International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Vision-Based Tracking Turret Using Esp32-Cam and Yolo with a Decoupled Host-Pc Architecture

Dr D Sreekanth[1], Angoth Tharun[2], Rudra Sridatta [3], Beesa Sanjana[4], Manda Pushpachandana[5]
[1]Department of ECE, CMR Technical Campus,India.
Corresponding Author mail id: 247r1a0447@cmrtc.ac.in

Abstract:

Commercial pan-tilt-zoom (PTZ) tracking cameras run detection and servo control on one embedded processor, which delays the response to a moving target and limits them to simple motion-based detection. This paper presents a Vision-Based Tracking Turret that separates the two tasks. An ESP32-CAM streams video over Wi-Fi to a host PC, where a YOLOv5/YOLOv8 model detects the target and computes the angle the turret must move. The angles are sent over a USB serial link to an ESP32 board, which drives two SG90 servos for pan and tilt and switches a laser indicator when the target is locked.

The prototype was tested indoors with a single person as the target. In well-lit conditions at 0.5 to 1.5 m the average detection confidence was 0.92 and the detection rate was 98%. The end-to-end delay from capture to servo command was about 80 to 150 ms. The detector, not the actuator, is the main source of delay, so it can be improved by moving inference to a GPU. The hardware costs about INR 1,940 and the design is easy to extend.

Keywords: Object Detection, YOLO, ESP32-CAM, Pan-Tilt Turret, OpenCV, Servo Control, Embedded Systems, Computer Vision.

I . INTRODUCTION

Automatic tracking of a moving target is needed in surveillance, human-robot interaction, industrial inspection and wildlife monitoring. Earlier tracking systems needed costly processors or FPGAs. Pre-trained deep-learning detectors and low-cost microcontrollers now make it possible for a student team to build a working tracker with commodity parts.

Commercial PTZ cameras usually decode video, run the tracking algorithm and issue motor commands on a single processor. Their control delay is typically 150 to 500 ms, and they depend on background subtraction, which fails when the background or lighting changes and cannot tell a person from any other moving object. Their firmware is also closed, so the detector cannot be replaced.

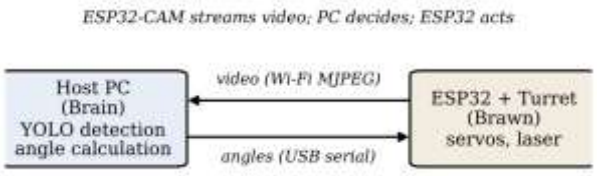


Figure 1: Concept of the decoupled tracking turret

The objective of this work is to build a low-cost two-axis turret that detects a target with YOLO, follows it with pan and tilt servos and lights a laser when the target is centred. The ESP32-CAM is used only as a wireless camera, the PC does all the detection, and a separate ESP32 board does only servo and laser control. The rest of the paper is organised as follows. Section II reviews related work, Section III describes the proposed method, Section IV presents the results and Section V concludes the paper.

II . LITERATURE REVIEW

Work on object detection and PTZ tracking that was studied for this project is summarised in Table I and discussed below.

Table I: Summary of related work

S.No	Authors	Method	Ref., Year
1	Viola and Jones	Haar features with a boosted cascade for face detection	[1], 2004
2	Dalal and Triggs	HOG features with an SVM for human detection	[2], 2005
3	Liu et al.	SSD single-shot multi-box detector	[3], 2016
4	Redmon et al.	YOLO single-pass real-time detector	[4], 2016
5	Raza et al.	Raspberry Pi 4 turret with Haar cascade, about 180 ms delay	[5], 2019
6	Jain and Kumar	Raspberry Pi 4 turret with MobileNet-SSD, about 220 ms delay	[6], 2021
7	This work	ESP32-CAM and ESP32 turret with YOLO on a host PC	-

Early detectors used hand-made features. Viola and Jones used Haar features with a boosted cascade, and Dalal and Triggs used histograms of oriented gradients with an SVM. Both scan the image with a sliding window, which is slow and breaks down when the scene changes [1], [2]. Region-based CNN detectors improved accuracy, but they need a separate region proposal stage and are too slow for a control loop. Single-stage detectors remove that stage. SSD predicts boxes at several scales in one pass [3], and YOLO divides the image into a grid and predicts boxes and classes in a single network pass [4]. The Ultralytics releases YOLOv5 and YOLOv8 add easy training and export, and their small variants run in real time on a CPU [7].

Embedded tracking turrets have been built on single-board computers. Raza et al. used a Raspberry Pi 4 with a Haar cascade and reported a servo delay of about 180 ms [5]. Jain and Kumar replaced it with MobileNet-SSD, which lowered missed detections but raised the delay to about 220 ms because the same processor also had to run the heavier model [6]. High-end commercial PTZ cameras also respond in about 200 ms.

It can be seen that existing low-cost systems either use a weak detector or put detection and actuation on one processor, which makes the delay long. This motivates a design in which a PC

runs the detector and a small microcontroller only drives the servos.

III . PROPOSED METHOD

System overview: The system has three parts: an ESP32-CAM that streams video, a host PC that detects the target, and an ESP32 board that drives the servos and the laser. The block diagram is shown in Fig. 2. The PC receives the video over Wi-Fi, finds the target, calculates the pan and tilt angles and sends them to the ESP32 board over a USB-to-serial link.

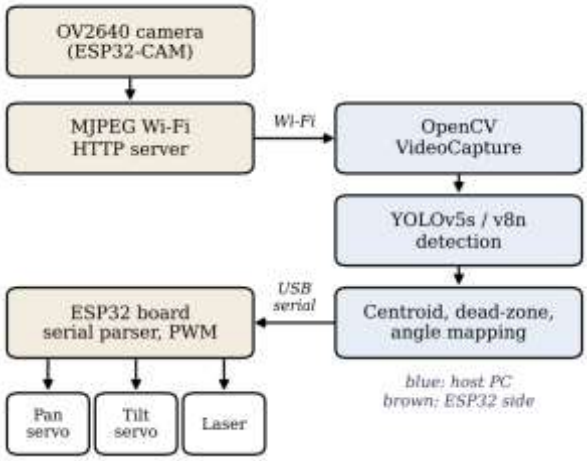


Figure 2: Block diagram of the Vision-Based Tracking Turret

Video acquisition: The ESP32-CAM carries an OV2640 sensor and is set to QVGA (320x240) to keep the frame rate high and the network load low. It joins the local Wi-Fi network and serves the frames as an MJPEG stream over HTTP. The PC opens the stream with the OpenCV VideoCapture class and treats it as a virtual camera. Each frame is compressed on its own, so it can be decoded without delay.

Object detection: A pre-trained YOLOv5s or YOLOv8n model from the Ultralytics package runs on the PC. The model is set to detect the person class, and only detections with a confidence of at least 0.5 are accepted. If several persons are found, the one with the highest confidence is tracked [7].

Centroid to angle mapping: The centre of the bounding box is calculated as cx = (x1 + x2)/2 and cy = (y1 + y2)/2. The error from the frame centre is dx = cx - W/2 and dy = cy - H/2, where W and H are the frame width and height. Each error is multiplied by a sensitivity factor in degrees per pixel (about 0.28 for a 320-pixel frame with a 90 degree field of view) and added to the current servo angle. The result is limited to 0 to 180 degrees. A dead-zone of 20 pixels around the centre stops the turret from jittering when the target is already centred.

Serial communication: The PC sends one ASCII command per frame in the form P<pan>T<tilt>L<laser>, for example P95T85L1, ending with a newline, over a 115200 baud 8N1 serial link using PySerial [8]. The ESP32 reads the bytes until the newline, extracts the angles and the laser state, limits the angles to 0 to 180 degrees and updates the outputs. This takes under 5 ms.

Actuation: Two SG90 micro servos form the pan and tilt axes. The ESP32 generates 50 Hz PWM with pulse widths of about 500 to 2400 microseconds through the ESP32Servo library [9]. A 5 mW, 650 nm laser module is switched by a GPIO pin. The PC sets the laser flag when the centroid stays inside the dead-zone for several consecutive frames, and clears it when no target is found. The wiring is shown in Fig. 3 and the pin connections are given in Table II.

Table II: ESP32 pin connections

ESP32 pin	Connected to	Signal
GPIO 13	Pan servo	PWM, 50 Hz
GPIO 12	Tilt servo	PWM, 50 Hz
GPIO 2	Laser module	Digital output
RX0 (GPIO 3)	USB-serial TX	UART, 115200 baud
VIN / GND	5 V buck output / common ground	Power

Power supply: A mini-360 or XL4016 buck converter gives a regulated 5 V from a 7.4 V Li-Po pack. The servos and both ESP32 boards are fed from this rail, and 100 uF and 10 uF capacitors are placed near the boards to absorb servo current spikes and prevent brown-out resets. The ground is common to all parts. The ESP32-CAM can draw about 500 mA during Wi-Fi transmission, and a servo can draw 500 mA to 1 A when stalled, so a stable supply is required.

Hardware and software: The code is written in Arduino C++ in the Arduino IDE for the two ESP32 boards and in Python 3.9 or later on the PC, with OpenCV, PySerial and the Ultralytics package. The ESP32-CAM firmware is based on the Espressif CameraWebServer example. The processing loop on the PC reads a frame, runs detection, selects the target, applies the dead-zone, sends the command and draws the box and centroid on the display.

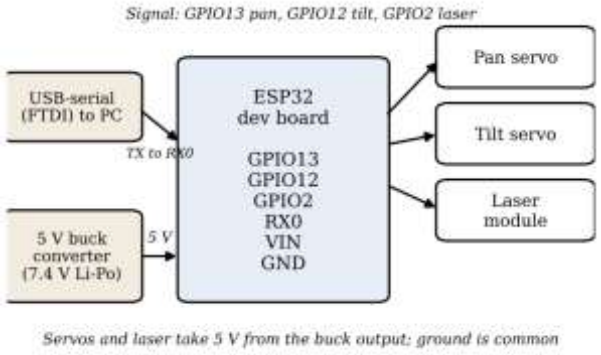


Figure 3: Wiring of the ESP32 board with the servos, laser, serial link and power supply

Comparison with existing methods: Table III compares the proposed turret with the existing methods described in Section II.

Table III: Existing methods and the proposed method

Parameter	Existing methods	Proposed method
Processing	Detection and control on one processor	Detection on PC, control on ESP32
Detector	Background subtraction, Haar cascade or MobileNet-SSD	YOLOv5s / YOLOv8n
Delay	About 150 to 500 ms	About 80 to 150 ms
Class aware	Often not	Yes, 80 COCO classes
Extensible	Closed firmware or partly open	Fully open
Cost	High for commercial units	About INR 1,940

[Insert Fig. 4: Photograph of the assembled turret prototype]

Figure 4: Hardware prototype of the tracking turret

IV . RESULTS AND DISCUSSION

Hardware: The prototype was assembled on a plastic pan-tilt bracket with the ESP32-CAM and laser mounted on the moving platform, as shown in Fig. 4. The buck converter held 5.0 V within 0.1 V under load, no brown-out resets occurred, and the ESP32-CAM connected to Wi-Fi within 3 to 5 seconds of power-up. Sweep tests over the full range of both servos showed smooth motion without binding. The laser dot lined up with the camera axis at the 90/90 degree position.

Detection performance: A single person was the target, at distances of 0.5 to 3 m and under three lighting conditions. The results are given in Table IV. In good light up to 1.5 m the confidence was 0.92 on average and the person was found in 98% of frames. Beyond 2.5 m the person occupies fewer pixels in the QVGA frame, the confidence drops to about 0.61 and some frames are missed, so the servos hold their last position until the next detection. In dim light the confidence falls below the 0.5 threshold and tracking stops. Raising the sensor gain helped only partly because it adds noise.

Table IV: Detection performance

Test condition	Avg. confidence	Detection rate	Remark
Well-lit, 0.5-1.5 m	0.92	98%	Stable
Well-lit, 1.5-2.5 m	0.79	93%	Minor jitter
Well-lit, 2.5-3.0 m	0.61	78%	Some misses
Low light, 0.5-1.5 m	0.48	55%	Frequent dropout
Partial occlusion	0.67	72%	Smaller box

Latency: Latency was logged in the Python program at frame capture, end of inference and command transmission, and the servo travel time was estimated from the servo speed. The averages over 50 detection events are given in Table V. The total from capture to servo command was about 80 to 150 ms in normal operation. Video transfer and YOLO inference together account for most of this time, while the centroid calculation, serial link and ESP32 firmware add less than 10 ms. For a small correction of under 20 degrees the servo travel takes less than 33 ms.

Table V: Latency of each stage

Stage	Time (ms)	Share
Wi-Fi MJPEG transfer	30 - 60	~35%
YOLO inference (CPU)	50 - 100	~45%
Centroid and angle mapping	< 2	< 2%
Serial transmission	2 - 5	~3%
ESP32 parse and PWM update	< 5	~3%
Servo travel (90 deg worst case)	~150	not in total

Discussion: Detection was reliable in good light at close and medium range, which suits indoor monitoring. Because inference is the largest delay, a GPU on the host PC is expected to reduce it to under 10 ms and the total delay to under 50 ms, but this was not measured. The turret needs a PC and a USB cable, so it is not a stand-alone unit.

Limitations: The tests were indoors with one target and a single camera. Servo travel time was estimated and not measured. Performance falls at long range and in low light because of the OV2640 sensor and the pre-trained model, and the system does not predict motion through missed detections. The laser is a Class II indicator only. Outdoor use and range beyond 3 m were not evaluated.

V . CONCLUSION

This paper presented a Vision-Based Tracking Turret in which a host PC runs YOLO detection and an ESP32 board drives the pan-tilt servos and a laser indicator, with an ESP32-CAM as the wireless camera. Splitting detection from actuation avoids the processor contention found in single-processor trackers. The prototype tracked a person indoors with an average confidence of 0.92 at close range and a delay of about 80 to 150 ms, at a hardware cost of about INR 1,940. Future work will add a thermal sensor for dark scenes, Kalman filter prediction to bridge missed detections, GPU or edge-AI inference to remove the PC, multi-target tracking and a web dashboard for remote monitoring.

VI . REFERENCES

1) P. Viola and M. J. Jones, "Robust Real-Time Face Detection," Int. J. Computer Vision, vol. 57, no. 2, pp. 137-154, May 2004.
2) N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in Proc. IEEE CVPR, 2005, vol. 1, pp. 886-893.
3) W. Liu et al., "SSD: Single Shot MultiBox Detector," in Eur. Conf. Computer Vision (ECCV), Springer, 2016, pp. 21-37.
4) J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE CVPR, 2016, pp. 779-788.
5) Raza et al., Raspberry Pi 4 face-tracking turret using an OpenCV Haar cascade, 2019. [Full citation to be added.]
6) Jain and Kumar, Raspberry Pi 4 tracking turret using MobileNet-SSD, 2021. [Full citation to be added.]
7) Ultralytics, "YOLOv8 Documentation." [Online]. Available: https://docs.ultralytics.com
8) PySerial Development Team, "PySerial Documentation." [Online]. Available: https://pyserial.readthedocs.io
9) Arduino, "ESP32Servo Library Documentation." [Online]. Available: https://madhephaestus.github.io/ESP32Servo/annotated.html