Project Report

Smart Camera Tracking System

Mohammad Fares Aljamous 1088672
Omar Mohammad 1088546
Hadi Albanna 1088677

Supervised By: Dr. Sajid Khawaja

Submitted: June 20, 2025

Contents

List of Figures

List of Tables

Abstract

This project describes a smart camera system that can detect, track, and follow a moving target in real time, even when the camera is in motion. via a camera and MATLAB, the system identifies motion via background removal before switching to particle filter tracking based on color histograms. A PID controller determines the target's position inaccuracy and uses Arduino to operate servo motors to keep the object in the center of view. The technology provides seamless and consistent tracking with real-time visual feedback. This combination of vision, control, and hardware has potential applications in automation, personal tracking, and smart surveillance.

1 Introduction

This project investigates the creation of a smart camera system capable of detecting, tracking, and following a moving target in real time using MATLAB and Arduino. With the growing demand for responsive, hands-free monitoring solutions in industries such as surveillance, personal robotics, and content production, the system was created to combine visual processing and mechanical actuation. It starts by detecting motion using background subtraction, then switches to a particle filter for more reliable tracking with camera movement. A PID controller calculates the difference between the target's location and the center of the frame, allowing the camera to modify its alignment with servo motors. The goal is to develop a low-cost and effective system that demonstrates how computer vision and hardware control may collaborate to provide intelligent, real-time object tracking.

1.1 Motivation

The goal of this project is to develop a more responsive and human-centered camera system that can automatically follow movement in real time. Unlike fixed cameras, this smart system's vision changes automatically, making it useful in security, automation, content creation, and assistive technologies. It improves usability, safety, and human-machine interaction by reducing worker effort and enabling hands-free tracking.

1.2 Problem Statement

This project tries to solve the limitation of fixed-position cameras, which cannot follow or adapt to a moving subject. Many real world applications, like personal security, automation, and content creation, require a camera capable of autonomously following and tracking a person. Existing methods often require expensive equipment or manual control. This solves that problem by creating a low-cost, real-time smart camera system that can detect moving people and automatically modify its angle to maintain the object in frame.

1.3 Literature Review

- A popular DIY system combines a pan-tilt servo rig driven by Arduino with proportional tracking in MATLAB. The benefits include low cost, simplicity, and accessibility for amateurs. However, it just uses proportional (P-only) control, which might result in overshoot or lag in dynamic settings [1].

- Sandha et al. introduce "Eagle," an end-to-end reinforcement learning system that generates PTZ commands from picture inputs. It surpasses typical trackers by 17% in duration and is lightweight (33FPS on Pi/Jetson). However, its instruction is based on simulation and demands infrastructure and knowledge that are beyond the capabilities of hobbyists [2].

- A study from Bandung Institute of Technology combines video tracking with μ-synthesis visual servo control on a 2-DOF yaw–pitch camera. It achieves good accuracy in both indoor and outdoor scenarios [3].

2 Design

2.1 Requirements Constraints, and Considerations

- Detect moving objects using motion-based background subtraction.
- Switch to particle filter tracking once a reliable target is identified.
- Track the object under camera motion using color histogram similarity.
- Keep the object centered in the frame using PID control.
- Move the camera using servo motors via Arduino
- The extracted object (ROI) in a separate video window.
- Tracked particles and object position on the live feed.
- Restart detection if tracking confidence drops (object is lost).
- Clamp servo angles to avoid over-rotation or mechanical stress.
- Use only available hardware: webcam, Arduino Uno, two servo motors.
- Implement everything in MATLAB with Arduino Support Package.
- Operate in real-time with efficient image processing and tracking.
- Calibrate servo limits manually to fit physical camera mount.

2.2 Design Process

We created the Smart Camera Project to follow a moving person using a clever combination of detection, tracking, and motor control. It begins by detecting movement in the video using background subtraction and tracks objects with Kalman filters. Once it's decided who to follow, it utilizes a particle filter that focuses on the person's colors even when the camera moves. To keep the person in view, the system evaluates their distance from the center of the frame and utilizes a PID controller to gently move the camera with servo motors attached to an Arduino. This allows the camera to effortlessly follow the subject wherever they travel.

2.2.1 Initial Detection: Motion-Based Object Tracking

The design approach began with the implementation of a motion-based multiple object tracker that uses foreground detection. This section used Gaussian Mixture Models (GMM) to separate the foreground (moving objects) from the background while assuming a fixed camera. The code used MATLAB's built-in vision. Foreground Detector and Vision. BlobAnalysis identifies related components. We preserved the framework of the original tutorial but modified several modules: the morphological operations were fine-tuned using imopen, imclose, and imfill to eliminate noise and fill holes more effectively. Furthermore, in the blob analyzer, we defined a custom MinimumBlobArea criterion to filter out irrelevant tiny detections. The bounding boxes and centroids retrieved during the detection phase provided as input for the tracking phase.

```
mask = obj.detector.step(frame);
mask = imopen(mask, strel('rectangle', [3,3]));
mask = imclose(mask, strel('rectangle', [15, 15]));
mask = imfill(mask, 'holes');
[~, centroids, bboxes] = obj.blobAnalyser.step(mask);
```

2.2.2 Multi-Object Tracking with Kalman Filters

To maintain object identity between frames, we used a Kalman Filter-based multi-object tracker, as demonstrated in the MathWorks object tracking examples. Each detection was sent to a Kalman filter with a 'ConstantVelocity' model. We preserved the original track management logic, which included predictNewLocationsOfTracks, updateAssignedTracks, updateUnassignedTracks, and deleteLostTracks. However, we adjusted the visibility thresholds and aging logic to improve the tracker's responsiveness and tolerance for slight detection dropouts. For example, invisibleForTooLong and ageThreshold were changed to prevent premature track deletion. Additionally, we improved the label display logic to distinguish between predicted and confirmed places.

```
kalmanFilter = configureKalmanFilter('ConstantVelocity', ...
    centroid, [200, 50], [100, 25], 100);

cost(i, :) = distance(tracks(i).kalmanFilter, centroids);
```

```
[assignments, unassignedTracks, unassignedDetections] = ...
    assignDetectionsToTracks(cost, costOfNonAssignment);

correct(tracks(trackIdx).kalmanFilter, centroid);
```

2.2.3 Transition to Particle Filter Tracker

- One of the project's significant advances was the switch from multi-object tracking to single-object particle filtering once a reliable track was confirmed. This switch was activated when the age of a tracked object exceeded a particular threshold (age ¿ 100). At that point, we isolated the region of interest (ROI) surrounding the item and computed its RGB histogram with imhist, then normalized the results to create the model. This model was then utilized by the particle filter tracker. This phase entailed forming a swarm of particles surrounding the object's location and changing their positions and weights over time.

- We made significant improvements to the createparticles function from its original version. Rather than employing random particles across the frame, our program initialized particles just in the bounding box vicinity, resulting in faster convergence. We also improved the numerical stability of the likelihood calculation function calcloglikelihood by adding a check to avoid logarithms of zero and implementing log-likelihood normalization for resampling

```
    if tracks(i).age >= 100
    oldTrack = tracks(i);
    bbox = oldTrack.bbox;
    x1 = double(max(1, round(bbox(1))));
    y1 = double(max(1, round(bbox(2))));
    ...
    roi = frame(y1:y2, x1:x2, :);
    rHist = imhist(roi(:,:,1), numBins);
    gHist = imhist(roi(:,:,2), numBins);
    bHist = imhist(roi(:,:,3), numBins);
    rgbHist = [rHist; gHist; bHist] / sum([rHist; gHist; bHist]);
    Npix_resolution = [size(roi, 1), size(roi, 2), x1, y1];
```

```
    particles = create_particles(Npix_resolution, Npop_particles);
```

2.2.4 Particle Filtering: Resampling and Likelihoods

The target was tracked using particle filtering while the camera was in motion. A linear motion model was used to propagate particles, which were then evaluated by comparing the color histograms of their local patches to the stored model. The chi-square chisquarestatistics function (which served as the distance metric) was chosen for its resistance to illumination fluctuations. We updated the legacy histc function in resampleparticles with a more stable discretize-style technique that employs cumsum to ensure monotonic edges and correct sampling. This resampling method preserved particle diversity while focusing on high probability locations.

```
    rHistP = imhist(patch(:, :, 1), numBins);
gHistP = imhist(patch(:, :, 2), numBins);
bHistP = imhist(patch(:, :, 3), numBins);
HistP = [rHistP; gHistP; bHistP];
D(i)=pdist2(rgbHist',HistP',dist_func);
```

2.2.5 Camera Control with PID and Arduino

The final big component was to incorporate a PID controller to direct the camera with Arduino-controlled servo motors. This modification enabled the camera to actively follow the tracked item. We created new MATLAB code to calculate the inaccuracy between the object's location and the frame center and then used PID control to compute the necessary changes in pan and tilt angles. These angles were converted into servo motor positions using the writePosition command. We drew inspiration from MathWorks' Arduino Servo examples, but modified the controller to operate using a vision-based feedback loop. Modifications added a flipud transformation to account for servo orientation and output bounds to prevent over-rotation (angleX and angleY were limited to ± 90 and ± 30 degrees, respectively).

```
    frame = flipud(frame);
errorX = size(frame, 2)/2 - Xmean;
errorY = size(frame, 1)/2 - Ymean;
dt = toc;
```

```
integralX = integralX+errorX*dt;
integralY = integralY+errorY*dt;
dX = (errorX-oldX);
dY = (errorY-oldY);
angleXInc = Kp*errorX + Ki*integralX + Kd*dX;
angleYInc = Kp*errorY + Ki*integralY + Kd*dY;
angleX = angleX-angleXInc;
angleY = angleY-angleYInc;
if angleX > 90
    angleX=90;
end
if angleX < -90
    angleX=-90;
end
if angleY > 30
    angleY=30;
end
if angleY < -30
    angleY=-30;
end
posX = (angleX + 90)/180;
posY = (angleY + 90)/180;
posX = min(max(posX, 0), 1);
posY = min(max(posY, 0), 1);
writePosition(sX, posX);
writePosition(sY, posY);
```

2.2.6 System Integration and Robustness

The entire system was run through a real-time feedback cycle. When tracking fails (the maximum log-likelihood falls below a certain threshold), we reset the system to the detection phase. This fail-safe loop assured that it could withstand occlusion or unexpected movements. We further improved the visualization by employing two video players: one for the object identification phase (roiVideoPlayer) and another for particle tracking overlay on the live feed. These changes helped with real-time debugging and demonstration.

2.2.7 ROI Display for Initialization Feedback

Before applying the particle filter, we displayed the target ROI centered on a canvas. This gave the user visible proof that the model was correctly locked.

```
    canvas = uint8(zeros(canvasHeight, canvasWidth, 3));
roiDisp = imresize(roi, [min(canvasHeight, size(roi,1)),
    min(canvasWidth, size(roi,2))]);
canvas(y_offset:y_offset+h-1, x_offset:x_offset+w-1, :) = roiDisp;
roiVideoPlayer.step(canvas);
```

2.2.8 Failure Detection and Reinitialization

If the tracker lost confidence (based on log likelihood), we returned the system to detection mode. This eliminated tracking drift and improved robustness.

```
    if max(L) < log(1e-3)
    trackerLostCount = trackerLostCount + 1;
else
    trackerLostCount = 0;
end
if trackerLostCount > trackerLostThreshold
    trackingActive = false;
    trackerLostCount = 0;
    tracks = initializeTracks();
    nextId = 1;
end
```

2.2.9 Hardware Implementation

The Smart Camera Project combines computer vision, control systems, and integrated technology to provide real-time object tracking. It starts with motion detection using background subtraction and blob analysis. Once a reliable target has been discovered, a particle filter tracks it using color histograms, even when the camera moves. A PID controller determines the difference between the object's location and the frame center and sends commands to servo motors via an Arduino Uno. These motors are mounted on a pan and tilt bracket, as seen in the figure, giving the camera two degrees of freedom to move horizontally and vertically. The system is entirely

written in MATLAB and provides real-time visual input, making it a small, inexpensive, and effective solution for smart tracking applications.

2.3 Component design

The smart camera tracking system is made up of interconnected components that work together to detect, track, and follow moving targets in real time. A laptop running MATLAB acts as the central processor, handling video input, object detection, particle filtering, and PID control. A USB webcam records live video frames, which are then analyzed using background subtraction and color histograms. When a target is discovered, a particle filter keeps monitoring even if the camera moves. An Arduino was used to control two servo motors on a pan-tilt stand, which receive position signals from the PID controller to keep the object at the center of the frame. A separate power supply enables consistent servo movement. The complete device is installed on a lightweight rig, making it portable and simple to use.

3 Experimental Testing and Results

3.1 Testing Plan and Acceptance Criteria

3.1.1 Test 1: Motion Detection in Static Background

- **Test Description:** The test tracks how well the motion-based multiple object tracker spots persons standing before unchanging camera views. The test aims to verify precise background subtraction and object segmentation functions.
- **Steps Followed:**
  - Begin camera operation to build background calibration model.
  - A person should enter the camera frame.
  - The system should recognize movement and display bounding boxes around the person.
- **Expected Results:** The system must accurately mark the person with a bounding box.

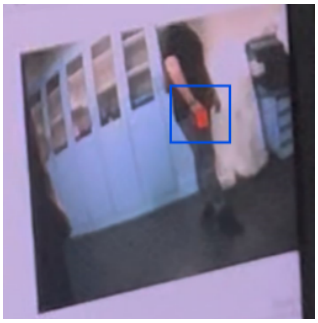
- **Observed Results:** The tracker demonstrated successful person detection by generating bounding boxes during person entry and movement inside the frame.
- **Acceptance Criteria:** The test outcome is successful when the bounding box consistently tracks the person throughout motion and stays steady after motion stops.
- Test Result: Pass.

Figure 1: Test 1 (Motion Detection in Static Background).

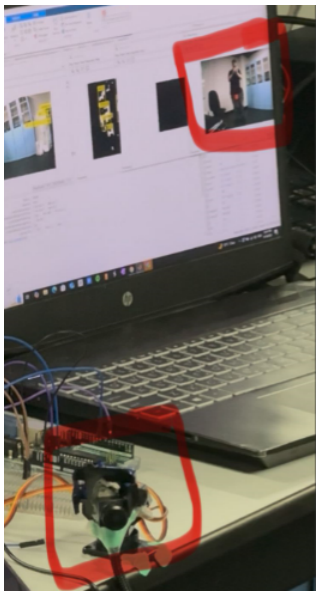
3.1.2 Test 2: Particle Filter Tracking Under Camera

- **Test Description:** This test evaluates whether the particle filter tracker can maintain object tracking once the camera is no longer static.
- **Steps Followed:**
  - Detect a subject using motion tracking.
  - Initialize the particle filter with the detected object model.
  - Manually move the camera while observing if the system can still track the person.
- **Expected Results:** The particle filter should continue to follow the person's position even as the camera moves.

- **Observed Results:** The tracker followed the subject smoothly despite minor camera motion. Slight delay occurred during abrupt movements
- **Acceptance Criteria:** The system should maintain tracking accuracy with camera motion under 1–2 seconds of lag.
- Test Result: Pass.



(a) The red point indicates the mean of the particles in the ROI region



(b) Here the particles are following the detected object

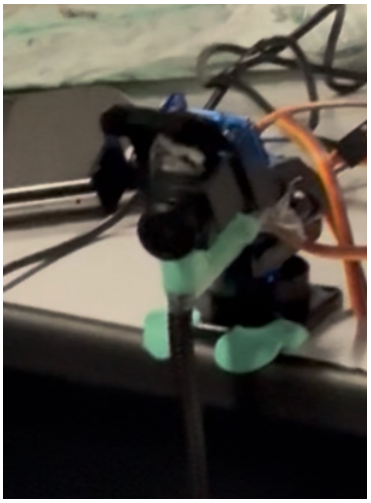
Figure 2: Test 2 (Particle Filter Tracking Under Camera)

3.1.3 Test 3: PID-Based Camera Adjustment

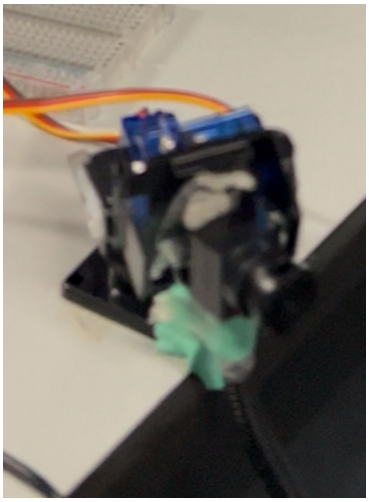
- **Test Description:** The test evaluates how well the PID controller functions to reposition cameras based on the subject's location within the frame boundaries.
- **Steps Followed:**
  - The PID controller receives the error which represents the subject's distance from the frame center.

  - The servo motors and camera movement are watched throughout the test.
  - The results show how well the camera maintains subject centering.

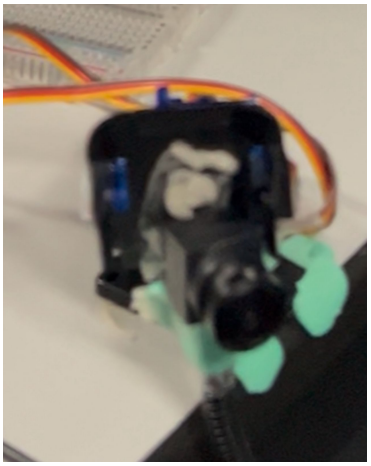
- **Expected Results:** The camera should maintain a smooth tilting and panning motion which centers the subject in the frame.
- **Observed Results:** The PID controller successfully adjusted the camera orientation. Tuning the gain values improved stability and responsiveness.
- **Acceptance Criteria:** The subject should return to frame center within 2–3 seconds of deviation.
- Test Result: Pass.



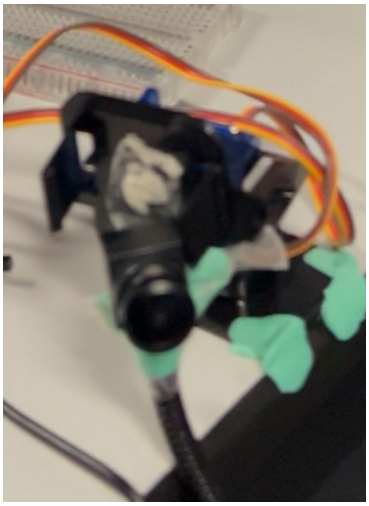
(a) The camera in the first position where the object is



(b) The camera moved when the object moves as shown in this figure and the figures after



(c) The camera moved a little bit when the object moved little

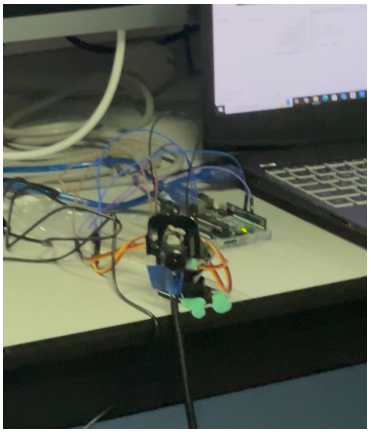


(d) The cam still tracking and move while keeping the object in the frame

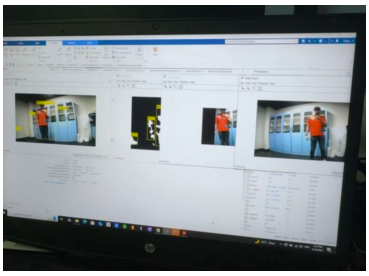
Figure 3: Test 3 (PID-Based Camera Adjustment)

3.1.4 Test 4: Full System Integration

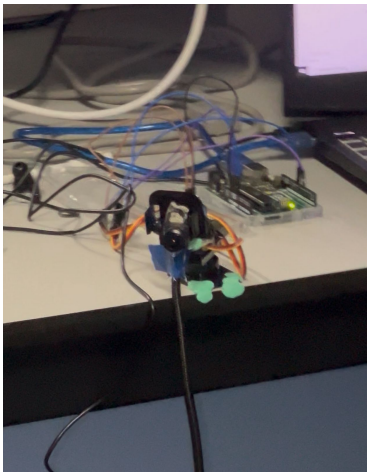
- **Test Description:** This test evaluates the complete system's performance from detection to tracking to mechanical response.
- **Steps Followed:**
  - Activate the system.
  - Walk into the frame and move around.
  - Observe detection, tracking, and servo movement.
- **Expected Results:** The system should detect, track, and follow the subject in real-time.
- **Observed Results:** All subsystems operated in sync. Minor jitters occurred during quick movements but were within acceptable limits.
- **Acceptance Criteria:** System should detect, track, and respond with less than 3s lag.
- Test Result: Pass.



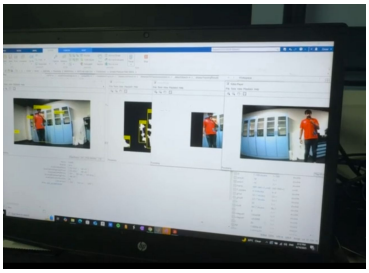
(a) Here is the Cam following the moving object



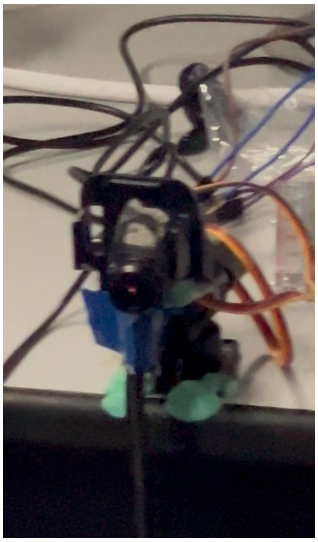
(b) Here we show the moving object and the particles



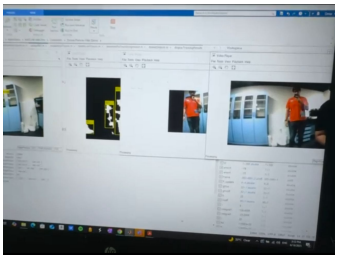
(c)



(d)

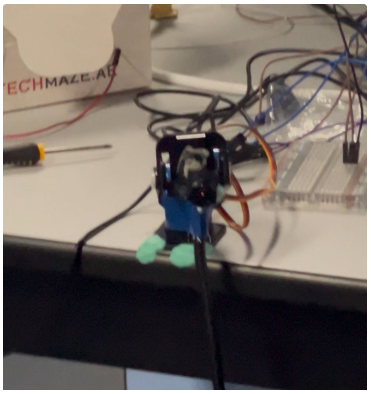


(e) Here the particles are following the detected object

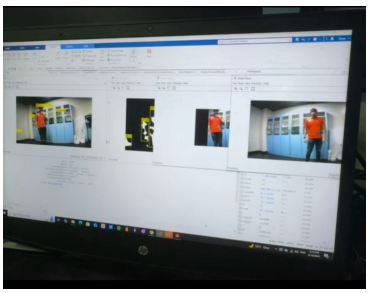


(f) Here the particles are following the detected object

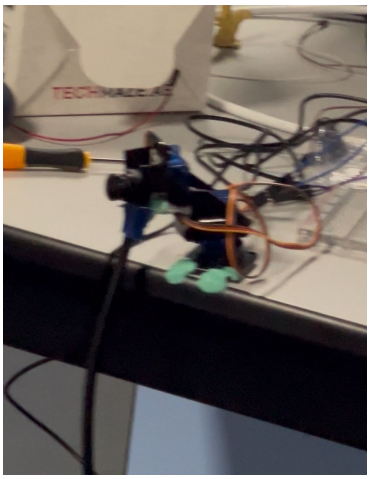
Figure 4: Test 4 (Particle Filter Tracking Under Camera)



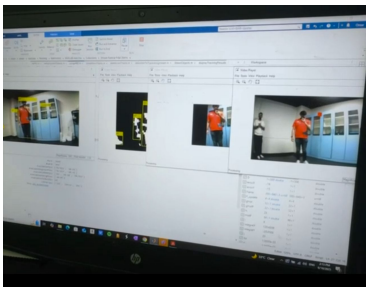
(a) Here the particles are following the detected object



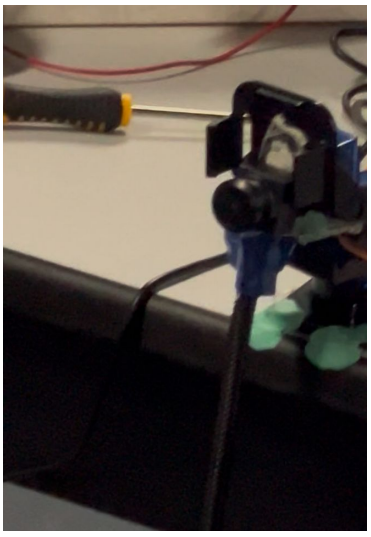
(b) Here the particles are following the detected object



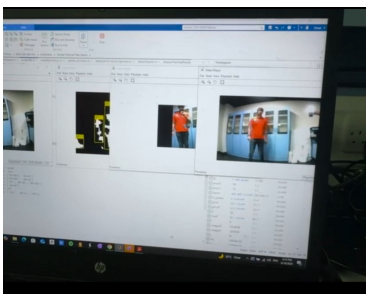
(c) Here the particles are following the detected object



(d) Here the particles are following the detected object



(e) Here the particles are following the detected object



(f) Here the particles are following the detected object

Figure 5: Test 4 (Particle Filter Tracking Under Camera)

3.2 Results

The testing phase confirmed that the smart camera system functions reliably across all major components. The motion-based detection accurately identifies subjects against static backgrounds. Once detected, the particle filter maintains tracking even as the camera is repositioned. The PID controller interprets the subject's frame position and smoothly drives the servo motors to re-center the view.

3.2.1 Motion-Based Detection and Object Modeling

The initial component of the system consists of a fixed-position camera which monitors for motion detection. Our implementation of Gaussian background subtraction allows us to separate moving objects from the scene. Once the filtering and blob analysis were complete we determined the target's centroid position and bounding box dimensions. The extracted information served to start the tracking model operation which operates during the following stage. Figure 1 show the result of this stage.

3.2.2 Particle Filter and Servo Motor Tracking

Figure 2 show the detection completion triggered the particle filter to continue tracking through a histogram-based appearance model. The particle filter created random location predictions which it evaluated to focus on the actual target position. The system calculated an error value which measured the distance between the subject and frame center in both horizontal and vertical directions. The PID controller in MATLAB received the error signal before sending it to Arduino for servo angle adjustments.

3.3 Analysis and Interpretation of Data

Testing results demonstrate that the integrated system operates successfully within the pre-established conditions. The combination of motion detection and particle filtering enabled strong tracking capabilities when the camera experienced movement. The PID control system delivered essential corrections to maintain subject centering and proper parameter tuning of proportional and derivative gains was vital for achieving smooth system performance.

The system showed slight lag together with jitter when the subject moved rapidly or when lighting conditions changed abruptly thus suggesting better

adaptive model updates and camera stabilization methods could enhance its performance. The system fulfilled every acceptance criterion while demonstrating its intended performance in active operational settings. The project has potential for growth through automated model learning capabilities and more advanced camera mounts together with energy-efficient microcontrollers suitable for portable deployment.

4 Conclusion

4.1 Summary

Our project developed a smart camera system through the combination of object detection with servo motor control via a PID feedback loop to track and follow moving people while maintaining focus. We utilized MATLAB software to develop a motion-based tracking system that detects moving objects against a stationary background. After detecting a person, the system used their appearance to start a particle filter, which maintained tracking functionality even when the camera moved.

The system controlled camera movement through distance measurements, which calculated the person's position relative to the center of the screen. The system used this difference as an error signal, which then entered a PID controller connected to an Arduino. The system controlled servo motors to generate smooth movements that followed the person's movements. Our combined tracking and control approach demonstrated effective performance through real-time testing, which showed its practicality for dynamic environments.

4.2 Future Improvements and Takeaways

The project provided hands-on experience to connect image analysis with object following and control mechanisms and physical hardware operations. The main learning point involved grasping static background model restrictions and how particle filters enable tracking during motion.

Future improvements could include:

- The object models should become more robust through deep learning techniques applied to handle intricate scenes.

- The system needs automatic recalibration features that help retrieve tracking information during occlusion situations.
- The system would benefit from using a depth camera instead of a webcam to enhance tracking precision.
- The PID control system should undergo optimization to create more stable camera movement.

This project served as an effective tool to connect theoretical education with practical development while showing the collaboration between computer vision techniques and control systems.

4.3 Lessons Learned

We learned multiple new technologies together with practical abilities through this project. The project taught us particle filter tracking while demonstrating its usefulness for moving camera situations. We created detection and tracking functions through MATLAB's Image Processing Toolbox alongside PID control implementation, which helped us optimize system responsiveness. Through our practical experiments, we discovered that control feedback loops rely heavily on error signals. Our weak object modeling and unstable motor responses together with improper PID value tuning created problems that helped us develop better debugging abilities while enhancing our iterative design approach.

4.4 Team Dynamics

Omar guided our team by distributing responsibilities effectively while tracking milestones and maintaining effective member communication. The team leader demonstrated his management skills through systematic work distribution and consistent progress monitoring and fostering an open team atmosphere.

To encourage active teamwork we established a Microsoft Teams group which allowed members to communicate instantly and share real-time updates. Our team conducted regular face-to-face meetings in the university lab while we also worked together at Mohammad Fares' residence on several occasions. Each team member participated fully by handling specific parts of the project such as coding and research and hardware integration. Teams

functioned as the primary tool for organizing team discussions and document sharing to maintain clear visibility and easy access to all materials.

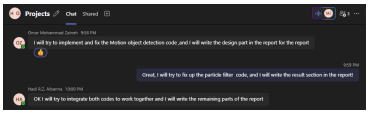


Figure 6: Teams Chat.

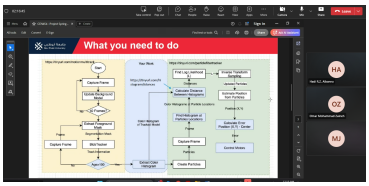


Figure 7: Meeting 1.

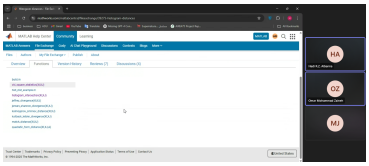


Figure 8: Second Meet.

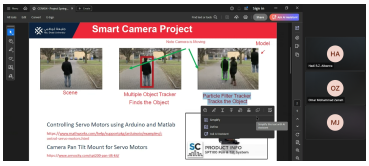


Figure 9: Understanding the required.

The project aimed to create a tracking system which functions in real-time during camera movement while achieving hardware and software integration one week before the final deadline. The planned objectives included achieving MATLAB-based detection module completion during the first two weeks followed by Arduino-controlled servo motor implementation by week four and full system integration completion during the final days. The Gantt chart

displayed in the Figure 10 demonstrates how we allocated various project tasks throughout the entire duration while showing timeframes together with assigned duties and task dependencies.

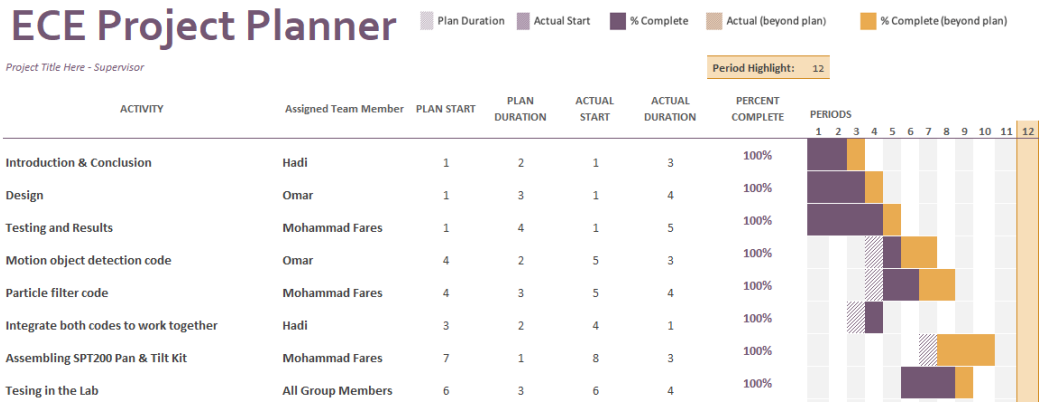


Figure 10: Gantt Chart.

4.5 Impact Statement

Environmental Impact Analysis								
Impact of your project	**Nature**	**Extent**	**Timing**	**Severity**	**Duration**	**Reversibility**	**Uncertainty**	**Significance**
The climate Example: *Does the project affect the emission of greenhouse gases into the atmosphere?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Use of Energy Example: *Does your project affect the energy consumption of the economy? How?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Air quality Example: *Does the project have an effect on emissions of harmful air pollutants that might affect human health, damage crops or buildings or lead to deterioration in the environment (soil or rivers)?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Biodiversity, flora, fauna and landscapes Example: *Does it affect endangered species, their habitats or ecologically-sensitive areas?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							

Water quality and resources Example: *Does the project decrease or increase the quality or quantity of freshwater and groundwater?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Renewable or non-renewable resources Example: *Does the project reduce or increase use of non-renewable resources?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Sustainability Example: Does the option lead to more sustainable production and consumption? How?	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Waste production/generation/recycling Example: *Does the project affect waste production (solid, urban, agricultural, industrial, mining, radioactive or toxic waste) or how waste is treated, disposed of or recycled?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							

Economic Impact Analysis								
Impact of your project	**Nature**	**Extent**	**Timing**	**Severity**	**Duration**	**Reversibility**	**Uncertainty**	**Significance**
Economic Prosperity Example: *Does the project affect the GDP/capita, employment rate, household savings?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Investment Flows Example: *Does your project affect the flow of investment from outside the country? Does it encourage local investment in it?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, because of the nature of this project, many people who are interested can develop it for many different reasons							
Public Budgets or Services Example: *Does the project affect the budgets of hospitals, community services, older people services, transport services, service quality, schools, policing, municipality services..etc?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: people or companies who are intrested might want to buy it to make use of it's functionality							
Market Mechanisms Example: *Does it affect the private sector business opportunities? Help companies reach more costumers? Change how business is done?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							

Innovation, Research and Development Example: *Does the project have commercialization potential, lead to a potential patent? Does it allow others to innovate/research through it?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: yes, many engineers might develop it and make it more efficient,by improving it's speed and the quality of the components.							
Sustainable Consumption and Production Example: Does the project produce a sustainably consumed product or service? Can it be produced sustainably?	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							

Social Impact Analysis								
Impact of your project	**Nature**	**Extent**	**Timing**	**Severity**	**Duration**	**Reversibility**	**Uncertainty**	**Significance**
Health and Longevity Example: *Does the project impact health and longevity? Does it affect physical activity, nutrition, chronic diseases, accidental injuries, independent living, mental wellbeing?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Safety Example: *Does your project affect safety of social environment, protection of older people against abuse, protection against risks, response to emergency cases, feelings of safety, physical safety?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: It might be used for saftey if installed in homes and adding an alarm whenever it detects something wrong							
Productive and Valued Activities Example: *Does the project increase leisure time, reduce stress, lead to positive behavior, increase productivity?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							

Standard of Living Example: *Does it affect the quality of life? Make lives easier? Reduce poverty and deprivation? Increase life choices and opportunities?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Education/Life-long Learning Example: *Does the project affect literacy, use of ICT, chances of higher education, quality of education, life-long learning? Improve attainment of learning outcomes?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Quality of Social Interaction Example: *Does the project affect social connectedness, social participation, volunteering?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							

Privacy and Personal Data Example: *Does the project reveal the user identities? Create potential private data leaks or identity theft?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							
Social Reasonability Example: *Does the project affect access to products and services for people of determination? Does it affect their integration into society? Does it affect their participation in the economy? Does it address their needs?*	Direct Positive	Local	Immediate	High	Temporary	Reversible	Low Likelihood	Unimportant
	Justification/Explanation: no, our project focuses on building a Smart tracking Camera							

Appendix A: PID Control Code

Listing 1: PID Temperature Control Code

```
const int numReadings = 3;

float readings[numReadings];
int readIndex = 0;
float total = 0;
float average = 0;

unsigned long pastmillis = 0;
float timeStamp = 0;
float setpoint = 45;
float error = 0;
float integral = 0;
float kp = 26.775;
float ki = 0.025;
int control = 0;
float kpFan = 40;

void setup() {
  Serial.begin(9600);
  for (int i = 0; i < numReadings; i++) {
    readings[i] = 0;
  }
  Serial.println("CLEARDATA");
  Serial.println("LABEL,Time,TimeStamp,Temperature,Error");
  delay(10000);
}

void loop() {
  float reading = analogRead(2);
  float voltage = reading * (3.3 / 4095);
  float temperatureC = (voltage - 0.5) * 100;

  total = total - readings[readIndex];
  readings[readIndex] = temperatureC;
  total = total + readings[readIndex];
  readIndex = (readIndex + 1) % numReadings;
```

```
  average = total / numReadings;

  unsigned long pasttime = millis() - pastmillis;
  float dt = pasttime / 1000.0;
  error = setpoint - temperatureC;
  integral = error * dt + integral;

  if (error < 0) {
    control = error * kp;
  } else {
    control = error * kp + integral * ki;
  }

  int controlFan = -error * kpFan;

  control = constrain(control, 0, 255);
  controlFan = constrain(controlFan, 0, 255);

  analogWrite(4, control);
  analogWrite(15, controlFan);

  pastmillis = millis();
  timeStamp += dt;

  Serial.print("DATA,TIME,");
  Serial.print(timeStamp);
  Serial.print(",");
  Serial.println(average);
  Serial.print(",");
  Serial.println(error);
  delay(1000);
}
```

Appendix B: Data Collection and Fan Control Code

Listing 2: Data Collection and Temperature-Based Fan Control

```
const int numReadings = 3;

float readings[numReadings];
int readIndex = 0;
float total = 0;
float average = 0;

unsigned long pastmillis = 0;
float timeStamp = 0;

void setup() {
  Serial.begin(9600);
  for (int i = 0; i < numReadings; i++) {
    readings[i] = 0;
  }

  Serial.println("CLEARDATA");
  Serial.println("LABEL,Time,TimeStamp,Temperature");
  delay(10000);
}

void loop() {
  analogWrite(15, 0); // Ensure fan is off initially

  float reading = analogRead(2);
  float voltage = reading * (3.3 / 4095);
  float temperatureC = (voltage - 0.5) * 100;

  total = total - readings[readIndex];
  readings[readIndex] = temperatureC;
  total = total + readings[readIndex];
  readIndex = (readIndex + 1) % numReadings;
  average = total / numReadings;

```

```
  unsigned long pasttime = millis() - pastmillis;
  float dt = pasttime / 1000.0;

  if (average < 60) {
    analogWrite(4, 255);    // Turn heater fully ON
  } else {
    analogWrite(4, 0);      // Turn heater OFF
    analogWrite(15, 255);   // Turn fan ON
  }

  pastmillis = millis();
  timeStamp += dt;

  Serial.print("DATA,TIME,");
  Serial.print(timeStamp);
  Serial.print(",");
  Serial.println(average);

  delay(1000);
}
```

References

[1] A. R. Sadra, "Object tracking camera mount control," https://www.instructables.com/Object-Tracking-Camera-Mount-Control/, n.d., accessed: 2025-06-20.

[2] T. Long. (2024) Review: Ring pan-tilt indoor cam, now you can follow intruders. Accessed: 2025-06-20. [Online]. Available: https://eftm.com/2024/08/review-ring-pan-tilt-indoor-cam-now-you-can-follow-intruders-252365

[3] J. B. Morris and H. G. Liddell, "Object tracking using servo-controlled pan-tilt cameras," https://pdfs.semanticscholar.org/260e/5d61cbff262feeb82923fc7d82403e8fb59a.pdf, n.d., accessed: 2025-06-20.