AMRITA VISHWA VIDYAPEETHAM

Real time Person Following Robot

Inroduction to robotics &
Maths for computing-4

TEAM MEMBERS-C12

Harish S.S.	- CB.SC.U4AIE23230
Veluru Bhuvanesh	- CB.SC.U4AIE23259
Gunda Harshavardhan	- CB.SC.U4AIE23226
Avvaru Bharadwaj eswar	- CB.SC.U4AIE23210

Introduction

01 Autonomous Human Tracking

02 Adaptive Speed Control

03 Gesture based control

04 Live Monitoring & Remote Access

Problem Statement

- Traditional human-following robots struggle to accurately track a specific individual in crowded or dynamic environments.
- Color-based tracking methods (like HSV) are highly sensitive to lighting changes and background colors, leading to false detection.
- Most existing robots lack a mechanism to predict the next position of the target, reducing tracking reliability.
- There's often no balance between smooth motion control and responsive real-time tracking in basic robotic systems.
- Manual control interfaces are usually unintuitive or missing, making it hard for users to intervene or switch modes when needed

Objective

- Hybrid Control Modes – Support both automatic tracking and manual user control.
- Human detection using color.
- Smart Control Algorithms – Use Kalman Filter, ADMM, and PID for stability, smooth motion, and precise distance control.
- User-Friendly Interface – Real-time camera feed, control toggles and emergency stop via an intuitive UI.

Software Requirements

a)Computer Vision & Machine Learning

- OpenCV – Image processing for object tracking, edge detection, and filtering
- YOLOv8 – Real-time object detection to identify the human
- HSV (Hue-Saturation-Value) – Color-based tracking for additional accuracy
- cv2, numpy, socket, json, time, mediapipe, ultralytics, pandas

b) Speed control :

- PID Controller – Maintains a constant distance between the robot and the man.

c. Safety

- Ensures robot do a 360 degree turn and alert if tracking fails.- voice announcement

d**. Remote Monitoring & Communication**

- Flask (Python Web Framework) – Enables a web-based interface for live monitoring.
- OpenCV Video Streaming – Captures and transmits real-time video feed.
- WebSockets – Ensures smooth real-time data transmission.
- HTTP Requests (Flask API) – Allows remote control commands.
- Flask + OpenCV + WebSockets – For real-time camera streaming on a web app.

Hardware

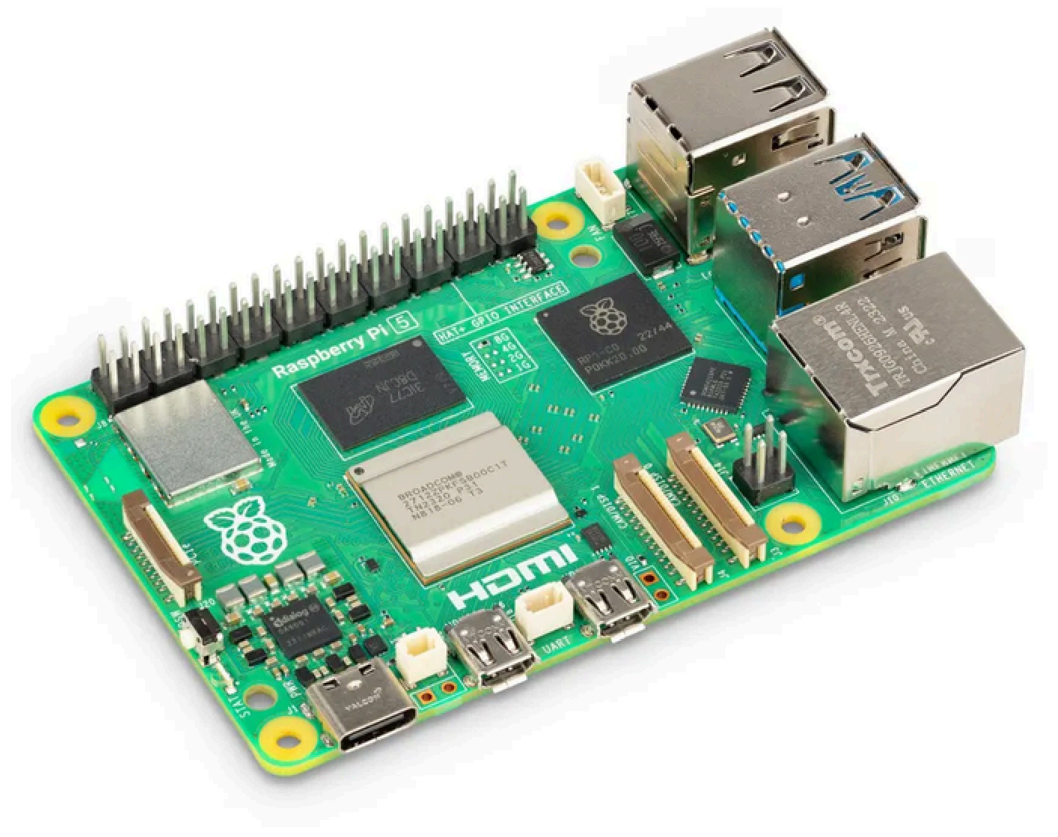
- **Processing Unit:**
  - **Raspberrypi** 5– Runs AI models (YOLO) and processes sensor data.
- **Vision & Tracking Sensors:**
  - Camera Module (web cam) – Captures real-time video for person detection
- **Motion :**
  - L298N– Controls motor speed and direction.
  - 4 x DC Motors
- **Power Supply:**
  - 12V Li-ion Battery – Provides power to motors and sensors.

Hardware Requirements

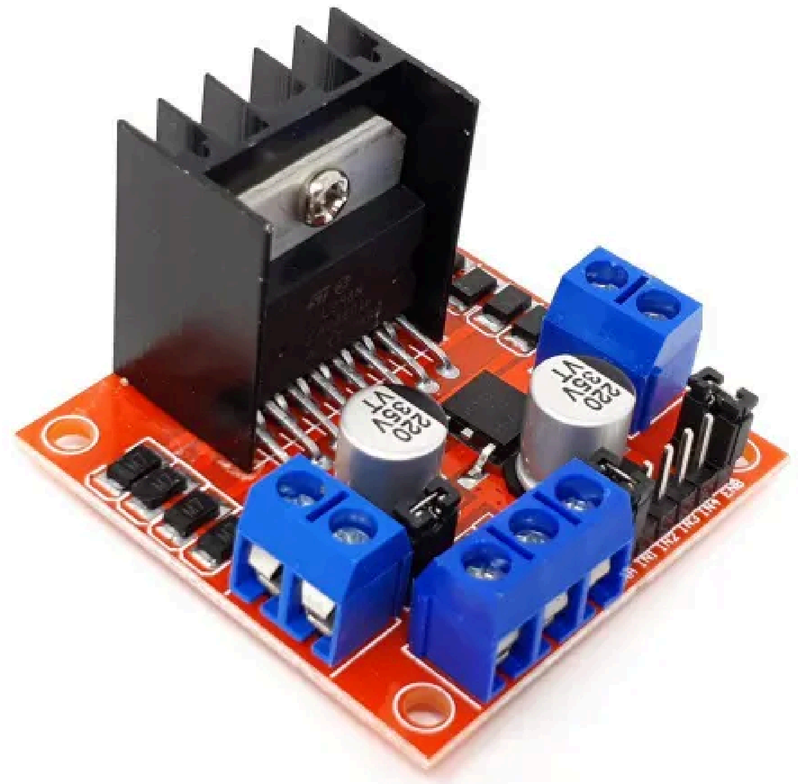
Web cam

Lithion ion battery

Hardware Requirements

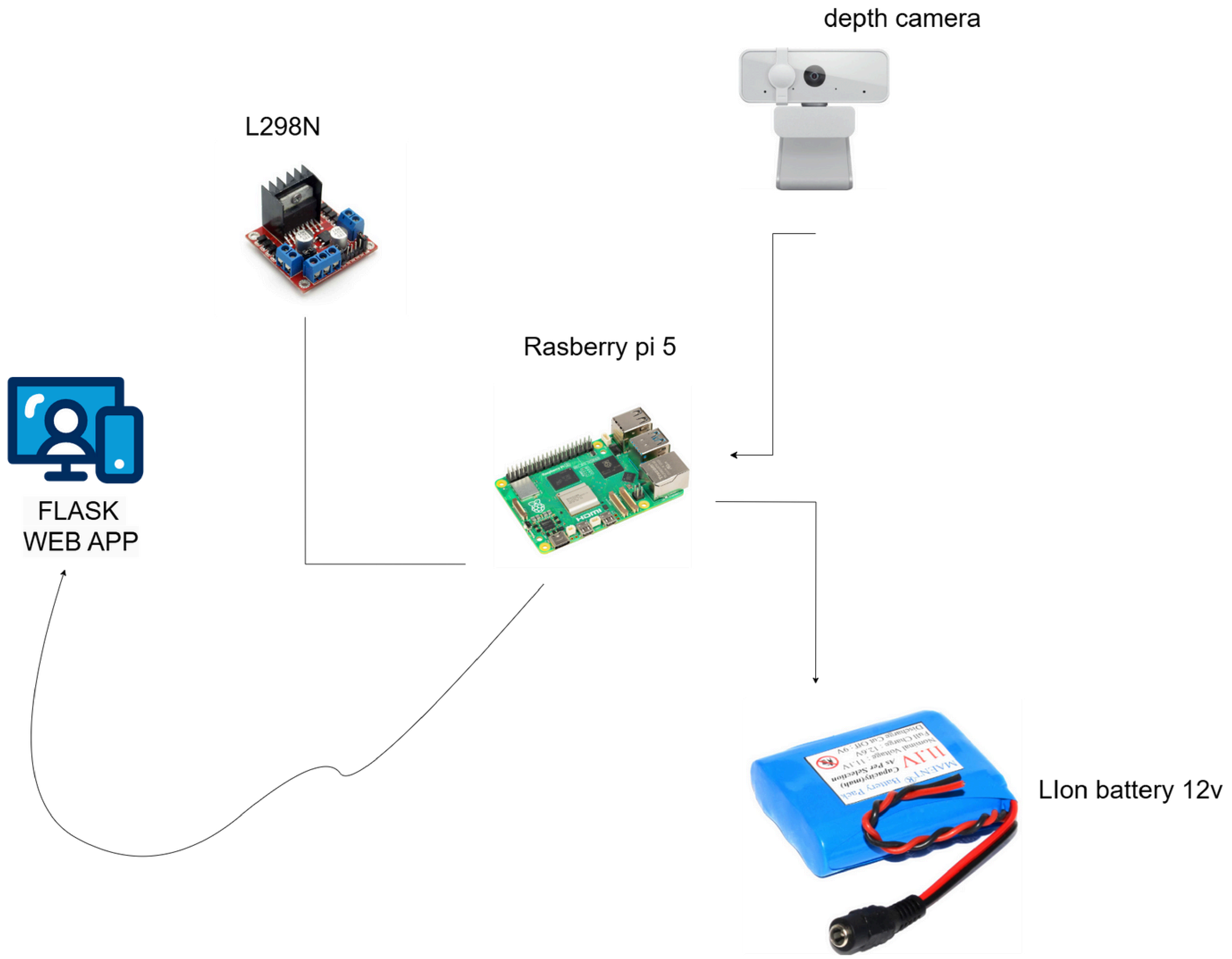


Raspberry pi 5



L298N Motor driver

Hardware architecture



Methodology

01.

Real-Time Video Capture

real time Capturing

02.

Human Detection & Control Input

HSV Filtering,HSV + YOLO,YOLO + Gesture Recognition

03.

Path Planning

Square Path

04.

KCF (Kernelized Correlation Filters)

human tracking after detceted

05.

Real-Time Streaming

Flask web app

06.

speaker alerts if tracking is los
or obstacle is too close

ALERT SYSTEM

Role of HSV + YOLO

1. Primary Role of YOLO
   - Detects people (class 0) with bounding boxes.
   - Runs at fixed intervals (every 0.5s) to reduce CPU load.
2. Role of HSV
   - Refines YOLO’s output by tracking a specific color (yellow) within detected person’s bounding box.
   - Converts the ROI (Region of Interest) to HSV space for robust color isolation.
3. Hybrid Workflow
   - Step 1: YOLO detects a person → extracts their bounding box.
   - Step 2: HSV processes the box → masks target color (yellow).
   - Step 3: Tracks the largest color blob → adjusts robot movement.
   - Fallback: If YOLO is lost, reverts to HSV for re-detection.

Role of YOLO + Gesture Control System

YOLOv8:

- Person detection (class=0) with confidence threshold (conf=0.4)
- Tracks largest detected person using bounding box area comparison
- Runs on downscaled frames (160x120) for speed

MediaPipe Hands:

- Detects 2 hands with min_detection_confidence=0.5

Gesture recognition based on:

- Relative wrist-to-finger positions (y-axis comparison)
- Finger count logic (4+ fingers + thumb = "open hand")

Control Logic:

- Follow Mode: Activated by 1 open hand gesture
- Person position → Robot steering (left/right/forward)
- Stop Mode: Activated by 2 open hands
- Deadzone: Center 1/3 of frame width (left_threshold, right_threshold)

KCF (Kernelized Correlation Filters)

KCF is a fast algorithm used to track objects in video after they've been detected.

It works by comparing the object's image from the first frame to the new frames to follow its movement.

Initial Detection

- Object (like a human) is detected using YOLO or HSV.
- KCF takes that region as a reference.

Tracking the Object

- In every new frame, KCF looks for the most similar-looking area.
- It slides the original image (template) over the new frame and compares.

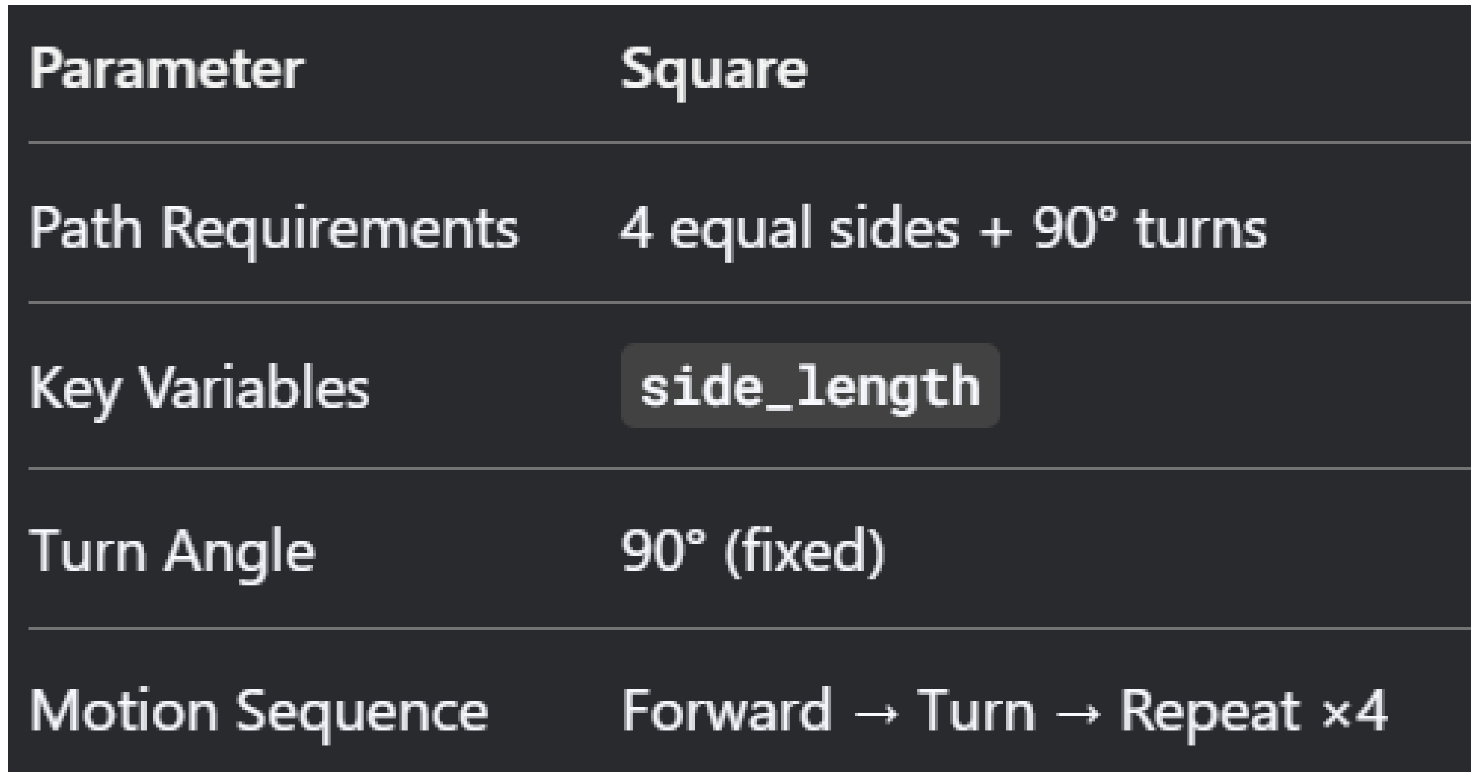
Using Kernels

- Improves matching accuracy using "kernel tricks" (math that helps match patterns better).
- Works even if the object looks a little different due to lighting or angle.

Square based path planning

Square based path planning is a simple and structured motion strategy where a robot navigates in a square trajectory.

This involves moving forward for a fixed distance (side length), followed by a 90° turn, and repeating this sequence four times to complete the square.



Parameter	Square
Path Requirements	4 equal sides + 90° turns
Key Variables	`side_length`
Turn Angle	90° (fixed)
Motion Sequence	Forward → Turn → Repeat ×4

Tracking Loss & Auto-Reacquisition System

- HSV / YOLO / KCF-based Tracking for real-time person monitoring
- Alert Triggered if person moves out of frame or tracking is lost
- Voice Alert System: Audio feedback – "Person not found!"
- 360° Rotational Scan to search and reacquire target
- Auto Resume Tracking once person is detected again

Web Streaming & Processing Synchronization

Goal:

To ensure seamless synchronization between real-time video capture and web-based viewing, allowing remote monitoring without affecting the robot’s local processing.

Working:

Camera & Video Processing

- The robot’s camera captures live video.
- Video frames are processed locally for tracking and movement decisions.

Web Streaming Integration

- The processed video is streamed via a Flask web server.
- The web interface continuously fetches and displays real-time video.

Parallel Execution

- The robot's movement and tracking run independently of web streaming.
- Web streaming is only for visualization, ensuring no delay in decision-making.

Mathematics

YOLO Bounding Box Calculation (For Person Detection)

- YOLO is a real-time object detection algorithm that detects multiple objects (like humans) in an image using a single forward pass through a convolutional neural network (CNN).

- Once the person is detected, the bounding box center is used to control the direction the robot should move (e.g., turn left, go forward).

  $B=(x,y,w,h)$
- $x,y \rightarrow$ Top-left coordinates of the detected person
- $w,h \rightarrow$ Width & height of the bounding box

- The box size is used to estimate the distance to the person — larger box = person is closer.

Mathematics

ADMM

- ADMM (Alternating Direction Method of Multipliers) is an optimization algorithm that solves complex problems by breaking them into smaller, easier sub-problems.

- It is especially effective for convex optimization problems with constraints and works well in distributed systems.

- ADMM optimizes the robot's movement by minimizing sudden changes in position, leading to smoother motion and better user experience.

We aim to minimize a cost function like:

$$\begin{aligned} &\text{minimize} \quad f(x) + g(z) \\ &\text{subject to} \quad Ax + Bz = c \end{aligned}$$

Mathematics

KCF (Kernelized Correlation Filters)

- KCF is a fast and efficient tracking algorithm that uses correlation filters in the frequency domain.

- It tracks a selected object across video frames by learning how it looks and finding it again in subsequent frames.

- KCF trains a classifier (correlation filter) to distinguish the object from the background using a single positive sample and many shifted negative samples.

- The filter is trained by minimizing:

  $\min \| \varphi(x) * \alpha - y \|^2 + \lambda \|\alpha\|^2$

Where:

- $\varphi(x)$: Feature map of input image patch
- α: Filter in dual space
- y: Desired output (usually a Gaussian peak centered on the object)
- λ: Regularization parameter

Mathematics

PID Controller for Distance Maintenance

Error Calculation (Distance Control)

e(t)=d_desired−d_current

- e(t)→ Error (difference between desired & actual distance)
- Robot increases/decreases speed based on this error

PID Control Equation (Adjusting Speed Dynamically)

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

- Kp → Proportional gain (reacts to current error)
- Ki → Integral gain (fixes accumulated errors)
- Kd → Derivative gain (prevents overshooting)

Mathematics

Position Tracking (Kalman Filter)

State Representation:
represents the position and velocity in 2D.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

State Transition Matrix (F):
Assuming constant velocity, the future state

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Observation Matrix (H):
Only position (not velocity) is directly obser

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Prediction Step:

$$\hat{\mathbf{x}}^- = \mathbf{F}\hat{\mathbf{x}}$$
$$\mathbf{P}^- = \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q}$$

Where:

$\hat{\mathbf{x}}^-$: predicted state

$\mathbf{P}^-$: predicted covariance

$\mathbf{Q}$: process noise

Mathematics

Update Step:

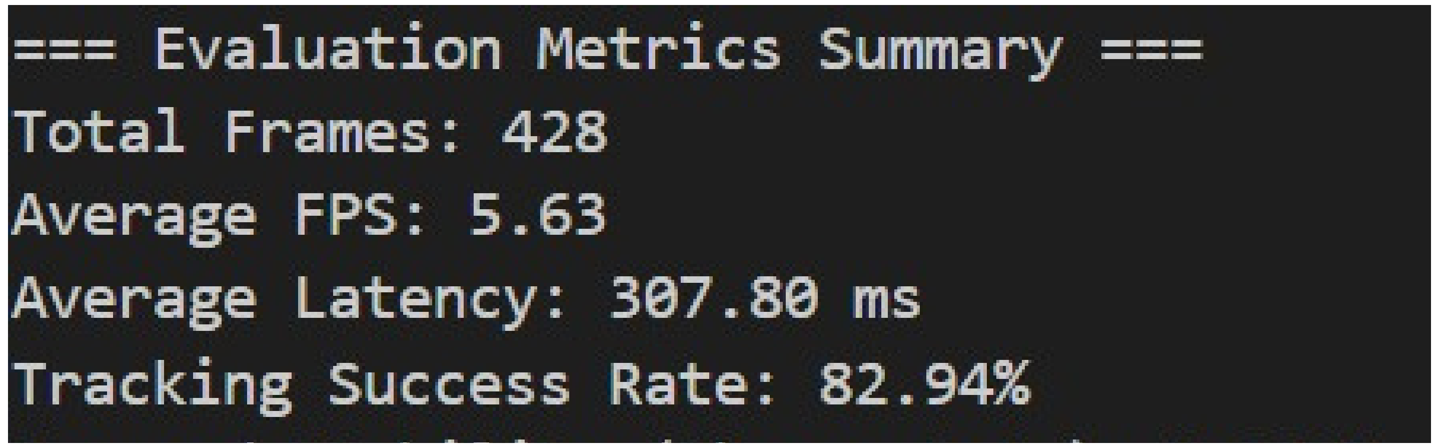
- z: observation (detected person's position)
- R: observation noise covariance

$$\mathbf{K} = \mathbf{P}^{-}\mathbf{H}^{T}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{T} + \mathbf{R})^{-1} \quad \text{(Kalman Gain)}$$
$$\hat{\mathbf{x}} = \hat{\mathbf{x}}^{-} + \mathbf{K}(\mathbf{z} - \mathbf{H}\hat{\mathbf{x}}^{-})$$
$$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}^{-}$$

- This recursive process predicts smoother and more accurate positions of the person, even with noise or momentary detection loss.

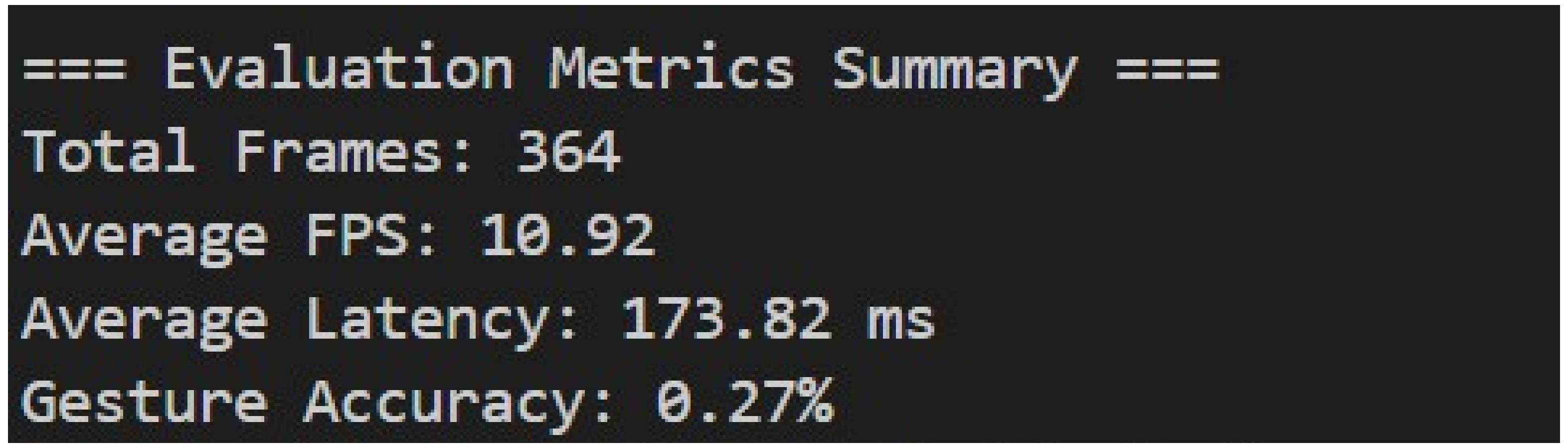
Comparison Mertrics

Lateral offset- 6 degrees for 105 cm



```
=== Evaluation Metrics Summary ===
Total Frames: 428
Average FPS: 5.63
Average Latency: 307.80 ms
Tracking Success Rate: 82.94%
```

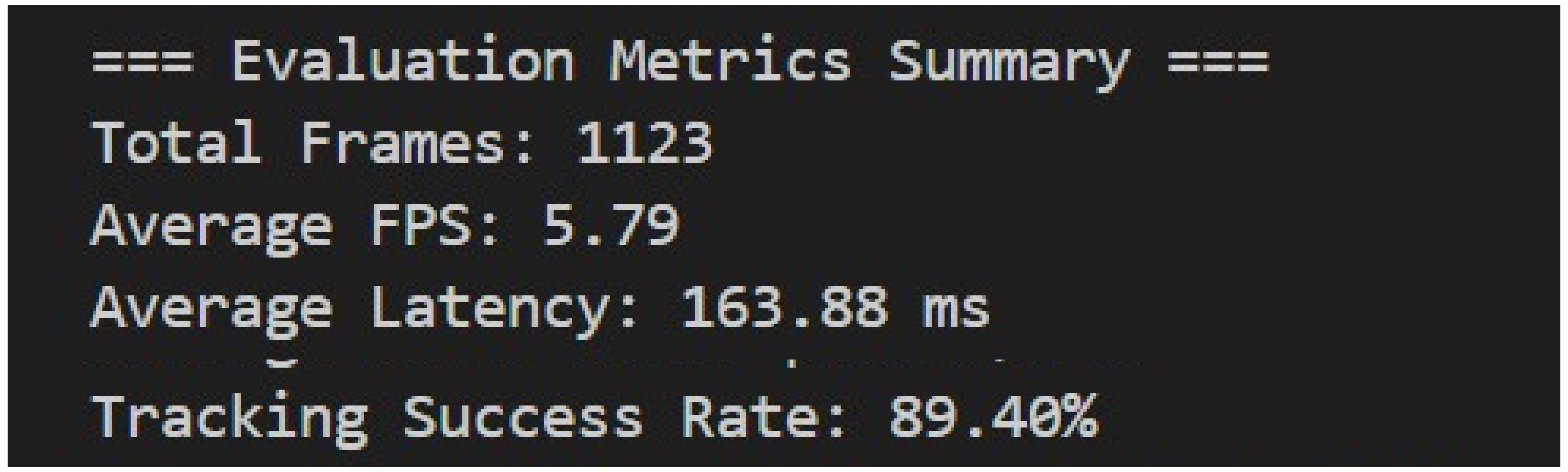
YOLO+HSV



```
=== Evaluation Metrics Summary ===
Total Frames: 364
Average FPS: 10.92
Average Latency: 173.82 ms
Gesture Accuracy: 0.27%
```

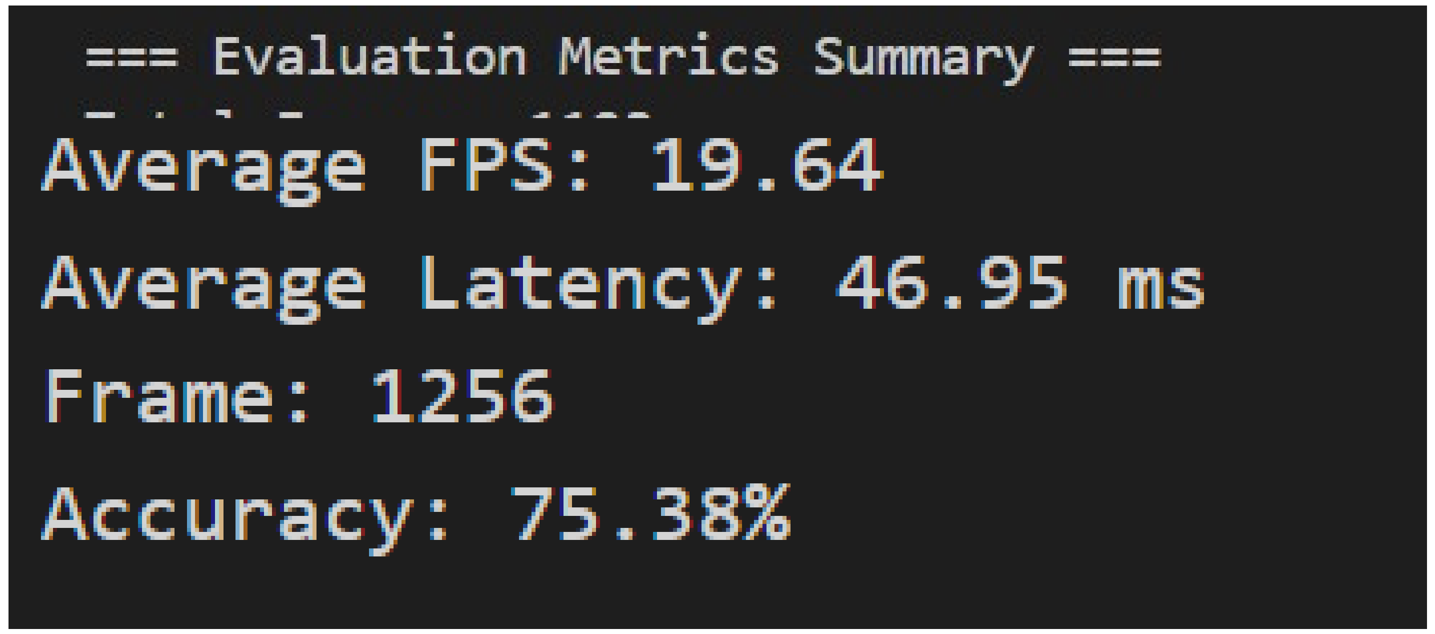
Meediapipe + YOLO

Comparison Mertrics



```
=== Evaluation Metrics Summary ===
Total Frames: 1123
Average FPS: 5.79
Average Latency: 163.88 ms
Tracking Success Rate: 89.40%
```

Kalman



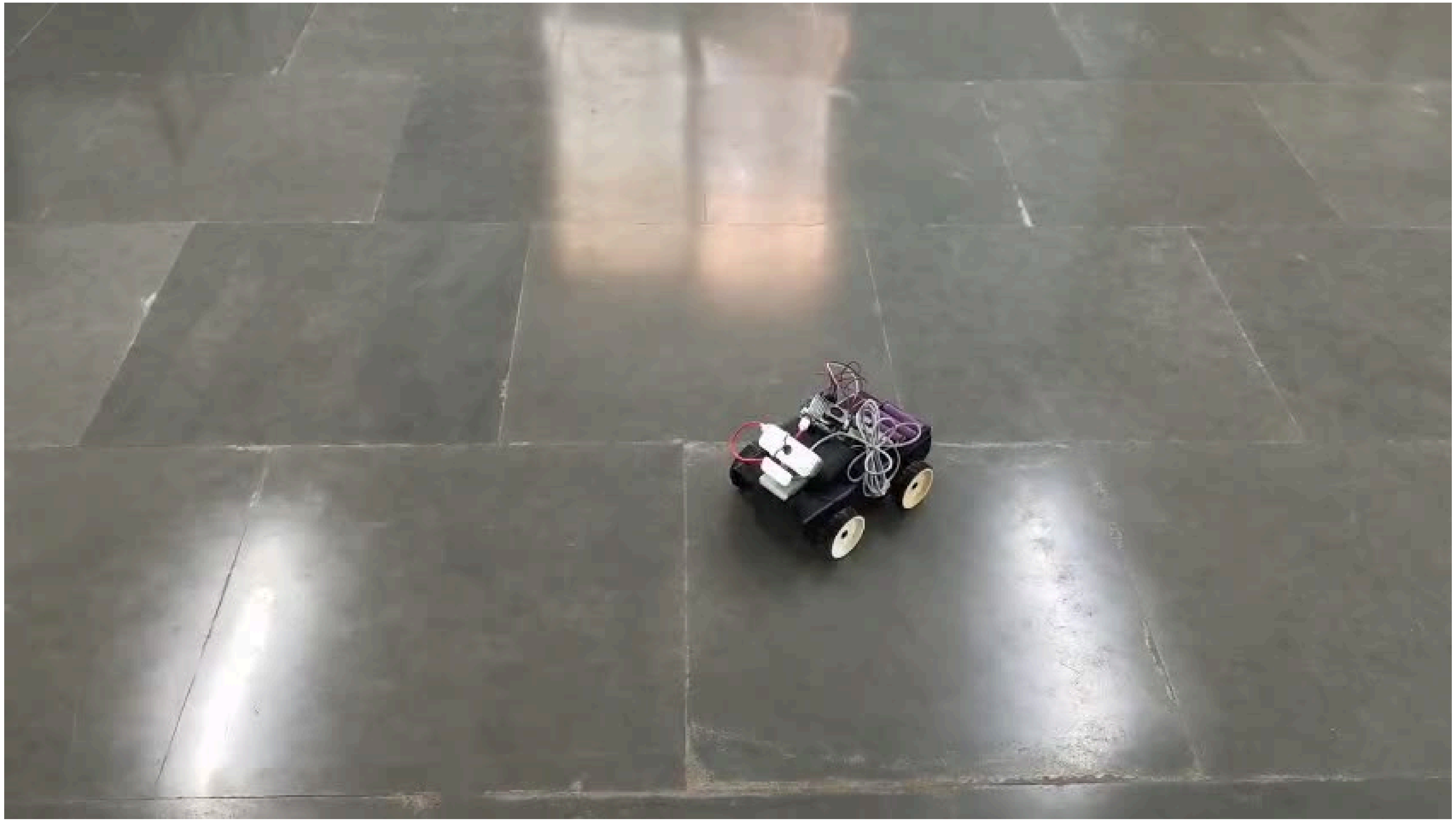
```
=== Evaluation Metrics Summary ===
Average FPS: 19.64
Average Latency: 46.95 ms
Frame: 1256
Accuracy: 75.38%
```

HSV

Results

Gesture control

Results



Square path following

Results

HSV

Results

Yolo+HSV

Conclusion

- This project presents a robust and intelligent human-following robot that combines computer vision and control algorithms for reliable person tracking.
- By integrating HSV, YOLO, and gesture recognition techniques, the robot accurately detects and follows a target in real time.
- The use of a Kalman Filter enhances position prediction, while ADMM ensures smooth trajectory generation and PID controls the robot's distance from the person.
- The system supports both manual and autonomous modes with a user-friendly interface, making it adaptable to different environments and users.

Thank you