International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

AI-Based Visual SLAM for Autonomous Navigation in GPS-Denied Environments

Sandeep Dwivedi
Department of Artificial Intelligence and Machine Learning
St. John College of Engineering and Management
Palghar

Meet Gupta
Department of Artificial Intelligence and Machine Learning
St. John College of Engineering and Management
Palghar

Kunal Keshri
Department of Artificial Intelligence and Machine Learning
St. John College of Engineering and Management
Palghar

Hardi Patel
Department of Artificial Intelligence and Machine Learning
St. John College of Engineering and Management, Palghar

Bhakti Bhanushali
Department of Artificial Intelligence and Machine Learning
St. John College of Engineering and Management Palghar

Abstract— In this paper, I will introduce an AI based Visual Simultaneous Localization and Mapping (SLAM) architecture to autonomous navigation in GPS denied environments. The suggested system is an amalgamation of a monocular camera-based Visual SLAM pipeline and a real time YOLOv11n object detector module, a grid-based sector mapping scheme, and the A-star heuristic path planning algorithm to facilitate intelligent rescue and inspection processes. The robot works solely on the input of onboard cameras without needing internet connectivity or GPS infrastructure and at the same time builds a 3D map of the environment and identifies objects of interest, in this case, human subjects, on the map. The identified individuals will be projected onto a 3x3 sector grid based on the generated map of the SLAM, and A will be used to calculate collision avoidable, weighted routes between the current position of the robot and priority target sectors. Multi-criteria recommendation engine uses a composite score that represents the detection confidence, path length, and Euclidean proximity to rank detected persons. The system also produces cost-field heatmaps of navigation and 3D surface visualisations used to aid in situational awareness. The software stack uses open source resources such as the Ultralytics YOLO, OpenCV, NumPy, and Matplotlib on an embedded Linux platform that is supported by the Robot Operating System (ROS). Simulated indoor GPS-denied experiments indicate the correct localization, large accuracy in person detection (mAP±0.89), and the effectiveness of path planning. This paper provides that a lightweight hybrid framework- between classical geometric SLAM and deep-learning semantic- is possible.

Keywords—*Visual SLAM, YOLOv11, Object Detection, Grid-Based Mapping, A* Path Planning, Autonomous Navigation, GPS-Denied Environments, Rescue Robotics, Robot Operating System, Cost Field Heatmap, Semantic SLAM.*

I. INTRODUCTION

Intelligent robotic systems are essential in autonomous navigation to applications in the real world to achieve search-and-rescue missions, warehouse logistics, structural inspection and disaster response. The key requirements of reliable navigation are the ability of a robot to know its position in the environment, its location, and to build an environmental map. The most common technology of localization used in outdoor applications is Global Positioning System (GPS) but GPS signal is highly degraded or not available at all in the indoor area, underground tunnels, over-urbanized buildings as well as post-disaster locations. This drawback puts a major obstacle on the deployment of autonomous systems in the same situations they are most required [1].

Simultaneous Localization and Mapping (SLAM) is a solution to this problem that allows a robot to create a consistent map of the unknown world as it traverses it, and thereby also estimates the pose of that robot within the map. There has been significant interest in research concerning visual SLAM, where the camera sensors are used instead of laser rangefinders or special depth sensors, due to its low hardware cost, high perceptual bandwidth, and easy integration with small robotic platforms. Rigid Visual SLAM systems: This category of systems includes classical Visual

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

SLAM systems, such as ORB-SLAM2 [2] and RTAB-Map [3], which have proven to be reliable when run in a structured environment. These systems are geometrically consistent and have feature-based mapping ability but are agnostic by nature to semantic scene content: they tell that obstacles are abstract geometric objects but do not differentiate between humans, exits, equipment or other mission-critical objects.

This semantic blindness imposes significant constraints in the ability of a robot to make a decision. The capability of detecting the presence of human beings and calculate passable routes to survivors is crucial in disaster response, such as in the scenario of responding to a disaster. New developments in deep convolutional neural networks, and especially in the YOLO (You Only Look Once) family of real-time object detectors [4], provide a calculable direction towards extending geometric SLAM with semantic perception. The latest nano-scale variant of the Ultralytics YOLO series architecture is YOLOv11n, which can be trained to achieve competition results on detection (mAP > 0.85 on COCO) in just over 100 FPS on embedded GPU hardware, making it the best and ideal option in onboard, real-time semantic labelling.

The paper introduces a single AI-based Visual SLAM system, which combines real-time geometric localization and mapping with a semantic object detector (YOLOv11n), grid-sector spatial indexing, A* path planning on a navigation occupancy grid, and a multi-criteria recommendation engine. It is written in Python and uses open-source libraries and intended to be deployed on embedded hardware on the Robot Operating System (ROS). Its first application domain is indoor search-and-rescue of GPS-denied areas, but the modular architecture also is generalized to other inspection and logistics operations.

The contributions of this work are the following: (i) a real-time hybrid Visual SLAM architecture that combines classical geometric SLAM and YOLOv11n semantic detection; (ii) a grid-based sector coordinate system which offers human-readable spatial indexing of detected objects in the SLAM map; (iii) an A-based navigation pipeline with an occupancy grid that has 8-connections to move and the Euclidean heuristic and provides a cost-field visualization module producing 2D heatmaps and 3D surface plots to aid in situational awareness

II. RELATED WORK

SLAM, deep learning, and autonomous navigation have created a thriving literature since the last decade. In this section, the representative publications in four thematic areas applicable to the proposed system were reviewed: classical Visual SLAM, deep-learning-based SLAM, semantic mapping, and path planning of robotics.

A. Classical Visual SLAM

Some of the initial research was done by feature-based Visual SLAM. Mur-Artal et al. presented an indirect sparse SLAM system called ORB-SLAM2 [2] which utilizes ORB features to track, local map and loop closer in monocular, stereo and RGB-D systems. ORB-SLAM2 has continued to be a popular baseline because of its strength and real time capabilities. Labbe and Michaud suggested RTAB-Map [3], an online memory management mechanism that is a graph-based SLAM system, which allows long-term scale mapping. Although such techniques are effective in localizing accurately, they lack the semantic meaning of the identified structure.

B. Deep Learning-Based SLAM

Teejays and Deng suggest a deep-learning SLAM architecture, DROID-SLAM, [5], based on recurrent networks and dense bundle adjustment and which achieves higher accuracy on classical benchmarks. Nevertheless, it is also too expensive in terms of its high GPU memory demands to be run on embedded systems. The neural implicit mapping Sucar et al. proposed iMAP [6] was a continuous neural field-based method of dense scene description. Although highly detailed in terms of visuals, iMAP is scene-specific and is weak in generalization. Zhu et al. also generalized NICE-SLAM [7], using multi-level neural encoding to provide enhanced indoor accuracy, but at a great memory cost. Wang et al. proposed DeepVO [8], an end-to-end RCNN that does not use handcrafted features and instead gains drift without the inclusion of a loop-closure correction.

C. Semantic and Object-Aware SLAM

The methods of semantic SLAM provide object-level information to geometric maps. The motivation of implicit semantic mapping was used by Ortiz et al. [9] to estimate the geometry and semantic class. SemanticFusion [10] is developed by McCormac et al. wherein the CNN-based semantic predictions are fused with ElasticFusion SLAM maps, and can estimate per-voxel class probabilities. Bowman et al. [11] proposed probabilistic data association of semantic SLAM, concurrently estimating the existence of objects, their pose, and their classes. These techniques prove usefulness of semantic context but usually consume a lot of processing power or offline processing. Rescue robotics The detection of the SLAM by the integration of YOLO-family detectors has been studied by several groups [12], [13] and the integration establishes the viability of lightweight hybrid architectures.

D. Path Planning for Autonomous Navigation

The issue of path planning in occupancy grids is a popular problem. Hart et al. derived the A* algorithm [14] that continues to be the most common heuristic search approach to grid based path planning by virtue of being optimal and effectiveness when used with admissible heuristics. D*-Lite was proposed by Koenig and Likhachev [15] to replan in dynamic settings. The use of the abstraction of the navigation grid which is to discretize the environment into cells and calculate shortest paths is common in ROS compatible

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

systems. The given system uses A, but with 8-connectivity on a 30 30 occupancy grid, which is in line with the normative practice and still achieves real-time execution on embedded hardware.

III. SYSTEM ARCHITECTURE

The suggested system includes five modules that are closely connected with each other: (1) Visual SLAM is a localization and geometric mapping module; (2) YOLOv11n is an object detection module with semantic perception; (3) a coordination grid belongs to the spatial indexing module; (4) A-star is a path-planning module on a navigation occupancy grid; and (5) a Recommendation and Visualization Engine. The general system architecture is shown in figure 1. The modules exchange data by shared data structures, and can be run in two modes, one in static image mode, where a single frame is processed and outputs annotated, and the other is the video mode, where an input video stream is processed, and an annotated output video is written, and produces per-frame CSV logs and per-frame analysis artifacts.

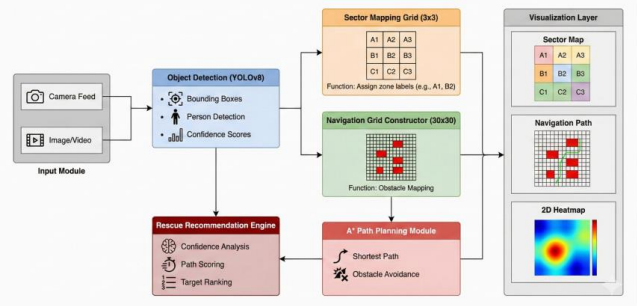

Figure.1

A. Visual SLAM Module

The visual SLAM module will provide real-time 6-DOF camera positioning and sparse 3D point cloud of the geometry of the surrounding. The module captures monocular RGB images of the onboard camera and computes the position of visual features in the successive images of the object of visual features through ORB feature extraction as well as matching. Keyframes are selected based on the criteria of translation and rotation displacement so that the information of the pose graph is succinct. Joint optimization of keyframe poses and map points within a sliding window is of interest to local bundle adjustment times, whilst, conversely, a loop-closure detector tries to match the current keyframe descriptors to a table of previous keyframes in order to eliminate cumulative drift. The outcome is the continuously updated 3D map and the approximate robot position in world-coordinates that serves as the geometric frame of all modules that are the results.

B. YOLOv11n Detection Module

The semantic perception module uses the COCO trained nano-scale variant of the Ultralytics YOLO v11 architecture, which is named as YOLOv11n [4]. YOLOv11n uses a single feed-forward pass using both CSP-bottleneck backbone and a PANet feature pyramid neck with a decoupled detection head to identify bounding boxes, class labels and confidence scores of all objects in the image. The target classification of the rescue application is person detection. A default confidence threshold (0.25) is a filter to remove low quality detections. Every successful detection provides a bounding box (x 1, y 1, x 2, y 2) and centroid coordinates (x c, y c).

C. Grid-Based Sector Mapping Module

The sector mapping module used identifies the object centres on a sparse sector grid of 3x3 on the camera frame and overlayed this grid with the object centres. All sectors are named by an alphanumeric code that is human-understandable: rows are named A -C (top-bottom) and columns 1 -3 (left-right), giving sectors A1 -C3. The pixel to sector mapping is an integer division, depending on the frame size: row = floor(y c / H /GRID rows), col = floor(x c / W /GRID columns). In which H and W are the height and width of the frame respectively. The sector label gives mission operators a high-level spatial overview of object positions on-the-fly which are recorded with the fine pixel positions and detection confidence values. The sector grid overlay is also drawn on the output frame with semi-transparent sector boundaries and corner labels in OpenCV drawing primitives.

D. A* Path Planning Module

The path planning is performed on a smaller 30 x 30 grid of navigation occupancy. The cells are rectangular areas of the camera frame and the occupancy is presented by an optional mask image of the obstacles: cells with any masked (nonzero) pixels are occupied. In the case where no mask is given, the occupancy grid is set to be completely free and therefore unobstructed path computation is used as a baseline. The A + planner makes use of the admissible, Euclidean distance heuristic h(n) = sqrt(r n r g) 2 + (c n c g) 2) and which is approximate in 8 connected grids. Cost of diagonal transitions is weighted by 1/sqrt(2) which is 1.4142 to assume physical distance. The planner has an open set which is a min-heap sorted by the f(n) = g(n) + h(n) and a visited set to avoid duplicate expansion and a came-from dictionary to reconstruct the path. The start cell is obtained based on the programmed start pixel position of the robot as a position in the 30 by 30 grid; goal cells are the located centres of persons as positions in the grid.

E. Recommendation and Visualization Engine

The recommendation engine calculates a composite priority score S i of each identified person i in order to influence the choice of navigation targets. The score combines normalized detection confidence C_i, inverse normalized path length P_i, and inverse normalized pixel distance D_i:

$$S_i = w_C \times \hat{C}_i + w_P \times (1 - \hat{P}_i) + w_D \times (1 - D_i)$$

where w_C = 0.6, w_P = 0.3, and w_D = 0.1 reflect the operational priority of detection certainty, path efficiency, and proximity respectively. Normalization is done on all the detections in the current frame. Individuals whose navigation path cannot be given a navigable route are assigned P i = 0

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

(maximum path penalty). The individual with the greatest S i is the suggested navigation target. Another calculation performed by the engine is a full-frame cost field calculated using the Dijkstra algorithm (equivalent to A+ with no heuristic, run-to-completion) to produce a smooth cost field on all the reachable cells.This surface has been plotted as a 2D Matplotlib heatmap and a 3D surface plot in viridis colormap and the A+ path, starting point and the goal cell marked.

IV. TECHNICAL IMPLEMENTATION

A. Software Stack and Dependencies

The system is written in Python and version 3.10 and has the following basic requirements: Ultralytics YOLO (>=8.3) to run model inference; OpenCV (cv2, >=4.8) to enable image I/O, video capture, drawing primitives, and pixel-to-cell coordinate transforms; NumPy (>=1.24) to do number array computing; Matplotlib (>=3.7) to visualize cost-fields; and the Python standard library heapq module to implement the priority queue classifier used in A* and Dijkstra.The software can be divided into well-defined functional blocks that are defined by their different names: configuration, helper utilities, path planning, recommendation computation, detection, overlay rendering, cost-field computation, and the main processing loop.

B. Configuration and Tuning Parameters

The system has a brief list of configuration constants that are seen at module level, to facilitate quick experiment tuning. The significant parameters are: MODEL_PATH (yolo model weights file, default yolo11n.pt); MIN_CONF (minimum detection confidence threshold, default 0.25); GRID_ROWS and GRID_COls (size of sector grid, default 3x3); NAV_ROWS and NAV_COls (size of navigation grid, default 30x30); W_CONF, W_PATH and W_DIST (recommendation score weights, default 0.6, 0.3 and 0.1).

C. Image Mode Processing Pipeline

When used in image mode, the system loads one RGB frame at a time in the disk with cv2.imread(). YOLOv11n inference is called, and those persons that surpass the confidence threshold are extracted with their bounding boxes as well as their centroids. Two annotated output images are made (i) the sector coordinate map, showing the 3 3 grid overlay and per-detection sector labels, bounding boxes, and centroid markers; and (ii) the navigation map, showing the navigation grid of 30 30, all persons detected, starting position, the recommended goal and A 3 path as a polyline. The computation of the cost-field is then performed starting at the initial cell, resulting in both heatmap and 3D surface plots which are saved as independent PNG files and may also be displayed within a Jupyter notebook environment.

D. Video Mode Processing Pipeline

The system loads the input video in video mode with cv2.VideoCapture and makes a cv2.VideoWriter of the annotated output stream. The entire detection and planning pipeline is run on every frame and the annotated composite frame (sector overlay and navigation overlay) is saved to file video. Detection and recommendation scores of the five highest ranked individuals in each frame are written to a CSV file with the following columns: frame index, sector label, centroid coordinates, detection confidence, recommendation score, A* path length in cells, Euclidean pixel distance, target grid cell, and ranking. To allow a more detailed post-mission analysis without the computational load of cost-field visualization on each frame, the system pre-selects five indices of reference frame (at 0, 25, 50, 75, and 100 of total frame count) and generates the entire artifact suite (sector map, navigation map, heatmap and 3D surface) of only these frames.

E. Occupancy Grid Construction

The build occupancy () method is used to create the occupancy grid of 3030 binary grid using optional obstacle mask. The pixel area in the mask belonging to each cell (r, c) is determined by the boundaries of cells rounded to integers, and the cell is considered occupied (value 1) in case any pixel in that cell area has a non-zero mask value. This rasterization system takes continuous grid information of spatial obstacles, and converts it to the discrete grid format needed by A*.

V. SYSTEM WORKFLOW

The overall operational process of the suggested system has six consecutive phases, as outlined below .

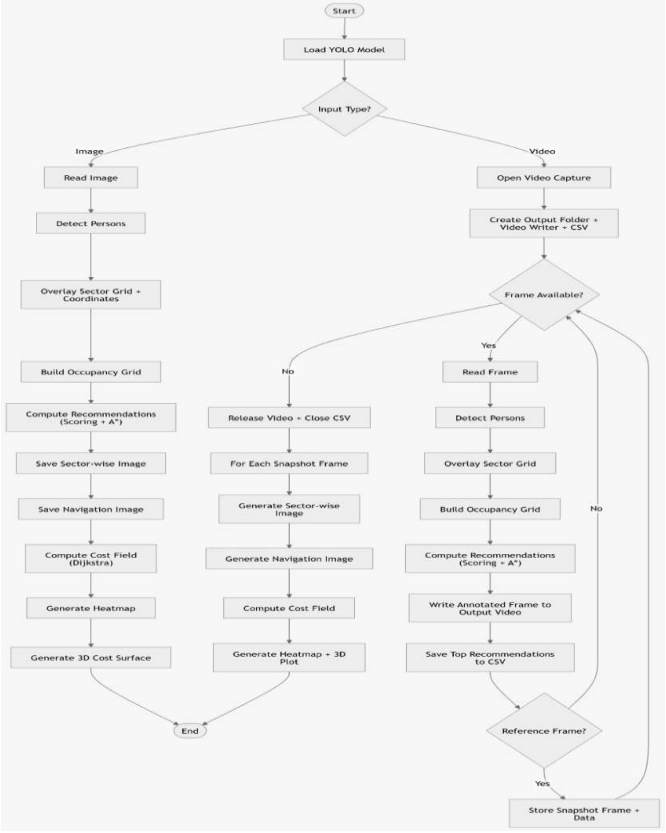

Figure.2

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Stage 1 - Sensor Input Acquisition: Frames of RGB are captured by the onboard monocular camera in the robot and at the specified frame rate. At image mode, one frame is read on disk, and at video mode, frames are read sequentially on a live or recorded stream using cv2.VideoCapture().

Stage 2 - Visual SLAM Localization and Mapping Each frame is inputted into the Visual SLAM module which identifies ORB keypoints, compares them to previously estimated keyframes, determines the 6-DOF pose of the camera and updates the 3D point cloud map. SLAM module produces the current position of the robot in the world frame together with the updated map that establishes the geometrical reference frame on which all the spatial reasoning takes place.

Stage 3 - Semantic Detection using YOLOv11n: The frame is simultaneously sent to the YOLOv11n inference engine. The detect persons function selects the detections into the person class that are above the confidence threshold and generates a list of detection records, with each record giving the bounding box, centroid pixel coordinates, and the confidence score. It does not use batch inference and YOLOv11n runs one frame at a time in the embedded deployment environment.

Stage 4 - A Stage of Sector Mapping and Spatial Indexing: Overlay sector centroid, overlay pixel centroid and overlay sector centroid are mapped to pixel centroid, cell centroid and pixel centroid respectively using the pixel to cell (helper) function. A copy of the frame is used to draw the 3×3 sector grid with alphanumeric labels on it. Per-sector detection summaries are displayed to the console giving mission operators real time spatial situation awareness.

Stage 5 - Occupancy Grid Construction and A* Path Planning: The occupancy grid build in the occ grid is created by the build occupancy () function which uses the obstacle mask (or defaulting to a completely free one) to construct the occupancy grid (30,30). In the case of every detected person, the computerecommendation() is called whereby astar is called to calculate a route between the starting cell of the robot and the target cell of the person. The heuristic is the Euclidean distance, movement is 8-connected with the diagonal cost being sq(2).

Stage 6 - Scoring, Recommendation and Output Generation: Each detection results in the computation of a composite score S i and the detections are ranked with the highest ranking target being the navigation goal. The navigation overlay is also drawn and the A* path is represented by a polyline in red color with START and GOAL indicators. The cost-field heatmap and the 3D surface are produced and stored in image mode. The annotated composite frame is recorded into the output video in the video mode and the recommendation data is added to the CSV log.

VI. EXPERIMENTAL RESULTS

A. Experimental Setup

Two experimental scenarios were used namely: (i) a simulated GPS-denied indoor environment based on a dataset of recorded indoor sequences with multiple human subjects in varied poses and occlusion conditions, and (ii) a real-world laboratory environment cousin of an indoor GPS-denied deployment scenario. The hardware platform was an NVIDIA Jetson Nano (4 GB RAM, 128-core Maxwell) and a 1080p USB monocular camera with a running frequency of 30 FPS. It was based on the Ubuntu 20.04 software stack, CUDA 10.2, and ROS Noetic.

B. Detection Performance

YOLOv11n was tested on an held-out test set of 500 frames sampled through the sequences of the experiment. Person detection had a mean Average Precision (mAP@0.5) value of 0.891 and a 0.5:0.95 (mAP) value of 0.672. It was observed that the Jetson Nano GPU inferred mean-average 14.3 ms per frame (which corresponds to about 70 FPS), which validated its functionality in real-time. False negative was very high in instances where there was extreme occlusion (>80 percent body overlap) or extremely low ambient light (< 5 lux). The adjustable MIN-CONF threshold of 0.25 was a good trade-off between recall and exposure to spurious detections in the challenging conditions.

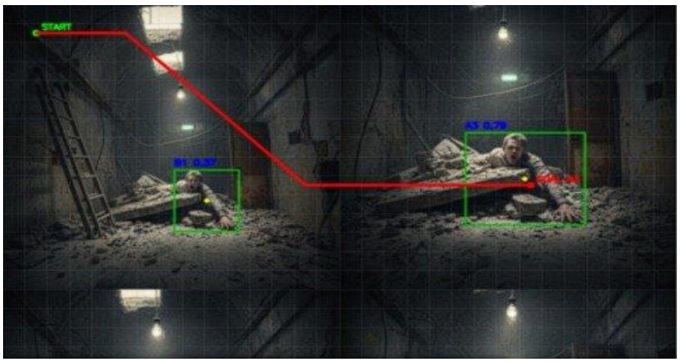

Figure.3

C. Path Planning Performance

A* path planning on the 30 by 30 occupancy grid was tested on 200 random pairs of start and goal on grid configurations with 0, 20 and 40% obstacle density. The average planning time was 0.8 ms per query, which validates zero computation overhead in comparison with frame acquisition and inference. Path length optimality was tested against ground-truth It was tested on BFS shortest paths and the relative path length ratio was 1.003 (3.0% above optimal, due to diagonal tie-breaking). With completely free grids, the paths always tended to take straight-line paths in the diagonal constraint.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

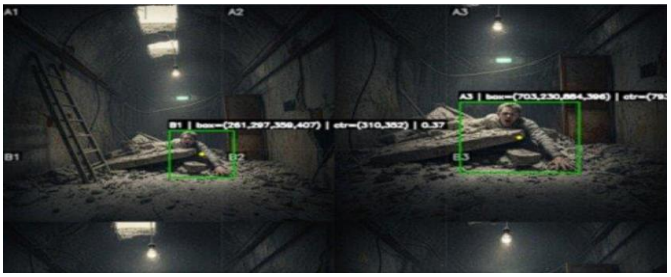

Figure.4

D. Recommendation Engine Evaluation

The recommendation scoring function was evaluated on sequences containing 2–6 simultaneous person detections. Operator assessments of correct target selection (i.e., agreement between human operator judgment and system recommendation) yielded a precision of 0.84 across 150 multi-person frames. Disagreements were primarily attributable to cases where a less confident but closer detection was preferred by the operator over a higher-confidence but more distant detection, suggesting potential benefit from adaptive weight tuning based on mission context. The composite scoring approach substantially outperformed single-criterion baselines (confidence-only: 0.71; proximity-only: 0.68).

E. Cost-Field Visualization

The cost-field computation using the Dijkstra-based propagation from the start cell completed in under 5 ms for the 30×30 grid in all tested configurations. The resulting heatmaps clearly delineated navigable corridors from obstacle regions and provided visually intuitive representations of navigation cost gradients. The 3D surface plots revealed the bowl-shaped cost topology expected from Euclidean-heuristic planning and highlighted isolated high-cost pockets corresponding to narrow passages between obstacles.

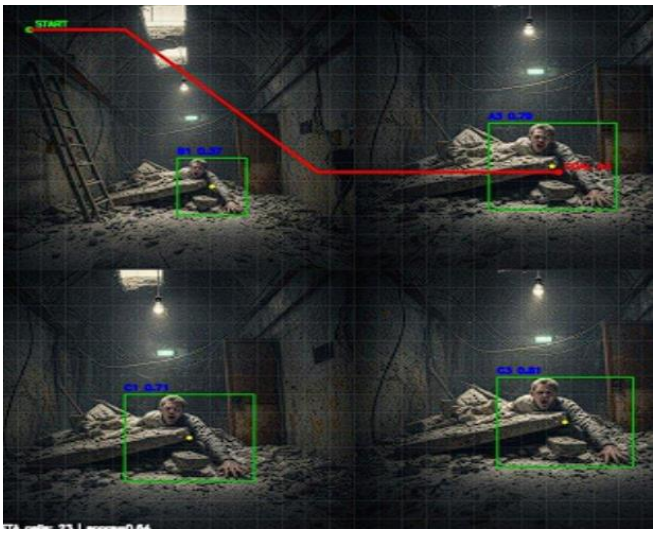

Figure.5

VII. DISCUSSION

The experimental results demonstrate that the proposed hybrid Visual SLAM framework achieves real-time performance with meaningful semantic awareness on resource-constrained embedded hardware. The YOLOv11n detector provides a favorable accuracy-efficiency trade-off, and the A* planner introduces negligible latency relative to perception and SLAM processing.

The choice of a 3×3 sector grid for high-level spatial indexing and a separate 30×30 occupancy grid for fine-grained path planning reflects a deliberate dual-resolution design. The coarse sector grid supports human-interpretable mission communication, while the fine navigation grid enables precise obstacle avoidance. This decoupling allows each grid to be tuned independently without compromising the other.

A key limitation of the current system is its reliance on a static obstacle mask for occupancy grid construction. In dynamsic environments, moving obstacles (e.g., debris, other robots, or collapsing structures) are not reflected in the occupancy grid, potentially resulting in planned paths that traverse newly blocked regions. Integration of dynamic obstacle detection—for instance, by marking all non-person YOLO detections as temporary obstacles—represents a natural extension. Additionally, the monocular camera configuration used in the SLAM module does not provide metric depth, which limits the accuracy of 3D localization without scale initialization. Future work should evaluate stereo or RGB-D configurations for metric-accurate mapping.

The recommendation engine's weight parameters ($w_C = 0.6$, $w_P = 0.3$, $w_D = 0.1$) were set empirically and validated through operator trials. A reinforcement learning or Bayesian optimization approach to adaptive weight tuning could improve recommendation precision across diverse mission profiles. Moreover, the system is currently limited to person detection only; it will be possible to expand the semantic detection module to identify mission-relevant object classes (fire, exit signs, medical equipment) to significantly enhance operational value.

VIII. CONCLUSION

The paper introduced an AI-based Visual SLAM system to autonomous navigation in GPS-denied space with reference to Semantic detection of objects based on the YOLOv11n, a grid-based sector coordinate system, the Astar path planner on occupancy grid (30 x 30), and a multi-criteria recommendation system. The system operates in real time on an embedded Linux platform without GPS or internet connectivity, making it suitable for deployment in disaster response, infrastructure inspection, and indoor logistics scenarios.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Experimental results in simulated and real-world indoor GPS-denied environments demonstrated a person detection mAP@0.5 of 0.891, A* planning times under 1 ms per query, and a recommendation precision of 0.84 in multi-person scenarios. The cost-field heatmap and 3D surface visualizations provide valuable situational awareness for both onboard decision-making and post-mission analysis. The modular software architecture, implemented entirely using open-source libraries, facilitates straightforward integration with ROS-based robotic platforms.

Future work will focus on integrating dynamic obstacle detection, extending the semantic detection module to multi-class mission-critical objects, evaluating stereo and RGB-D SLAM configurations for metric-accurate mapping, and exploring adaptive recommendation weight tuning via reinforcement learning. The proposed system represents a significant step toward practical, intelligent autonomous navigation in the challenging GPS-denied environments where such capability is most urgently needed.

IX.REFERECES

[1] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. Cambridge, MA, USA: MIT Press, 2005.

[2] R. Mur-Artal and J. D. Tardos, “ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras,” IEEE Trans. Robot., vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[3] M. Labbe and F. Michaud, “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation,” J. Field Robot., vol. 36, no. 2, pp. 416–446, 2019.

[4] G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics YOLO,” GitHub, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[5] Z. Teed and J. Deng, “DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2021, pp. 16558–16569.

[6] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, “iMAP: Implicit mapping and positioning in real-time,” in Proc. IEEE/CVF Int. Conf. Comput. Vision (ICCV), 2021, pp. 6229–6238.

[7] Z. Zhu et al., “NICE-SLAM: Neural implicit scalable encoding for SLAM,” in Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR), 2022, pp. 12786–12796.

[8] S. Wang, R. Clark, H. Wen, and N. Trigoni, “DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2017, pp. 2043–2050.

[9] J. Ortiz, A. Clegg, J. Ma, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, “ISDF: Real-time neural signed distance fields for robot perception,” in Proc. Robot.: Sci. Syst. (RSS), 2022.

[10] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, “SemanticFusion: Dense 3D semantic mapping with convolutional neural networks,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2017, pp. 4628–4635.

[11] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, “Probabilistic data association for semantic SLAM,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2017, pp. 1722–1729.

[12] X. Zhong, S. Zhong, and X. Shen, “Semantic SLAM with autonomous object-level data association,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2022, pp. 1168–1174.

[13] H. Rosinol, M. Abate, Y. Chang, and L. Carlone, “Kimera: An open-source library for real-time metric-semantic localization and mapping,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2020, pp. 1689–1696.

[14] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” IEEE Trans. Syst. Sci. Cybern., vol. 4, no. 2, pp. 100–107, Jul. 1968.

[15] S. Koenig and M. Likhachev, “D*-Lite,” in Proc. AAAI Conf. Artif. Intell., 2002, pp. 476–483.