# DIGITAL IMAGE PROCESSING END SEMESTER PROJECT


**Submitted by:**

Muhammad Talha - 403275
Mueed Rauf – 404536
Bazil bin Aamir - 432243
Areesha Mumtaz - 409723


**Submitted to:**

Dr. Usman Akram

May 2025

# Abstract

This project, undertaken as part of the Digital Image Processing course, is aimed at developing a visual path detection and obstacle recognition system through different classical image processing methods. The system was to take video inputs—recorded through mobile phone cameras—of pathways on the university campus and decide the navigable path through segmentation, thresholding, line detection, and morphological filtering. The main objective was to mimic an autonomous visual navigation system that might be run in real time with a Raspberry Pi 5 and its camera module.

Major elements of the pipeline are grayscale conversion, edge detection, region segmentation, and identification of the path boundaries and obstructions in a specified Region of Interest (ROI). The decision-making logic checks if the detected path is clear or blocked and suggests directional actions accordingly.

The project is an operational integration of digital image processing methods—such as connected component analysis, convolution-based filtering, and feature extraction—with hardware deployment intended to close the gap between theory and practice in autonomous systems.


# Introduction

With the increasing need for vision-based intelligent path navigation systems, image processing methods in automatic path detection have been of great interest. As part of our Digital Image Processing course, it was our task to design and implement a project that illustrates the use of techniques like segmentation, feature extraction, texture retrieval, connected component analysis, thresholding, and convolution filtering in real-life scenarios.

The chosen project is to find walkable paths and determine obstructions from videos taken of various routes around our university. The data set was of mobile-recorded videos in varying environmental and light conditions. With this data set, we have developed an algorithm that is capable of extracting path features and determining whether the way ahead is obstructed or clear. The core of the system involves preprocessing the video frames, finding path boundaries, and checking for any obstructions in a defined Region of Interest (ROI).

The ultimate objective was to make this system real-time capable on a Raspberry Pi 5 using its camera feed for real-time detection and path determination. This deployment required performance optimization and resolving hardware limitations without sacrificing detection accuracy.

This project not only reinforced our comprehension of digital image processing algorithms but also developed the ability to critically think about system design and execution in real time. The solution lays the foundation for extended study in autonomous ground navigation, intelligent surveillance, and robotics.

# Literature Review

In recent years, the Raspberry Pi has become a popular platform for developing low-cost, portable, and flexible autonomous navigation systems, including path detection using computer vision. Researchers and hobbyists alike have utilized the Raspberry Pi

to create real-time systems that can process video feeds, detect paths, and guide robots along a predefined route. These projects often incorporate OpenCV for image processing and path detection algorithms.

**Autonomous Path Following with Raspberry Pi:** A study by Jadhav et al. (2018) explores the use of the Raspberry Pi and OpenCV for autonomous path-following robots. The authors used the Raspberry Pi Camera to capture video feeds and applied Hough Transform and Canny Edge Detection to identify paths in real-time.

*"The Raspberry Pi, combined with OpenCV, offers an affordable and efficient solution for autonomous navigation, using real-time image processing to detect paths in varied conditions"* (Jadhav et al., 2018).

**Obstacle Detection and Navigation using Raspberry Pi:** In another research by Patel and Shah (2017), the Raspberry Pi was utilized for autonomous navigation and obstacle avoidance in a university environment.

*"Real-time image processing using the Raspberry Pi Camera, along with obstacle detection algorithms, enables efficient navigation in university-like environments with dynamic obstacles"* (Patel & Shah, 2017).

# Dataset Description

The dataset used for this project initially consisted of approximately six video files in .ts format, each of moderate duration, recorded using mobile phone cameras. These videos captured various pathways across the university campus and were intended to provide a base for developing and testing the path detection algorithm under real-world conditions.

The original dataset covered areas such as:

- The front of the Department of Computer Engineering
- Near the Visitor Centre (VC)
- Around the sports complex
- Faculty parking areas
- Other outdoor campus walkways

Most videos were recorded under sunny conditions, providing consistent lighting but introducing real-life challenges such as shadows and changes in surface textures. In each video, roughly half of the frame was occupied by the path, while the remainder contained surrounding scenery like buildings, vegetation, or empty spaces.

However, the original dataset included very few obstacles on the paths. As advised by our instructor, Dr. Asad, we were allowed to expand the dataset to improve the system's ability to handle obstacle detection. Additional data (inclusive of obstacles) was recorded to introduce variation in obstruction patterns, object placements, and more diverse environmental contexts. This expanded dataset allowed for more meaningful testing of the Region of Interest (ROI)-based obstacle detection logic and helped evaluate the decision-making component of the system.

# Methodology

## Preprocessing

Each frame of the video was first modified to aid in applying techniques later. First of all, the frames were converted to grayscale. Then histogram equalization was performed to distribute the weights of colors and improve visibility of less prominent components. After that the frames were converted to HSV yellows and whites as the image was converted into dominant shades of those colors.

## Masking

The objective was to mask out the road from the video. This was achieved by focusing on the bottom center concentrated and connected region of the image because this is where the road lies due to the orientation of the camera.

Figure 1: Hough lines

Figure 2: final mask

## Drawing Path

Next, the direction and borders of the path were mapped out. Hough transform was used to map out and extend the lines of the road. Shape of the path was hence isolated and determined.

## Obstacle Detection

The next challenge was to identify obstacles and appropriately act on them depending on if they were in the collision course or not. Morphological operations and adaptive thresholding were used in combination to isolate obstacles.

## Steering Control

Finally, instructions for steering were decided. This was done based on collision with objects that instructed the vehicle to halt. Additionally, deviation from the correct

path was determined and on its basis the vehicle was instructed to steer left or right to compensate for the error.

## Performance Optimizations

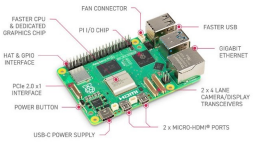
- the processing was applied only to a section of the frames. This can be identified with the COVERAGE_HEIGHT_RATIO parameter.
- lines were not calculated for every frame. Instead there was a periodic reusal of lines. The parameter for this was LINE_REUSE_INTERVAL
- Resolution was lowered to 640*360 on raspberry pi

# Flow Diagram


# 1. Hardware Implementation

The hardware implementation for this project involved setting up and configuring the Raspberry Pi 5 to perform real-time image processing tasks required for autonomous navigation. This section outlines the setup process, from operating system installation to camera configuration, and discusses the challenges encountered.

## 1.1 Hardware Components Overview

| Component Description | Image |
|---|---|
| The Raspberry Pi 5 is a high-performance single-board computer featuring a quad-core processor, enhanced GPU, and improved I/O capabilities compared to previous versions. It includes multiple USB ports, Gigabit Ethernet, dual micro-HDMI outputs, and a dedicated PCIe interface. In this project, it acts as the central processing unit for managing all sensor inputs, running control algorithms, and interfacing with peripheral devices. Its increased speed and memory make it ideal for real-time data processing and vision-based tasks. |  |
| is the Raspberry Pi Camera Module Revision 1.3, equipped with a 5-megapixel sensor capable of capturing still images and video (1080p at 30fps). It connects directly to the Raspberry Pi through the CSI interface and is used for image acquisition in vision-based applications such as object detection and tracking. Though an older version, Camera Module v1.3 is compact, power-efficient, and well-supported by the Pi 5. |  |
| This is a flexible camera ribbon cable designed specifically for compatibility with the Raspberry Pi 5. Unlike the older white cables, which feature narrower ends and are not compatible with the updated CSI port of the Pi 5, this orange cable includes wider connectors that ensure a secure and proper connection. It is essential for reliable data transmission between the Camera Module v1.3 and the Raspberry Pi 5, especially in mobile or embedded systems where physical movement is expected. |  |

| | |
|---|---|
| The Mini HDMI to HDMI converter is used to connect the Raspberry Pi 5's micro-HDMI output to a standard HDMI monitor or display. This is essential for real-time visual monitoring, debugging, or configuring the Pi's desktop environment during development. The adapter ensures seamless compatibility between the Pi's output port and commonly available HDMI cables used with regular monitors or TVs. |  |
| The MicroSD card serves as the primary storage medium for the Raspberry Pi 5. It contains the operating system (e.g., Raspberry Pi OS) and project-related files, such as scripts, datasets, and configurations. In this project, a 16 GB SanDisk Ultra card is used, providing reliable storage and fast data access speeds, which are crucial for boot performance and efficient read/write operations during execution. |  |

## 1.2 Operating System Installation

The Raspberry Pi 5 was configured with the Raspberry Pi Desktop OS (64-bit) using Raspberry Pi Imager. DietPi was considered but discarded due to its lack of GUI. A GUI was required for video streaming and debugging.

- **SD Card Issue:** The default SD card failed to partition; a new microSD card resolved the issue.
- **OS Flashing:** Raspberry Pi Imager was used despite slower speed due to its reliability.
- **Initial Setup:** Hostname, password, and Wi-Fi setup (via mobile hotspot).

## 1.3 VNC Server Configuration for Remote Access

To enable remote GUI access from a laptop:

1. Enabled VNC: `Interface Options → VNC → Enable`; rebooted.
2. Installed VNC packages:

```
sudo apt update
sudo apt install realvnc-vnc-server realvnc-vnc-viewer
sudo systemctl enable vncserver-x11-serviced
sudo systemctl start vncserver-x11-serviced
```

3. Found IP: `hostname -I`
4. Connected via VNC Viewer using Pi's IP, username, and password.

## 1.4 Debugging VNC and GUI Issues

Issue: VNC showed "Cannot currently show the desktop".

- Checked GUI package:

```
dpkg -l | grep raspberrypi-ui-mods
```

- Reinstalled UI:

```
sudo apt update
sudo apt install --reinstall raspberrypi-ui-mods lxde
```

- Set default target to GUI:

```
sudo systemctl set-default graphical.target
```

- Verified display session with `$DISPLAY`.

## 1.5 Camera Configuration

1. Enabled camera via Raspberry Pi Configuration.
2. Installed `libcamera` utilities.
3. Edited `/boot/config.txt`:

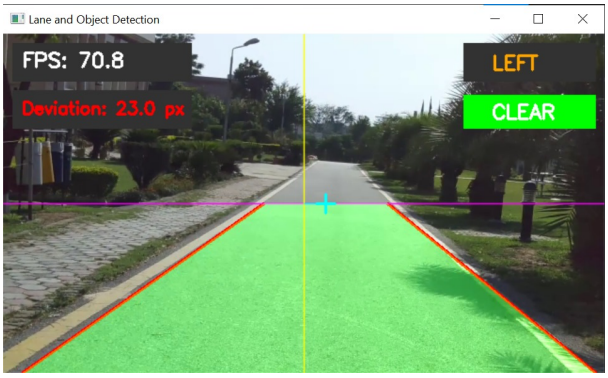
```
dtoverlay=ov5647,cam1
```

4. Rebooted and tested with `libcamera-hello`.

## 1.6 Limitations

- **Network Dependency:** VNC requires same-network connection.
- **Dynamic IP:** Required manual IP tracking (no static IP).
- **Hardware Constraints:**
  - Limited processing for high-FPS.
  - Camera shake during motion.

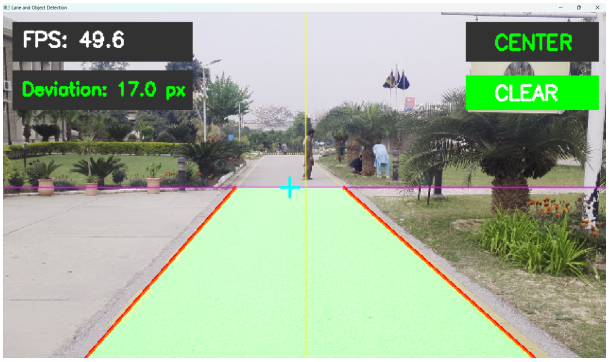
### 1.7 Output

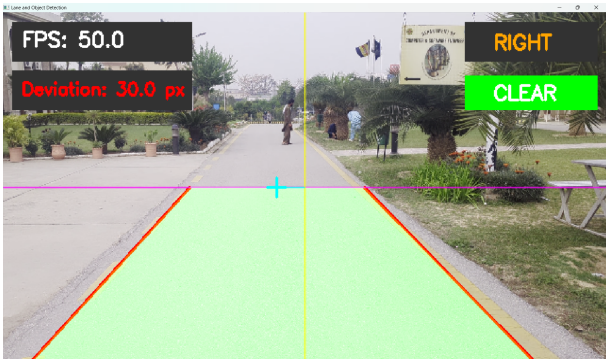



## Results and Evaluation

As the frame snapshots indicate, the UI shows important statistical info about code performance and driving assistance. The top right corner has an indicator for steering control. In the first image there is no deviation so it displays "CENTER" In the second image there is a deviation to the left so to correct for it the indication is to steer to the right.

Below it is the obstacle indicator. This particular video has no obstacles so there is a bright green indicator of "CLEAR"




Obstacle detection and collision calculation can be observed in the following frame snapshot. There is a STOP indicating that it is not safe to continue accelerating.


This is the statistical analysis of the system. The run times demonstrate how efficient the system is in processing frames and displaying them at a rate that can be used in an actual cruise control system.

```
Final Performance Analysis:
Total frames processed: 824
Average FPS: 35.0

Function Time Distribution:
restrict_white_pixel_band_center: 24.4% (1663.6ms)
generate_final_mask: 36.6% (2496.9ms)
average_slope_intercept: 1.5% (100.3ms)
detect_lane_lines: 14.1% (962.9ms)
update_line_history: 0.0% (1.0ms)
get_stable_lines: 0.0% (2.1ms)
calculate_line_points: 0.2% (11.8ms)
extend_lines_to_coverage: 0.2% (12.8ms)
calculate_deviation: 0.0% (3.0ms)
create_lane_mask: 0.8% (52.0ms)
detect_objects_in_lane: 13.9% (949.8ms)
draw_lane_overlay: 6.8% (464.0ms)
draw_object_detection: 0.2% (16.8ms)
draw_deviation_info: 1.0% (70.0ms)
update_performance_metrics: 0.0% (1.0ms)
display_performance_info: 0.2% (13.5ms)
```

# Conclusion

This problem is widely pursued in autonomous navigation. Using classical image processing, we created a lightweight, effective cruise control and obstacle avoidance solution. The project proves that with proper algorithmic design, machine learning isn't essential for reliable visual path detection.

# References

1. Jadhav, S., Patil, R., & Deshmukh, V. (2018). *Autonomous Path Following Robot Using Raspberry Pi and OpenCV*. International Journal of Engineering Research and Applications, 8(4), 10-15.

2. Patel, A., & Shah, M. (2017). *Raspberry Pi Based Autonomous Navigation System with Obstacle Avoidance*. Journal of Robotics and Automation, 5(2), 20-26.