

BAB II

DASAR TEORI

2.1. Tinjauan Pustaka

Tinjauan pustaka dilakukan untuk mengetahui perkembangan penelitian yang berkaitan dengan sistem navigasi robot mobile, khususnya penggunaan LiDAR, SLAM, perencanaan jalur, dan sistem navigasi berbasis *Robot Operating System* (ROS). Beberapa penelitian sebelumnya telah menunjukkan bahwa LiDAR dapat digunakan untuk memperoleh informasi lingkungan dan mendukung proses pemetaan serta navigasi robot. Selain itu, penggunaan SLAM memungkinkan robot melakukan pemetaan dan mengetahui posisi secara bersamaan, sedangkan metode perencanaan jalur digunakan untuk menentukan rute robot menuju titik tujuan. Berdasarkan penelitian terdahulu, penelitian ini mengembangkan sistem navigasi pada AGV forklift dengan mengintegrasikan LiDAR 2D, SLAM, lokalisasi menggunakan AMCL, serta perencanaan jalur global menggunakan Hybrid A* melalui Nav2 SmacPlannerHybrid. Pengembangan tersebut dilakukan agar sistem navigasi dapat disesuaikan dengan karakteristik gerak AGV forklift dan dapat diterapkan secara langsung pada perangkat AGV. Empat penelitian yang dikaji beserta hasil dan perbedaannya terhadap penelitian ini disajikan pada Tabel 2.1.

Tabel 2. 1 Tinjauan Pustaka

No	Penelitian dan Judul	Hasil Penelitan	Perbedaan
1	<i>“Correlation Filter of 2D Laser Scans for Indoor Environment.</i> (Krinkin & Filatov 2021)	Penelitian mengembangkan metode correlation filter untuk mengurangi data pemindaian LiDAR 2D yang redundan sebelum diteruskan ke algoritma SLAM. Pengujian menggunakan dataset MIT dan TUM menunjukkan bahwa lebih dari setengah laser scan dapat dieliminasi ketika tidak membawa	Penelitian berfokus pada penyaringan laser scan LiDAR 2D untuk mengurangi data redundan dan waktu pemrosesan sebelum data digunakan oleh algoritma SLAM, sehingga belum membahas lokalisasi robot menggunakan peta statis serta

No	Penelitian dan Judul	Hasil Penelitian	Perbedaan
		informasi lingkungan baru. Metode tersebut juga mampu mengurangi rata-rata waktu pemrosesan scan lebih dari 40%, sehingga beban komputasi SLAM pada robot indoor dapat dikurangi	pengujian navigasi robot dari posisi awal menuju titik tujuan.
2	<i>Indoor Localization of Mobile Robots Based on the Fusion of an Improved AMCL Algorithm and a Collision Algorithm</i> (Zhu & Luo, 2024)	Penelitian mengembangkan metode lokalisasi indoor dengan mengintegrasikan Adaptive Monte Carlo Localization (AMCL) dan pembaruan semantic building map secara real-time. Objek nonstruktural seperti pintu, perabot, dan peralatan yang terdeteksi digunakan untuk memperbarui kondisi peta sehingga perubahan lingkungan tidak menurunkan kualitas estimasi posisi robot. Pengujian pada kondisi lokalisasi normal dan kidnapped robot problem menunjukkan peningkatan akurasi dan ketahanan estimasi pose dibandingkan penggunaan peta yang tidak diperbarui.	Penelitian berfokus pada peningkatan ketahanan <i>Adaptive Monte Carlo Localization</i> (AMCL) melalui pembaruan semantic building map secara real-time untuk menghadapi perubahan objek nonstruktural pada lingkungan indoor, sehingga belum membahas perencanaan jalur global dan pengujian kemampuan robot dalam mengikuti lintasan menuju goal pose.
3	<i>High-Efficiency Navigation of Nonholonomic</i>	Penelitian mengembangkan algoritma Improved Hybrid A* untuk meningkatkan efisiensi	Penelitian berfokus pada pengembangan Improved Hybrid A* untuk meningkatkan

No	Penelitian dan Judul	Hasil Penelitian	Perbedaan
	<i>Mobile Robots Based on Improved Hybrid A* Algorithm</i> (Chi et al. 2023)	perencanaan jalur robot mobile nonholonomik berbasis ROS. Untuk navigasi lokal dan penghindaran rintangan digunakan metode Model Predictive Control (MPC). Dibandingkan metode A* + DWA, metode yang dikembangkan mampu mengurangi rata-rata waktu pencarian sebesar 12,62–24,5% dan panjang jalur sebesar 9,25–9,5%. Pada pengujian navigasi nyata, waktu pencarian berkurang sekitar 18–24% dan panjang jalur berkurang sekitar 10,3–12%, dengan lintasan yang lebih halus	efisiensi pencarian dan memperpendek lintasan robot mobile nonholonomik serta menggunakan MPC untuk navigasi lokal, sehingga belum membahas evaluasi navigasi pada platform AGV forklift yang memiliki dimensi kendaraan dan karakteristik manuver khusus pada lingkungan pengujian.
4	<i>Utilizability of Navigation2/ROS2 in Highly Automated and Distributed Multi-Robotic Systems for Industrial Facilitie</i> (Horeličan 2022)	Penelitian menerapkan ROS 2 Navigation2 (Nav2) pada platform robot mobile yang dikembangkan untuk lingkungan industri. Navigation2 diimplementasikan melalui waypoint follower, planner server, controller server, recovery server, costmap, dan Behavior Tree. Pada perencanaan jalur global digunakan NavfnPlanner berbasis algoritma Dijkstra, sedangkan koreksi	Penelitian berfokus pada implementasi ROS 2 Navigation2 pada robot mobile industri dengan NavfnPlanner berbasis Dijkstra sebagai global planner dan DWBLocalPlanner sebagai local planner, sehingga belum membahas penerapan Hybrid A melalui SmacPlannerHybrid sebagai global planner dan evaluasi jalur berdasarkan karakteristik

No	Penelitian dan Judul	Hasil Penelitian	Perbedaan
		lintasan lokal menggunakan DWBLocalPlanner. Penelitian menunjukkan bahwa Nav2 dapat diintegrasikan pada platform robot khusus berbasis ROS 2 untuk mendukung navigasi otonom pada lingkungan industri yang memiliki rintangan.	gerak kendaraan nonholonomik seperti yang digunakan pada AGV forklift

Berdasarkan Tabel 2.1, keempat penelitian tersebut menyoroti bagian yang berbeda dari rantai navigasi. Penelitian pertama berfokus pada penyaringan data pemindaian sebelum masuk ke algoritma SLAM, penelitian kedua pada peningkatan ketahanan lokalisasi terhadap perubahan lingkungan, penelitian ketiga pada efisiensi perencanaan jalur untuk robot nonholonomik, dan penelitian keempat pada penerapan kerangka kerja *Navigation2* di lingkungan industri. Keempatnya menunjukkan bahwa pemetaan, lokalisasi, dan perencanaan jalur telah matang secara terpisah, tetapi belum diuji sebagai satu rantai utuh pada kendaraan dengan batas manuver seperti *forklift*.

Perbedaan yang paling menonjol terletak pada platform dan model gerak yang digunakan. Sebagian penelitian dilakukan pada simulasi atau pada robot beroda diferensial yang dapat berputar di tempat, sehingga jalur hasil perencanaan hampir selalu dapat diikuti. Pada AGV *forklift*, kendaraan tidak dapat berputar di tempat dan memiliki radius belok minimum, sehingga jalur yang benar secara geometris belum tentu dapat dieksekusi. Selain itu, penelitian yang telah menerapkan *Navigation2* masih menggunakan perencana berbasis grid yang mengabaikan orientasi kendaraan.

Berdasarkan permasalahan tersebut, penelitian ini mengintegrasikan LiDAR 2D, SLAM, lokalisasi AMCL, dan perencanaan jalur global *Hybrid A** melalui *SmacPlannerHybrid* dengan model gerak *Reeds-Shepp*, kemudian mengeksekusinya menggunakan pengendali MPPI bermodel *Ackermann* pada AGV *forklift* nyata. Fokus penelitian bukan pada pengembangan algoritma baru, melainkan pada perancangan, integrasi, dan evaluasi menyeluruh rantai navigasi tersebut pada kendaraan dengan batas

manuver yang nyata. Teori pada subbab berikutnya disusun mengikuti urutan rantai tersebut, yaitu dari platform dan sensor, pemetaan, lokalisasi, perencanaan jalur, hingga pengendalian dan keselamatan. Dengan integrasi tersebut, sistem diharapkan mampu melakukan pemetaan, lokalisasi, perencanaan jalur, dan navigasi secara terintegrasi pada lingkungan *indoor*.

2.2. *Automated Guided Vehicle (AGV)*

Automated Guided Vehicle (AGV) adalah kendaraan otomatis yang digunakan dalam sistem industri modern untuk memindahkan material secara efisien tanpa keterlibatan operator manusia secara langsung. Dalam beberapa tahun terakhir, perkembangan AGV menunjukkan peningkatan signifikan, terutama pada integrasi sistem navigasi cerdas dan otomatisasi berbasis sensor. AGV kini menjadi bagian penting dalam sistem logistik cerdas dan *smart warehouse* karena kemampuannya dalam meningkatkan produktivitas serta fleksibilitas operasional [7]. Sistem AGV modern memanfaatkan berbagai sensor seperti LiDAR, kamera, dan encoder untuk mendukung navigasi serta penghindaran rintangan. Selain itu, pendekatan berbasis SLAM (*Simultaneous Localization and Mapping*) banyak digunakan untuk memungkinkan AGV melakukan pemetaan lingkungan dan lokalisasi secara *real-time*. Pendekatan ini terbukti mampu meningkatkan kemampuan AGV dalam beroperasi pada lingkungan dinamis dibandingkan metode konvensional berbasis jalur tetap [8].

Selain navigasi berbasis SLAM, pengembangan AGV juga mengarah pada penggunaan sistem navigasi *hybrid* yang menggabungkan beberapa metode, seperti *magnetic guidance* dan *laser-based navigation*. Pendekatan ini memungkinkan peningkatan akurasi dan efisiensi tanpa perlu mengganti seluruh infrastruktur sistem yang telah ada, sehingga lebih fleksibel untuk diterapkan pada lingkungan industri yang sudah berjalan [7].

Dalam hal perencanaan lintasan, AGV menggunakan berbagai algoritma seperti A*, Dijkstra, dan Hybrid A* sebagai *global planner*, serta dikombinasikan dengan metode local planner seperti *Adaptive Monte Carlo Localization (AMCL)*, (*Dynamic Window Approach*) DWA dan TEB untuk menghasilkan jalur yang optimal dan aman. Integrasi antara perencanaan lintasan dan sistem navigasi berbasis SLAM terbukti mampu meningkatkan efisiensi waktu tempuh serta keberhasilan navigasi dalam lingkungan

gudang otomatis [8]. Dengan perkembangan tersebut, AGV tidak hanya berfungsi sebagai alat transportasi otomatis, tetapi juga sebagai bagian dari sistem cerdas yang terintegrasi dengan teknologi visualisasi dan monitoring. Namun demikian, tantangan seperti kompleksitas sistem, kebutuhan komputasi yang tinggi, serta integrasi *multi-sensor* masih menjadi fokus utama dalam penelitian AGV saat ini [9].



Gambar 2. 1 *Automated Guided Vehicle (AGV)* [4]

Gambar 2.1 memperlihatkan bentuk umum AGV beserta lingkungan kerjanya. Penelitian ini menggunakan AGV berbentuk *forklift* karena mampu meningkatkan efisiensi operasional serta mengurangi ketergantungan terhadap tenaga manusia dalam proses pemindahan barang. Selain itu, penggunaan AGV memungkinkan integrasi dengan sistem navigasi cerdas berbasis sensor dan algoritma, sehingga sesuai dengan kebutuhan penelitian yang berfokus pada pengembangan sistem navigasi otonom pada lingkungan industri. Dengan kemampuan adaptasi terhadap lingkungan dinamis, AGV menjadi platform yang tepat untuk implementasi teknologi SLAM dan perencanaan jalur secara *real-time*.

2.3. NVIDIA Jetson Orin Nano

NVIDIA Jetson Orin Nano merupakan sebuah *embedded system* berbasis GPU yang dikembangkan oleh NVIDIA untuk mendukung aplikasi komputasi tepi (*edge computing*), khususnya pada bidang kecerdasan buatan (*Artificial Intelligence/AI*) dan

robotika. Perangkat ini dirancang untuk menjalankan proses komputasi dengan performa tinggi namun tetap memiliki konsumsi daya yang relatif rendah, sehingga sangat cocok digunakan pada sistem *robot mobile* seperti *Automated Guided Vehicle* (AGV) [10]. Jetson Orin Nano dilengkapi dengan arsitektur GPU berbasis NVIDIA Ampere yang mampu mempercepat proses komputasi paralel, terutama untuk aplikasi seperti pengolahan citra, deteksi objek, serta pemrosesan data sensor secara *real-time*. Selain itu, perangkat ini juga mendukung berbagai *framework* AI seperti CUDA, TensorRT, dan *Deep Learning* SDK yang memungkinkan implementasi algoritma cerdas secara efisien pada perangkat embedded [11].

Dalam sistem robotika, Jetson Orin Nano berfungsi sebagai unit pemrosesan utama (*main controller*) yang mengintegrasikan berbagai sensor seperti LiDAR, IMU, kamera, dan encoder. Data dari sensor tersebut diproses untuk kebutuhan navigasi, seperti pemetaan lingkungan menggunakan SLAM, deteksi rintangan, serta perencanaan jalur. Kemampuan komputasi yang tinggi pada Jetson Orin Nano memungkinkan sistem untuk menjalankan algoritma kompleks secara *real-time*, sehingga meningkatkan performa dan responsivitas robot dalam lingkungan dinamis [10].

Selain itu, Jetson Orin Nano memiliki keunggulan dalam efisiensi energi dibandingkan perangkat komputasi konvensional, sehingga cocok untuk sistem mobile yang bergantung pada sumber daya baterai. Penelitian menunjukkan bahwa platform Jetson mampu memberikan keseimbangan antara performa komputasi dan konsumsi daya, sehingga banyak digunakan dalam aplikasi seperti robot otonom, sistem visi komputer, dan *Internet of Things* (IoT) berbasis AI [11]. Dengan kemampuan tersebut, Jetson Orin Nano menjadi salah satu solusi utama dalam pengembangan sistem AGV berbasis navigasi otonom, khususnya yang membutuhkan integrasi antara pemrosesan sensor, kecerdasan buatan, dan kontrol sistem secara *real-time*.



Gambar 2. 2 Jetson Orin Nano [12]

Gambar 2.2 memperlihatkan bentuk fisik perangkat yang digunakan. Jetson Orin Nano dipilih sebagai unit pemrosesan utama karena memiliki kemampuan komputasi tinggi yang diperlukan untuk menjalankan algoritma kompleks seperti SLAM, pengolahan data LiDAR, serta perencanaan jalur secara *real-time*. Selain itu, efisiensi daya yang dimiliki menjadikannya untuk sistem robot mobile berbasis baterai seperti AGV. Dengan dukungan ekosistem AI dan ROS, perangkat ini mampu mengintegrasikan seluruh proses komputasi dalam satu platform yang stabil. Pemilihan jetson orin nano ini berdasarkan tabel 2.2 sebagai spesifikasi alat.

Tabel 2. 2 Spesifikasi Jetson Orin Nano

Komponen	Jetson Orin Nano 4GB	Jetson Orin Nano 8GB
Arsitektur GPU	NVIDIA Ampere (512 CUDA, 16 Tensor Cores)	NVIDIA Ampere (1024 CUDA, 32 Tensor Cores)
Performa AI	Hingga 20 TOPS	Hingga 40 TOPS (67 TOPS pada pembaruan Super)
Prosesor (CPU)	6-core Arm Cortex-A78AE v8.2 64-bit	6-core Arm Cortex-A78AE v8.2 64-bit
Memori (RAM)	4GB 64-bit LPDDR5 (34 GB/s)	8GB 128-bit LPDDR5 (68 - 102 GB/s)

Penyimpanan	Eksternal (M.2 NVMe SSD / microSD)	Eksternal (M.2 NVMe SSD / microSD)
Video Encode	1080p30 (Diproses oleh CPU)	1080p30 (Diproses oleh CPU)
Video Decode	1x 4K60 / 2x 4K30 (H.265)	1x 4K60 / 2x 4K30 (H.265)
Konsumsi Daya	5W - 10W	7W - 15W

Berdasarkan Tabel 2.2, Jetson Orin Nano memiliki kapasitas komputasi dan ketersediaan antarmuka yang memadai untuk mendukung kebutuhan sistem pada penelitian ini. Perangkat tersebut mampu menerima data dari dua sensor berbasis komunikasi serial secara bersamaan, menjalankan proses navigasi, serta mendukung antarmuka pemantauan sistem. Dengan kemampuan tersebut, Jetson Orin Nano digunakan sebagai unit pemrosesan tingkat tinggi yang bertugas melakukan pengolahan data sensor dan menjalankan algoritma navigasi. Sementara itu, proses pengendalian aktuator dilakukan oleh unit kontrol tingkat rendah yang terpisah agar pembagian tugas komputasi pada sistem menjadi lebih terstruktur dan efisien.

2.4. LiDAR 2D (*Light Detection and Ranging*)

LiDAR (*Light Detection and Ranging*) adalah salah satu sensor yang digunakan dalam sistem robotika dan kendaraan otonom untuk melakukan pengukuran jarak serta pemetaan lingkungan secara akurat. Dalam aplikasi *robot mobile*, LiDAR banyak dimanfaatkan untuk kebutuhan navigasi, deteksi rintangan, serta pemetaan lingkungan karena tidak bergantung pada kondisi pencahayaan seperti kamera [13].

Prinsip kerja tersebut yang digunakan pada LiDAR *Time of Flight*. Jarak antara sensor dan objek diperoleh dari setengah waktu tempuh pulsa laser, karena pulsa menempuh lintasan pergi dan kembali. Hubungan tersebut dinyatakan pada Persamaan (2.1).

$$d = \frac{c\Delta t}{2} \quad (2.1)$$

Keterangan :

- d : jarak antara sensor dan objek (m);
- c : kecepatan cahaya di udara (m/s);
- Δt : waktu tempuh pulsa laser pergi dan kembali (s).

Persamaan tersebut menunjukkan bahwa jarak berbanding lurus terhadap waktu tempuh, sehingga ketelitian pengukuran ditentukan oleh ketelitian pencacahan waktu

pada sensor. Faktor setengah digunakan karena waktu yang terukur mencakup lintasan pergi sekaligus lintasan kembali. Dalam penelitian ini, nilai jarak hasil hubungan tersebut merupakan besaran mentah yang diterbitkan sensor untuk setiap berkas laser, dan besaran inilah yang dievaluasi ketelitiannya pada pengujian sensor sebelum digunakan untuk pemetaan maupun lokalisasi.

LiDAR 2D umumnya digunakan pada lingkungan *indoor* karena mampu melakukan pemindaian secara horizontal hingga 360° , sehingga menghasilkan data dalam bentuk *scan* dua dimensi yang merepresentasikan kondisi lingkungan di sekitar robot. Data ini berupa kumpulan titik (*point cloud* 2D) yang digunakan dalam proses *Simultaneous Localization and Mapping* (SLAM) untuk membangun peta serta menentukan posisi robot secara *real-time*. Penggunaan LiDAR dalam SLAM terbukti mampu memberikan akurasi yang tinggi dan stabilitas yang baik, terutama pada lingkungan yang terstruktur seperti gudang atau area industri [5].

Setiap data *LaserScan* memiliki pasangan jarak r dan sudut θ . Agar data polar tersebut dapat dianalisis sebagai titik pada bidang dua dimensi, koordinatnya perlu dikonversikan ke sumbu x dan y . Hubungan konversi koordinat polar ke koordinat Kartesian dapat dinyatakan pada Persamaan (2.2) dan Persamaan (2.3).

$$x = r \cos(\theta) \quad (2.2)$$

$$y = r \sin(\theta) \quad (2.3)$$

Keterangan:

- x : koordinat titik pada sumbu mendatar kerangka sensor (m);
- y : koordinat titik pada sumbu tegak kerangka sensor (m);
- r : jarak hasil pengukuran LiDAR terhadap objek (m);
- θ : sudut arah pemindaian terhadap sumbu referensi sensor (rad).

Kedua persamaan tersebut memperlihatkan bahwa posisi sebuah titik sepenuhnya ditentukan oleh pasangan jarak dan sudut, sehingga kesalahan pada salah satunya akan menggeser titik pada bidang. Kesalahan jarak menggeser titik menjauh atau mendekat terhadap sensor, sedangkan kesalahan sudut menggesernya secara menyamping dengan pergeseran yang membesar seiring bertambahnya jarak. Transformasi ini digunakan untuk mengubah data LiDAR yang awalnya berbentuk koordinat polar berupa jarak dan sudut menjadi koordinat Kartesian dua dimensi sehingga setiap titik hasil pemindaian

dapat direpresentasikan terhadap posisi robot dan digunakan dalam proses pemetaan maupun lokalisasi.

Salah satu jenis LiDAR 2D yang banyak digunakan adalah YDLiDAR, yang dirancang untuk aplikasi robotika mobile dengan kebutuhan navigasi dan pemetaan lingkungan. YDLiDAR mampu melakukan pemindaian secara 360° pada bidang *horizontal* serta menghasilkan data jarak secara kontinu dan *real-time*. Data tersebut digunakan dalam berbagai proses seperti deteksi rintangan (*obstacle detection*), pemetaan lingkungan (*mapping*), serta estimasi posisi robot (*localization*). Selain itu, YDLiDAR memiliki keunggulan dalam hal ukuran yang relatif kecil (*compact*), konsumsi daya yang rendah, kemudahan integrasi dengan sistem berbasis ROS, serta biaya yang lebih terjangkau dibandingkan sensor LiDAR lainnya. Meskipun memiliki keterbatasan dibandingkan LiDAR 3D, sensor ini tetap efektif digunakan pada sistem navigasi AGV di lingkungan *indoor* [4].



Gambar 2. 3 Sensor YDLiDAR T-Mini-Plus 25 [14]

Gambar 2.3 memperlihatkan bentuk fisik YDLiDAR T-Mini-Plus 25 yang digunakan sebagai sensor utama untuk membaca kondisi lingkungan di sekitar AGV. Sensor ini melakukan pemindaian pada bidang horizontal secara terus-menerus sehingga jarak objek dari berbagai arah dapat diperoleh dalam satu putaran. Data hasil pemindaian tersebut selanjutnya digunakan sebagai informasi lingkungan untuk mendukung proses pemetaan, lokalisasi, dan pendeteksian rintangan pada AGV. Karakteristik teknis sensor yang menjadi dasar penggunaannya pada sistem ini disajikan pada Tabel 2.3.

Tabel 2. 3 Spesifikasi YDLiDAR

Komponen	YDLiDAR T-Mini-Plus 25
Arsitektur Sensor	LiDAR 2D berbasis <i>Time-of-Flight</i> (ToF)

Komponen	YDLiDAR T-Mini-Plus 25
Performa SLAM	<i>Real-time mapping & localization</i> (2D SLAM)
Jangkauan	0.1 – 25 m
Akurasi	$\pm 2 - 3$ cm
<i>Field of View</i>	360° (horizontal)
Frekuensi Scan	5 – 15 Hz
Resolusi sudut	$\pm 0.5^\circ - 1^\circ$
Prosesor (CPU)	ARM Cortex / terintegrasi dengan SBC (Jetson/Raspberry Pi)
Interface	USB, UART
Output Data	Peta 2D, point cloud, pose (x, y, θ)
Konsumsi Daya	$\pm 2W - 5W$

Berdasarkan Gambar 2.3 dan Tabel 2.3, sensor yang digunakan pada penelitian ini adalah YDLiDAR T-mini Plus 25 dengan cakupan pemindaian 360 derajat pada bidang mendatar. Jangkauan dan resolusi sudut menentukan kerapatan titik yang dihasilkan setiap putaran, sedangkan frekuensi pemindaian menentukan seberapa sering representasi lingkungan dapat diperbarui. Kedua besaran tersebut menjadi pertimbangan utama pada perancangan sistem, karena kerapatan titik memengaruhi kualitas pencocokan pemindaian sementara frekuensi pemindaian memengaruhi kemampuan sistem merespons rintangan yang muncul mendadak.

2.4.1. Evaluasi Akurasi dan Presisi Data LiDAR

Kualitas data LiDAR tidak cukup dinilai dari kemampuannya menampilkan bentuk ruangan, melainkan harus diukur terhadap jarak referensi yang diketahui. Pendekatan yang umum digunakan adalah menghitung selisih antara jarak hasil pembacaan dan jarak referensi sebagai error pengukuran [15]. Hubungan tersebut pada penelitian ini dinyatakan Persamaan (2.4).

$$e_i = d_i - d_{ref} \quad (2.4)$$

Keterangan:

e_i : error pengukuran pada sampel ke- i (m);

d_i : jarak hasil pembacaan LiDAR pada sampel ke- i (m);

d_{ref} : jarak referensi hasil pengukuran langsung (m).

Nilai error yang positif menunjukkan pembacaan sensor lebih besar daripada jarak sebenarnya, sedangkan nilai negatif menunjukkan sebaliknya. Karena tanda error membawa informasi arah penyimpangan, nilai ini tidak dirata-ratakan secara langsung agar penyimpangan positif dan negatif tidak saling meniadakan. Dalam penelitian ini, error tiap sampel menjadi masukan bagi ketiga besaran berikutnya yang menilai akurasi rata-rata, pengaruh penyimpangan besar, dan kestabilan pembacaan.

Rata-rata besar penyimpangan pada seluruh sampel dinyatakan menggunakan *Mean Absolute Error* pada Persamaan (2.5).

$$MAE = \frac{1}{n} \sum_{i=1}^n |e_i| \quad (2.5)$$

Keterangan:

MAE : rata-rata error absolut pengukuran LiDAR (m);

n : jumlah sampel pengukuran;

e_i : error pengukuran pada sampel ke- i (m).

Penggunaan nilai absolut membuat setiap penyimpangan dihitung menurut besarnya saja, sehingga hasilnya menggambarkan seberapa jauh pembacaan sensor menyimpang secara tipikal. Semakin kecil nilainya, semakin dekat pembacaan LiDAR terhadap jarak sebenarnya. Dalam penelitian ini besaran tersebut dipakai sebagai indikator utama akurasi LiDAR dan dilaporkan untuk setiap jarak referensi maupun setiap kondisi pengoperasian aktuator yang diuji.

Mean Absolute Error memperlakukan seluruh penyimpangan secara setara, sehingga beberapa penyimpangan besar dapat tersamarkan oleh banyak penyimpangan kecil. Untuk menyoroiti penyimpangan besar digunakan Root Mean Square Error pada Persamaan (2.6).

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n e_i^2} \quad (2.6)$$

Keterangan:

RMSE : akar rata-rata kuadrat error pengukuran LiDAR (m);

n : jumlah sampel pengukuran;

e_i : error pengukuran pada sampel ke- i (m).

Proses pengkuadratan memberi bobot lebih besar pada penyimpangan yang besar, sehingga nilai RMSE akan jauh melebihi MAE apabila terdapat pencilan pada data. Perbandingan antara keduanya karena itu bersifat diagnostic, selisih yang kecil menunjukkan penyimpangan tersebar merata, sedangkan selisih yang besar menunjukkan adanya pembacaan menyimpang yang perlu ditelusuri. Dalam penelitian ini keduanya dilaporkan berpasangan agar kesimpulan mengenai kualitas LiDAR tidak diambil hanya dari satu angka.

Akurasi menyatakan kedekatan terhadap nilai sebenarnya, sedangkan konsistensi pembacaan pada kondisi yang sama dinyatakan melalui simpangan baku pada Persamaan (2.7).

$$\sigma_x = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2} \quad (2.7)$$

Ketrangan:

- σ_x : simpangan baku data pengukuran (m);
- x_i : nilai sampel ke- i (m);
- $\bar{x}$: nilai rata-rata seluruh sampel (m);
- n : jumlah sampel pengukuran.

Simpangan baku menggambarkan sebaran pembacaan terhadap nilai rata-ratanya, sehingga sebuah sensor dapat memiliki simpangan baku kecil meskipun nilai rata-ratanya bergeser dari jarak sebenarnya. Karena itu simpangan baku dibaca bersama *Mean Absolute Error* yaitu yang pertama menilai kestabilan, yang kedua menilai ketepatan. Dalam penelitian ini simpangan baku menjadi tolak ukur penting karena pemetaan dan lokalisasi mencocokkan pemindaian antarwaktu, sehingga pembacaan yang berubah-ubah pada objek diam akan menurunkan kualitas pencocokan meskipun rata-ratanya benar.

Keempat besaran pada Persamaan (2.4) sampai Persamaan (2.7) digunakan secara saling melengkapi. *Error* menunjukkan arah dan besar penyimpangan tiap sampel, *Mean Absolute Error* menunjukkan penyimpangan tipikal, *Root Mean Square Error* menyoroti penyimpangan besar, dan simpangan baku menunjukkan kestabilan pembacaan.

Kombinasi parameter tersebut memungkinkan kualitas data LiDAR dianalisis tidak hanya dari sisi akurasi, tetapi juga dari sisi presisi dan konsistensi selama AGV beroperasi

2.4.2. Pengaruh Getaran dan Aktuator terhadap LiDAR

Getaran mekanik dapat memengaruhi kualitas pembacaan LiDAR karena sensor dipasang pada rangka AGV yang menerima getaran selama sistem bekerja. Getaran tersebut dapat menimbulkan perubahan kecil pada posisi dan orientasi LiDAR saat proses pemindaian berlangsung, sehingga titik hasil pengukuran pada beberapa putaran scan dapat mengalami penyebaran, noise, atau pergeseran relatif terhadap objek yang sebenarnya diam. Apabila kondisi ini terjadi secara terus-menerus, data scan yang kurang konsisten dapat menyebabkan dinding atau obstacle pada peta terlihat lebih tebal, muncul titik pencilan, dan menurunkan kestabilan proses pencocokan scan pada SLAM. Oleh karena itu, kekakuanudukan LiDAR dan kestabilan mekanik rangka menjadi faktor penting agar data jarak yang digunakan untuk pemetaan dan navigasi tetap konsisten.

Pada AGV forklift yang digunakan dalam penelitian ini, sumber getaran aktuator yang diperhatikan adalah motor BLDC dan *electric winch*. Motor BLDC dapat meneruskan getaran ke rangka ketika motor mulai berputar, mengalami perubahan kecepatan, atau menerima perubahan beban selama AGV bergerak. Sementara itu, *electric winch* dapat menghasilkan getaran dan hentakan mekanik pada saat proses pengangkatan maupun penurunan beban, terutama ketika motor winch mulai bekerja, berhenti, atau menahan beban. Getaran dari kedua aktuator tersebut dapat diteruskan menujuudukan LiDAR sehingga berpotensi meningkatkan variasi pembacaan jarak. Karena itu, pengujian LiDAR perlu membandingkan kondisi aktuator tidak aktif dengan kondisi motor BLDC dan *electric winch* aktif agar perubahan error, MAE, RMSE, dan simpangan baku dapat digunakan untuk menilai seberapa besar pengaruh getaran aktuator terhadap kualitas data LiDAR.

2.5. Inertial Measurement Unit (IMU)

Inertial Measurement Unit (IMU) merupakan sensor inersial yang digunakan untuk memperoleh informasi mengenai gerakan dan orientasi suatu objek berdasarkan perubahan percepatan dan kecepatan sudut. IMU umumnya terdiri dari *accelerometer* dan *gyroscope* yang bekerja pada beberapa sumbu pengukuran. *Accelerometer* digunakan untuk mengukur percepatan linear pada sumbu tertentu, sedangkan *gyroscope* digunakan

untuk mengukur kecepatan sudut atau *angular velocity* yang menunjukkan perubahan rotasi sensor. Informasi dari kedua sensor tersebut dapat digunakan untuk memperkirakan orientasi robot dalam bentuk *roll*, *pitch*, dan *yaw*. Roll menunjukkan rotasi terhadap sumbu *longitudinal*, *pitch* menunjukkan rotasi terhadap sumbu lateral, sedangkan *yaw* menunjukkan perubahan arah atau rotasi terhadap sumbu vertikal robot. Dalam penerapannya, data IMU memiliki *noise* dan bias pengukuran yang dapat menyebabkan kesalahan estimasi, terutama ketika data *gyroscope* diintegrasikan dalam waktu yang lama sehingga menimbulkan *drift*. Oleh karena itu, data IMU umumnya digunakan sebagai informasi pendukung yang perlu dikombinasikan dengan sensor atau metode estimasi lainnya untuk memperoleh informasi gerak dan orientasi yang lebih stabil.

Berdasarkan penelitian Mahmood dan Baluch (2025), IMU dapat digunakan sebagai bagian dari sistem navigasi robot bergerak untuk menyediakan informasi gerak inersial, khususnya melalui penggabungan data accelerometer dan gyroscope dalam estimasi orientasi, sementara penggabungan dengan sensor lain dapat membantu mengurangi pengaruh *drift* dan meningkatkan kontinuitas estimasi posisi [16]. Berdasarkan penelitian tersebut, IMU pada sistem AGV dapat dimanfaatkan sebagai informasi gerak inersial yang mendukung estimasi pergerakan robot, terutama dalam memberikan informasi perubahan orientasi melalui gyroscope dan percepatan melalui accelerometer. Dengan demikian, IMU tidak digunakan sebagai sumber utama localization pada penelitian ini, melainkan sebagai data pendukung navigasi yang dapat membantu sistem memahami perubahan gerak dan orientasi AGV selama bergerak.



Gambar 2. 4 Yahboom IMU 10-Axis [17]

Gambar 2.4 memperlihatkan bentuk fisik Yahboom IMU 10-Axis yang digunakan sebagai sensor pendukung pada sistem AGV. Sensor ini berperan dalam membaca

perubahan gerak dan orientasi kendaraan selama beroperasi melalui data *accelerometer*, *gyroscope*, dan *magnetometer*. Informasi tersebut memberikan gambaran mengenai perubahan arah dan gerakan AGV, sehingga dapat digunakan sebagai data pendukung dalam proses estimasi pergerakan pada sistem navigasi. Pada penelitian ini, informasi orientasi dari IMU terutama dimanfaatkan untuk mengetahui perubahan arah hadap atau *yaw* kendaraan selama bergerak. Data tersebut membantu sistem dalam mengikuti perubahan orientasi AGV ketika bergerak lurus maupun melakukan manuver belok. Dengan adanya informasi orientasi dari IMU, perubahan arah kendaraan dapat diamati secara lebih kontinu dan tidak hanya bergantung pada perubahan posisi yang diperoleh dari sensor lainnya. Hal ini penting karena selama proses navigasi, AGV tidak hanya membutuhkan informasi posisi pada sumbu x dan y, tetapi juga membutuhkan informasi arah hadap kendaraan agar estimasi gerakannya sesuai dengan kondisi aktual. Oleh karena itu, IMU digunakan sebagai salah satu sumber informasi pendukung dalam proses estimasi gerak dan lokalisasi AGV. Karakteristik teknis sensor yang menjadi dasar penggunaannya pada penelitian ini disajikan pada tabel 2.4 berikut.

Tabel 2. 4 Spesifikasi Yahboom IMU 10-Axis

Komponen	Yahboom IMU 10-Axis
Arsitektur Sensor	IMU 10-axis berbasis MEMS
Sensor Inersial	3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer, dan barometer
Rentang Accelerometer	± 16 g
Rentang Gyroscope	$\pm 2000^\circ/\text{s}$
Rentang Magnetometer	± 2 Gauss
Rentang Barometer	300 – 1100 hPa
Akurasi Pitch & Roll	$\pm 0,1^\circ$ (statis), $\pm 0,5^\circ$ (dinamis)
Akurasi Heading (Yaw)	$\pm 1^\circ$ tanpa gangguan medan magnet
Frekuensi Output	0,2 – 200 Hz
Baud Rate	4800 – 921600 bps
Interface	USB, UART, I2C
Output Data	Akselerasi, kecepatan sudut, medan magnet, Roll, Pitch, Yaw, quaternion, tekanan udara, dan ketinggian

Komponen	Yahboom IMU 10-Axis
Tegangan Operasi	3,3 V / 5 V

Berdasarkan karakteristik tersebut, Yahboom IMU 10-Axis dipilih sebagai sensor pendukung pada sistem AGV karena mampu menyediakan informasi gerak dan orientasi secara real-time melalui *accelerometer*, *gyroscope*, dan *magnetometer*. Sensor ini dapat menghasilkan data percepatan, kecepatan sudut, serta orientasi berupa Roll, Pitch, dan Yaw, sehingga dapat membantu sistem dalam mengetahui perubahan gerak dan arah kendaraan selama beroperasi. Dengan frekuensi output hingga 200 Hz dan dukungan komunikasi USB, UART, serta I2C, sensor dapat diintegrasikan dengan sistem kontrol dan komputasi AGV. Dalam penelitian ini, IMU digunakan sebagai informasi gerak inersial pendukung, terutama untuk memberikan informasi perubahan orientasi dan pergerakan robot, sedangkan proses localization utama tetap menggunakan data LiDAR dan metode SLAM/AMCL.

Orientasi yaw umumnya tidak dikeluarkan langsung sebagai sudut, melainkan dalam bentuk quaternion agar bebas dari singularitas. Nilai yaw diperoleh dari komponen quaternion menggunakan Persamaan (2.8).

$$\psi = \text{atan2} \left(2(q_w q_z + q_x q_y), 1 - 2(q_y^2 + q_z^2) \right) \quad (2.8)$$

Keterangan:

- Ψ : sudut yaw hasil perhitungan (rad);
- q_w : komponen skalar quaternion orientasi;
- q_x, q_y, q_z : komponen vektor quaternion orientasi;
- atan2 : fungsi arkus tangen dua argumen yang mempertahankan kuadran sudut.

Penggunaan fungsi arkus tangen dua argumen diperlukan agar sudut yang dihasilkan tetap benar pada seluruh kuadran, sehingga arah hadap kendaraan tidak tertukar ketika AGV menghadap ke belakang. Nilai yaw yang diperoleh berada pada rentang satu putaran penuh dan bersifat periodik. Dalam penelitian ini, nilai inilah yang dipakai sebagai satu-satunya variabel orientasi dari IMU, baik pada pemantauan gerak maupun pada pengujian ketelitian orientasi.

Penyimpangan orientasi terhadap sudut referensi dihitung sebagai error yaw pada Persamaan (2.9).

$$e_{\psi,i} = \text{wrap}(\psi_i - \psi_{\text{ref},i}) \quad (2.9)$$

Keterangan:

- $e_{\psi,i}$: error yaw pada sampel ke- i (rad);
 ψ_i : yaw hasil pengukuran IMU pada sampel ke- i (rad);
 $\psi_{\text{ref},i}$: yaw referensi pada sampel ke- i (rad);
 wrap : fungsi pembatas selisih sudut ke rentang $-\pi$ sampai π (rad).

Fungsi pembatas sudut diperlukan karena yaw bersifat periodik, sehingga selisih antara dua sudut yang sebenarnya berdekatan tetapi berada di sekitar batas putaran dapat terbaca sangat besar apabila dihitung secara langsung. Tanpa pembatasan tersebut, perbedaan orientasi yang sesungguhnya kecil dapat tercatat mendekati satu putaran penuh dan merusak perhitungan rata-rata. *Error yaw* hasil Persamaan (2.9) menjadi dasar bagi kedua besaran orientasi berikutnya. Rata-rata besar penyimpangan orientasi dinyatakan pada Persamaan (2.10).

$$\text{MAE}_{\psi} = \frac{1}{n} \sum_{i=1}^n |e_{\psi,i}| \quad (2.10)$$

Keterangan:

- MAE_{ψ} : rata-rata error absolut orientasi yaw (rad);
 n : jumlah sampel pengukuran;
 $e_{\psi,i}$: error yaw pada sampel ke- i (rad).

Besaran tersebut menunjukkan penyimpangan orientasi secara tipikal tanpa memperhatikan arah penyimpangan. Karena orientasi menentukan arah proyeksi seluruh titik pemindaian, penyimpangan yaw yang kecil sekalipun dapat menggeser posisi rintangan pada peta secara cukup besar pada jarak yang jauh. Dalam penelitian ini nilai tersebut dipakai untuk menilai apakah orientasi yang dilaporkan IMU cukup andal untuk mendukung estimasi gerak AGV. Kepekaan terhadap penyimpangan orientasi yang besar dinyatakan pada Persamaan (2.11).

$$\text{RMSE}_{\psi} = \sqrt{\frac{1}{n} \sum_{i=1}^n e_{\psi,i}^2} \quad (2.11)$$

Keterangan:

$RMSE^\psi$: akar rata-rata kuadrat error orientasi yaw (rad);

N : jumlah sampel pengukuran;

$e_{\psi,i}$: error yaw pada sampel ke- i (rad).

Sebagaimana pada pengujian jarak, pengkuadratan membuat lonjakan orientasi memberi pengaruh besar terhadap hasil. Selisih yang lebar antara kedua besaran orientasi menandakan adanya lonjakan sesaat atau pergeseran orientasi yang berlangsung perlahan, dan keduanya memerlukan penanganan yang berbeda. Dengan demikian Persamaan (2.8) sampai Persamaan (2.11) membentuk satu rangkaian utuh, yaitu memperoleh yaw dari quaternion, menghitung penyimpangannya terhadap referensi, lalu menilainya secara rata-rata dan secara kuadratik. Hasil evaluasi tersebut digunakan untuk memastikan bahwa perubahan orientasi yang dipublikasikan oleh IMU cukup stabil untuk mendukung estimasi gerak AGV.

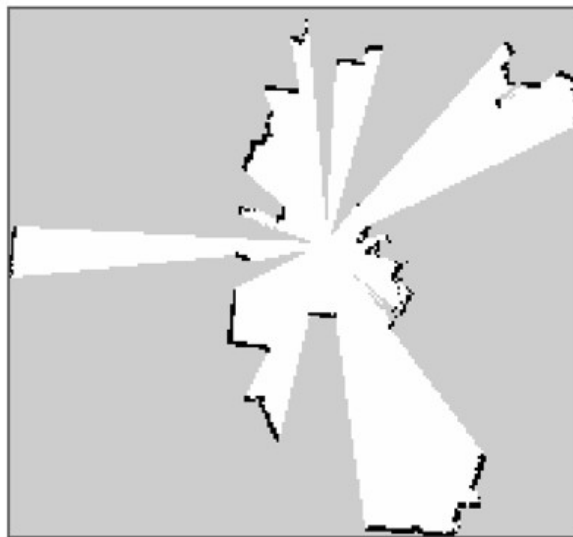
2.6. *Simultaneous Localization and Mapping (SLAM)*

Simultaneous Localization and Mapping (SLAM) yaitu metode yang digunakan pada *robot mobile* untuk melakukan pemetaan lingkungan (*mapping*) sekaligus menentukan posisi robot (*localization*) secara bersamaan dalam lingkungan yang belum diketahui. SLAM menjadi komponen utama dalam sistem navigasi otonom karena memungkinkan robot bergerak tanpa bergantung pada peta awal maupun jalur tetap [4]. Prinsip dasar SLAM adalah memanfaatkan data sensor seperti LiDAR untuk membangun representasi lingkungan serta memperkirakan posisi robot secara *real-time*. Pada pendekatan *LiDAR-based SLAM*, proses ini dilakukan dengan memanfaatkan data hasil pemindaian (*scan*) yang kemudian dicocokkan antar waktu untuk memperoleh perubahan posisi robot. Metode seperti F-LOAM menggunakan pendekatan *scan-to-scan* dan *scan-to-map matching* untuk meningkatkan akurasi estimasi posisi sekaligus menjaga efisiensi komputasi [4]. Berdasarkan penelitian Wang et al., penerapan LiDAR SLAM pada AGV di lingkungan gudang mampu melakukan pelacakan posisi robot dengan tingkat kesalahan lokalisasi rata-rata sebesar 2 cm, serta memberikan solusi SLAM yang cepat dan akurat untuk aplikasi robot secara *real-time* [4].

Secara umum, SLAM terdiri dari dua bagian utama, yaitu *front-end* dan *back-end*. Pada bagian *front-end*, sistem melakukan ekstraksi fitur dari data LiDAR, seperti fitur

tepi (*edge*) dan bidang (*plane*), kemudian mencocokkannya dengan data sebelumnya untuk memperoleh estimasi pergerakan robot. Sedangkan pada bagian *back-end*, dilakukan optimasi untuk memperbaiki kesalahan estimasi serta menjaga konsistensi peta secara global, termasuk melalui proses seperti *loop closure* [5]. Dalam implementasinya, SLAM berbasis LiDAR banyak digunakan pada lingkungan *indoor* karena memiliki tingkat akurasi yang tinggi dan tidak dipengaruhi oleh kondisi pencahayaan. Selain itu, pengembangan terbaru seperti ASL-SLAM menunjukkan bahwa penambahan informasi semantik dan teknik *loop closure* dapat meningkatkan akurasi peta serta stabilitas sistem dalam lingkungan yang kompleks [5]. Berdasarkan penelitian Zhou et al., penerapan *loop closure* berbasis informasi aktivitas dan optimasi graf dapat mengurangi kesalahan kumulatif, meningkatkan akurasi lintasan global, serta meningkatkan konsistensi peta yang dihasilkan.

Meskipun SLAM memiliki banyak keunggulan, terdapat beberapa tantangan dalam implementasinya, seperti akumulasi *error (drift)*, kebutuhan komputasi yang tinggi, serta kesulitan dalam menangani lingkungan yang dinamis. Oleh karena itu, penelitian saat ini terus berkembang ke arah peningkatan efisiensi algoritma serta integrasi berbagai metode untuk menghasilkan sistem navigasi yang lebih akurat [5].



Gambar 2. 5 Pengujian AGV menggunakan SLAM

Pada Gambar 2.5 menunjukkan penerapan metode *Simultaneous Localization and Mapping* (SLAM) pada AGV di area pengujian, serta hasil pemetaan yang terbentuk

selama proses berlangsung. Metode SLAM dipilih karena mampu memungkinkan robot untuk melakukan pemetaan lingkungan sekaligus menentukan posisi secara bersamaan tanpa memerlukan peta awal. Hal ini sangat penting pada lingkungan yang dinamis dan belum terpetakan. Dengan memanfaatkan data LiDAR, SLAM mampu menghasilkan peta yang akurat serta mendukung navigasi otonom secara *real-time*, sehingga menjadi metode utama dalam sistem yang dikembangkan pada penelitian ini.

Kualitas peta hasil SLAM ditinjau dari ketegasan batas dinding, kesinambungan koridor, serta kemunculan garis ganda dan pergeseran struktur. Pengamatan dilakukan pada bagian lingkungan yang bersesuaian agar hasil pemetaan dapat dibandingkan secara konsisten. Durasi pembentukan *map* juga digunakan untuk melengkapi perbandingan hasil. Waktu pemetaan perlu dibaca bersama bentuk peta yang dihasilkan karena durasi yang lebih singkat belum tentu menunjukkan kualitas yang lebih baik.

Selain pengamatan struktur, konsistensi estimasi posisi ditinjau ketika AGV kembali ke titik fisik awal setelah menempuh lintasan tertutup. Selisih posisi awal dan akhir dihitung dalam kerangka koordinat yang sama menggunakan Persamaan (2.12).

$$e_{loop} = \sqrt{(x_f - x_0)^2 + (y_f - y_0)^2} \quad (2.12)$$

Keterangan:

- e_{loop} : error penutupan lintasan (m);
- x_0 dan y_0 : koordinat posisi awal lintasan tertutup (m);
- x_f dan y_f : koordinat posisi akhir setelah kembali ke titik awal (m).

Persamaan tersebut digunakan untuk mengetahui besarnya perbedaan posisi antara titik awal dan titik akhir setelah AGV menyelesaikan lintasan tertutup dan kembali ke lokasi awal. Nilai e_{loop} yang kecil menunjukkan bahwa posisi akhir AGV berada semakin dekat dengan posisi awal, sehingga pergeseran posisi selama proses pemetaan relatif kecil. Sebaliknya, nilai e_{loop} yang semakin besar menunjukkan adanya perbedaan posisi yang lebih besar setelah AGV menyelesaikan lintasan. Oleh karena itu, nilai e_{loop} dapat digunakan sebagai salah satu parameter untuk mengevaluasi konsistensi hasil pemetaan, khususnya dalam melihat kemampuan sistem SLAM mempertahankan kesesuaian posisi ketika AGV kembali ke titik awal.

2.7. Adaptive Monte Carlo Localization (AMCL)

Adaptive Monte Carlo Localization (AMCL) merupakan metode lokalisasi yang digunakan untuk menentukan posisi dan orientasi robot pada peta yang telah tersedia. AMCL bekerja menggunakan pendekatan *particle filter*, yaitu merepresentasikan kemungkinan posisi robot dalam sejumlah partikel yang memiliki nilai probabilitas tertentu. Data odometri digunakan untuk memperkirakan perpindahan robot, sedangkan data sensor LiDAR digunakan untuk memperbarui bobot partikel berdasarkan kesesuaian hasil pengukuran sensor terhadap peta lingkungan. Jumlah partikel pada AMCL dapat disesuaikan secara adaptif sesuai dengan tingkat ketidakpastian posisi robot, sehingga proses estimasi posisi dapat dilakukan secara efisien. Dalam sistem navigasi robot, AMCL digunakan setelah proses pemetaan selesai untuk melakukan estimasi posisi robot secara *real-time* terhadap peta yang telah dibuat.

Berdasarkan penelitian Zhu dan Luo (2024), metode AMCL dapat digunakan untuk melakukan lokalisasi *robot mobile* pada lingkungan indoor, tetapi akurasi lokalisasi dapat dipengaruhi oleh kompleksitas lingkungan dan keterbatasan model sensor LiDAR. Penelitian tersebut menunjukkan bahwa pengembangan metode AMCL dengan mekanisme tambahan dapat meningkatkan akurasi dan ketahanan proses lokalisasi dibandingkan AMCL konvensional [18]. Berdasarkan penelitian tersebut, AMCL pada penelitian ini digunakan untuk menentukan posisi AGV forklift pada peta hasil SLAM menggunakan informasi LiDAR 2D dan data odometri, sehingga posisi robot dapat diketahui secara *real-time* dan selanjutnya digunakan sebagai informasi dasar dalam proses navigasi menuju titik tujuan. Ketelitian pose hasil lokalisasi terhadap posisi referensi dinyatakan pada Persamaan (2.13).

$$e_{p,i} = \sqrt{(x_i - x_{\text{ref},i})^2 + (y_i - y_{\text{ref},i})^2} \quad (2.13)$$

Keterangan:

- $e_{p,i}$: error posisi hasil lokalisasi pada sampel ke- i (m);
- x_i dan y_i : koordinat pose hasil estimasi AMCL (m);
- $x_{\text{ref},i}$ dan $y_{\text{ref},i}$: koordinat posisi referensi (m).

Jarak Euclidean dipakai agar penyimpangan pada kedua sumbu dinyatakan dalam satu nilai skalar, sehingga hasil beberapa pengulangan dapat dibandingkan secara

langsung. Nilai ini menjadi ukuran pokok ketelitian lokalisasi pada pengujian pose statis maupun pengujian keterulangan. Dalam penelitian ini, besarnya penyimpangan tersebut juga menjadi pertimbangan ketika menetapkan toleransi pencapaian tujuan, karena toleransi yang lebih ketat daripada ketelitian lokalisasi tidak mungkin dipenuhi secara konsisten.

Selain ketelitian, kecepatan sistem mencapai kondisi stabil juga perlu diukur. Waktu konvergensi dinyatakan pada Persamaan (2.14).

$$t_{conv} = \min t: e_{(xy)}(\tau) \leq \tau_{(xy)}, |e_{(\theta)}(\tau)| \leq \tau_{(\theta)}, \forall \tau \in [t, t + \Delta T] \quad (2.14)$$

Keterangan:

- t_{conv} : waktu konvergensi lokalisasi (s);
- e_{xy} : error posisi pada waktu pengamatan (m);
- e_{θ} : error orientasi pada waktu pengamatan (rad);
- τ_{xy} : batas toleransi error posisi (m);
- τ_{θ} : batas toleransi error orientasi (rad);
- ΔT : lama selang pengamatan kestabilan (s).

Persamaan tersebut menyatakan bahwa sistem baru dinyatakan konvergen apabila error posisi dan orientasi tetap berada di bawah batas toleransi selama seluruh selang pengamatan, bukan sekadar menyentuh batas tersebut sesaat. Syarat kestabilan ini penting karena sebaran partikel dapat menyempit sementara lalu melebar kembali ketika robot melewati area dengan struktur yang berulang. Dalam penelitian ini definisi tersebut dipakai sebagai dasar penentuan waktu konvergensi pada pengujian inisialisasi pose dan relokalisasi, sekaligus menjadi syarat sebelum tahap navigasi diizinkan berjalan. Kedua besaran pada Persamaan (2.13) dan Persamaan (2.14) karena itu mewakili dua sisi kinerja lokalisasi, yaitu ketepatan pose dan kecepatan mencapai kondisi stabil.

2.8. Odometri

Odometri merupakan metode estimasi pergerakan robot yang digunakan untuk mengetahui perubahan posisi dan orientasi berdasarkan informasi gerak dari roda, umumnya diperoleh melalui *wheel encoder*. Pada robot beroda, perubahan putaran roda dikonversikan menjadi perpindahan linear dan perubahan sudut robot sehingga menghasilkan estimasi pose berupa posisi x , y , dan orientasi θ . Data odometri dapat diperoleh secara kontinu dan digunakan sebagai informasi pergerakan robot dari satu

waktu ke waktu berikutnya. Namun, estimasi odometri memiliki keterbatasan karena adanya kesalahan pengukuran encoder, ketidaktepatan parameter roda, serta wheel slip yang dapat menyebabkan akumulasi kesalahan atau drift seiring bertambahnya jarak tempuh. Oleh karena itu, odometri lebih tepat digunakan sebagai informasi pergerakan relatif dan bukan sebagai satu-satunya sumber *localization*.

Berdasarkan penelitian Fariña, Toledo, dan Acosta (2025), kesalahan sistematis pada odometri dapat memengaruhi estimasi pose robot, sehingga diperlukan kalibrasi untuk meningkatkan akurasi odometri, penelitian tersebut juga menunjukkan bahwa penggabungan wheel encoder dengan IMU dan LiDAR dapat menghasilkan estimasi posisi yang lebih *robust* [19]. Berdasarkan penelitian tersebut, odometri pada sistem AGV *forklift* digunakan sebagai informasi pendukung pergerakan robot, terutama untuk memberikan estimasi perpindahan dan perubahan orientasi selama robot bergerak. Data odometri selanjutnya dapat digunakan bersama data sensor LiDAR dan metode lokalisasi untuk membantu menghasilkan estimasi posisi AGV yang lebih stabil, sehingga odometri tidak menjadi sumber utama *localization* dalam sistem yang dikembangkan

Sesuai dengan ruang lingkup penelitian, variabel odometri yang digunakan dibatasi pada kecepatan longitudinal v_x , yaitu komponen kecepatan pada sumbu memanjang kendaraan. Pembatasan ini menyesuaikan karakteristik AGV forklift dengan sistem kemudi Ackermann yang bergerak dominan pada arah maju dan mundur serta tidak memiliki pergerakan lateral secara langsung. Data kecepatan longitudinal digunakan sebagai informasi pendukung untuk menggambarkan pergerakan relatif AGV selama sistem beroperasi.

Pada penelitian ini, data odometri tidak dievaluasi sebagai pengujian akurasi tersendiri, melainkan digunakan sebagai salah satu masukan pendukung dalam rantai lokalisasi dan navigasi. Penentuan posisi AGV terhadap peta tetap dilakukan menggunakan AMCL dengan memanfaatkan peta hasil SLAM dan data LiDAR. Setelah posisi dan orientasi AGV terhadap peta diperoleh, informasi tersebut digunakan sebagai dasar pada tahap perencanaan jalur untuk menentukan lintasan dari posisi awal menuju posisi tujuan.

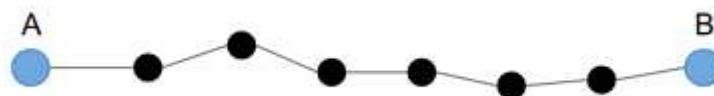
Setelah posisi dan orientasi AGV terhadap peta diperoleh melalui proses lokalisasi, informasi tersebut digunakan bersama peta lingkungan sebagai masukan pada tahap

perencanaan jalur. Tahap ini menentukan lintasan dari posisi awal menuju posisi tujuan dengan mempertimbangkan kondisi peta, keberadaan rintangan, serta karakteristik gerak kendaraan.

2.9. Global Path Planning

Global path planning merupakan proses penentuan jalur optimal dari posisi awal menuju titik tujuan berdasarkan peta lingkungan yang telah diketahui. Dalam sistem *robot mobile*, peta ini umumnya diperoleh dari proses SLAM dan direpresentasikan dalam bentuk grid atau graf. *Global planner* bertugas untuk menghitung jalur terbaik dengan mempertimbangkan jarak, hambatan, serta efisiensi pergerakan robot. Proses ini menjadi komponen penting dalam sistem navigasi otonom karena menentukan rute utama yang akan diikuti oleh robot sebelum dilakukan penyesuaian secara *real-time*.

Berbagai algoritma telah dikembangkan untuk global path planning, seperti Dijkstra, A*, dan *Rapidly-exploring Random Tree* (RRT). Algoritma Dijkstra digunakan untuk mencari jalur terpendek secara optimal pada graf berbobot, sedangkan A* merupakan pengembangan yang menggunakan fungsi heuristik untuk mempercepat pencarian jalur. Sementara itu, algoritma berbasis sampling seperti RRT digunakan untuk lingkungan yang lebih kompleks dan tidak terstruktur. Pemilihan algoritma *global planner* sangat bergantung pada karakteristik lingkungan, kebutuhan komputasi, serta tingkat kompleksitas sistem navigasi yang digunakan [20]. Berdasarkan penelitian Kahar et al., algoritma A*, Dijkstra, dan RRT merupakan beberapa algoritma yang umum digunakan dalam *global path planning*, dengan karakteristik yang berbeda dalam hal efisiensi komputasi, optimalitas jalur, dan kemampuan adaptasi terhadap lingkungan. Pemilihan algoritma global path planning perlu disesuaikan dengan karakteristik lingkungan dan kebutuhan navigasi robot.



Gambar 2. 6 *Global Path Planning* [21].

Dalam implementasi pada sistem AGV, global path planning berperan dalam menghasilkan jalur yang efisien dan bebas dari rintangan berdasarkan peta lingkungan. Jalur yang dihasilkan kemudian digunakan sebagai acuan pergerakan robot dalam

mencapai tujuan. Integrasi antara SLAM dan *global path planning* memungkinkan robot untuk melakukan navigasi secara otonom dengan tingkat akurasi dan efisiensi yang tinggi, terutama pada lingkungan indoor seperti gudang atau area industri. Penelitian menunjukkan bahwa penggunaan global planner yang tepat dapat meningkatkan performa navigasi, mengurangi waktu tempuh, serta meningkatkan keberhasilan robot dalam mencapai target secara aman [8]. Berdasarkan penelitian Abdul Majid et al., penerapan global path planning pada robot otonom di lingkungan gudang menunjukkan bahwa kombinasi algoritma global dan local planner dapat memengaruhi waktu navigasi dan tingkat keberhasilan robot. Hasil penelitian menunjukkan bahwa kombinasi Hybrid A* dengan TEB memperoleh tingkat keberhasilan sebesar 90% dan menghasilkan waktu navigasi yang lebih singkat dibandingkan kombinasi RRT dengan DWA pada pengujian lingkungan Gudang [8].

Global path planning digunakan karena berperan dalam menentukan jalur optimal dari posisi awal menuju tujuan berdasarkan peta hasil SLAM. Proses ini penting untuk memastikan robot dapat bergerak secara efisien dan menghindari rintangan dalam lingkungan operasional. Dengan adanya perencanaan jalur global, sistem navigasi menjadi lebih terstruktur dan mampu meningkatkan keberhasilan robot dalam mencapai target secara aman.

2.10. Hybrid A*

Hybrid A* merupakan pengembangan dari algoritma A* yang digunakan untuk melakukan perencanaan jalur global (*global path planning*) pada robot bergerak dengan mempertimbangkan posisi dan orientasi robot. Algoritma ini melakukan pencarian jalur dengan memperhitungkan keadaan robot dalam koordinat x , y , dan θ , sehingga jalur yang dihasilkan tidak hanya mempertimbangkan jarak antar titik, tetapi juga karakteristik gerak robot dan batasan kinematikanya. Pendekatan tersebut membuat Hybrid A* sesuai untuk robot yang memiliki keterbatasan dalam arah pergerakan atau perubahan orientasi, karena jalur yang dihasilkan dapat dibuat lebih halus dan memungkinkan untuk diikuti oleh robot.

Berdasarkan penelitian Schäfle dan Uchiyama (2021), Hybrid A* dapat digunakan untuk melakukan perencanaan jalur pada robot mobile dengan *nonholonomic constraints* dengan mempertimbangkan ketidakpastian gerakan robot serta kondisi lingkungan.

Penelitian tersebut menunjukkan bahwa pendekatan Hybrid A* dapat menghasilkan jalur yang lebih aman dengan mempertimbangkan kemungkinan terjadinya tabrakan terhadap *obstacle* [6]. Berdasarkan penelitian tersebut, Hybrid A* pada penelitian ini digunakan sebagai *global path planner* untuk menentukan jalur AGV forklift dari posisi awal menuju titik tujuan dengan mempertimbangkan posisi, orientasi, serta keterbatasan gerak AGV, sehingga jalur yang dihasilkan dapat digunakan sebagai acuan bagi proses navigasi robot.

Panjang jalur global yang dihasilkan oleh Hybrid A* dihitung dari penjumlahan jarak antar titik berurutan pada path menggunakan persamaan (2.15) berikut.

$$L_{\text{path}} = \sum_{i=1}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (2.15)$$

Keterangan:

L_{path} : panjang total jalur global (m);

N : jumlah titik penyusun jalur;

x_i dan y_i : koordinat titik jalur ke- i (m);

x_{i+1} dan y_{i+1} : koordinat titik jalur berikutnya (m).

Panjang jalur menunjukkan efisiensi rute yang dipilih perencana untuk sepasang titik awal dan titik tujuan tertentu. Nilai ini tidak dapat dibaca sendirian, karena jalur terpendek belum tentu paling baik apabila diperoleh dengan melewati area yang sangat dekat dengan rintangan atau dengan menambah manuver mundur. Dalam penelitian ini panjang jalur dibandingkan antarpengulangan pada pasangan titik yang sama untuk menilai konsistensi hasil perencanaan.

Beban komputasi perencana dinyatakan sebagai waktu perencanaan pada Persamaan (2.16).

$$t_{\text{plan}} = t_{\text{path}} - t_{\text{request}} \quad (2.16)$$

Keterangan:

t_{plan} : waktu perencanaan jalur (s);

t_{request} : waktu permintaan perencanaan dikirimkan (s);

t_{path} : waktu jalur hasil perencanaan tersedia (s).

Waktu perencanaan mencerminkan besarnya ruang pencarian yang harus ditelusuri algoritma, sehingga nilainya meningkat pada manuver sempit atau pada jarak tujuan yang jauh. Besaran ini penting karena perencana bekerja pada batas waktu tertentu; apabila

batas tersebut terlampaui, permintaan perencanaan dinyatakan gagal dan AGV tidak bergerak. Dalam penelitian ini waktu perencanaan dilaporkan bersama panjang jalur agar dapat dinilai apakah lintasan yang lebih pendek diperoleh dengan mengorbankan waktu komputasi.

Kesesuaian lintasan terhadap batas manuver kendaraan dinilai dari hubungan antara kelengkungan jalur dan radius belok pada Persamaan (2.17).

$$R = \frac{1}{|\kappa|} \quad (2.17)$$

Keterangan:

R : radius kelengkungan lintasan (m);

κ : kelengkungan lintasan pada titik yang ditinjau (1/m).

Hubungan berbanding terbalik tersebut menunjukkan bahwa kelengkungan yang besar berarti radius belok yang kecil, yaitu belokan yang tajam. Karena kendaraan berkemudi Ackermann memiliki radius belok minimum, lintasan yang mengandung kelengkungan melebihi batas tersebut tidak dapat diikuti meskipun bebas rintangan. Dalam penelitian ini hubungan tersebut menjadi dasar pemeriksaan apakah radius belok minimum yang dikonfigurasi pada rencana benar-benar konsisten dengan kemampuan kemudi fisik AGV. Ketiga besaran pada Persamaan (2.15) sampai Persamaan (2.17) karena itu saling melengkapi, yaitu menilai efisiensi rute, beban komputasi, dan keterlaksanaan lintasan. Parameter tersebut menjadi dasar pengujian SmacPlannerHybrid pada beberapa variasi start-goal dan tuning minimum turning radius.

2.11. *Reeds-Shepp*

Reeds-Shepp merupakan metode untuk menghasilkan jalur bagi robot beroda yang memiliki kendala *nonholonomic*, yaitu robot tidak dapat bergerak bebas ke segala arah. Metode ini mempertimbangkan posisi awal dan posisi tujuan robot yang meliputi koordinat x , y , serta orientasi θ , dengan memungkinkan robot melakukan gerakan maju dan mundur. Jalur *Reeds-Shepp* dibentuk dari kombinasi segmen garis lurus dan gerakan berbelok dengan radius tertentu, sehingga dapat digunakan untuk menghasilkan lintasan yang sesuai dengan kemampuan manuver robot. Kemampuan bergerak maju dan mundur tersebut menjadi salah satu keunggulan *Reeds-Shepp* dibandingkan pendekatan yang

hanya mengizinkan pergerakan maju, terutama ketika robot harus melakukan manuver pada ruang yang sempit.

Berdasarkan penelitian Dang et al. (2022), kurva *Reeds-Shepp* digunakan pada tahap *analytic expansion* dalam *Hybrid A** untuk meningkatkan kecepatan dan ketepatan pencarian jalur pada robot *nonholonomic*. Penelitian tersebut menunjukkan bahwa penggunaan *Reeds-Shepp* dapat membantu menghubungkan posisi robot menuju konfigurasi tujuan, tetapi jalur yang dihasilkan berpotensi terlalu dekat dengan *obstacle* sehingga perlu mempertimbangkan aspek keselamatan dalam pemilihan jalur [22]. Berdasarkan penelitian tersebut, Reeds-Shepp pada penelitian ini digunakan sebagai pembentuk jalur pada Hybrid A* untuk menghasilkan lintasan yang mempertimbangkan posisi dan orientasi AGV forklift serta memungkinkan kombinasi gerakan maju dan mundur, sehingga jalur yang dihasilkan lebih sesuai dengan karakteristik manuver AGV.

2.12. Navigation2 (Nav2)

Navigation 2 (Nav2) merupakan navigation stack pada *Robot Operating System 2* (ROS 2) yang digunakan untuk mengatur proses navigasi robot secara terintegrasi. Nav2 menyediakan berbagai komponen yang mendukung proses navigasi, seperti global planner, local controller, costmap, behavior, serta pengelolaan tujuan navigasi. Sistem ini memungkinkan robot menggunakan informasi peta, data sensor, dan estimasi posisi untuk menentukan serta mengikuti jalur menuju titik tujuan sambil mempertimbangkan keberadaan obstacle. Nav2 juga memiliki struktur yang modular sehingga algoritma perencanaan jalur dan pengendalian gerak dapat disesuaikan dengan karakteristik robot yang digunakan.

Berdasarkan penelitian Horelican (2022), Navigation2/ROS 2 dapat digunakan untuk membangun sistem navigasi robot pada lingkungan industri dengan mengintegrasikan komponen localization, path planning, dan pengendalian pergerakan robot. Penelitian tersebut menunjukkan bahwa Nav2 memiliki modularitas dan fleksibilitas yang mendukung integrasi berbagai sensor serta algoritma navigasi untuk kebutuhan robot mobile [23]. Berdasarkan penelitian tersebut, Nav2 pada penelitian ini digunakan sebagai kerangka navigasi AGV forklift yang mengintegrasikan AMCL sebagai localization, Hybrid A* sebagai global planner, serta komponen pengendalian dan

costmap untuk membantu AGV bergerak menuju titik tujuan dengan mempertimbangkan kondisi lingkungan.

Kinerja navigasi secara menyeluruh dinilai dari waktu yang dibutuhkan sejak tujuan dieksekusi sampai dinyatakan tercapai, sebagaimana dinyatakan pada Persamaan (2.18).

$$t_{nav} = t_{goal\ reached} - t_{start} \quad (2.18)$$

Keterangan:

- t_{nav} : waktu navigasi total (s);
 t_{start} : waktu eksekusi tujuan navigasi dimulai (s);
 $t_{goal\ reached}$: waktu tujuan dinyatakan tercapai (s).

Besaran tersebut mencakup seluruh proses yang terjadi selama navigasi, yaitu pengikutan jalur, manuver, perlambatan, serta penyesuaian terhadap rintangan. Karena itu waktu navigasi tidak dapat dibandingkan secara langsung antarlintasan yang berbeda panjang, melainkan dibandingkan antarpengulangan pada lintasan yang sama untuk menilai konsistensi. Dalam penelitian ini nilai tersebut juga dipakai untuk menelusuri pengaruh pemicuan zona keselamatan terhadap durasi navigasi.

Keandalan sistem pada sekumpulan percobaan dinyatakan sebagai tingkat keberhasilan pada Persamaan (2.19).

$$SR = \frac{N_{berhasil}}{N_{total}} \times 100\% \quad (2.19)$$

Keterangan:

- SR : tingkat keberhasilan navigasi (%);
 $N_{berhasil}$: jumlah percobaan yang mencapai tujuan sesuai kriteria;
 N_{total} : jumlah seluruh percobaan navigasi.

Persamaan waktu navigasi dan success rate digunakan bersama untuk menggambarkan performa navigasi AGV secara *end-to-end*. Waktu navigasi menunjukkan durasi yang dibutuhkan sistem sejak goal dieksekusi hingga tercapai, sedangkan success rate menunjukkan konsistensi keberhasilan dari seluruh percobaan.

Nav2 pada penelitian ini mengintegrasikan map server, AMCL, global/local costmap, SmacPlannerHybrid, MPPI Controller, velocity smoother, behavior, BT Navigator, dan Collision Monitor. Source menggunakan readiness gate sehingga

komponen tidak diaktifkan hanya berdasarkan delay waktu, tetapi menunggu sensor, lifecycle, map, dan transformasi TF dinyatakan siap

2.13. *Costmap 2D*

Costmap 2D merupakan representasi lingkungan berbentuk grid dua dimensi yang digunakan dalam sistem navigasi robot untuk menggambarkan kondisi ruang di sekitar robot berdasarkan tingkat keterjangkauan atau risiko terhadap *obstacle*. Setiap sel pada *grid* memiliki nilai biaya (*cost*) yang menunjukkan kondisi area tersebut, mulai dari area bebas hingga area yang dianggap tidak dapat dilalui. Dalam Nav2, *Costmap 2D* dapat memanfaatkan informasi dari sensor seperti LiDAR untuk memperbarui keberadaan *obstacle* secara langsung. *Costmap* juga dapat memiliki beberapa *layer*, seperti *static layer* untuk informasi dari peta, *obstacle layer* untuk mendeteksi *obstacle* berdasarkan sensor, dan *inflation layer* untuk memberikan area aman di sekitar *obstacle*. Dengan demikian, *costmap* membantu *planner* dan *controller* menentukan jalur yang tidak hanya menuju tujuan, tetapi juga mempertimbangkan jarak aman terhadap *obstacle*.

Berdasarkan penelitian Sani et al. (2024), *costmap* pada Nav2 dapat diperbarui secara *real-time* dengan menggabungkan informasi sensor untuk menyesuaikan representasi lingkungan terhadap kondisi aktual. Penelitian tersebut menunjukkan bahwa pembaruan *local costmap* dapat meningkatkan kemampuan robot dalam menghadapi kondisi lingkungan yang tidak sepenuhnya direpresentasikan oleh peta statis, sehingga robot dapat menentukan perilaku navigasi yang lebih sesuai [24]. Berdasarkan penelitian tersebut, *Costmap 2D* pada penelitian ini digunakan sebagai representasi kondisi lingkungan yang mendukung proses navigasi AGV forklift, khususnya dengan memanfaatkan data LiDAR 2D untuk memperbarui informasi *obstacle* di sekitar robot. Informasi tersebut kemudian digunakan oleh komponen Nav2 dalam proses perencanaan dan pengendalian jalur agar AGV dapat bergerak menuju tujuan dengan mempertimbangkan keberadaan *obstacle*.

2.14. *Model Predictive Path Integral (MPPI)*

Model Predictive Path Integral (MPPI) merupakan metode kontrol dan perencanaan lintasan lokal berbasis sampling yang digunakan untuk menentukan input kendali robot berdasarkan prediksi beberapa kemungkinan lintasan pada beberapa langkah ke depan. MPPI mengevaluasi sejumlah kandidat lintasan berdasarkan fungsi biaya yang dapat

mempertimbangkan kesalahan terhadap jalur referensi, jarak terhadap obstacle, kecepatan, dan batasan gerak robot. Dari kandidat tersebut, sistem memilih atau memberikan bobot lebih besar pada lintasan dengan biaya yang lebih rendah untuk menentukan perintah kendali berikutnya. Pendekatan ini memungkinkan MPPI menangani sistem robot yang memiliki dinamika dan batasan gerak yang kompleks serta melakukan penyesuaian lintasan secara langsung selama robot bergerak.

Berdasarkan penelitian Trivedi et al. (2025), MPPI berbasis sampling dapat digunakan untuk menggabungkan path following dan obstacle avoidance pada robot bergerak dengan mempertimbangkan ketidakpastian model gerak. Penelitian tersebut menunjukkan bahwa pendekatan MPPI mampu menghasilkan pelacakan lintasan dan penghindaran obstacle secara lebih baik pada pengujian simulasi maupun robot nyata [25]. Berdasarkan penelitian tersebut, MPPI pada penelitian ini digunakan sebagai local controller dalam sistem Nav2 untuk mengendalikan pergerakan AGV forklift agar dapat mengikuti jalur global yang telah direncanakan oleh Hybrid A* sekaligus menyesuaikan pergerakannya terhadap kondisi lokal dan obstacle yang terdeteksi selama navigasi.

2.15. Collision Monitor

Collision Monitor merupakan komponen pada sistem navigasi ROS 2 Nav2 yang digunakan untuk meningkatkan keselamatan robot dengan memantau kondisi di sekitar robot secara terus-menerus berdasarkan data sensor. Komponen ini bekerja dengan mendeteksi keberadaan obstacle pada area tertentu di sekitar robot dan membandingkannya dengan batas keselamatan yang telah ditentukan. Berdasarkan kondisi tersebut, Collision Monitor dapat memberikan respons seperti membatasi kecepatan atau menghentikan pergerakan robot ketika obstacle berada pada jarak yang berpotensi menyebabkan tabrakan. Dengan demikian, Collision Monitor berfungsi sebagai lapisan pengaman tambahan yang bekerja secara langsung terhadap kondisi lingkungan dan dapat membantu mengurangi risiko tabrakan selama robot menjalankan proses navigasi.

Berdasarkan penelitian Sani et al. (2024), sistem navigasi ROS 2 yang menggunakan informasi sensor secara real-time dapat memperbarui representasi lingkungan dan meningkatkan kemampuan robot dalam menghadapi obstacle yang tidak terdapat pada peta awal [24]. Berdasarkan penelitian tersebut, *Collision Monitor* pada

penelitian ini digunakan sebagai lapisan keselamatan tambahan pada navigasi AGV forklift dengan memanfaatkan data LiDAR 2D untuk memantau jarak terhadap obstacle secara real-time. Ketika obstacle terdeteksi pada zona keselamatan yang telah ditentukan, sistem dapat melakukan pembatasan atau penghentian pergerakan AGV sehingga proses navigasi tetap memperhatikan aspek keselamatan.

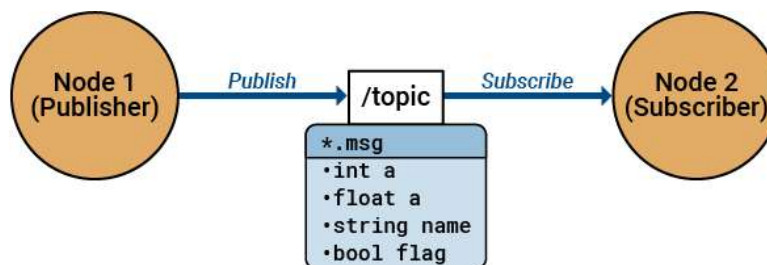
2.16. *Robot Operating System 2 (ROS 2)*

Robot Operating System 2 (ROS 2) yaitu pengembangan lanjutan dari ROS yang dirancang untuk meningkatkan performa, fleksibilitas, serta keandalan sistem robotika, terutama pada aplikasi *real-time* dan sistem terdistribusi. ROS 2 menggunakan *middleware* berbasis *Data Distribution Service (DDS)* yang memungkinkan komunikasi data antar node menjadi lebih stabil, aman, dan mendukung *Quality of Service (QoS)*, sehingga sangat sesuai untuk sistem robot mobile seperti *Automated Guided Vehicle (AGV)* [26]. Dalam arsitektur ROS 2, sistem dibangun secara modular menggunakan *node-node* yang saling terhubung melalui mekanisme komunikasi berbasis *publisher-subscriber*, *service*, dan *action*. Pendekatan ini memungkinkan setiap komponen seperti sensor, aktuator, serta algoritma navigasi dapat bekerja secara terpisah namun tetap terintegrasi dalam satu sistem. Dibandingkan dengan ROS versi sebelumnya, ROS 2 memiliki keunggulan dalam hal komunikasi *real-time*, manajemen *multi-threading*, serta dukungan terhadap sistem operasi yang lebih luas, termasuk sistem *embedded* [26].

Dalam sistem navigasi robot, *Robot Operating System 2 (ROS 2)* menyediakan *framework* yang mendukung integrasi perangkat keras dan perangkat lunak, termasuk abstraksi perangkat keras, *device drivers*, komunikasi antar proses, serta *tools* untuk pengujian dan visualisasi. Pada penelitian yang dilakukan oleh Wan Zakaria et al. (2022), ROS digunakan sebagai *framework* dalam pengembangan sistem navigasi otonom yang mengintegrasikan sensor, odometri, SLAM, *path planning*, *obstacle avoidance*, serta sistem kontrol dan aktuasi. Data dari odometri dan sensor diproses melalui ROS *navigation stack* untuk menghasilkan keluaran berupa perintah kecepatan yang diteruskan ke sistem aktuasi kendaraan. Penelitian tersebut juga menunjukkan bahwa arsitektur ROS memungkinkan setiap komponen sistem navigasi berkomunikasi melalui *nodes* dan *topics*, sehingga proses pemetaan, perencanaan jalur, penghindaran rintangan, dan pengendalian kendaraan dapat diintegrasikan dalam satu sistem [26].

Dalam penerapannya pada sistem navigasi otonom, ROS mendukung pembagian proses perencanaan jalur menjadi *global planning* dan *local planning*. *Global planning* digunakan untuk menentukan lintasan kendaraan secara keseluruhan menuju posisi tujuan berdasarkan informasi lingkungan, sedangkan *local planning* bekerja pada area di sekitar kendaraan untuk menyesuaikan pergerakan terhadap perubahan kondisi lingkungan dan menghindari rintangan. Penelitian Wan Zakaria et al. (2022) menerapkan pendekatan tersebut pada sistem navigasi kendaraan otonom dengan memanfaatkan peta *occupancy grid* yang diperoleh dari SLAM sebagai dasar perencanaan jalur. Hasil penelitian menunjukkan bahwa sistem berbasis ROS dapat mengintegrasikan SLAM, *path planning*, *obstacle avoidance*, dan aktuasi sehingga kendaraan mampu menentukan lintasan serta melakukan penyesuaian pergerakan terhadap rintangan selama navigasi [26].

Selain itu, ROS 2 mendukung integrasi berbagai sensor seperti LiDAR, kamera, dan encoder dalam satu sistem yang terkoordinasi. Data dari sensor tersebut diproses secara paralel untuk menghasilkan informasi lingkungan yang akurat, yang kemudian digunakan dalam proses navigasi. Dukungan terhadap komunikasi *real-time* pada ROS 2 memungkinkan sistem untuk merespons perubahan lingkungan dengan lebih cepat dan stabil, terutama pada kondisi dinamis. Penelitian menunjukkan bahwa penggunaan *framework* berbasis ROS dalam sistem navigasi robot mampu meningkatkan efisiensi pengembangan serta mempermudah integrasi antara algoritma SLAM, estimasi posisi, dan perencanaan jalur. Dengan peningkatan fitur pada ROS 2, sistem navigasi menjadi lebih andal, skalabel, serta siap untuk implementasi pada sistem robot otonom di lingkungan industri [27].



Gambar 2. 7 *Publisher Subscriber ROS 2* [28]

Gambar 2.16 memperlihatkan mekanisme komunikasi tersebut, dengan node penerbit mengirimkan pesan pada sebuah topic dan node pelanggan menerimanya tanpa keduanya perlu saling mengetahui secara langsung. ROS 2 dipilih sebagai *framework*

utama dalam penelitian ini karena mampu menyediakan mekanisme komunikasi antarkomponen sistem secara terstruktur dan mendukung integrasi berbagai perangkat serta proses dalam sistem navigasi robot. Dalam sistem yang dikembangkan, ROS 2 digunakan untuk mengintegrasikan sensor LiDAR, data odometri, proses pemetaan menggunakan SLAM, lokalisasi menggunakan AMCL, serta perencanaan jalur global menggunakan Hybrid A* melalui Nav2 *SmacPlannerHybrid*. Data yang diperoleh dari sensor dan proses navigasi dipertukarkan melalui mekanisme komunikasi berbasis *publisher-subscriber* dengan pengaturan *Quality of Service* (QoS) yang disesuaikan dengan kebutuhan masing-masing data. Hasil pemetaan dan lokalisasi digunakan sebagai dasar dalam proses perencanaan jalur menuju titik tujuan, sedangkan informasi sensor digunakan untuk mendukung pengenalan kondisi lingkungan dan penghindaran rintangan. Dengan integrasi tersebut, ROS 2 berperan sebagai *framework* utama yang menghubungkan proses persepsi, pemetaan, lokalisasi, perencanaan jalur, dan pengendalian pergerakan AGV sehingga seluruh proses navigasi dapat berjalan secara terkoordinasi.

Dalam sistem ROS 2, data LiDAR, IMU, dan odometri memiliki frekuensi publikasi yang berbeda sehingga waktu penerimaan setiap data tidak selalu sama. Oleh karena itu, sinkronisasi data perlu diperhatikan agar informasi sensor yang digunakan dalam proses navigasi merepresentasikan kondisi AGV pada waktu yang relatif berdekatan. Evaluasi sinkronisasi pada penelitian ini dilakukan dengan mengamati timestamp, kontinuitas publikasi, *freshness*, dan keterlambatan data masing-masing sensor. Data yang terlambat atau tidak diperbarui dalam periode tertentu dapat menunjukkan adanya gangguan komunikasi yang berpotensi memengaruhi proses pemetaan, lokalisasi, maupun navigasi.