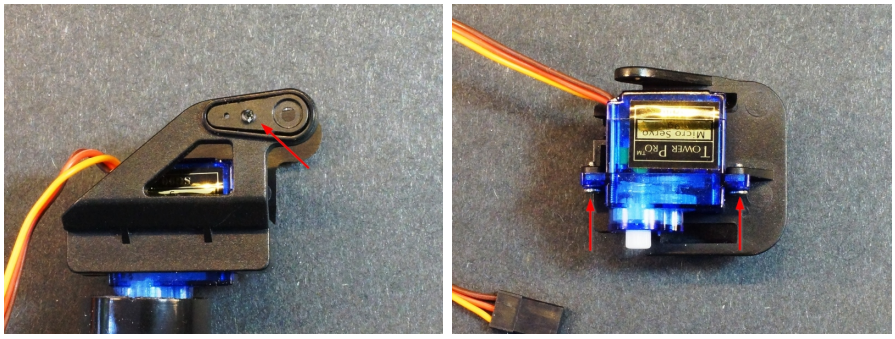
(a) Servo between bracket (b) Servo with star horn

Figure 4.2: Install the star horn

Next we installed the star horn (piece 5 in figure 4.2 (b)) on the servo shaft. It is fit in piece 2 at the bottom part of the pan-tilt. We made sure the servo was able to move after installing it as shown in figure 4.2 (b). After that we attached it to the bottom piece 2.

The next step was to install the second servo into the bottom of the tilt bracket.

We used the two short machine screws to attach the servo to the bracket as shown in Figure 4.3 (a) and 4.3 (b).



(a) Tilt bracket (b) Servo with machine screws

Figure 4.3: Install the second servo into the of the tilt bracket

Last step was to attach the tilt bracket to the base. We inserted the tilt bracket servo shaft into the single horn we installed previously, then we slipped piece 3 into the hole in piece 1. We ended up with the results in figure 4.4.

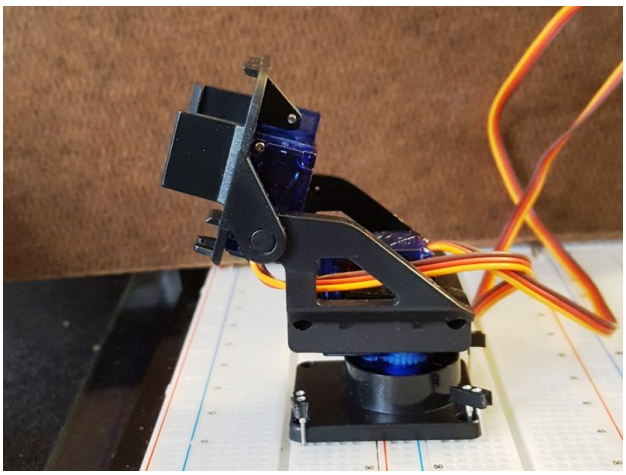


Figure 4.4: Assembled pan-tilt.

After assembling the pan-tilt, we attached each servo into the raspberry pi GPIO pins. The pins for the servos are shown in Figure 4.5. We wired the pan servo into pin 31, ground to pin 7 and power to pin 2. The tilt servo was wired to pin 11, ground to pin 6, and power to pin 4.

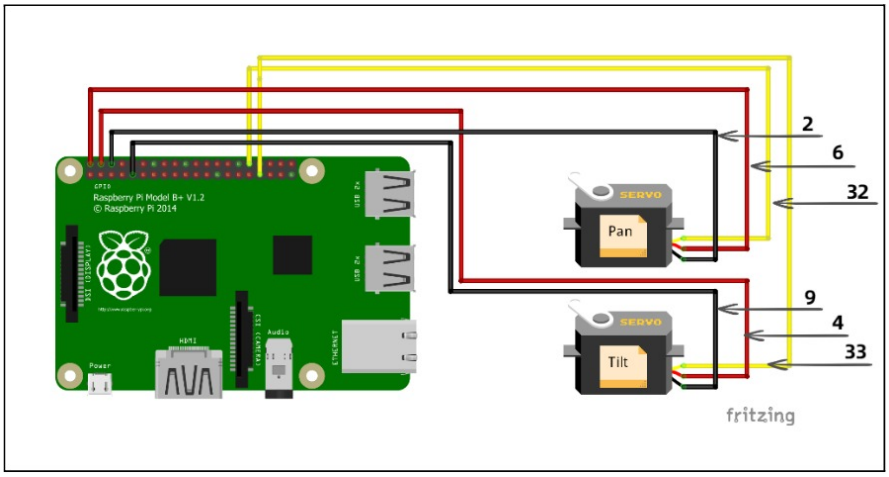


Figure 4.5: Raspberry pi pins wiring with the servos.

The last part was connecting the raspberry pi camera into the camera module output. We used a ribbon cable to connect the camera into the raspberry pi after removing the plastic clip.

4.3 Libraries Installation

In order for the system to start working, we needed several libraries to be installed on the OS of the raspberry pi. This required writing a Raspbian OS image to an SD card. We chose a high speed SD card with a read speed around 100 MB/S. The image we used had every necessary library preinstalled, including OpenCV, Tensorflow Lite, GStreamer, and Python's version. Then we installed Pigpio library for the servos.

4.3.1 pigpiod

We used the pigpiod utility of the pigpio library that runs it as a daemon. It allows the library to run in the background, so before we run the exe program, the daemon should already be running. This requires sudo privileges [20].

4.3.2 Coding

In this section, we'll describe setup, detection and tracking functions on the raspberry pi.

Setup

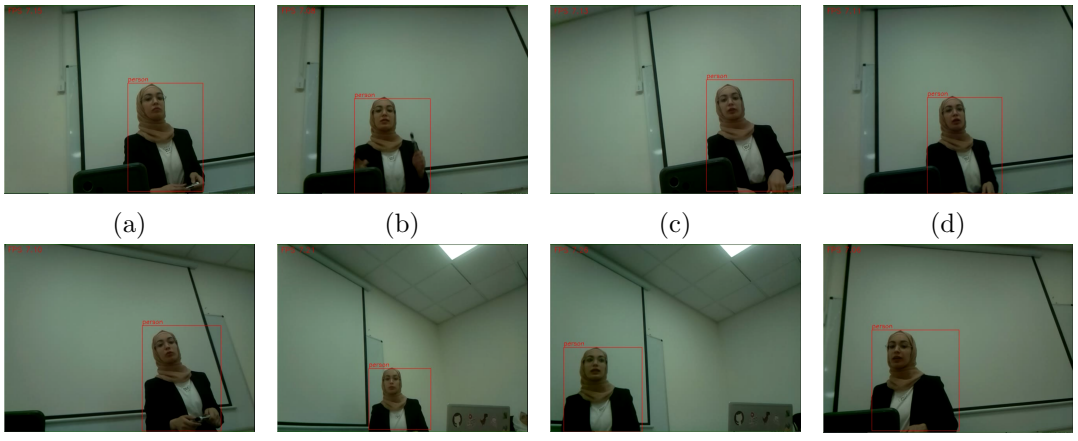
First, we initialize several parts:

- Object detection model (MobileNet SSD V1) using Tensorflow Lite interpreter.
- Face detection model (Ultra-Light-Fast-Generic-Face-Detector-1MB).
- Connect to the pigpio daemon.

Loop

Object Detection

After the models are loaded into the system, the detection process will start when the camera is on. The system will keep capturing images from the video and processing it using the detection model, until we receive a keyboard interrupt that will end the process. The Tensorflow lite API will process the received frame and showcase if a person is found. The model is capable of detecting multiple persons in the same frame that has a detection score higher than the threshold (0.6). We chose the person who has the highest detection score to be tracked.



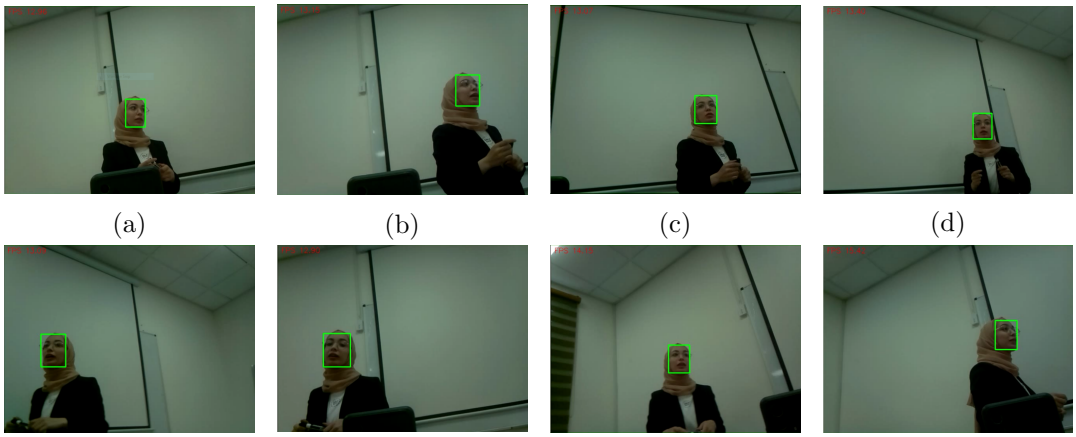
(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.6: Multiple frames representing the servo motor's movement according to the presenter's movement using the object detection model

Face Detection

Using the face detection model "Ultra-Light-Fast-Generic-Face-Detector-1MB", the algorithm detects faces for every captured frame. If there's a human object already, we take the face detection model to increase the accuracy and efficiency of the tracking algorithm. This will help us focus the camera on the face of the person instead of their center. It also comes in handy when the object detection algorithm fails to find a person. For example when a person has their body covered behind a barrier or when the person has moved too close to the camera, the model fails to identify them, but the face detection model can do the job for it. So instead of sending the centroids of the person object, we send the centroids of the face.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.7: Multiple frames representing the servo motor's movement according to the presenter's movement using the face detection model

Person's Tracking

After receiving the centroids of the object that we're going to track, we'll calculate the change of the angle of each servo. For each of the pan and the tilt servo, we calculate the error between the camera's center and the received coordinates. Since we're talking about frames, the result from the latter is in pixels, but that's not the actual error distance. The calculated error is multiplied by a constant that makes the movement of the servo motor realistic. We experimented several factors with different accuracies, and ended with a factor of 1/75. The equation is represented in figure 4.3.1.

```
errorPan = x - x_camera_center
errorTilt = y - y_camera_center

if (abs(errorPan) > 15)
    pan_degree -= errorPan / 75

if (abs(errorTilt) > 15)
    tilt_degree += errorTilt / 75
```

Figure 4.8: servo error mapping algorithm.

Object Search

When the system has no objects nor faces in its frame, it'll wait for 5 seconds and then search for objects in its 180 degree radar. This helps when the person gets out of the frame while the system doesn't manage to track them. We compute how many seconds passed with no detections using the std::chrono library. By taking the timestamp after each detection, if the detection is successful, we don't store the timestamp, but if it's not, then it'll be stored. We keep storing the timestamps where we didn't have any detections and we calculate the difference between the newest and the oldest timestamp. If it's 5 seconds or greater, we do a scan on the room. The scan will be exactly three spins on the whole area. The timestamps vector will be cleared if we find an object or if we perform a scan.

Chapter 5

Results and Validation

5.1 Overview

In this chapter we will discuss the testing of all components of the system and the results obtained. We tested all the parts to ensure that all of the functions work as expected and without errors.

5.2 Hardware Testing

This section discusses the testing process of each of our hardware components.

5.2.1 Testing MG90 Servos

We had two different servos. Each one of them was connected individually with the Raspberry pi as described in the wiring diagram below and we did three tests. In the first test, we checked if both of the servos were working well by sweeping them to different degrees. One of them was broken so we had to replace it. In the second test, both of the servos was sent from the degree 0, 90, 180 separately by a 5 seconds interval to check the range of the motors. The third test was going gradually from 0 to 180 and back to 0, this test was done to check the smoothness of the movement. In this test both of the servos were shaking "motors jittering" and they were moving slowly. The problem was with the library that we used so, we had to change it from RPi.GPIO to pigpio.

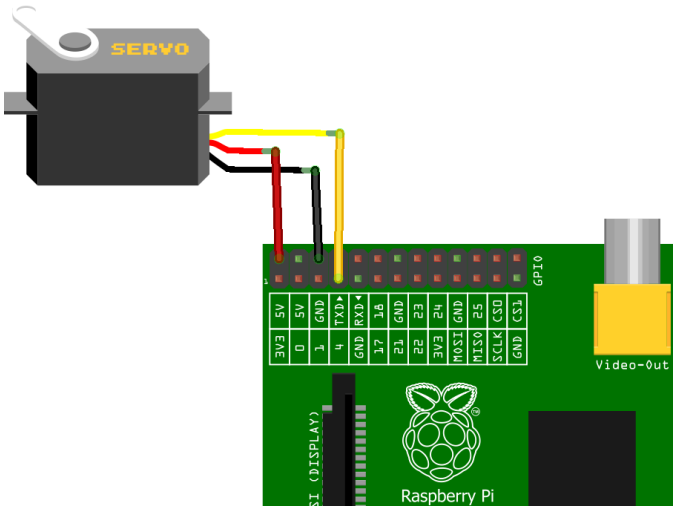


Figure 5.1: Raspberry pi pins wiring with the servos [10].

5.2.2 Testing Raspberry Pi

We did point testing on the Raspberry Pi by checking every point so it can help with troubleshooting hardware issues. We had a problem with the power. it wasn't taking enough voltage and the Raspberry was shutting down each time we're using OpenCV and the Raspberry Pi Camera so we had to replace the charger with one that fits the Raspberry.

5.2.3 Testing Raspberry Pi Camera

We connected the camera on the Raspberry using the MIPI CSI interface (the port in the center near to the USB ports) as shown in Figure 5.2 . Then we enabled the Camera Interface from Raspberry Pi configuration as shown in the Figure 5.3, and run the below commands to test it by taking an image and recording a video.

raspistill -o Desktop/image.jpg
raspivid -o Desktop/video.h264
It captured the photo and a video.

5.2.4 Testing model movement

Our main concern with the movement of the model was checking if the camera will move probably with the presenter. We did multiple tests first one, with one presenter

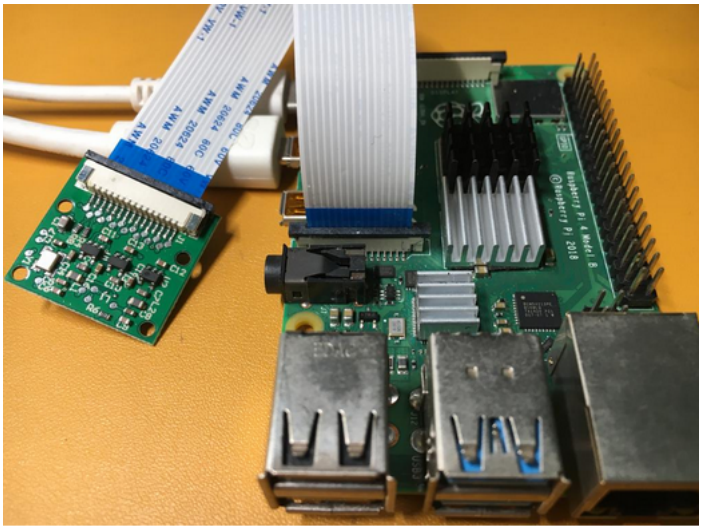


Figure 5.2: Camera connected to Raspberry.

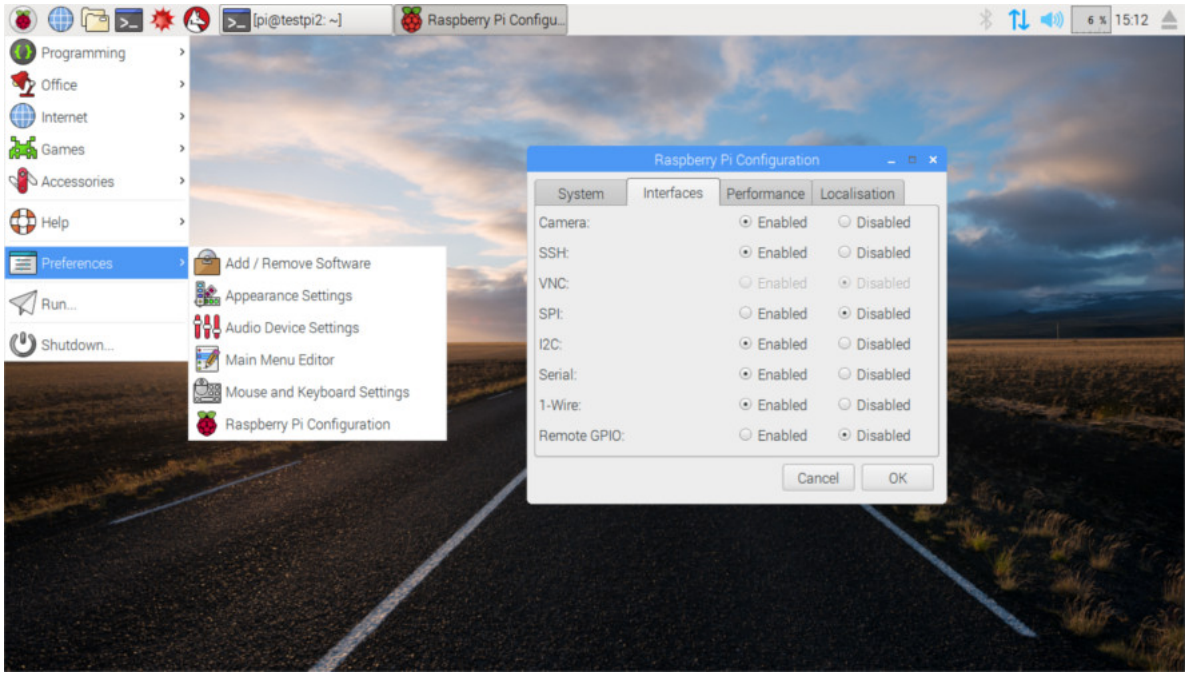


Figure 5.3: Enabling the Camera Interface from Raspberry Pi configuration.

and the camera moved along with the presenter and kept him at the center of the image. second test, if there was multiple presenter the camera will move along with the higher percentage. Last test was if the presenter come closer to the camera it'll start face detection.

5.3 Software Testing

This section discusses the testing process of each of our software testing.

5.3.1 Raspberry Pi OS installation

We installed and tested various versions of "Raspbian OS" until we found the one that fits our requirements. The list below represents each test of the versions we used :

- **Bullseye 32 bit :**
  - Incompatible with Picamera that is used by the tracking package.
  - Tensorflow version for the package can't be installed here.
  - Python's version isn't compatible with the tracking package.
- **Bullseye 64 bit :**
  - Python's version isn't backward compatible.
  - Picamera isn't supported.
  - Can't install the required tensorflow version here.
- **Buster 32 bit :**
  - Takes a lot of time to boot and is too slow.
  - Does Not use all the resources on the Raspberry Pi.
  - Response time for the camera isn't real time.
- **Buster 64 bit :**
  We used Buster 64 bit and took a version of the Os with all the needed dependencies and libraries already installed. It fitted our requirements and we had the least problems with it for many reasons :
  - The booting is faster.
  - Opencv showed a faster fps when using this os compared with the 32 bit version.

– Streaming was faster and smoother on VNC and XRDP servers.

5.3.2 Testing the object detection and tracking package

It worked on the 32 bit version of Raspbian but it had a few problems :

- The speed of the package was around 3 - 5 FBS.
- It was using an older version of TensorFlow lite and the installation process was time consuming and tedious.
- We decided not to move forward with this package because it did not serve our requirements.

5.3.3 Testing face detection

The detection model we used is "Ultra-Light-Fast-Generic-Face-Detector-1MB". We used it because it provides a lightweight and fast face detection model for edge devices. We installed the model and tested it with OpenCV. It resulted with a fast detection speed around 15 - 17 FBS without adding the object detection model.

5.3.4 Testing Opencv

We tested the Opencv version's compatibility with SSD Mobilenet_v3 and the model worked fine with it. Then we tested the RPI camera with Opencv and it worked fine as well.

5.3.5 Testing Tensorflow lite

Tensorflow lite's version was compatible with our requirements in C++ environment. We tested it by trying to connect it to OpenCV and the object detection model. We used its detection API to run the model. It worked as expected.

5.3.6 Testing SSD Mobilenet Object Detection Model

Testing this model included testing it on several objects with COCO object detection dataset. The first test was on version 2 in python's environment and the Opencv api

dnn_Detection Model for SSD Mobilenet models. Then we tested the large version of SSD MobileNetV3 in python's environment, it's fbs was almost similar to V2 with around 5 to 10 fbs on average. Also, We tested version 1 in C++ environment and it resulted with almost 10 - 16 fbs on average (almost double the python's environment).

5.3.7 Testing C++

Used c++ from the libraries, tested compatibility with the pigpio library, tensorflow, and opencv.

5.3.8 Testing Rpi.GPIO

This library resulted in jittering when using multiple servos, we tried fixing the problem by using an outer power source but it wasn't the problem. The PWM source coming from the raspberry pi itself isn't stable. The reason it's happening is because the pulses are generated by software, and if the software is out by even one millisecond then the motor thinks it needs to be somewhere else because it's done in a millisecond.

5.3.9 Testing pigpio

This library was an alternative to the Rpi.GPIO because it solved the jittering problem. pigpio is a special library from gpiozero library that allows you to use different sub-systems for controlling the pins. It's main feature that it's hardware based timing for pwm and for servo pulses. We tested this library using python's environment and the response had no delay. The problem with this library is that it requires sudo privileges , so we can't run the library without it. That means that only root users can run our program. The library didn't work when used with C++ because of some compatibility issues, so we replaced it with pigpiod_if2 which is a compiled version of the library. It also requires the library's demon to run in the background, but it didn't show any compatibility issues. The pigpio library controls the servos by running an instance of pigpio and it controls all the servos [20].

5.3.10 Test Scenarios

Presenters No	Environment	Expectations	Results
One presenter	Bright No obstacles Normal walking pace Standing in **front** of the camera	The camera should be able to track the presenter's position and move in real-time according to the presenter's movement	The system passed this test most of the time, with a little lag in the tilt servo. The camera was able to track the presenter most of the time
One presenter	Bright **Some** obstacles Normal walking pace Standing in front of the camera	The camera should be able to track the presenter's position most of the time and be able to catch parts of the presenter behind the obstacle	The camera was able to catch the person when the obstacle didn't hide most parts of the person
One presenter	Bright No obstacles **Fast** walking pace Standing in front of the camera		The camera wasn't able to catch the person when walking in high speed, although when doing a scan after 5 seconds it was able to correct itself and find the person

Two presenters (switching between presenters)	Bright No obstacles Normal walking pace Standing in front of the camera and the **other presenter is out of the frame**	The camera should be able to switch between the two presenters and track the new presenter	This test passes when the presenter leaving is moving faster than the one coming. It fails when the presenter is moving slowly and exiting to the range the camera can access (180 degree camera view)
No presenters (for 5 seconds)	Bright No obstacles	The camera should scan the area and search for a person to track	This test passed all of the time. It did a scan every 5 seconds whenever there are no detections found in the camera's view.

Chapter 6

Summary and Future work

6.1 Summary

In this project, we have addressed the limitation of the traditional ways of recording and streaming a live video to create a dynamic live stream without the added cost and complexity of camera crews and video equipment. Using two tracking cases, with the object detection model or the face detection model we were able to track the user to give the camera the ability to find the presenter during detection. The model is represented by a stand that holds the camera and is controlled by two servos that are connected with the Raspberry pi.

6.2 Future Work

With this project, we aim to make the presenter's experience as smooth, fast and fault free as possible. An enhancement for the current model's is using an accelerator to transfer the model's calculations on, which will speed up the process vastly. Another possible addition for the multiple presenters' part is to use a human activity model that can detect when a person is presenting. It will allow multiple people to be in the same camera view and still, the camera will be able to detect the person presenting rather than people walking by or sitting or doing any other activity.

Bibliography

[1] "What is object detection?" Mathworks.com, 2019. [Online]. Available: https://www.mathworks.com/discovery/object-detection.html

[2] I. Törnberg, "Real time object tracking on raspberry pi 2," Ph.D. dissertation, 06 2016. [Online]. Available: https://kth.diva-portal.org/smash/get/diva2:957920/FULLTEXT01.pdf

[3] "Aver tr530+ 30x auto tracking and live streaming ptz camera," www.averusa.com. [Online]. Available: https://www.averusa.com/products/ptz-camera/tr530plus

[4] R. Mohandas, "Tensorflow enabled deep learning model optimization for enhanced realtime person detection using raspberry pi operating at the edge?" Ph.D. dissertation, 2020. [Online]. Available: http://ceur-ws.org/Vol-2771/AICS2020_paper_61.pdf

[5] "Raspberry4," www.canakit.com. [Online]. Available: https://www.canakit.com/raspberry-pi-4-4gb.html

[6] "Raspberry pi camera board v2 - 8 megapixels :: Micro jpm," www.microjpm.com. [Online]. Available: https://www.microjpm.com/products/raspberry-pi-camera-board/

[7] "2 axis pan tilt servo brackets for sg90 servo motor mg90 servo motor - high quality platform for camera - roboelectrics - robotics electronics." [Online]. Available: https://roboelectrics.com/product/servo-brackets-servo-motor/

[8] "Gooseneck usb microphone for dictation and recording – vec gn1," American Dictation. [Online]. Available: https://www.americandictation.com/shop/accessories/microphones/vec-gn1-usb-uni-directional-gooseneck-microphone/

[9] "2-axis pan and tilt mount kit," ProtoSupplies. [Online]. Available: https://protosupplies.com/product/2-axis-pan-and-tilt-mount-kit/

[10] Santhana_krishnanMore, "Servo motor test," Instructables. [Online]. Available: https://www.instructables.com/SERVO-MOTOR-TEST/

[11] "Face detection - what is face detection? — 10 benefits," Sightcorp. [Online]. Available: https://sightcorp.com/knowledge-base/face-detection/

[12] P. Prakas and T. Nagalakshmi, "Fast and economical object tracking using raspberry pi 3.0," *International Journal of Engineering and Advanced Technology*, vol. 8, pp. 336–338, 09 2019.

[13] J. Sabhadiya, "What is servo motor?- definition, working, and types," Engineering Choice, 09 2021. [Online]. Available: https://www.engineeringchoice.com/servo-motor/

[14] "Amazon.com: Smo 5v 3a power supply adapter+on/off switch for raspberry pi 3 model b b+ us plug : Electronics," www.amazon.com. [Online]. Available: https://www.amazon.com/Power-Supply-Adapter-Switch-Raspberry/dp/B07KSXV2H8

[15] "Rpi distributions - elinux.org," elinux.org. [Online]. Available: https://elinux.org/RPi_Distributions

[16] "Frontpage - raspbian," www.raspbian.org. [Online]. Available: https://www.raspbian.org/#:~:text=Raspbian%20is%20a%20free%20operating

[17] TensorFlow, "Tensorflow," TensorFlow, 2019. [Online]. Available: https://www.tensorflow.org/

[18] "ssd_mobilenet_v1_coco documentation," docs.openvino.ai. [Online]. Available: https://docs.openvino.ai/latest/omz_models_model_ssd_mobilenet_v1_coco.html

[19] "Use cases about the v4l2 driver for the dual raspberry pi camera module," www.its404.com. [Online]. Available: https://www.its404.com/article/hnllc2012/46226489

[20] "pigpio library," abyz.me.uk. [Online]. Available: https://abyz.me.uk/rpi/pigpio/cif.html