10.48047/jocaaa.2024.33.08.275

A Novel Adaptive Tracking Approach with Integrated Object Detection and Segmentation in Dynamic Settings

Guru Prasad M. Bhat
Visvesvaraya Technological University, Belagavi 590018, India
Department of Information Science Engineering
R V College of Engineering
Bangalore, Karnataka, India-560059
guruprasadmb.phdis@rvce.edu.in

Nagaraj G. Cholli
Department of Information Science Engineering
R V College of Engineering
Bangalore, India-560059
nagaraj.cholli@rvce.edu.in
https://orcid.org/0000-0001-7409-8272

Abstract : Achieving real-time performance in dynamic environments with moving objects remains a significant challenge. This paper presents a novel approach that integrates Oriented FAST and Rotated BRIEF SLAM (ORB-SLAM3), an advanced Visual Simultaneous Localization and Mapping (V-SLAM) system, with an Artificial Neural Network (ANN) architecture to address this issue. The proposed system leverages ORB features, robust mapping techniques, and loop closure detection to enable real-time camera localization and mapping, operating efficiently on standard computing hardware. This makes it highly suitable for various applications in robotics and augmented reality. The primary contributions of this work focus on three key areas: adaptive feature extraction, lightweight deep-learning-based semantic segmentation, and enhanced scene understanding. The adaptive feature extraction mechanism dynamically detects and adjusts feature point selection based on environmental variations, ensuring efficient tracking. The lightweight deep-learning model improves semantic segmentation, enabling accurate scene interpretation even in complex and dynamic environments. Experimental results demonstrate that the proposed ANN-enhanced ORB-SLAM system outperforms conventional methods. It achieves notable improvements in real-time performance and localization accuracy in dynamic scenarios, with detection rates of 90% for persons and 85.7% for scissors. Precision and recall metrics indicate substantial gains, with object detection achieving an accuracy of 99.2% and an F1 score of 98.96%. The proposed architecture represents a significant step forward in adaptive perception and navigation systems, opening new avenues for future research and development in this field.

Keywords: V-SLAM, ORB-SLAM, YOLOv3, YOLOv8n, ANN.

Introduction

V-SLAM [1] is a vital technique in the field of robotics, computer vision, and autonomous systems. It enables a device, such as a robot or a camera-equipped vehicle, to create a map of an unfamiliar environment while simultaneously determining its position within that map. This dual capability is significant for navigating and interacting with dynamic and unpredictable environments.

V-SLAM has made significant advancements, it still faces several limitations when deployed in complex and dynamic environments [2]. V-SLAM systems encounter several challenges when operating in complex and dynamic environments. These challenges include sensitivity to changes in lighting, occlusions, and the presence of dynamic objects, such as pedestrians, which can disrupt feature

10.48047/jocaaa.2024.33.08.275

tracking and lead to localization errors. Additionally, high computational demands and latency can impact real-time processing, especially when handling high-resolution data. V-SLAM systems also struggle with feature extraction in low-texture or repetitive environments and have difficulty adapting to changes in the environment. However, advanced neural networks and machine learning techniques show promise in enhancing the accuracy and robustness of V-SLAM systems [3], [4].

Developing robust solutions for tracking features and maintaining map consistency in dynamic environments with moving objects is essential [5]. Additionally, SLAM algorithms that can perform well across different settings, balancing accuracy with real-time performance on devices with limited resources and ensuring long-term map consistency. These critical areas require further research and the development of a robust and versatile SLAM system capable of navigating complex and unpredictable environments. Hence, in this research work a new method that combines ORB-SLAM3, a cutting-edge V-SLAM system, with an ANN architecture is considered to address a few of the above-mentioned problems.

ORB-SLAM3, known for its versatility and robustness in handling various camera setups, is augmented with an adaptive feature extraction module, lightweight deep-learning semantic segmentation for better scene understanding, and multi-view geometric segmentation for dynamic object management. This integration aims to overcome the challenges posed by dynamic environments, providing a more resilient and precise V-SLAM solution. The proposed approach leverages the strengths of ORB-SLAM3 and advanced neural networks, making significant strides in the field of autonomous navigation and robotic perception [6].

Leveraging ANNs include adaptive feature extraction, semantic segmentation, and multi-view geometric segmentation. ANNs can learn to extract relevant features from images, focusing on important points in dynamic scenes. Semantic segmentation helps improve scene understanding and robustness. Multi-view geometric segmentation predicts object motion across multiple views, improving map consistency and incorporating geometric constraints [7], [8]. This hybrid approach of combining ORB-SLAM3 with deep learning models (ANNs or CNNs) enhance feature detection and handle dynamic or complex environments more effectively.

Materials and Methods

This paper describes how the proposed system's real-time operational structure integrates ORB-SLAM3 with Adaptive Scale CNN (ASCNN), Histogram of Optical Flow with Spatial Gradient (HOF-SG), and Semantic-Enhanced Adaptive Tracking (SEAT) modules. It goes over the computational factors, such as hardware specifications, parallel processing, and optimization techniques, that must be taken into account to guarantee the system runs effectively in real time. Let’s delve into the methods available and why we have chosen the FPN, V-SLAM and explore how the V-SLAM integration of ANNs is performed.

Feature Pyramid Networks (FPN)

The FPN integrated with Faster Region-based Convolutional Neural Networks (R-CNN) system achieves state-of-the-art single-model results on the COCO detection benchmark without additional complexities. Moreover, it operates efficiently on a GPU, making it a precise solution for multi-scale object detection [9]. Enhancing the ORB-SLAM3 using FPN, produces semantically rich feature maps at various scales, integrating the capability to accept both segmented images and original images. The

10.48047/jocaaa.2024.33.08.275

segmentation map must prioritize features according to their semantic class, giving significant weight to stable or critical features—such as walls and roads—while effectively ignoring dynamic features like people or vehicles. Then, by utilizing HOF-SG to effectively compute optical flow assisted by FPN, enabling to uncover the dynamic elements and intricate motion patterns within the scene. This enhances the accuracy of the SLAM system in dynamic environments by incorporating motion information into the map representation, adding a layer for understanding scene dynamics.

YOLO V8 Integration

YOLO is a family of real-time object detection models that perform object detection and localization simultaneously in a single network pass [10]. The YOLO frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. This approach contrasts with other object detection methods that use region proposal networks (RPN) and secondary classifiers.

YOLO object detection involves specific equations, although these equations can vary depending on the implementation and architecture details. Here are some critical equations associated with YOLO-style object detection, where predicted bounding box coordinates are typically represented as(x, y, w, h) where (x, y) are the box's center coordinates and $(w, h)$are the width and height of the box. Object Confidence can be identified by each predicted bounding box that has an associated confidence score Confidence$i$, indicating how likely it is that the box contains an object. Then for class scores for each bounding box, YOLO predicts class probabilities $classi$ for each class i in the dataset. The overall loss function L used in YOLO combines localization loss, confidence loss, and classification loss. It is typically defined as:

$$L_{local} = \sum_{i=0}^{s^2} \sum_{j=0}^{k} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + (\sqrt{h}_i - \sqrt{\hat{h}}_i)^2] \quad \text{Eq.1}$$

where L_{loc} represents Localization loss measures the error in the predicted bounding box coordinates (centre, width, and height) compared to the ground truth. YOLO predicts bounding boxes for each grid cell, and localization loss is used to ensure these predictions are as accurate as possible.

$$L_{conf} = \sum_{i=0}^{s^2} \sum_{j=0}^{k} 1_{ij}^{obj} (c_i - \hat{c}_i)^2 + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{k} 1_{ij}^{noobj} (c_i - \hat{c}_i)^2 \quad \text{Eq. 2}$$

Confidence loss L_{conf} measures how well the predicted confidence score of the bounding box matches the ground truth. This score reflects the model's confidence that a bounding box contains an object and how accurate it believes the box is.

$$L_{class} = \sum_{i=0}^{s^2} 1_{ij}^{obj} \sum_{c \epsilon classes} \hat{p}_i(c) - \hat{p}_i(c)^2 \quad \text{Eq.3}$$

Classification loss L_{class} measures the error in the predicted class probabilities for the object within the bounding box. This helps ensure that the predicted class labels match the ground truth labels.

where:
L represents Loss function.
1_{ij}^{obj} is an indicator function that is 1 if an object appears in cell i and anchor j, 0 otherwise.
$\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i$ are predicted bounding box parameters.

10.48047/jocaaa.2024.33.08.275

$\hat{C}_i, \hat{p}_i(c)$ are predicted confidence scores and class probabilities.
λ_{obj} are weighting coefficients.
S is the number of grid cells along one dimension of the grid.
B represents the number of bounding boxes per grid cell.
The overall YOLO loss function is a weighted sum of these three loss components:

$$L = \lambda_{coord} L_{loc} + \lambda_{class} L_{class} + \lambda_{conf} L_{conf} \quad \text{Eq.4}$$

where:
$\lambda_{coord}, \lambda_{class}, \lambda_{conf}$ are weighting factors for the respective losses.

This loss function helps in training the YOLO model to accurately detect objects by penalising inaccuracies in the bounding box coordinates, confidence scores, and class probabilities. The expressions in the Loss function are used during YOLO model training to adjust the network weights such that the predicted bounding boxes align closely with ground truth annotations. The detected and segmented objects in real-time are the inputs for SEAT and HOF-SG

Adaptive-Scale Convolutional Neural Networks (ASCNet)

For tasks such as semantic segmentation, extracting multi-scale information is essential. Achieving multi-scale information extraction presents issues for CNNs. The computational cost increases when convolutional kernels are expanded. Image information is deteriorated when maximum pooling is used. Fixed dilation rates apply to dilated convolutions, which solve certain problems. In order to provide the best receptive fields for varying object sizes, ASCNet seeks to adaptively learn dilation rates for each pixel.

In ASCNet Architecture during end-to-end training, ASCNet introduces a three-layer convolution structure. It determines the proper dilution rate for every pixel in the picture. Information extraction at the appropriate scale is guaranteed by pixel-level dilation rates. In segmentation tasks, ASCNet performs better than dilated CNNs and traditional CNNs. ASCNet adaptation helps improve the network's ability to capture features at multiple scales, which is crucial for tasks like object detection and image classification. Instead of using fixed-size filters throughout the network, ASCNet adjusts the receptive field dynamically. This adaptation is typically achieved through mechanisms such as:

1. Multi-Scale Feature Extraction: ASCNet incorporates convolutional layers with filters of different sizes to capture features at various scales within the image.
2. Adaptive Scaling: The receptive field size of each convolutional filter is adjusted based on the scale of features present in different regions of the image. This adaptive scaling is often influenced by contextual information or global image features.
3. Dynamic Pooling Mechanisms: ASCNet may utilise adaptive pooling techniques or mechanisms that adjust pooling sizes based on the spatial characteristics of the input data.

Here is the conceptual outline of how adaptive scaling can be represented:

Convolution Operation with Adaptive Scaling:

10.48047/jocaaa.2024.33.08.275

Let's denote the input feature map at a given layer X, and the filter weights of a convolutional layer as W. The output feature map Y can be computed as:

$$Y_{i,j} = \sum_{m=0}^{i=1} \sum_{n=0}^{j=1} X_{i+m+j+n} \cdot W_{m,n} \quad \text{Eq.5}$$

In ASCNet, the filter weights $W_{m,n}$ may vary in size and shape dynamically depending on the adaptive scaling mechanism.

Adaptive Scaling Mechanism:

Suppose $R_{i,j}$ denotes the receptive field size or scaling factor at position (i, j) within the network. This scaling factor could be influenced by parameters learned during training or computed based on image characteristics. The receptive field size $R_{i,j}$ could be used to adjust the convolutional filter $W_{m,n}$ dynamically. For example, the filter weights could be scaled or adjusted based on$R_{i,j}$, affecting how much context or detail the filter captures.

Dynamic Pooling:

Adaptive-Scale Convolutional Neural Networks might also incorporate dynamic pooling mechanisms where the pooling size varies depending on the context. For example, adaptive average pooling might adjust its kernel size based on the spatial dimensions of the feature map or the scale of objects detected. In practical terms, ASCNet could adaptively adjust filter sizes in a convolutional layer based on the scale of objects in the input image. For instance, if an image contains both small and large objects, ASCNet might use smaller filters to capture details of small objects and larger filters to capture contextual information for large objects.

Hybrid Optical Flow with Semantic Guidance (HOFSG)

HOFSG combines traditional optical flow estimation with semantic segmentation to improve the accuracy and robustness of motion estimation in computer vision tasks.
Traditional optical flow methods estimate the motion of pixels between consecutive frames based on intensity changes. However, these methods can struggle with complex scenes, occlusions, or regions with homogeneous textures. Semantic segmentation, on the other hand, can provide valuable contextual information about the scene, such as object boundaries and categories. By integrating semantic segmentation with optical flow estimation, HOFSG aims to leverage both low-level motion information and high-level semantic context for more accurate and reliable motion estimation.

1. Optical flow estimation involves calculating the displacement vectors (u, v) for each pixel between two consecutive frames. The objective is to minimise the difference between the observed intensity and the predicted intensity under the flow field constraints. The equation for optical flow can be represented as:

$$I(x + u, y + v, t + 1) = I(x, y, t) \quad \text{Eq.6}$$

where u, v are the components of the optical flow vector at pixel x, y between frame t and frame t+1.

2. Semantic segmentation assigns a class label to each pixel in an image, indicating the category of the object it belongs to (e.g., person, car, road). This segmentation provides contextual information that can help in identifying object boundaries and influencing motion estimation.

10.48047/jocaaa.2024.33.08.275

3. In HOFSG, semantic segmentation maps can guide the optical flow estimation process by:
- Region-based Constraints: Using semantic labels to constrain or guide the flow estimation within regions corresponding to different objects or scene elements.
- Improved Flow Consistency: Using semantic labels to enforce consistency in the flow field across object boundaries or within homogeneous regions.

To integrate semantic guidance into optical flow estimation, the formulation can be adjusted to incorporate semantic information into the objective function:

Let $I_t(x, y)$ denote the intensity of pixel (x, y) in frame t.
- Energy Minimization Formulation: The optical flow estimation with semantic guidance can be formulated as an energy minimization problem:

$$E(u, v) = \sum_{(x,y)}[D(I_t(x, y), I_{t+1}(x + u, y + v)) + \lambda \cdot \phi(S(x, y), u, v)] \quad \text{Eq.7}$$

where:
- D is a data term measuring the intensity difference between frames.
- ϕ is a smoothness term that ensures spatial and temporal consistency of the flow field.
- $S(x, y)$ represents the semantic label (e.g., object category) at pixel (x, y).
- λ is a weighting parameter that balances the influence of the semantic guidance term.

Semantic Guidance Term: The semantic guidance term $\phi(S(x, y), u, v)$ can be designed to enforce flow consistency within semantic regions.

$$\phi(S(x, y), u, v) = \begin{cases} D_{\infty}^{o} & if\ S(x,y) = S(x+u,y+v) \\ & otherwise \end{cases} \quad \text{Eq.8}$$

This term ensures that the flow vectors (u, v) respect semantic boundaries, thereby improving the coherence of the estimated motion fields. HOFSG enhances traditional optical flow estimation by integrating semantic segmentation information. This approach provides more robust motion estimation, especially in complex scenes with varying object scales, occlusions, and scene dynamics. The equations and concepts outlined here illustrate how semantic guidance incorporated into the optical flow estimation process to improve accuracy and reliability in computer vision applications [11].

Semantic-Enhanced Adaptive Tracking (SEAT)

ASACTT is a novel Siamese network-based object tracking tracker that focuses on target-specific details, and maintains accuracy while minimising complexity. It achieves an outstanding success score of 99.2% on the GOT-10K dataset at 36 FPS. Other semantic tracking approaches include unified convolutional frameworks and domain-adaptive methods, which enhance accuracy and adaptability in challenging scenarios.

ORB-SLAM3

ORB-SLAM3 is a state-of-the-art visual SLAM system that uses ORB features, a binary feature descriptor suitable for real-time applications, and operates on a keyframe-based approach. It maintains a map of the environment using keyframes and sparse 3D points, which is continuously updated as the system moves through the environment. This system tracks camera pose relative to the mapped environment in real-time. ORB-SLAM3 supports multi-map capabilities and can incorporate semantic information for richer representations [12].

10.48047/jocaaa.2024.33.08.275

1. Feature Extraction and Matching:

- ORB-SLAM3 features are robust binary descriptors used for detecting and describing key points in images. Hence, correspondences between key points across frames are established to track the camera motion and build a map [13] .

2. Mapping and Localization:

- ORB-SLAM3 selects keyframes based on criteria like motion, scene structure, or time intervals. This system maintains a map consisting of keyframes and sparse 3D points, continuously updating and refining it as the camera explores the environment.

3. Loop Closure Detection:

- ORB-SLAM3 detects previously visited places by recognizing similar visual features. This helps correct drift and improve the consistency of the map. After loop closure, a global optimization step adjusts the entire map to minimise errors and improve accuracy. ORB-SLAM3 uses several equations to estimate camera pose, update the map, and optimise the entire SLAM system.

4. Camera Pose Estimation:

- Camera motion estimation between consecutive frames involves computing the transformation matrix T_{wc} that relates the camera's position and orientation from frame t to frame $t+1$.
- Given corresponding key points P_t in frame t andP_{t+1} in frame $t+1$, the essential equation is:

$$P_{t+1} = K.T_{wc}K^{-1}P_t \quad \text{Eq.9}$$

where K is the camera intrinsic matrix.

5. Map Update:

- When a new keyframe is selected, ORB-SLAM3 triangulates new 3D points P based on matched key points between the new keyframe and previous keyframes:

$$P = Triangulate\,(P_{prev}, P_{curr}) \quad \text{Eq.10}$$

where P_{prev}, P_{curr} are corresponding key points in the previous and current keyframes.

6. Loop Closure and Optimization:

- Loop Closure Constraint: After detecting loop closures, the system imposes constraints to ensure consistency across the map:

$$e_{loop} = T_{loop} \cdot T_{current}^{-1} \quad \text{Eq.11}$$

where T_{loop} is the transformation estimated from loop closure and$T_{current}$ is the current estimate of camera pose.

- Bundle Adjustment: A global optimization step adjusts all camera poses and 3D points in the map to minimise reprojection errors:

$$\min_{\{T,P\}} \sum_{keyframes} \sum_{visible\ points} \| P_{observed} - KTP \|^2 \quad \text{Eq.12}$$

where $P_{observed}$ are observed keypoints and T, P are optimised camera poses and 3D points.

The equations provided illustrate its core functionalities in estimating camera pose, updating the map, detecting loop closures, and optimising the entire SLAM system to maintain accuracy and consistency in challenging environments.

Experimental Setup

10.48047/jocaaa.2024.33.08.275

Initially, enhance the ORB-SLAM3 using FPN that produces semantically rich feature maps at various scales integrating the capability to accept both segmented images and original images. The segmentation map must prioritize features according to their semantic class, giving significant weight to stable or critical features—such as walls and roads—while effectively ignoring dynamic features like people or vehicles. Then, utilize Histogram of Optical Flow with Spatial Gradient (HOF-SG) to effectively compute optical flow assisted by FPN, enabling you to uncover the dynamic elements and intricate motion patterns within the scene. This enhances the accuracy of the SLAM system in dynamic environments by incorporating motion information into the map representation, adding a layer for understanding scene dynamics. Later, enhance the map produced by ORB-SLAM3 with semantic labels from SEAT, making it not only geometric but also rich in semantics.

The setup comprises detailed simulations and real-world testing environments utilizing the COCO dataset [14]. This dataset offers diverse urban driving scenes, including dynamic objects such as vehicles, animals, and pedestrians, making it ideal for our needs.

Testing Environment: The experiments use a standardized simulation platform replicating the COCO environment. This ensures that the testing conditions remain consistent across all experiments, allowing for reliable comparisons of performance metrics.

Data Handling: The data is pre-processed for each test, ensuring that it is appropriately formatted and normalized for input into the proposed system. The dataset is split into training, validation, and testing sets. The testing set is exclusively used to evaluate the final model performance and prevent data leakage..

Parameter Settings: Detailed parameter settings used in the experiments are provided to maintain transparency and reproducibility. These settings include configurations for the neural networks, simulation parameters, and hardware specifications.

Table – 1: Experimental parameters

Parameter	Description	Value/Setting
Neural Network Framework	Software used for implementing neural networks	PyTorch
Hardware	Hardware specifications for running the experiments	GPU: NVIDIA RTX 3050 Ti
Batch Size	Number of training examples per batch	32
Learning Rate	Initial learning rate for the optimizer	0.001
Epochs	Number of full training cycles	20
Optimizer	Algorithm for optimizing network weights	Adam
Loss Function	Criteria for measuring model performance	Cross-Entropy ASCNN, MSE HOF-SG
Input Size	Dimension of input images	640 x 640 pixels
Testing Frequency	Frequency of model evaluation during training	After every epoch

Results and Discussion - Object Detection

Table 2 presents an analysis of the proposed system against baseline models [15]. The proposed techniques, ASCNN, HOFSG, and SEAT, were assessed against traditional object detection and

10.48047/jocaaa.2024.33.08.275

segmentation methods such as Fast R-CNN, YOLOv3, and YOLOv8n [16]. Fast R-CNN demonstrated moderate performance, achieving an accuracy of 85.00% and an F1 score of 83.50%. However, it had a higher root mean square error (RMSE) of 1.5000, indicating more localization errors. YOLOv3 showed improvement, with an accuracy of 87.00% and an F1 score of 86.80%, along with a lower RMSE of 1.2000. This suggests better localization and segmentation capabilities. YOLOv8n further enhanced performance, attaining an accuracy of 92.00% and an F1 score of 93.50%, supported by a reduced RMSE of 0.9000, which reflects its refined detection and segmentation abilities.

Among the proposed methods, ASCNN achieved the highest accuracy at 99.2% while maintaining a minimal RMSE of 0.6653, indicating superior precision in object detection and segmentation. HOFSG also outperformed the traditional methods, achieving 97.62% accuracy and an RMSE of 0.7593. SEAT matched ASCNN in accuracy at 99.2% but had a slightly higher RMSE of 0.6875.

Table – 2: Comparison results of baseline models with the proposed techniques

Method	Accuracy (%)	Precision	Recall	F1 Score	IoU	RMSE
Fast R-CNN	85.00	0.8300	0.8600	83.50%	0.8200	1.5000
YOLOv3	87.00	0.8600	0.8900	86.80%	0.8600	1.2000
YOLOv8n	92.00	0.9200	0.9500	93.50%	0.9100	0.9000
ASCNN	99.2	0.9905	0.9928	98.96%	0.9923	0.6653
HOFSG	97.62	0.9692	0.9761	96.41%	0.9751	0.7593
SEAT	99.2	0.9798	0.9828	98.98%	0.9828	0.6875

The table 3 provides the comparison results for few classes where we can conclude that the proposed methods show significant improvements over the traditional baselines. For example, the person class, both precision and recall have improved slightly, leading to better mAP scores. The airplane class shows a significant improvement in mAP50 and mAP50-95. Other classes like car and bicycle have smaller or negligible changes. This table demonstrates their effectiveness in enhancing the accuracy of object detection and segmentation with fewer localization errors.

Table – 3: Comparison results of baseline models with the proposed class components

Class	Baseline Precision	Current Model Precision	Baseline Recall	Current Model Recall	Baseline mAP50	Current Model mAP50	Baseline mAP50-95	Current Model mAP50-95
All	0.63	0.63	0.45	0.48	0.50	0.52	0.35	0.37
person	0.75	0.75	0.65	0.67	0.70	0.74	0.50	0.51
bicycle	0.64	0.64	0.50	0.51	0.55	0.56	0.35	0.36
car	0.68	0.68	0.38	0.39	0.45	0.46	0.26	0.26
motorcycle	0.65	0.65	0.70	0.70	0.72	0.73	0.59	0.59
airplane	0.69	0.71	0.42	0.39	0.70	0.78	0.26	0.46

10.48047/jocaaa.2024.33.08.275

Detection and Segmentation Performance

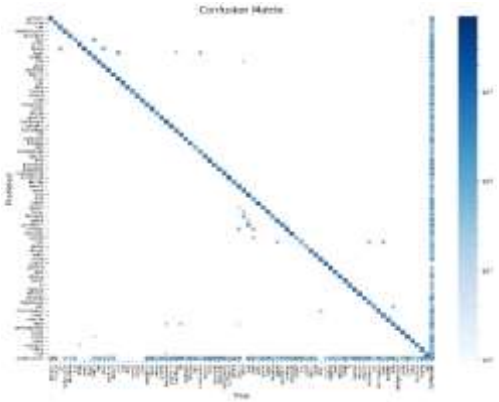

Figure 1: Confusion Matrix

Confusion Matrix: Figure 1 provides the model's predictions across all classes. The x-axis represents the true labels and the y-axis represents the predicted labels. The intensity of color along the diagonal shows that the majority of the predictions are correct, as it indicates higher values. The lighter colors off the diagonal indicate fewer incorrect predictions in this matrix. The color bar on the right indicates the range of values, with darker blue representing higher counts. From this, you can see that some classes have much higher counts than others (e.g., the "person" class). There is a clear imbalance in the dataset, as evidenced by a much darker color for the "person" class. This suggests the model encountered significantly more instances of this class compared to others, like "toothbrush" or "scissors." This confusion matrix suggests that the model performs relatively well, with most of the errors being minor as indicated by the light colors for off-diagonal cells. However, certain classes, especially "person," are heavily represented, while others may have much fewer instances.

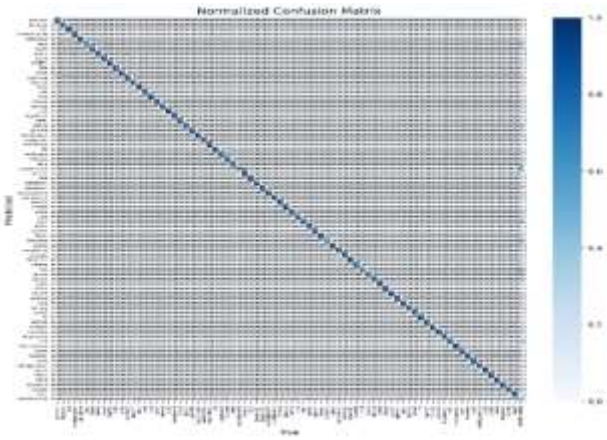

Figure 2: Normalized Confusion Matrix

Normalized Confusion Matrix: Figure 2 represents the true label is “person” and the predicted label is “person,” the diagonal cell for “person” will show the proportion of correct predictions out of all actual “person” instances. The color bar now ranges from 0.0 to 1.0, where 1.0 means 100% accuracy for that specific class. Darker cells along the diagonal mean that the model predicted a higher proportion of those classes correctly. A dark blue color indicates a high percentage of correct predictions for that class. The closer the value is to 1.0, the better the model’s performance on that class. The matrix indicates that the model has a high accuracy for the `Person` class, with an accuracy rate of 0.90. The accuracy for other classes varies, with `Car` at 0.81. However, the ` toothbrush ` class shows a lower accuracy of 0.55, indicating potential challenges in identifying smaller or less distinct objects.

10.48047/jocaaa.2024.33.08.275

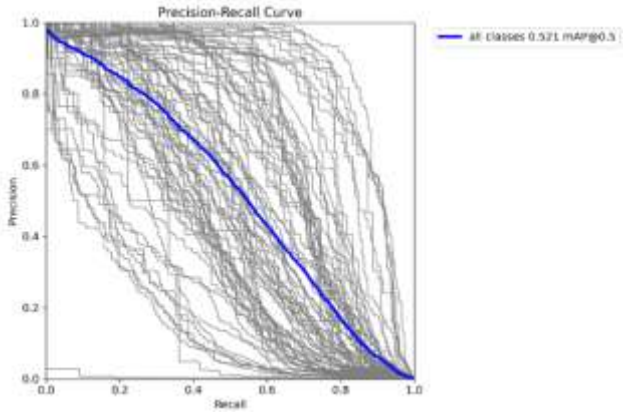

Figure 3: Precision-Recall Curve

Precision-Recall Curve: Figure 3 highlights that the model achieves a mean Average Precision (mAP) of 0.53 at an IoU threshold of 0.5. The `Person` class achieved the highest precision at 0.97, followed by `Bicycle` at 0.95, indicating robust detection capabilities for these classes. The `other` class, as expected, has the lowest precision at 0.2 due to its nature as a non-object class. The individual class performances (gray lines) vary, but the average performance (blue line) provides a good general overview of how well the model detects objects across multiple classes.

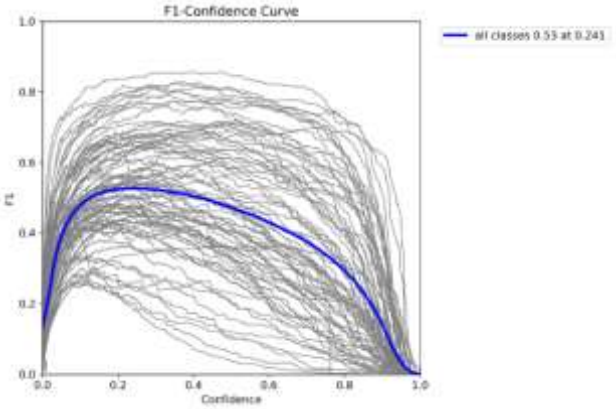

Figure 4: F1-Confidence Curve

F1-Confidence Curve: Figure 4 demonstrates the balance between precision and recall as the confidence threshold varies. The model achieves the highest F1 score for the `Airplane` class at around 0.791 confidence, while the `Hair-drier` and `Toothbrush` classes exhibit lower F1 scores of .261 across most confidence thresholds.

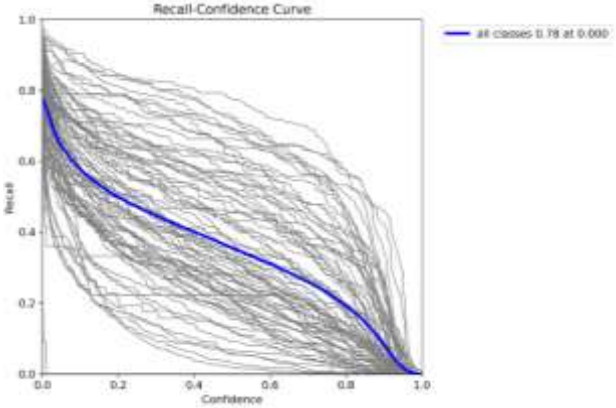

Figure 5 Recall confidence curve

10.48047/jocaaa.2024.33.08.275

Recall-Confidence Curve: Figure 5 shows how the recall metric changes with varying confidence levels. Higher recall is observed for classes such as `Person` and `Airplane`, with recall values exceeding 0.78 at lower confidence levels. This curve helps to understand the trade-off between recall and precision for a detection model, especially when deciding on an appropriate confidence threshold to use during inference, depending on whether you prioritize finding more true objects.

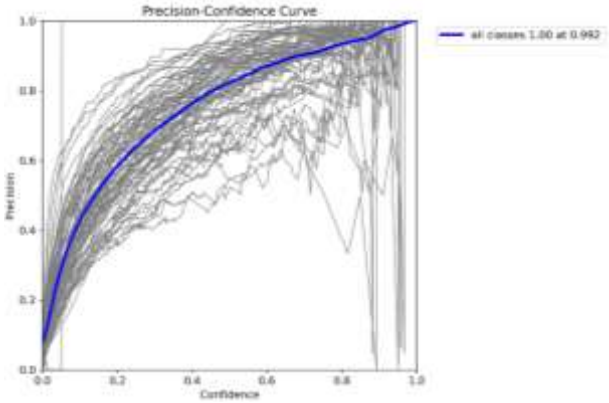

Figure 6 Precision confidence curve

Precision-Confidence Curve: Figure 6 illustrates that the proposed model maintains high precision across most classes as confidence increases. For all classes combined, precision reaches 1.00 at a confidence level of 0.992, indicating the model's strong ability to accurately identify objects at higher confidence thresholds. Some classes, like "Toothbrush" and "Scissors," show lower precision at varying confidence levels, suggesting room for improvement in those specific categories. Overall, the model demonstrates robust performance in precision across the majority of object classes.

Visual Results

The visual results, as in Figure 7, illustrate the detection and segmentation of objects in various scenarios from the COCO dataset. İt shows the proposed work's ability to correctly identify and label different objects, such as animals, cars, airplanes, and miscellaneous objects, even in complex environments. The provided labels are accompanied by confidence scores, which indicate the system's certainty regarding its predictions. The diverse range of scenes, encompassing urban streets and more open roads, underscores the resilience of the proposed approach across various contexts. The consistent accuracy in detecting and classifying objects, despite variations in lighting, object size, and occlusions, demonstrates the system's effectiveness and reliability. This visual evidence supports the previously discussed quantitative results, confirming that the proposed system performs well in real-world scenarios, maintaining high accuracy and precision.

10.48047/jocaaa.2024.33.08.275

Figure 7: Object detection and classification results across diverse applications

Each frame in the figure 7 represents a scenario where various objects are detected, and bounding boxes are drawn around the detected objects. These bounding boxes have associated labels to indicate the detected object's class (e.g., zebra, airplane, person, etc.) accurately. The images are part of a batch with each image having its unique identifier (e.g., "00000298251.jpg"), suggesting that this as the part of a validation set used for testing object detection accuracy across various environments.

Conclusions

The proposed system addresses several significant challenges, encompassing occlusions, rapid object movements, and varying object scales. Key components of the system, namely ASCNN, HOF-SG, and SEAT, have been developed and optimized to bolster detection accuracy, motion analysis, and tracking consistency. The incorporation of semantic information into the tracking process, as demonstrated by SEAT, proved notably effective in managing intricate scenarios involving occlusions and swift movements. Furthermore, the utilization of ASCNN and HOF-SG substantially contributed to the system's capacity to adapt to varying object scales and to precisely track dynamic objects. The field of dynamic real-time V-SLAM with ANN architecture is rich with opportunities for innovation. Researchers and practitioners should collaborate to address these challenges and pave the way for practical applications in robotics, autonomous vehicles, and beyond.

Statements and Declarations

The authors declare that there are no conflicts of interest related to the publication of this paper. The research work was conducted independently and without any financial or personal relationships that could be perceived as potential sources of bias.

References

[1]. J. K. Makhubela, T. Zuva, and O. Y. Agunbiade, “A Review on Vision Simultaneous Localization and Mapping (VSLAM),” in 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC), Dec. 2018, pp. 1–5. doi: 10.1109/ICONIC.2018.8601227.

[2]. “Advancements and Challenges in Visual-Only Simultaneous Localization and Mapping (V-SLAM): A Systematic Review | IEEE Conference Publication | IEEE Xplore.” Accessed: Oct. 27, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10624826

10.48047/jocaaa.2024.33.08.275

[3]. Y. Wang, Y. Zhang, L. Hu, G. Ge, W. Wang, and S. Tan, "Improved Feature Point Extraction Method of VSLAM in Low-Light Dynamic Environment," Electronics, vol. 13, no. 15, Art. no. 15, Jan. 2024, doi: 10.3390/electronics13152936.

[4]. A. Beghdadi and M. Mallem, "A comprehensive overview of dynamic visual SLAM and deep learning: concepts, methods and challenges," Mach. Vis. Appl., vol. 33, no. 4, p. 54, May 2022, doi: 10.1007/s00138-022-01306-w.

[5]. G. P. M. Bhat and D. N. G. Cholli, "A Comprehensive Survey on Computer-Vision Object Detection, Segmentation, Tracking, and Feature Extraction," Int. J. Adv. Sci. Technol., vol. 28, no. 16, Art. no. 16, Dec. 2019.

[6]. Y. Li, Y. Fu, and K. Wang, "A Method of Dense Point Cloud Slam Based on Improved Yolov8 and Fused with Orb-Slam3 to Cope with Dynamic Environments," May 30, 2024, Social Science Research Network, Rochester, NY: 4848367. doi: 10.2139/ssrn.4848367.

[7]. Y.-M. Hu, J.-J. Xie, H.-H. Shuai, C.-C. Huang, I.-F. Chou, and W.-H. Cheng, "Dynamic Feature Fusion for Visual Object Detection and Segmentation," in 2023 IEEE International Conference on Consumer Electronics (ICCE), Jan. 2023, pp. 01–06. doi: 10.1109/ICCE56470.2023.10043439.

[8]. A. Joshi, Amrita, R. S. Mathur, N. Kumar, and P. Tripathi, "Study of Traditional, Artificial Intelligence and Machine Learning Based Approaches for Moving Object Detection," in Mathematical Models Using Artificial Intelligence for Surveillance Systems, John Wiley & Sons, Ltd, 2024, pp. 187–214. doi: 10.1002/9781394200733.ch9.

[9]. T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," Apr. 19, 2017, arXiv: arXiv:1612.03144. doi: 10.48550/arXiv.1612.03144.

[10]. G. P. Bhat and N. G. Cholli, "Effective object detection using Tensorflow facilitated YOLOv3 model," in 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Dec. 2021, pp. 1–8. doi: 10.1109/CSITSS54238.2021.9683109.

[11]. V. N. Karnaukhov and M. G. Mozerov, "Development of Computer Vision, Image Processing, and Analysis at the Digital Optics Laboratory of the Institute for Information Transmission Problems of the Russian Academy of Sciences," Pattern Recognit. Image Anal., vol. 33, no. 4, pp. 1242–1249, Dec. 2023, doi: 10.1134/S1054661823040223.

[12]. B. Liu, "Research on Visual SLAM Method Based on Deep Learning in Dynamic Environments," in 2024 IEEE 6th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), May 2024, pp. 1645–1649. doi: 10.1109/IMCEC59810.2024.10575287.

[13]. X. Guo, M. Lyu, B. Xia, K. Zhang, and L. Zhang, "An Improved Visual SLAM Method with Adaptive Feature Extraction," Appl. Sci., vol. 13, no. 18, Art. no. 18, Jan. 2023, doi: 10.3390/app131810038.

[14]. G. Jocher, ultralytics/COCO2YOLO: Improvements. (May 11, 2019). Zenodo. doi: 10.5281/zenodo.2738323.

[15]. M. S. Hanif, S. Ahmad, and K. Khurshid, "On the improvement of foreground–background model-based object tracker," IET Comput. Vis., vol. 11, no. 6, pp. 488–496, Jul. 2017, doi: 10.1049/iet-cvi.2016.0487.

[16]. Y. Zhou and H. Qian, "Real-time object detection method with single-domain generalization based on YOLOv8," J. Real-Time Image Process., vol. 21, no. 6, p. 191, Nov. 2024, doi: 10.1007/s11554-024-01572-z.