

**SISTEM NAVIGASI OTONOM BERBASIS FSM DAN VISUAL
SERVOING UNTUK PENCARIAN DAN PENDEKATAN
OBJEK PADA MOBILE ROBOT**



Disusun Oleh:

N a m a : Wildan Syaifudin Ahmad
NIM : 21523222

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2026**

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**SISTEM NAVIGASI OTONOM BERBASIS FSM DAN VISUAL
SERVOING UNTUK PENCARIAN DAN PENDEKATAN
OBJEK PADA MOBILE ROBOT**

TUGAS AKHIR



Yogyakarta, 28 April 2026

Pembimbing,

(Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D.)

HALAMAN PENGESAHAN DOSEN PENGUJI

SISTEM NAVIGASI OTONOM BERBASIS FSM DAN VISUAL SERVOING UNTUK Pencarian dan Pendekatan Objek pada MOBILE ROBOT

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana

di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 19 Mei 2026

Tim Penguji

Chandra Kusuma Dewa, S.Kom., M.Cs.,
Ph.D.

Anggota 1

Dr. Nur Wijyaning Rahayu, S.Kom.,
M.Cs.

Anggota 2

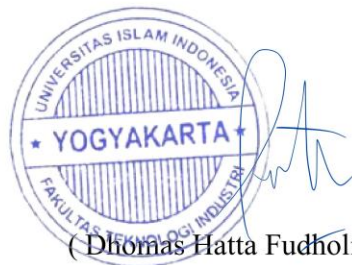
Elyza Gustri Wahyuni, S.T., M.Cs.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia



(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Wildan Syaifudin Ahmad

NIM : 21523222

Tugas akhir dengan judul:

SISTEM NAVIGASI OTONOM BERBASIS FSM DAN VISUAL SERVOING UNTUK PENCARIAN DAN PENDEKATAN OBJEK PADA MOBILE ROBOT

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 25 April 2026



(Wildan Syaifudin Ahmad)

HALAMAN PERSEMBAHAN

Assalamualaikum Warahmatullahi Wabarakatuh.

Alhamdulillah, puji syukur penulis panjatkan ke hadirat Allah SWT karena atas rida dan kemudahan-Nya, laporan tugas akhir ini dapat diselesaikan. Karya ini penulis persembahkan pertama-tama kepada orang tua dan keluarga tercinta. Terima kasih atas segala doa, dukungan, serta kepercayaan yang telah diberikan selama ini sehingga penulis dapat menyelesaikan masa studi dengan baik.

Selanjutnya, penulis persembahkan karya ini kepada program studi Informatika Universitas Islam Indonesia yang telah menjadi tempat bagi penulis dalam menuntut ilmu dan mengembangkan diri. Terima kasih kepada seluruh dosen dan civitas akademika atas bimbingan serta wawasan yang telah diberikan selama masa perkuliahan. Penulis juga menyampaikan apresiasi kepada rekan-rekan mahasiswa dan sahabat seperjuangan yang telah memberikan dukungan serta berbagi pengalaman selama proses penyusunan skripsi ini.

Penulis berharap hasil penelitian ini dapat memberikan manfaat bagi pengembangan ilmu pengetahuan dan menjadi langkah awal yang bermanfaat bagi perjalanan hidup penulis selanjutnya.

Wassalamualaikum Warahmatullahi Wabarakatuh.

HALAMAN MOTO

“Dan bahwa manusia hanya memperoleh apa yang telah diusahakannya, dan sesungguhnya usahanya itu kelak akan diperlihatkan (kepadanya)”

(QS. An-Najm: 39-40)

“Maka sesungguhnya bersama kesulitan ada kemudahan. Sesungguhnya bersama kesulitan ada kemudahan”

(QS. Al-Insyirah:5-6)

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah Subhanahu wa Ta'ala atas segala limpahan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi ini dengan baik. Penyusunan skripsi ini dilakukan sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Dalam proses penyusunan dan penyelesaian skripsi ini, penulis menyadari bahwa banyak tantangan dan keterbatasan yang dihadapi. Namun, berkat bantuan, bimbingan, serta dukungan dari berbagai pihak, kesulitan tersebut dapat teratasi. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang tulus kepada:

1. Allah Subhanahu wa Ta'ala, atas segala kemudahan dan rida-Nya yang senantiasa menyertai setiap langkah penulis hingga tugas akhir ini dapat dirampungkan.
2. Bapak Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D., selaku dosen pembimbing yang telah meluangkan waktu, memberikan arahan, serta bimbingan yang sangat berharga selama proses penyusunan skripsi ini.
3. Keluarga tercinta, yang selalu memberikan doa, semangat, dan dukungan yang luar biasa kepada penulis dalam menempuh pendidikan hingga tahap akhir ini.
4. Reiky Aryanando Pratama, S.Kom., yang telah banyak membantu, memberikan dukungan moral, serta menemani penulis selama proses pengerjaan skripsi ini berlangsung.
5. Seluruh teman-teman dan pihak terkait, yang tidak dapat penulis sebutkan satu per satu, atas segala bantuan dan motivasi yang telah diberikan selama masa perkuliahan.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, penulis sangat mengharapkan kritik maupun saran yang membangun demi perbaikan di masa mendatang. Akhir kata, penulis berharap semoga karya ini dapat memberikan manfaat bagi pengembangan ilmu pengetahuan dan berguna bagi pembaca sekalian.

Yogyakarta, 28 April 2026



(Wildan Syaifudin Ahmad)

SARI

Sistem navigasi pada Autonomous Mobile Robot (AMR) saat ini masih sering mengalami kendala fragmentasi fungsional, di mana robot kerap berhenti setelah mendeteksi objek karena ketidakmampuan melakukan transisi yang mulus menuju manuver pendekatan fisik. Penelitian ini bertujuan mengembangkan sistem robot mobile otonom berbasis Finite State Machine (FSM) untuk mengintegrasikan fase pencarian, deteksi, dan pendekatan presisi ke dalam satu kerangka kerja yang terkoordinasi secara utuh. Sistem dibangun pada lingkungan simulasi ROS 2 Gazebo menggunakan algoritma Nav2 untuk navigasi global, YOLO11n untuk persepsi visual, pelacak sentroid, serta Image-Based Visual Servoing (IBVS). Pengujian eksperimental dilakukan melalui 40 uji coba untuk membandingkan performa strategi Random Exploration dan Waypoint-Based Search. Hasil penelitian menunjukkan arsitektur FSM berhasil mencapai tingkat keberhasilan misi 100% dan terbukti tangguh dengan kemampuan self-recovery yang secara otomatis mampu memulihkan robot berulang kali dari kondisi terjebak hambatan (deadlock). Evaluasi kuantitatif menyimpulkan bahwa Waypoint-Based Search lebih efisien untuk target di sekitar jalur patroli terstruktur (50,55 detik), sedangkan Random Exploration unggul pada lokasi target tak terduga (87,00 detik). Temuan ini menegaskan bahwa efisiensi sistem AMR sangat bergantung pada kecocokan antara strategi pencarian yang digunakan dengan karakteristik letak target di dalam ruangan.

Kata kunci: *Autonomous Mobile Robot*, Finite State Machine, Visual Servoing, YOLO11n, *Autonomous Search*

GLOSARIUM

AMCL	(Adaptive Monte Carlo Localization) algoritma lokalisasi robot yang menggunakan particle filter untuk memperkirakan posisi robot pada peta grid.
Centroid Tracker	algoritma pelacakan objek yang bekerja dengan menghitung titik pusat masa dari bounding box hasil deteksi.
Costmap	representasi peta dalam bentuk grid biaya yang digunakan oleh sistem navigasi untuk menentukan jalur aman dan menghindari hambatan.
Deadlock	kondisi di mana sistem atau robot terhenti sepenuhnya dan tidak dapat melanjutkan eksekusi program akibat konflik instruksi.
Differential Drive	jenis penggerak robot yang menggunakan dua roda independen untuk mengontrol arah dan kecepatan melalui perbedaan putaran masing-masing roda.
FSM	(Finite State Machine) model kontrol logika yang mengatur perilaku robot melalui sejumlah keadaan (state) yang terbatas dan aturan transisi tertentu.
IBVS	(Image-Based Visual Servoing) metode pengendalian robot yang menggunakan fitur visual 2D langsung dari citra kamera sebagai umpan balik kontrol.
LiDAR	(Light Detection and Ranging) sensor jarak laser yang memindai lingkungan untuk mendeteksi rintangan dan membantu navigasi robot.
Nav2	(Navigation2) kumpulan framework navigasi standar pada ROS 2 untuk perencanaan jalur dan penghindaran halangan.
Occupancy Grid Map	representasi peta ruangan dalam bentuk sel grid yang berisi informasi status area (kosong, terisi halangan, atau tidak diketahui).
ROS 2	(Robot Operating System 2) kumpulan framework perangkat lunak tingkat menengah (middleware) untuk pengembangan aplikasi robotika.

SLAM	(Simultaneous Localization and Mapping) teknik yang memungkinkan robot untuk membangun peta lingkungan sekaligus menentukan posisi dirinya secara bersamaan.
URDF	(Unified Robot Description Format) format berkas berbasis XML yang digunakan untuk mendeskripsikan struktur fisik dan visual dari sebuah robot.
Visual Servoing	pendekatan pengendalian gerak robot yang menggunakan informasi visual dari sensor kamera untuk memandu robot menuju target.
Waypoint	titik koordinat tujuan yang telah ditentukan sebelumnya sebagai acuan rute navigasi robot.
YOLO	(You Only Look Once) arsitektur jaringan saraf tiruan untuk deteksi objek secara real-time dalam satu kali proses inferensi citra.

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING.....	ii
HALAMAN PENGESAHAN DOSEN PENGUJI.....	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR	vii
SARI	viii
GLOSARIUM.....	ix
DAFTAR ISI.....	xi
DAFTAR TABEL.....	xiii
DAFTAR GAMBAR.....	xiv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah.....	3
1.4 Tujuan Penelitian.....	4
1.5 Manfaat Penelitian.....	5
1.6 Metodologi Penelitian	5
1.7 Sistematika Penulisan.....	6
1.7.1 Pendahuluan	7
1.7.2 Tinjauan Pustaka	7
1.7.3 Metodologi Penelitian	7
1.7.4 Hasil dan Pembahasan.....	7
1.7.5 Kesimpulan dan Saran.....	7
BAB II TINJAUAN PUSTAKA	8
2.1 Autonomous Mobile Robot.....	8
2.2 ROS2 dan Nav2.....	9
2.3 SLAM dan AMCL.....	10
2.4 Deteksi Objek dan YOLO11	11
2.5 Object Tracking.....	13
2.6 Visual Servoing	16
2.7 Finite State Machine (FSM).....	16
2.8 Analisis Kajian	17
BAB III METODOLOGI PENELITIAN	21
3.1 Spesifikasi Penelitian	21
3.2 Perancangan Sistem	22
3.2.1 Arsitektur Sistem.....	23
3.2.2 Perancangan Finite State Machine (FSM).....	24
3.2.3 Perancangan Strategi Pencarian	26
3.3 Menyiapkan Lingkungan Simulasi.....	30
3.3.1 Konfigurasi Robot (URDF).....	30
3.3.2 Lingkungan Simulasi Gazebo	31
3.3.3 Pemetaan (SLAM Toolbox)	32
3.3.4 Navigasi dan Lokalisasi (Nav2 dan AMCL).....	33
3.4 Implementasi Sistem	34
3.4.1 Deteksi Objek (YOLO11n Pretrained).....	34

3.4.2	Object Tracking (Centroid Tracker).....	35
3.4.3	Visual Servoing (IBVS)	35
3.4.4	Mekanisme Safety Stop.....	36
3.4.5	GUI Pemilihan Target	36
3.4.6	Pencatatan Metrik Otomatis	37
3.5	Skenario Pengujian.....	38
3.5.1	Desain Eksperimen.....	38
3.5.2	Metrik Evaluasi	40
3.5.3	Prosedur Pelaksanaan Pengujian	41
	BAB IV HASIL DAN PEMBAHASAN.....	47
4.1	Implementasi dan Validasi Sistem	47
4.1.1	Validasi Deteksi Objek (YOLO11n).....	47
4.1.2	Validasi Alur FSM	49
4.1.3	Validasi Visual Servoing dan Centroid Tracker.....	51
4.2	Hasil Eksperimen	52
4.2.1	Parameter Statistik yang Digunakan	53
4.2.2	Hasil Eksperimen Posisi Target 1 (map_fix1).....	54
4.2.3	Hasil Eksperimen Posisi Target 2 (map_fix2).....	55
4.2.4	Rekapitulasi Keseluruhan.....	56
4.3	Perbandingan Strategi	56
4.3.1	Perbandingan Time to Find.....	57
4.3.2	Perbandingan Total Time	58
4.3.3	Perbandingan Path Length	59
4.3.4	Perbandingan Safety Stops.....	60
4.4	Pembahasan dan Temuan Penting.....	61
4.4.1	Analisis Self-Recovery dan Robustness Sistem	61
4.4.2	Hubungan Strategi dengan Tata Letak Lingkungan	62
4.4.3	Perbandingan Karakteristik Kedua Strategi	63
4.5	Evaluasi Sistem	64
4.5.1	Kelebihan Sistem	64
4.5.2	Keterbatasan Sistem	65
4.6	Pembahasan	66
4.7	Komparasi dengan Penelitian Terdahulu.....	67
4.7.1	Aspek Navigasi dan Eksplorasi	67
4.7.2	Aspek Deteksi dan Pelacakan Objek.....	67
4.7.3	Aspek Manuver Pendekatan (Visual Servoing)	68
	BAB V KESIMPULAN DAN SARAN	69
5.1	Kesimpulan.....	69
5.2	Saran.....	70
	DAFTAR PUSTAKA	72
	LAMPIRAN	77

DAFTAR TABEL

Tabel 2.1 Analisis Kajian Penelitian Terdahulu.....	18
Tabel 3.1 Tabel Spesifikasi Perangkat Keras.....	22
Tabel 3.2 Tabel Spesifikasi Perangkat Lunak.....	22
Tabel 3.3 Tabel Daftar ROS 2 Node.....	24
Tabel 3.4 Tabel Deskripsi State FSM.....	25
Tabel 3.5 Tabel Transisi State FSM.....	26
Tabel 3.6 Tabel Komponen Robot.....	30
Tabel 3.7 Tabel Komponen Nav2 yang Digunakan.....	34
Tabel 3.8 Tabel Kolom Metrik pada File CSV.....	37
Tabel 3.9 Tabel Variabel Penelitian.....	38
Tabel 3.10 Tabel Parameter Eksperimen.....	38
Tabel 3.11 Tabel Metrik Evaluasi.....	41
Tabel 4.1 Hasil Eksperimen Posisi Target 1.....	54
Tabel 4.2 Hasil Eksperimen Posisi Target 2.....	55
Tabel 4.3 Tabel Rekapitulasi (Mean $\pm$ SD).....	56
Tabel 4.4 Tabel Perbandingan Karakteristik Strategi Pencarian.....	63

DAFTAR GAMBAR

Gambar 2.1 Plot skema dari (a) one-stage detectors dan (b) two-stage detectors.....	12
Gambar 2.2 Arsitektur model YOLO11	13
Gambar 2.3 Rumus Persamaan Centroid Tracker	14
Gambar 2.4 Centroid Tracking	14
Gambar 2.5 Rumus Persamaan Euclidean Distance	15
Gambar 2.6 Euclidean Distance	15
Gambar 3.1 Tahapan Penelitian	21
Gambar 3.2 Diagram blok arsitektur sistem.....	23
Gambar 3.3 Diagram state FSM.....	25
Gambar 3.4 Pseudocode Algoritma Random Exploration	27
Gambar 3.5 Pseudocode Algoritma Waypoint-Based Search.....	28
Gambar 3.6 Ilustrasi 5 posisi waypoint pada peta (center + 4 sudut)	29
Gambar 3.7 Model robot di Gazebo/RViz	31
Gambar 3.8 Perintah Penginstalan Gazebo	31
Gambar 3.9 Screenshot lingkungan Gazebo (a) map_fix1 dan (b) map_fix2.....	32
Gambar 3.10 Perintah Penginstalan SLAM Toolbox.....	32
Gambar 3.11 Hasil peta OccupancyGrid dari (a) map_fix1 dan (b) map_fix2	33
Gambar 3.12 Perintah Penginstalan ros-galactic-navigation2 dan ros-galactic-nav2-bringup	34
Gambar 3.13 Perintah Penginstalan Pustaka Ultralytics	35
Gambar 3.14 <i>Screenshot</i> GUI pemilihan target	36
Gambar 3.15 Denah peta dengan 2 posisi target yang ditandai	40
Gambar 3.16 Perintah menjalankan lingkungan simulasi	42
Gambar 3.17 Perintah menjalankan RViz.....	42
Gambar 3.18 Tampilan RViz	43
Gambar 3.19 Perintah menjalankan lingkungan simulasi	43
Gambar 3.20 Tampilan RViz ketika telah berhasil dilakukan lokalisasi	44
Gambar 3.21 Perintah menjalankan sistem navigasi Nav2	44
Gambar 3.22 Perintah menjalankan sistem navigasi Nav2	44
Gambar 3.23 Perintah menjalankan sistem navigasi Nav2 dengan strategi random exploration	45
Gambar 3.24 Perintah menjalankan sistem navigasi Nav2 dengan strategi waypoint based exploration.....	45

Gambar 4.1 YOLO11n mendeteksi kotak halangan sebagai kelas lain disertai confidence score yang rendah	48
Gambar 4.2 YOLO11n mendeteksi objek target person dengan bounding box dan label kelas yang akurat	49
Gambar 4.3 YOLO11n mendeteksi target sehingga Nav2 dibatalkan dan sistem bertransisi dari fase SEARCHING ke APPROACHING.....	50
Gambar 4.4 Log terminal yang menampilkan bukti transisi terjadi tanpa jeda.....	50
Gambar 4.5 Tampilan kamera robot ketika subscribe topic /inference_result ...	52
Gambar 4.6 Tampilan kamera robot ketika subscribe topic /annotated_image	52
Gambar 4.7 Rumus Perhitungan Mean	53
Gambar 4.8 Rumus Perhitungan Standar Deviasi (SD)	53
Gambar 4.9 Bar Chart Perbandingan Time to Find Keempat Kombinasi.....	57
Gambar 4.10 Bar Chart Perbandingan Total Time	58
Gambar 4.11 Bar Chart Perbandingan Path Length.....	59
Gambar 4.12 Bar Chart Perbandingan Safety Stops	60
Gambar 4.13 Visualisasi lintasan robot di RViz ketika robot melakukan safety stop	62

BAB I

PENDAHULUAN

1.1 Latar Belakang

Autonomous Mobile Robot (AMR) adalah sistem robot yang dirancang untuk beroperasi di lingkungan yang tidak sepenuhnya diketahui atau dapat diprediksi. Robot jenis ini harus mampu bernavigasi dan menghindari rintangan di area pergerakannya dengan sedikit atau bahkan tanpa campur tangan manusia, baik di lingkungan dalam ruangan maupun luar ruangan (Alatise & Hancke, 2020). Dalam sepuluh tahun terakhir, penggunaan AMR telah meluas ke berbagai sektor, mulai dari lingkungan domestik, logistik pergudangan, hingga industri. Menurut Keith dan La (2024), keunggulan utama AMR terletak pada kemampuannya untuk melakukan navigasi, deteksi objek, dan manipulasi secara mandiri tanpa bergantung pada infrastruktur tetap. Gonzalez-Aguirre dkk. (2021) dan Siciliano dkk. (2009) juga mencatat bahwa kebutuhan akan aplikasi robotic service di fasilitas kesehatan, gudang, dan smart home terus meningkat seiring dengan upaya untuk meningkatkan efisiensi operasional dan mengurangi ketergantungan pada tenaga kerja manusia.

Perkembangan penggunaan AMR ini juga didorong oleh kemajuan teknologi pendukungnya. Alatise dan Hancke (2020) mencatat bahwa kemajuan pada teknologi sensor, daya komputasi, dan perangkat lunak menjadi faktor utama yang memperluas kemampuan mobile robot di berbagai bidang. Secara umum, AMR memerlukan integrasi dari beberapa komponen teknologi utama agar dapat beroperasi secara otonom. Komponen-komponen tersebut meliputi sistem navigasi untuk perencanaan jalur, sistem persepsi visual untuk mengenali objek di lingkungan, serta mekanisme object tracking untuk menjaga identitas target yang telah terdeteksi antarframe (Keith & La, 2024; Moksyakov dkk., 2024). Saat ini, ekosistem Robot Operating System 2 (ROS 2) menyediakan kerangka kerja yang memungkinkan integrasi komponen-komponen tersebut dalam satu platform (Macenski dkk., 2023). Dari berbagai penelitian yang telah dikaji, misi pencarian objek oleh AMR secara umum melibatkan beberapa fungsi utama yang bersifat sekuensial, seperti eksplorasi lingkungan untuk mencari objek (Gargouri dkk., 2025; Ma dkk., 2020), deteksi dan identifikasi objek target (Chung & Nguyen, 2023; Gargouri dkk., 2025) serta pendekatan presisi ke posisi objek (Dai & Lee, 2025; Ma dkk., 2020). Meskipun masing-masing komponen teknologi telah menunjukkan kemajuan yang baik secara individual, integrasi seluruh fungsi tersebut ke dalam satu sistem yang terkoordinasi masih menghadapi sejumlah tantangan.

Beberapa penelitian sebelumnya telah berupaya mengintegrasikan sebagian dari fungsi-fungsi tersebut, namun belum ada yang mencakup seluruh fungsi dari pencarian hingga pendekatan secara utuh. Penelitian yang mengintegrasikan pencarian dan deteksi objek (Gargouri dkk., 2025) belum dilengkapi mekanisme pendekatan ke target; setelah objek berhasil dideteksi, algoritma pencarian hanya mempublikasikan lokasi objek pada peta dan berhenti, tanpa melanjutkan manuver pendekatan ke posisi objek. Sebaliknya, penelitian yang menggabungkan deteksi dan object tracking (Chung & Nguyen, 2023) tidak memiliki fase pencarian otonom, sehingga robot hanya dapat berfungsi jika target sudah berada dalam jangkauan kamera sejak awal. Penelitian lain yang mengintegrasikan pencarian dan pendekatan dalam arsitektur FSM (Ma dkk., 2020) masih menggunakan deteksi berbasis template matching, bukan deep learning, serta hanya menerapkan satu strategi pencarian (quasi-random) tanpa evaluasi komparatif. Penelitian yang menerapkan pendekatan presisi berbasis multi-sensor (Dai & Lee, 2025) terbatas pada manuver docking ke marker statis (AprilTag), di mana robot sudah diarahkan ke area target terlebih dahulu dan tidak melakukan eksplorasi lingkungan. Pada lingkup institusional, pengujian deteksi objek pada simulasi ROS (Ananda & Dewa, 2023; Tamadjoe & Dewa, 2024) juga belum menerapkan object tracking maupun pencarian otonom. Secara keseluruhan, penelitian terdahulu umumnya hanya mengimplementasikan sebagian dari fungsi yang diperlukan, sehingga belum membentuk satu sistem yang mampu menjalankan misi pencarian objek secara utuh dari tahap pencarian hingga pendekatan.

Berdasarkan kajian tersebut, terdapat dua celah penelitian utama. Pertama, belum terdapat sistem yang mengintegrasikan seluruh fungsi operasional, yaitu pencarian otonom, deteksi objek berbasis deep learning, dan pendekatan presisi dalam satu arsitektur yang terkoordinasi menggunakan FSM. Setiap sistem yang dikaji hanya mencakup sebagian fungsi ada yang berhenti setelah mendeteksi objek tanpa mendekatinya (Gargouri dkk., 2025), ada yang mengasumsikan target sudah terlihat tanpa tahap pencarian (Chung & Nguyen, 2023; Dai & Lee, 2025), dan ada yang sudah mengintegrasikan pencarian dan pendekatan namun belum memanfaatkan deteksi berbasis deep learning (Ma dkk., 2020). Integrasi yang utuh diperlukan agar robot dapat menjalankan misi pencarian objek secara mandiri dari awal hingga akhir tanpa bergantung pada intervensi manual di setiap transisi antar tahap. Kedua, dari kajian yang telah ditinjau, belum ditemukan perbandingan kuantitatif antara strategi eksplorasi, seperti random exploration dan waypoint-based search, dalam konteks sistem pencarian objek terintegrasi pada robot mobile tunggal di lingkungan indoor yang telah dipetakan. Perbandingan ini penting

untuk menyediakan dasar empiris dalam memilih strategi pencarian yang lebih efektif serta menjadi baseline bagi pengembangan penelitian selanjutnya.

Oleh karena itu, penelitian ini mengembangkan sistem robot mobile otonom berbasis finite state machine (FSM) yang mengintegrasikan tahap pencarian, deteksi, dan pendekatan objek dalam satu sistem yang terkoordinasi. Sistem dibangun menggunakan ROS 2 dengan memanfaatkan Nav2 sebagai navigation stack, YOLO11n untuk deteksi objek, serta visual servoing untuk mendekati target secara presisi pada jarak dekat. Selain itu, sistem dilengkapi centroid tracker untuk menjaga kontinuitas identitas objek target antarframe deteksi. Selain merancang dan mengimplementasikan sistem tersebut, penelitian ini juga membandingkan dua strategi pencarian, yaitu random exploration dan waypoint-based search, untuk memperoleh gambaran performa masing-masing pendekatan pada lingkungan simulasi indoor.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, rumusan masalah yang diangkat pada penelitian ini adalah sebagai berikut.

1. Bagaimana merancang dan mengimplementasikan autonomous mobile robot (AMR) yang mampu mencari, mendeteksi, dan mendekati objek target secara otonom melalui koordinasi tahap-tahap tersebut menggunakan arsitektur finite state machine (FSM) pada lingkungan simulasi indoor?
2. Bagaimana perbandingan kinerja antara strategi pencarian berbasis random exploration (di mana robot memilih arah gerak secara acak) dan waypoint-based search (di mana robot mengunjungi titik-titik yang sudah ditentukan secara urut), diukur berdasarkan metrik waktu penemuan (time to find), waktu pencapaian (time to reach), waktu total misi (total time), panjang lintasan (path length), jumlah safety stop, dan status keberhasilan misi (result)?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini dibatasi pada beberapa aspek tertentu agar penelitian tetap relevan dengan tujuan yang ingin dicapai. Adapun batasan masalah pada penelitian ini adalah sebagai berikut.

1. Pengujian dilakukan sepenuhnya dalam lingkungan simulasi Gazebo menggunakan framework ROS 2 Galactic Geochelone, sehingga hasil yang diperoleh belum tentu mencerminkan kinerja pada perangkat keras fisik.

2. Lingkungan pengujian berupa peta yang sudah di-generate sebelumnya menggunakan SLAM. Robot tidak melakukan pemetaan secara bersamaan saat beroperasi, melainkan menggunakan peta yang sudah tersedia dan melakukan lokalisasi menggunakan AMCL.
3. Model deteksi objek yang digunakan adalah YOLO11 dengan bobot pretrained COCO (Ultralytics) tanpa dilakukan pelatihan ulang (fine-tuning). Kelas objek target yang tersedia terbatas pada kelas yang terdapat dalam dataset COCO sesuai kebutuhan skenario pengujian.
4. Objek target bersifat statis dan tidak bergerak selama proses pencarian maupun pendekatan.
5. Lingkungan pengujian berupa ruangan indoor tunggal dengan tata letak tetap dan halangan yang tidak berubah.
6. Perbandingan strategi pencarian hanya dilakukan pada dua pendekatan, yaitu random exploration dan waypoint-based search.
7. Pada fase pendekatan objek (APPROACHING), yang menggunakan visual servoing dan object tracking, robot tidak menggunakan Nav2 navigation stack untuk menghindari halangan. Sebagai gantinya, digunakan mekanisme keamanan berbasis pemindaian LiDAR sederhana (safety stop) yang melakukan manuver mundur dan rotasi saat halangan terdeteksi dalam jarak kritis.
8. Robot hanya dapat mencari satu kelas objek target dalam satu siklus perintah. Untuk mencari objek dengan kelas berbeda, pengguna perlu memberikan perintah baru melalui GUI.

1.4 Tujuan Penelitian

1. Merancang dan mengimplementasikan autonomous mobile robot (AMR) dengan pipeline operasional terintegrasi berbasis FSM untuk pencarian, deteksi, dan pendekatan objek target pada lingkungan simulasi indoor menggunakan framework ROS 2.
2. Membandingkan efektivitas dua strategi pencarian, yaitu random exploration dan waypoint-based search, berdasarkan metrik kuantitatif yang dicatat secara otomatis oleh sistem, meliputi waktu penemuan target (time to find), waktu pencapaian target (time to reach), waktu total misi (total time), panjang lintasan (path length), jumlah safety stop, dan status keberhasilan misi (result).

1.5 Manfaat Penelitian

1. Bagi Penulis
 - a. Menambah pengetahuan dan keterampilan dalam mengembangkan sistem robotika otonom berbasis ROS 2, khususnya dalam mengintegrasikan computer vision dengan navigasi robot mobile.
 - b. Memperluas pemahaman mengenai arsitektur FSM dan koordinasi multikomponen dalam pipeline operasional robot otonom.
2. Bagi Komunitas Akademik dan Peneliti
 - a. Menyediakan data perbandingan kuantitatif antara strategi eksplorasi berbasis random exploration dan waypoint-based search dalam konteks pipeline pencarian, deteksi, dan pendekatan objek pada single robot mobile di lingkungan indoor yang sudah dipetakan.
 - b. Memberikan referensi arsitektur sistem modular yang dapat ditiru dan dikembangkan untuk penelitian AMR selanjutnya.
3. Bagi Pengembang Sistem Robotika
 - a. Menunjukkan pipeline operasional lengkap yang dapat diadaptasi untuk keperluan nyata seperti robot patroli, service robot, atau robot pencari di lingkungan indoor.
 - b. Mengidentifikasi keterbatasan implementasi, khususnya terkait penghindaran halangan pada fase pendekatan presisi, sebagai bahan pertimbangan untuk perbaikan di penelitian selanjutnya.

1.6 Metodologi Penelitian

Pelaksanaan penelitian ini dibagi ke dalam beberapa tahapan. Uraian singkat untuk setiap tahapannya adalah sebagai berikut.

- a. Studi Literatur

Tahap pertama adalah mengidentifikasi celah penelitian melalui kajian literatur yang berkaitan dengan deteksi objek, object tracking, navigasi otonom, dan visual servoing pada robot mobile. Kebutuhan sistem kemudian dirumuskan berdasarkan celah yang ditemukan.
- b. Perancangan Sistem

Pada tahap ini, ditentukan spesifikasi perangkat keras dan perangkat lunak yang digunakan, serta dirancang arsitektur pipeline terintegrasi yang mencakup desain FSM

beserta definisi state dan aturan transisinya, pemilihan komponen utama (YOLO11, centroid tracker, Nav2, visual servoing), perancangan mekanisme koordinasi antar modul, dan perancangan dua strategi pencarian objek yang akan dibandingkan.

c. Menyiapkan Lingkungan Simulasi

Lingkungan simulasi disiapkan menggunakan Gazebo, mencakup konfigurasi robot (URDF) dengan sensor kamera RGB dan LiDAR, pembuatan lingkungan simulasi indoor, pemetaan menggunakan SLAM Toolbox, serta konfigurasi navigasi dan lokalisasi menggunakan Nav2 dan AMCL.

d. Implementasi Sistem

Setiap modul, yaitu deteksi objek (YOLO11n), object tracking (centroid tracker), visual servoing, mekanisme safety stop, GUI pemilihan target, dan pencatatan metrik otomatis, diimplementasikan secara terpisah menggunakan Python pada framework ROS 2 Galactic dan dikoordinasikan oleh FSM utama.

e. Pengujian dan Pengumpulan Data

Pengujian dilakukan melalui skenario pencarian objek dengan dua strategi pencarian yang berbeda pada dua posisi target. Metrik kinerja dicatat secara otomatis ke file CSV pada setiap trial, mencakup waktu penemuan (time to find), waktu pencapaian (time to reach), waktu total misi (total time), panjang lintasan (path length), jumlah safety stop, dan status keberhasilan misi.

f. Analisis Data dan Pembahasan

Data yang terkumpul dianalisis secara komparatif antara kedua strategi pencarian. Evaluasi dilakukan dengan membandingkan nilai rata-rata (Mean) dan standar deviasi (Standard Deviation/SD) tiap metrik utama, serta mengidentifikasi kelebihan dan keterbatasan masing-masing pendekatan berdasarkan pola performa yang teramati.

g. Penarikan Kesimpulan

Berdasarkan hasil analisis, ditarik kesimpulan mengenai efektivitas pipeline terintegrasi serta perbandingan performa kedua strategi pencarian untuk menjawab rumusan masalah penelitian.

1.7 Sistematika Penulisan

Sistematika penulisan ini bertujuan untuk memastikan alur penelitian berjalan secara konsisten dari awal hingga akhir. Pembahasan dalam penelitian ini dibagi ke dalam beberapa bab dengan rincian sebagai berikut.

1.7.1 Pendahuluan

Bab ini membahas latar belakang penelitian, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, serta sistematika penulisan.

1.7.2 Tinjauan Pustaka

Bab ini berisi tinjauan pustaka yang digunakan dalam penelitian, mencakup autonomous mobile robot (AMR), deteksi objek berbasis deep learning (YOLO), object tracking, navigasi otonom (ROS 2, Nav2), visual servoing, finite state machine, serta kajian terhadap penelitian terdahulu yang relevan.

1.7.3 Metodologi Penelitian

Bab ini menjelaskan metode yang digunakan dalam penelitian, meliputi perancangan arsitektur sistem, implementasi pada ROS 2 dan Gazebo, desain skenario eksperimen, serta metode pengumpulan dan analisis data.

1.7.4 Hasil dan Pembahasan

Bab ini menyajikan hasil validasi komponen sistem, data kuantitatif dari pengujian eksperimen, perbandingan performa antara strategi pencarian, analisis metrik kinerja, serta evaluasi kelebihan dan keterbatasan sistem yang ditemukan.

1.7.5 Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil penelitian serta saran untuk pengembangan sistem di masa mendatang.

BAB II

TINJAUAN PUSTAKA

2.1 Autonomous Mobile Robot

Autonomous Mobile Robot (AMR) adalah sistem robot yang dirancang untuk beroperasi di lingkungan yang tidak sepenuhnya diketahui atau dapat diprediksi. Robot jenis ini harus mampu bernavigasi dan menghindari rintangan di area pergerakannya dengan sedikit atau bahkan tanpa campur tangan manusia, baik di lingkungan dalam ruangan maupun luar ruangan (Alatise & Hancke, 2020). Menurut Keith dan La (2024), AMR umumnya dibangun dari tiga komponen utama, yaitu persepsi (perception), lokalisasi (localization), dan navigasi (navigation).

Persepsi merupakan kemampuan robot untuk memperoleh pengetahuan tentang lingkungannya. Menurut Siegwart dkk. (Siegwart dkk., 2011), hal ini dilakukan dengan mengambil pengukuran dari berbagai sensor dan kemudian mengekstrak informasi yang bermakna dari pengukuran tersebut. Sensor yang digunakan pada AMR secara umum seperti LiDAR, sensor ultrasonik, dan kamera (Alatise & Hancke, 2020). Lokalisasi adalah proses penentuan posisi dan orientasi robot di dalam lingkungan kerjanya. Siegwart dkk. (2011) menempatkan lokalisasi sebagai salah satu blok pembangun utama navigasi, karena robot harus terlebih dahulu mengetahui posisi dirinya di lingkungan sebelum dapat merencanakan pergerakan. Untuk mendukung proses lokalisasi, telah dikembangkan teknik-teknik seperti Simultaneous Localization and Mapping (SLAM), yang merupakan teknik untuk mengestimasi posisi robot sekaligus membangun peta lingkungan secara bertahap (Gao & Zhang, 2021), serta Adaptive Monte Carlo Localization (AMCL), yang mengestimasi posisi robot dengan melakukan sampling terhadap sejumlah asumsi posisi dan kemudian mengevaluasi serta memperbarui asumsi-asumsi tersebut secara iteratif (Zhu & Luo, 2024).

Navigasi merupakan salah satu kompetensi paling menantang yang dibutuhkan oleh robot mobile (Siegwart dkk., 2011). Secara umum, navigasi mencakup kemampuan robot untuk bertindak berdasarkan pengetahuan dan nilai sensornya guna mencapai posisi tujuan seefisien dan sehandal mungkin. Menurut Siegwart dkk. (2011), keberhasilan navigasi memerlukan empat komponen: persepsi untuk menginterpretasi data sensor, lokalisasi untuk menentukan posisi robot, kognisi untuk memutuskan tindakan yang harus diambil, serta kendali gerak (motion control) untuk mengeksekusi lintasan yang diinginkan. Pada sebagian besar kasus, robot tidak dapat mengambil jalur langsung dari titik awal ke tujuan karena adanya halangan,

sehingga diperlukan teknik perencanaan gerak (motion planning) yang mengintegrasikan informasi dari komponen persepsi dan lokalisasi.

2.2 ROS2 dan Nav2

Robot Operating System (ROS) merupakan kumpulan framework perangkat lunak yang dirancang khusus untuk pengembangan aplikasi robotika. ROS berfungsi sebagai middleware yang menghubungkan aplikasi perangkat lunak dengan sistem operasi, serta menyediakan layanan-layanan standar seperti hardware abstraction, kontrol perangkat keras tingkat rendah, message-passing, dan manajemen paket (Deng dkk., 2017).

ROS 2 merupakan generasi kedua dari ROS yang dirancang ulang untuk mengatasi keterbatasan versi pertamanya. ROS 2 dibangun di atas standar komunikasi Data Distribution Service (DDS) yang memungkinkan dukungan terhadap keamanan, sistem real-time, komunikasi multi-robot, serta operasi pada kondisi jaringan yang tidak ideal (Macenski dkk., 2022). Sebagai platform bersifat open source dengan lisensi Apache 2.0, ROS 2 mendukung berbagai kebutuhan aplikasi robotika, mulai dari riset hingga pengembangan produk. Ekosistem perangkat lunaknya terdiri dari tiga kategori utama: middleware untuk komunikasi antarkomponen, algoritma bawaan seperti persepsi, SLAM, dan perencanaan gerak, serta perangkat pengembang (developer tools) untuk konfigurasi, visualisasi, debugging, dan simulasi (Macenski dkk., 2022).

Komunikasi antarkomponen dalam ROS 2 dilakukan melalui tiga mekanisme utama. Node adalah entitas dasar yang bertanggung jawab mengendalikan perangkat atau menjalankan proses tertentu. Topic merupakan aliran data, seperti kecepatan motor atau data sensor, yang memungkinkan node saling bertukar informasi melalui mekanisme publish-subscribe. Sementara itu, service memungkinkan komunikasi dua arah antara node, di mana satu node menyediakan layanan dan node lain dapat mengirim permintaan serta menerima respons (Ruman Talukder dkk., 2024).

Framework navigasi yang dikembangkan di atas ekosistem ROS 2 adalah Navigation2 (Nav2). Nav2 merupakan framework navigasi standar untuk ROS 2 yang menyediakan arsitektur modular untuk persepsi, lokalisasi, perencanaan jalur, dan kendali gerak (Macenski dkk., 2020). Nav2 mengkoordinasikan proses navigasi melalui behaviour tree (BT) yang mengatur beberapa server asinkron berbasis plug-in, sementara lifecycle manager mengawasi proses inisialisasi dan transisi status setiap node. Untuk beroperasi, Nav2 memerlukan rantai transformasi koordinat (TF) standar, yang mencakup transformasi dari frame peta ke frame

odometri untuk lokalisasi global, serta transformasi dari frame odometri ke frame basis robot untuk estimasi posisi lokal.

Dalam merepresentasikan lingkungan, Nav2 menggunakan dua costmap. Global costmap digunakan untuk perencanaan jalur jarak jauh dan umumnya dihasilkan dari peta statis (misalnya dari SLAM) dengan penambahan area inflasi di sekitar halangan. Local costmap diperbarui pada frekuensi yang lebih tinggi di sekitar robot untuk mendukung penghindaran halangan secara reaktif. Proses navigasi ini dibagi antara planner server dan controller server: planner menghitung jalur dengan biaya terendah menuju tujuan menggunakan global costmap, sedangkan controller mengubah jalur tersebut menjadi perintah kecepatan pada topik `/cmd_vel` sambil memanfaatkan local costmap untuk eksekusi yang halus dan bebas tabrakan (Macenski dkk., 2023). Apabila perencanaan atau eksekusi jalur gagal, Nav2 dapat menjalankan perilaku pemulihan (recovery behaviours) seperti berputar, menunggu, atau mundur, yang dikelola oleh BT navigator (Macenski dkk., 2020).

Pada penelitian ini, ROS 2 Galactic digunakan sebagai platform utama untuk mengintegrasikan seluruh komponen sistem robot. Setiap komponen, yaitu deteksi objek, object tracking, navigasi, dan visual servoing, diimplementasikan sebagai node yang saling berkomunikasi melalui mekanisme topic dan service. Arsitektur berbasis node ini memungkinkan setiap modul dikembangkan dan diuji secara independen, sebelum dikoordinasikan oleh finite state machine (FSM) utama. Sementara itu, Nav2 dimanfaatkan sebagai navigation stack pada fase pencarian objek, di mana robot mengirimkan goal position ke Nav2 sesuai strategi pencarian yang dipilih (random exploration atau waypoint-based search), dan Nav2 merencanakan serta mengeksekusi jalur menuju titik tersebut secara otonom.

2.3 SLAM dan AMCL

Simultaneous Localization and Mapping (SLAM) adalah teknik yang memungkinkan robot mengestimasi posisinya sekaligus membangun peta lingkungan secara bertahap (Gao & Zhang, 2021). Selama beberapa dekade, algoritma lokalisasi dan navigasi berbasis SLAM telah menjadi topik penelitian yang aktif, dengan berbagai pendekatan yang diusulkan dan diuji pada beragam skenario (Qiao dkk., 2024). Secara umum, algoritma SLAM terdiri dari dua komponen utama. Komponen pemetaan (mapping) membangun representasi lingkungan menggunakan data sensor seperti LiDAR, kamera, atau sensor sonar. Komponen lokalisasi mengestimasi posisi robot di dalam peta yang telah dibangun, umumnya menggunakan teknik probabilistik seperti particle filter (Qiao dkk., 2024). Algoritma SLAM telah berhasil

diterapkan pada berbagai aplikasi, termasuk kendaraan otonom, navigasi dalam ruangan, dan misi eksplorasi (Ruman Talukder dkk., 2024).

Dalam ekosistem ROS 2, salah satu implementasi SLAM yang banyak digunakan adalah SLAM Toolbox. SLAM Toolbox dipilih secara luas karena merupakan solusi SLAM 2D yang terawat dengan dukungan komunitas aktif, terintegrasi dengan Nav2, dan dapat digunakan dengan konfigurasi minimal pada berbagai platform (Macenski & Jambrecic, 2021). Formulasi 2D yang digunakan oleh SLAM Toolbox mampu menghasilkan peta dan lokalisasi yang andal pada lingkungan indoor maupun lingkungan kerja industrial.

Setelah peta tersedia, robot memerlukan mekanisme untuk menentukan posisinya di dalam peta tersebut secara real-time saat beroperasi. Adaptive Monte Carlo Localization (AMCL) merupakan algoritma yang mengestimasi posisi dan orientasi robot secara real-time menggunakan particle filter (Ruman Talukder dkk., 2024). AMCL bekerja dengan melakukan sampling terhadap sejumlah hipotesis posisi robot, kemudian mengevaluasi dan memperbarui hipotesis-hipotesis tersebut secara iteratif berdasarkan data sensor dan model pergerakan robot. AMCL merupakan pengembangan dari algoritma Monte Carlo Localization (MCL) yang menghasilkan akurasi dan ketahanan lokalisasi yang lebih baik (Zhu & Luo, 2024).

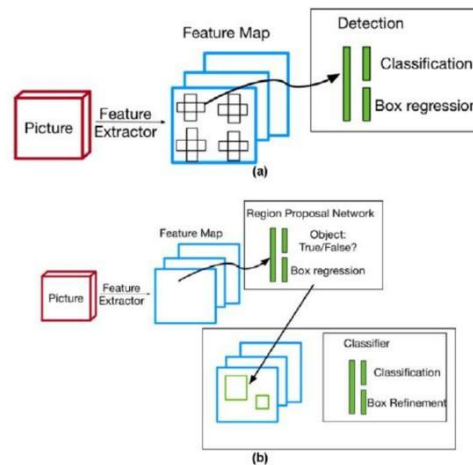
Pada penelitian ini, SLAM Toolbox digunakan pada tahap awal untuk membangun peta lingkungan simulasi. Peta yang dihasilkan kemudian disimpan dan digunakan sebagai referensi tetap selama pengujian. Saat robot beroperasi, AMCL digunakan untuk mengestimasi posisi robot di dalam peta tersebut secara kontinu sehingga Nav2 dapat merencanakan dan mengeksekusi navigasi berdasarkan posisi yang akurat.

2.4 Deteksi Objek dan YOLO11

Deteksi objek merupakan salah satu tugas penting dalam bidang computer vision yang berfokus pada identifikasi dan lokalisasi objek dari kelas tertentu dalam citra digital. Deteksi objek juga menjadi dasar bagi berbagai tugas computer vision lainnya, seperti segmentasi, object tracking, dan image captioning (Zou dkk., 2023).

Pada periode deteksi objek berbasis deep learning, object detector dibagi menjadi dua kelompok utama, yaitu two-stage detector dan one-stage detector (Zou dkk., 2023). Two-stage detector, seperti Faster R-CNN, menggunakan Region Proposal Network (RPN) pada tahap awal untuk mengidentifikasi regions of interest, yang kemudian diproses lebih lanjut untuk klasifikasi dan penyempurnaan bounding box. Pendekatan ini menghasilkan akurasi yang lebih tinggi, tetapi cenderung lebih lambat karena prosesnya bersifat multi-tahap. Sebaliknya, one-

stage detector melakukan deteksi dan klasifikasi secara bersamaan dalam satu langkah langsung dari gambar input, sehingga menghasilkan kecepatan pemrosesan yang jauh lebih tinggi (Zou dkk., 2023). Ilustrasi perbandingan kedua pendekatan tersebut dapat dilihat pada Gambar 2.1.



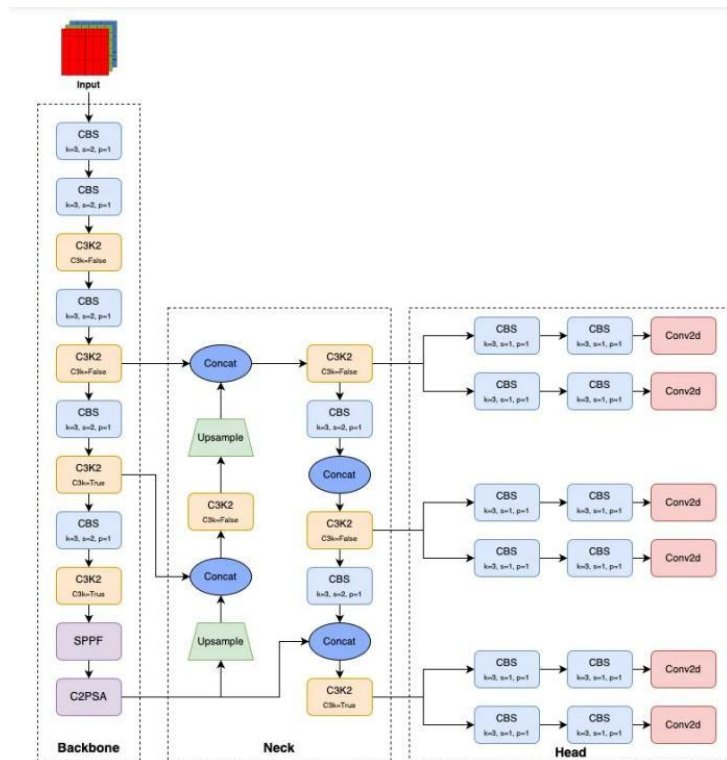
Gambar 2.1 Plot skema dari (a) one-stage detectors dan (b) two-stage detectors

Sumber: (Kemajou dkk., 2019)

YOLO (You Only Look Once) merupakan salah satu one-stage detector yang paling banyak digunakan. Berbeda dengan pendekatan yang menggunakan classifier untuk melakukan deteksi, YOLO memandang deteksi objek sebagai masalah regresi dengan fokus pada prediksi bounding box dan probabilitas kelas secara langsung dari gambar penuh dalam satu langkah menggunakan satu jaringan saraf tunggal. Pendekatan terintegrasi ini memungkinkan pemrosesan secara real-time sambil mempertahankan tingkat presisi yang baik (Redmon dkk., 2016). Sejak diperkenalkan pada tahun 2015, arsitektur YOLO terus mengalami evolusi melalui berbagai versi, dengan setiap generasi meningkatkan keseimbangan antara akurasi deteksi dan efisiensi komputasi (M. L. Ali & Zhang, 2024).

YOLO11 (2024) merupakan salah satu versi terkini dalam seri YOLO yang dikembangkan oleh Ultralytics dengan dasar pengembangan dari YOLOv8. Versi ini menawarkan lima model berskala mulai dari nano hingga extra-large dan mendukung beragam tugas, termasuk deteksi objek, segmentasi, serta estimasi pose. Salah satu peningkatan penting pada YOLO11 adalah pengenalan modul C2PSA (Cross-Stage Partial with Self-Attention), yang menggabungkan keunggulan cross-stage partial networks dengan mekanisme self-attention untuk menangkap informasi kontekstual secara lebih efektif. Selain itu, YOLO11

menggantikan blok C2f dengan C3k2, sebuah implementasi CSP Bottleneck yang menggunakan ukuran kernel lebih kecil untuk mempertahankan akurasi sambil meningkatkan efisiensi (Jegham dkk., 2025). Arsitektur YOLO11 ditunjukkan pada Gambar 2.2.



Gambar 2.2 Arsitektur model YOLO11

Sumber: (Jegham dkk., 2025)

Pada penelitian ini, YOLO11 dipilih karena merupakan arsitektur terkini yang tersedia pada saat pengembangan sistem, dengan peningkatan akurasi dan efisiensi dibanding versi sebelumnya. Detail mengenai pemilihan varian model dan konfigurasi yang digunakan akan dibahas pada Bab III.

2.5 Object Tracking

Object tracking adalah proses memantau perubahan spasial dan temporal suatu objek dalam urutan frame video, mencakup keberadaan, posisi, ukuran, dan bentuk objek tersebut (A. Ali & Mirza, 2006). Inti dari object tracking adalah mengestimasi secara andal keadaan gerak suatu target, seperti lokasi, orientasi, dan ukuran, pada setiap frame dari urutan citra (X. Li dkk., 2013). Object tracking menjembatani jeda antarsiklus inferensi model deteksi sehingga identitas objek yang sudah terdeteksi dapat dipertahankan (Moksyakov dkk., 2024).

Meskipun object tracking telah berkembang pesat, sejumlah tantangan utama masih melekat pada tugas ini, di antaranya oklusi objek, perubahan pencahayaan, gerakan kamera yang cepat, serta variasi bentuk objek (Lee dkk., 2014; X. Li dkk., 2013). Untuk mengatasi tantangan tersebut, berbagai pendekatan telah dikembangkan, mulai dari metode berbasis filter seperti Kalman filter, hingga metode berbasis deep learning seperti DeepSORT dan StrongSORT yang menggabungkan deteksi objek dengan model re-identification untuk mempertahankan identitas objek meskipun terjadi oklusi (Moksyakov dkk., 2024). Namun, pendekatan berbasis deep learning ini umumnya memerlukan daya komputasi yang tinggi dan tidak selalu diperlukan untuk setiap skenario aplikasi.

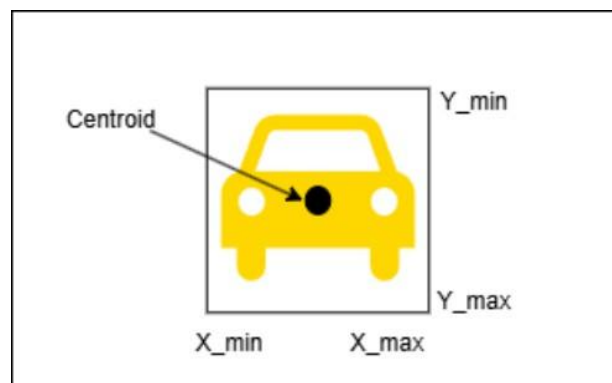
Pada skenario di mana objek target bersifat statis dan lingkungan relatif terkendali, pendekatan lightweight tracker seperti centroid tracker dapat menjadi alternatif yang efisien. Centroid tracker bekerja dengan menghitung titik pusat (centroid) dari setiap bounding box hasil deteksi menggunakan persamaan khusus seperti pada Gambar 2.3 dan Gambar 2.4 (Dharmasaputra & Hartato, 2025).

$$X_{\text{Centroid}} = \frac{x_{\text{Min}} + x_{\text{Max}}}{2}$$

$$Y_{\text{Centroid}} = \frac{y_{\text{Min}} + y_{\text{Max}}}{2}$$

Gambar 2.3 Rumus Persamaan Centroid Tracker

Sumber: (Dharmasaputra & Hartato, 2025)



Gambar 2.4 Centroid Tracking

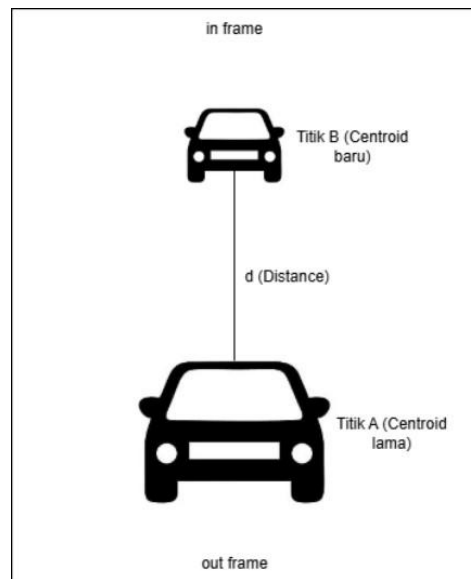
Sumber: (Dharmasaputra & Hartato, 2025)

Pada frame berikutnya, centroid baru yang diperoleh dari hasil deteksi diasosiasikan dengan centroid dari frame sebelumnya berdasarkan jarak Euclidean minimum seperti pada Gambar 2.5 dan Gambar 2.6.

$$d = \sqrt{\{(x_1 - x_2)^2 + (y_1 - y_2)^2\}}$$

Gambar 2.5 Rumus Persamaan Euclidean Distance

Sumber: (Dharmasaputra & Hartato, 2025)



Gambar 2.6 Euclidean Distance

Sumber: (Dharmasaputra & Hartato, 2025)

Jika jarak antar centroid berada di bawah ambang batas tertentu, sistem mengidentifikasi keduanya sebagai objek yang sama sehingga identitas target dapat dipertahankan sepanjang urutan frame. Pendekatan jarak Euclidean dipilih karena sederhana, cepat secara komputasi, dan efektif untuk objek yang bergerak relatif halus, sehingga cocok untuk aplikasi real-time pada perangkat dengan sumber daya terbatas (Dharmasaputra & Hartato, 2025). Apabila suatu objek tidak terdeteksi selama sejumlah frame tertentu, tracker menandai objek tersebut sebagai hilang (disappeared) untuk mengatasi gangguan sementara pada proses deteksi.

Pada penelitian ini, centroid tracker dipilih karena objek target bersifat statis dan lingkungan simulasi relatif terkendali, sehingga tantangan seperti oklusi berkepanjangan dan perubahan penampilan objek yang ekstrem tidak menjadi faktor dominan. Peran utama tracker dalam sistem ini adalah menjaga kesinambungan identitas objek antarframe deteksi, sehingga

FSM tidak salah menginterpretasikan ketiadaan deteksi sesaat sebagai kehilangan target secara permanen.

2.6 Visual Servoing

Visual servoing adalah pendekatan pengendalian robot yang memanfaatkan informasi visual dari kamera untuk memandu pergerakan robot menuju target. Pendekatan ini mengintegrasikan pemrosesan citra, robotika, dan teori kendali untuk mengatur gerak robot berdasarkan informasi yang diekstrak dari citra yang ditangkap oleh satu atau beberapa kamera (Pomares, 2019). Visual servoing telah diaplikasikan secara luas pada berbagai bidang, termasuk navigasi robot mobile, manipulasi objek, dan transportasi cerdas (Pomares, 2019).

Berdasarkan cara data citra digunakan untuk mengendalikan robot, visual servoing diklasifikasikan menjadi tiga kategori utama: Position-Based Visual Servoing (PBVS), Image-Based Visual Servoing (IBVS), dan Hybrid Visual Servoing (HVS) (Y.-R. Li dkk., 2023).

PBVS menggunakan informasi posisi tiga dimensi (3-DOF) yang diestimasi dari citra untuk mengendalikan robot, sehingga memerlukan model geometri objek dan kalibrasi kamera yang akurat. IBVS menggunakan koordinat fitur dua dimensi (2-DOF) secara langsung dari citra untuk kendali closed-loop, tanpa memerlukan estimasi posisi 3D. HVS mengintegrasikan informasi dari kedua pendekatan tersebut. Dibandingkan PBVS, IBVS memiliki toleransi yang lebih besar terhadap kesalahan pemodelan robot, kesalahan kalibrasi kamera, dan noise pada citra. Selain itu, IBVS dapat menghindari kompleksitas komputasi yang tinggi pada HVS, sehingga lebih cocok untuk sistem yang mengutamakan performa real-time (J. Li dkk., 2024).

Pada penelitian ini, pendekatan IBVS digunakan pada fase pendekatan objek. Sistem memanfaatkan fitur dua dimensi dari bounding box hasil deteksi YOLO11, yaitu posisi horizontal centroid dan rasio luas bounding box terhadap luas citra, sebagai umpan balik untuk menghitung perintah kecepatan robot secara langsung. Posisi horizontal centroid digunakan untuk mengoreksi arah hadap robot terhadap target, sedangkan rasio luas bounding box digunakan sebagai estimasi jarak relatif antara robot dan target. Dengan pendekatan ini, robot dapat mendekati objek target secara adaptif tanpa memerlukan estimasi posisi 3D atau kalibrasi kamera yang presisi.

2.7 Finite State Machine (FSM)

Finite State Machine (FSM) merupakan model kontrol yang menggambarkan perilaku sistem melalui sejumlah state (keadaan) yang terbatas, di mana perpindahan antar-state

ditentukan oleh event (kejadian) tertentu dan menghasilkan action (aksi) yang sesuai (Fiqhi & Joni, 2018). FSM banyak digunakan dalam perancangan sistem kontrol karena kesederhanaan komputasinya, kemudahan pemahaman, dan kemudahan dalam menerjemahkan rancangan abstrak menjadi kode program (Rizaldi dkk., 2017).

Dalam konteks robotika, FSM telah diterapkan secara luas sebagai arsitektur pengambilan keputusan, misalnya pada robot sepak bola yang memerlukan perpindahan state antara mencari bola, mendekati bola, dan menendang bola berdasarkan data sensor secara real-time (Fiqhi & Joni, 2018; Putra dkk., 2018). Perpindahan antar-state pada FSM bersifat closed-loop, di mana data sensor digunakan sebagai umpan balik untuk menentukan transisi yang tepat. Apabila pada suatu state sistem tidak lagi memenuhi kondisi tertentu, maka akan terjadi perpindahan ke state yang berbeda untuk menjalankan tindakan yang sesuai (Putra dkk., 2018).

Pada penelitian ini, FSM digunakan sebagai arsitektur utama yang mengoordinasikan seluruh modul sistem robot, termasuk pencarian objek melalui Nav2, deteksi objek dengan YOLO11, object tracking dengan centroid tracker, dan pendekatan objek melalui visual servoing. Robot beroperasi dalam empat state utama, yaitu IDLE (menunggu pemilihan target), SEARCHING (mencari objek menggunakan Nav2), APPROACHING (mendekati objek yang terdeteksi melalui visual servoing yang dilengkapi object tracking), dan MISSION COMPLETE (target berhasil didekati). Transisi antar-state ditentukan oleh event berbasis sensor, seperti terdeteksi atau hilangnya objek target dari frame kamera, serta tercapainya ambang batas jarak tertentu. Pendekatan FSM dipilih dibandingkan arsitektur yang lebih kompleks seperti behaviour tree karena jumlah state pada sistem ini masih terbatas dan alur transisinya bersifat linear, sehingga FSM memberikan keseimbangan yang tepat antara kesederhanaan implementasi dan keandalan koordinasi modul.

2.8 Analisis Kajian

Subbab ini menyajikan tinjauan terhadap hasil penelitian terdahulu yang relevan dengan pengembangan sistem pencarian, deteksi, dan pendekatan objek pada robot mobile otonom. Penelitian-penelitian tersebut dianalisis untuk memperoleh gambaran mengenai pendekatan, metode, serta teknologi yang telah digunakan, sehingga dapat menjadi dasar dalam penentuan arah dan fokus penelitian yang dilakukan.

Tabel 2.1 Analisis Kajian Penelitian Terdahulu

No	Judul Penelitian	Tujuan	Metode	Platform	Hasil	Keterbatasan
1	Design and development of an autonomous mobile robot for unstructured indoor environments (Gargouri dkk., 2025)	Membangun AMR yang dapat mencari dan mendeteksi objek di lingkungan indoor	YOLO11n, Nav2, grid-based search, aplikasi mobile (Flutter)	ROS 2, Raspberry Pi, real robot	Akurasi deteksi 95%, success rate navigasi 95%, ketepatan lokalisasi 3 cm	Tidak ada object tracking antarframe; robot hanya mempublikasikan lokasi objek tanpa melakukan pendekatan; hanya satu strategi pencarian (grid-based)
2	Real-time object detection and tracking for mobile robot using YOLOv8 and Strong SORT (Chung & Nguyen, 2023)	Integrasi deteksi dan tracking untuk person following pada robot differential drive	YOLOv8, StrongSORT, kendali PID	PLC, robot fisik	Mampu mempertahankan tracking ID saat oklusi < 4 detik; lintasan stabil pada kecepatan target < 1 m/s	Tidak ada fase pencarian otonom; target diasumsikan sudah dalam field of view; kontrol PID tanpa navigation stack menyebabkan osilasi lateral
3	Multi-sensor fusion for autonomous mobile robot docking (Dai & Lee, 2025)	Implementasi autonomous docking ke marker statis menggunakan FSM	YOLOv8, depth-aided 3D localization, koreksi orientasi LiDAR, FSM	ROS 2, robot fisik	Konvergensi sudut $\approx 0^\circ$, posisi stabil pada jarak 1,0 m dalam 8,7 detik	Hanya untuk docking ke AprilTag statis; tidak mencakup pencarian objek di lingkungan luas; jenis target terbatas
4	FSM for robot target search and retrieval under semi-	Koordinasi pencarian dan pengambilan objek	FSM, template matching (warna &	ROS, Jackal UGV	Robot berhasil mencari dan mengambil target di	Deteksi berbasis warna/bentuk, bukan deep learning; tidak

No	Judul Penelitian	Tujuan	Metode	Platform	Hasil	Keterbatasan
	constructed environment (Maddik, 2020)	menggunakan FSM	bentuk), VFH		lingkungan semi-unknown	ada object tracking; hanya satu strategi pencarian (quasi-random) tanpa evaluasi komparatif
5	Implementasi robot patroli sederhana berbasis computer vision dan deep learning (Ananda & Dewa, 2023)	Pengujian deteksi objek YOLOv5 pada robot patroli di simulasi ROS	YOLOv5, ROS Noetic	ROS Noetic, Gazebo	Akurasi deteksi manusia 80–90% pada jarak 1–10 m	Tidak ada object tracking; tidak ada pencarian otonom; terjadi misidentifikasi kelas objek
6	Penerapan model deteksi objek untuk robot menggunakan model SSD di lingkungan simulasi ROS (Tamadjoe & Dewa, 2024)	Pengujian model SSD MobileNet V2 untuk deteksi bola dan gawang	SSD MobileNet V2	ROS, Gazebo	mAP 58,6%; akurasi turun dari 63% (1 m) ke 50,3% (>1,35 m); tidak mendeteksi di atas 2,71 m	Tidak ada object tracking; tidak ada pencarian otonom; akurasi deteksi relatif rendah
7	Object detection and tracking with YOLO and the sliding innovation filter	Integrasi deteksi objek YOLO dengan tracking berbasis sliding	YOLO, sliding innovation filter	Non-ROS	Metode SIF terbukti lebih akurat dan stabil dalam melacak objek dibandingkan Kalman Filter, terutama saat terjadi	Fokus hanya pada deteksi dan tracking; tidak ada integrasi dengan navigasi otonom atau

N o	Judul Penelitian	Tujuan	Metode	Platfor m	Hasil	Keterbatasan
	(Moksyakov dkk., 2024)	innovation filter			gangguan berupa hilangnya deteksi pada beberapa frame	pendekatan objek

Berdasarkan Tabel 2.1, penelitian terdahulu yang relevan telah mengeksplorasi berbagai aspek dari sistem robot mobile otonom, mulai dari deteksi objek, object tracking, navigasi, hingga pendekatan objek. Akan tetapi, masing-masing penelitian umumnya hanya mengintegrasikan sebagian tahapan dari keseluruhan pipeline operasional. Penelitian yang menggabungkan pencarian dan deteksi (Gargouri dkk., 2025) belum dilengkapi mekanisme object tracking dan pendekatan presisi. Penelitian yang menerapkan deteksi dan tracking (Chung & Nguyen, 2023) tidak memiliki fase pencarian otonom. Penelitian yang mengkoordinasikan pencarian dan pendekatan menggunakan FSM (Ma dkk., 2020) masih menggunakan deteksi berbasis template matching dan hanya menerapkan satu strategi pencarian tanpa evaluasi komparatif, sementara penelitian yang menerapkan FSM untuk pendekatan presisi (Dai & Lee, 2025) terbatas pada docking ke marker statis tanpa tahap pencarian otonom.

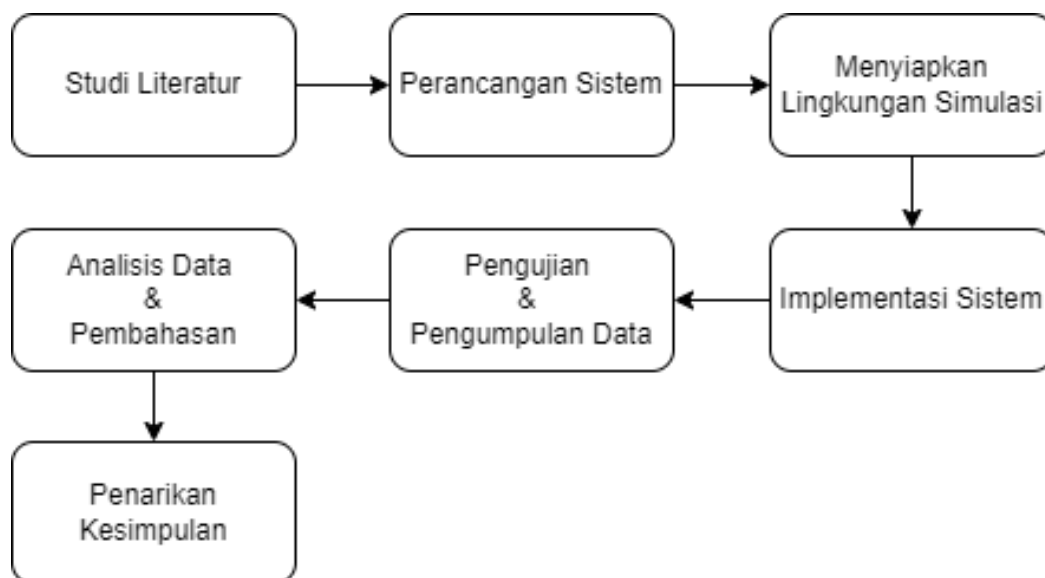
Sehubungan dengan hal tersebut, penelitian ini difokuskan pada pengembangan sistem yang mengintegrasikan seluruh pipeline operasional, yaitu pencarian, deteksi, dan pendekatan objek dalam satu arsitektur berbasis FSM, yang dilengkapi mekanisme object tracking untuk menjaga kontinuitas identitas target. Selain itu, penelitian ini juga membandingkan dua strategi pencarian, yaitu random exploration dan waypoint-based search, untuk menyediakan dasar empiris dalam memilih pendekatan pencarian yang lebih efektif pada lingkungan simulasi indoor.

BAB III

METODOLOGI PENELITIAN

Penelitian ini bertujuan untuk mengembangkan sistem robot mobile otonom yang mampu mencari, mendeteksi, dan mendekati objek target secara otonom di lingkungan simulasi indoor. Sistem ini mengintegrasikan beberapa teknologi utama yaitu ROS 2 Galactic sebagai framework robotika, Nav2 sebagai navigation stack, YOLO11n sebagai detektor objek, centroid tracker sebagai pelacak objek, dan visual servoing sebagai pengendali pendekatan. Seluruh komponen dikoordinasikan oleh sebuah Finite State Machine (FSM).

Penelitian ini akan dilakukan melalui beberapa tahap seperti yang telah diilustrasikan dalam bentuk flowchart pada Gambar 3.1. Tahapan penelitian terdiri dari studi literatur, perancangan sistem, menyiapkan lingkungan simulasi, implementasi sistem, pengujian dan pengumpulan data, analisis data dan pembahasan, dan penarikan kesimpulan.



Gambar 3.1 Tahapan Penelitian

3.1 Spesifikasi Penelitian

Pada pembuatan simulasi robot, terdapat beberapa spesifikasi, baik itu perangkat keras dan perangkat lunak yang digunakan oleh peneliti untuk menjalankan simulasi. Detail spesifikasi dapat dilihat pada Tabel 3.1 dan Tabel 3.2.

Tabel 3.1 Tabel Spesifikasi Perangkat Keras

Komponen	Spesifikasi
Prosesor	Intel Core i5-11400H @ 2,70 GHz (6 core, 12 thread)
RAM	8 GB DDR4
GPU	NVIDIA GeForce RTX 3050
Penyimpanan	512 GB SSD

Tabel 3.2 Tabel Spesifikasi Perangkat Lunak

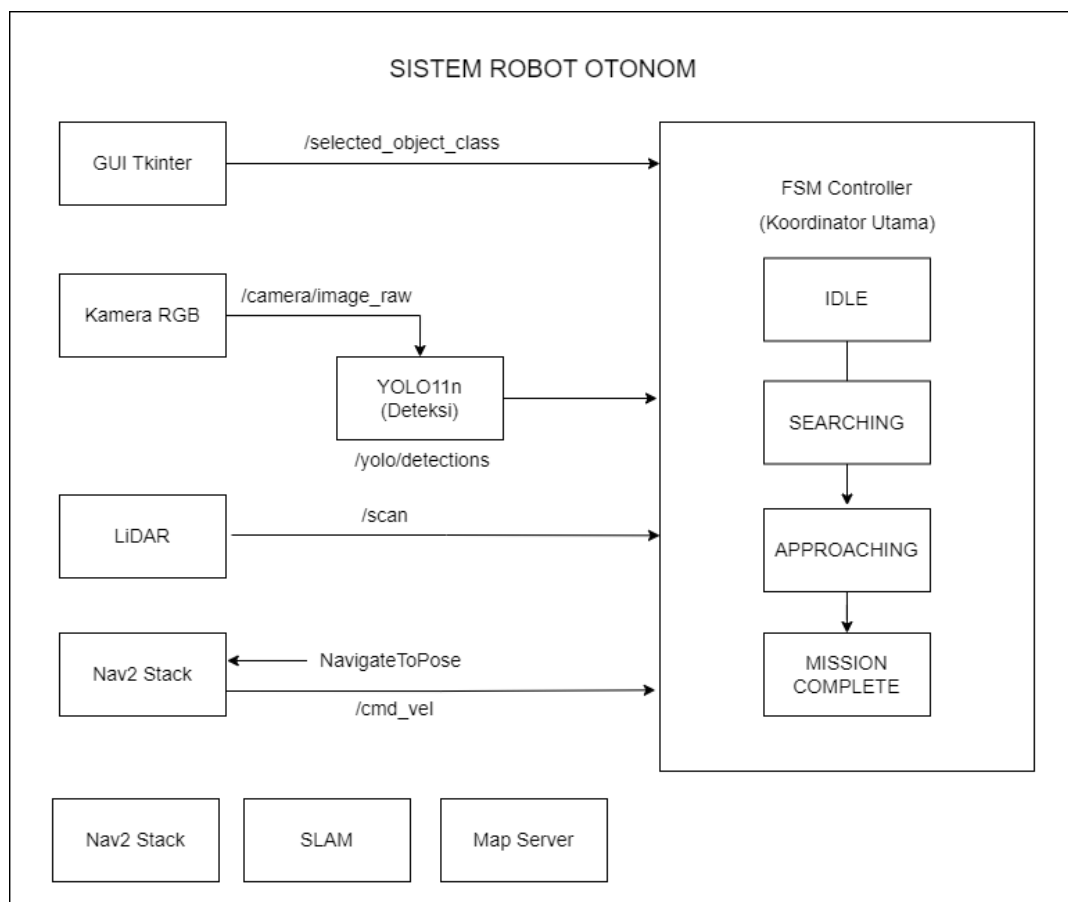
Perangkat Lunak	Versi	Fungsi
Ubuntu	20.04.6 LTS	Sistem Operasi
ROS 2	Galactic Geochelone	Framework robotika
Gazebo	11.15.1 (Classic)	Simulator robot
Nav2	1.0.12	Navigation stack
SLAM Toolbox	2.5.1	Pemetaan lingkungan
Python	3.8.10	Bahasa pemrograman
Ultralytics	8.4.39	Inferensi YOLO11n
OpenCV	4.8.0	Pemrosesan citra
NumPy	1.24.4	Komputasi numerik
Tkinter	-	Graphical User Interface untuk kontrol robot

3.2 Perancangan Sistem

Pada sub-bab ini dijelaskan perancangan sistem yang mencakup tiga aspek utama, yaitu arsitektur sistem secara keseluruhan, perancangan Finite State Machine (FSM) sebagai koordinator antar modul, dan perancangan dua strategi pencarian objek yang akan dibandingkan performanya. Perancangan sistem ini didasarkan pada konsep-konsep yang telah dibahas pada BAB II, khususnya mengenai ROS 2, Nav2, deteksi objek, object tracking, visual servoing, dan FSM.

3.2.1 Arsitektur Sistem

Sistem yang dikembangkan mengintegrasikan beberapa komponen utama dalam ekosistem ROS 2. Pada fase pencarian objek, robot menggunakan Nav2 untuk perencanaan jalur dan penghindaran halangan berdasarkan peta yang telah dibuat sebelumnya. Pada fase pendekatan objek, robot beralih ke visual servoing yang mengendalikan pergerakan robot secara langsung berdasarkan umpan balik visual dari kamera. Peralihan antar fase ini dikoordinasikan oleh sebuah Finite State Machine (FSM). Diagram blok arsitektur sistem dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram blok arsitektur sistem

Seluruh modul sistem berkomunikasi melalui topic dan action ROS 2. Alur komunikasi utama dimulai dari pengguna yang memilih target melalui GUI, yang kemudian mempublikasikan nama kelas objek ke topic `/selected_object_class`. FSM menerima informasi ini dan memulai proses pencarian dengan mengirimkan goal navigasi ke Nav2 melalui action `NavigateToPose`. Secara paralel, node YOLO11n secara terus-menerus memproses citra dari kamera dan mempublikasikan hasil deteksi ke topic

/yolo/detections, yang dibaca oleh FSM untuk memantau keberadaan objek target. Ketika target terdeteksi, FSM beralih ke mode visual servoing dan mengirimkan perintah kecepatan langsung ke topic /cmd_vel. Data LiDAR dari topic /scan digunakan secara paralel oleh mekanisme safety stop untuk mencegah tabrakan selama fase pendekatan. Tabel 3.3 menyajikan node utama yang digunakan beserta fungsinya.

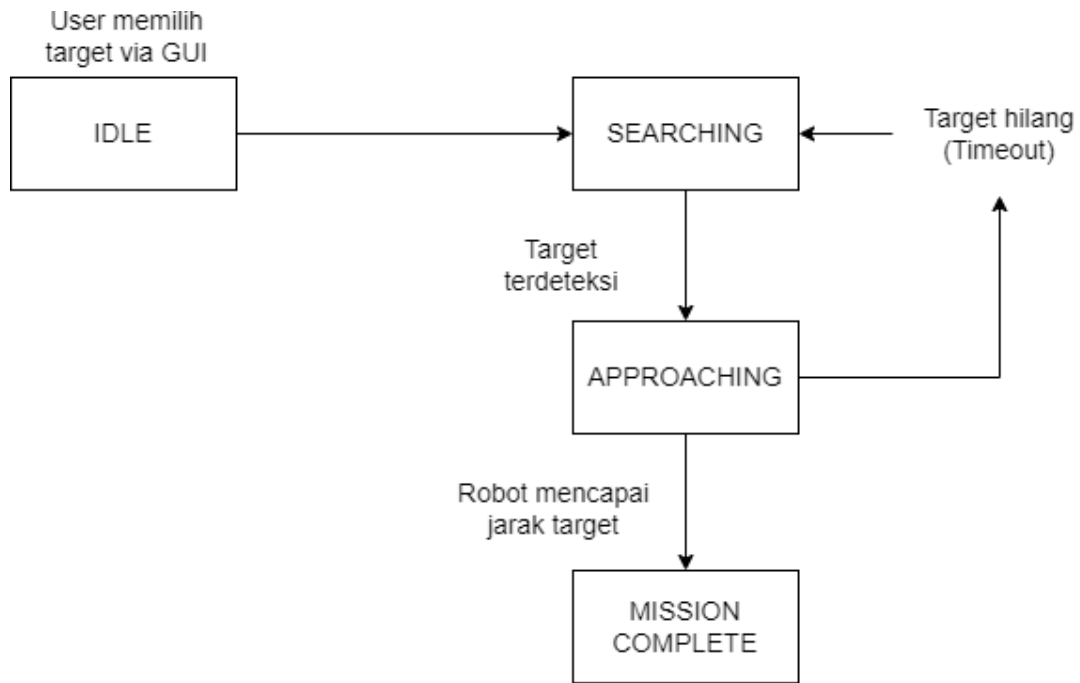
Tabel 3.3 Tabel Daftar ROS 2 Node

Node	Fungsi	Topic Publish	Topic Subscribe
object_follower	FSM utama, koordinasi modul	/cmd_vel	/yolo/detections, /scan, /selected_object_class
yolo_node	Inferensi YOLO11n	/yolo/detections	/camera/image_raw
object_selector_gui	GUI pemilihan target	/selected_object_class	-
Nav2 nodes	Navigasi otonom	/cmd_vel	/map, /scan, /tf
AMCL	Lokalisasi	/tf (map -> odom)	/scan, /map

3.2.2 Perancangan Finite State Machine (FSM)

FSM yang dirancang memiliki empat state utama sebagaimana diilustrasikan pada Gambar 3.3, yaitu IDLE, SEARCHING, APPROACHING, dan MISSION COMPLETE. Setiap state memiliki fungsi dan komponen aktif yang berbeda, serta kondisi transisi yang spesifik. Sistem dimulai dari state IDLE, di mana robot menunggu pengguna memilih kelas objek target melalui GUI. Setelah pemilihan dilakukan, sistem bertransisi ke state SEARCHING, di mana robot bernavigasi secara otonom menggunakan Nav2 sambil menjalankan deteksi YOLO11n secara terus-menerus. Apabila YOLO11n mendeteksi objek target di frame kamera, sistem bertransisi ke state APPROACHING. Pada state ini, Nav2 dinonaktifkan dan kendali pergerakan robot diambil alih oleh visual servoing, dengan bantuan centroid tracker untuk menjaga stabilitas identitas objek. Jika objek hilang dari kamera selama 8 detik, sistem kembali ke state SEARCHING. Ketika ukuran bounding box objek pada citra melebihi ambang batas yang ditentukan (yaitu ketika rasio tinggi bounding box terhadap tinggi citra melebihi 0,55), hal ini menandakan bahwa robot sudah berada pada jarak yang cukup

dekat. Selanjutnya, sistem bertransisi ke state MISSION COMPLETE, robot berhenti, dan metrik dicatat secara otomatis. Tabel 3.4 menjelaskan fungsi masing-masing state, dan Tabel 3.5 menjelaskan transisi antarstate beserta kondisi pemicunya.



Gambar 3.3 Diagram state FSM

Tabel 3.4 Tabel Deskripsi State FSM

State	Fungsi	Komponen Aktif
IDLE	Menunggu pengguna memilih kelas objek target melalui GUI	GUI
SEARCHING	Robot bernavigasi ke titik-titik tujuan menggunakan Nav2 sambil menjalankan deteksi YOLO11n untuk mencari objek target. Jika navigasi gagal atau waypoint tercapai tanpa deteksi, robot melakukan rotasi 360° untuk memindai sekitar	Nav2, YOLO11n
APPROACHING	Robot mendeteksi objek target dan mendekatinya secara presisi menggunakan visual servoing. Posisi objek di citra kamera dilacak menggunakan centroid tracker	YOLO11n, Centroid Tracker, Visual Servo, Safety Stop (LiDAR)
MISSION COMPLETE	Robot berhasil mencapai jarak yang cukup dekat dengan objek target. Pergerakan dihentikan dan metrik dicatat secara otomatis	Pencatatan metrik

Tabel 3.5 Tabel Transisi State FSM

State Awal	State Tujuan	Kondisi Transisi
IDLE	SEARCHING	Pengguna memilih target dari GUI
SEARCHING	APPROACHING	YOLO11n mendeteksi objek target di frame kamera
APPROACHING	MISSION COMPLETE	Robot mencapai jarak yang cukup dekat dengan objek target (ukuran objek pada citra melebihi ambang batas)
APPROACHING	SEARCHING	Target hilang dari kamera selama 8 detik

3.2.3 Perancangan Strategi Pencarian

Pada state SEARCHING, robot memerlukan strategi untuk menentukan arah dan tujuan navigasi. Penelitian ini mengimplementasikan dua strategi pencarian berdasarkan konsep random goal generation dan predefined waypoint navigation. Pemilihan random exploration dan waypoint-based search sebagai dua strategi yang dibandingkan didasarkan pada cara pendekatan yang berbeda dalam navigasi pencarian objek. Pada random exploration, robot menentukan tujuan navigasi secara acak sehingga robot berpotensi menjelajahi seluruh area ruangan tanpa urutan yang tetap. Pendekatan ini tidak memerlukan pengetahuan awal tentang posisi target dan cocok digunakan ketika lokasi target tidak dapat diperkirakan sebelumnya. Sebaliknya, pada waypoint-based search, robot mengunjungi sejumlah titik yang telah ditentukan secara berurutan, sehingga pola pencariannya lebih terstruktur dan konsisten. Perbedaan mendasar ini menjadikan keduanya layak dibandingkan untuk mengetahui performa pencarian objek dalam lingkungan simulasi.

a. Algoritma Random Exploration

Pada strategi ini, robot menentukan tujuan navigasi secara acak dalam area peta yang tersedia. Gambar 3.4 menyajikan pseudocode algoritma ini.

Algoritma 1: Random Exploration

```

procedure RANDOMSEARCH
    Muat occupancy grid map dari Map Server
    Tentukan batas area navigasi dari informasi peta
  
```

```

while target not found do
    Generate posisi acak (x, y) dalam batas area
    Validasi posisi: pastikan berada pada free cell

    if posisi tidak valid then
        Ulangi generate (maks. 40 percobaan)
    end if

    Kirim posisi sebagai goal ke Nav2

    while robot bernavigasi ke goal do
        Proses deteksi YOLO11n pada setiap frame

        if target terdeteksi then
            Transisi ke APPROACHING
            return
        end if
    end while

    if goal tercapai tanpa deteksi then
        Generate posisi acak baru (kembali ke langkah 5)
    end if

    if navigasi gagal then
        Generate posisi acak baru (kembali ke langkah 5)
    end if
end while
end prosedur

```

Gambar 3.4 Pseudocode Algoritma Random Exploration

b. Algoritma Waypoint-Based Search

Pada strategi ini, robot mengunjungi titik-titik strategis yang telah ditentukan secara otomatis berdasarkan peta. Gambar 3.5 menyajikan pseudocode algoritma ini.

Algoritma 2: Waypoint-Based Search

```

procedure WAYPOINTSEARCH
    Muat occupancy grid map dari Map Server
    Hitung 5 waypoint: 1 titik pusat + 4 titik sudut peta
    Validasi setiap waypoint: snap to free cell
    Simpan daftar waypoint yang valid
    index ← 0

```

```

while target not found do
    Pilih waypoint[index] dari daftar
    Kirim waypoint sebagai goal ke Nav2

    while robot bernavigasi ke waypoint do
        Proses deteksi YOLO11n pada setiap frame

        if target terdeteksi then
            Transisi ke APPROACHING
            return
        end if
    end while

    if waypoint tercapai tanpa deteksi then
        Lakukan rotasi 360° selama 6 detik

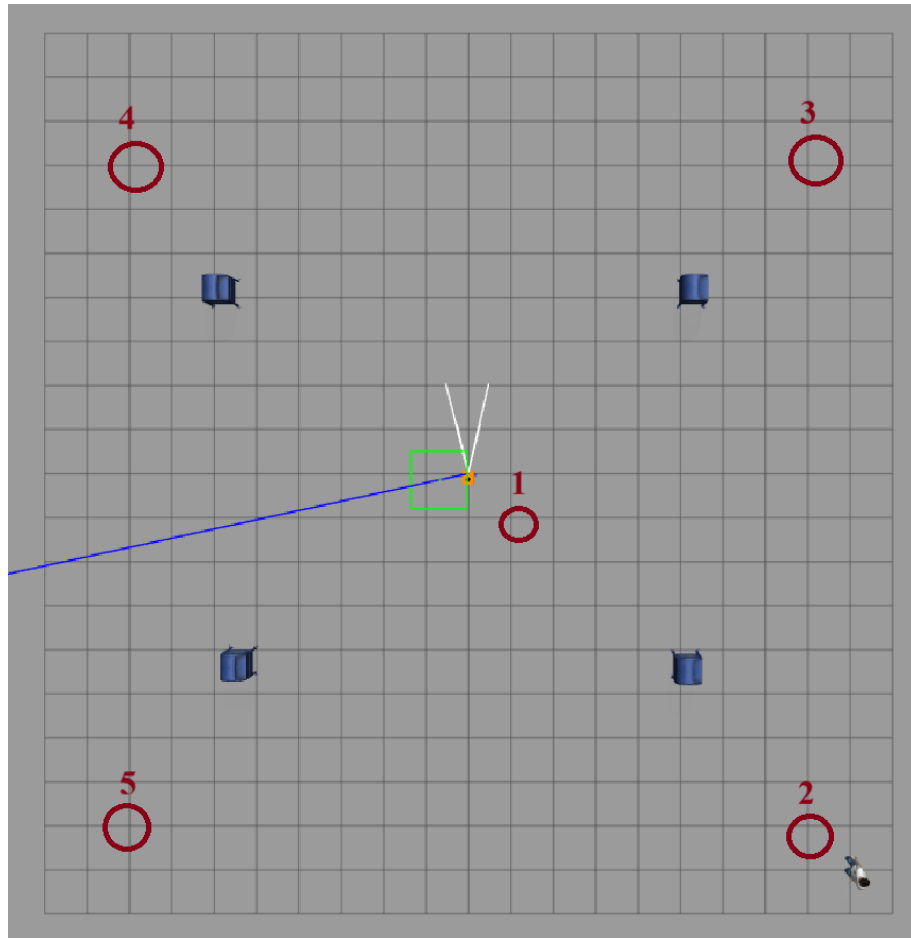
        if target terdeteksi selama rotasi then
            Transisi ke APPROACHING
            return
        end if
    end if

    if navigasi gagal then
        Tandai waypoint sebagai gagal
        Ulangi hingga 3 kali
    end if

    index ← (index + 1) mod jumlah_waypoint
end while
end procedure

```

Gambar 3.5 Pseudocode Algoritma Waypoint-Based Search



Gambar 3.6 Ilustrasi 5 posisi waypoint pada peta (center + 4 sudut)

Berikut beberapa penjelasan dari beberapa istilah teknis yang digunakan dalam kedua algoritma di atas.

1. Occupancy Grid Map

Representasi peta dalam bentuk grid dua dimensi, di mana setiap sel memiliki nilai yang menunjukkan apakah area tersebut kosong (free), terisi halangan (occupied), atau belum diketahui (unknown). Peta ini dihasilkan oleh SLAM Toolbox dan digunakan oleh Nav2 untuk perencanaan jalur.

2. Free Cell

Sel pada occupancy grid map yang bernilai 0 (kosong), menandakan bahwa area tersebut tidak terdapat halangan dan aman untuk dilalui robot.

3. Snap to Free Cell

Proses validasi waypoint yang memastikan posisi tujuan berada pada free cell. Jika posisi yang di-generate berada pada sel yang terisi halangan, sistem mencari free cell terdekat dalam radius 0,8 meter dari posisi tersebut.

4. NavigateToPose

Action standar pada Nav2 yang digunakan untuk mengirimkan tujuan navigasi (goal). Nav2 secara otomatis menghitung jalur dan mengendalikan robot untuk mencapai posisi tersebut sambil menghindari halangan.

3.3 Menyiapkan Lingkungan Simulasi

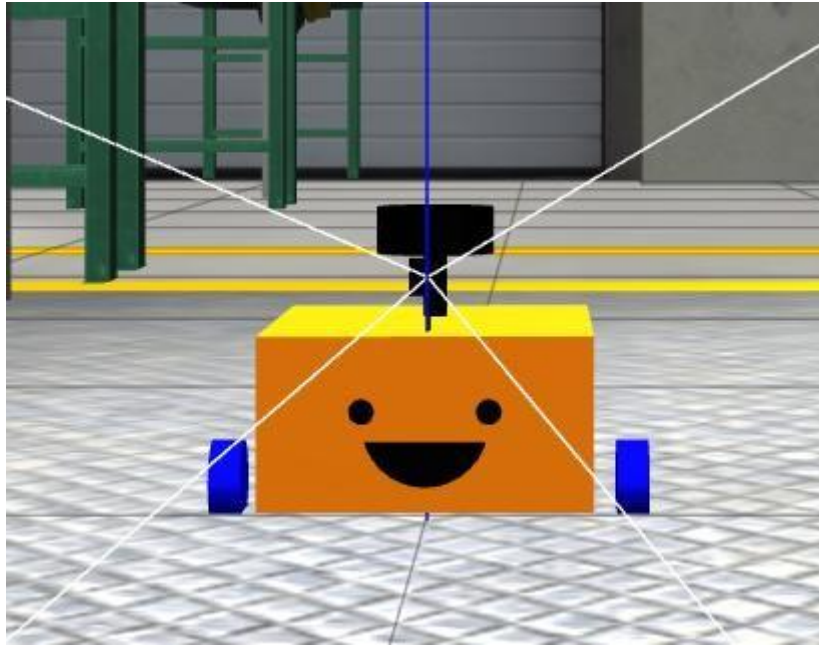
Pada sub-bab ini dijelaskan langkah-langkah dalam menyiapkan lingkungan simulasi yang digunakan untuk pengujian sistem. Langkah-langkah tersebut mencakup konfigurasi robot, pembuatan lingkungan di Gazebo, proses pemetaan menggunakan SLAM Toolbox, serta konfigurasi navigasi dan lokalisasi menggunakan Nav2 dan AMCL.

3.3.1 Konfigurasi Robot (URDF)

Robot yang digunakan dalam simulasi bernama Articubot, yaitu robot differential drive yang dikembangkan berdasarkan tutorial dari kanal YouTube Articulated Robotics. Tutorial ini dipilih sebagai acuan karena menjelaskan proses pembuatan robot dari awal (scratch), termasuk pembuatan URDF, integrasi sensor, konfigurasi SLAM, dan navigasi menggunakan ROS 2. Robot ini telah memenuhi kebutuhan pengujian dalam penelitian ini karena dilengkapi sensor kamera RGB dan LiDAR yang diperlukan untuk deteksi objek dan navigasi. Robot didefinisikan menggunakan format URDF (Unified Robot Description Format) dengan file .xacro dan terdiri dari komponen-komponen berikut.

Tabel 3.6 Tabel Komponen Robot

Komponen	Deskripsi
Chassis	Badan utama robot dengan penggerak differential drive
Roda	Dua roda penggerak (kiri dan kanan) + satu caster wheel
Kamera RGB	Sensor visual untuk menangkap citra (resolusi 640×480 piksel)
LiDAR	Sensor jarak laser untuk navigasi dan safety stop



Gambar 3.7 Model robot di Gazebo/RViz

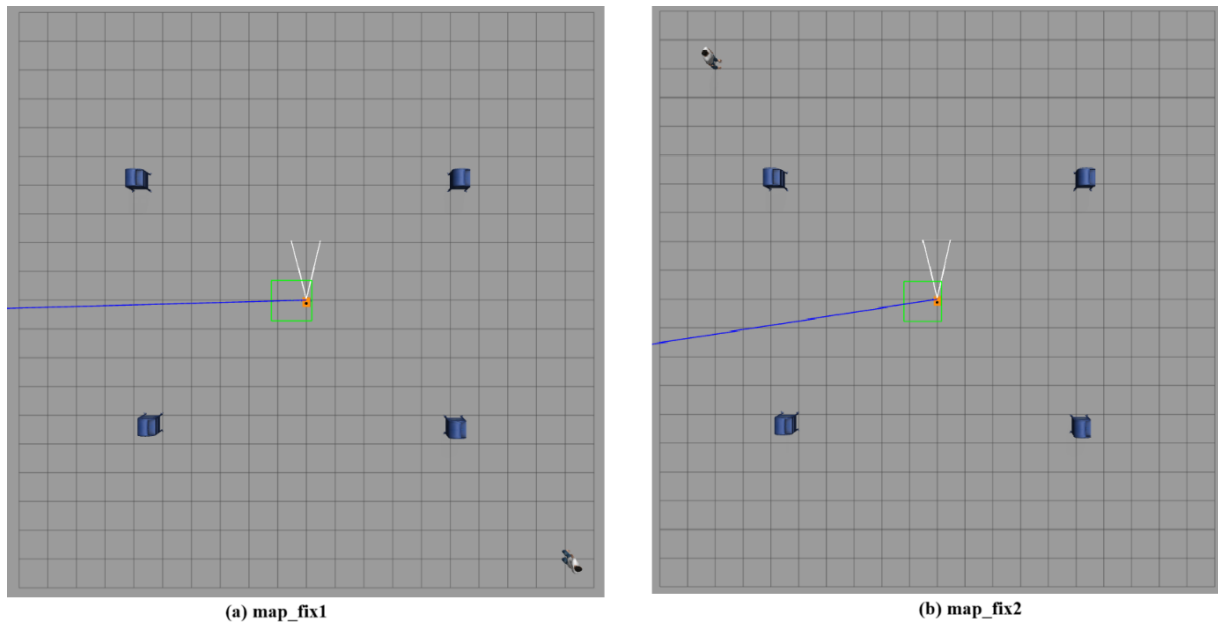
3.3.2 Lingkungan Simulasi Gazebo

Lingkungan simulasi dibuat menggunakan Gazebo 11 yang merepresentasikan ruangan indoor dengan halangan statis. Gazebo diinstal sebagai bagian dari paket ROS 2 Galactic melalui perintah.

```
sudo apt install ros-galactic-gazebo-ros-pkgs
```

Gambar 3.8 Perintah Penginstalan Gazebo

File world yang digunakan merupakan lingkungan yang dibuat secara mandiri (custom) menggunakan Gazebo. Dalam penelitian ini digunakan dua konfigurasi lingkungan simulasi dengan perbedaan pada posisi objek target. Konfigurasi pertama disebut `map_fix1`, di mana objek target ditempatkan di sudut kanan bawah peta. Konfigurasi kedua disebut `map_fix2`, di mana objek target ditempatkan di sudut kiri atas. Tata letak ruangan, posisi robot awal, dan posisi seluruh halangan statis pada kedua konfigurasi adalah identik; Perbedaan hanya terletak pada posisi objek target. Pemisahan menjadi dua konfigurasi dilakukan agar setiap skenario pengujian dapat diulang secara konsisten pada setiap trial.



Gambar 3.9 Screenshot lingkungan Gazebo (a) map_fix1 dan (b) map_fix2

3.3.3 Pemetaan (SLAM Toolbox)

Peta lingkungan simulasi dibuat menggunakan SLAM Toolbox dengan metode online asynchronous. Pemilihan SLAM Toolbox sebagai perangkat pemetaan pada penelitian ini didasarkan pada dua pertimbangan. Pertama, SLAM Toolbox merupakan implementasi SLAM yang secara resmi didukung oleh ekosistem ROS 2 dan terintegrasi langsung dengan Nav2 navigation stack yang digunakan dalam penelitian ini, sehingga tidak memerlukan konfigurasi tambahan untuk menghubungkan keduanya (Macenski dkk., 2023). Kedua, SLAM Toolbox menyediakan mode online asynchronous yang memungkinkan peta dibangun secara bertahap saat robot bergerak, tanpa harus menghentikan pergerakan robot setiap kali data sensor diterima. Cara kerja ini lebih efisien untuk sesi pemetaan manual (teleop) yang digunakan dalam penelitian ini dibandingkan mode synchronous yang memerlukan sinkronisasi ketat antara pergerakan robot dan pembaruan peta. SLAM Toolbox diinstal melalui perintah berikut.

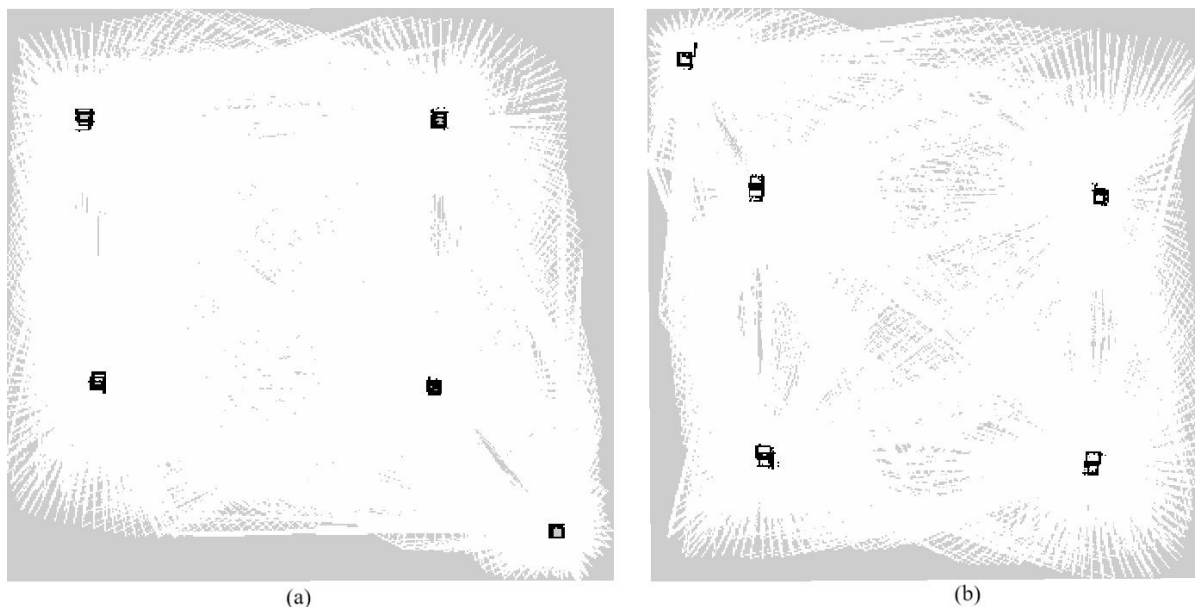
```
sudo apt install ros-galactic-slam-toolbox
```

Gambar 3.10 Perintah Penginstalan SLAM Toolbox

Sesuai dengan dua konfigurasi lingkungan yang telah dijelaskan pada sub-bab 3.3.2, dalam penelitian ini dibuat dua peta yang masing-masing berkaitan dengan file world yang berbeda, yaitu map_fix1 untuk lingkungan posisi target 1 dan map_fix2 untuk lingkungan posisi target 2. Proses pemetaan untuk setiap peta dilakukan dengan langkah-langkah berikut.

- a. Menjalankan simulasi Gazebo dengan robot di dalamnya.
- b. Menjalankan node SLAM Toolbox untuk memulai proses pemetaan.
- c. Mengendalikan robot secara manual menggunakan teleop keyboard untuk menjelajahi seluruh area.
- d. Menyimpan peta yang telah selesai menggunakan map_saver ke dalam format .pgm (citra) dan .yaml (metadata).

Hasil pemetaan berupa occupancy grid map yang digunakan sebagai acuan untuk navigasi dan lokalisasi pada masing-masing skenario pengujian. Gambar 3.11 menampilkan hasil peta dari kedua lingkungan simulasi tersebut.



Gambar 3.11 Hasil peta OccupancyGrid dari (a) map_fix1 dan (b) map_fix2

3.3.4 Navigasi dan Lokalisasi (Nav2 dan AMCL)

Navigasi robot menggunakan Nav2 stack yang menyediakan perencanaan jalur global, kontrol lokal, dan penghindaran halangan. Lokalisasi robot pada peta menggunakan AMCL (Adaptive Monte Carlo Localization). Nav2 diinstal melalui dua paket terpisah: `ros-galactic-navigation2` yang menyediakan algoritma inti Nav2 (perencana jalur dan pengontrol), serta `ros-galactic-nav2-bringup` yang menyediakan launch file dan konfigurasi default untuk menjalankan Nav2 secara terintegrasi. Berikut perintah untuk menginstall dua paket tersebut.

```
sudo apt install ros-galactic-navigation2
sudo apt install ros-galactic-nav2-bringup
```

Gambar 3.12 Perintah Penginstalan ros-galactic-navigation2 dan ros-galactic-nav2-bringup

Konfigurasi parameter Nav2 dan AMCL disimpan dalam file `nav2_params.yaml`. Komponen utama yang digunakan dapat dilihat pada Tabel 3.7.

Tabel 3.7 Tabel Komponen Nav2 yang Digunakan

Komponen	Fungsi
NavfnPlanner	Global path planner yang menghitung jalur dari posisi robot ke tujuan menggunakan algoritma A*
DWB Local Planner	Local controller yang mengacu global path sambil menghindari halangan
Costmap	Representasi peta dalam bentuk grid biaya untuk navigasi
AMCL	Lokalisasi robot pada peta menggunakan particle filter
Recovery Behaviors	Perilaku pemulihan otomatis (spin, backup, wait) saat robot terhalang

3.4 Implementasi Sistem

Pada sub-bab ini dijelaskan implementasi masing-masing komponen perangkat lunak yang digunakan dalam sistem. Seluruh komponen diimplementasikan dalam dua package ROS 2 yang dikembangkan secara mandiri dalam penelitian ini. Package `object_follower` memuat FSM utama beserta seluruh modul pendukung (deteksi, tracking, visual servoing, navigasi, GUI, dan pencatatan metrik), sedangkan package `yolobot_recognition` memuat node inferensi YOLO11n. Setiap modul diimplementasikan sebagai file Python terpisah untuk menjaga modularitas dan kemudahan pemeliharaan.

3.4.1 Deteksi Objek (YOLO11n Pretrained)

Deteksi objek menggunakan model YOLO11n dengan bobot pretrained dari dataset COCO yang mampu mengenali 80 kelas objek standar. Model pretrained digunakan tanpa proses fine-tuning karena objek-objek target yang digunakan dalam pengujian (seperti person) merupakan kelas standar yang sudah termasuk dalam dataset COCO. Pemilihan varian nano (n) didasarkan pada kebutuhan inferensi *real-time* dengan sumber daya komputasi yang terbatas. Pustaka Ultralytics diinstal melalui perintah.

```
pip install ultralytics==8.4.39
```

Gambar 3.13 Perintah Penginstalan Pustaka Ultralytics

Inferensi YOLO11n dijalankan oleh package `yolobot_recognition`. Node deteksi (`yolo_node`) menerima citra dari topic `/camera/image_raw`, menjalankan inferensi YOLO11n, lalu mempublikasikan hasil deteksi berupa bounding box dan kelas objek ke topic `/yolo/detections` menggunakan custom message type `Yolov8Inference`.

3.4.2 Object Tracking (Centroid Tracker)

Object tracking diimplementasikan menggunakan centroid tracker yang berfungsi untuk mempertahankan identitas objek antara frame deteksi. Centroid tracker bekerja dengan menghitung titik pusat (centroid) dari setiap bounding box hasil deteksi YOLO11n, kemudian mengasosiasikan deteksi antarframe berdasarkan jarak Euclidean minimum antara centroid pada frame sebelumnya dan frame saat ini.

3.4.3 Visual Servoing (IBVS)

Pendekatan objek pada state APPROACHING menggunakan metode Image-Based Visual Servoing (IBVS). Pada metode ini, robot mengoreksi arah dan kecepatannya secara langsung berdasarkan posisi objek pada citra kamera, tanpa memerlukan estimasi jarak 3D. Sistem menggunakan dua umpan balik dari citra, di antaranya.

a. Posisi horizontal objek

Digunakan untuk mengoreksi arah hadap robot (belok kiri/kanan) agar objek tetap berada di tengah frame.

b. Ukuran relatif objek

rasio tinggi bounding box terhadap tinggi citra digunakan sebagai indikator jarak. Semakin besar objek di citra, semakin dekat robot dengan target. Ketika rasio ini melebihi threshold 0,55, robot dianggap telah cukup dekat dengan target.

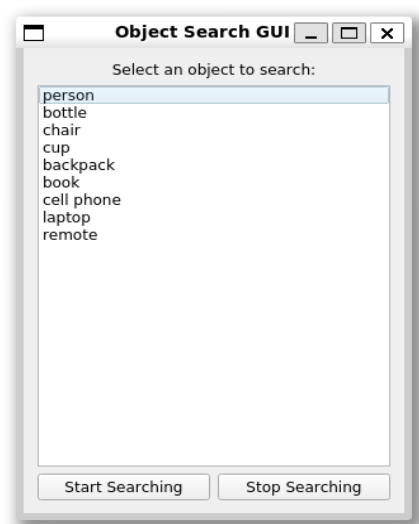
Robot berhenti bergerak (state MISSION COMPLETE) ketika ukuran objek pada citra sudah melebihi ambang batas tertentu, yang menunjukkan bahwa robot telah cukup dekat dengan target.

3.4.4 Mekanisme Safety Stop

Pada state APPROACHING, Nav2 tidak aktif karena robot dikendalikan sepenuhnya oleh Visual Servoing. Untuk mengisi kekosongan fungsi penghindaran halangan, sistem menggunakan mekanisme safety stop berbasis LiDAR yang membaca topic /scan setiap iterasi loop. Jika LiDAR mendeteksi halangan dalam jarak kurang dari 0,8 meter di sektor depan robot, sistem menghentikan gerakan maju dan menjalankan manuver pemulihan: robot mundur sambil memutar badan ke sisi yang lebih lapang berdasarkan perbandingan jarak LiDAR kiri dan kanan. Apabila halangan masih menghalangi jalur setelah 3 detik, sistem mengaktifkan bypass Nav2 dengan mengirimkan tujuan sementara sejauh 1,5 meter secara tegak lurus dari posisi robot, kemudian kembali ke mode pendekatan target setelah titik bypass dicapai. Jumlah iterasi di mana safety stop aktif dicatat sebagai metrik `safety_stop_count` dalam berkas CSV.

3.4.5 GUI Pemilihan Target

GUI pemilihan target diimplementasikan menggunakan pustaka tkinter pada Python. GUI ini menampilkan daftar kelas objek yang dapat dipilih oleh pengguna sebagai target pencarian. Ketika pengguna memilih suatu kelas objek, GUI mempublikasikan nama kelas tersebut melalui topic ROS 2 `/selected_object_class`, yang kemudian diterima oleh FSM untuk memulai proses pencarian. Tampilan GUI pemilihan target dapat dilihat pada Gambar 3.14.



Gambar 3.14 *Screenshot* GUI pemilihan target

3.4.6 Pencatatan Metrik Otomatis

Sistem dilengkapi mekanisme pencatatan metrik secara otomatis ke dalam file CSV. Pencatatan metrik dimulai saat pengguna memilih target melalui GUI dan berakhir saat sistem mencapai salah satu kondisi terminal, yaitu misi berhasil dengan nilai SUCCESS (robot berhasil mendekati target) atau misi gagal dengan nilai FAILURE (robot tidak dapat menyelesaikan misi). Data kemudian disimpan secara otomatis ke file metrics_results.csv.

Tabel 3.8 Tabel Kolom Metrik pada File CSV

Kolom	Tipe	Jenis	Keterangan
trial_id	Integer	Identitas	Nomor urut percobaan
map_name	String	Identitas	Nama peta yang digunakan (map_fix1 atau map_fix2)
target_class	String	Identitas	Kelas objek target
search_mode	String	Identitas	Strategi pencarian (RANDOM/ SYSTEMATIC)
time_to_find_s	Float	Utama	Waktu pencarian (detik)
time_to_reach_s	Float	Utama	Waktu pendekatan (detik)
total_time_s	Float	Utama	Total waktu misi (detik)
path_length_m	Float	Utama	Panjang lintasan robot (meter)
safety_stops	Integer	Utama	Jumlah safety stop selama misi
result	String	Utama	Hasil misi (SUCCESS/FAIL)
nav_goal_sent	Integer	Diagnostik	Jumlah goal Nav2 yang dikirim
nav_goals_canceled	Integer	Diagnostik	Jumlah goal Nav2 yang dibatalkan

3.5 Skenario Pengujian

Pada sub-bab ini dijelaskan skenario pengujian yang digunakan untuk mengevaluasi dan membandingkan performa kedua strategi pencarian. Skenario pengujian mencakup desain eksperimen yang menentukan variabel dan jumlah percobaan, serta metrik evaluasi yang digunakan untuk mengukur performa sistem.

3.5.1 Desain Eksperimen

Pengujian dilakukan untuk membandingkan performa kedua strategi pencarian (Algoritma 1: Random Exploration dan Algoritma 2: Waypoint-Based Search) dengan desain eksperimen sebagaimana disajikan pada Tabel 3.9 dan Tabel 3.10.

Tabel 3.9 Tabel Variabel Penelitian

Variabel	Keterangan
Variabel independen	Strategi pencarian (random exploration dan waypoint-based search)
Variabel dependen	Metrik performa (Time to Find, Time to Reach, Total Time, Path Length, Safety Stops, Success Rate)
Variabel kontrol	Posisi target, Posisi awal robot, peta lingkungan, parameter robot (kecepatan, ambang batas safety stop), model dan parameter deteksi objek

Penelitian ini dirancang sebagai eksperimen komparatif, di mana hanya strategi pencarian yang dibedakan antara satu kelompok pengujian dengan kelompok lainnya, sementara seluruh kondisi lainnya dijaga tetap sama. Tujuan dari pengaturan ini adalah agar perbedaan hasil yang teramati pada metrik kinerja dapat dikaitkan dengan perbedaan strategi yang digunakan, bukan disebabkan oleh faktor lain.

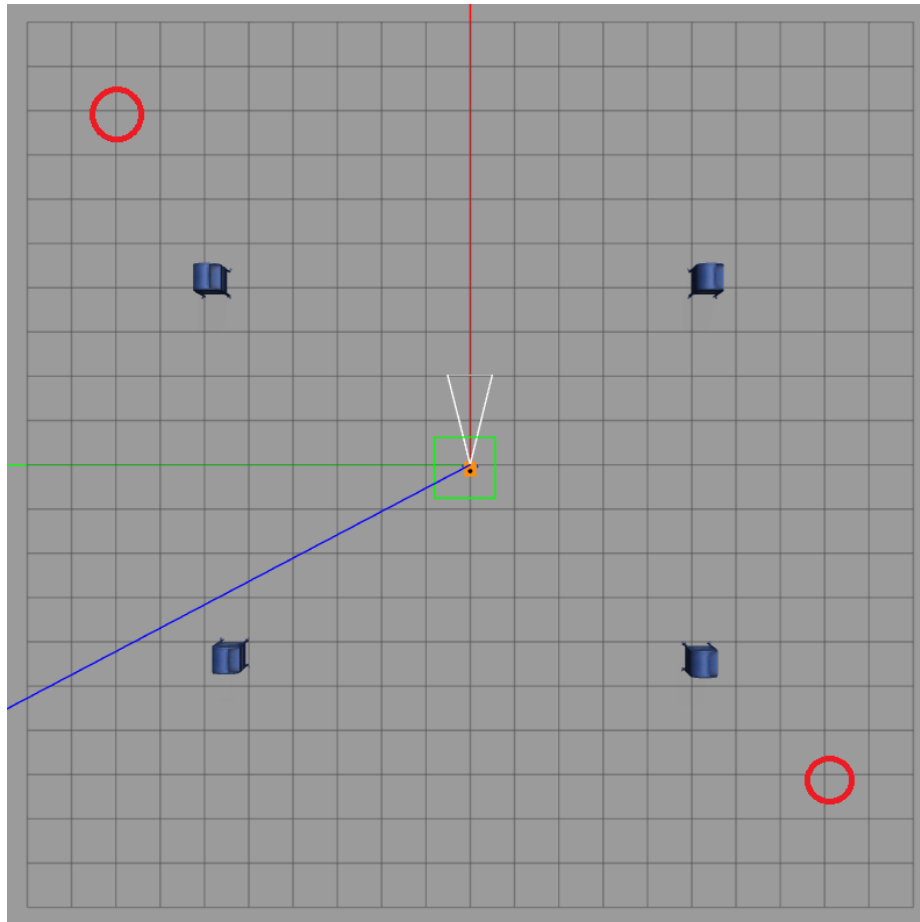
Tabel 3.10 Tabel Parameter Eksperimen

Variabel	Keterangan
Jumlah posisi target	2 posisi berbeda di dalam peta

Variabel	Keterangan
Jumlah pengulangan	10 kali per kombinasi
Total trial	$2 \times 2 \times 10 = 40$ trial

Penentuan jumlah pengujian sebanyak 10 kali untuk setiap skenario ini merujuk pada metodologi dari Ginanjar (2025). Penelitian tersebut menguji kinerja algoritma navigasi robot menggunakan simulasi ROS dan Gazebo. Pada pengujian berbasis simulasi, 10 kali pengulangan sudah cukup untuk mendapatkan data yang valid. Jumlah ini terbukti mampu menghasilkan nilai rata-rata dan standar deviasi yang stabil. Jumlah pengujian tersebut juga dinilai cukup untuk melihat konsistensi gerakan robot dan mendeteksi masalah seperti kebuntuan arah (deadlock). Oleh karena itu, total 40 kali pengujian dalam penelitian ini sudah memenuhi standar evaluasi yang baik tanpa membebani komputer secara berlebihan.

Pada setiap trial, robot memulai dari posisi awal yang sama. Pengguna memilih kelas objek target melalui GUI, lalu robot menjalankan pencarian menggunakan salah satu strategi hingga objek berhasil ditemukan dan didekati. Dua posisi target dipilih untuk menguji apakah performa masing-masing strategi bersifat konsisten pada kondisi penempatan target yang berbeda, atau menunjukkan perbedaan yang bergantung pada lokasi target di dalam ruangan. Posisi masing-masing target dapat dilihat pada Gambar 3.15.



Gambar 3.15 Denah peta dengan 2 posisi target yang ditandai

3.5.2 Metrik Evaluasi

Metrik yang digunakan dipilih berdasarkan relevansinya terhadap tujuan penelitian. Mengingat inti penelitian adalah membandingkan efisiensi dua strategi pencarian, maka metrik utama yang dipilih adalah yang secara langsung mencerminkan seberapa cepat dan seberapa efisien robot menemukan dan mendekati target. Selain itu, metrik keandalan (success rate) dan metrik keamanan (safety stops) ditambahkan untuk memberikan gambaran yang lebih lengkap tentang performa sistem secara menyeluruh. Pendekatan pemilihan metrik ini selaras dengan metrik yang digunakan pada penelitian serupa, khususnya Ma dkk. (2020) yang mengukur waktu dan keberhasilan misi, serta Gargouri dkk. (2025) yang menggunakan success rate navigasi sebagai ukuran reliabilitas sistem. Metrik evaluasi yang digunakan untuk membandingkan kedua strategi pencarian disajikan pada Tabel 3.11.

Tabel 3.11 Tabel Metrik Evaluasi

Metrik	Definisi	Satuan	Alasan Pemilihan
Time to Find	Waktu dari mulai pencarian hingga target pertama kali terdeteksi oleh YOLO11n	detik (s)	Mengukur efisiensi fase SEARCHING
Time to Reach	Waktu dari target pertama kali terdeteksi hingga robot mencapai target	detik (s)	Mengukur efisiensi fase APPROACHING
Total Time	Time to Find + Time to Reach	detik (s)	Mengukur durasi misi secara keseluruhan dari awal hingga selesai
Path Length	Total jarak kumulatif yang ditempuh robot selama misi	meter (m)	Mengukur efisiensi pergerakan robot
Safety Stops	Jumlah kali mekanisme safety stop aktif selama misi	kali	Mengukur seberapa sering robot menghadapi situasi berisiko (terhalang obstacle)
Success Rate	Persentase trial dengan hasil SUCCESS dari total trial per kombinasi strategi dan posisi target	%	Mengukur tingkat keberhasilan dasar sistem

3.5.3 Prosedur Pelaksanaan Pengujian

Pengambilan data eksperimen memerlukan beberapa komponen perangkat lunak yang aktif secara bersamaan, di antaranya simulator Gazebo, sistem lokalisasi dan navigasi, serta node pencarian objek. Karena tiap komponen bergantung pada output dari komponen sebelumnya, pengaktifan dilakukan secara bertahap mengikuti urutan yang telah ditentukan. Selain itu, seluruh komponen dikonfigurasi dengan parameter `use_sim_time: true` agar semua node menggunakan jam internal simulasi Gazebo, bukan jam sistem. Sinkronisasi ini penting mengingat kecepatan waktu simulasi tidak selalu setara dengan waktu nyata. Ketidaksesuaian

jam dapat menyebabkan data sensor yang diterima node menjadi tidak sinkron dan berujung pada kegagalan komunikasi antar-node. Berikut prosedur pelaksanaan pengujian.

A. Menjalankan lingkungan simulasi gazebo

Lingkungan simulasi diaktifkan melalui perintah.

```
ros2 launch articubot_TA launch_sim.launch.py
world:=$(pwd)/src/articubot_TA/worlds/map_fix1.world use_sim_time:=true
```

Gambar 3.16 Perintah menjalankan lingkungan simulasi

Argumen `world:` menentukan file world Gazebo yang akan dimuat, sekaligus menentukan tata letak ruangan, posisi halangan, dan lokasi objek target sesuai konfigurasi yang sedang diuji. Jika berhasil, jendela Gazebo akan terbuka dan menampilkan robot di tengah lingkungan virtual.

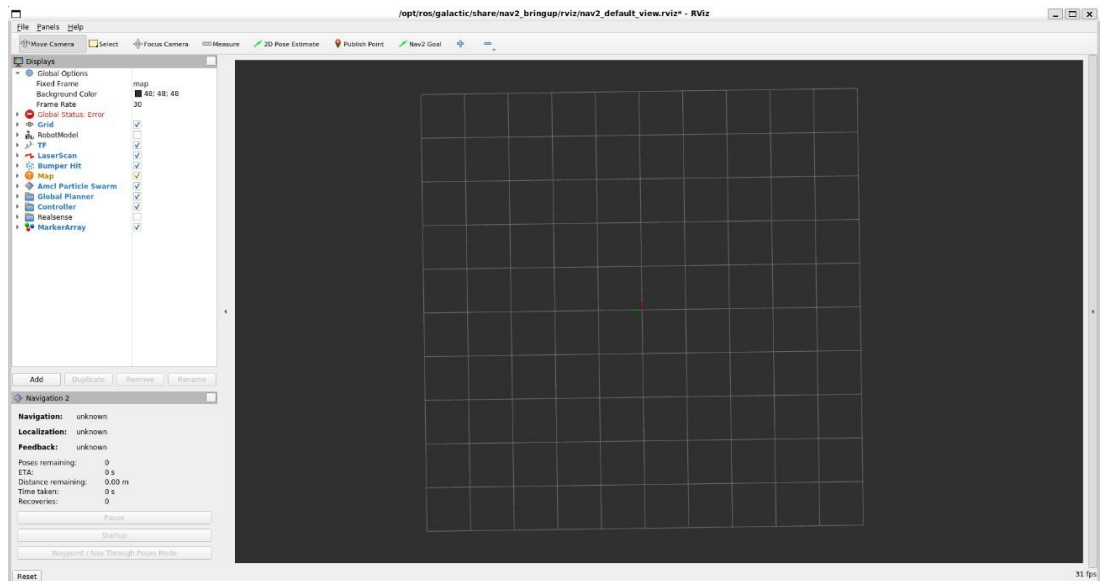
B. Membuka Antarmuka Visualisasi RViz

RViz dibuka menggunakan konfigurasi tampilan bawaan Nav2.

```
ros2 run rviz2 rviz2 -d $(ros2 pkg prefix
nav2_bringup)/share/nav2_bringup/rviz/nav2_default_view.rviz
```

Gambar 3.17 Perintah menjalankan RViz

RViz berfungsi sebagai jendela pemantauan kondisi robot secara *real-time*, mencakup posisi robot pada peta, costmap yang digunakan Nav2 dalam membuat keputusan navigasi, dan konfirmasi visual lokalisasi. File konfigurasi `nav2_default_view.rviz` sudah menyertakan tampilan yang dibutuhkan sehingga tidak perlu dikonfigurasi ulang.



Gambar 3.18 Tampilan RViz

C. Menjalankan sistem lokalisasi (AMCL)

Sistem lokalisasi dijalankan dengan merujuk pada peta yang telah dibuat di tahap perancangan. Lokalisasi dijalankan melalui perintah.

```
ros2 launch articubot_TA localization_launch.py use_sim_time:=true
slam:=False map:=/home/sancross354/articubot_TA/map_fix1.yaml
```

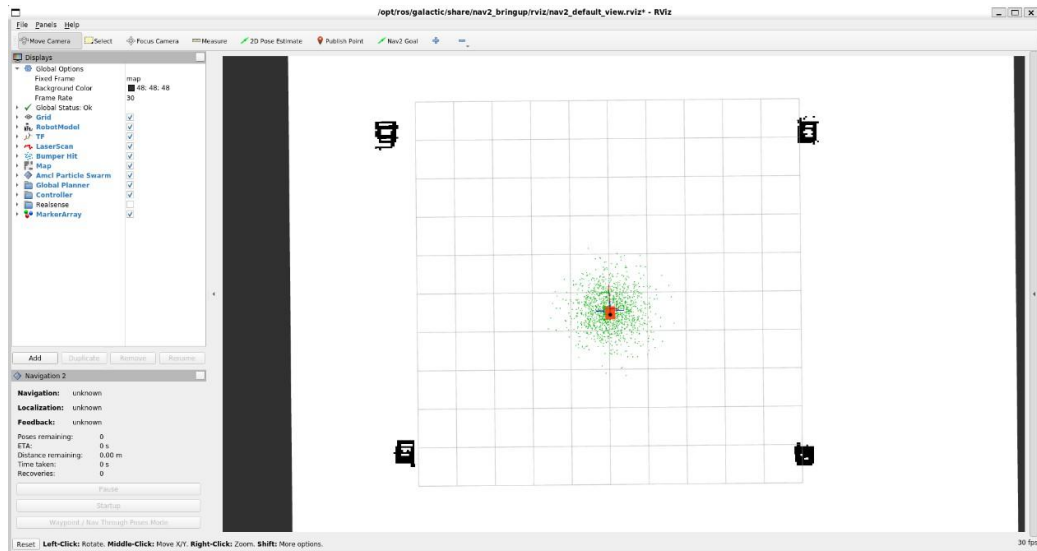
Gambar 3.19 Perintah menjalankan lingkungan simulasi

Nilai `slam:=False` menegaskan bahwa tahap pemetaan sudah selesai; yang berjalan sekarang hanyalah proses lokalisasi, yakni penentuan posisi robot di dalam peta yang sudah ada menggunakan algoritma AMCL (Adaptive Monte Carlo Localization). File `.yaml` yang dirujuk pada argumen `map:=` memuat informasi occupancy grid yang menjadi referensi lokalisasi.

D. Inisialisasi Posisi Awal Robot (2D Pose Estimate)

Setelah AMCL aktif, posisi awal robot perlu ditetapkan secara manual melalui RViz. Caranya adalah dengan mengklik tombol "2D Pose Estimate" pada toolbar RViz, kemudian mengklik dan menyeret cursor pada area peta yang sesuai dengan posisi dan arah hadap robot di simulasi. Alasan langkah ini diperlukan adalah cara kerja AMCL itu sendiri. Algoritma ini menyebarkan sejumlah besar partikel hipotesis secara acak di seluruh peta untuk memperkirakan posisi robot. Apabila tidak ada petunjuk awal, proses konvergensi partikel bisa sangat lambat. Dengan memberikan estimasi posisi

awal yang mendekati kenyataan, partikel langsung terpusat di sekitar lokasi yang tepat dan lokalisasi dapat berlangsung jauh lebih cepat.



Gambar 3.20 Tampilan RViz ketika telah berhasil dilakukan lokalisasi

E. Menjalankan Sistem Navigasi Nav2

Navigation stack Nav2 diaktifkan melalui perintah.

```
ros2 launch articubot_TA navigation_launch.py use_sim_time:=true
map_subscribe_transient_local:=true
```

Gambar 3.21 Perintah menjalankan sistem navigasi Nav2

map_subscribe_transient_local: true dipasang untuk mengatasi perbedaan kebijakan Quality of Service (QoS) antara sistem lokalisasi yang mengirimkan topik peta dan Nav2 yang menerimanya. Tanpa penyesuaian ini, Nav2 berisiko tidak mendapatkan data peta karena perbedaan pengaturan durability pada lapisan komunikasi ROS 2.

F. Menjalankan GUI Pemilihan Target

Antarmuka GUI dijalankan agar pengguna dapat menentukan kelas objek yang akan dicari setiap kali trial dimulai.

```
ros2 run object_follower object_selector_node_pyqt5 --ros-args --param
use_sim_time:=true
```

Gambar 3.22 Perintah menjalankan sistem navigasi Nav2

G. Menjalankan Node Pencarian Objek

Node utama sistem pencarian dijalankan dengan parameter strategi yang disesuaikan dengan sesi pengujian yang sedang berlangsung. Jika ingin melakukan pengujian dengan random exploration dapat melalui perintah.

```
ros2 run object_follower object_selector_node_pyqt5 --ros-args --param
use_sim_time:=true
```

Gambar 3.23 Perintah menjalankan sistem navigasi Nav2 dengan strategi random exploration

Sebaliknya, jika ingin melakukan pengujian dengan waypoint based search maka jalankan perintah.

```
ros2 run object_follower object_selector_node_pyqt5 --ros-args --param
use_sim_time:=true
```

Gambar 3.24 Perintah menjalankan sistem navigasi Nav2 dengan strategi waypoint based exploration

H. Pelaksanaan Trial dan Pencatatan Data

Setelah semua komponen berjalan, pengguna memilih kelas objek target (person) pada GUI dan menekan tombol Start Searching. FSM beralih dari state IDLE ke SEARCHING, dan robot memulai pencarian secara otonom sesuai strategi yang dipilih. Misi dianggap selesai ketika robot berhasil mendekati target dan sistem mencatat status SUCCESS, atau ketika terjadi kegagalan yang tidak dapat dipulihkan dengan status FAIL. Begitu misi selesai, seluruh nilai metrik secara otomatis ditulis ke baris baru dalam file metrics_results.csv. Prosedur pada langkah ini diulang sebanyak 10 kali untuk tiap kombinasi strategi pencarian dan posisi target, menghasilkan total 40 trial. Sebelum setiap trial berikutnya dimulai, simulasi di-reset ke kondisi awal dan Langkah 4 (2D Pose Estimate) diulang kembali guna memastikan lokalisasi robot kembali akurat.

BAB IV

HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil pengujian dan pembahasan dari sistem robot mobile otonom yang telah dikembangkan. Pengujian ini dilakukan dalam dua tahap. Tahap pertama adalah validasi fungsional untuk memastikan setiap komponen sistem bekerja sesuai dengan perancangan. Tahap kedua adalah pengujian eksperimental untuk membandingkan performa dua strategi pencarian objek, yaitu Random Exploration dan Waypoint-Based Search, pada dua posisi target yang berbeda.

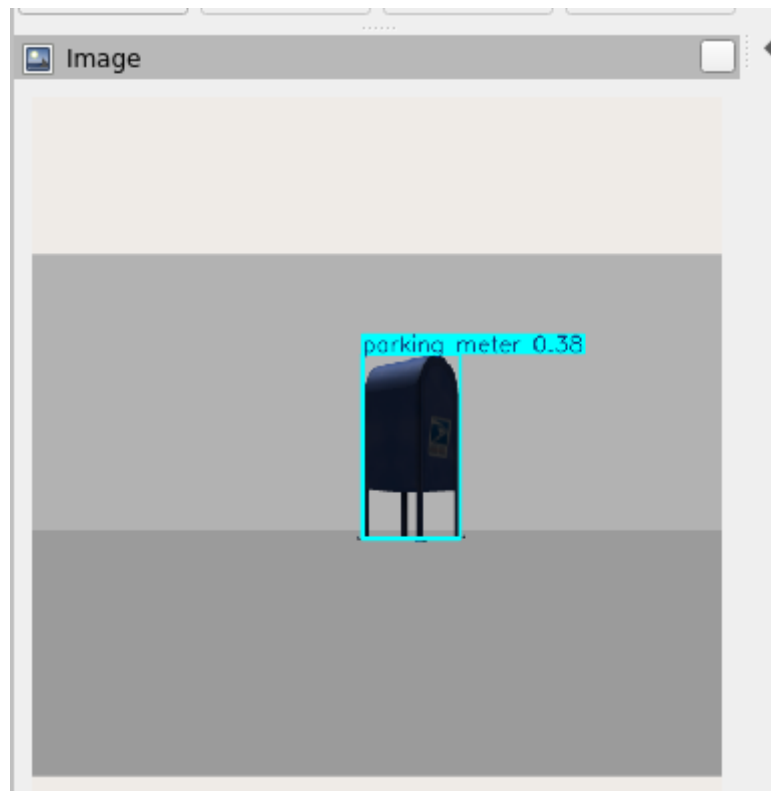
4.1 Implementasi dan Validasi Sistem

Sebelum data eksperimen dapat dianalisis, seluruh komponen inti sistem harus divalidasi terlebih dahulu guna memastikan masing-masing berfungsi sesuai perancangan. Validasi mencakup tiga aspek utama, di antaranya kemampuan deteksi objek oleh YOLO11n, alur transisi state pada FSM, serta keandalan Visual Servoing dan Centroid Tracker pada fase pendekatan target.

4.1.1 Validasi Deteksi Objek (YOLO11n)

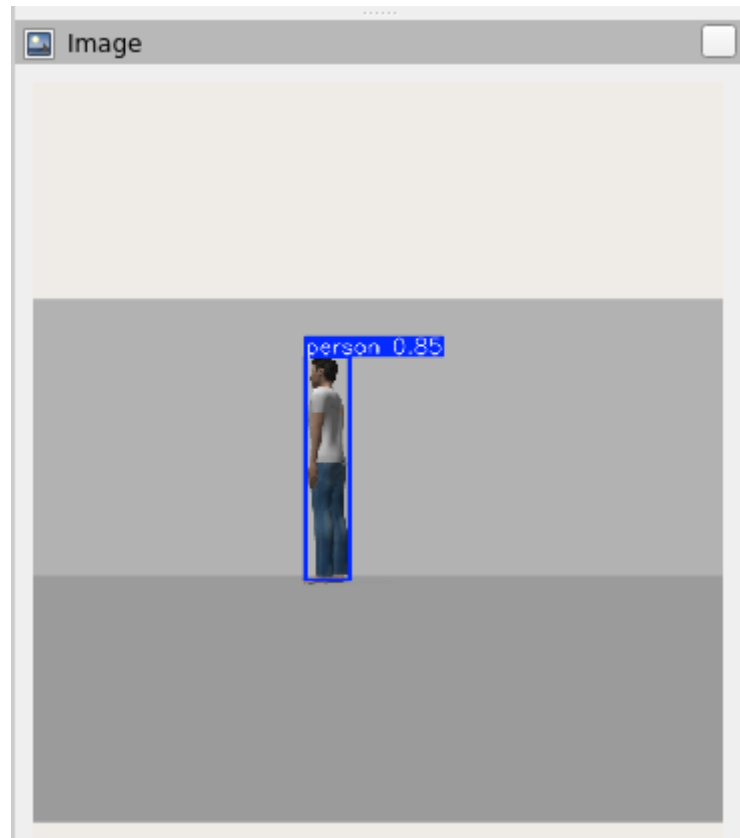
Komponen pertama yang divalidasi adalah modul deteksi objek. Keandalan modul ini merupakan prasyarat utama keberhasilan sistem secara keseluruhan, karena tanpa deteksi yang akurat, robot tidak akan mampu mengidentifikasi target dan seluruh pipeline misi tidak dapat dieksekusi. Model YOLO11n dengan bobot pretrained dari dataset COCO digunakan untuk mendeteksi objek target (kelas person) secara *real-time* dalam lingkungan simulasi Gazebo. Proses inferensi dijalankan oleh node `yolo_node`, yang membaca citra dari topic `/camera/image_raw` dan mempublikasikan hasil deteksi ke topic `/yolo/detections`.

Dalam lingkungan simulasi yang digunakan, terdapat dua jenis objek fisik, yaitu standing person sebagai objek target misi dan kotak halangan (mailbox) sebagai rintangan statis. Pada saat validasi berlangsung, ditemukan bahwa model YOLO11n tidak mampu mengenali kotak halangan tersebut sebagai "mailbox" karena kelas yang bersesuaian tidak tersedia dalam dataset COCO. Model mengenalinya sebagai kelas yang paling mendekati secara visual, misalnya parking meter, dengan nilai confidence score yang rendah. Kondisi ini tidak mengganggu jalannya misi, karena FSM dirancang untuk hanya merespons deteksi pada kelas yang dipilih pengguna melalui GUI (person). Hal ini dapat dilihat pada Gambar 4.1.



Gambar 4.1 YOLO11n mendeteksi kotak halangan sebagai kelas lain disertai confidence score yang rendah

Objek target (person) berhasil terdeteksi secara akurat pada seluruh 40 trial, yang mengonfirmasi bahwa model pretrained COCO memiliki keandalan yang memadai untuk skenario pengujian ini Gambar 4.2 menunjukkan objek target (person) berhasil terdeteksi secara akurat.

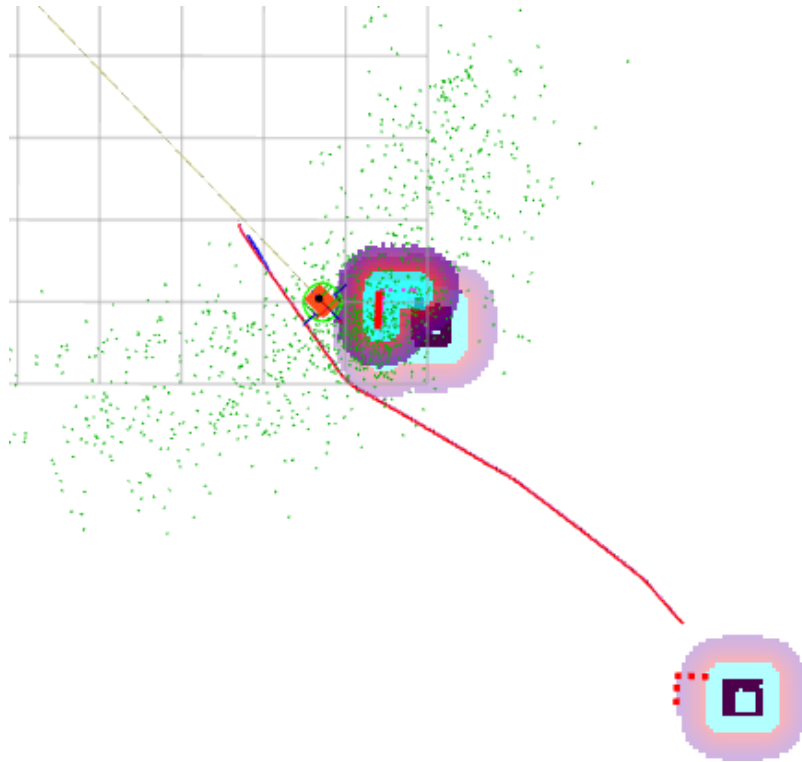


Gambar 4.2 YOLO11n mendeteksi objek target person dengan bounding box dan label kelas yang akurat

4.1.2 Validasi Alur FSM

Setelah objek target terdeteksi, tanggung jawab pengambilan keputusan berpindah ke FSM terkait kapan perintah navigasi global harus dihentikan dan kapan kendali dialihkan ke fase APPROACHING. Subbab ini memvalidasi bahwa proses perpindahan state tersebut terjadi secara benar dan tidak menimbulkan konflik perintah pada sistem penggerak robot. FSM yang dirancang mencakup empat state utama, di antaranya IDLE, SEARCHING, APPROACHING (yang mengeksekusi fase tracking melalui Visual Servoing), dan MISSION COMPLETE. Validasi dilakukan dengan mengamati log terminal selama pengujian berlangsung.

Transisi dari state SEARCHING ke APPROACHING terbukti berlangsung secara mulus. Transisi ini merupakan tahapan yang kritis dalam sistem, karena pada saat YOLO11n mendeteksi target, perintah navigasi global (Nav2) harus segera dibatalkan dan kendali robot beralih ke Visual Servoing.



Gambar 4.3 YOLO11n mendeteksi target sehingga Nav2 dibatalkan dan sistem bertransisi dari fase SEARCHING ke APPROACHING

Kegagalan dalam proses transisi ini berpotensi menimbulkan konflik antara dua sumber perintah kecepatan yang beroperasi secara bersamaan, yang dapat mengakibatkan perilaku robot yang tidak terprediksi. Keberhasilan seluruh 40 trial dalam menyelesaikan transisi state tersebut tanpa tabrakan maupun situasi di mana robot terhenti dan tidak dapat melanjutkan misi (deadlock) membuktikan bahwa logika prioritas perintah dalam FSM telah terintegrasi dengan baik.

```
[INFO] [1777043131.551169499] [object_follower_modular]: Nav2 action server is available
[INFO] [1777043191.572008926] [object_follower_modular]: Metrics recording STARTED for target: person
[INFO] [1777043191.573294834] [object_follower_modular]: Selected object: person
[INFO] [1777043191.574910236] [object_follower_modular]: Selected 'person'. Searching...
[INFO] [1777043191.789479182] [object_follower_modular]: Searching: rotating

[INFO] [1777043250.551027873] [object_follower_modular]: Object close (ratio=0.250), switching to TRACKING
[INFO] [1777043250.552036031] [object_follower_modular]: [NAV2 CANCEL] Requesting cancellation
[INFO] [1777043250.558342722] [object_follower_modular]: [NAV2 CANCEL] Cancellation processed
[INFO] [1777043250.566101707] [object_follower_modular]: [NAV2 RESULT] Status: CANCELED
[INFO] [1777043250.650279271] [object_follower_modular]: TRACKING LOOP STARTED!

[INFO] [1777043279.824197006] [object_follower_modular]: MISSION SUCCESS! Target reached and acquired.
[INFO] [1777043279.829967091] [object_follower_modular]: MISSION SUCCESS! Target acquired.
[INFO] [1777043279.835309093] [object_follower_modular]: =====
```

Gambar 4.4 Log terminal yang menampilkan bukti transisi terjadi tanpa jeda

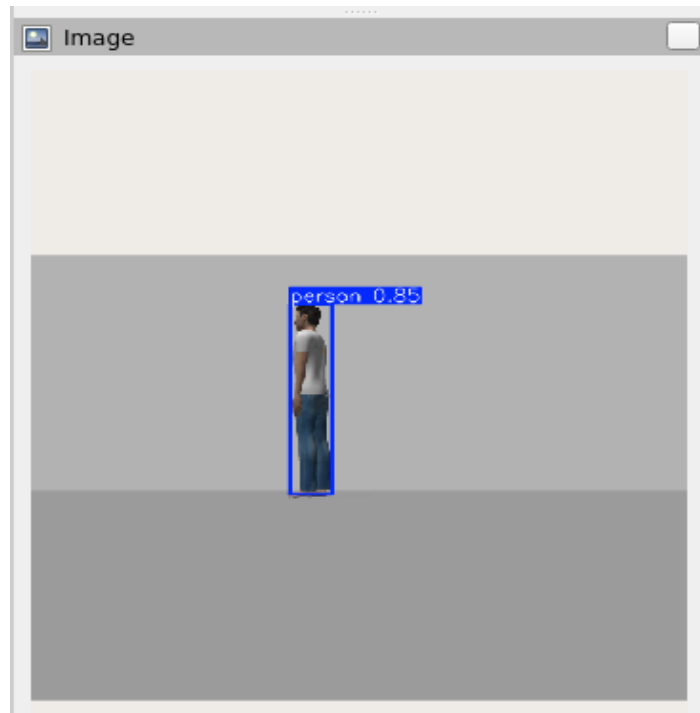
4.1.3 Validasi Visual Servoing dan Centroid Tracker

Setelah FSM berpindah ke state APPROACHING, kendali robot sepenuhnya dialihkan ke modul Visual Servoing. Subbab ini memvalidasi bahwa robot mampu mendekati target secara presisi dan Centroid Tracker berhasil mempertahankan identitas objek sepanjang fase pendekatan berlangsung.

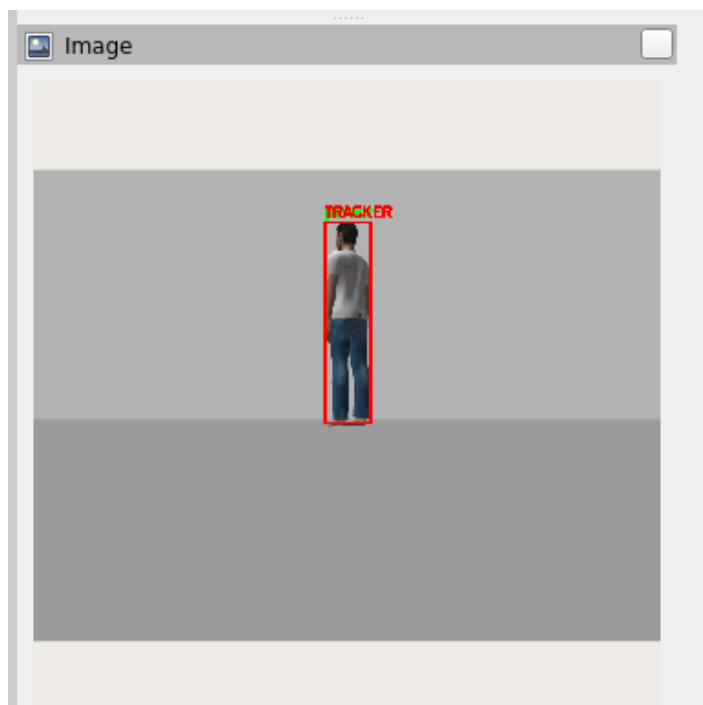
Pada state APPROACHING, robot mendekati objek target menggunakan metode Image-Based Visual Servoing (IBVS). Robot mengoreksi arah hadapnya berdasarkan posisi horizontal pusat bounding box objek pada frame kamera, dan menghentikan pergerakannya ketika rasio tinggi bounding box terhadap tinggi citra melampaui ambang batas 0,55. Keberhasilan fase ini dikonfirmasi oleh nilai `result = SUCCESS` pada seluruh 40 trial yang tercatat dalam berkas CSV.

Selama fase APPROACHING berlangsung, Centroid Tracker beroperasi secara paralel untuk mempertahankan identitas objek target antarframe. Validasi terhadap modul ini dilakukan secara implisit melalui keberhasilan fase pendekatan: apabila tracker gagal mempertahankan identitas objek, robot akan kembali ke state SEARCHING sebelum misi selesai. Kenyataannya, tidak satu pun trial yang mengalami hal demikian, sehingga Centroid Tracker dinyatakan bekerja secara stabil.

Terdapat perbedaan visualisasi antara dua topic yang perlu dipahami, yaitu bounding box deteksi YOLO11n ditampilkan berwarna hijau dan bounding box dari Centroid Tracker berwarna merah pada topic `/annotated_image`. Warna biru yang tampak pada topic `/inference_result` merupakan warna default dari fungsi bawaan Ultralytics (`results[0].plot()`) dan tidak merepresentasikan informasi tracker.



Gambar 4.5 Tampilan kamera robot ketika subscribe topic `/inference_result`



Gambar 4.6 Tampilan kamera robot ketika subscribe topic `/annotated_image`

4.2 Hasil Eksperimen

Bagian ini menyajikan data kuantitatif dari 40 kali uji coba yang terbagi dalam empat kombinasi strategi dan posisi target. Sebelum data disajikan, subbab pertama akan menjelaskan

aturan penamaan strategi dan cara membaca parameter statistik yang digunakan, agar seluruh tabel berikutnya dapat dipahami secara konsisten.

4.2.1 Parameter Statistik yang Digunakan

Sebelum data disajikan, dua hal perlu dijelaskan, yaitu aturan penamaan strategi yang digunakan dan definisi parameter statistik yang menjadi dasar analisis. Pertama, aturan penamaan strategi. Dalam sistem, dua mode pencarian diberi label RANDOM dan SYSTEMATIC. Pada naskah ini, label tersebut disajikan dengan nama yang lebih deskriptif: RANDOM disebut Random Exploration dan SYSTEMATIC disebut Waypoint-Based Search, karena robot mengikuti urutan waypoint yang telah ditetapkan. Penamaan ini digunakan secara konsisten di seluruh tabel dan narasi. Kedua, parameter statistik yang digunakan. Penelitian ini menggunakan dua ukuran statistik utama. Mean (rata-rata) dihitung dengan rumus:

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

Gambar 4.7 Rumus Perhitungan Mean

Keterangan:

$\bar{x}$ = Nilai rata-rata (mean)

X_i = Nilai data ke-i (e.g waktu trial ke-1, trial ke-2, dst.)

n = Jumlah total data pengujian (dalam eksperimen ini, $n = 10$ per kelompok)

Nilai Mean merepresentasikan performa rata-rata robot dalam satu kelompok pengujian; semakin kecil nilainya, semakin cepat atau efisien robot dalam menyelesaikan misi. Standard Deviation (SD) dihitung dengan rumus:

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

Gambar 4.8 Rumus Perhitungan Standar Deviasi (SD)

Keterangan:

SD = Standar deviasi

X_i = Nilai data ke- i

$\bar{x}$ = Nilai rata-rata yang telah dihitung sebelumnya

$n - 1$ = Jumlah total data pengujian (dalam eksperimen ini, $n = 10$ per kelompok)

SD merepresentasikan tingkat konsistensi atau stabilitas hasil pengujian. Nilai SD yang rendah mengindikasikan bahwa robot menghasilkan performa yang relatif seragam di setiap trial. Sebaliknya, nilai SD yang tinggi mengindikasikan variabilitas yang besar, di mana terdapat perbedaan signifikan antara trial terbaik dan terburuk dalam kelompok yang sama.

4.2.2 Hasil Eksperimen Posisi Target 1 (map_fix1)

Pengujian pertama dilakukan dengan target ditempatkan di Posisi 1 pada peta map_fix1. Posisi ini berfungsi sebagai konfigurasi dasar untuk membandingkan kedua strategi dalam kondisi lingkungan yang identik. Masing-masing strategi dijalankan sebanyak 10 trial, sehingga total terdapat 20 trial pada posisi ini. Seluruh trial berhasil diselesaikan dengan success rate 100%. Rekapitulasi statistik pengujian disajikan pada Tabel 4.1.

Tabel 4.1 Hasil Eksperimen Posisi Target 1

Metrik	Random Exploration		Waypoint-Based Search	
	Mean	SD	Mean	SD
Time to Find (s)	164,80	142,48	50,55	8,46
Time to Reach (s)	31,78	16,21	27,30	6,00
Total Time (s)	196,58	142,83	77,85	9,29
Path Length (m)	32,34	25,75	10,50	0,93
Safety Stops (kali)	2,20	6,96	2,20	6,96
Success Rate	100%		100%	

Pada konfigurasi Posisi 1, strategi Waypoint-Based Search mencatat keunggulan yang konsisten di seluruh metrik pencarian, baik dari sisi kecepatan menemukan target maupun efisiensi jalur yang ditempuh. Nilai rata-rata Time to Find strategi Waypoint sebesar 50,55 detik dengan SD 8,46 detik membuktikan konsistensi robot dalam menemukan target pada rentang waktu yang sempit (34 s hingga 65 s). Sementara itu, strategi Random Exploration

memerlukan rata-rata 164,80 detik dengan SD jauh lebih besar (142,48 s), mencerminkan hasil antar-trial yang sangat bervariasi: ada trial yang selesai dalam 34 detik, namun ada pula yang memerlukan hingga 437 detik. Pola serupa tampak pada metrik Path Length. Strategi Waypoint rata-rata menempuh 10,50 meter, sedangkan Random dapat mencapai hingga 81,39 meter pada trial terlama.

Sebagai gambaran konkret, strategi Waypoint rata-rata menemukan target dalam waktu kurang dari satu menit pada Posisi 1, sedangkan pada trial terlama strategi Random membutuhkan lebih dari tujuh menit dengan rute perjalanan yang hampir delapan kali lebih panjang.

4.2.3 Hasil Eksperimen Posisi Target 2 (map_fix2)

Pada pengujian kedua, target dipindahkan ke Posisi 2 menggunakan peta map_fix2. Perubahan posisi ini bertujuan untuk menguji apakah keunggulan salah satu strategi tetap konsisten terhadap posisi target atau bergantung pada kesesuaian antara strategi dengan tata letak lingkungan. Konfigurasi pengujian identik dengan sebelumnya (10 trial per strategi, success rate 100%). Hasil statistik disajikan pada Tabel 4.2.

Tabel 4.2 Hasil Eksperimen Posisi Target 2

Metrik	Random Exploration		Waypoint-Based Search	
	Mean	SD	Mean	SD
Time to Find (s)	87,00	68,05	183,64	13,27
Time to Reach (s)	35,63	27,26	27,50	3,71
Total Time (s)	122,63	75,49	211,14	14,61
Path Length (m)	18,92	12,96	38,26	2,00
Safety Stops (kali)	3,70	7,89	0,00	0,00
Success Rate	100%		100%	

Pada Posisi 2, hasil pengujian memperlihatkan kondisi yang berlawanan dengan Posisi 1: strategi Random Exploration kali ini mencatat waktu penemuan target yang lebih singkat dibandingkan Waypoint-Based Search. Strategi Random Exploration mencatat rata-rata Time to Find sebesar 87,00 detik, lebih rendah dibandingkan Waypoint-Based Search yang memerlukan 183,63 detik. Meskipun demikian, nilai SD Random Exploration (68,05 s) tetap lebih besar dibandingkan Waypoint-Based Search (13,27 s), yang menunjukkan konsistensi

hasil Waypoint masih lebih terjaga. Temuan yang signifikan pada posisi ini adalah strategi Waypoint-Based Search tidak mencatat satupun safety stop di seluruh 10 trial (mean = 0,00; SD = 0,00), menunjukkan jalur waypoint yang ditetapkan sepenuhnya bebas dari area berpotensi hambatan.

Temuan ini menunjukkan keunggulan strategi Waypoint-Based Search tidak bersifat mutlak. Pada kondisi tanpa titik patroli yang berdekatan dengan lokasi target, strategi Random Exploration memiliki lebih banyak peluang untuk mendeteksi target lebih awal, meskipun hasilnya tidak dapat diprediksi secara konsisten dari satu trial ke trial berikutnya.

4.2.4 Rekapitulasi Keseluruhan

Untuk memudahkan perbandingan secara menyeluruh, seluruh hasil dari keempat kombinasi dirangkum dalam satu tabel ringkasan. Tabel ini berfungsi sebagai referensi komprehensif sebelum analisis komparatif yang lebih mendalam disajikan pada bagian berikutnya.

Tabel 4.3 Tabel Rekapitulasi (Mean $\pm$ SD)

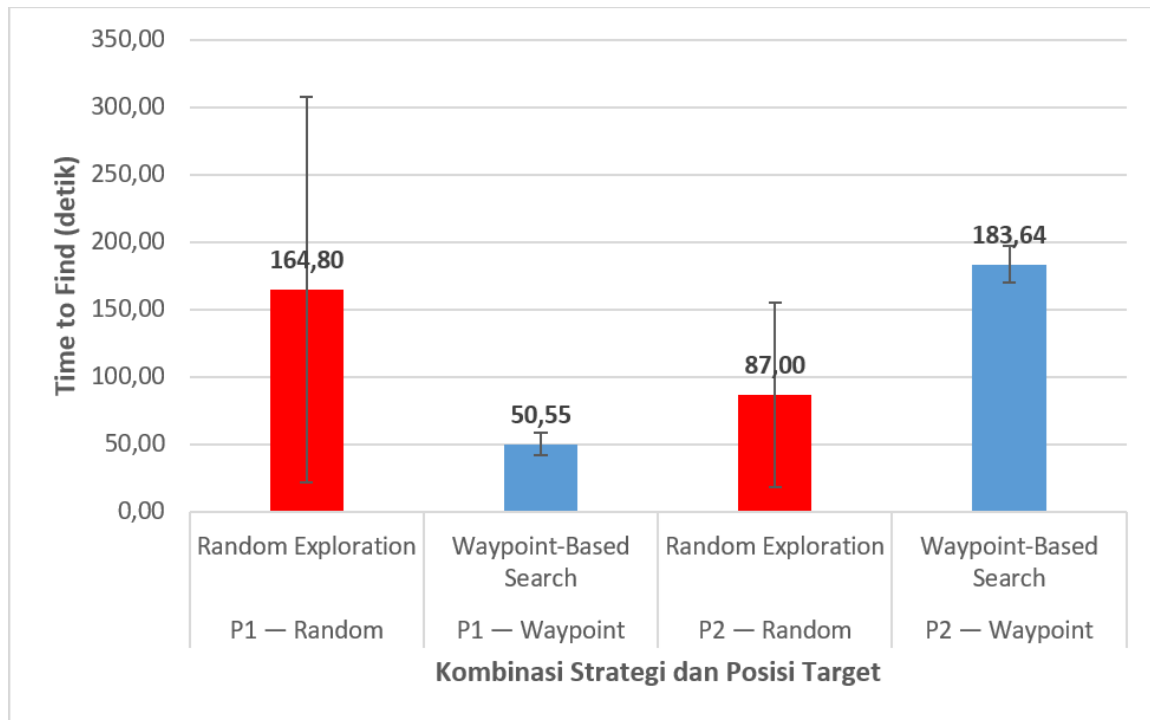
Kombinasi	Time to Find (s)	Total Time (s)	Time to Reach (s)	Path Length (m)	Safety Stops (kali)
Posisi 1 – Waypoint	50,55 $\pm$ 8,46	77,84 $\pm$ 9,29	10,50 $\pm$ 0,93	2,20 $\pm$ 6,96	100%
Posisi 1 – Random	164,80 $\pm$ 142,48	196,58 $\pm$ 142,83	32,34 $\pm$ 25,75	2,20 $\pm$ 6,96	100%
Posisi 2 – Waypoint	183,64 $\pm$ 13,27	211,14 $\pm$ 14,61	38,26 $\pm$ 2,00	0,00 $\pm$ 0,00	100%
Posisi 2 – Random	87,00 $\pm$ 68,05	122,62 $\pm$ 75,49	18,92 $\pm$ 12,96	3,70 $\pm$ 7,89	100%

4.3 Perbandingan Strategi

Bagian sebelumnya menyajikan data tiap posisi secara terpisah. Pada bagian ini, kedua strategi dibandingkan secara langsung melalui visualisasi grafik untuk mempermudah identifikasi selisih performa dan tingkat konsistensi masing-masing. Empat metrik utama yang dibandingkan adalah Time to Find, Total Time, Path Length, dan Safety Stops.

4.3.1 Perbandingan Time to Find

Time to Find merupakan metrik paling representatif dalam penelitian ini karena secara langsung mengukur efisiensi fase pencarian, yang merupakan inti dari misi yang dirancang.



Gambar 4.9 Bar Chart Perbandingan Time to Find Keempat Kombinasi

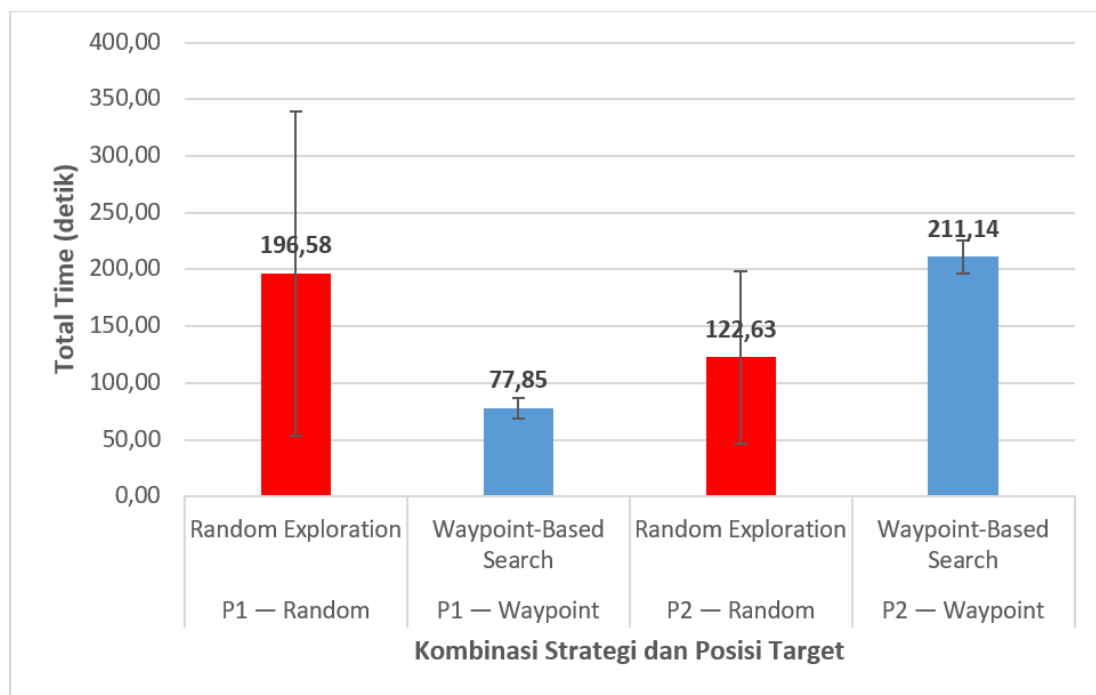
Grafik pada Gambar 4.9 memperlihatkan secara visual perbedaan yang tidak hanya pada tinggi batang, tetapi juga pada panjang garis error bar yang mencerminkan konsistensi masing-masing strategi. Berdasarkan Gambar 4.9, perbedaan karakteristik kedua strategi tampak secara jelas melalui panjang error bar masing-masing. Strategi Waypoint-Based Search menampilkan error bar yang pendek di kedua posisi target ($SD = 8,46$ s pada Posisi 1 dan $SD = 13,27$ s pada Posisi 2), yang mengonfirmasi urutan kunjungan waypoint menghasilkan waktu temuan target yang konsisten dan mudah diprediksi. Di sisi lain, strategi Random Exploration menampilkan error bar yang jauh lebih panjang ($SD = 142,48$ s pada Posisi 1 dan $SD = 68,05$ s pada Posisi 2). Artinya, meskipun strategi ini berhasil menemukan target dalam waktu minimum 33,53 detik pada Posisi 1, performa tersebut tidak menjamin konsistensi dari satu trial ke trial berikutnya; pada trial terburuk, waktu yang diperlukan mencapai 436,55 detik.

Grafik ini memperlihatkan strategi Waypoint-Based Search tidak hanya lebih cepat pada Posisi 1, tetapi juga lebih dapat diandalkan di kedua posisi target. Strategi Random Exploration,

meski sesekali mencatat waktu pencarian yang sangat singkat, tidak menjamin konsistensi hasil di setiap trial.

4.3.2 Perbandingan Total Time

Total Time merupakan ukuran efisiensi misi secara end-to-end, yang mencakup durasi dari inisiasi pencarian hingga robot berhenti di hadapan target. Metrik ini memberikan gambaran performa sistem secara keseluruhan, tidak terbatas pada fase pencarian saja.



Gambar 4.10 Bar Chart Perbandingan Total Time

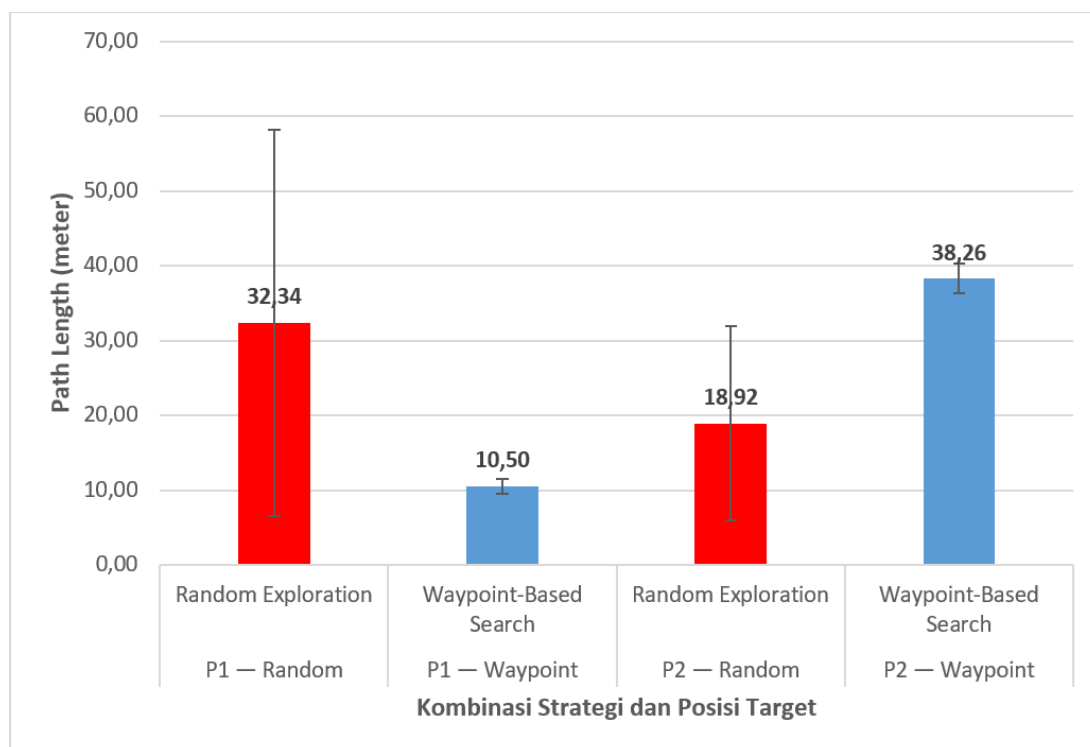
Distribusi Total Time mencerminkan pola yang hampir identik dengan pola Time to Find, karena durasi fase pendekatan pada kedua strategi tidak menunjukkan perbedaan yang signifikan. Perbedaan ini terlihat dari nilai Time to Reach kedua strategi yang berada dalam rentang yang berdekatan, berkisar antara 27 hingga 36 detik. Selisih besar yang terlihat pada Total Time sepenuhnya berasal dari perbedaan pada fase pencarian. Strategi Waypoint lebih unggul pada Posisi 1 (77,85 s berbanding 196,58 s), sedangkan strategi Random mencatat Total Time lebih rendah pada Posisi 2 (122,63 s berbanding 211,14 s).

Oleh karena itu, perbedaan efisiensi misi secara keseluruhan antara kedua strategi sebagian besar ditentukan oleh efisiensi fase pencarian, bukan oleh kemampuan robot dalam mendekati target setelah target berhasil ditemukan. Pola panjang error bar pada grafik ini pun

konsisten dengan temuan pada 4.3.1: kelompok Waypoint menampilkan sebaran yang lebih sempit, sedangkan kelompok Random menampilkan sebaran yang lebih lebar di kedua posisi.

4.3.3 Perbandingan Path Length

Path Length mencerminkan efisiensi penggunaan ruang gerak. Semakin pendek total jarak yang ditempuh robot untuk menemukan target, semakin efisien pula strategi tersebut dalam memanfaatkan lingkungan simulasi.



Gambar 4.11 Bar Chart Perbandingan Path Length

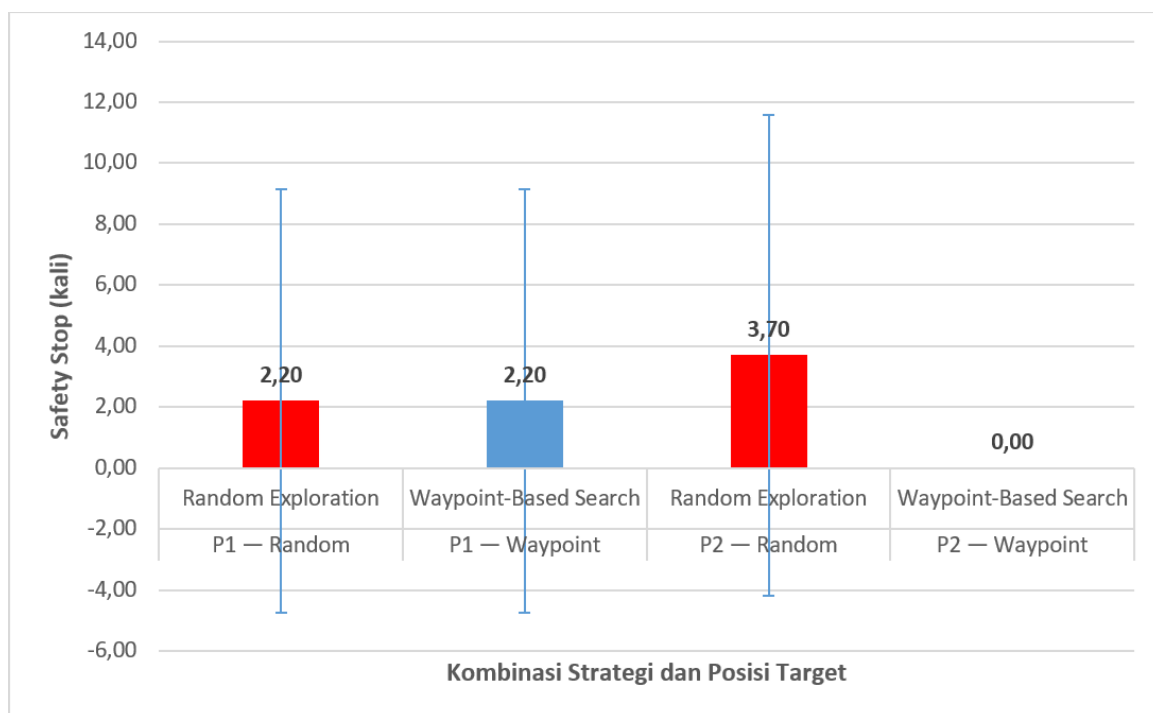
Metrik Path Length memberikan gambaran tentang sejauh mana pola navigasi masing-masing strategi memengaruhi jarak total yang ditempuh robot selama satu sesi misi. Strategi Random Exploration berpotensi menghasilkan lintasan yang jauh lebih panjang dibandingkan Waypoint-Based Search, karena robot bernavigasi tanpa struktur arah yang terencana. Pada Posisi 1, strategi Random dapat menempuh hingga 81,39 meter dalam satu trial, sedangkan strategi Waypoint mencatat nilai maksimum hanya 12,57 meter. Nilai SD Path Length pada strategi Waypoint di Posisi 2 yang sangat kecil ($SD = 2,00$ m) juga mengonfirmasi konsistensi rute yang ditempuh robot di setiap trial.

Strategi Waypoint secara konsisten menghasilkan lintasan yang lebih terstruktur dan teratur, sementara strategi Random berpotensi membawa robot bergerak dalam jarak yang jauh

lebih panjang tanpa jaminan efisiensi perjalanan. Sebagaimana pada subbab sebelumnya, perbedaan ini juga tercermin pada panjang error bar grafik: error bar yang pendek pada kelompok Waypoint mengonfirmasi keteraturan rute yang ditempuh, sementara error bar yang panjang pada kelompok Random mencerminkan variasi jarak yang besar antar-trial..

4.3.4 Perbandingan Safety Stops

Safety stop terjadi setiap kali sensor LiDAR mendeteksi objek pada jarak yang terlalu dekat di depan robot, sehingga sistem menghentikan pergerakan secara sementara. Frekuensi safety stop mencerminkan seberapa sering robot berhadapan dengan situasi berisiko sepanjang misi berlangsung.



Gambar 4.12 Bar Chart Perbandingan Safety Stops

Frekuensi safety stop mencerminkan seberapa sering robot menghadapi situasi yang memerlukan respons penghindaran selama misi. Nilainya sangat dipengaruhi oleh rute yang dihasilkan pada setiap trial. Pada grafik Safety Stops, error bar bawah dari beberapa kelompok tampak menembus ke bawah nilai nol pada sumbu Y. Kondisi ini terjadi karena nilai SD pada kelompok tersebut melebihi nilai rata-ratanya (contoh: kelompok P1 -- Random, mean = 2,20 dan SD = 6,96). Secara matematis, kondisi ini wajar: sebaran data sangat lebar karena sebagian besar trial mencatat nol safety stop, sementara sejumlah kecil trial outlier mencatat nilai yang

sangat tinggi. Dalam konteks pengujian ini, nilai safety stop tidak mungkin negatif, sehingga batas bawah error bar secara praktis dibaca sebagai nol.

Strategi Waypoint-Based Search pada Posisi 2 tidak mencatat satu pun safety stop di seluruh 10 trial (mean = 0,00; SD = 0,00), menunjukkan jalur waypoint bebas dari area hambatan. Sementara itu, strategi Random Exploration pada posisi yang sama mencatat rata-rata 3,70 safety stop dengan SD 7,89, menandakan jalur navigasi acak lebih rentan membawa robot ke posisi berisiko. Besarnya nilai SD pada seluruh kelompok menunjukkan frekuensi safety stop yang tinggi hanya terjadi pada sejumlah kecil trial tertentu; sebagian besar trial dari kedua strategi mencatat nol safety stop, sehingga nilai rata-rata yang muncul di tabel lebih mencerminkan dampak dari trial outlier tersebut daripada perilaku khas masing-masing strategi.

Hasil yang paling signifikan pada metrik ini pada metrik ini adalah jalur patroli Waypoint pada Posisi 2 sepenuhnya terbebas dari safety stop di semua 10 trial, membuktikan rute yang ditetapkan tidak melintasi area yang berpotensi memicu sensor jarak.

4.4 Pembahasan dan Temuan Penting

Data yang disajikan pada bagian sebelumnya menggambarkan performa sistem secara kuantitatif. Bagian ini menganalisis makna di balik angka-angka tersebut, di antaranya faktor apa yang menyebabkan suatu strategi lebih efisien pada kondisi tertentu, bagaimana sistem merespons situasi yang tidak ideal, dan apa implikasi dari temuan-temuan ini terhadap pengembangan sistem lebih lanjut.

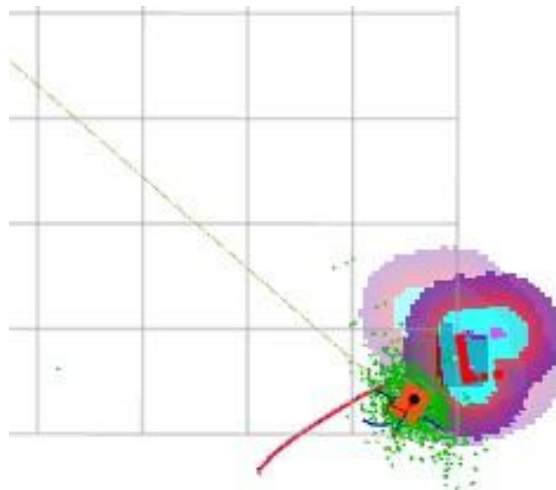
4.4.1 Analisis Self-Recovery dan Robustness Sistem

Keberhasilan sistem otonom tidak hanya diukur dari seberapa cepat ia menyelesaikan misi, tetapi juga dari kemampuannya untuk tetap berfungsi ketika kondisi lapangan tidak berjalan sesuai rencana. Subbab ini menganalisis kejadian safety stop berulang sebagai bukti empiris ketangguhan (robustness) arsitektur FSM yang dirancang. Sepanjang 40 trial pengujian, terdapat beberapa trial yang mencatat jumlah safety stop jauh di atas rata-rata kelompoknya, yaitu:

- a. Trial 8 Posisi 1 (Random Exploration): 22 kali safety stop
- b. Trial 3 Posisi 1 (Waypoint-Based Search): 22 kali safety stop
- c. Trial 1 Posisi 2 (Random Exploration): 21 kali safety stop
- d. Trial 9 Posisi 2 (Random Exploration): 16 kali safety stop

Temuan yang signifikan adalah bahwa seluruh trial tersebut tetap berakhir dengan status SUCCESS, meskipun safety stop yang berulang menyebabkan kenaikan pada nilai Total Time dan Path Length. Tidak satu pun trial mengalami kondisi macet (situasi di mana robot terhenti sepenuhnya dan tidak dapat melanjutkan misi) ataupun penghentian total akibat kegagalan sistem.

Kondisi ini merupakan bukti keberhasilan mekanisme self-recovery pada arsitektur FSM. Ketika sensor LiDAR mendeteksi hambatan di depan robot pada fase pendekatan, sistem secara otomatis mengeksekusi urutan pemulihan, yaitu robot mundur dari zona hambatan, mengevaluasi sisi kiri dan kanan berdasarkan data LiDAR, kemudian berputar ke arah yang lebih lapang. Apabila hambatan masih terdeteksi setelah durasi tiga detik, FSM mengaktifkan mode bypass dengan mengirimkan tujuan sementara ke Nav2 pada jarak 1,5 meter dari posisi robot saat itu, sebelum mengembalikan sistem ke state SEARCHING untuk melanjutkan misi dari posisi yang lebih menguntungkan. Kemampuan ini membuktikan bahwa sistem dirancang tidak hanya untuk kondisi ideal, melainkan juga mampu memulihkan diri secara mandiri tanpa intervensi operator.



Gambar 4.13 Visualisasi lintasan robot di RViz ketika robot melakukan safety stop

4.4.2 Hubungan Strategi dengan Tata Letak Lingkungan

Salah satu temuan terpenting dari keseluruhan pengujian ini adalah tidak terdapat strategi yang secara universal lebih unggul dalam semua kondisi. Performa masing-masing strategi

sangat bergantung pada kesesuaian antara pola navigasi yang dilaksanakan dengan posisi target di lingkungan simulasi.

Pada Posisi Target 1, strategi Waypoint-Based Search mencatat keunggulan yang signifikan (50,55 s berbanding 164,80 s). Hal ini disebabkan oleh kedekatan salah satu titik waypoint dengan lokasi target, sehingga robot berhasil mendeteksi target pada iterasi awal patroli. Nilai SD Waypoint yang rendah (8,46 s) mengonfirmasi bahwa pola ini terjadi secara konsisten di setiap trial.

Pada Posisi Target 2, kondisi berbalik: strategi Random Exploration lebih unggul (87,00 s berbanding 183,64 s). Bukti empiris atas penjelasan ini ditemukan pada nilai SD strategi Waypoint di posisi ini yang sangat kecil (13,27 s), dengan rentang nilai antara 167 s hingga 209 s di seluruh 10 trial. Konsistensi yang demikian ekstrem ini mengindikasikan bahwa robot selalu harus menyelesaikan hampir seluruh urutan kunjungan waypoint sebelum mencapai area target, karena tidak ada titik waypoint yang berada dekat dengan lokasi target. Strategi Random Exploration, di sisi lain, memiliki peluang probabilistik untuk menemukan target pada iterasi awal pergerakannya, walaupun dengan konsekuensi variabilitas yang lebih tinggi (SD = 68,05 s).

Temuan ini dapat dipahami melalui prinsip dasar navigasi berbasis waypoint, yaitu efisiensi strategi ini bergantung langsung pada seberapa dekat titik-titik patroli yang ditetapkan dengan lokasi target di lingkungan. Apabila tidak ada titik waypoint yang berada di dekat target, robot harus menyelesaikan hampir seluruh urutan kunjungan sebelum target dapat terdeteksi, sebagaimana yang terjadi pada Posisi 2 dalam penelitian ini. Sebaliknya, strategi Random Exploration tidak bergantung pada pengetahuan tentang posisi target, sehingga mekanisme pencariannya tidak terpengaruh oleh perubahan posisi target, meski dengan konsekuensi hasil yang kurang dapat diprediksi.

4.4.3 Perbandingan Karakteristik Kedua Strategi

Berdasarkan seluruh temuan eksperimen yang telah diuraikan, Tabel 4.4 merangkum karakteristik utama kedua strategi untuk memudahkan pengambilan keputusan dalam pemilihan strategi pada skenario berbeda.

Tabel 4.4 Tabel Perbandingan Karakteristik Strategi Pencarian

Aspek	Random Exploration	Waypoint-Based Search
Prediktabilitas	Rendah (SD tinggi: 68-142 s)	Tinggi (SD rendah: 8-13 s)

Aspek	Random Exploration	Waypoint-Based Search
Fleksibilitas	Tinggi (tidak memerlukan konfigurasi titik patroli)	Rendah (terikat urutan waypoint yang ditetapkan)
Efisiensi Jalur	Rendah (potensi path length sangat panjang)	Tinggi (jalur lebih terstruktur dan terukur)
Ketergantungan Posisi Target	Tinggi (bergantung pada peluang acak robot bertemu target)	Tinggi (bergantung kesesuaian urutan waypoint dengan posisi target)
Rekomendasi Penggunaan	Distribusi posisi target tidak diketahui atau tidak dapat diprediksi	Titik patroli dapat dirancang mendekati area kemungkinan target

4.5 Evaluasi Sistem

Bagian ini mendiskusikan kelebihan yang berhasil ditunjukkan sistem selama pengujian dan keterbatasan-keterbatasan yang diidentifikasi selama proses pengembangan dan pengujian sistem. Pengakuan terhadap keterbatasan ini bukan bertujuan untuk melemahkan hasil yang telah dicapai, melainkan untuk menunjukkan pemahaman yang komprehensif terhadap sistem yang dibangun serta batas-batas validitas temuan yang diperoleh.

4.5.1 Kelebihan Sistem

Berdasarkan hasil pengujian keseluruhan, sistem yang dikembangkan menunjukkan sejumlah kelebihan yang relevan sebagai dasar penilaian kelayakan sistem dalam lingkup penelitian ini:

- Tingkat keberhasilan misi 100%
Seluruh 40 trial diselesaikan dengan status SUCCESS tanpa satu pun kegagalan, membuktikan keandalan sistem dalam kondisi pengujian yang ditetapkan.
- Kemampuan self-recovery yang andal
Pada trial dengan safety stop berulang (hingga 22 kali), robot berhasil memulihkan diri secara mandiri melalui mekanisme mundur, evaluasi arah, dan bypass Nav2. Tidak satu pun trial berakhir dengan kondisi macet (deadlock).
- Konsistensi tinggi pada strategi Waypoint.

Nilai SD Time to Find yang rendah (8,46 s pada Posisi 1 dan 13,27 s pada Posisi 2) menunjukkan robot menghasilkan waktu temuan target yang dapat diprediksi, berguna untuk skenario operasional yang memerlukan estimasi waktu misi yang akurat.

d. Integrasi pipeline yang mulus

Transisi antar-state FSM (SEARCHING → APPROACHING → MISSION COMPLETE) berlangsung tanpa konflik perintah di seluruh 40 trial, membuktikan koordinasi Nav2, YOLO11n, dan Visual Servoing berjalan sebagaimana dirancang.

4.5.2 Keterbatasan Sistem

Di samping kelebihan yang telah dipaparkan, terdapat sejumlah keterbatasan yang diidentifikasi selama proses. Meskipun seluruh 40 trial berhasil diselesaikan dengan success rate 100%, terdapat beberapa keterbatasan dengan detail sebagai berikut.

a. Keterbatasan deteksi pada kondisi jarak jauh

Model YOLO11n varian nano dipilih karena kecepatan inferensinya yang tinggi, namun kemampuan deteksinya terhadap objek yang tampak kecil di frame kamera (karena berada pada jarak jauh atau sudut pandang yang sempit) lebih terbatas dibandingkan varian YOLO yang lebih besar. Kondisi ini berpotensi menyebabkan robot melewati area sekitar target tanpa mendeteksinya.

b. Potensi akumulasi drift odometri pada lintasan panjang

Robot memperkirakan posisinya di peta berdasarkan akumulasi data putaran roda (odometri). Semakin jauh robot bergerak, semakin besar risiko perkiraan posisi tersebut menyimpang dari posisi sebenarnya. Pada trial dengan Path Length lebih dari 50 meter, sistem koreksi posisi otomatis (AMCL) berjalan secara periodik untuk mengurangi penyimpangan ini, namun potensi ketidakakuratan sesaat tetap perlu diantisipasi.

c. Cakupan pengujian terbatas pada satu lingkungan simulasi dan dua posisi target

Generalisasi hasil penelitian ini ke lingkungan yang lebih kompleks, seperti ruangan dengan banyak halangan dinamis, area dengan pencahayaan yang bervariasi, atau lingkungan bertingkat, memerlukan pengujian tambahan yang lebih beragam.

d. Sistem hanya mendukung satu target per misi

Skenario dengan beberapa objek target secara bersamaan, atau target yang bergerak, belum tercakup dalam pengujian ini.

4.6 Pembahasan

Berdasarkan rangkaian pengujian yang telah dilaksanakan, sistem robot mobile otonom yang dikembangkan terbukti mampu menjalankan misi pencarian dan pendekatan objek secara penuh. Hasil validasi pada subbab 4.1 mengonfirmasi seluruh komponen sistem (YOLO11n, FSM, dan Visual Servoing) berfungsi sesuai perancangan dan berhasil dikoordinasikan tanpa konflik perintah di seluruh 40 trial. Success rate 100%, bahkan pada trial yang mengalami safety stop hingga 22 kali, menegaskan sistem tidak hanya berjalan pada kondisi ideal tetapi juga mampu memulihkan diri secara mandiri saat menghadapi hambatan berulang.

Lebih jauh, hasil perbandingan kuantitatif antara kedua strategi pencarian menunjukkan pola yang menarik, yaitu tidak terdapat strategi yang secara konsisten lebih unggul di semua kondisi. Di Posisi 1, strategi Waypoint-Based Search mencatat rata-rata Time to Find jauh lebih rendah (50,55 s berbanding 164,80 s) karena salah satu titik patroli berada dekat lokasi target, sehingga deteksi terjadi pada awal siklus pencarian. Di Posisi 2, kondisi sebaliknya terjadi: tidak ada waypoint yang berdekatan dengan target, menyebabkan robot harus menyelesaikan hampir seluruh urutan patroli sebelum mendeteksi target (183,64 s), sedangkan Random Exploration berhasil lebih cepat secara rata-rata (87,00 s).

Pola tersebut dapat dipahami lebih mendalam melalui perbandingan metrik Time to Reach, yaitu durasi fase pendekatan setelah target terdeteksi. Metrik ini menunjukkan nilai yang hampir seragam di kedua strategi dan kedua posisi target (27 hingga 36 detik). Keseragaman ini membuktikan kemampuan robot mendekati target setelah terdeteksi tidak dipengaruhi oleh strategi pencarian yang digunakan. Dengan demikian, seluruh perbedaan yang terlihat pada Total Time berasal sepenuhnya dari perbedaan efisiensi fase pencarian, bukan dari fase pendekatan.

Temuan mengenai perbedaan hasil antar posisi ini memiliki implikasi langsung bagi penerapan sistem di kondisi nyata. Waypoint-Based Search lebih tepat digunakan apabila pola distribusi posisi target dapat diperkirakan dan titik patroli dapat dirancang mendekati area tersebut. Sebaliknya, Random Exploration lebih sesuai pada kondisi di mana posisi target tidak dapat diprediksi, meski dengan konsekuensi variasi waktu temuan yang lebih besar antar-trial. Pemilihan strategi, pada akhirnya, merupakan keputusan yang perlu disesuaikan dengan karakteristik lingkungan operasional masing-masing skenario.

Seluruh temuan di atas berasal dari pengujian pada satu lingkungan simulasi dengan dua posisi target yang ditetapkan sebelumnya. Penerapan sistem pada lingkungan yang lebih

beragam, seperti ruangan dengan halangan dinamis atau posisi target yang tidak tetap, masih memerlukan pengujian lanjutan sebelum hasilnya dapat diterapkan lebih jauh.

Secara keseluruhan, penelitian ini berhasil membuktikan bahwa arsitektur pipeline berbasis FSM merupakan pendekatan yang layak dan andal untuk misi pencarian serta pendekatan objek pada mobile robot otonom. Ketergantungan performa terhadap relasi antara titik patroli dan posisi target memberikan arah yang konkret bagi pengembangan selanjutnya, yaitu merancang mekanisme pencarian yang lebih adaptif terhadap kondisi lingkungan yang bervariasi.

4.7 Komparasi dengan Penelitian Terdahulu

Agar perbandingan lebih objektif, kinerja sistem pada penelitian ini dikomparasikan dengan penelitian terdahulu yang sama-sama diuji pada lingkungan simulasi (ROS dan Gazebo). Pendekatan ini dipilih agar kondisi lingkungan virtual dan batasan sensor yang digunakan relatif setara. Dua penelitian yang menjadi acuan komparasi adalah "Implementasi robot patroli sederhana berbasis computer vision dan deep learning" oleh Ananda & Dewa (2023) dan "Penerapan model deteksi objek untuk robot menggunakan model SSD di lingkungan simulasi ROS" oleh Tamadjoe & Dewa (2024).

4.7.1 Aspek Navigasi dan Eksplorasi

Pada penelitian Ananda & Dewa (2023), pergerakan robot masih bergantung pada titik koordinat tujuan yang sudah diatur secara statis sejak awal. Hal serupa juga terlihat pada simulasi Tamadjoe & Dewa (2024), di mana robot hanya ditempatkan diam di satu titik untuk mengukur jarak objek. Sebagai bentuk pengembangan, penelitian ini menerapkan sistem navigasi yang lebih mandiri. Dengan menggunakan Nav2 pada state SEARCHING, robot dapat bergerak menjelajahi area untuk mencari target, baik menggunakan pola pencarian acak (random) maupun rute yang telah ditentukan (waypoint).

4.7.2 Aspek Deteksi dan Pelacakan Objek

Dari sisi persepsi visual, penelitian Tamadjoe & Dewa (2024) yang menggunakan model SSD MobileNet V2 mendapati bahwa batas deteksi maksimal robot hanya sejauh 2,71 meter (dengan nilai mAP 58,6%). Sementara itu, simulasi Ananda & Dewa (2023) menggunakan YOLOv5 memiliki kemampuan deteksi yang lebih jauh hingga 10 meter. Namun, sistem mereka hanya sebatas mendeteksi target dan menyimpan gambar, tanpa adanya fitur untuk

melacak identitas objek secara berkelanjutan antar-frame. Penelitian ini memperbaiki kedua kelemahan tersebut. Penggunaan model YOLO11n membuat kemampuan deteksi objek di dalam Gazebo menjadi lebih andal. Selain itu, penambahan algoritma Centroid Tracker memastikan robot tetap dapat mengenali dan mengunci satu target yang sama, meskipun robot dan target tersebut sedang sama-sama bergerak.

4.7.3 Aspek Manuver Pendekatan (Visual Servoing)

Keunggulan utama dari penelitian ini dibandingkan kedua simulasi sebelumnya ada pada fase pendekatan target (APPROACHING). Setelah target berhasil dikunci oleh kamera, sistem FSM akan mengalihkan kendali gerak robot dari Nav2 ke algoritma Image-Based Visual Servoing (IBVS). Fitur ini memungkinkan robot untuk bermanuver mendekati target secara halus dan berhenti otomatis pada jarak aman (saat rasio bounding box menyentuh angka 0,55). Rangkaian proses yang berkesinambungan dari mencari target, melacak identitas, hingga bermanuver mendekat ini belum diimplementasikan pada kedua penelitian simulasi sebelumnya.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan seluruh tahapan perancangan, implementasi, dan pengujian yang telah dilakukan, penelitian ini berfokus pada pengembangan sistem robot mobile otonom berbasis Finite State Machine (FSM) yang mengintegrasikan navigasi waypoint, deteksi objek dengan YOLO11n, dan pendekatan target melalui Visual Servoing. Adapun kesimpulan yang dapat ditarik dari penelitian ini adalah sebagai berikut.

- A. Penelitian ini berhasil mengembangkan sistem robot mobile otonom dengan arsitektur pipeline terintegrasi yang mencakup empat fase operasi secara berurutan: pencarian target, deteksi visual, pelacakan objek, dan pendekatan target. Seluruh transisi antar-state FSM (SEARCHING → APPROACHING → MISSION COMPLETE) berlangsung tanpa konflik perintah di semua 40 trial pengujian, membuktikan keberhasilan integrasi komponen Nav2, YOLO11n, dan Visual Servoing dalam satu pipeline yang kohesif.
- B. Strategi Waypoint-Based Search menunjukkan keunggulan yang nyata pada Posisi Target 1, dengan rata-rata Time to Find sebesar 50,55 s (SD = 8,46 s) dibandingkan 164,80 s (SD = 142,48 s) pada strategi Random Exploration. Nilai SD yang rendah pada strategi Waypoint mengonfirmasi konsistensi hasil yang tinggi dan waktu temuan target yang dapat diprediksi.
- C. Pada Posisi Target 2, strategi Random Exploration mencatat rata-rata Time to Find yang lebih rendah (87,00 s) dibandingkan Waypoint-Based Search (183,64 s). Temuan ini menunjukkan tidak terdapat strategi yang secara universal lebih unggul; efisiensi masing-masing strategi sangat bergantung pada kesesuaian antara pola navigasinya dengan posisi target di lingkungan kerja. Strategi Waypoint unggul apabila salah satu titik patroli berada dekat dengan posisi target, sedangkan strategi Random berpotensi lebih cepat apabila tidak ada titik patroli yang berdekatan.
- D. Mekanisme self-recovery pada arsitektur FSM terbukti andal dalam menghadapi kondisi hambatan yang tidak terduga. Seluruh 40 trial berhasil diselesaikan dengan success rate 100%, termasuk trial yang mencatat jumlah safety stop tertinggi (22 kali), tanpa satu pun trial berakhir dalam kondisi macet (deadlock) maupun kegagalan sistem.

- E. Secara keseluruhan, sistem yang dikembangkan mampu menjalankan misi pencarian dan pendekatan target secara otonom di lingkungan simulasi dengan tingkat keberhasilan penuh. Hasil ini menunjukkan bahwa integrasi FSM berbasis waypoint dengan deteksi objek berbasis deep learning merupakan pendekatan yang layak untuk sistem robot mobile otonom pada skenario pencarian target dalam ruang tertutup.

5.2 Saran

Berdasarkan hasil penelitian dan keterbatasan yang diidentifikasi, beberapa saran untuk pengembangan selanjutnya adalah sebagai berikut.

- A. Peningkatan kemampuan deteksi objek

Penelitian selanjutnya dapat mempertimbangkan penggunaan model YOLO dengan varian yang lebih besar (small atau medium) apabila kapasitas komputasi memungkinkan, atau melakukan fine-tuning pada dataset yang lebih representatif terhadap lingkungan pengujian. Peningkatan ini diharapkan memperluas jangkauan deteksi yang efektif, terutama pada kondisi objek target berada pada jarak jauh atau sudut pandang yang sempit.

- B. Adaptive Waypoint Generation

Titik-titik patroli pada strategi Waypoint-Based Search saat ini ditentukan secara statis berdasarkan dimensi peta. Penelitian selanjutnya dapat mengeksplorasi pendekatan adaptive waypoint generation, di mana titik patroli diperbarui secara dinamis berdasarkan riwayat deteksi pada sesi sebelumnya. Pendekatan ini berpotensi meningkatkan efisiensi pencarian secara bertahap seiring akumulasi pengalaman operasional robot.

- C. Validasi pada lingkungan fisik nyata

Pengujian dalam penelitian ini dilakukan pada lingkungan simulasi Gazebo. Validasi pada platform robot fisik diperlukan untuk menguji ketahanan sistem terhadap variabilitas kondisi nyata yang tidak tertangkap dalam simulasi, seperti noise sensor yang lebih tinggi, pencahayaan yang berubah-ubah, dan rintangan yang tidak terstruktur.

- D. Perluasan cakupan misi ke skenario multi-target

Sistem saat ini dirancang untuk menangani satu objek target per misi. Pengembangan selanjutnya dapat memperluas cakupan misi ke skenario multi-target, di mana robot

mampu mencari, mendeteksi, dan mendekati lebih dari satu objek target dalam satu sesi operasi, guna meningkatkan kompleksitas dan kegunaan sistem secara praktis.

DAFTAR PUSTAKA

- Alatise, M. B., & Hancke, G. P. (2020). A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. *IEEE Access*, 8, 39830–39846. <https://doi.org/10.1109/ACCESS.2020.2975643>
- Ali, A., & Mirza, S. (2006). Object Tracking using Correlation, Kalman Filter and Fast Means Shift Algorithms. *2006 International Conference on Emerging Technologies*, 174–178. <https://doi.org/10.1109/ICET.2006.335916>
- Ali, M. L., & Zhang, Z. (2024). The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection. *Computers*, 13(12), 336. <https://doi.org/10.3390/computers13120336>
- Ananda, I. S., & Dewa, C. K. (2023). Implementasi Robot Patroli Sederhana Berbasis Computer Vision dan Deep Learning. *AUTOMATA*, 4(2). <https://journal.uui.ac.id/AUTOMATA/article/view/28611>
- Chung, L. B., & Nguyen, D. D. (2023). REAL-TIME OBJECT DETECTION AND TRACKING FOR MOBILE ROBOT USING YOLOV8 AND STRONG SORT. *Universum: Technical sciences*, 116(11). <https://doi.org/10.32743/UniTech.2023.116.11.16223>
- Dai, Y., & Lee, K. (2025). Multi-Sensor Fusion for Autonomous Mobile Robot Docking: Integrating LiDAR, YOLO-Based AprilTag Detection, and Depth-Aided Localization. *Electronics*, 14(14), 2769. <https://doi.org/10.3390/electronics14142769>
- Deng, H., Xiong, J., & Xia, Z. (2017). Mobile manipulation task simulation using ROS with MoveIt. *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 612–616. <https://doi.org/10.1109/RCAR.2017.8311930>

- Dharmasaputra, K. D., & Hartato, B. P. (2025). Pengujian YOLOv8 dan Centroid Tracking pada Sistem Deteksi, Klasifikasi, dan Penghitungan Jumlah Kendaraan. *Journal of Applied Informatics and Computing*, 9(5), 2310–2319. <https://doi.org/10.30871/jaic.v9i5.10873>
- Fiqhi, I. A. R. A., & Joni, K. (2018). Sistem Deteksi Objek dan Posisi pada Robot Sepak Bola Beroda Middle Size pada Sistem Kamera Omni Vision dengan Scan Lines. *The 6th Indonesian Symposium on Roboting System and Control (ISRSC)*, 54–58.
- Gao, X., & Zhang, T. (2021). *Introduction to Visual SLAM: From Theory to Practice*. Springer Singapore. <https://doi.org/10.1007/978-981-16-4939-4>
- Gargouri, A., Karray, M., Zalila, B., & Ksantini, M. (2025). Design and Development of an Autonomous Mobile Robot for Unstructured Indoor Environments. *Machines*, 13(11), 1044. <https://doi.org/10.3390/machines13111044>
- Ginanjar, O. H. B. (2025). *Benchmarking Algoritma Perencanaan Gerak Berbasis Sampling Pada Robot UR5 dengan Ros Moveit dan Gazebo* [Universitas Islam Indonesia]. <http://dspace.uui.ac.id/123456789/61962>
- Gonzalez-Aguirre, J. A., Osorio-Oliveros, R., Rodríguez-Hernández, K. L., Lizárraga-Iturralde, J., Morales Menendez, R., Ramírez-Mendoza, R. A., Ramírez-Moreno, M. A., & Lozoya-Santos, J. D. J. (2021). Service Robots: Trends and Technology. *Applied Sciences*, 11(22), 10702. <https://doi.org/10.3390/app112210702>
- Jegham, N., Koh, C. Y., Abdelatti, M., & Hendawi, A. (2025). *YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions* (arXiv:2411.00201). arXiv. <https://doi.org/10.48550/arXiv.2411.00201>
- Keith, R., & La, H. M. (2024). *Review of Autonomous Mobile Robots for the Warehouse Environment* (arXiv:2406.08333). arXiv. <https://doi.org/10.48550/arXiv.2406.08333>

- Kemajou, V. N., Bao, A., & Germain, O. (2019). Wellbore Schematics to Structured Data Using Artificial Intelligence Tools. *Offshore Technology Conference*, D011S010R006. <https://doi.org/10.4043/29490-MS>
- Lee, B. Y., Liew, L. H., Cheah, W. S., & Wang, Y. C. (2014). Occlusion handling in videos object tracking: A survey. *IOP Conference Series: Earth and Environmental Science*, 18, 012020. <https://doi.org/10.1088/1755-1315/18/1/012020>
- Li, J., Peng, X., Li, B., Li, M., & Wu, J. (2024). Image-Based Visual Servoing for Three Degree-of-Freedom Robotic Arm with Actuator Faults. *Actuators*, 13(6), 223. <https://doi.org/10.3390/act13060223>
- Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., & Hengel, A. V. D. (2013). A survey of appearance models in visual object tracking. *ACM Transactions on Intelligent Systems and Technology*, 4(4), 1–48. <https://doi.org/10.1145/2508037.2508039>
- Li, Y.-R., Lien, W.-Y., Huang, Z.-H., & Chen, C.-T. (2023). Hybrid Visual Servo Control of a Robotic Manipulator for Cherry Tomato Harvesting. *Actuators*, 12(6), 253. <https://doi.org/10.3390/act12060253>
- Ma, X., Sun, H., Xu, E., Cui, S., Yin, B., & Faied, M. (2020). FSM for Robot Target Search and Retrieval under Semi-constructed Environment. *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, 296–301. <https://doi.org/10.1109/ICMA49215.2020.9233729>
- Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66), eabm6074. <https://doi.org/10.1126/scirobotics.abm6074>
- Macenski, S., & Jambrecic, I. (2021). SLAM Toolbox: SLAM for the dynamic world. *Journal of Open Source Software*, 6(61), 2783. <https://doi.org/10.21105/joss.02783>

- Macenski, S., Martín, F., White, R., & Clavero, J. G. (2020). The Marathon 2: A Navigation System. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2718–2725. <https://doi.org/10.1109/IROS45743.2020.9341207>
- Macenski, S., Moore, T., Lu, D. V., Merzlyakov, A., & Ferguson, M. (2023). From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the robot operating system 2. *Robotics and Autonomous Systems*, 168, 104493. <https://doi.org/10.1016/j.robot.2023.104493>
- Moksyakov, A., Wu, Y., Gadsden, S. A., Yawney, J., & AlShabi, M. (2024). Object Detection and Tracking with YOLO and the Sliding Innovation Filter. *Sensors*, 24(7), 2107. <https://doi.org/10.3390/s24072107>
- Pomares, J. (2019). Visual Servoing in Robotics. *Electronics*, 8(11), 1298. <https://doi.org/10.3390/electronics8111298>
- Putra, H., Fanshuri, R., & Radar, S. S. (2018). Metode Pendeteksi Bola dan Gawang pada Robot Sepak Bola Beroda. *The 6th Indonesian Symposium on Robotics System and Control (ISRSC)*, 114–116.
- Qiao, J., Guo, J., & Li, Y. (2024). Simultaneous localization and mapping (SLAM)-based robot localization and navigation algorithm. *Applied Water Science*, 14(7), 151. <https://doi.org/10.1007/s13201-024-02183-6>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. <https://doi.org/10.1109/CVPR.2016.91>
- Rizaldi, M. I., Rismanto, R., & Ekojono, E. (2017). *IMPLEMENTASI FINITE STATE MACHINE PADA GAME “SAVE THE ANIMALS.”* <https://api.semanticscholar.org/CorpusID:198572249>

- Ruman Talukder, I. H., Talukdar, A., Faisal, S. M. F., Hassan, M., Das, S. C., & Paul, C. (2024). Autonomous Navigation and Object Localization in Unstructured Environments Using SLAM and Path Planning Algorithms on Turtlebot4. *2024 27th International Conference on Computer and Information Technology (ICCIT)*, 1323–1327. <https://doi.org/10.1109/ICCIT64611.2024.11022508>
- Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics*. Springer London. <https://doi.org/10.1007/978-1-84628-642-1>
- Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots, second edition*. MIT Press. <https://books.google.co.id/books?id=4of6AQAAQBAJ>
- Tamadjoe, I. R., & Dewa, C. K. (2024). *PENERAPANAN MODEL DETEKSI OBJEK UNTUK ROBOT MENGGUNAKAN MODEL SSD DI LINGKUNGAN SIMULASI ROS | JATISI*. <https://jurnal.mdp.ac.id/index.php/jatisi/article/view/8679>
- Ultralytics. (2024, September 30). *Ultralytics YOLO11*. <https://docs.ultralytics.com/models/yolo11/>
- Zhu, H., & Luo, Q. (2024). Indoor Localization of Mobile Robots Based on the Fusion of an Improved AMCL Algorithm and a Collision Algorithm. *IEEE Access*, 12, 67199–67208. <https://doi.org/10.1109/ACCESS.2024.3399192>
- Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object Detection in 20 Years: A Survey. *Proceedings of the IEEE*, 111(3), 257–276. <https://doi.org/10.1109/JPROC.2023.3238524>

LAMPIRAN

Lampiran A: Repository Github Project

SanCross354 / autonomous-robot

[Code](#)
[Issues](#)
[Pull requests](#)
[Actions](#)
[Projects](#)
[Wiki](#)
[Security and quality](#)
[Insights](#)
[Settings](#)

autonomous-robot
Public

Pin
 Watch 0
 Fork 0
 Star 0

master had recent pushes 11 minutes ago
 [Compare & pull request](#)

master
 3 Branches
 0 Tags

[Add file](#)
[Code](#)

This branch is 4 commits ahead of and 5 commits behind main
 [Contribute](#)

SanCross354	feat: update all core nodes and configs for final thesis version	8537830 · 11 minutes ago	4 Commits
src	feat: update all core nodes and configs for final thesis version	11 minutes ago	
.gitignore	Fix: Visual servo bugs (tracker trust, target_ratio, min velocity...	4 months ago	
IET Image Processing - 2021 - Tsai - A robust tr...	Fix: Visual servo bugs (tracker trust, target_ratio, min velocity...	4 months ago	
IET Image Processing - 2021 - Tsai - A robust tr...	Fix: Visual servo bugs (tracker trust, target_ratio, min velocity...	4 months ago	
README.md	fix: portable YOLO model path + add full README with setup...	23 minutes ago	
bt_navigator.yaml	Fix: Visual servo bugs (tracker trust, target_ratio, min velocity...	4 months ago	

About

No description, website, or topics provided.

- Readme
- Activity
- 0 stars
- 0 watching
- 0 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Github URL: <https://github.com/SanCross354/autonomous-robot/tree/master>