



Computer Vision-Based Traffic and Pedestrian Analysis Using Video Feeds

Assistant Professor Smita Mangesh Junnarkar¹, Sunil Manoj²

Department of DataScience, SIES COLLEGE OF ARTS, SCIENCE AND COMMERCE (EMPOWERED AUTONOMOUS), Sion(W), Mumbai.

¹Smita Mangesh Junnarkar , smitaj@sies.edu.in

Department of DataScience, SIES COLLEGE OF ARTS, SCIENCE AND COMMERCE (EMPOWERED AUTONOMOUS), Sion(W), Mumbai.

²Sunil Manoj , sunilr70214@gmail.com

ABSTRACT :

Urban roads are shared by vehicles and pedestrians, and planners need dependable measurements of how both groups use that space. Manual counting is slow and expensive, and many sensor-based methods cover only one kind of road user. This paper presents a computer vision pipeline that analyzes ordinary video feeds to detect, track, and count vehicles and pedestrians in a single pass. The system combines a pretrained YOLO object detector, a multi-object tracker that assigns persistent identities, and a lightweight analytics layer that performs virtual line counting, zone occupancy estimation, and proximity-based interaction indicators. The pipeline is implemented in Python with OpenCV and is exposed through a Streamlit dashboard that runs on a standard Windows laptop without specialized hardware. The paper describes the design rationale, the processing stages, an evaluation protocol based on detection, tracking, and counting metrics, and the practical limits that arise from occlusion, camera geometry, illumination, and the use of a model that has not been fine-tuned to local traffic. Privacy and ethical safeguards are also discussed, including the decision to analyze only aggregate movement and to avoid face or license plate recognition. The work shows that a modular, open-source pipeline can serve as an accessible starting point for traffic and pedestrian analytics in academic and small-scale civic settings.

Keywords—computer vision, traffic analysis, pedestrian detection, YOLO, object tracking, OpenCV, video analytics, intelligent transportation systems

1. INTRODUCTION

Roads in most cities serve several kinds of users at the same time. Cars, buses, trucks, motorcycles, and bicycles share the carriageway, while pedestrians cross it, wait beside it, and walk along it. Decisions about signal timing, crossing placement, lane allocation, and road safety depend on knowing how many of these users are present, where they move, and how their paths interact. Obtaining that knowledge has traditionally required field staff who count users by hand, or physical sensors such as inductive loops, pneumatic tubes, and radar units. Manual counting is accurate for short periods but cannot be sustained, and most physical sensors are designed for vehicles and say little about people on foot.

Video cameras are already widespread along urban roads, at campus gates, and around public buildings. Each camera records a continuous visual description of the scene, but a human observer cannot watch more than a small fraction of that footage. Computer vision offers a way to convert the footage into structured numbers: how many objects of each class crossed a given line, how many people stood inside a given area, and how those quantities changed over time. The recent growth of deep learning object detectors has made this conversion practical even for modest hardware, because modern detectors process each frame in a single forward pass and are distributed with weights trained on large public datasets.

This paper presents the design of a computer vision system for traffic and pedestrian analysis from video feeds. The system accepts a recorded video file, a webcam stream, or a network camera stream. It detects road users in each frame with a pretrained YOLO model, links detections across frames using a tracker so that each object keeps a stable identity, and then computes simple but useful statistics from the resulting trajectories. The statistics include directional counts across a virtual line, occupancy of a user-defined pedestrian zone, and an exploratory indicator of how close pedestrians and vehicles come to one another. The results are written to an annotated video and a tabular log, and they are presented through an interactive dashboard.

The goals of the work are modest and practical. The first goal is to build a pipeline that a student or a small municipal team can run on an ordinary laptop. The second goal is to keep every stage transparent, so that each design choice can be explained, tested, and replaced. The third goal is to describe honestly what such a system can and cannot claim. A pretrained detector applied to unseen footage does not guarantee a particular level of accuracy, and any number reported for a deployment must come from measurements on that deployment. For this reason the paper devotes considerable space to the evaluation protocol and to the limitations of the approach, rather than to headline figures.

The main contributions of this paper are as follows. (i) It describes a modular detect-track-analyze architecture for joint vehicle and pedestrian monitoring. (ii) It formalizes the geometric rules used for line crossing, zone occupancy, and proximity indicators, together with the safeguards that prevent double counting. (iii) It proposes an evaluation protocol that combines detection, tracking, and counting metrics with a manual ground-truth procedure. (iv) It discusses privacy, ethical, and practical constraints that should accompany any deployment of video analytics in public spaces.

The remainder of the paper is organized as follows. Section 2 reviews related work on classical and deep learning approaches to detection and tracking. Section 3 states the problem and the design requirements. Section 4 explains the proposed methodology in detail. Section 5 describes the implementation

and the dashboard. Section 6 presents the evaluation protocol. Section 7 discusses challenges, applications, and ethical considerations, Section 8 outlines future work, and Section 9 concludes.

2. RELATED WORK

2.1 Classical Approaches to Traffic Video Analysis

Early video-based traffic systems relied on background modeling. A static camera observes the scene for some time, a statistical model of the background is built for each pixel, and pixels that deviate from the model are labeled as foreground. Stauffer and Grimson [14] modeled every pixel with a mixture of Gaussian distributions that adapts to gradual illumination change, and Zivkovic [15] later improved the efficiency and adaptivity of that idea. Connected foreground regions were then treated as vehicles or people, and their sizes and motion were used to separate classes. The approach is fast and needs no training, which explains its long use in traffic monitoring. Its weaknesses are equally well known: cast shadows are labeled as foreground, stopped vehicles slowly fade into the background, adjacent objects merge into a single blob, and camera motion corrupts the model. Buch, Velastin, and Orwell [16] surveyed these techniques for urban traffic and noted how strongly they depend on controlled viewing conditions.

For pedestrians, researchers developed dedicated appearance-based detectors. Dalal and Triggs [17] introduced histograms of oriented gradients, which describe local edge structure and, when combined with a linear classifier, gave a strong detector for upright people. The Caltech pedestrian benchmark and the evaluation by Dollár et al. [11] later showed that performance of such hand-crafted detectors dropped sharply for small, partially occluded, or unusually posed pedestrians. These results motivated the shift toward learned features.

2.2 Deep Learning Object Detectors

Convolutional neural networks changed object detection by learning features directly from labeled images. Two-stage detectors, of which Faster R-CNN [10] is the best-known example, first propose candidate regions and then classify and refine each one. They achieve strong localization but require more computation per image. Single-shot detectors remove the separate proposal stage. SSD [12] predicts class scores and box offsets from feature maps at several scales, which helps with objects of different sizes. YOLO [1] went further by treating detection as a single regression problem over a grid, so that one network evaluates the entire image at once. This design made real-time operation realistic and led to a long series of successors, including YOLOv3 [2], YOLOv4 [3], YOLOv7 [4], and the Ultralytics YOLOv8 family [5].

These models are usually trained on the COCO dataset [6], which contains images of everyday scenes annotated with bounding boxes for 80 object categories in the detection challenge. Six of those categories (person, bicycle, car, motorcycle, bus, and truck) correspond directly to the road users of interest in this work. Because the weights are publicly available, a project can start from a model that already recognizes these classes and can be improved later through fine-tuning on local footage. More recent work has also explored transformer-based real-time detectors such as RT-DETR [18], which remove the need for non-maximum suppression and offer an alternative to the YOLO line. Table 1 summarizes the main families of approaches and their typical trade-offs.

Approach	Typical strengths	Typical weaknesses
Background subtraction (mixture models)	Fast; needs no training; suits a fixed camera	Sensitive to shadows, lighting, and camera motion; merges close objects; no class labels
HOG features with a linear classifier	Interpretable; effective for upright people	Hand-crafted features; weak under occlusion and unusual poses; single class
Two-stage CNN (Faster R-CNN)	Strong localization accuracy	Heavier computation; harder to run in real time on a CPU
Single-shot detector (SSD)	Faster than two-stage models; multi-scale prediction	Small objects can be missed
YOLO family	One pass per frame; several model sizes; mature tooling	Tiny or heavily occluded objects remain difficult without fine-tuning

Table 1. Comparison of common approaches to detecting road users.

2.3 Multi-Object Tracking

A detector reports objects frame by frame but does not know that the car in one frame is the same car in the next. Counting, trajectory analysis, and dwell-time estimation all need that continuity, so detection is normally followed by tracking. The tracking-by-detection paradigm links the detections of consecutive frames into tracks. SORT [7] predicts the next position of each track with a Kalman filter [19] and assigns new detections to existing tracks by solving an assignment problem on box overlap with the Hungarian algorithm [20]. DeepSORT [8] adds an appearance descriptor learned by a small network, which reduces identity switches when objects cross paths or disappear briefly. ByteTrack [9] observed that detections with low confidence scores often belong to partially occluded objects, and it keeps them in a second association stage instead of discarding them. BoT-SORT [21] combines motion and appearance cues with camera motion compensation. Open-source detection libraries now ship with ready-made tracker configurations, which makes it straightforward to attach a robust tracker to a pretrained detector.

Tracking quality is measured with metrics that go beyond detection accuracy. The CLEAR MOT metrics [13] combine missed objects, false alarms, and identity switches into a single score, the IDF1 measure [22] focuses on how consistently identities are maintained, and HOTA [23] balances detection

and association quality. Public benchmarks such as MOT16 [24] and UA-DETRAC [25] provide annotated sequences on which trackers can be compared, and KITTI [26] offers driving scenes with both vehicles and pedestrians.

2.4 Counting and Flow Analysis

Once objects are detected and tracked, two families of statistics are commonly derived. Screen-line counting records how many trajectories cross a reference line, in the same way that a human observer tallies vehicles passing a marker on the road. Region-based analysis instead measures how many objects are inside a region at a given moment, or how long each object stays there. The first family supports flow estimation and turning-movement studies, and the second supports studies of queueing, waiting, and crowding. Both depend on stable identities, because an identity switch can create or remove a crossing, and both are sensitive to the choice of reference point on the bounding box. Several practical systems therefore combine counting lines with polygonal regions, and they report results by class so that the contribution of heavy vehicles, two-wheelers, and pedestrians can be examined separately.

Counting with a single camera also has well-known limits. Objects that are hidden behind others cannot be counted, objects that appear very small near the horizon are detected less reliably, and perspective makes the apparent speed of an object depend on its distance from the camera. These limits explain why published counting systems usually report errors for specific viewpoints and conditions, and why a claim of high accuracy is meaningful only when the scene, the camera position, and the counting rules are described.

2.5 Gap Addressed by This Work

Most published detectors and trackers are evaluated on benchmark datasets and are described at the level of network architecture. Fewer accessible works describe how to assemble these components into a complete, explainable application that combines vehicles and pedestrians, produces directly useful statistics, and runs on commodity hardware. This paper addresses that practical gap. It does not propose a new detector. Instead, it contributes a clear system design, a set of well-defined analytics rules, and an evaluation procedure that a reader can reproduce on their own footage.

3. PROBLEM DEFINITION AND DESIGN REQUIREMENTS

The problem can be stated as follows. Given a video sequence of a road scene captured by a roughly fixed camera, produce for each analysis interval (a) the number of vehicles of each class and of pedestrians that crossed a user-defined line in each direction, (b) the number of pedestrians present inside a user-defined zone, and (c) a record of moments when a pedestrian and a vehicle came closer than a chosen threshold. The output must be reproducible, exportable as a table, and presented in a form that a non-specialist can read.

Several requirements guided the design. The system must run on a typical laptop, which may have only a CPU, so the model size and the frame processing rate must be adjustable. It must handle several input types, including recorded files for repeatable experiments and live streams for demonstration. It must be modular, so that the detector, the tracker, or the analytics rules can be replaced independently. It must avoid collecting personally identifying information; therefore it should reason about positions and classes only, and never about faces or license plates. Finally, it must be honest about its outputs. Counts are estimates, proximity indicators are not safety verdicts, and the system must not make claims, such as detecting criminal behavior, that its design cannot support.

The scope is deliberately limited. The system does not attempt license plate recognition, face recognition, identification of individuals, recognition of intent or behavior, or enforcement tasks such as issuing penalties for speeding. It also does not attempt to explain why an event occurred. It measures where objects are and how they move, and it leaves interpretation to people who know the site.

Three assumptions simplify the problem. The camera is assumed to be fixed during analysis, so that the counting line and the zone keep a stable meaning. The camera is assumed to view the scene from an elevated oblique angle, as is typical for pole-mounted or building-mounted cameras. The pretrained detector is assumed to recognize the six target classes well enough for exploratory analysis, although the evaluation protocol in Section 6 exists precisely to test this assumption on real footage.

4. PROPOSED METHODOLOGY

The proposed system follows a detect-track-analyze architecture, shown in Figure 1. Each stage consumes the output of the previous one and produces a well-defined data structure, which keeps the interfaces between stages narrow. The following subsections describe the stages in order.

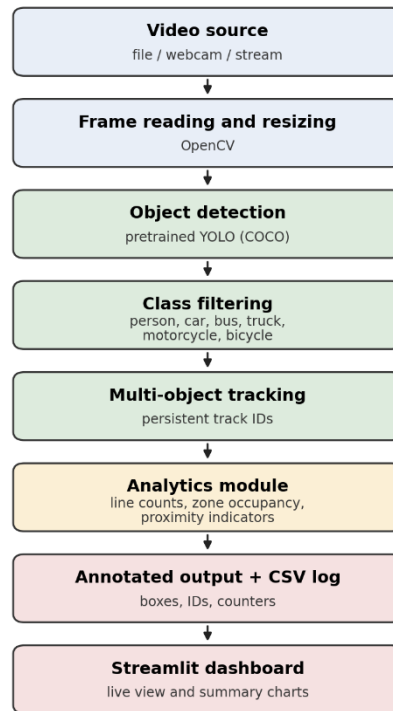


Fig. 1. Overall architecture of the proposed detect-track-analyze pipeline.

4.1 Video Acquisition and Preprocessing

Frames are read sequentially with OpenCV [27], which supports video files, local cameras, and network streams through a common interface. Each frame is an array of pixel intensities in blue-green-red order. Before detection, frames are resized so that the longer side matches the input resolution of the model, and the aspect ratio is preserved by padding. A larger input resolution improves the recall of small and distant objects but increases the cost per frame, so the resolution is exposed as a configurable parameter. When the processing rate falls below the video frame rate, the system can skip frames and process every n -th frame. Skipping lowers the computational load, but it also increases the distance an object moves between processed frames, which makes tracking harder. The value of n therefore has to be chosen together with the expected object speed.

4.2 Object Detection

Detection is performed by a pretrained YOLO model from the Ultralytics library [5]. For each frame, the model returns a set of bounding boxes, each with a class label and a confidence score. Boxes whose score is below a confidence threshold are discarded, and overlapping boxes of the same class are merged by non-maximum suppression using an intersection-over-union criterion. The intersection over union of two boxes A and B is defined in Equation 1.

$$IoU(A, B) = |A \cap B| / |A \cup B| \quad (1)$$

A moderate confidence threshold, commonly in the range of 0.25 to 0.5, is a reasonable starting point. A lower value increases recall but admits more false positives such as billboards, shadows, or statues mistaken for people. A higher value suppresses these errors but misses partially visible objects. Because the best value depends on the scene, the threshold is adjustable from the dashboard.

The model is not fine-tuned in the present work. It is used exactly as distributed, with weights trained on COCO. This choice is intentional: it keeps the system reproducible and avoids any suggestion that a new model was trained. The consequence is that detection quality on local footage must be measured rather than assumed, which is the purpose of Section 6. Smaller model variants run faster on a CPU, whereas larger variants detect small objects more reliably. The user can therefore trade speed for accuracy by selecting a different model size.

4.3 Class Filtering and Grouping

The COCO label set contains many categories that are irrelevant to road analysis, such as animals, furniture, and food. After detection, the system keeps only the six classes listed in Table 2 and assigns each to a group. Grouping allows the analytics layer to treat all motorized vehicles together or to report each class separately. Filtering early also reduces the load on the tracker and removes a class of spurious detections from the output.

COCO class	Class index	Group used in analytics
person	0	Pedestrian
bicycle	1	Two-wheeler
car	2	Motor vehicle
motorcycle	3	Two-wheeler
bus	5	Motor vehicle
truck	7	Motor vehicle

Table 2. Classes retained after filtering and the group assigned to each.

A limitation of this grouping is that a person riding a bicycle or motorcycle may be detected both as a person and as a two-wheeler. The system treats the two detections as separate objects unless they overlap strongly, in which case the rider is merged with the vehicle by a simple overlap rule. This is an approximation, and the evaluation protocol checks how often it fails.

4.4 Multi-Object Tracking

Tracking converts per-frame detections into trajectories. The system uses one of the tracker configurations supplied with the detection library, such as ByteTrack [9] or BoT-SORT [21]. Each detection is either matched to an existing track or starts a new one. Tracks that remain unmatched for a number of frames are terminated. Every active track carries a unique integer identifier, the class label of the object, and a history of box centers. These histories drive all later analysis.

Two properties of the tracker matter for counting. The first is identity stability. If a single car receives a new identifier halfway across the frame, the line-crossing logic may count it twice. The second is track fragmentation, in which a brief occlusion splits one trajectory into two. Both effects are reduced by associating low-confidence detections, by tolerating short gaps, and by choosing a camera position that limits occlusion. They cannot be removed entirely, which is why the evaluation protocol includes an explicit check for identity switches.

4.5 Virtual Line Counting

Directional counting uses a virtual line defined by two endpoints, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, drawn by the user over the scene. For a point $p = (x, y)$, the signed side value is computed as in Equation 2.

$$s(p) = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1) \quad (2)$$

The sign of $s(p)$ indicates on which side of the line the point lies. For each track, the system stores the sign observed at the previous processed frame. When the sign at the current frame differs from the stored one, and the segment between the two centroid positions actually intersects the finite line segment, a crossing is recorded. The direction of the crossing follows from the change of sign, so that movements from the positive side to the negative side and the reverse are counted separately. Each crossing event is stored together with the track identifier, the class, the direction, and the frame number. Three safeguards reduce counting errors. First, each track identifier can contribute at most one crossing per direction, which prevents repeated counts when an object lingers near the line. Second, a narrow hysteresis band around the line ignores sign changes caused by small jitter in the box position. Third, the reference point for a vehicle is the box center, whereas for a pedestrian it is the bottom center of the box, which approximates the point of ground contact. The geometry is illustrated in Figure 2.

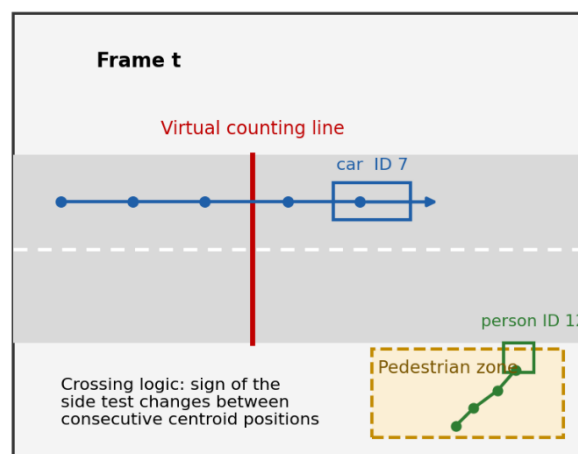


Fig. 2. Virtual counting line and pedestrian zone, with tracked centroids.

4.6 Zone Occupancy and Density

A pedestrian zone is defined as a polygon over areas such as a crossing, a bus stop, or a waiting area. At each processed frame, the system tests whether the reference point of every tracked pedestrian lies inside the polygon, using a standard point-in-polygon test. The occupancy at time t is then the number

of such pedestrians, as in Equation 3, where c_i is the reference point of pedestrian i and Z is the zone.

$$O(t) = \sum_i 1[c_i \in Z] \quad (3)$$

The occupancy series can be averaged over a time window, and its maximum can be reported to describe peak crowding. A density value may also be computed by dividing the occupancy by the zone area. Because the camera views the scene obliquely, a pixel near the top of the image covers more ground than a pixel near the bottom. Density in image units is therefore only a relative measure and should not be read as people per square meter unless a ground-plane calibration has been performed.

4.7 Pedestrian-Vehicle Proximity Indicator

Road safety studies often look for situations in which pedestrians and vehicles come close to one another, since such moments can reveal hazardous locations before collisions occur. The present system includes a deliberately simple indicator for this purpose. For each pedestrian i and vehicle j , the image distance d_{ij} between the pedestrian's ground contact point and the nearest edge of the vehicle box is computed. To compensate roughly for perspective, the distance is divided by the pedestrian's box height h_i , which serves as a local scale reference. A proximity event is recorded when the normalized distance falls below a threshold, as in Equation 4.

$$d_{ij} / h_i < \tau \quad (4)$$

This indicator should be interpreted with care. It uses no speed information, no time-to-collision estimate, and no calibrated ground distances. It therefore flags closeness in the image, not danger in the world. It is included as an exploratory measure that can help a human reviewer find moments worth examining, and the paper makes no claim that it detects near-misses or collisions in a validated sense. Estimating speed and true separation would require camera calibration, which is discussed in Section 8.

4.8 Algorithm Summary

Algorithm 1 summarizes the processing loop that links the stages described above.

Algorithm 1. Detect-track-analyze loop

Input: video source V , model M , line L , zone Z , threshold τ
 Initialize counters, crossing set S , event log E
 for each frame f read from V do
 (skip f if frame-skip rule applies)
 $D \leftarrow M.\text{detect}(f)$ and keep target classes
 $T \leftarrow \text{tracker.update}(D)$
 for each track k in T do
 append reference point of k to its history
 if side sign of k changed across L and (k, dir) not in S then
 increment count[class(k), dir]; add (k, dir) to S
 occupancy $\leftarrow$ number of pedestrian tracks inside Z
 for each pedestrian-vehicle pair (i, j) do
 if $d_{ij} / h_i < \tau$ then log proximity event in E
 draw boxes, IDs, counters on f ; write f and E
 return counts, occupancy series, event log E

5. IMPLEMENTATION AND DASHBOARD

5.1 Software Stack

The system is implemented in Python, which offers mature libraries for every stage of the pipeline. Table 3 lists the main components and the role each one plays. All of them are open source and can be installed with the standard Python package manager, which is important for reproducibility and for use in teaching environments.

Component	Role in the system
Python	Application logic and orchestration
OpenCV	Video input and output, drawing, geometric tests
Ultralytics YOLO	Pretrained detection and integrated tracking
NumPy	Array operations for boxes and trajectories
pandas	Event logging, aggregation, and CSV export
Streamlit	Interactive browser-based dashboard

Table 3. Software components and their roles.

5.2 Configuration Parameters

A small set of parameters controls the behavior of the system. These include the model size, the input resolution, the confidence threshold, the frame-skip interval, the tracker type, the endpoints of the counting line, the vertices of the pedestrian zone, and the proximity threshold τ . Keeping these values in a single configuration object has two benefits. It lets the user adapt the system to a new camera by changing parameters rather than code, and it makes each experiment documented, because the configuration can be saved alongside the results.

Table 4 offers starting ranges for the main parameters. These ranges are only starting points drawn from common practice. They are not tuned values, and each should be adjusted by testing on the footage of the intended site.

Parameter	Effect of increasing it	Suggested starting point
Confidence threshold	Fewer false alarms; more missed objects	0.25 to 0.5
Input resolution	Better recall for small objects; slower processing	Model default, then test larger
Model size	Better accuracy; slower on a CPU	Smallest for live use
Frame-skip interval	Faster processing; harder tracking	1 to 3 frames
Proximity threshold	More proximity events flagged	Tune by reviewing flagged clips

Table 4. Starting ranges for the main configuration parameters.

5.3 Dashboard and Outputs

The Streamlit dashboard provides a browser interface. The user uploads a video or selects a camera, adjusts the confidence threshold and model size, and starts processing. The dashboard shows the annotated video with boxes, class labels, and track identifiers, together with running counters for each class and direction. Summary charts display counts over time, the occupancy of the pedestrian zone, and the number of proximity events. When processing ends, the user can download the annotated video and a CSV file containing the event log. The CSV format was chosen because it opens in any spreadsheet program, which allows further analysis without special software.

The event log follows a simple, documented schema, summarized in Table 5, so that results can be reloaded and analyzed later. Each row describes one event, such as a line crossing or a proximity episode, and every row carries enough information to locate the moment in the video.

Field	Meaning
frame_index	Index of the processed frame
time_s	Time from the start of the video, in seconds
track_id	Temporary identifier assigned by the tracker
class_name	Detected class, such as car or person
event_type	Line crossing, or proximity event
direction	Crossing direction, where applicable
x, y	Reference point of the object, in pixels

Table 5. Fields of the exported event log.

5.4 Hardware Considerations

The pipeline was designed to be usable on a Windows laptop. On a CPU-only machine, the smallest model variants are the practical choice for live streams, and larger variants are better reserved for offline processing of recorded video. If a compatible graphics processor is available, the same code runs substantially faster without modification, because the detection library selects the available device. Processing speed is reported in frames per second, as defined in Section 6, and is a property of the machine, the model size, and the input resolution together, not of the pipeline alone. For that reason, any speed figure should always be quoted together with the hardware on which it was measured.

5.5 Robustness and Error Handling

A video application meets many unexpected conditions, and the pipeline is designed to fail in a controlled way. If a file cannot be opened or a frame cannot be decoded, the system reports a clear message and stops without producing a misleading partial result. If a network stream drops, the reader attempts to reconnect for a limited number of tries and then ends the session. If a frame contains no detections, the tracker still advances, so that existing tracks age correctly and are not terminated or extended by mistake. When the processing speed falls far below the frame rate of a live stream, the dashboard displays a warning, because the counts of a system that silently drops many frames cannot be trusted. Each of these behaviors protects the user from drawing conclusions from output that looks normal but is not.

5.6 Typical Usage Workflow

A typical session begins with the choice of a video source and a short inspection of a few frames. The user then draws the counting line across the part of the road where objects are well separated and moving in clear directions, and outlines the pedestrian zone around the area of interest. A trial run on a short segment follows, during which the user compares the annotated output with what is visible in the video and adjusts the confidence threshold and the model size. Only after this calibration step is the full video processed. At the end of the run, the CSV log is exported, the counts are compared with a manual tally for a sample of the footage, and the configuration used is saved with the results. This workflow keeps a human in the loop at the points where judgment is most valuable, namely the placement of the geometry and the verification of the output.

5.7 Illustrative Deployment Scenario

To make the workflow concrete, consider a hypothetical study of a pedestrian crossing outside a college gate during the evening hours. This scenario is illustrative only and does not report measured results. A camera is mounted on a first-floor balcony with an oblique view of the road and the crossing.

The analyst records several short clips at different times, draws the counting line on the carriageway just before the crossing, and outlines the crossing area and the adjacent waiting area as a pedestrian zone.

The pipeline then reports, for each clip, the number of cars, two-wheelers, and buses that pass in each direction, the number of pedestrians waiting in the zone over time, and a list of moments at which a pedestrian and a vehicle came within the chosen proximity threshold. The analyst reviews the flagged moments in the annotated video. Some of them are genuine close encounters, such as a pedestrian stepping off the curb as a motorcycle passes, whereas others are artifacts of perspective, such as a vehicle in the far lane that merely appears close in the image. This review step is essential. It shows why the indicator is useful for focusing attention but cannot replace human judgment.

Finally, the analyst compares the automatic counts with a manual tally for two of the clips and records the differences by class and direction. If the differences are large for a particular class, for example because auto-rickshaws are being labeled inconsistently, the analyst notes that class as a candidate for fine-tuning. If they are small, the pipeline can be used with reasonable confidence at that site and camera angle. In both cases the study produces not only statistics about the road but also a documented estimate of how far those statistics can be trusted.

6. EVALUATION METHODOLOGY

A system of this kind should be judged on several levels, because good detection does not imply good tracking, and good tracking does not imply accurate counts. This section describes the metrics and the procedure proposed for evaluating the pipeline. The procedure is designed so that anyone can apply it to their own footage and obtain defensible numbers.

6.1 Detection Metrics

A detection is counted as a true positive when its intersection over union with a ground-truth box exceeds a chosen threshold and the class label matches. Unmatched detections are false positives, and unmatched ground-truth boxes are false negatives. Precision and recall follow from these counts, as shown in Equation 5, and their harmonic mean is the F1 score, shown in Equation 6.

$$P = TP / (TP + FP), \quad R = TP / (TP + FN) \quad (5)$$

$$F_1 = 2PR / (P + R) \quad (6)$$

Average precision, computed from the precision-recall curve for each class and averaged over classes to give mean average precision, is the standard summary used in the COCO benchmark [6]. Reporting these values separately for pedestrians and for each vehicle class is important, since a model that performs well on cars may be much weaker on bicycles or on distant people.

6.2 Tracking Metrics

Tracking is evaluated with the CLEAR MOT accuracy score [13], the IDF1 score [22], and, where annotation allows, HOTA [23]. The multiple object tracking accuracy is defined in Equation 7, where FN, FP, and IDSW denote the numbers of missed objects, false alarms, and identity switches, and GT denotes the number of ground-truth objects, each summed over all frames.

$$MOTA = 1 - (FN + FP + IDSW) / GT \quad (7)$$

Full tracking annotation of long videos is expensive. As a practical alternative, the protocol suggests annotating short representative segments and manually counting identity switches by reviewing the annotated output video, recording each moment at which an object visibly changes its identifier.

6.3 Counting Metrics

For the end user, the most important question is whether the final counts are right. The protocol therefore compares the system's count N_{sys} with a manually obtained count N_{gt} for each clip, class, and direction. The relative counting error is defined in Equation 8, and the mean absolute error over all clips summarizes overall behavior.

$$E = |N_{sys} - N_{gt}| / N_{gt} \times 100\% \quad (8)$$

Manual counts should be produced by two independent observers who watch the clip and tally each crossing. Differences between the observers are resolved by re-watching, and the level of disagreement before resolution is reported, because it indicates how much error even human counting contains. Reporting both the signed error and the absolute error is useful, since a system that overcounts in one direction and undercounts in another can look accurate in total while being unreliable in detail.

6.4 Efficiency Metrics

Throughput is measured as the number of processed frames divided by the total processing time, as in Equation 9, and latency as the time needed to process one frame.

$$FPS = N_{frames} / T_{total} \quad (9)$$

Both quantities should be reported with the hardware description, model size, and input resolution. Table 6 collects the metrics used at each level of evaluation.

Level	Metrics	Ground truth needed
Detection	Precision, recall, F1, mAP	Annotated boxes on sampled frames
Tracking	MOTA, IDF1, identity switches	Annotated tracks or reviewed video
Counting	Relative error, mean absolute error	Manual counts by two observers
Efficiency	Frames per second, latency	None; timing only

Table 6. Metrics proposed for each level of evaluation.

6.5 Evaluation Data and Procedure

Two kinds of data are recommended. Public benchmarks such as UA-DETRAC [25], MOT16 [24], and KITTI [26] allow results to be compared with published work, although their camera viewpoints and annotation conventions differ from those of a local deployment. Locally recorded clips, collected with the camera position that will actually be used, show how the pipeline behaves under real conditions. A balanced local test set should include different times of day, different traffic densities, and, where possible, adverse conditions such as rain or glare. Each clip should be long enough to contain a meaningful number of crossings, and the clips used for tuning thresholds should be kept separate from those used for the final report, so that the reported numbers are not inflated by tuning.

It is worth stating plainly that this paper reports a design and an evaluation procedure, not a leaderboard result. Any accuracy figure for a particular deployment must come from running the procedure above on that deployment's footage. Quoting benchmark numbers for the underlying detector as if they described the whole pipeline would be misleading, because the final counts also depend on the tracker, the camera geometry, and the analytics rules.

6.6 Threats to Validity

Any evaluation of this kind faces threats to validity that should be acknowledged. Internal threats arise from the way the experiment is run. If thresholds are tuned on the same clips that are used for reporting, the results will look better than they should. If the manual counts are produced by a person who has already seen the system output, they may be biased toward it. These risks are reduced by separating tuning clips from test clips and by asking observers to count before viewing the annotated video.

External threats concern how far the results can be generalized. A system that performs well at one junction, in daylight, with a particular camera angle, may perform very differently at another site. The results of any evaluation should therefore be described as applying to the conditions tested, and a deployment at a new site should be preceded by a fresh round of testing. Finally, the choice of metrics itself shapes conclusions. A system can score well on detection metrics and still produce inaccurate counts because of identity switches, which is why the protocol reports metrics at several levels instead of relying on any single number.

7. DISCUSSION

7.1 Practical Challenges and Mitigation

Real road scenes expose a pipeline to many difficulties that benchmark images hide. Table 7 lists the most important ones, their typical effect on the output, and the mitigation measures available within the proposed design.

Challenge	Typical effect	Mitigation
Occlusion	Missed objects; identity switches	Associate low-confidence detections; raise the camera angle
Low light, rain, glare	Lower recall; more false alarms	Adjust threshold; improve exposure; fine-tune on local data
Small, distant objects	Missed detections	Increase input resolution; define the line closer to the camera
Perspective distortion	Unreliable distances and density	Calibrate the ground plane with a homography
Camera shake or drift	Shifted line and zone	Use a fixed mount; stabilize frames
Double counting	Inflated totals	One crossing per identity; hysteresis band

Table 7. Common challenges and mitigation measures.

Occlusion deserves special attention. In dense traffic, a bus can hide several pedestrians, and a group of people walking together can be detected as fewer individuals than are actually present. Camera placement is often the most effective remedy. A higher viewpoint reduces overlap between objects, and a line placed where objects are well separated yields more reliable counts than one placed in a congested area.

Class confusion is another source of error. Vans and pickup trucks may be labeled as cars or trucks depending on the viewing angle, and auto-rickshaws, which are common in many Indian cities, are not a COCO class and may be detected as cars, motorcycles, or not at all. This is a clear example of the limits of using a model trained on a general dataset. Fine-tuning on locally annotated images that include such vehicles would be the appropriate response, and it is identified as future work.

7.2 Comparison with Sensor-Based Counting

It is useful to compare the video-based approach with the traditional sensors it might complement. Inductive loops and tubes are robust and well understood, and they count vehicles accurately in many conditions, but they require physical installation in the road surface, they do not observe pedestrians, and they cannot distinguish among vehicle types beyond simple length or axle classes. Radar units are less intrusive but offer limited information about object class. Video analysis requires no work on the road surface, can classify several kinds of road users, and leaves a visual record that can be audited. Its disadvantages are sensitivity to lighting and weather, the need for a clear view, and the privacy considerations discussed later in this section. In practice the approaches are complementary, and a video system can be validated or calibrated against a conventional sensor at a pilot site.

7.3 Strengths of the Approach

The pipeline has several strengths. It uses a single camera and ordinary video, so the cost of adoption is low. It reports pedestrians and vehicles together, which supports analysis of their interaction rather than of each group in isolation. Its modular structure allows individual components to be upgraded as better models appear. The output is available both as a human-readable annotated video and as a machine-readable log, so a reviewer can always check a suspicious number against the footage that produced it. This traceability is valuable for building trust in automated statistics.

7.4 Potential Applications

Several applications follow naturally from the outputs of the system. Traffic engineers can use directional counts by class to study demand patterns and to evaluate signal timing. Pedestrian occupancy near a crossing or a bus stop can inform decisions about waiting-area size and crossing duration. Proximity indicators can help safety auditors locate places where road users frequently come close, so that site visits can be prioritized. Campus administrators can use the same tools to study gate traffic and pedestrian flow, and researchers can use the event log as input for further statistical analysis. In each case, the system supports human decisions and does not replace them.

7.5 Privacy, Ethics, and Responsible Use

Video analytics in public spaces raises legitimate privacy concerns, and a responsible design must address them from the start. The proposed system analyzes only the position and class of each object. It performs no face recognition, no license plate reading, and no attempt to identify individuals. Track identifiers are temporary labels that exist only to link detections across frames, and they carry no personal meaning. Aggregated counts and anonymous trajectories are sufficient for the intended purposes.

Several further practices are recommended. Footage should be collected only with proper authorization and for a stated purpose. Retention periods should be short, and raw video should be deleted once the required statistics have been extracted. Signs that inform the public of camera use are advisable. In India, the handling of digital personal data is governed by the Digital Personal Data Protection Act, 2023 [28], and any deployment should be reviewed against current legal requirements and institutional ethics policies. Finally, the output should never be presented as evidence of intent or wrongdoing. The system measures movement and proximity; it does not detect crime, and its design intentionally avoids claims of that kind.

7.6 Limitations

The study has several limitations that readers should keep in mind. The detector is used without fine-tuning, so its behavior on local vehicle types, clothing, and lighting is untested until the evaluation protocol is applied. The system assumes a fixed camera and a roughly oblique viewpoint, and it will perform poorly on footage from moving vehicles or drones. Distances and densities are expressed in image units, not in physical units. The proximity indicator has not been validated against recorded conflicts or collisions. Finally, this paper presents the design and evaluation procedure rather than a large-scale quantitative comparison, and conclusions about accuracy in any specific setting must be drawn from measurements made in that setting.

8. FUTURE WORK

Several extensions would make the system more accurate and more useful. The most important is fine-tuning the detector on locally collected and annotated footage, including vehicle types that are underrepresented in COCO. Even a modest annotated set can improve recall for such classes, and the evaluation protocol would show the size of the improvement.

A second extension is camera calibration. By identifying four or more ground-plane points with known real-world positions, a homography can map image coordinates to ground coordinates. This would allow the system to estimate speed, true separation distance, and density in people per square meter, and it would turn the proximity indicator into a measure with physical meaning, such as time to closest approach.

Further directions include testing transformer-based detectors such as RT-DETR [18], exporting models to optimized inference formats for faster execution on edge devices, combining several cameras to follow objects across a wider area, and improving robustness to rain, glare, and night-time conditions. Another direction is to build an annotated local benchmark that other students and researchers could share, since the absence of such data is a common obstacle for work in this area.

Human factors also deserve attention. If the dashboard is to be used by traffic staff and not only by its developers, it should explain its outputs in plain language, show uncertainty where it is known, and make it easy to review the video behind each number. Periodic revalidation is advisable, since changes in camera position, season, or traffic composition can quietly reduce accuracy long after the first evaluation.

9. CONCLUSION

This paper presented a computer vision pipeline for analyzing traffic and pedestrian activity from video feeds. The pipeline joins a pretrained YOLO detector, a multi-object tracker, and a small set of transparent analytics rules for line counting, zone occupancy, and proximity indicators. It is implemented with open-source Python tools and offered through a Streamlit dashboard that runs on an ordinary laptop. The paper explained each design choice, defined the geometric rules in a form that can be reproduced, and proposed an evaluation protocol that covers detection, tracking, counting, and efficiency.

The central message of the work is that a useful traffic and pedestrian analytics tool can be assembled from accessible components, provided that its outputs are treated as measurements with known error and not as facts. Accurate counts depend on camera placement, careful parameter choice, and honest validation against manual observation. Privacy and ethical safeguards are equally important, and the design therefore avoids identification of individuals and unsupported claims. Future work on fine-tuning, calibration, and local benchmarking will show how far the approach can be pushed, but the present design already provides a clear and explainable foundation for further study.

REFERENCES

- [1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
- [2] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv:1804.02767, 2018.
- [3] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv:2004.10934, 2020.
- [4] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 7464–7475.
- [5] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," software, 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [6] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. European Conf. Computer Vision (ECCV)*, 2014, pp. 740–755.
- [7] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2016,

pp. 3464–3468.

- [8] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2017, pp. 3645–3649.
- [9] Y. Zhang *et al.*, “ByteTrack: Multi-object tracking by associating every detection box,” in *Proc. European Conf. Computer Vision (ECCV)*, 2022, pp. 1–21.
- [10] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.
- [11] P. Dollár, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: An evaluation of the state of the art,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.
- [12] W. Liu *et al.*, “SSD: Single shot multibox detector,” in *Proc. European Conf. Computer Vision (ECCV)*, 2016, pp. 21–37.
- [13] K. Bernardin and R. Stiefelhagen, “Evaluating multiple object tracking performance: The CLEAR MOT metrics,” *EURASIP J. Image and Video Processing*, vol. 2008, Art. no. 246309, 2008.
- [14] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 1999, pp. 246–252.
- [15] Z. Zivkovic, “Improved adaptive Gaussian mixture model for background subtraction,” in *Proc. Int. Conf. Pattern Recognition (ICPR)*, 2004, vol. 2, pp. 28–31.
- [16] N. Buch, S. A. Velastin, and J. Orwell, “A review of computer vision techniques for the analysis of urban traffic,” *IEEE Trans. Intelligent Transportation Systems*, vol. 12, no. 3, pp. 920–939, 2011.
- [17] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893.
- [18] Y. Zhao *et al.*, “DETRs beat YOLOs on real-time object detection,” in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [19] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *J. Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
- [20] H. W. Kuhn, “The Hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, no. 1–2, pp. 83–97, 1955.
- [21] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, “BoT-SORT: Robust associations multi-pedestrian tracking,” arXiv:2206.14651, 2022.
- [22] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, “Performance measures and a data set for multi-target, multi-camera tracking,” in *ECCV Workshops*, 2016, pp. 17–35.
- [23] J. Luiten *et al.*, “HOTA: A higher order metric for evaluating multi-object tracking,” *Int. J. Computer Vision*, vol. 129, no. 2, pp. 548–578, 2021.
- [24] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, “MOT16: A benchmark for multi-object tracking,” arXiv:1603.00831, 2016.
- [25] L. Wen *et al.*, “UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking,” *Computer Vision and Image Understanding*, vol. 193, Art. no. 102907, 2020.
- [26] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.
- [27] G. Bradski, “The OpenCV library,” *Dr. Dobbs's Journal of Software Tools*, 2000.
- [28] Government of India, Ministry of Law and Justice, “The Digital Personal Data Protection Act, 2023,” 2023.