Thesis Proposal

AUTONOMOUS VEHICLE NAVIGATION SYSTEM USING MACHINE LEARNING AND GRAPH BASED SLAM

AZZAM WILDAN MAULANA
NRP 6022241101

SUPERVISOR
Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Ahmad Zaini, S.T., M.Sc.
Dr. Rudy Dikairono, S.T., M.T.

MASTER PROGRAM
DEPARTMENT OF ELECTRICAL ENGINEERING
FACULTY OF INTELLIGENT ELECTRICAL AND INFORMATICS TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2025

This page is intentionally left blank

VALIDITY SHEET
THESIS PROPOSAL

Judul	:	AUTONOMOUS VEHICLE NAVIGATION SYSTEM USING MACHINE LEARNING AND GRAPH BASED SLAM
Oleh	:	Azzam Wildan Maulana
Nrp	:	6022241101

Has Been Presented On

Day	:	Thursday
Date	:	26 June 2025
Place	:	B211 Room

Examiner

1. Dr. Arief Kurniawan, S.T., M.T.
   Nip :197409072002121001

2. Dr. Diah Puspito Wulandari, S.T., M.Sc.
   Nip :198012192005012001

3. Dr. Supeno Mardi Susiki Nugroho,S.T.,M.T.
   Nip :197003131995121001

Supervisor

1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
   Nip :195809161986011001

2. Ahmad Zaini, S.T., M.Sc.
   Nip :197504192002121003

3. Dr. Rudy Dikairono, S.T., M.T.
   Nip :198103252005011002

This page is intentionally left blank

PREFACE

Praise and gratitude to the presence of Allah SWT. for all His grace, the author was able to complete this research with the title **AUTONOMOUS VEHICLE NAVIGATION SYSTEM USING MACHINE LEARNING AND GRAPH BASED SLAM**. This research was written as a requirement for completing a master's degree at the Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember.
The author also expresses great thanks to:

- Parents and family for their encouragement and support during the author's study.
- One girl who has always been there for the author, providing support and motivation during the author's study.
- My friends especially AWM and The Dangereous Family for supporting the author during the study.

Surabaya, June 2025

The Author

This page is intentionally left blank

AUTONOMOUS VEHICLE NAVIGATION SYSTEM USING MACHINE LEARNING AND GRAPH BASED SLAM

By	:	Azzam Wildan Maulana
Student Identity Number	:	6022241101
Supervisor	:	1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
		2. Ahmad Zaini, S.T., M.Sc.
		3. Dr. Rudy Dikairono, S.T., M.T.

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.

Keyword: Autonomous vehicle, Icar, GNSS, Odometry, Pose Estimation, Stereo Depth Camera, Semantic Segmentation, SLAM, ICP, Graph-Based Optimization, Deep Learning

This page is intentionally left blank

TABLE OF CONTENTS

LIST OF FIGURES

This page is intentionally left blank

LIST OF TABLES

This page is intentionally left blank

NOMENKLATUR

Symbol / Acronym	Description
x_{waypoint}	X-coordinate of the waypoint from the predefined trajectory
y_{waypoint}	Y-coordinate of the waypoint from the predefined trajectory
x_{position}	Current X-coordinate of the vehicle
y_{position}	Current Y-coordinate of the vehicle
$\theta_{\text{direction}}$	Heading angle toward the waypoint
$\theta_{\text{orientation}}$	Current orientation of the vehicle
v_{target}	Target velocity of the vehicle
v_{max}	Maximum velocity allowed
δ_{target}	Target steering angle based on bicycle model
L	Distance between the front and rear wheels
$D_{\text{lookahead}}$	Lookahead distance for path tracking
GNSS	Global Navigation Satellite System
SLAM	Simultaneous Localization and Mapping
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
EKF	Extended Kalman Filter
GTSAM	Georgia Tech Smoothing and Mapping
DoF	Degrees of Freedom
RTK	Real-Time Kinematic (high-precision GNSS)
DGNSS	Differential GNSS
ML	Machine Learning
CNN	Convolutional Neural Network
Fast-SCNN	Fast Semantic Segmentation Convolutional Neural Network
ReLU	Rectified Linear Unit
SVD	Singular Value Decomposition
LIDAR	Light Detection and Ranging
PCL	Point Cloud Library
Stereo Depth Camera	A pair of cameras used to calculate depth via disparity
Odometry	Estimation of position based on wheel rotations and IMU data
Pose	A combination of position and orientation in 2D or 3D space
Yaw	Rotation around the vertical (z) axis
Kalman Filter	Recursive algorithm for optimal state estimation
Multipath	GNSS signal distortion caused by reflection or obstruction
Trajectory	A planned path for the robot to follow
Feature Matching	Process of aligning keypoints across frames or sensors

This page is intentionally left blank

BAB 1
INTRODUCTION

1.1 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 [22].

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 [8].

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 [1].

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 machine learning algorithm—specifically, semantic segmentation—trained on labeled datasets to distinguish between road and non-road areas [18].

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 [19], which is expected to potentially replace the role of GNSS entirely.

1.2 Formulation of the Problem

The main issue with Icar lies in its navigation system's heavy dependence on GNSS. When GNSS signal errors occur, or when odometry is affected by slippage or IMU drift, the resulting pose estimation becomes inaccurate. Consequently, Icar may deviate from its intended path, increasing the risk of collisions or deviation from public transportation routes.

1.3 Objective

The objective of this research is to develop a robust navigation system for Icar that is capable of correcting pose estimation errors caused by GNSS inaccuracies or odometry drift. This system incorporates a stereo depth camera and LIDAR to perform both road detection and pose correction. The goal is to enable Icar to navigate more accurately and reliably in dynamic environments.

1.4 Scope and Limitations

This study focuses exclusively on Icar, an autonomous vehicle developed by ITS. The scope is limited to the development and implementation of a navigation system that utilizes a stereo depth camera and LIDAR. This research does not cover the electronic or mechanical aspects of Icar's control system.

1.5 Contribution

The expected contribution of this research is to provide a reference framework for enhancing the accuracy and safety of autonomous vehicle navigation systems. By integrating advanced perception and localization technologies, Icar is expected to achieve improved trajectory tracking and obstacle avoidance, supporting its intended operation in complex urban environments.

This page is intentionally left blank

BAB 2

LITERATURE REVIEW

To support this research, several relevant theories are required as references and conceptual foundations. This will help ensure the research is well-directed and aligned with existing scientific knowledge.

2.1 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 [22].

2.2 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 [16]. In robotics and electrical engineering, the bicycle model is commonly used to develop navigation and motion control algorithms for autonomous ground vehicles.

$$\delta_{\text{target}} = \tan^{-1}\left(\frac{2 \cdot L \cdot \sin(\theta_{\text{direction}})}{D_{\text{lookahead}}}\right) \tag{2.1}$$

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 [14].

2.3 GNSS

The Global Navigation Satellite System (GNSS) is a satellite-based system that provides highly accurate global information on position, velocity, and time. GNSS encompasses multiple satellite navigation systems, including GPS (USA), GLONASS (Russia), Galileo (EU), and BeiDou (China), which can function independently or in combination [12]. In electrical engineering and robotics, GNSS plays a critical role in navigation and positioning, particularly for mobile robots operating in open environments.

GNSS operates by measuring the time required for satellite signals to reach a receiver. A GNSS receiver can compute a 3D position using signals from at least four satellites. Techniques such as Differential GNSS (DGNSS) and Real-Time Kinematic (RTK) improve accuracy by applying corrections from fixed reference stations [7]. These techniques are especially beneficial in applications like precision agriculture, drones, and autonomous vehicles.

GNSS can be integrated with other systems such as Inertial Navigation Systems (INS) to enhance reliability in environments where satellite signals are blocked, such as under tree canopies or between buildings [5]. Integration with sensors like IMUs, LiDARs, and cameras also enhances performance in SLAM systems.

Beyond navigation, GNSS contributes to control systems, trajectory planning, and multi-robot coordination. With ongoing advances in satellite infrastructure and error correction algorithms, GNSS remains a cornerstone for the development of efficient and precise outdoor robotic systems.

2.4 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.

The algorithm consists of two stages: prediction and update. During the prediction stage, the system state is estimated using a motion model. In the update stage, the estimate is refined based on new sensor measurements. This iterative process results in increasingly accurate state estimations [20].

2.5 SLAM

Simultaneous Localization and Mapping (SLAM) is a foundational technology in autonomous robotics. It allows a robot to construct a map of an unknown environment while simultaneously estimating its own position within that map. SLAM is particularly valuable in GPS-denied environments and is widely used in autonomous vehicles, drones, and mobile robots [2].

2.6 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 [19].

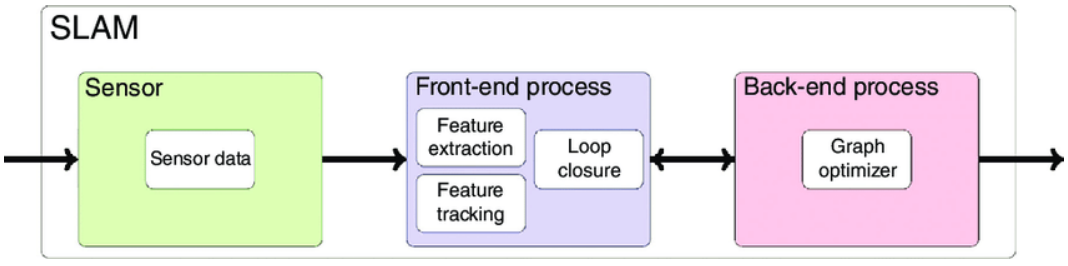

Figure 2.1. Basic Graph-Based SLAM Architecture [21]

There are three main process that can be seen from figure 2.1: 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 [21].

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 [3].

2.7 RTAB-Map

RTAB-Map (Real-Time Appearance-Based Mapping) is a graph-based SLAM framework designed for real-time applications in robotics. It integrates various sensors, including cameras, LIDARs, GPS, IMUs, and Odometry, to build a map of the environment while simultaneously localizing the robot within that map. RTAB-Map is particularly effective in large-scale environments and can handle loop closures and dynamic changes in the environment [10].

2.8 Stereo Depth Camera

A stereo depth camera is a vision-based depth sensing system that uses two calibrated cameras to capture two images from slightly different perspectives—similar to human binocular vision. By analyzing the disparity between corresponding points in the image pair, the system calculates the depth or distance of objects [17].

Stereo cameras are widely used in robotics due to their ability to generate real-time depth maps without requiring active sensors like LiDAR. They are especially useful in lightweight and power-efficient robotic platforms such as drones and small autonomous vehicles [9].

2.9 Machine Learning

Machine learning (ML) is a subset of artificial intelligence that focuses on creating algorithms capable of learning from data and improving performance without being explicitly programmed [13]. In robotics and electrical engineering, ML is essential for building intelligent systems that can recognize patterns, make decisions, and adapt to changing environments.

ML methods are typically categorized into three types: supervised learning, unsupervised learning, and reinforcement learning [4]. Supervised learning uses labeled datasets for tasks like classification and regression. Unsupervised learning deals with unlabeled data for clustering or dimensionality reduction. Reinforcement learning enables systems to learn optimal behaviors through reward-based interactions with the environment.

A key ML technique in robotic perception is semantic segmentation, which classifies each pixel in an image into predefined categories. This technique is widely used for road detection, scene understanding, and navigation.

2.10 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.

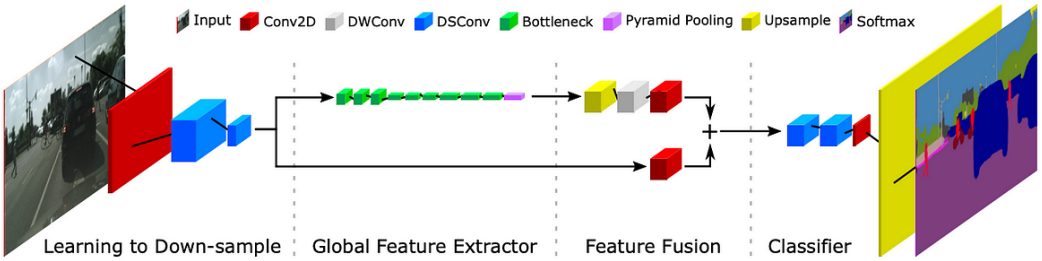

Figure 2.2. Fast SCNN Architecture [15]

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 [15].

2.11 Iterative Closest Point (ICP)

Iterative Closest Point (ICP) is a widely used algorithm in known-map localization that aligns two 3D point clouds: one from the robot's current observation and one from a reference map [23]. The output is a transformation matrix that minimizes the difference between the two point sets.

The ICP algorithm operates in three main steps: first, for each point in the input point cloud, it finds the nearest point in the reference point cloud (using nearest-neighbor search). Second, it calculates the centroids (center of mass) of both point clouds and aligns them via translation. Third, it applies rotation—typically using Singular Value Decomposition (SVD)—to best align the clouds.

These steps are repeated iteratively until convergence, defined by either reaching a maximum number of iterations or achieving a sufficiently low alignment error. The final result is the transformation matrix that aligns the

current observation with the reference map, enabling accurate pose estimation.

2.12 Binary Robust invariant scalable keypoints (BRISK)

BRISK Binary Robust invariant scalable keypoints is a real-time feature method that combines a novel scale-space FAST detector with a binary bit-string descriptor built from intensity-comparison sampling. Keypoints are detected across scales via a FAST-based pyramid, then each is described by a 512-bit string derived from Gaussian-smoothed intensity comparisons on concentric sampling rings, with an orientation normalization step for rotation invariance. Evaluated on standard benchmarks, BRISK matches the accuracy of heavy float descriptors like SURF and SIFT while running up to an order of magnitude faster, making it highly suited for resource-constrained, real-time vision applications [11].

This page is intentionally left blank

BAB 3

METHODOLOGY

3.1 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.

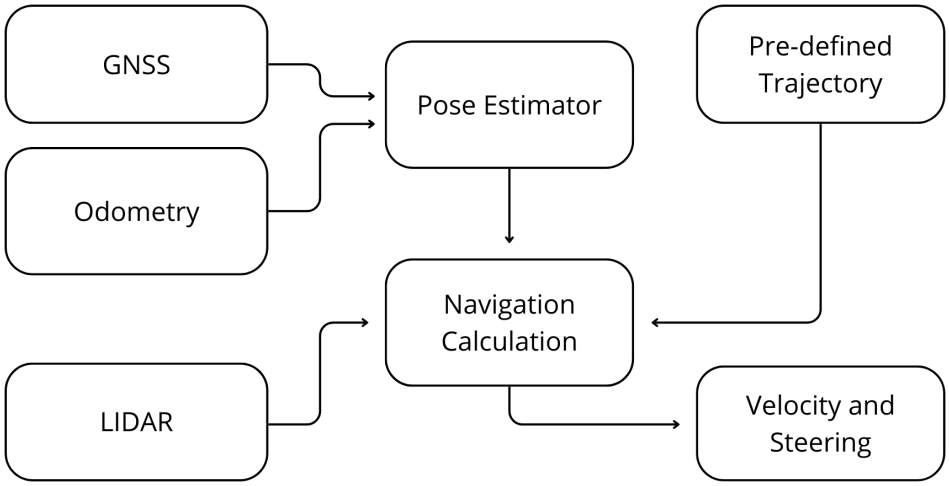

Figure 3.1. Block Diagram of the Legacy Icar System with GNSS

Figure 3.1 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 2.1.

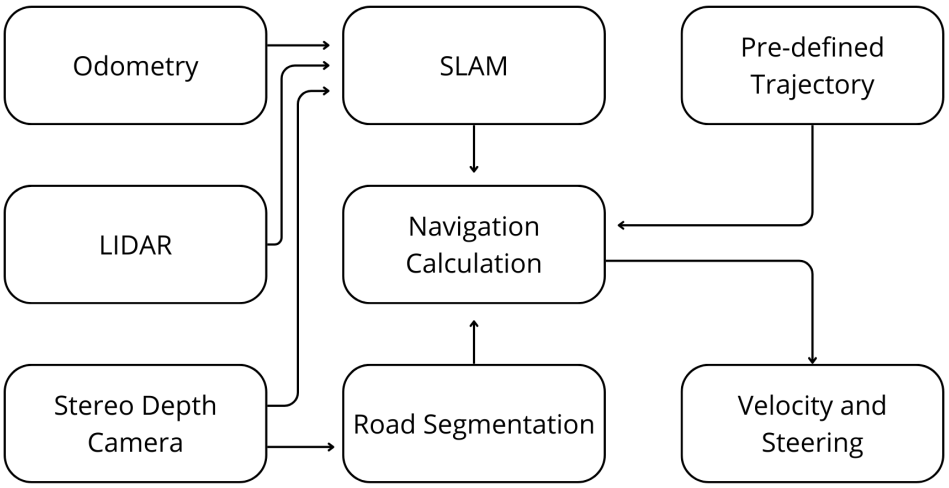

Figure 3.2. Block Diagram of the New Icar System without GNSS

Figure 3.2 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 [16], as shown in equation 2.1. The first step is finding the target waypoint angle using equation below.

$$\theta_{\text{direction}} = \tan^{-1}\left(\frac{y_{\text{waypoint}} - y_{\text{position}}}{x_{\text{waypoint}} - x_{\text{position}}}\right) - \theta_{\text{orientation}} \tag{3.1}$$

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, and $\theta_{\text{orientation}}$ is the current orientation of the vehicle. $\theta_{\text{direction}}$ is the heading angle towards the waypoint. With using that angle, we can calculate the target steering angle by using the Bicycle Model as follows:

$$\delta_{\text{target}} = \tan^{-1}\left(\frac{2 \cdot L \cdot \sin(\theta_{\text{direction}})}{D_{\text{lookahead}}}\right) \tag{3.2}$$

Where L is the distance between the front and rear wheels, and $D_{\text{lookahead}}$ is the lookahead distance. δ_{target} is the target steering angle. The target speed is calculated using the following equation:

$$v_{\text{target}} = \min\left(\sqrt{(y_{\text{waypoint}} - y_{\text{position}})^2 + (y_{\text{waypoint}} - y_{\text{position}})^2},\ v_{\max}\right) \tag{3.3}$$

Where $v_{\max}$ is the maximum speed of the vehicle. The target speed v_{target} is determined by the distance to the waypoint, ensuring it does not exceed the maximum speed.

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.

3.2 Road Detection System

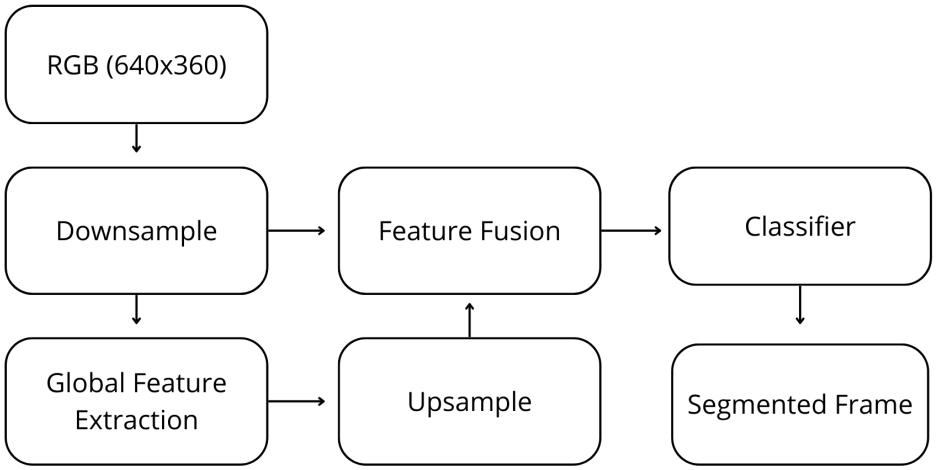

Figure 3.3. Block Diagram of the Machine Learning Architecture on Icar

From Figure 3.3, it is shown that the road detection system on Icar utilizes a machine learning architecture, specifically using Fast-SCNN [15] 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.

3.2.1 Downsampling

Figure 3.4 shows the raw image captured by the stereo depth camera. The image is then processed through a series of convolution layers to downsample it.

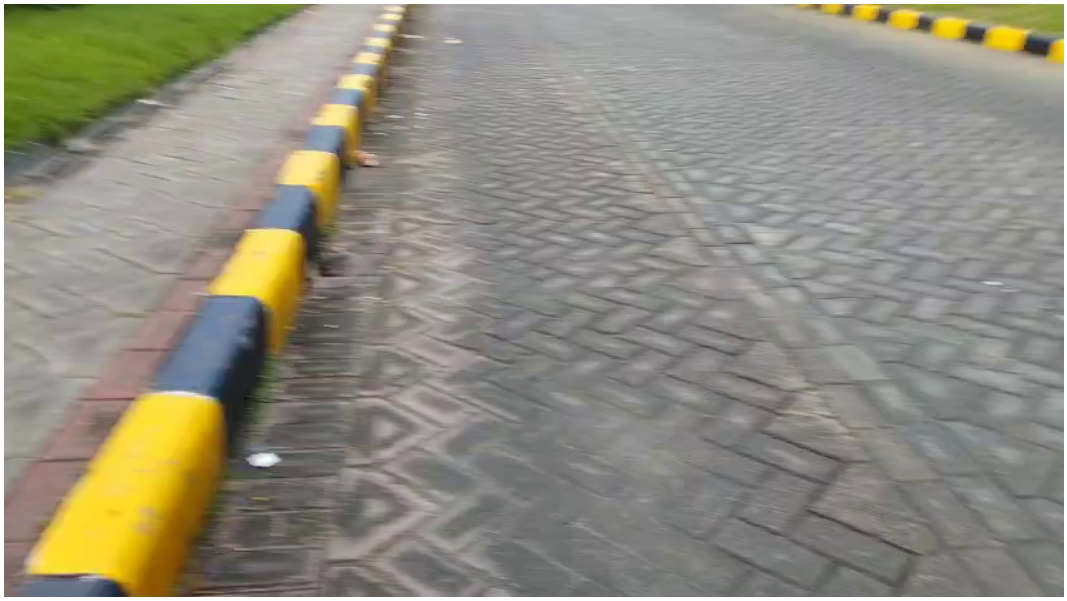

Figure 3.4. Raw Image

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→8 channels, then 8→16, and finally 16→32 channels. The resulting image is then split into two paths: one goes to global feature extraction, the other to feature fusion.

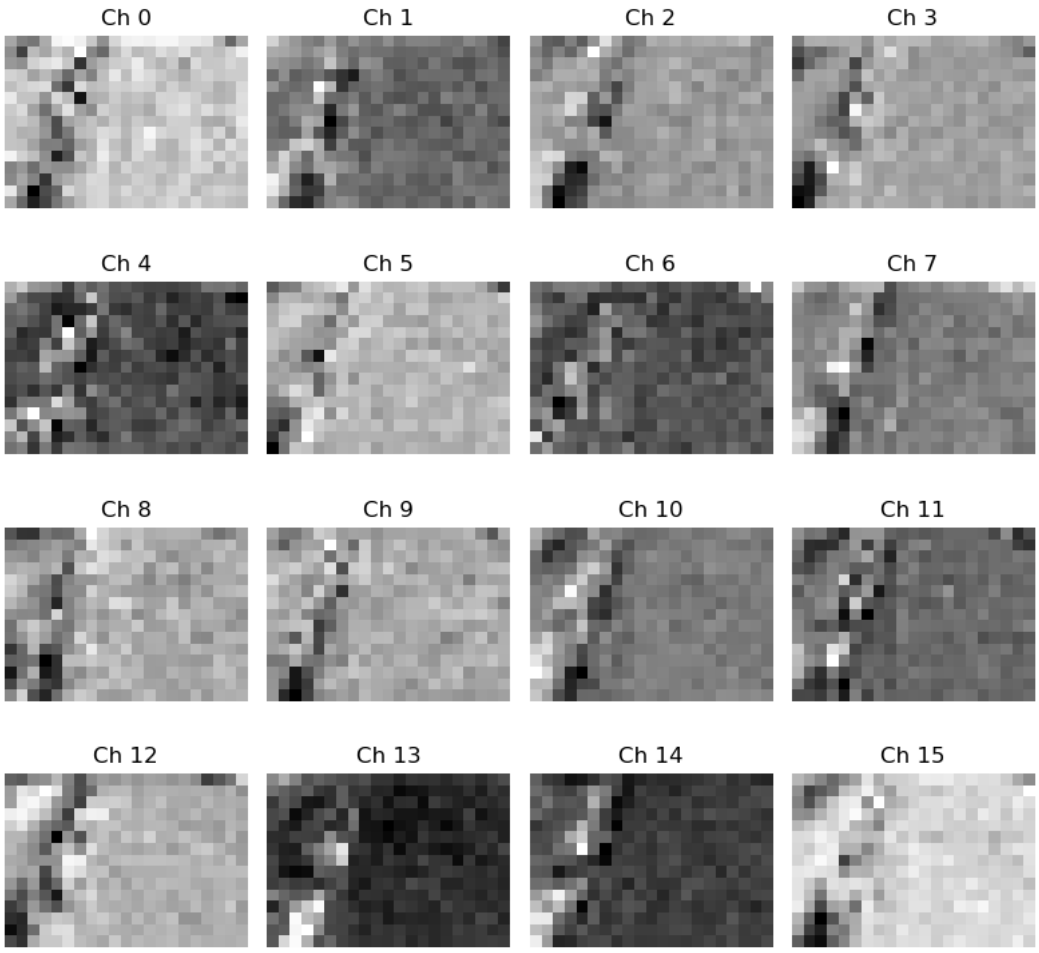

Figure 3.5. Result of Downsampling Stage

Figure 3.5 shows the result of the downsampling stage, where the original 3-channel image is transformed into a 32-channel image (but the figure only shows 16 channel because it was too much to show all 32 channels). This image is then processed further in the global feature extraction and upsampling stages.

3.2.2 Global Feature Extraction

This stage performs a depthwise-separable convolution to extract features from the 32-channel image. This method is chosen for its computational efficiency. The result is a 48-channel image, which is passed on to the upsampling stage.

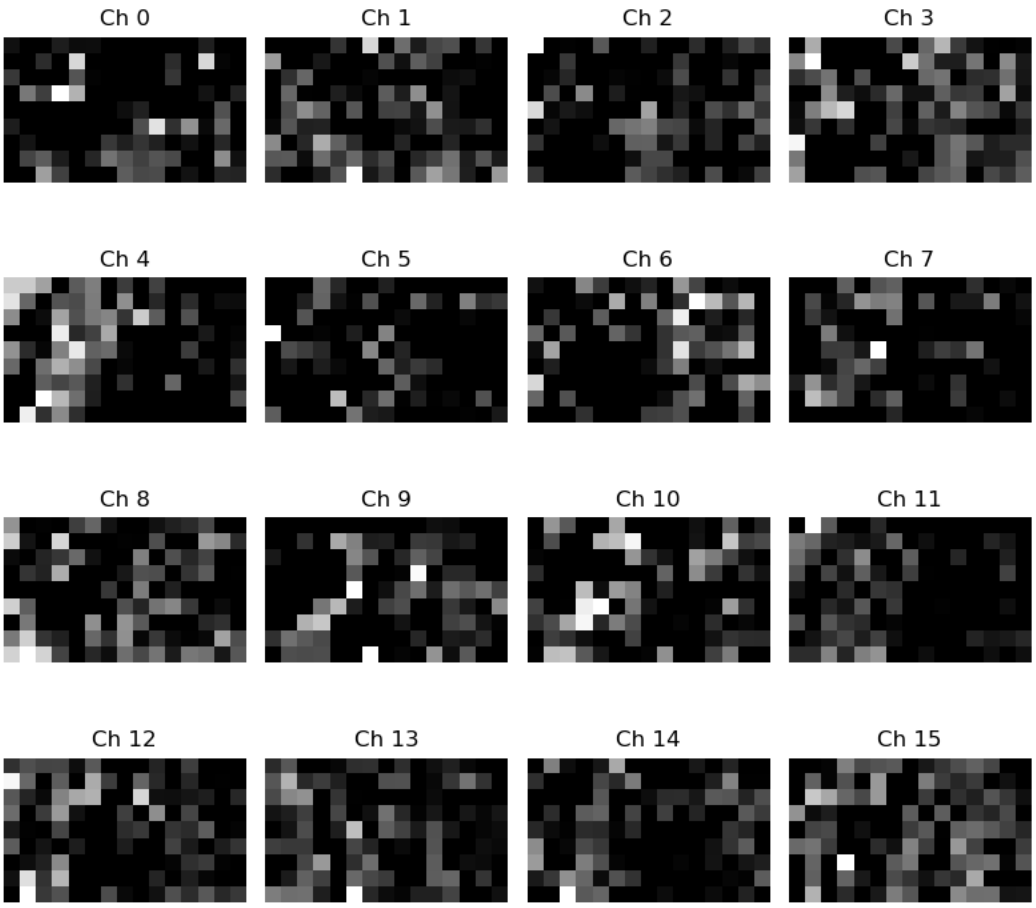

Figure 3.6. Result of Feature Extraction Stage

Figure 3.6 shows the result of the global feature extraction stage, where the 32-channel image is processed to produce a 48-channel image. This image contains the extracted features that will be used in the subsequent upsampling and feature fusion stages.

3.2.3 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 64-channel output, which is passed to feature fusion.

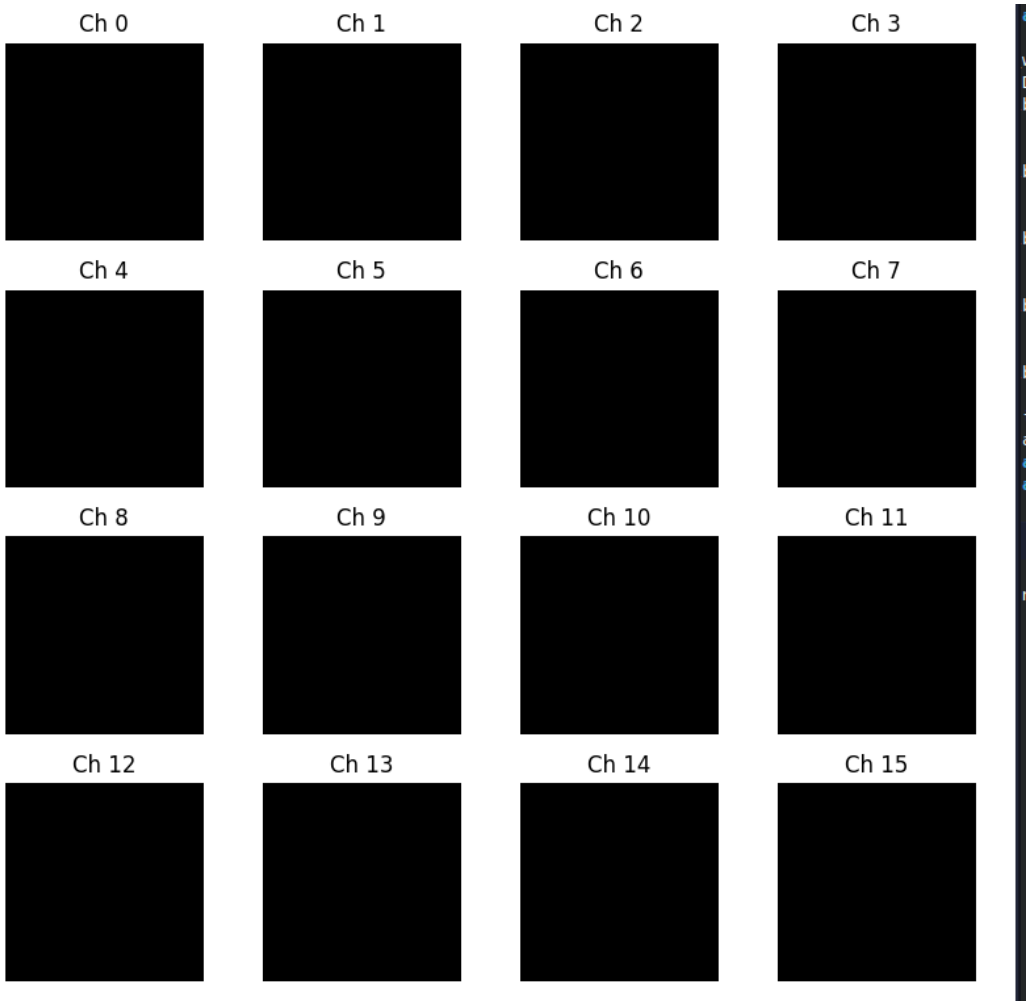

Figure 3.7. Result of Upsampling Stage

Figure 3.5 shows the result of the upsampling stage, where the 48-channel image from the global feature extraction stage is processed to produce a 64-channel image. This image is then ready for merging with the downsampled features in the feature fusion stage.

3.2.4 Feature Fusion

Here, outputs from the downsampling (32 channel) and global feature extraction (64 channel) stages are merged into a 96-channel image. Before merging, image dimensions are matched using bilinear interpolation. The merged result is passed to the classification stage.

3.2.5 Classification

In this final stage, the 96-channel image undergoes classification using ReLU, producing a 1-channel image indicating road areas. This image then proceeds to post processing.

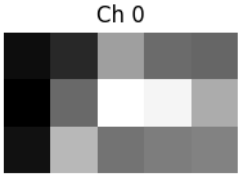

Figure 3.8. Result of Classification Stage

Figure 3.8 shows the result of the classification stage, where the 96-channel image from the feature fusion stage is processed to produce a 1-channel image.

3.3 Pose and Orientation Correction System

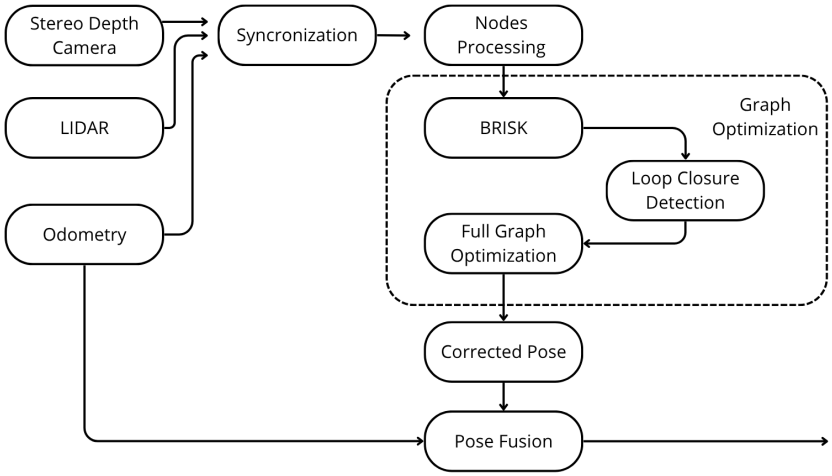

Figure 3.9. Block Diagram of Pose and Orientation Correction System on Icar

From Figure 3.9, the pose correction system consists of four main processes: data synchronization, nodes processing, graph optimization, and pose fusion.

3.3.1 Data Synchronization

This initial step synchronizes data from the stereo depth camera and odometry based on timestamps to ensure aligned data for further processing. The technique that used is the approximate time synchronization. Approximate time synchronization assume that the error timestamp below threshold like 10 milliseconds is eligible to be synchronized.

3.3.2 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 this process, the first mode is mapping mode. In mapping mode, the system creates new nodes for each pose, keypoint detected by the stereo depth camera and LIDAR laser scan while the robot has moved 1 meter away from the last node created. The second mode is localization mode, where existing nodes are used without adding new ones. This mode is used when the vehicle is already in a known area, and it needs to localize itself within that area without creating new nodes. Mapping mode only used once, while localization mode is used every time the vehicle is moving after finished mapping mode.

3.3.3 Graph Optimization

The first step is finding the loop closure by using BRISK algorithm [11]. BRISK will doing image matching between the current image and the previous saved image for each node. If the image matches, it will be considered as loop closure hypothesis. Then by using GTSAM [3], the optimization error will be calculated for all nodes by doing ICP [23] from the saved LIDAR laser scan for each node. If the optimization error less than the threshold (3.0), the loop closure will be accepted, and the graph will be optimized. The optimization process uses a factor graph approach, where each node represents a pose and each edge represents a constraint between poses. The optimization minimizes the error in pose estimation by adjusting the poses based on the constraints.

3.3.4 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.

3.4 Icar Navigation System with Road Detection and Graph-Based SLAM

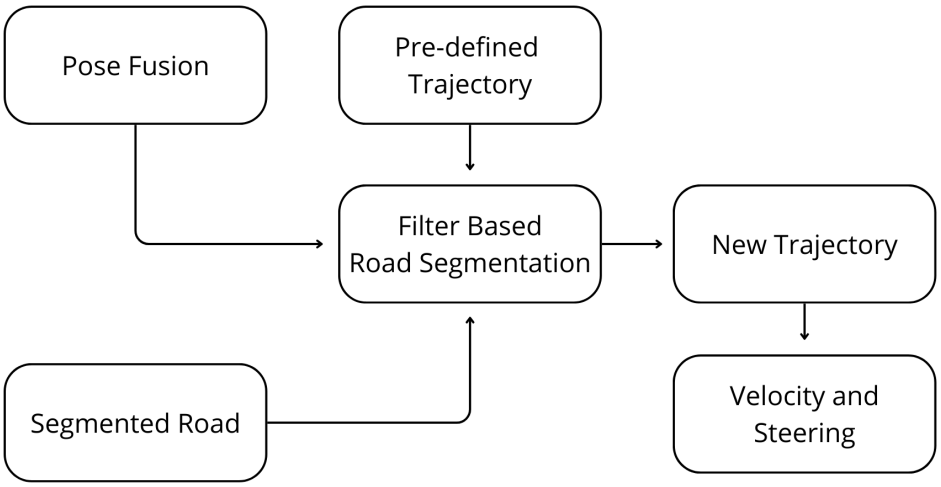

Figure 3.10. Block Diagram of Icar Navigation System with Road Detection and Graph-Based SLAM

This diagram expands upon Figure 3.2, showing how the segmented road image from Figure 3.3 and the pose data from Figure 3.9 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.

Figure 3.11. Visualization of the Road Detection and Graph-Based SLAM Results on Icar

Where the green mask is the segmented road, the blue line is the trajectory that coming from the pre-defined trajectory, the red line is the new trajectory that generated from this research. The vehicle will follow the red line as the new trajectory.

This new trajectory is then used for vehicle navigation. The vehicle follows it using the Bicycle Model as defined in Equation 3.2 and 3.3 to calculate target speed and steering angle.

This page is intentionally left blank

BAB 4

RESEARCH PLAN

Table 4.1. Research Schedule

Activities	Weeks-															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Literature Study																
Road Image Dataset Creation																
Road Segmentation Training and Test																
Map Creation for SLAM																
End to End Test																
Analysis																

This page is intentionally left blank

Bibliography

[1] Martin BROSSARD and Silvère BONNABEL. Learning wheel odometry and imu errors for localization. In *2*019 International Conference on Robotics and Automation (ICRA), pages 291–297, 2019.

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *I*EEE Transactions on Robotics, 32(6):1309–1332, 2016.

[3] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. *G*eorgia Institute of Technology, 2012. Available at `https://gtsam.org`.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *D*eep Learning. MIT Press, Cambridge, MA, 2016.

[5] Mohinder S. Grewal, Lawrence R. Weill, and Angus P. Andrews. *G*lobal Navigation Satellite Systems, Inertial Navigation, and Integration. John Wiley & Sons, Hoboken, NJ, 2nd edition, 2013.

[6] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *J*ournal of Basic Engineering, 82(1):35–45, 1960.

[7] Elliott D. Kaplan and Christopher J. Hegarty. *U*nderstanding GPS/GNSS: Principles and Applications. Artech House, Norwood, MA, 3rd edition, 2017.

[8] Samer Khanafseh, Birendra Kujur, Mathieu Joerger, Todd Walter, Sam Pullen, Juan Blanch, Kevin Doherty, Laura Norman, Lance de Groot, and Boris Pervan. Gnss multipath error modeling for automotive applications. In *P*roceedings of the 31st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2018), pages 1573–1589, Miami, Florida, 2018. Institute of Navigation (ION).

[9] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *S*ensors, 12(2):1437–1454, 2012.

[10] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. In *J*ournal of Field Robotics, volume 36, pages 416–446. Wiley Online Library, 2019.

[11] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary Robust Invariant Scalable Keypoints. In *2*011 IEEE International Conference on Computer Vision (ICCV), pages 2548–2555. IEEE, 2011.

[12] Pratap Misra and Per Enge. *G*lobal Positioning System: Signals, Measurements and Performance. Ganga-Jamuna Press, Lincoln, MA, 2006.

[13] Tom M. Mitchell. *M*achine Learning. McGraw-Hill, New York, 1997.

[14] Brian Paden, Michal Čáp Câmara, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *I*EEE Transactions on Intelligent Vehicles, 1(1):33–55, 2016.

[15] Arunava Chakraborty Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-scnn: Fast semantic segmentation network. 2019.

[16] Rajesh Rajamani. *V*ehicle Dynamics and Control. Springer, Boston, MA, 2nd edition, 2011.

[17] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *I*nternational Journal of Computer Vision, 47(1-3):7–42, 2002.

[18] Pandu Surya Tantra, Rudy Dikairono, Hendra Kusuma, Muhtadin, and Tasripan. Automated lidar-based dataset labelling method for road image segmentation in autonomous vehicles. In *2*024 International Conference on Green Energy, Computing and Sustainable Technology (GECOST), pages 1–5, 2024.

[19] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *P*robabilistic Robotics. MIT Press, Cambridge, MA, 2005.

[20] Greg Welch and Gary Bishop. An introduction to the kalman filter. `https://www.cs.unc.edu/~welch/kalman/`, 2006. Online tutorial, University of North Carolina at Chapel Hill.

[21] Ami Woo, Baris Fidan, W.W. Melek, and R. Buehrer. Localization for autonomous driving. pages 1051–1087, 01 2019.

[22] Moh. Ismarintan Zazuli, Rudy Dikairono, Djoko Purwanto, Muhtadin, and Muhammad Lukman Hakim. Sensor fusion system for localization of autonomous car. In *2*024 International Conference on Green Energy, Computing and Sustainable Technology (GECOST), pages 1–5, 2024.

[23] Juyong Zhang, Yuxin Yao, and Bailin Deng. Fast and robust iterative closest point. *I*EEE Transactions on Pattern Analysis and Machine Intelligence, 44(7):3450–3466, 2022.

AUTHOR BIODATA

Personal Information

Name	:	Azzam Widan Maulana
Place of Birth	:	Jember
Date of Birth	:	June 4, 2002
Address	:	Arif Rahman Hakim Street No. 14A, Keputih, Sukolilo, Surabaya

Educational Background

2024–Present	:	Master's Program (M.Sc.), Department of Electrical Engineering, Faculty of Intelligent Electrical and Informatics Technology, Institut Teknologi Sepuluh Nopember (ITS)
2020–2024	:	Bachelor Program (B.Eng.), Department of Computer Engineering, Faculty of Intelligent Electrical and Informatics Technology, Institut Teknologi Sepuluh Nopember (ITS)
2017–2020	:	SMAN 1 Jember
2014–2017	:	SMPN 3 Jember
2008–2014	:	SDN Kaliwates 2 Jember
2006–2008	:	TK Miftahul Ulum Kaliwates Jember

List of Publications

1. Muhtadin, A. W. Maulana, R. Dikairono and A. Zaini, "Omnivision Calibration on Mobile Robot Using Machine Learning," *2024 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, Surabaya, Indonesia, 2024, pp. 1–8, doi: 10.1109/CENIM64038.2024.10882712.

Research History

1. Multicast Communication System on IRIS Robot 2022
2. Natural Ball Handling System on IRIS Robot 2022
3. Omnivision Camera Calibration on Mobile Robot Using Machine Learning 2023
4. Robot Positioning System Using Gaussian Model on IRIS Robot 2023
5. Empty Goal Detection System Using Omnivision Camera on IRIS Robot 2023
6. Content Management System and User Interface for RAISA ITS Robot 2024
7. Custom Operating System for Autonomous Vehicle and IRIS Robot 2024
8. Empty Goal Detection System Using Stereo Depth Camera on IRIS Robot 2024
9. Ball Detection System Using Spatial Image Classification on IRIS Robot 2024
10. Hardware Communication System Using Layer 2 Communication for Autonomous Vehicle and IRIS Robot 2024
11. Hardware Communication System Using CAN-bus for Autonomous Vehicle and IRIS Robot 2024
12. End to End Machine Learning System for Autonomous Vehicle 2025
13. Fleet Management System for AMR (Autonomous Mobile Robot) 2025

Other Background

Before venturing into the world of robotics and programming, the author was involved in music. As a musician, he mastered various musical instruments such as guitar, bass, and keyboard. In addition to playing instruments, he was also active in the field of mixing and mastering. His hobby of signal processing sparked an interest in learning how computers process signals. This passion led him to pursue a degree in Computer Engineering at ITS. Moreover, he delved into robotics to understand how computers and hardware operate, enabling him to perform advanced digital signal processing techniques.