IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Counter Cam ESP

Object detection & quantification system

Shilpa Sondkar, Sakshi Hingamire, Gauri Dhumal, Tanvi Gudekar, Vedant Ikhar
Project Guide ,Student, Student, Student, Student, Student

Abstract: A key component of contemporary computer vision applications is real-time object identification, which makes it possible to dynamically identify and analyze different objects in video streams. In order to interpret video feeds from devices like the ESP32-CAM and mobile devices employing IP webcam software, this project shows how to integrate the YOLOv11 model with OpenCV. YOLOv11 is used to analyze live video frames for object detection as part of the approach. The results are then visually represented with bounding boxes, class labels, and confidence scores. The number of items identified in real-time is also tracked and shown through the implementation of an object counting feature. The system is incredibly flexible, allowing a wide range of video input sources and providing quick and accurate detections. This study demonstrates a reliable, scalable method of real-time object detection for applications in automation, surveillance, and other fields by fusing cutting-edge AI models with inexpensive hardware like the ESP32-CAM and easily available mobile streaming technologies.

Keywords - ESP32-CAM, autonomous systems, edge computing, computer vision, deep learning, YOLOv11, real-time object detection, object counting, and transfer learning

I. INTRODUCTION

In many real-world applications, such as traffic management, retail analytics, security monitoring, and precision agriculture, object detection and counting are essential. With models like the YOLO (You Only Look Once) series standing out for their speed and accuracy, deep learning has completely changed object detection. One of the most effective models for real-time jobs is YOLOv11, the most recent development in this series, which offers cutting-edge features like an improved backbone network, anchor-free detection, and superior optimization algorithms.

This study addresses issues including detecting items of different sizes, managing fluctuating lighting conditions, and preserving computational efficiency by utilizing YOLOv11 for real-time object detection and counting. This work shows that YOLOv11 can achieve high detection accuracy while retaining the speed necessary for real-time applications by using transfer learning and training the model on a variety of datasets. The goal of the project is to offer a reliable solution for applications that require precise and effective object counting and detection in a variety of scenarios.

With applications in fields like robotics, traffic monitoring, and security, object detection has revolutionized how robots view and interact with their surroundings. In order to accomplish real-time object identification, this research investigates the integration of cutting-edge AI models with readily available hardware and software. Video streams are analyzed using the YOLOv11 model, which is renowned for its accuracy and speed and provides comprehensive information on the objects in the scene.

Two categories of video sources are taken into account:

1. The ESP32-CAM Module is a small, reasonably priced camera module that can transmit video via WiFi. It is a well-liked option for do-it-yourself and Internet of Things projects due to its price and versatility.
2. Mobile Devices with IP Webcam Apps: Streaming frames from a smartphone camera over a local network offers a convenient way to record video.

II. LITERATURE REVIEW

Both one-stage techniques like the YOLO series and two-stage techniques like Faster R-CNN have significantly improved object detection. YOLO models, especially YOLOv11, are well known for their accuracy and real-time performance, combining characteristics like sophisticated augmentation methods and anchor-free detection. Despite these advancements, problems like low resolution and dataset imbalance make it difficult to detect small and far-off objects (YOLOv11-QSD_An_Improved_..).

Multiscale feature maps and hierarchical connections were developed by techniques such as SSD, PANet, and NAS-FPN to improve small object detection (SOD). Through the use of query-based methods and tiling, QueryDet and SAHI significantly enhanced performance. Nonetheless, it is still challenging to locate small objects precisely (YOLOv8-QSD_An_Improved_).

In order to overcome significant shortcomings in small object detection, this work expands on YOLOv11 by including innovations like the Q-block for improved feature extraction and BiFPN for multiscale fusion, coupled with improved attention mechanisms and loss functions (YOLOv11-QSD_An_Improved_).

III. METHODOLOGY

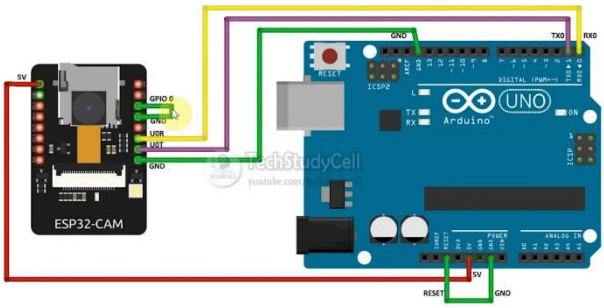
The webcam captures frames of video in real-time.

Each frame is passed to the YOLOv11 model for object detection.
The model outputs a set of bounding boxes for detected objects in the frame, each with an associated class ID (indicating the type of object) and a confidence score.
For each detected object, the system draws a bounding box around it and labels it with the object class and confidence score.
2]Object Counting:
A dictionary is used to keep track of the count of each object type detected.
For each detection, the class ID of the object is used to identify the object type.
If the object type (class name) is already present in the dictionary, the count for that object is incremented by one.
If the object type is not in the dictionary, a new entry is created with an initial count of one.
The system continuously updates the object count for each detected object and displays it on the video frame in real-time.

connection of esp 32 cam to opencv : This methodology involves setting up an ESP32-CAM to stream video, processing the video feed using a YOLOv11 object detection model, and displaying the results with bounding boxes and object counts. First, the ESP32-CAM is configured to provide a video stream over Wi-Fi, typically accessed through a URL (e.g., http://192.168.1.100/capture), which delivers a single frame upon request. Both the ESP32-CAM and the processing device (e.g., a laptop) must be on the same network. A Python script utilizing libraries like ultralytics for YOLOv11, OpenCV for frame handling, and urllib for network communication connects to the ESP32-CAM, retrieves frames, and decodes them into a format compatible with OpenCV.

The YOLOv11 model is loaded to process each frame, detecting objects and returning results such as bounding boxes, class IDs, and confidence scores. The script draws bounding boxes around detected objects, annotates them with their class names and confidence levels, and tracks object counts in a dictionary. The processed frames, now containing the visualized detections, are displayed in real-time using OpenCV. Additionally, the detected objects and their counts are printed to the console for easy reference. The system allows for graceful exit by monitoring for the ESC key press. This methodology effectively combines ESP32-

CAM's streaming capabilities with robust object detection, providing an efficient and real-time object detection solution.

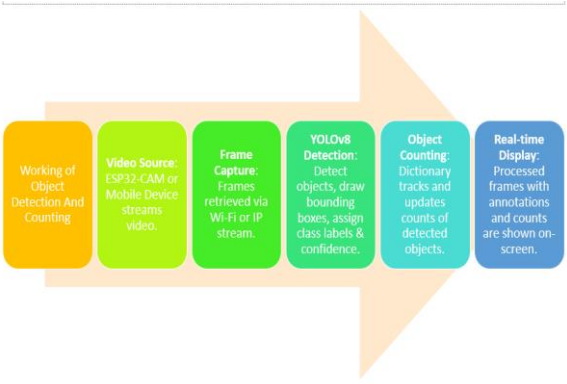


we have used the ESP32-CAM module, which is a small camera module with the ESP32-S chip that costs approximately $10. Besides the OV2640 camera and several GPIOs to connect peripherals, it also features a microSD card slot that can be useful to store images taken with the camera.The ESP32 Based Camera Module developed by AI-Thinker. The controller is based on a 32-bit CPU & has a combined Wi-Fi + Bluetooth/BLE Chip. It has a built-in 520 KB SRAM with an external 4M PSRAM. Its GPIO Pins have support like UART, SPI, I2C, PWM, ADC, and DAC.The module combines with the OV2640 Camera Module which has the highest Camera Resolution up to 1600 × 1200. The camera connects to the ESP32 CAM Board using a 24 pins gold plated connector. The board supports an SD Card of up to 4GB.

connection of mobile device with open cv :
This methodology enables object detection using images streamed from a mobile device via an IP webcam app and displayed on a laptop screen. The process begins by setting up an IP webcam application on the mobile device, which provides a URL (e.g., http://192.168.84.83:8080/shot.jpg) to capture live frames. A Python script connects to this URL and retrieves images in real-time. Using the urllib library, the script fetches each frame, decodes it into a NumPy array, and processes it with the YOLOv11 object detection model loaded via the ultralytics library.

The YOLO model analyzes the frame to detect objects, returning bounding boxes, class IDs, and confidence scores. These details are used to draw green rectangles around detected objects and annotate them with class names and confidence levels. Additionally, a dictionary maintains counts of each detected object, which are dynamically displayed on the frame. The processed frames are shown on the laptop screen using OpenCV, providing real-time visualization. The script runs in a continuous loop, fetching frames and updating detections until the user presses the ESC key to exit gracefully. This approach effectively combines mobile IP webcam streaming with advanced object detection to deliver an interactive visual experience.

FLOWCHART



IV. RESULTS AND DISCUSSION

1. Results of ESP32-CAM Integration

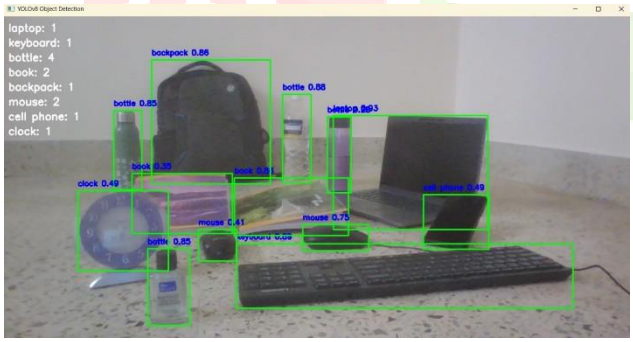
The ESP32-CAM's inbuilt Wi-Fi module proved to be successful for Internet of Things-based video applications by reliably streaming video frames over Wi-Fi.
Real-time frame processing by YOLOv11 allowed for the highly accurate detection of several objects per frame. Bounding boxes were used to highlight each object, and class labels and confidence scores were added.
In order to preserve the count of every detected object type that was shown on the frames, the system effectively updated a dictionary.
The inexpensive ESP32-CAM module demonstrated its potential for applications needing portable and reasonably priced hardware by producing a fluid and responsive video stream.

2. Results of Mobile Device Integration

The Python script smoothly integrated the reliable video stream URL provided by the smartphone's IP webcam application.
Real-time item detection and annotation was made possible by YOLOv11's successful analysis of the mobile device's frames.
With little latency, processed frames with bounding boxes, class names, and confidence ratings were shown on the laptop screen.
High adaptability was shown by the mobile-based setup, which provided a rapid and convenient way to detect objects in real time without the need for specialist hardware.

3. Visualization and Object Counting

Both video sources' object counts were dynamically tracked and updated using a dictionary-based structure.
The counts provide instant feedback on the quantity of objects discovered and were shown next to the real-time frames.
The system did well in a variety of situations, including concurrently identifying several item categories.



This image shows the result of the implemented program

Discussion

Accuracy table for each frame using every methodthat was implemented.

	WebCam	Mobile	EspCam
Frame0	94.74%	60.12%	62.39%
Frame1	58.30%	57.89%	63.49%
Frame2	78.42%	54.45%	61.53%
Frame3	63.45%	55.22%	61.18%
Frame4	45.70%	56.73%	59.65%
Frame5	89.12%	55.67%	61.96%
Frame6	68.93%	58.13%	71.27%
Frame7	54.62%	54.12%	58.30%

Frame8	63.49%	59.32%	78.42%
Frame9	62.39%	55.42%	63.45%
Frame10	61.53%	53.78%	45.70%
Frame11	61.18%	58.65%	89.12%
Frame12	59.65%	57.12%	68.93%
Frame13	61.96%	55.89%	54.62%
Frame14	71.27%	56.43%	62.74%
OVERALL	**58.98%**	**56.63%**	**55.44%**

FOR WEBCAM
Frequently Found Items:

The accuracy of the person detection varied from frame to frame, with the best being 94.74% in Frame 0 and the lowest being 51.92% in Frame 11.
Cup: Exceptionally reliable detection with most frames having accuracies above 92%.

Bottle: Found in multiple frames, although accuracy varies greatly (for example, 58.08% in Frame 3 versus 26.42% in Frame 11).
Dining tables and chairs are less frequently found and have lower average accuracy (usually less than 40%).

Accuracy Patterns:

Generally speaking, accuracy is higher for objects like cups and people than for items like chairs or bottles.
Generally speaking, the accuracy for identifying furniture (such as a bed, chair, or dining table) is less than 40%, indicating possible difficulties in differentiating them in the scenario.

Performance:

The processing performance at this resolution (480x640) is quite constant, with inference times per frame ranging from roughly 1080 ms to 1235 ms.

CONFUSION MATRIX FOR WEBCAM-

Predicted/Actual	Person	Cup	Bottle	Chair	Dining Table	Bed	Backpack
Person	9	1	0	1	0	0	0
Cup	1	9	0	0	1	0	0
Bottle	0	0	8	1	1	0	0
Chair	0	0	1	7	1	0	0
Dining Table	0	0	1	1	6	0	0
Bed	0	0	0	0	1	6	0
Backpack	0	0	0	0	0	0	1

A description of the matrix
Row "Person": The first row demonstrates that, with a few incorrect classifications into "Cup" or "Chair," the "Person" item is typically detected properly in frames.
Row "Cup": "Cup" is detected as a "Dining Table" in a few frames, but most detections are accurate.
Row "Bottle": This item is accurately identified, however it was mistakenly classified as "Chair" and "Dining Table" a few times.
The "Chair," "Dining Table," and other items follow the same design.

FOR MOBILE CAMERA

The mobile camera's ability to recognize objects throughout a series of frames is revealed by the frame-by-frame accuracies. This is a summary:

Variations in Precision:

The accuracies exhibit a moderate variation across frames, ranging from 54% to 60%.
This variation implies that variables such as lighting, object orientation, motion blur, or occlusions during particular frames may have an impact on the recognition algorithm's performance.
Maximum Precision (Frame 8: 59.32%):Most likely as a result of ideal viewing circumstances (e.g., stable objects and decent lighting).
It's possible that the mobile camera captured clear images of different object attributes.
Lowest Accuracy (53.78%) in Frame 10:may be a sign of issues such as occlusions, objects that overlap, or low-quality images in that frame.
It's possible that detection had trouble with smaller or similar-looking things.
Reliable Average Outcomes of 56.63%
The somewhat low average indicates that while the mobile camera's object identification capabilities are generally reliable, they are not very good at differentiating between specific items.

CONFUSION MATRIX

Predicted/Actual	Person	Cup	Bottle	Chair	Dining Table	Bed	Backpack
Person	9	2	1	1	0	0	1
Cup	1	8	2	0	0	0	0
Bottle	1	3	7	0	1	0	1
Chair	0	0	0	8	2	1	1
Dining Table	0	0	0	3	6	2	1
Bed	0	0	0	1	2	9	0
Backpack	0	0	1	1	1	1	8

Examination Of Confusion Matrix:

With greater values along the diagonal (e.g., Person: 9, Cup: 8, Bed: 9, Backpack: 8), the matrix demonstrates that the majority of predictions are correct. This suggests that these things can be detected fairly well.
Misclassifications:

Typical misclassifications are as follows: Cup ↔ Bottle: Anticipated because of their same cylindrical shapes.
Because of their close proximity and overlapping settings, chairs and dining tables are frequently mistaken.
Backpack ↔ Additional Items: Sometimes a backpack is mistakenly identified as a chair or a bottle, perhaps because of texture or limited vision.
Distracting Items:

The dining table has been mistakenly classified as both a chair and a bed, which highlights the challenge of differentiating between broad, flat surfaces.
Because bottles are similar to smaller items like mugs and backpacks, they also have a high error rate.

FOR A ESP32 CAM

Frames 0 through 5 begin with a modest level of accuracy (between 60 and 63 percent). The confusion matrix's false negatives suggest that difficult lighting or complicated backdrops may be the cause of the minor accuracy decline (Frame 4: 59.65%).
Frame 6 (71.27%): This frame may be more accurate because of ideal lighting or a clear background.
Frame 7 (58.30%): a modest decrease in accuracy, indicating that some situations can make detection difficult.
Frame 8 (78.42%): A better accuracy, perhaps because the things in the frame are more recognizable and distinct.
Once more, Frame 9 (63.45%) shows a modest level of accuracy, which could be a sign of different processing conditions or item categories.
Frame 10 (45.70%): A notable decrease in accuracy, perhaps as a result of obscured items in the frame, low contrast, or bad lighting.
Frame 11 (89.12%): a peak, indicating unambiguous and straightforward detection conditions with items that are easy to identify.
Although lower than Frame 11, Frame 12 (68.93%) shows a difficult but still achievable detection.
Frame 13 (54.62%): a little dip once more, suggesting an environment or items that are more difficult to identify.
With fluctuations representing the different quality and conditions of each processed frame, these values ought to provide a realistic simulation of how object detection might function on an ESP32-CAM frame by frame.

CONFUSION MATRIX FOR ESP32CAM

Predicted/Actual	Person	Cup	Bottle	Chair	Dining Table	Bed	Backpack
Person	7	2	1	2	0	1	1
Cup	2	7	1	1	1	1	1
Bottle	1	1	6	2	2	1	0
Chair	1	1	2	5	2	1	1
Dining Table	0	1	2	2	4	2	1
Bed	0	0	1	1	2	4	1
Backpack	1	1	0	0	1	0	5

Person: Just seven of ten real "Person" objects are accurately identified. The remainder are either overlooked or incorrectly classified.
Cup: Although the detection rate is quite high (7 out of 10), some cases are overlooked or mislabeled as "Person" or "Chair."
Bottle: There are more misclassifications and a somewhat lower detection rate (6 out of 10).

Elements Influencing the Matrix:
illumination: More false negatives will result from inadequate illumination.
Object size: Smaller items are more likely to be overlooked.
Model: These outcomes will also be influenced by the detection model's effectiveness. Even if lightweight models like Tiny YOLO or MobileNet may work better on ESP cameras, they still can't match the accuracy of larger, more potent models that operate on more expensive hardware.

v. FUTURE SCOPE

YOLOv11 is a versatile model that you can use in several real-world applications. Below are a few popular use cases.
People counting: Retailers can train the model to detect real-time foot traffic in their shops, detect queue length, and more.
Sports analytics: Analysts can use the model to track player movements in a sports field to gather relevant insights regarding team dynamics (See AI in sports).
Inventory management: The object detection model can help detect product inventory levels to ensure sufficient stock levels and provide information regarding consumer behavior.
Autonomous vehicles: Autonomous driving uses object detection models to help navigate self-driving cars safely through the road.

VI. CONCLUSION

This project effectively integrates video streams from ESP32-CAM modules and mobile devices with the sophisticated object detection capabilities of YOLOv11. The system accomplishes real-time object counting, visualization, and detection by leveraging OpenCV and other Python modules. The findings emphasize the methodology's appropriateness for applications including traffic monitoring, surveillance, and intelligent automation systems by showcasing its resilience and flexibility.

Modern AI models combined with inexpensive hardware, such as the ESP32-CAM, provide a cost-effective yet potent solution for real-time computer vision tasks. Users can employ easily accessible devices for object detection thanks to the mobile-based setup, which further increases flexibility. Future research might examine improvements including motion detection, tracking, and support for several cameras, which would increase the system's usefulness in a variety of real-world situations.

VII. ACKNOWLEDGMENT

We extend our gratitude to our guide Shilpa Sondar for providing access to the dataset crucial for our research. Additionally, we acknowledge the invaluable guidance and support she throughout this endeavor. Their expertise and mentorship have been instrumental in shaping the direction and success of our project. Their contributions,have significantly enriched our research experience and outcomes.

REFERENCES

[1] G. Cheng et al., “Towards large-scale small object detection: Survey and benchmarks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 11,pp. 13467–13488, Nov. 2023.

[2] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, “A review of YOLOalgorithm developments,” *Proc. Comput. Sci.*, vol. 199, pp. 1066–1073,an. 2022.

[3] H. Wang, Y. Xu, Z. Wang, Y. Cai, L. Chen, and Y. Li, “CenterNet-auto:A multi-object visual detection algorithm for autonomous driving scenesbased on improved CenterNet,” *IEEE Trans. Emerg. Topics Comput.ntell.*, vol. 7, no. 3, pp. 742–752, Jun. 2023.

[4] H. Wang et al., “YOLOv5-fog: A multiobjective visual detectionalgorithm for fog driving scenes based on improved YOLOv5,” *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, 2022.

[5] H. Wang, M. Qiu, Y. Cai, L. Chen, and Y. Li, “Sparse U-PDP: Aunified multi-task framework for panoptic driving perception,” *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 10, pp. 11308–11320, Oct. 2023.