AUTONOMOUS VEHICLE NAVIGATION FOR HEAVY INDUSTRIAL TRANSPORT WITH HIGH ACCURACY AND PRECISION POSE USING GRAPH-BASED SLAM AND REAL-TIME MACHINE LEARNING

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*—Indonesia Smelting Technology (IST) is an industry below Toyota Tsusho Corp that focuses on producing aluminum liquid. The melted aluminum liquid will be used by CMWI (Central Motor Wheel Indonesia) to make the wheel rims. IST has a vision to become a smart factory that utilizes the latest technology in its operations. One of the technologies that can be implemented is autonomous vehicles, which can transport a dangereous aluminum liquid (molten) from IST to CMW. The total weight of the molten aluminum liquid is around 1.8 tons, and the total weight that the vehicle need to carry is 4.8 tons. Previously, IST has used a small train with a rail to transport the molten aluminum liquid, but this method has several limitations, such as the need for a rail infrastructure that very expensive and the limited flexibility of the vehicle. So, IST wants to implement an autonomous vehicle that can transport the molten aluminum liquid without the need for a rail infrastructure. The autonomous vehicle will be equipped with a navigation system that can detect the lane and obstacles, and can navigate autonomously to the destination. The navigation system will use a combination of machine learning and graph-based SLAM (Simultaneous Localization and Mapping) to achieve this goal. The machine learning algorithm will be used to detect the lane and obstacles, while the graph-based SLAM will be used to estimate the pose of the vehicle and create a map of the environment. The result of this research is a cm level error (below 1 cm) in robot pose estimation and a lightweight machine learning model that can run only on CPU with below 10 ms inference time.**

***Index Terms*—autonomous vehicle, navigation system, machine learning, graph-based SLAM, lane detection**

I. Background

Indonesia Smelting Technology (IST) is an industry below Toyota Tsusho Corp that focuses on producing aluminum liquid. The environment in IST is very dangereous because of the molten aluminum liquid that has a temperature of around 700 degrees Celsius. The production goes in 24 hours a day. The production team divided into three shifts. The research from [1] said that there will be more sleepy workers in the night shift. The sleepy workers will be more prone to accidents even if doing micro sleep for 1-2 seconds. The industry (IST) said there were much accidents in the night shift, such as workers grazed to nearby pole or wall and the spilled of the molten aluminum liquid that been carried by the workers. By many accidents, the industry (IST) wants to implement an autonomous vehicle that can transport the molten aluminum liquid from IST to CMWI (Central Motor Wheel Indonesia).

For the vehicle itself, there is a Towing that previously operated by the workers. The goal of my team is to modify that Towing to be fully autonomous. To design the autonomous vehicle for the harsh environment, the vehicle need to be robust on its hardware and software. The hardware need to be always on and can handle the harsh environment such is high temperature, high conductive dust level, and high vibration. In the other hand, the software need to be low power, low latency, but has high accuracy.

For the high accuracy, we choose the Graph Based SLAM (Simultaneous Localization and Mapping) as the main algorithm to estimate the pose of the vehicle. There is a framework to do that called RTAB-Map. The RTAB-Map is a graph-based SLAM that can handle large scale environment and can run on low power hardware. The RTAB-Map can also handle the loop closure detection, which is very important for the autonomous vehicle to navigate in the environment. The RTAB-Map can also handle the dynamic environment, which is very important for the autonomous vehicle to navigate in the environment. The RTAB-Map can also handle the large scale environment, which is very important for the autonomous vehicle to navigate in the environment [2].

For the low power and low latency of lane detection, we modify the Fast-SCNN model [3] to be more lightweight and can run on CPU. Fast-SCNN is a semantic segmentation model that use a skip connection inside its network architecture to get the spatial information from the input image. But the Fast-SCNN model is still too heavy to run on CPU, so we modify the model to be more lightweight.

II. Methodology

The methodology of this research is divided into four parts. First part is the pose estimation using graph-based SLAM, the third part is the lane detection using machine learning, and the last part is the integration of the navigation system and safety system.

A. Pose Estimation using Graph-Based SLAM

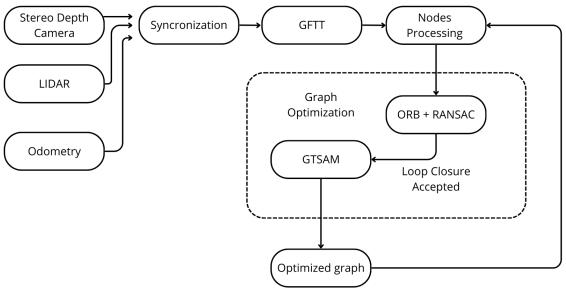

Fig. 1. Block Diagram of Graph Based SLAM in Mapping Mode

From the figure 1, we can see that the navigation system used on Autonomous Towing. The first step is calculate the odometry from the Towing. The odometry is calculated from the encoder speed and IMU data. By using inverse kinematics from encoder speed and IMU data, we can calculate the speed of Towing.

$$v\theta = \frac{\text{current\_IMUyaw} - \text{previous\_IMUyaw}}{\Delta t} \tag{1}$$

Where Δt is the time difference between the current and previous IMU data. The current_IMUyaw and previous_IMUyaw are the yaw angle of the IMU in radians. $v\theta$ is the angular velocity of the Towing in radians per second in yaw component.

$$v_x = \frac{\text{encoder\_speed} \cdot \cos(\theta) \cdot \text{encoder\_to\_meter}}{\Delta t} \tag{2}$$

Where encoder_speed is the speed of the Towing from the encoder in RPM, θ is the yaw angle of the Towing in radians, and encoder_to_meter is the conversion factor from encoder speed to meter per second. The Δt is the time difference between the current and previous encoder data. v_x is the linear velocity of the Towing in meter per second in x component.

$$v_y = \frac{\text{encoder\_speed} \cdot \sin(\theta) \cdot \text{encoder\_to\_meter}}{\Delta t} \tag{3}$$

Where encoder_speed is the speed of the Towing from the encoder in RPM, θ is the yaw angle of the Towing in radians, and encoder_to_meter is the conversion factor from encoder speed to meter per second. The Δt is the time difference between the current and previous encoder data. v_y is the linear velocity of the Towing in meter per second in y component.

The second step is data acquisition and data syncronization. This can be done by using a routine timer that grab the data with same stamped time. The technique called approximate time syncronization with the delay tolerance about 10 ms.

The third step is Nodes processing, which is the process of creating a node from the data acquired. There is a bit different in Nodes processing between Mapping mode and localization mode. In mapping mode, The nodes will always be updated. In localization mode, the nodes will not be updated. Each node in the graph contains:

- **Pose**: The pose of the Towing in the world frame that come from integration of odometry.
- **Optimized Pose**: The pose of the Towing after optimized.
- **Sensor Data**: The sensor data from the camera and lidars.
- **Timestamp**: The timestamp of the node.
- **Keypoint**: The result of the GFTT features.
- **Weight**: Weight for graph optimization (GTSAM).

Next step is ORB + RANSAC process, while the Towing has moved to map the area, the ORB process will find the matched keypoint from current synchronized camera data to all the saved node. After ORB found some matched keypoint, the next process is RANSAC filter. The RANSAC filter will filter the matched keypoint geometrically [4]. The output of RANSAC filter is the inliers between two image. If the inliers is more than the mininimum thershold, the loop closure hypothesis will be added. Below is the result example of ORB + RANSAC.

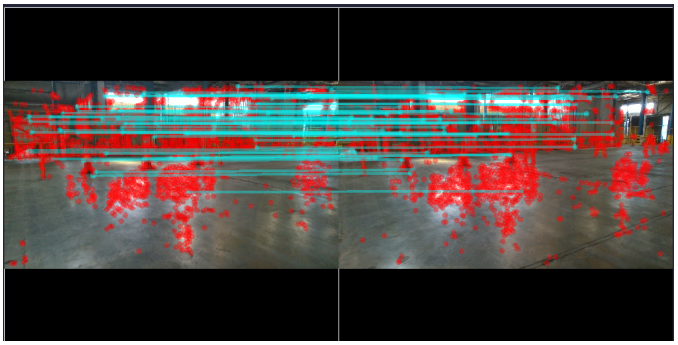

Fig. 2. Result of ORB + RANSAC

The figure 2 shows the result of ORB + RANSAC process. The left image is the current synchronized camera data, and the right image is the saved node. The cyan lines are the matched keypoint between two images. The red points are the features detected by GFTT (Good Features to Track) algorithm [5].

After loop closure hypothesis is added, the next step is to optimize the graph. The optimization process will use GTSAM library [6] to optimize the graph. The optimization process will use the ICP as the constraint between the nodes. All the nodes link constraint calculated using ICP from the lidar laser scan. If the iterations for optimizing has reached the maximum iterations, the optimization process will stop and the minimum error will be noted. If the minimum error is below the threshold, the loop closure will be accepted and the graph will be updated. The optimized pose will be calculated for every node and updated for all nodes in the graph. The optimized pose will be used for the next localization process. Below is the example of the map that has been built with and without loop closure found.

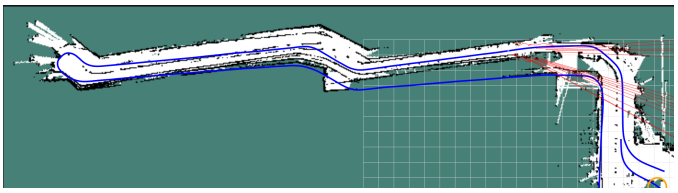

Fig. 3. Map of the area without loop closure found.

The figure 3 shows the map of the area with and without loop closure found. We can see that the map has not match the real world environment because of the drift odometry error. After loop closure has been found and all the graph were optimized, the map will be updated. Below is the example of the map that has been built with loop closure found.

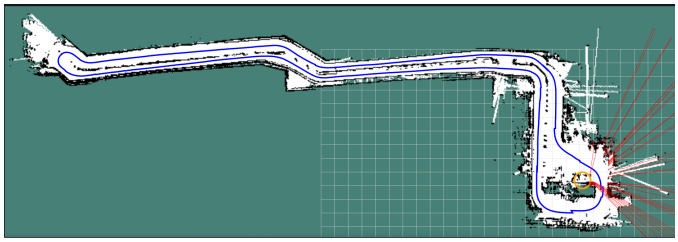

Fig. 4. Map of the area with loop closure found.

The figure 4 shows the map of the area with loop closure found. We can see that the map has been corrected and match the real world environment. The optimized pose will be used for the next localization process.

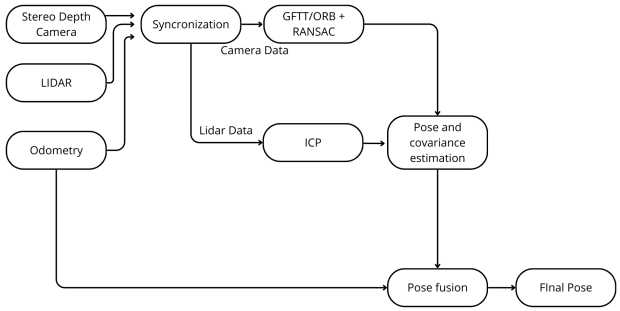

Fig. 5. Block Diagram of Graph Based SLAM in Localization Mode

In the localization mode, the process is almost the same as in mapping mode. The difference is that the nodes will not be updated, and the optimized pose will be used for localization. The localization process will use the current synchronized camera data to find the matched keypoint to all the saved node. After ORB found some matched keypoint, the next process is RANSAC filter. The RANSAC filter will filter the matched keypoint geometrically. The output of RANSAC filter is the inliers between two image. If the inliers is more than the minimum threshold, the corrected pose and covariance will be calculated using ICP constraint.

After the corrected pose and covariance is calculated, the next step is the pose fusion between corrected pose and covariance with the raw odometry pose and covariance. The raw odometry is coming from equation 1, 2, and 3. The pose fusion will use the Kalman Filter to fuse the corrected pose and covariance with the raw odometry pose and covariance. The output of the pose fusion is the final pose that will be used in navigation system.

B. Lane Detection using Machine Learning

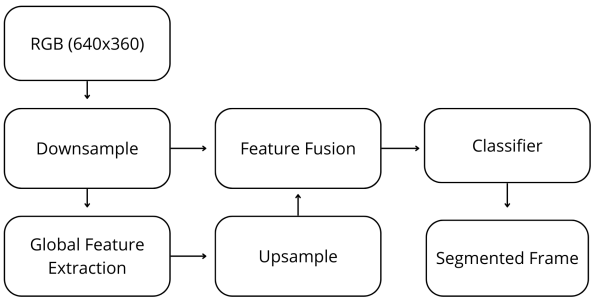

Fig. 6. Block Diagram of the Machine Learning Architecture

From Figure 6, it is shown that the lane detection system on Autonomous Towing utilizes a machine learning architecture, specifically using Fast-SCNN [3] as the core model. Our 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 Autonomous Towing.

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→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.

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.

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.

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.

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.

C. Integration of Navigation System and Safety System

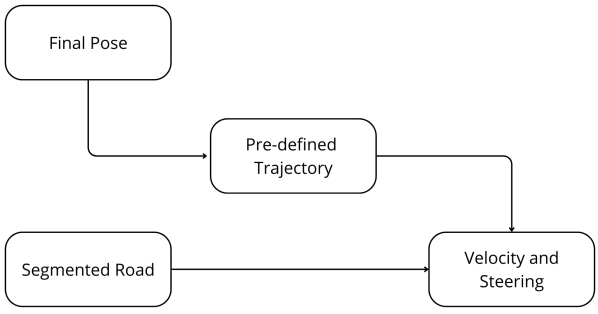

Fig. 7. Block Diagram Navigation System with Lane Detection and Graph-Based SLAM

The figure 7 shows the integration of the navigation system with lane detection and graph-based SLAM. The first step is acquire velocity and steering using Final Pose and Pre-defined Trajectory. This can be done using step below:

$$\theta_{\text{direction}} = \tan^{-1}\left(\frac{y_{\text{wp}} - y_{\text{position}}}{x_{\text{wp}} - x_{\text{position}}}\right) - \theta_{\text{orientation}} \tag{4}$$

Where x_{wp} and y_{wp} 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:

$$\delta_{\text{target}} = \tan^{-1}\left(\frac{2 \cdot L \cdot \sin(\theta_{\text{direction}})}{D_{\text{lookahead}}}\right) \tag{5}$$

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:

$$v_{\text{target}} = \min\left(\sqrt{(y_{\text{wp}} - y_{\text{position}})^2 + (y_{\text{wp}} - y_{\text{position}})^2},\ v_{\text{max}}\right) \tag{6}$$

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 next step is to calculate the confidence level from segmented lane detection. Before we calculate the confidence level, we need to do ROI (Region of Interest) calculate. First, we create a Circle ROI at center above of the image. The next step is to do bitwise AND between the segmented lane detection and the circle of ROI. After that, we can calculate the normalization of area by divide the final segmented area with maximum ROI.

$$\text{normal} = \frac{\text{area of segmented lane detection}}{\text{area of ROI}} \tag{7}$$

Where area of segmented lane detection is the area of the segmented lane detection image, and area of ROI is the area of the ROI image. The normal is the normalization value between 0 to 1. The next step is to calculate the confidence speed based on the normalization value. The confidence speed is calculated by multiplying the maximum speed with the square of the normalization value. The formula is as follows:

$$v_{\text{target\_max}} = v_{\text{max}} \cdot \text{normal}^2 \tag{8}$$

Where $v_{\text{target\_max}}$ is the maximum target speed based on the confidence level, and v_{max} is the maximum speed of the vehicle. The confidence speed will be used to limit the target speed of the vehicle. To get more smooth transition, the simple low pass filter was used to get the final target speed. The low pass filter is calculated using the following equation:

$$v_{\text{target}} = \alpha \cdot v_{\text{target\_max}} + (1 - \alpha) \cdot v_{\text{target}} \tag{9}$$

Where α is the low pass filter coefficient (currently we use 0.055), $v_{\text{target\_max}}$ is the maximum target speed based on the confidence level, and v_{target} is the previous target speed. The final target speed will be used to control the speed of the vehicle. The use of the low pass filter can prevent the sudden change of the target speed because of image flickering or noise.

The last step is to combine it, The target velocity speed from equation 6 will be clipped with the final target speed from equation 9. The target steering angle from equation 5 will be used to control the steering of the vehicle. The final target velocity will be sent to the motor driver as a Volt signal and the final target steering angle will be sent to the EPS (Electronic Power Steering) using CAN bus.

III. Results and Discussion

There were several results that we got from this research. The first result is the cm level error (below 1 cm) in robot pose estimation. The second result is the fast lightweight and near accurate lane detection. The third result is the total time to transport the molten aluminum liquid from IST to CMWI.

A. Robot Pose Estimation

There is the result on pose estimation without SLAM. The result is comparing the odometry before and after 500 meters long path. The result is shown in the table I.

TABLE I
Pose Estimation Result without SLAM

Lap-	X (m)	Y (m)	Error (m)
1	0.26	0.37	0.45
2	0.33	0.45	0.56
3	0.27	0.44	0.52
4	0.15	0.35	0.38
5	0.35	0.42	0.55
Average Error			0.492

Where the **Lap-** is the lap number, **X (m)** and **Y (m)** are the position of the Towing in meter after 500 meter moving, and **Error (m)** is the error of the pose estimation in meter. The error calculated using the ground truth zero condition (x = 0 and y = 0) The average error is 0.492 meters or 49.2 cm. This is not good enough for autonomous navigation, so we need to use SLAM to improve the pose estimation. There is the result on pose estimation using SLAM. The result is shown in the table II.

TABLE II
Pose Estimation Result with SLAM

Lap-	X (m)	Y (m)	Error (m)
1	0.02	0.01	0.02
2	0.00	0.03	0.03
3	0.01	0.02	0.02
4	0.01	0.01	0.01
5	0.02	0.04	0.04
Average Error			0.024

Where the **Lap-** is the lap number, **X (m)** and **Y (m)** are the position of the Towing in meter after 500 meter moving, and **Error (m)** is the error of the pose estimation in meter. The error calculated using the ground truth zero condition (x = 0 and y = 0) The average error is 0.024 meters or 2.4 cm. This is excellent enough for autonomous navigation, so we can conclude that our architecture in graph based SLAM is working well.

B. Lane Detection

There are the result from the lane detection using machine learning. The result is shown in the figure 8.

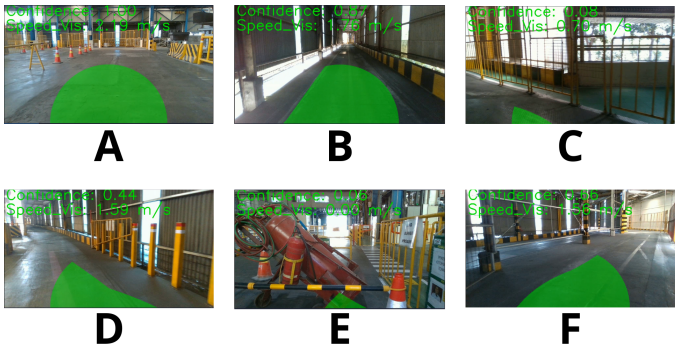

Fig. 8. Lane Detection Result using Machine Learning

The figure 8 shows the results of lane detection using machine learning. The image A is the condition where Towing at full straight lane, the output velocity is 2.2 meter per seconds. Image B is the condition where Towing is inside narrow straight lane, the output velocity is 1.76 meter per seconds. The image C is the condition where Towing in sharp turn and facing the fences, the output velocity only 0.70 meter per seconds. The image D is the condition where Towing going to turn left with narrow area, the output velocity is 1.59 meter per seconds. The image E is the condition where Towing cannot move because there is an obstacle in the front of Towing, the velocity is 0.00 meter per seconds. The last

image F is where Towing goinf turn right with narrow area, the output velocity is 1.56 meter per seconds.

Another result is the inference time for one frame of lane detection. The result is shown in the table III.

TABLE III
LANE DETECTION INFERENCE TIME

Frame Number	Inference Time (ms)
1	7.2
2	7.7
3	7.8
4	7.7
5	7.4
6	8.0
Averate	7.63

Table III shows the inference time for lane detection using machine learning. The average inference time is 7.63 ms, which is fast enough to run on CPU with low power usage. This is very important for autonomous navigation because the lane detection need to be fast and accurate to detect the lane and obstacles in real-time. That can be obtained by using onnx runtime that can run on CPU [7].

C. Total Time to Transport the Molten Aluminum Liquid

There is the result on total time to transport the molten aluminum liquid from IST to CMWI. The result is shown in the table IV.

TABLE IV
TOTAL TIME TO TRANSPORT THE MOLTEN ALUMINUM LIQUID (AUTONOMOUS TOWING)

Lap-	Time taken (Minutes)	Molten Sent (Ton)
1	6.59	1.8
2	7.03	1.8
3	7.06	1.8
4	7.08	1.8
5	7.02	1.8
6	7.03	1.8
Total	42.21	10.8

There is another result on total time to transport the molten aluminum liquid from IST to CMWI with using Manual Drivable Towing.

TABLE V
TOTAL TIME TO TRANSPORT THE MOLTEN ALUMINUM LIQUID (MANUAL TOWING)

Lap-	Time taken (Minutes)	Molten Sent (Ton)
1	3.29	1
2	4.23	1
3	4.60	1
4	4.56	1
5	3.52	1
6	3.40	1
Total	28.20	6

From table IV and V, we can see that the total time to transport the molten aluminum liquid from IST to CMWI using Autonomous Towing is 42.21 minutes for 6 laps with total 10.8 tons of molten aluminum liquid sent, while using Manual Towing is 28.20 minutes for 6 laps with total 6 tons of molten aluminum liquid sent. The Autonomous Towing has a longer time because it has to navigate autonomously and detect the lane and obstacles, while the Manual Towing is driven manually by the workers. If we calculate the average time per ton, the Autonomous Towing has 3.91 minutes per ton, while the Manual Towing has 4.70 minutes per ton. This shows that the Autonomous Towing is more efficient in terms of time per ton sent. The 1.8 Tons of molten aluminum liquid can be sent using Autonomous Towing because there will be high risk if the molten sent by manual Towing that drive by the workers.

IV. CONCLUSION

In this research, we have successfully implemented a full autonomous system on the Towing to transport the molten aluminum liquid from IST to CMWI. The system is robust and can handle the harsh environment in IST. The system can also navigate autonomously using the graph-based SLAM and lane detection using machine learning. The system can also detect the lane and obstacles, and can navigate autonomously to the destination. The result of this research is a cm level error (below 1 cm) in robot pose estimation and a lightweight machine learning model that can run only on CPU with below 10 ms inference time.

V. ACKNOWLEDGMENT

This research is supported by Indonesia Smelting Technology (IST), Manufaktur Robot Industri (MRI), and ITS Robotics. We would like to thank the people who have helped us in this research, especially the team from IST and MRI. We also would like to thank the ITS Robotics team for their support in this research.

REFERENCES

[1] R. Kazemi, R. Haidarimoghadam, M. Motamedzadeh, R. Golmohammadi, A. Soltanian, and M. R. Zoghipaydar, "Effects of shift work on cognitive performance, sleep quality, and sleepiness among petrochemical control room operators," *Journal of Circadian Rhythms*, vol. 14, p. 1, 2016. [Online]. Available: https://jcircadianrhythms.com/articles/10.5334/jcr.134

[2] M. Labbé 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," in *Journal of Field Robotics*, vol. 36, no. 2. Wiley Online Library, 2019, pp. 416–446.

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

[4] A. Vinay, A. S. Rao, V. S. Shekhar, A. C. Kumar, and K. N. B. Murthy, "Feature extraction using orb-ransac for face recognition," *Procedia Computer Science*, vol. 70, pp. 174–184, 2015, presented at 4th International Conference on Eco-friendly Computing and Communication Systems (ICECCS 2015).

[5] S. Dong, J. Li, U. A. Bhatti, J. Ma, F. Dong, and Y. Li, "Robust zero-watermarking algorithm for medical images based on gftt-kaze and dct," in *2023 26th ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter)*, 2023, pp. 290–297.

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

[7] "Faster scalable ml model deployment using onnx and open source tools," in *2020 IEEE Infrastructure Conference*, 2020, pp. i–i.