AUTONOMOUS VEHICLE NAVIGATION SYSTEM USING MACHINE LEARNING AND GRAPH BASED SLAM

Azzam Wildan Maulana
Departemen of Electrical Engineering
Faculty of Intelligent Electrical and Informatics Technolog
Sepuluh Nopember Institute of Technology
Surabaya, Indonesia
azzamwildan462@gmail.com

Rudy Dikairono
Department of Electrical Engineering
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
rudydikairono@its.ac.id

Ahmad Zaini
Department of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
zaini@its.ac.id

Mauridhi Hery Purnomo
Departemen of Computer Engineering
Faculty of Intelligent Electrical and Informatics Technolog
Sepuluh Nopember Institute of Technology
Surabaya, Indonesia
hery@ee.its.ac.id

***Abstract*—Icar is an autonomous vehicle developed by Institut Teknologi Sepuluh Nopember (ITS) to autonomously navigate the campus environment. Its navigation system relies on pose estimation using sensor fusion between odometry and the Global Navigation Satellite System (GNSS). However, GNSS signals are prone to degradation in environments with obstacles such as tall buildings and trees, while odometry suffers from inaccuracies caused by wheel slippage, encoder drift, and inertial sensor errors. These limitations result in unreliable localization, potentially causing Icar to deviate from its designated trajectory.**

To address these issues, this research proposes an enhanced navigation system by integrating a stereo depth camera and LIDAR. Road detection is performed using a semantic segmentation model based on deep learning, enabling Icar to distinguish road surfaces from non-road areas. Pose refinement is carried out using graph-based optimization within a SLAM framework, incorporating data from the depth camera and LIDAR processed via the Iterative Closest Point (ICP) algorithm. The proposed system improves localization robustness and offers a GNSS-independent alternative for safe and accurate autonomous navigation. The final system is expected to reduce dependency on GNSS and increase Icar's reliability in semi-structured or dynamic environments.

***Index Terms*—Autonomous vehicle, Icar, GNSS, Odometry, Pose Estimation, Stereo Depth Camera, Semantic Segmentation, SLAM, ICP, Graph-Based Optimization, Deep Learning**

I. Background

Icar is an autonomous vehicle developed by Institut Teknologi Sepuluh Nopember (ITS). Icar is designed to operate automatically and navigate throughout the ITS campus. Its control system relies on pose estimation (i.e., position and orientation) derived from odometry and sensor fusion with the Global Navigation Satellite System (GNSS). The estimated pose is then used to guide Icar toward a predetermined destination.

The pose of Icar is determined through the combination of odometry and GNSS data. Odometry estimates pose based on wheel rotations, augmented by a gyroscope to obtain orientation. GNSS, on the other hand, estimates pose based on satellite signals. These two sources are fused using the Extended Kalman Filter algorithm to achieve higher accuracy in pose estimation [1].

However, the data provided by GNSS is not always fully reliable. Several environmental conditions, such as tall buildings or dense tree canopies, can obstruct satellite signals. This may cause a phenomenon known as multipath, in which satellite signals reach the receiver via multiple indirect paths, resulting in errors in position and orientation estimation [2].

Errors can also arise from the odometry system due to factors such as inaccurate wheel travel distance, incorrect wheel rotation angle calculations, or faulty IMU sensor readings. These inaccuracies may result from differences in wheel diameters, uneven tire pressure, or wheel slippage. Moreover, errors in wheel rotation angle measurements may originate from malfunctioning encoder sensors [3].

When errors occur in pose estimation, Icar may deviate from its intended trajectory. This can lead to safety concerns, such as collisions with obstacles or deviation from designated transportation lanes. Therefore, an additional navigation system is required to enhance pose accuracy and ensure safe autonomous navigation.

A promising enhancement is the integration of a stereo depth camera, which provides visual depth information to detect roads. This road detection process employs a ma-

chine learning algorithm—specifically, semantic segmentation—trained on labeled datasets to distinguish between road and non-road areas [4].

Beyond road detection, the stereo depth camera is also used to refine pose estimation through graph-based optimization. Furthermore, a LIDAR sensor is incorporated and processed using the Iterative Closest Point (ICP) algorithm to further improve localization accuracy. Together, these methods form a Simultaneous Localization and Mapping (SLAM) system [5], which is expected to potentially replace the role of GNSS entirely.

II. Literature Review

There are several studies to dive deeper into the topic of autonomous vehicle navigation systems. These studies focus on various aspects, including pose estimation, road detection, and SLAM implementation.

A. Odometry

Odometry is an algorithm used to estimate how far a robot has traveled. In a 2D pose system, the robot's movement is represented as $x, y,$ yaw. The simplest form of 2D odometry combines motor rotation data with orientation data from an Inertial Measurement Unit (IMU). By applying forward kinematics to this sensor data, the robot's pose can be estimated relative to its initial position [1].

B. Kalman Filter

The Kalman Filter is a powerful linear estimation algorithm used to predict and update the state of dynamic systems based on noisy sensor data. Introduced by Rudolf E. Kalman in 1960, it enables accurate real-time estimation of variables such as position and velocity [6]. In robotics and control systems, the Kalman Filter is instrumental in fusing data from multiple sensors such as GNSS, IMUs, and wheel encoders.

C. Semantic Segmentation

Fast-SCNN (Fast Semantic Segmentation Convolutional Neural Network) was designed for real-time performance on resource-constrained devices, making it ideal for embedded systems like autonomous vehicles. It combines a lightweight encoder-decoder structure with efficient components such as depthwise separable convolutions and a streamlined feature fusion strategy. These design choices enable Fast-SCNN to achieve a good balance between accuracy and speed, ensuring that road segmentation can be performed in real time without requiring GPU-level hardware.

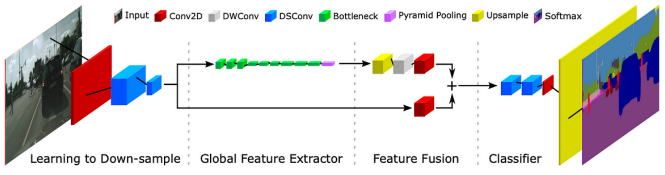

Fig. 1. Fast SCNN Architecture [7]

Fast-SCNN consists of four main modules: Learning to Downsample, Global Feature Extractor, Feature Fusion Module, and Classifier. The network first reduces the spatial dimensions of the input image while increasing feature richness, then extracts global context features and fuses them with fine spatial details. This fused representation is finally classified at the pixel level to distinguish road and non-road areas. In the Icar system, this segmented output supports safe navigation by identifying drivable regions based on real-time camera input [7].

D. Graph-Based SLAM

SLAM techniques can be divided into two broad categories: filter-based SLAM (e.g., Extended Kalman Filter SLAM) and graph-based SLAM. In filter-based SLAM, the robot's pose and landmark positions are treated as stochastic variables updated recursively. In contrast, graph-based SLAM builds a graph where nodes represent robot poses and landmarks, and edges represent spatial constraints derived from sensor observations [5].

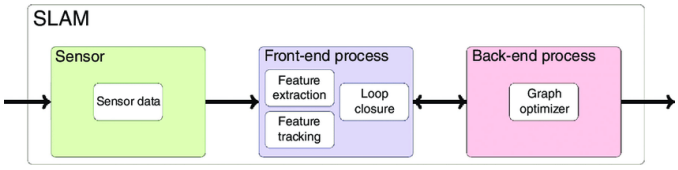

Fig. 2. Basic Graph-Based SLAM Architecture [8]

There are three main process that can be seen from figure 2: The first process is sensor data, it means like acquiring data from sensors such as cameras, LIDAR, or IMU. The second process is front end process, The front end process is responsible for extracting features from the sensor data and finding a loop closure. The last process is back end process, the back end process is responsible for optimizing the graph to find the best estimate of the robot's trajectory and the map of the environment [8].

The optimization in Graph-based SLAM can be efficiently implemented using frameworks like GTSAM (Georgia Tech Smoothing and Mapping), a C++ library that applies nonlinear optimization over factor graphs. Factor graphs represent probabilistic relationships among variables such as odometry, landmarks, and sensor observations, and allow for efficient optimization of robot trajectories [9].

E. Bicycle Model

The bicycle model is a simplified mathematical representation of vehicle dynamics. It models a vehicle as a two-wheel system—one at the front and one at the rear—mimicking the dynamics of a bicycle. This model is widely used in vehicle control and trajectory planning due to its simplicity and ability to capture basic vehicle behavior [10]. In robotics and electrical engineering, the bicycle model is commonly used to develop navigation and motion control algorithms for autonomous ground vehicles.

$$\delta_{\text{target}} = \tan^{-1}\left(\frac{2 \cdot L \cdot \sin(\theta_{\text{direction}})}{D_{\text{lookahead}}}\right) \tag{1}$$

where δ_{target} is the target steering angle, L is the distance between the front and rear wheels, $\theta_{\text{direction}}$ is the heading angle, and $D_{\text{lookahead}}$ is the lookahead distance.

The main advantage of the bicycle model is its applicability in control and simulation algorithms such as PID, LQR, and MPC. Many autonomous vehicle systems use it as a foundational model for motion prediction and path following. For instance, it enables calculation of the optimal steering angle required for a vehicle to accurately and stably follow a predefined trajectory [11].

III. Methodology

There are some proposed methods to improve the navigation system of Icar.

A. Overview of the Icar System

The navigation system in Icar is divided into two types: the legacy navigation system using GNSS and the new navigation system that operates without GNSS. Below are block diagrams of Icar's navigation systems with and without GNSS.

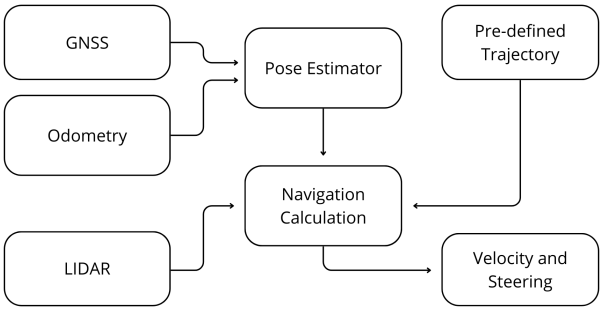

Fig. 3. Block Diagram of the Legacy Icar System with GNSS

Figure 3 illustrates the legacy Icar system, which relies on GNSS and odometry for pose estimation. The system uses a Pre-defined Trajectory to determine the target speed and steering angle using the Bicycle Model, as shown in equation 1.

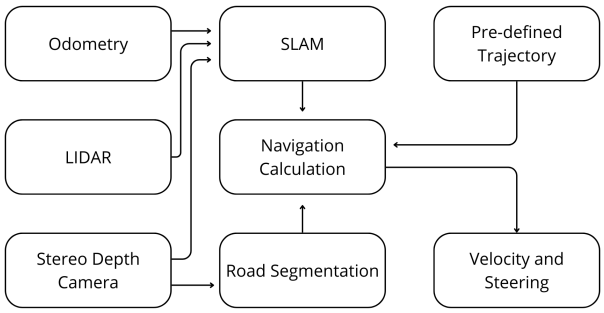

Fig. 4. Block Diagram of the New Icar System without GNSS

Figure 4 illustrates the new proposed Icar system, which does not rely on GNSS. Instead, it uses a stereo depth camera and LIDAR for pose estimation. The system also employs a Pre-defined Trajectory to determine the target speed and steering angle using the Bicycle Model, as shown in equation 2. Below is the full equation for the Bicycle Model used in the new system:

$$\begin{aligned}
\Delta x &= x_{\text{waypoint}} - x_{\text{position}} \\
\Delta y &= y_{\text{waypoint}} - y_{\text{position}} \\
\theta_{\text{direction}} &= \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) - \theta_{\text{orientation}} \\
v_{\text{target}} &= \min\left(\sqrt{\Delta x^2 + \Delta y^2},\ v_{\text{max}}\right) \\
\delta_{\text{target}} &= \tan^{-1}\left(\frac{2 \cdot L \cdot \sin(\theta_{\text{direction}})}{D_{\text{lookahead}}}\right)
\end{aligned} \tag{2}$$

Where x_{waypoint} and y_{waypoint} are the coordinates of the waypoint obtained from the Pre-defined Trajectory, x_{position} and y_{position} represent the vehicle's current position, $\theta_{\text{direction}}$ is the heading angle, $\theta_{\text{orientation}}$ is the current orientation, v_{target} is the target velocity, v_{max} is the maximum velocity, δ_{target} is the target steering angle, L is the distance between the front and rear wheels, and $D_{\text{lookahead}}$ is the lookahead distance.

The primary difference between the two systems is the source of position and orientation data. The legacy system uses GNSS and odometry, while the new system uses stereo depth cameras, LIDAR, and odometry. Additionally, the new system integrates road detection to enhance navigation safety.

B. Road Detection System

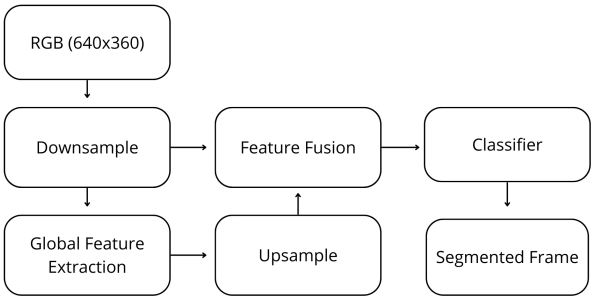

Fig. 5. Block Diagram of the Machine Learning Architecture on Icar

From Figure 5, it is shown that the road detection system on Icar utilizes a machine learning architecture, specifically using Fast-SCNN [7] as the core model. Its architecture differs in the convolution channels and pooling methods. The smaller convolution channels and the use of average pooling instead of pyramid pooling are intended to reduce computational load, making it suitable for real-time applications on Icar.

1) Downsampling: In this step, the image captured by the stereo depth camera, originally a 3-channel image, is converted to 16 channels. This is achieved via three successive 3x3 convolution layers with a stride of 2: first converting 3→4 channels, then 4→8, and finally 8→16 channels. The resulting image is then split into two paths: one goes to global feature extraction, the other to feature fusion.

2) Global Feature Extraction: This stage performs a depthwise-separable convolution to extract features from the 16-channel image. This method is chosen for its computational efficiency. The result is a 24-channel image, which is passed on to the upsampling stage.

3) Upsampling: This step performs pooling using average pooling, which is lighter and faster compared to the pyramid pooling used in the original Fast-SCNN. A convolution then produces a 32-channel output, which is passed to feature fusion.

4) Feature Fusion: Here, outputs from the downsampling and global feature extraction stages are merged into a 48-channel image. Before merging, image dimensions are matched using bilinear interpolation. The merged result is passed to the classification stage.

5) Classification: In this final stage, the 48-channel image undergoes classification using ReLU, producing a 2-channel image indicating road and non-road areas. This image then proceeds to post processing.

C. Pose and Orientation Correction System

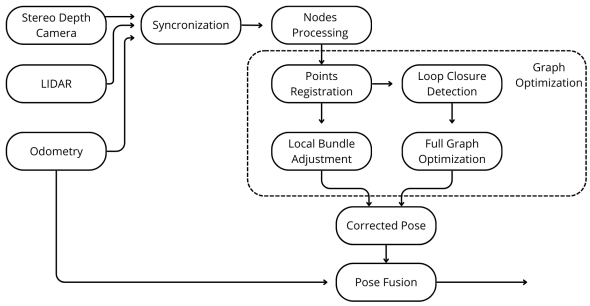

Fig. 6. Block Diagram of Pose and Orientation Correction System on Icar

From Figure 6, the pose correction system consists of four main processes: data synchronization, nodes processing, graph optimization, and pose fusion.

1) Data Synchronization: This initial step synchronizes data from the stereo depth camera and odometry based on timestamps to ensure aligned data for further processing.

2) Nodes Processing: This initiates the graph optimization process. Nodes represent vehicle poses and keypoints from stereo depth camera and LIDAR. There are two modes: - In mapping mode, new nodes are continuously added. - In localization mode, existing nodes are used without adding new ones.

3) Graph Optimization: This is the core component, using the GTSAM library. It starts with points registration using the Iterative Closest Point (ICP) algorithm to align point clouds, resulting in a transformation matrix.

Next are two procedures: - Local bundle adjustment optimizes surrounding nodes using the transformation, based on the Levenberg-Marquardt algorithm. - Loop closure finding that identifies revisited nodes. When a loop closure is found, a full graph optimization is performed, optimizing all nodes globally.

4) Pose Fusion: The final step combines poses from graph optimization and odometry using a Kalman Filter, resulting in more accurate pose estimation than using odometry alone. Using the differential data from Odometry (Wheel encoder and IMU) combined with estimated pose that coming from Graph Optimization, the system can correct the pose and orientation of Icar.

D. Icar Navigation System with Road Detection and Graph-Based SLAM

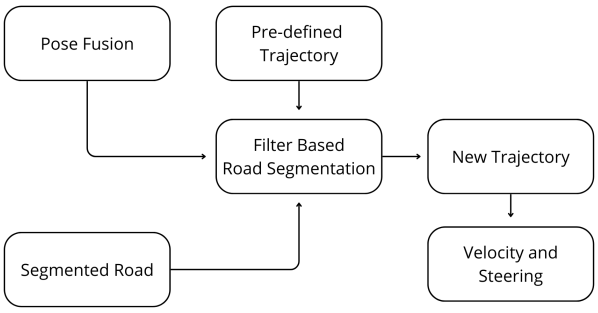

Fig. 7. Block Diagram of Icar Navigation System with Road Detection and Graph-Based SLAM

This diagram expands upon Figure 4, showing how the segmented road image from Figure 5 and the pose data from Figure 6 are integrated into a unified and improved navigation system.

The first step selects waypoints based on current pose (from graph-based SLAM) and the pre-defined trajectory. These waypoints are overlaid on the road segmentation image.

Next, a check is performed using a bitwise AND operation to determine if all waypoints lie within the road area. If so, the selected waypoints are used as the new trajectory.

If any waypoint lies outside the road, a new trajectory is generated using the image's center of mass, converted into vehicle coordinates. A straight line is then created from the current position to the center of mass.

This new trajectory is then used for vehicle navigation. The vehicle follows it using the Bicycle Model as defined in Equation 2 to calculate target speed and steering angle.

REFERENCES

[1] M. I. Zazuli, R. Dikairono, D. Purwanto, Muhtadin, and M. L. Hakim, "Sensor fusion system for localization of autonomous car," in *2024 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*, 2024, pp. 1–5.

[2] S. Khanafseh, B. Kujur, M. Joerger, T. Walter, S. Pullen, J. Blanch, K. Doherty, L. Norman, L. de Groot, and B. Pervan, "Gnss multipath error modeling for automotive applications," in *Proceedings of the 31st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2018)*. Miami, Florida: Institute of Navigation (ION), 2018, pp. 1573–1589.

[3] M. BROSSARD and S. BONNABEL, "Learning wheel odometry and imu errors for localization," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 291–297.

[4] P. S. Tantra, R. Dikairono, H. Kusuma, Muhtadin, and Tasripan, "Automated lidar-based dataset labelling method for road image segmentation in autonomous vehicles," in *2024 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*, 2024, pp. 1–5.

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

[6] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[7] A. C. Poudel, S. Liwicki, and R. Cipolla, "Fast-scnn: Fast semantic segmentation network," 2019.

[8] A. Woo, B. Fidan, W. Melek, and R. Buehrer, "Localization for autonomous driving," pp. 1051–1087, 01 2019.

[9] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," *Georgia Institute of Technology*, 2012, available at https://gtsam.org.

[10] R. Rajamani, *Vehicle Dynamics and Control*, 2nd ed. Boston, MA: Springer, 2011.

[11] B. Paden, M. Č. Câmara, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.