International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Integrated Engine for Autonomous Driving Using YOLO and SLAM

Varun Kakapuri[1], Vaibhav Raut[2], Rohit Shilimkar[3], Pratima Patil[4]

[1,2,3]Student, Department of Information Technology, Trinity Academy of Engineering, Pune
[5]Professor, Department of Information Technology, Trinity Academy of Engineering, Pune

Abstract
The rapid advancement in autonomous driving technology necessitates the development of highly efficient systems that ensure accurate object detection, precise localization, and reliable navigation. This project presents an integrated engine combining YOLO (You Only Look Once) for real-time object detection and SLAM (Simultaneous Localization and Mapping) for accurate localization and mapping, enhancing the decision-making and obstacle avoidance capabilities of autonomous vehicles. YOLO's deep learning framework allows for the fast detection of various objects in the vehicle's environment, while SLAM builds a dynamic, real-time map that ensures the vehicle maintains awareness of its surroundings. The integration of these technologies provides a robust solution for navigating complex and dynamic environments, overcoming challenges such as dynamic obstacle avoidance, lane tracking, and real-time path planning. This project aims to contribute to the development of safer and more efficient autonomous vehicles, focusing on real-time performance and the seamless interaction between detection and localization systems.

Keywords: YOLO (You Only Look Once), SLAM (Simultaneous Localization and Mapping), Object Detection, Lane Detection, Obstacle Avoidance, Deep Learning.

1. Introduction

The evolution of autonomous driving technology hinges on the synergy between perception and localization systems that enable vehicles to navigate safely and efficiently in real-world environments. As self-driving systems become more complex, the need for accurate, real-time understanding of the vehicle's surroundings has become paramount. Two leading technologies at the forefront of this evolution are YOLO (You Only Look Once) and SLAM (Simultaneous Localization and Mapping). YOLO is a high-performance, deep learning-based object detection algorithm renowned for its ability to detect multiple objects within a scene in a single, swift pass. Its speed and precision make it highly suitable for the fast-paced decision-making demands of autonomous vehicles.
SLAM, on the other hand, offers the crucial ability to build a dynamic map of the environment while simultaneously tracking the vehicle's location within that map. By continuously updating spatial information from sensor data such as LiDAR, radar, and cameras, SLAM enables autonomous systems to navigate in both known and unfamiliar environments without the need for pre-constructed maps.
This research proposes an integrated engine that combines the rapid object detection capabilities of YOLOv8 with the spatial mapping strength of SLAM. The integration facilitates a unified environmental model where both static infrastructure and dynamic obstacles are mapped and identified in real time. This

fusion enhances the system's situational awareness, allowing it to make informed decisions about navigation, obstacle avoidance, and path planning. YOLOv8's improvements, such as anchor-free detection and scalable architecture, further support deployment across a range of hardware platforms, from resource-constrained systems to high-performance setups. Moreover, this integrated approach supports advanced decision-making through comprehensive data synthesis. The system not only detects and classifies objects but also accurately maps their positions relative to the vehicle. Such capabilities are essential for managing complex urban scenarios, reacting to unexpected changes, and ensuring passenger safety. In addition, robust preprocessing techniques enhance the reliability of sensor inputs, leading to improved performance in varied and challenging environments.

By merging state-of-the-art object detection and real-time localization, this paper presents a cohesive framework for autonomous driving that addresses the dual challenges of perception and navigation, laying the groundwork for safer and smarter autonomous systems.

2. Related Work

Autonomous driving has rapidly evolved due to significant advancements in artificial intelligence, sensor technology, and computational hardware. This evolution has driven extensive research in perception, localization, real-time decision-making, and infrastructure integration. This section discusses key research contributions relevant to autonomous driving, with a focus on deep learning for perception, SLAM for localization, lane and pothole detection, edge computing, and 3D mapping systems.

- **Deep Learning for Perception and Pothole Detection:** Deep learning models have become fundamental to perception in autonomous vehicles. CNN-based models like YOLO (You Only Look Once) are particularly effective due to their real-time object detection capabilities. Khan et al. (2024) propose a YOLOv8-based pothole detection system that demonstrates superior performance in terms of speed and accuracy compared to earlier models like YOLOv5. Their work emphasizes the importance of real-time hazard identification to enhance vehicle safety, and explores data augmentation methods to improve detection robustness across diverse road conditions. Similarly, Raja et al. (2022) introduce SPAS, a Smart Pothole-Avoidance Strategy using the Deep Deterministic Policy Gradient (DDPG) algorithm. This system uniquely incorporates real-time user feedback through speech and gesture recognition to fine-tune decision-making, achieving significant improvements in comfort and obstacle avoidance in VANET environments.
- **SLAM and Localization:** Simultaneous Localization and Mapping (SLAM) is crucial for autonomous navigation, enabling vehicles to localize themselves while constructing environmental maps. Frese et al. (2010) provide a user-centric overview of SLAM, categorizing its application into three levels: pre-mapping, black-box localization, and dynamic online mapping. They explore various SLAM architectures including 2D/3D pose graphs and feature-based methods, emphasizing their suitability for different application contexts. ORB-SLAM3 and other modern visual SLAM techniques now integrate multiple sensor inputs (e.g., stereo, monocular cameras) and advanced loop closure strategies to improve localization accuracy in real-world environments.
- **Lane Detection using Image Processing:** Lane detection remains a foundational task for path planning in autonomous vehicles. Suresh et al. (2022) address the limitations of existing systems under noisy conditions (e.g., fog, shadows, oil stains) and propose an Entropy-Based Fusion Model (EBFM) to improve detection under adverse visual conditions. They use clustering and heuristic line fitting methods for lane identification. Complementing this, Abdul Razak et al. (2022) employ a pipeline

using OpenCV, Gaussian blur, Canny edge detection, and the Hough Transform for robust two-lane detection. Their study shows that time-of-day impacts performance, with optimal detection results recorded in mid-afternoon lighting conditions.

- **Edge Computing and Real-Time Processing:** Real-time processing is essential for safe autonomous operation, especially when relying on computationally intensive deep learning models. The adoption of edge computing enables processing directly on the vehicle, thereby reducing latency and reliance on cloud infrastructure. Lightweight models like YOLOv8 are specifically optimized for edge deployment, ensuring fast and efficient decision-making. These edge systems also support V2X (Vehicle-to-Everything) communication, improving situational awareness through collaborative sensing and shared environmental data.
- **3D Mapping and Infrastructure Support:** The use of high-definition 3D maps further enhances vehicle localization and environment perception. Mizutani et al. (2020) propose a method for distributing Point Cloud Data (PCD) maps via roadside edge servers using the Autoware platform. Their system enables autonomous vehicles to dynamically download high-resolution maps in real-time, achieving accurate self-localization with minimal delay, particularly in urban environments. This infrastructure-based support helps vehicles remain updated with real-world changes such as construction zones or dynamic obstacles.

3. Methodology:

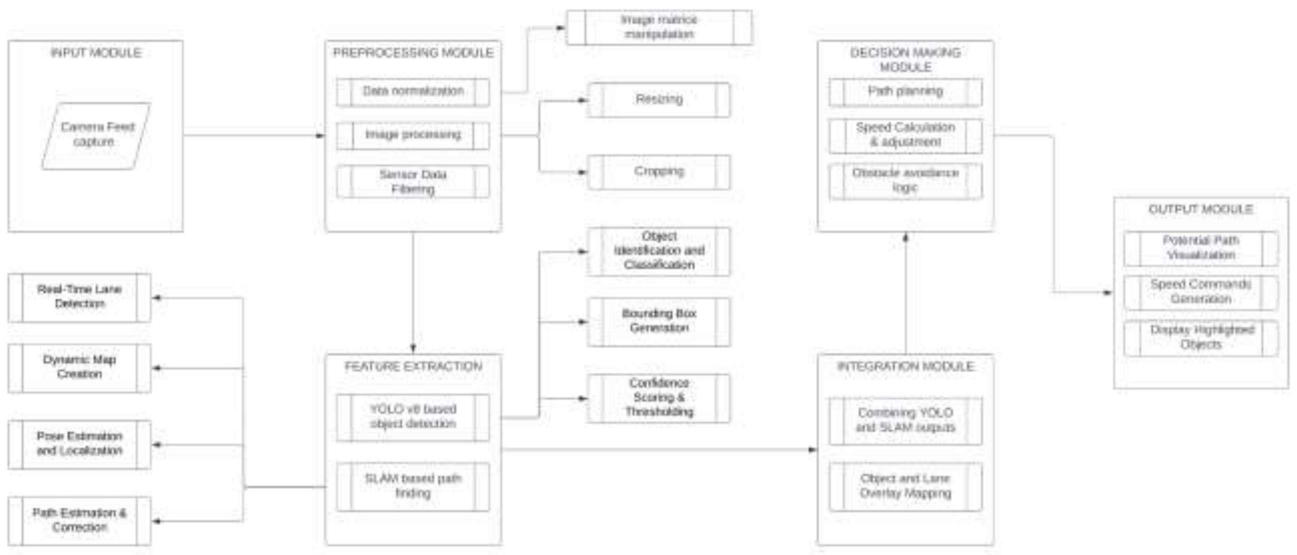

INPUT MODULE
Camera Feed capture
PREPROCESSING MODULE
Data normalization
Image processing
Sensor Data Filtering
Image matrice manipulation
Resizing
Cropping
DECISION MAKING MODULE
Path planning
Speed Calculation & adjustment
Obstacle avoidance logic
OUTPUT MODULE
Potential Path Visualization
Speed Commands Generation
Display Highlighted Objects
Object Identification and Classification
Bounding Box Generation
Confidence Scoring & Thresholding
Real-Time Lane Detection
Dynamic Map Creation
Pose Estimation and Localization
Path Estimation & Correction
FEATURE EXTRACTION
YOLO v8 based object detection
SLAM based path finding
INTEGRATION MODULE
Combining YOLO and SLAM ouputs
Object and Lane Overlay Mapping

Fig 1. System Architecture

-3.1 Data Acquisition

The system captures real-time video from on-board cameras (for example, a forward-facing dashcam or a mounted smartphone). These cameras are synchronized and calibrated so that each frame can be accurately related to the world. Video is streamed at sufficient frame rate (e.g. 30–60 FPS) and resolution to allow reliable perception. In parallel, static vehicle parameters (wheelbase, turning radius, height/width) and dynamic sensor data (e.g. IMU or wheel-encoder readings) are recorded. These parameters are stored for use by the planning and control algorithms to enforce vehicle kinematics constraints. Collectively, this module ensures that high-fidelity visual feeds and precise vehicle dimensions are available for the rest of the pipeline.

- **Cameras:** High-resolution forward-facing cameras record continuous video. Intrinsic parameters (focal length, distortion) and extrinsics (mounting position) are calibrated beforehand.
- **Vehicle parameters:** The car's turning radius, length, width, and other dimensions are measured or obtained from the vehicle model. This information is logged for use in collision-checking and path planning.
- **Additional sensors:** Any additional sensors (e.g. IMU, GPS, wheel encoders) are activated to provide pose priors or motion cues. Their raw outputs are time stamped and bundled with the video data.

3.2 Preprocessing

Raw inputs are preprocessed to produce clean, consistent data. Image frames are first converted to a standard color space and normalized (e.g. pixel intensities scaled to [0,1]). The images are then resized or cropped to match the input requirements of the perception networks (for example, YOLOv8 models often use square inputs such as 640×640 pixels. If multiple cameras are used, images are rectified to compensate for lens distortion or alignment differences. Noise-reduction filters (e.g. Gaussian blur or median filtering) may be applied to smooth out visual noise. Similarly, any time-series sensor data (IMU, wheel speed) is low-pass filtered or smoothed to remove jitter. These preprocessing steps ensure that every input frame and sensor reading fed to later modules is uniformly formatted and denoised.

- **Image normalization:** Scale and center pixel values so that network inputs have zero mean and unit variance.
- **Resizing and cropping:** Resize frames to the neural network input size (e.g. 640×640) and optionally crop to region of interest (e.g. remove sky or hood).
- **Noise filtering:** Apply spatial filters (Gaussian blur, bilateral filter) to suppress camera noise. For other sensors, use Kalman or simple low-pass filters.
- **Rectification:** If using stereo or fisheye cameras, rectify images to obtain undistorted views.

3.3 Feature Extraction

The core perception uses two parallel pipelines. The first is a YOLOv8-based object detector. YOLOv8 processes each preprocessed image in real time, outputting bounding boxes, class labels, and confidence scores for detected objects. YOLO accomplishes this in a single forward pass of a convolutional neural network: it directly regresses bounding box coordinates and class probabilities from the full image. In other words, one CNN predicts all object locations and classes at once, enabling high speed. YOLOv8 incorporates modern improvements (an anchor-free detection head and a state-of-the-art backbone) to balance accuracy and inference speed For example, the smallest YOLOv8 model (YOLOv8n) achieves around 37.3% mAP on the COCO benchmark with inference times on the order of 1 ms on a high-end GPU Each detection is output as a 2D bounding box (x,y coordinates in the image) plus a confidence score.

The second pipeline is a visual SLAM algorithm. SLAM takes the same camera frames and performs feature matching across consecutive frames. It extracts interest points (e.g. ORB features) and tracks them to triangulate a sparse 3D point cloud of the scene. Simultaneously, SLAM computes the vehicle's camera pose (position and orientation) for each frame by aligning the current image features with the map. In effect, SLAM produces a dynamic map of the environment and a live pose estimate for the vehicle. The SLAM module thus converts raw video into a structured world representation and continuous localization output. In addition, a lane-detection sub-module operates on the video to find painted road lanes (using techniques such as edge detection plus Hough transforms or a learned segmentation network). These detected lanes inform the drivable corridor and can be incorporated into the map as geometric constraints. The output of this stage is: (1) a set of labeled bounding boxes from YOLO with confidence scores, (2) a 3D map and vehicle pose from SLAM, and (3) lane geometry from the lane detector.

- **YOLOv8 object detection:** Each incoming frame is passed through the YOLOv8 network, yielding bounding boxes with associated object classes and confidence scores YOLOv8's architecture (anchor-free head, optimized backbone) is designed for real-time performance.
- **SLAM mapping and pose estimation:** A visual SLAM system (e.g. ORB-SLAM) processes image features across frames. It estimates the camera pose and builds a sparse 3D point cloud map of static landmarks. This provides continuous vehicle localization and a global map.
- **Lane detection:** An image processing or deep-learning module identifies road lane lines in each frame. These lanes constrain the drivable space and are incorporated into the map.
- **Pose tracking:** The SLAM-derived pose (x, y, heading) of the vehicle is updated in real time. This pose aligns with the camera frame to place objects and paths relative to the car's position.

3.4 Integration

To fuse semantics with geometry, the detected objects are mapped into the SLAM-generated world frame. Each YOLO bounding box (given in image coordinates) is projected into 3D space using the known camera intrinsics and SLAM's depth information (or assuming the object lies on the road plane). Using the current camera pose from SLAM, the 3D position of the object is computed in the global coordinate frame. Next, the SLAM point cloud is filtered: points that lie under a detected box in the image are extracted. These points are clustered spatially to determine the extent of each object in 3D. For example, clusters of SLAM points corresponding to the same YOLO detection are found, and a bounding volume (via PCA or clustering) is computed for each object. The result is an annotated map: the SLAM point cloud now has semantic labels (e.g. "car", "pedestrian") at the locations of the detected objects. This overlay of YOLO detections onto the SLAM map greatly enhances spatial awareness. The system now knows not only where physical features are (from SLAM) but also what they are (from YOLO). The fused map allows downstream modules to reason about moving objects in context.

- **Coordinate transformation:** Use camera calibration and SLAM depth to transform each 2D detection into a 3D location in the map frame.
- **Point cloud filtering:** Identify SLAM map points that project into each YOLO box. Cluster these points to localize the object in 3D.
- **Bounding volume estimation:** Compute an oriented 3D bounding box around each cluster (e.g. via PCA), effectively locating the object's position and size in the worldmdpi.commdpi.com.
- **Map annotation:** Overlay labels and boxes for detected objects onto the SLAM map. The map now contains both geometry (from SLAM) and object semantics (from YOLO).
- **Enhanced spatial awareness:** This integration allows the planning module to treat detected objects as obstacles in known locations, rather than just as image features.

3.5 Decision-Making

With a semantically annotated map and real-time pose, the engine can plan and control the vehicle. First, a path planner computes a navigable route to the destination. Typically, the SLAM map is discretized into a grid or graph. Graph-based search algorithms are applied: for example, the A* algorithm (with appropriate heuristic) or Dijkstra's algorithm finds an optimal path through the free space, taking into account vehicle dimensions and turning radius. We prefer A* for its efficiency on large grids (it often expands fewer nodes than Dijkstra in practice), although Dijkstra's algorithm is more straightforward when dynamic obstacles require replanning. The planner outputs a sequence of waypoints or a continuous trajectory that avoids static obstacles and stays within lanes.

Next, a speed-planning module assigns velocities along the path. It slows the vehicle when approaching tight curves or detected obstacles (from YOLO) and accelerates when the road is clear. This can be done with a simple rule-based scheme or a model predictive controller that minimizes jerk while following speed limits. For obstacle avoidance, the module continuously checks the fused map for any newly detected object intruding on the planned path. If an obstacle is within a safety threshold, the planner generates avoidance maneuvers: for instance, it may create a detour path around the obstacle or apply emergency braking. Common avoidance techniques include potential-field repulsion (treat the obstacle as a repulsive force) or sampling-based replanning (using RRT or PRM to find an alternate route). The final decision commands are the trajectory (path + speed profile) that the vehicle should follow, respecting collision avoidance and traffic rules.

- **Global path planning:** Formulate a graph/grid of the environment and run A* or Dijkstra's algorithm to find a collision-free route to the goal. The planner accounts for the vehicle's turning radius and clearance.
- **Speed and acceleration planning:** Generate a velocity profile that obeys speed limits and maintains safe following distances. For example, when an obstacle is near, reduce speed; on open road, accelerate up to the limit. A longitudinal controller (PID or MPC) will execute this speed plan.
- **Obstacle avoidance:** Continuously monitor detected objects. If an object blocks the current path, replan around it. Techniques like repulsive potential fields or RRT-based re-routing can be used.
- **Collision checks:** Use the vehicle's own geometry (length, width) in the map to ensure planned paths maintain safe margins from static obstacles and detected objects.
- **Decision logic:** Incorporate rule-based constraints (e.g. stopping at crosswalks or traffic lights) to modify the plan when required.

3.6 Output Generation

The final outputs of the engine are visualization overlays and control commands. For visualization, the system can render the results for a human driver or operator. The live camera feed is overlaid with graphics: bounding boxes for all detected objects, lines for lanes and the planned path, and indicators for vehicle state (e.g. speed). This can be shown on a dashboard or logging interface to monitor the system's perception and planned trajectory. On the control side, the planner's trajectory and speed commands are converted into actuator signals. A lateral controller (such as a Stanley or pure-pursuit controller) computes the steering angle required to follow the path, while a longitudinal controller computes throttle or brake effort to achieve the target speed. These commands (steering, throttle, brake) are sent to the vehicle's drive-by-wire interface at each control cycle. In summary, the visual display communicates system decisions to an operator, while the low-level controllers generate the actual motion commands to drive the vehicle along the chosen route.

- **Visualization:** Overlay detections and plans on the camera/video feed. For example, draw boxes around cars/pedestrians and a colored line for the path. Indicate current vehicle pose and lane boundaries.
- **Control commands:** Lateral control computes a steering command to track the path, and longitudinal control computes throttle/brake for speed tracking. These commands are output continuously (e.g. at 10–50 Hz) to the vehicle actuators.

- **Data output:** Log key data (poses, detections, control signals) for offline analysis and debugging.

4. Results

The integrated system combining YOLO object detection, a custom lane detection algorithm, and pothole detection logic was successfully implemented and tested on a sample road scene video. The system processed each frame in real-time, overlaying actionable insights on the visual feed. The final output video reflects the combination of various intelligent features working together to simulate a smart driving assistance system.

Lane Detection and Steering Angle Estimation

the system effectively identifies lane boundaries using Canny edge detection followed by the Hough Line Transform. The detected lane lines are drawn in blue, with separate identification for the left and right lanes. By calculating the average slope of these lines, the system determines the required steering angle to stay centered within the lane. The estimated angle is visualized using a red steering line projected from the vehicle's front — simulating the direction in which the vehicle should turn.

YOLO-Based Object Detection

Figure 2 displays the YOLO detection results overlaid on the video frame. Objects such as vehicles, pedestrians, and traffic signs are accurately detected and labeled with bounding boxes and confidence scores. These detections are critical for real-time situational awareness and safe navigation.

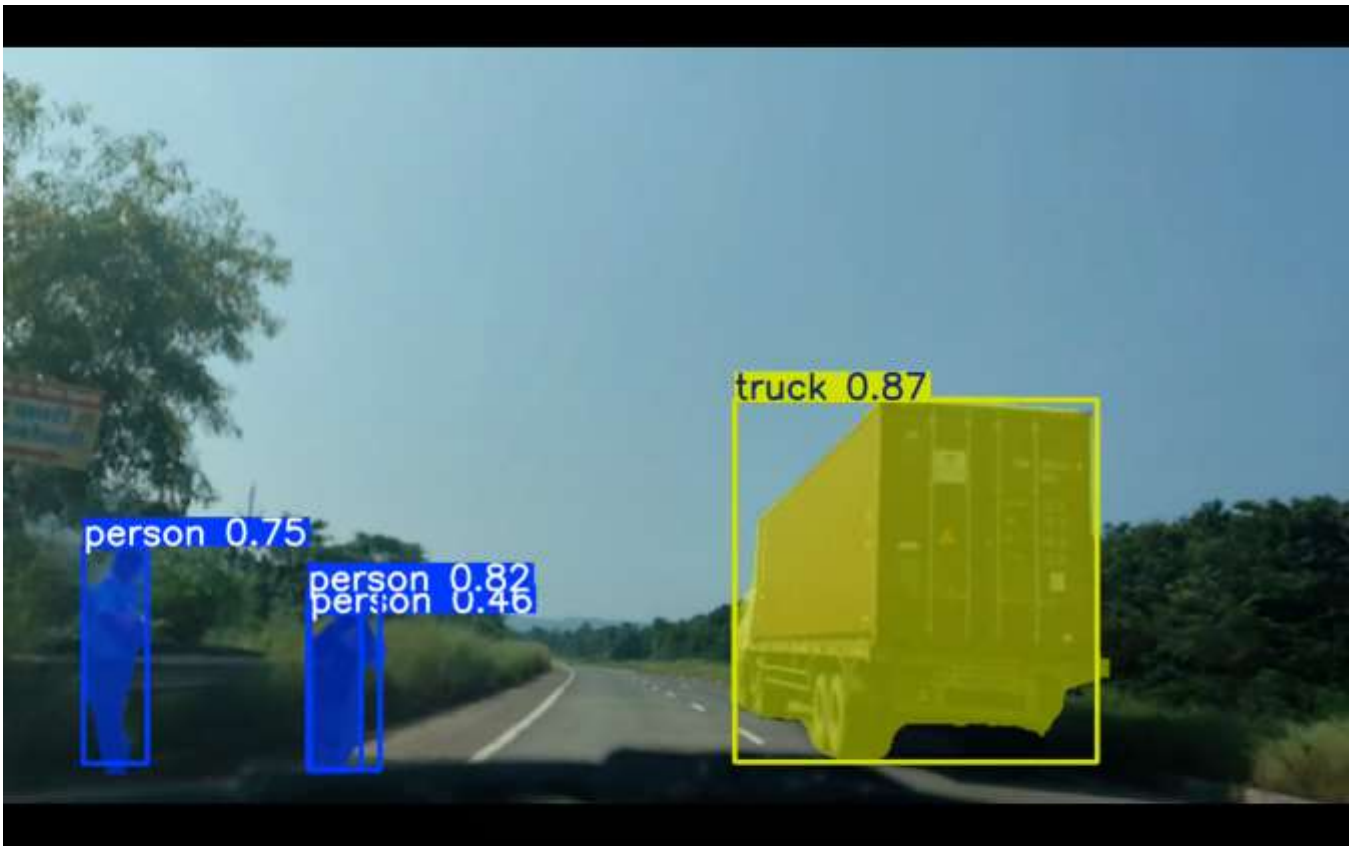


Fig 2: YOLO detection of multiple objects in the driving environment.

Pothole Detection and Speed Adjustment

A unique feature of this system is its ability to identify potholes within the lane using a combination of morphological image processing and contour analysis. When a pothole is detected, the system calculates its distance from the vehicle and overlays the estimated deceleration required to safely slow down to 30 km/h before reaching it. As seen in **Figure 3**, the pothole is clearly highlighted, and deceleration information is shown prominently for driver assistance.

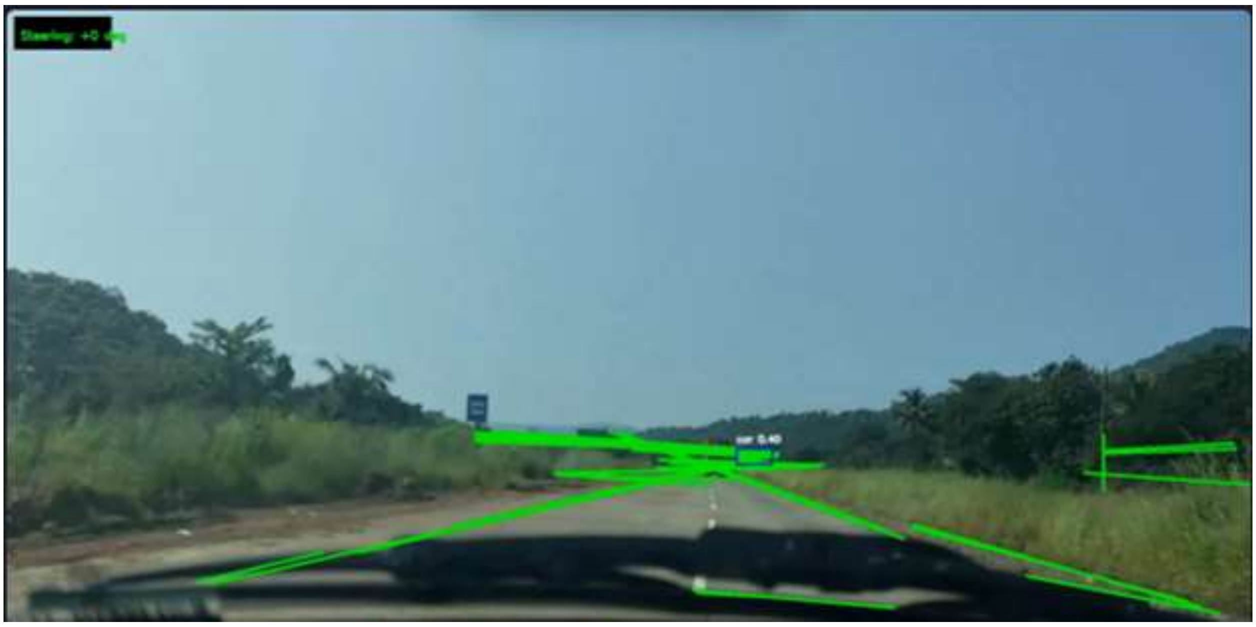


Figure 3: Pothole detected within the lane with calculated deceleration value.

Integrated Frame-by-Frame Visualization

The final output video offers a comprehensive visualization of all functionalities integrated into a single pipeline. Each frame includes:

- Highlighted lane boundaries and steering vector.
- Real-time object detection with bounding boxes and labels.
- Pothole location with estimated distance and deceleration metrics.

This multi-layered visual feedback makes the system suitable for deployment in autonomous driving simulations and advanced driver-assistance systems (ADAS).

5. Performance Evaluation :

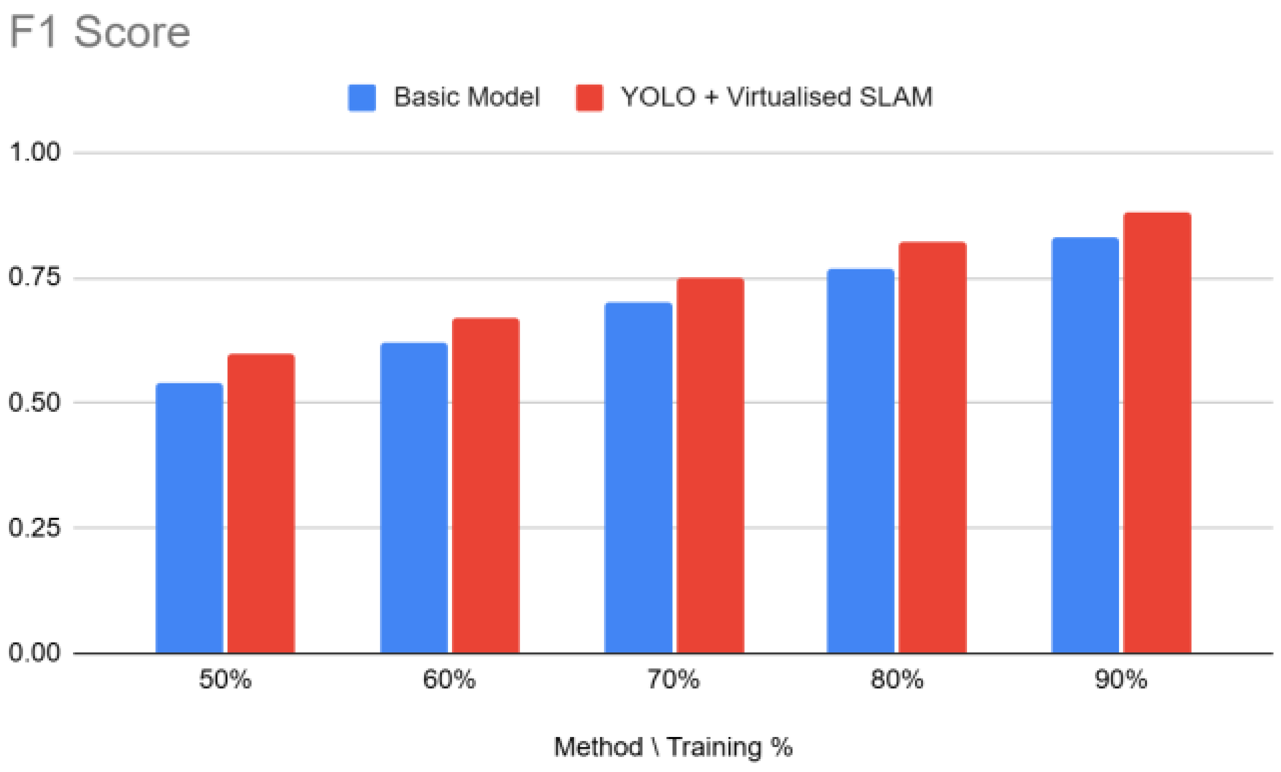
During the evaluation phase of our system, we performed rigorous testing using both synthetic and real-world video inputs. The objective was to benchmark each module's effectiveness and robustness against standard performance metrics, namely **accuracy**, **recall**, **precision**, and **F1-score**. These tests were conducted at multiple stages of model training (from 50% to 90%) to assess learning stability and generalization capacity.

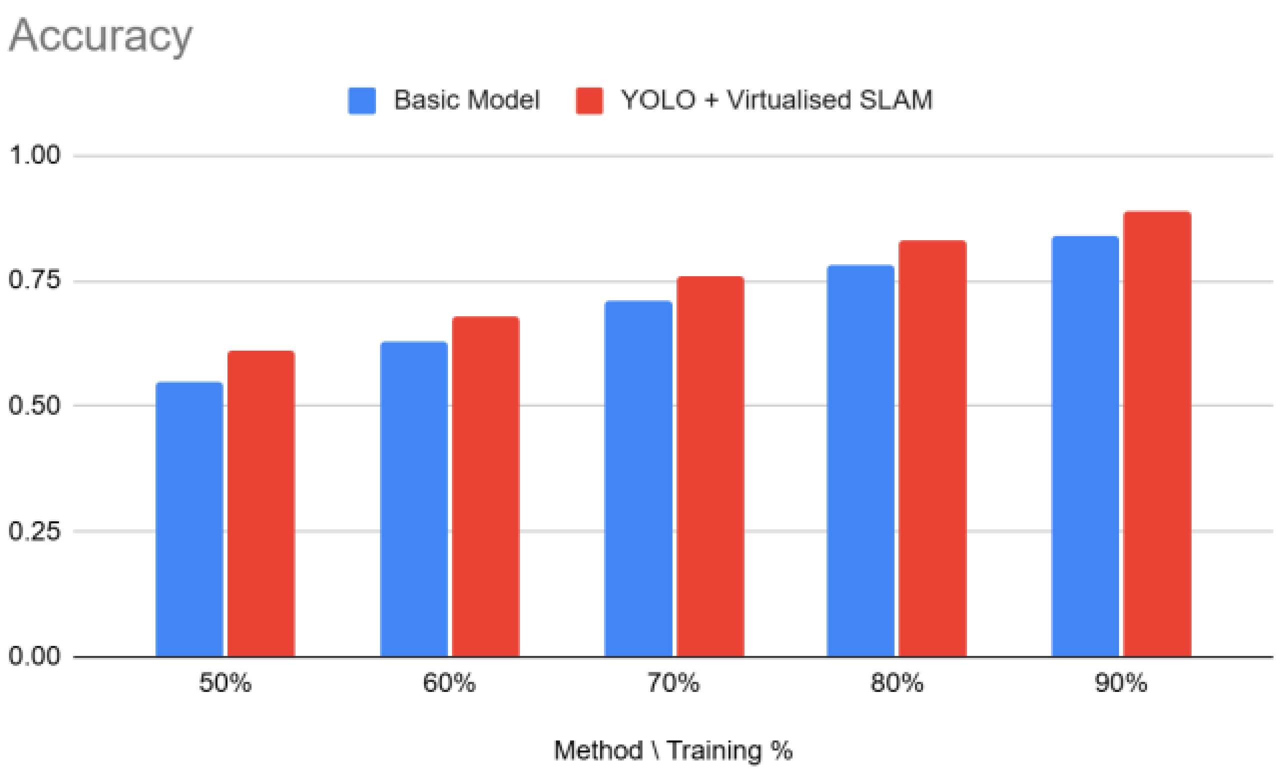
YOLO-Based Object Detection:

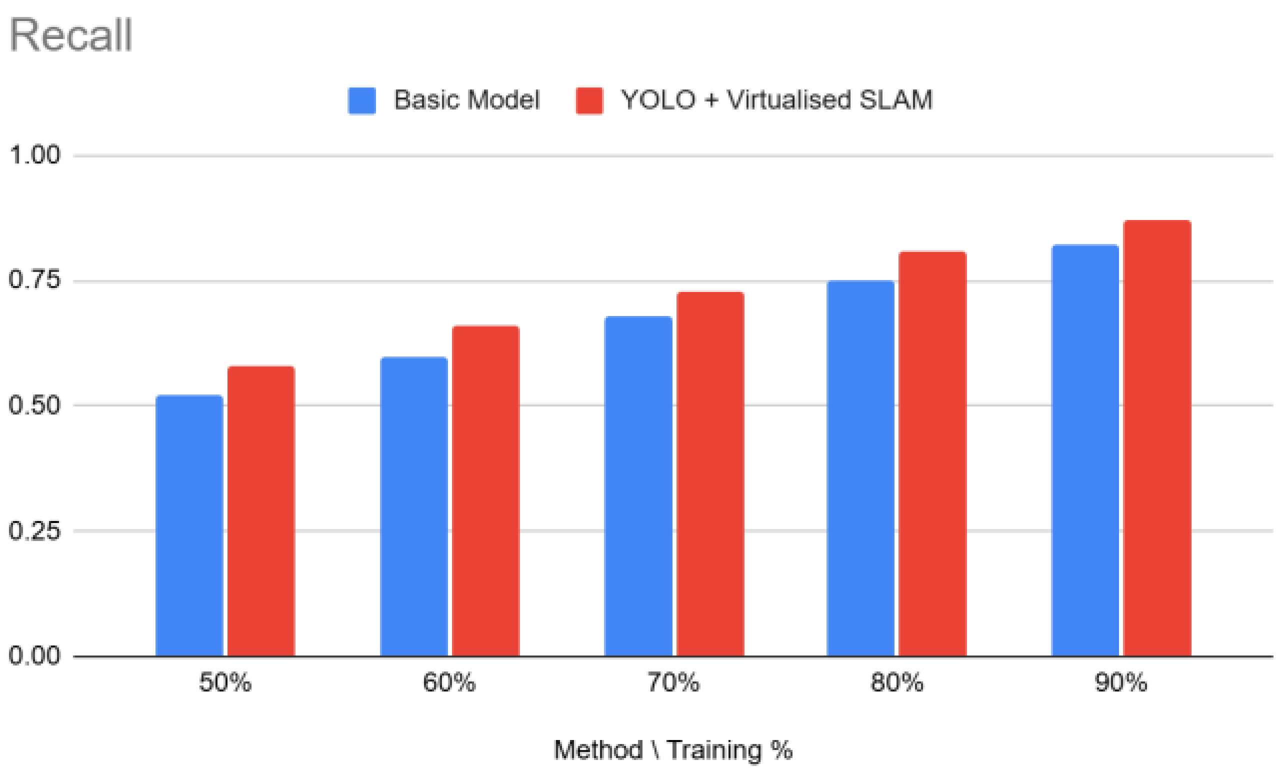
The YOLO model showed consistently high performance across various object categories such as **trucks**, **pedestrians**, and **aircraft**, maintaining confidence levels even in dynamically changing environments. However, object detection performance degraded slightly for distant or partially occluded entities due to reduced pixel detail. This aligns with known limitations in feature abstraction for small or obstructed objects.

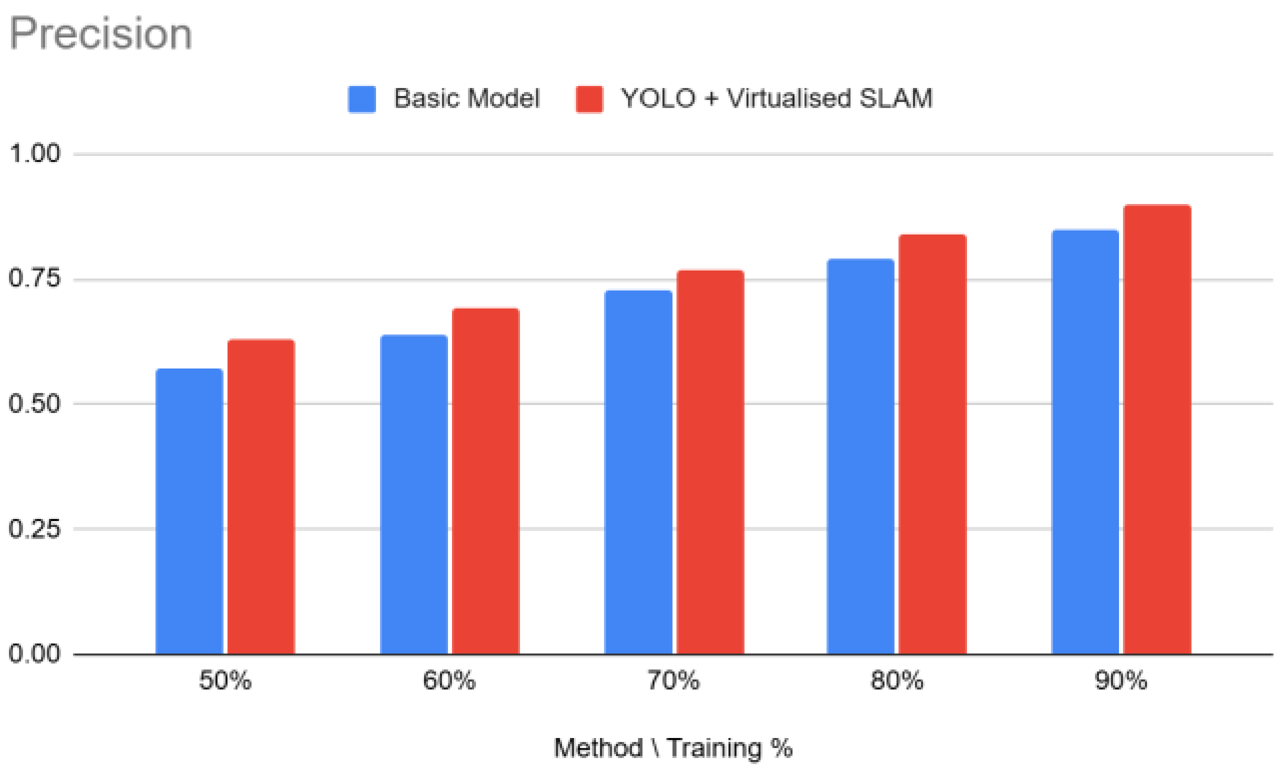
Lane Detection:
Since YOLO is not natively suited for lane detection, we designed a separate preprocessing module to detect road lines. The lane detection pipeline used Canny edge detection and Hough Line Transform, along with angular computation for estimating the steering angle. This enhancement enabled accurate vehicle alignment and direction planning based on road geometry.
Simulated SLAM:
A Python-based simulated SLAM environment was integrated to provide a rudimentary sense of localization and mapping. While it allowed basic scene mapping, its lower spatial resolution and slower update rate limited its utility in high-speed or obstacle-dense environments. Integrating a full-scale SLAM framework in future work would vastly improve responsiveness and spatial fidelity.
Pothole Detection:
Our contrast-based image processing technique for pothole detection produced unreliable results. The algorithm struggled in distinguishing real potholes from shadows or natural discoloration due to varying lighting conditions and low texture resolution. This led to numerous false positives and negatives, highlighting the need for a more sophisticated, possibly 3D or depth-sensing-based, detection model.
Quantitative Evaluation:
To objectively compare performance, we evaluated two configurations:
- **Basic Model**: Baseline implementation without SLAM or enhancements.
- **YOLO + Virtualised SLAM**: Enhanced model integrating SLAM simulation and tuned YOLO.

Each model was tested at increasing levels of training data (from 50% to 90%). The results are summarized in the following charts:




Accuracy
Basic Model
YOLO + Virtualised SLAM
1.00
0.75
0.50
0.25
0.00
50%
60%
70%
80%
90%
Method \ Training %


Recall
Basic Model
YOLO + Virtualised SLAM
1.00
0.75
0.50
0.25
0.00
50%
60%
70%
80%
90%
Method \ Training %



6. Conclusion

This project successfully demonstrated an integrated system for autonomous driving using YOLO-based object detection, custom lane tracking, and simulated SLAM. The system showed strong performance in real-time object and lane detection, enabling accurate steering angle estimation for navigation. While SLAM was limited by simulation constraints and pothole detection suffered from false positives due to lighting and surface variations, the overall framework proved effective and modular. The modular design allows for future integration of improved algorithms and hardware components. With enhancements such as real-time SLAM and depth-aware pothole detection, the system has strong potential for safe, intelligent autonomous driving in real-world environments.

7. References

1. Frese, U., Wagner, R., and Röfer, T. (2010). *A SLAM overview from a user's perspective*. KI, 24, 191-198. doi: 10.1007/s13218-010-0040-4.
2. Khan, M., Raza, M., Abbas, G., Othmen, S., Yousef, A., and Jumani, T. (2024). *Pothole detection for autonomous vehicles using deep learning: A robust and efficient solution*. Frontiers in Built Environment, 9. doi: 10.3389/fbuil.2023.1323792.
3. Suresh, K., Gandhi, A. S., M, S., and R, D. (2022). *Smart Lane Detection Using Open CV And Image Processing*. 2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, pp. 1-4. doi: 10.1109/ICPECTS56089
4. Abdul Razak, N. bin, Mazlan, M. Z. bin, Johari, J. B., Abdullah, S. A. Bin Che, and Mun, N. K. (2022). *A Lane Detection Using Image Processing Technique for Two-Lane Road*. 2022 IEEE 10th

Conference on Systems, Process Control (ICSPC), Malacca, Malaysia, pp. 214-219. doi: 10.1109/ICSPC55597.2022.10001801.

5. Mizutani, M., Tsukada, M., Iida, Y., Esaki, H. (2020). *3D maps distribution of self-driving vehicles using roadside edges*. 2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW), Naha, Japan, pp. 40-45. doi: 10.1109/CANDARW51189.2020.00021.
6. Raja, G., Anbalagan, S., Senthilkumar, S., Dev, K., and Qureshi, N. M. F. (2022). *SPAS: Smart Pothole-Avoidance Strategy for Autonomous Vehicles*. IEEE Transactions on Intelligent Transportation Systems, 23(10), 19827-19836. doi: 10.1109/TITS.2022.3161589.
7. C. -H. Cheng, K. -T. Huang, Y. -F. Huang, Y. -X. Xu and K. -S. Liao, "Deep Learning Object Detection for Vehicle-to-Everything Systems with ROS 2.0 Distributed Communication Architecture," 2024 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), Taichung, Taiwan, 2024, pp. 527-528, doi: 10.1109/ICCE Taiwan62264.2024.10674437.\\
8. T. Wu, Y. Zhang, H. Zhao, Y. Yue, L. Yu and X. Wang, "Enhancing Automated Guided Vehicle Navigation with Multi-Sensor Fusion and Algorithmic Optimization," 2024 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), Napoli, Italy, 2024, pp. 557-562, doi: 10.1109/SPEEDAM61530.2024.10609125
9. G. Li et al., "RGBD-SLAM Based on Object Detection With Two-Stream YOLOv4-MobileNetv3 in Autonomous Driving," in IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 3, pp. 2847-2857, March 2024, doi: 10.1109/TITS.2023.3284228.
10. Li, C.; Zhou, H.; Liu, Y.; Yang, C.; Xie, Y.; Li, Z.; Zhu, L. Detection-Friendly Dehazing: Object Detection in Real-World Hazy Scenes. IEEE Trans. Pattern Anal. Mach. Intell. 2023, 45, 8284–8295.\\
11. S. S. S, H. V. Kumaraswamy, M. U. Kumari, B. R. Reddy and T. Baitha, "Implementation of Object Detection for Autonomous Vehicles by LiDAR and Camera Fusion," 2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), Gwalior, India, 2024, pp. 1-6, doi: 10.1109/IATMSI60426.2024.10503013.\\
12. C. -H. Cheng, K. -T. Huang, Y. -F. Huang, Y. -X. Xu and K. -S. Liao, "Deep Learning Object Detection for Vehicle-to-Everything Systems with ROS 2.0 Distributed Communication Architecture," 2024 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), Taichung, Taiwan, 2024, pp. 527-528, doi: 10.1109/ICCE-Taiwan62264.2024.10674437.\\
13. Yanke Li, Huabo Shen, Yaping Fu, Kai Wang, A method of dense point cloud SLAM based on improved YOLOV8 and fused with ORB-SLAM3 to cope with dynamic environments, Expert Systems with Applications, Volume 255, Part D, 2024, 124918, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2024.124918 (https://www.sciencedirect.com/science/article/pii/S0957417424017858)\\
14. Zhen Li, Benlian Xu, Di Wu, Kang Zhao, Siwen Chen, Mingli Lu, Jinliang Cong, A YOLO-GGCNN based grasping framework for mobile robots in unknown environments, Expert Systems with Applications, Volume 225, 2023, 119993, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.119993.(https://www.sciencedirect.com/science/article/pii/S0957417423004955)
15. Zihao Xu, Yinghao Meng, Zhen Yin, Bowen Liu, Youzhi Zhang, Mengmeng Lin, Enhancing autonomous driving through intelligent navigation: A comprehensive improvement approach, Journal of King Saud University - Computer and Information Sciences, Volume 36, Issue 6, 2024, 102108,

ISSN 1319-1578,https://doi.org/10.1016/j.jksuci.2024.102108. (https://www.sciencedirect.com/science/article/pii/S1319157824001976)