Implementasi Perencanaan Jalur menggunakan Algoritma Dijkstra pada Robot Roda Mecanum

Senanjung Prayoga[1*], Elmaria Ompu Sunggu[1]
[1]Jurusan Teknik Elektro, Politeknik Negeri Batam, Batam, Indonesia

*Email: senanjung@polibatam.ac.id

***Abstrak*—Dengan kemajuan teknologi, banyak pekerjaan manusia yang berisiko digantikan oleh robot. Robot harus dapat bermanuver secara mandiri tanpa kendali manusia. Penelitian ini mengembangkan robot yang mampu bergerak dari titik awal ke tujuan serta menghindari tabrakan menggunakan algoritma Dijkstra, yang efektif dalam mencari jalur terpendek. Algoritma ini diimplementasikan pada robot mecanum yang dilengkapi dengan mikrokontroler Arduino, mini PC, baterai, motor DC dengan encoder, driver motor, dan perangkat lunak terkait. Penelitian ini menguji lima peta dengan titik awal dan tujuan yang sama, namun dengan letak halangan yang berbeda. Hasil pengujian menunjukkan bahwa robot mecanum berhasil menemukan jalur dan menghindari halangan dengan baik. Dengan kecepatan maksimal rata-rata 2.6 m/s, robot dapat menempuh jarak 3.2 meter tanpa halangan dalam waktu 14.96 detik, serta jarak 4.2 meter dengan halangan dalam waktu 22.82 detik. Penelitian ini menunjukkan potensi algoritma Dijkstra dalam perencanaan jalur dan navigasi robot, serta menggarisbawahi pentingnya penggunaan perangkat keras dan perangkat lunak yang tepat untuk mencapai kinerja optimal. Hasil ini dapat menjadi dasar bagi pengembangan lebih lanjut dalam bidang robotika, khususnya dalam aplikasi robot otonom di lingkungan yang dinamis.**

Kata Kunci : Implementasi Algoritma Dijkstra, Perencanaan Jalur, Robot Mecanum

***Abstract* - This research develops a robot capable of moving from a starting point to a destination as well as avoiding collisions using Dijkstra's algorithm, which is effective in finding the shortest path. This algorithm is implemented on a mecanum robot equipped with an Arduino microcontroller, mini PC, battery, DC motor with encoder, motor driver, and related software. This study tested five maps with the same starting point and destination, but with different obstacle locations. The test results showed that the mecanum robot managed to find the path and avoid obstacles well. With an average maximum speed of 2.6 m/s, the robot can cover a distance of 3.2 meters without obstacles in 14.96 seconds, and a distance of 4.2 meters with obstacles in 22.82 seconds. This research demonstrates the potential of Dijkstra's algorithm in path planning and robot navigation, and underscores the importance of using the right hardware and software to achieve optimal performance. These results can serve as a basis for further development in the field of robotics, particularly in the application of autonomous robots in dynamic environments.**

Keywords: Dijkstra Algorithm Implementation, Path Planning Mecanum Robot

I. PENDAHULUAN

DUNIA bergerak menuju teknologi otomatis dimana robot menggantikan manusia di pekerjaan berisiko tinggi. Kecerdasan buatan dan teknik komputasi lunak memainkan peran penting dalam meniru aktivitas manusia, memungkinkan robot digunakan di berbagai sektor seperti industri, militer, medis, logistik, pertanian, dan transportasi. Pada tahun 2022, terdapat 422000 unit robot di sektor industri, dengan pertumbuhan 12% per tahun. Robot harus mampu bergerak mandiri, menjadikan perencanaan jalur dan penghindaran rintangan sebagai aspek penting dalam robotika. Perencanaan jalur memungkinkan robot mencapai tujuan tanpa bertabrakan dengan rintangan, bahkan dalam lingkungan kompleks, dan berusaha untuk membuat jalur se-optimal mungkin [1].

Untuk memungkinkan robot menggantikan peran manusia secara efektif, navigasi yang baik merupakan hal yang krusial. Perencanaan jalur menjadi aspek kunci dalam penelitian teknologi robot, yang menjamin robot dapat menyelesaikan tugas yang diberikan dengan spesifik [2].

Navigasi *mobile* robot melibatkan empat langkah utama: persepsi, lokalisasi, kognisi dan perencanaan jalur, serta kontrol gerak. Pertama, robot mengumpulkan informasi lingkungan melalui sensor. Kedua, robot menentukan posisi dan orientasinya relatif terhadap lingkungan. Ketiga, robot merencanakan jalur optimal menuju tujuan. Terakhir, robot mengendalikan pergerakannya untuk mengikuti jalur yang telah direncanakan [3].

Perencanaan jalur hanya dapat dilakukan dengan baik jika robot memiliki informasi lengkap tentang peta lingkungan sekitarnya. Oleh karena itu, robot harus memiliki kemampuan untuk melakukan lokalisasi (mengetahui posisinya sendiri dalam peta) dan pemetaan (membuat peta lingkungan sekitarnya) secara simultan. Dengan demikian, robot dapat merencanakan jalur yang tepat dan aman untuk bergerak di lingkungan yang kompleks [4].

Perencanaan jalur meiliki tujuan utama untuk memastikan memilih jalur yang pendek dan waktu perjalanan singkat dari

posisi awal ke target yang telah ditentukan. Selain itu, juga bertujuan untuk menemukan jalur yang aman, optimal, dan bebas rintangan [5].

Salah satu metode yang sering digunakan untuk menyelesaikan masalah perencanaan jalur adalah menggunakan algoritma graf. Beberapa algoritma graf yang populer adalah algoritma Dijkstra dan algoritma Floyd-Warshall. Algoritma Dijkstra adalah salah satu algoritma perencanaan jalur yang pertama kali diusulkan. Algoritma ini cocok untuk menyelesaikan masalah jalur terpendek dari satu titik ke titik lainnya [6].

Algoritma Dijkstra adalah salah satu algoritma yang digunakan dalam teori graf untuk mencari jalur terpendek antara dua simpul dalam sebuah graf yang memiliki bobot positif. Algoritma ini bekerja dengan menghitung bobot terkecil dari simpul awal ke simpul tujuan, sehingga menemukan jalur yang paling efisien berdasarkan bobot yang diberikan [7].

Masalah jalur terpendek mengacu pada proses mencari jalur terpendek dalam sebuah graf dengan cara menghitung jumlah bobot antar simpul. Dalam teori graf, setiap simpul dihubungkan oleh tepi yang memiliki bobot tertentu, dan tujuan dari masalah ini adalah menemukan jalur yang memiliki jumlah bobot terkecil dari titik awal ke titik tujuan [8].

Fitur utama dari algoritma ini adalah memulai pencarian dari titik awal, kemudian memperluas pencarian secara bertahap ke lapisan luar. Algoritma ini menemukan titik berikutnya yang paling dekat dengan target dan terus melakukan iterasi lapis demi lapis hingga mencapai titik target. Hal ini bertujuan untuk meningkatkan efisiensi dan akurasi dalam perencanaan jalur [9].

Pada peneltian ini akan diimplementasikan algoritma perencanaan jalur Dijkstra pada mobile robot dengan 4 roda mecanum. Robot ini mempunya kemampuan untuk bisa bergerak segala arah, memungkinkan penggunaan di industri, kedokteran, dan kompetisi robot [10].

Berdasarkan hasil penelitian dan pengujian di beberapa lingkungan, algoritma Dijkstra terbukti lebih efisien dan praktis. Algoritma ini dapat memberikan rute terpendek dengan cepat pada peta yang sudah diketahui letak rintangan dan rutenya, yang datanya diperoleh dari SLAM. Penelitian ini bertujuan untuk membuktikan bahwa algoritma ini dapat memecahkan masalah perencanaan jalur, sehingga robot mecanum dapat berjalan dengan baik dan menghindari rintangan saat bergerak.

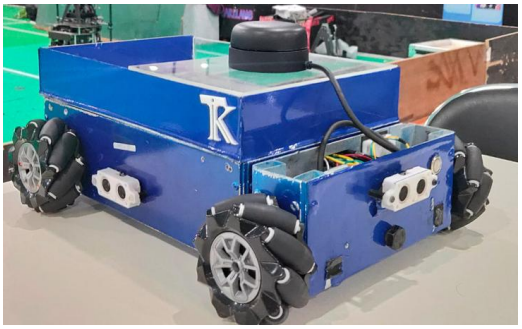
Jurnal ini disusun sebagai berikut: bagian II menyajikan landasan teori, komponen utama robot, *pseudocode* algoritma Dijkstra beserta pengertian dari algoritma tersebut, bagian III menyajikan data analisis dan hasil dari implementasi algoritma Dijkstra pada robot dan bagian IV menyajikan kesimpulan dari isi seluruh hasil penelitian pada jurnal ini beserta referensi yang digunakan.

II. METODE PENELITIAN

Penelitian ini bertujuan membuat perencanaan jalur yang efisien, cepat, terpendek, dan mampu menghindari halangan, sehingga robot dapat bernavigasi dengan menggunakan ROS dan mikrokontroler Arduino Atmega 2560 serta komunikasi rosserial. Hasil penelitian ini adalah robot yang bergerak mengikuti jalur dari titik awal ke tujuan berdasarkan data input.

A. *Design Robot*

Robot mecanum dalam penelitian ini menggunakan 4 roda mecanum dan dikendalikan oleh mikrokontroller Arduino Atmega 2560. Motor penggeraknya adalah motor DC 12V dengan encoder, dan driver motor L298N digunakan untuk menggerakkan robot. Robot ini memiliki kecepatan 2.6 meter per detik. Ukuran robotnya adalah 379 mm x 210 mm x 90 mm, dengan diameter roda 97 mm dan berat 3.2 kg. Untuk design robot roda mecanum ada pada gambar 1.



Gambar. 1. Design robot mecanum

B. *Perangkat Keras Robot*

Perangkat keras yang digunakan untuk penelitian perencanaan jalur terhadap robot terdapat pada Tabel 1.

TABEL 1 Daftar Perangkat Keras

Jenis	Jumlah
Arduino Atmega 2560	1
Motor driver L298N	2
Motor Dc dengan encoder	4
RPLidar A2M12	1
Mini PC intel NUC	1
Roda Mecanum	4
Baterai Li Po 12V	3
Step down DC	1

Robot ini menggunakan kontroler Arduino Atmega 2560 untuk mengendalikan driver motor L298N yang mengontrol kecepatan dan arah motor DC motor. Arduino ini mempunyai 54 pin digital input/output, termasuk didalamnya adalam 14 pin PWM output dan 16 pin analog input. Robot ini juga menggunakan mini-PC untuk memproses data sensor LIDAR.

Motor DC pada robot ini dilengkapi dengan encoder sebagai sebagai umpan balik untuk mendeteksi kecepatan motor. Encoder adalah perangkat yang mengubah gerakan rotasi menjadi sinyal listrik untuk menentukan posisi atau kecepatan

putaran motor. Encoder ini menggunakan piringan berlubang, LED dan sensor photo-transistor untuk mendeteksi pulsa cahaya yang melewati lubang saat piringan berputar. Kecepatan motor diukur dengan menghitung jumlah pulsa yang dihasilkan dalam waktu tertentu [11].

RPLidar A2M12 adalah sensor lidar 360^o yang dikembangkan oleh SLAMTEC, dengan radius pindai maksimum 12 meter. Sensor ini memancarkan cahaya laser ke objek, lalu menangkap dan menganalisis pantulannya untuk mendeteksi objek. LIDAR menggunakan dua metode pengukuran jarak: triangulasi, yang menggunakan prinsip trigonometri, dan *time of flight* (ToF), yang mengukur waktu tempuh cahaya. RPLIDAR digunakan pada robot otonom untuk menghindari tabrakan dan mendeteksi rintangan dalam ruangan [12].

Roda mecanum sebagai alat gerak robot, roda mecanum adalah suatu sistem yang dirancang untuk robot *autonomous*, sistem ini diharapkan dapat berfungsi untuk menggerakkan robot dengan fleksibilitas, kelebihan dari roda mecanum yaitu dapat bergerak secara serentak dan mandiri arah rotasi dan translasi artinya keempatnya dapat bergerak setiap saat ke segala arah tanpa orientasi. Baterai digunakan sebagai daya robot dan yang terakhir yaitu *step down* atau penurun tegangan yang digunakan untuk menurunkan tegangan dari 24 V ke 19 V yang akan disalurkan ke mini PC sebagai power mini PC [13].

C. Perangkat Lunak Robot

Perangkat lunak robot yang digunakan pada penelitian ini dapat dilihat pada Tabel 2.

TABEL 2 Perangkat Lunak Robot

Nama software	Versi
Arduino IDE	versi 2.2.1
ROS Noetic	versi ubuntu 20.04
Visual Studio Code	versi 1.84.2
Rviz	versi 1.14.20

Arduino IDE di penelitian ini digunakan untuk mengontrol motor mulai dari arah putaran, kecepatan dan odometry robot. *robot operating system* (ROS) adalah kumpulan perpustakaan perangkat lunak dan alat yang menyediakan kerangka kerja untuk mengembangkan dan menjalankan aplikasi robotika. ROS bersifat *open source* dan modular, memungkinkan penggunaan, modifikasi, dan kontribusi dari siapapun. Pengguna dapat memilih komponen dan paket sesuai kebutuhan proyek. ROS menggunakan konsep node, yaitu proses yang melakukan tugas tertentu, seperti mengendalikan sensor atau kamera. Node berkomunikasi melalui topik, yang membawa pesan jenis tertentu, seperti data sensor, perintah, atau gambar [14].

Dipenelitian ini menggunakan ROS jenis Noetic sebagai *middleware* komunikasi antar program. RViz adalah antarmuka grafis ROS untuk memvisualisasikan informasi dari berbagai topik. Dalam penelitian ini, RViz digunakan untuk visualisasi perencanaan jalur. Visual Studio Code, editor buatan Microsoft yang kompatibel dengan Windows, macOS, dan Linux, digunakan untuk mengedit dan menyimpan kode C++ robot [15].

D. Perencanaan Jalur Metode Algoritma Dijkstra

Perencanaan jalur adalah aspek penting dalam penelitian pengendalian gerak robot. Tujuan utamanya adalah memastikan bahwa robot dapat bergerak melalui area yang kompleks tanpa mengalami tabrakan, meskipun terdapat berbagai rintangan atau kondisi jalan yang tidak stabil [16].

Metode perencanaan jalur terbagi menjadi dua kategori utama: metode klasik dan metode heuristik. Metode klasik menggunakan pendekatan matematika dan geometris untuk menemukan jalur optimal dari titik awal ke titik tujuan. Metode ini mencakup *probabilistic road map* (PRM), *rapidly random tree* (RRT), *cell decomposition* (CD) dan *potential field* (PF). Metode heuristik menggunakan pendekatan berbasis aturan atau pengalaman untuk menemukan solusi yang baik, meskipun mungkin tidak optimal. Metode ini lebih cepat dan lebih efisien untuk lingkungan yang kompleks atau dinamis. Metode ini meliputi *algorithm genetic* (GA), Algoritma Dijkstra, D*, *artificial neural network* (ANN), A* dan *particle swarm optimization* (PSO). Pendekatan ini digunakan dalam perencanaan jalur robot mobile karena kelengkapan, efisiensi, dan optimalitasnya. Penelitian ini menggunakan Algoritma Dijkstra, yang dikenal sebagai salah satu algoritma terbaik untuk mencari jalur terpendek antara dua titik sejak diperkenalkan pada tahun 1968 [3].

Algoritma ini andal dan tahan terhadap gangguan, namun memiliki kompleksitas tinggi dan rentan terhadap optimum lokal [17].

Perencanaan jalur terbagi menjadi dua: global untuk lingkungan yang dikenal dan lokal untuk lingkungan yang tidak diketahui. Penelitian ini menggunakan perencanaan jalur global menggunakan Dijkstra untuk mencari jalur terdekat dan lokal memakai *time elastic band* (TED) untuk mengontrol *velocity* robot dan hindari rintangan untuk mencapai tujuan yang ditentukan [18].

Dalam pencarian jalur terpendek, algoritma Dijkstra mencari bobot minimal dari graf berbobot untuk mendapatkan jarak terpendek antara dua titik. Algoritma dimulai dari node sumber dan mengiterasi untuk menemukan node dengan bobot terkecil. Titik yang dipilih dipisahkan dan tidak diperhatikan lagi.

Langkah-langkah Dijkstra adalah:

1) Inisialisasi peta dengan node sumber bernilai 0 dan node tujuan ∞.
2) Telusuri node tetangga dari node sumber dan perbarui jarak jika lebih kecil.
3) Beri label node yang sudah diperiksa dan abaikan di iterasi berikutnya.
4) Pilih node dengan jarak terkecil yang belum dilewati sebagai node sumber baru.
5) Ulangi langkah sampai mencapai node tujuan.
6) Algoritma selesai dan menghasilkan jalur terpendek.

Pengujian dilakukan untuk mendapatkan kesesuaian jarak dari titik awal sampai titik akhir pada peta yang sudah ditentukan jaraknya oleh algoritma Dijkstra. Berikut dibawah ini adalah pseudocode dari algoritma Dijkstra.

```
1: Function Dijkstra (S, source)
2:   dist = map ()
3:   Q = set ()
4:   for i in S:
5:       if ( i = source) :
6:           dist[i] := 0
7:       else:
8:           dist[i] :=infinity
9:           add i to Q
10:  while Q is not empty:
11:     i :=i in Q with minimum dist [i]
12:     remove i from Q
13:     for each neighbor u for i and u in Q:
14:         newDist := dist[i] + weight(i,u)
15:         if  (newDist < dist[u]):
16:             dist[u] := newDist
17:  return dist[]
```

Gambar. 2. Algoritma Djikstra

Berikut ini penjelasan pada gambar 2 diatas tentang algoritma Dijkstra yang digunakan untuk menemukan jalur terpendek dari satu titik sumber ke semua titik lain di graf berbobot. Proses dalam pencarian jalur terpendek dibagi menjadi 3 proses yaitu proses inisialisasi, proses algoritma dan yang terakhir penyelesaian. Langkah pertama yaitu inisialisasi: dist adalah peta yang menyimpan jarak minimum dari sumber ke tiap simpul. Di awal jarak ini akan diinisialisasi, Q adalah himpunan kosong yang menyimpan semua node yang belum diproses (node yang belum ditemukan jarak terpendeknya). Untuk setiap pilih node i dihimpunan s (daftar node):

- Jika node i adalah node sumber, maka jaraknya diatur menjadi 0 (dist[i] = 0).
- Jika bukan node sumber, maka jaraknya diatur ke tak hingga (∞) (dist [i] = infinity ∞).
- Setiap node ditambahkan ke dalam himpunan Q (node yang belum diproses).

Langkah kedua yaitu proses algoritma yaitu selama Q tidak kosong, algoritma akan terus mencari node dengan jarak minimum yang belum diproses. Pilih node i di Q yang memiliki jarak terpendek (dist[i] paling kecil). Kemudian hapus node i dari Q (karena jarak terpendeknya sudah diketahui) dan tidak perlu diproses lagi node yang jaraknya lebih besar dari himpunan Q. Dan untuk setiap tetangga node i (node yang paling kecil) yang sudah disimpan akan dibandingkan lagi apakah jarak melalui i lebih pendek dibandingkan jarak yang sudah disimpan, jika iya maka update jarak node tetangga (dist [u]) menjadi node yang paling kecil dari node sebelumnya dan disimpan menjadi node paling kecil kemudian node sebelumnya yang sudah disimpan akan dihapus karena sudah ada node yang paling kecil dan proses ini diulangi hingga semua node diproses, memastikan bahwa jarak terpendek dari sumber ke node lainnya ditemukan. Langkah ketiga yaitu penyelesaian yaitu algoritma akan berhenti ketika semua node telah diproses dari titik sumber ke titik tujuan dan jarak terpendek dari node sumber ke node lainnya hingga mencapai titik tujuan ditemukan.

E. TEB (Time Elastic Band)

TEB adalah pendekatan yang mengoptimalkan jalur robot dalam dimensi ruang dan waktu secara bersamaan dengan memodelkan jalur sebagai *band elastic*. Pendekatan ini menganggap jalur sebagai kumpulan segmen yang bisa diregangkan atau dikompresi untuk meminimalkan biaya perjalanan sambil tetap memenuhi berbagai kendala [19]. TEB dan Algoritma Dijkstra adalah dua pendekatan yang dapat digabungkan untuk memberikan solusi optimal dalam perencanaan jalur dengan menghindari rintangan secara efisien. Berikut konsep kerja dari TEB dalam menghindari rintangan dengan bantuan algoritma Dijkstra:

1) Penggunaan Dijkstra untuk jalur awal
   a) Algoritma dijkstra digunakan untuk menghitung jalur awal dilingkungan berdasarkan graf
   b) Graf mewakili area navigasi dengan node sebagai titik posisi dan edge sebagai koneksi antar node dengan bobot yang merepresentasikan jarak atau biaya
   c) Dijkstra akan menghasilkan jalur terpendek di graf yang menghindari node dengan rintangan
2) Integrasi dengan TEB
   a) Jalur awal dari dijkstra dimasukkan ke dalam TEB sebagai jalur awal untuk optimasi lebih lanjut
   b) TEB kemudian mengubah jalur ini menjadi jalur elastik yang dapat diregangkan atau di kompresi

3) Optimasi jalur oleh TEB
a) Penghindaran rintangan: jalur elastik secara iteratif disesuaikan untuk menghindari rintangan sambil tetap berada di dekat jalur awal.
b) Kendala temporal: TEB menambahkan dimensi waktu untuk memastikan jalur memenuhi batas kecepatan dan akselerasi robot
c) Efisiensi jalur: TEB mengoptimalkan panjang jalur, waktu perjalanan dan jarak aman dari rintangan

4) Perbaruan Dinamis
a) Jika lingkungan berubah (muncul rintangan baru), algoritma Dijkstra dapat digunakan untuk kembali untuk menghitung jalur awal baru
b) TEB kemudian melakukan optimasi ulang alur elastik dengan memperhatikan perubahan.

Kelebihan menggunakan kombinasi TEB dan Dijkstra yaitu dijkstra dapat memberikan jalur awal yang cepat dan TEB menyempurnakan jalur untuk efisiensi lebih lanjut. Dalam menghindari rintangan baik karena menggunakan graf (dijkstra) dan jalur elastik TEB dan dapat menangani lingkungan dinamis dan statis.

F. Kinematika Mecanum

Penurunan kinematika robot mecanum adalah langkah awal dalam merancang trajectory, yaitu lintasan optimal yang harus diikuti robot untuk mencapai tujuannya dari posisi awal. Penurunan kinematika ini didasarkan pada konfigurasi roda pada robot mekanum yang bisa dilihat pada gambar 3, gambar tersebut menunjukkan skema robot dengan sistem roda mekanum yang memungkinkan pergerakan robot secara bebas ke segala arah (maju, mundur, kesamping, diagonal atau berputar) tanpa perlu memutar arah fisik robot secara keseluruhan. Kombinasi sudut roda dan vektor gaya pada gambar menunjukkan bagaimana setiap roda berkontribusi pada pergerakan *omnidirectional* robot.

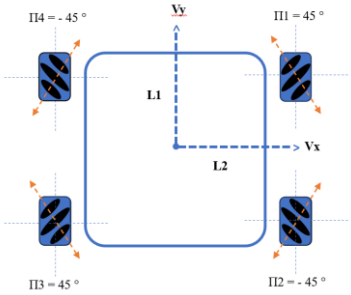
Setiap roda memiliki roller yang disusun miring dengan sudut *Π* dan berdasarkan pada gambar 3, dua roda (Π_1 dan Π_3) memiliki sudut +45° terhadap sumbu horizontal sedangkan dua roda (Π_2 dan Π_4) memiliki sudut -45°. Sudut ini menentukan bagaimana setiap roda memberikan dorongan atau vektor gerakan pada sumbu X dan sumbu Y. Sumbu koordinat Vx yaitu menggambarkan kecepatan atau pergerakan dalam sumbu x (horizontal) dan Vy yaitu menggambarkan kecepatan atau pergerakan dalam sumbu Y (*vertical*). Kombinasi dari pergerakan ini memungkinkan robot untuk bergerak keberbagai arah, tidak hanya maju-mundur atau kiri-kanan tetapi juga diagonal atau rotasi tanpa memutar seluruh badan robot.

L1 dan L2 adalah dimensi dari robot, menunjukkan panjang dan lebar antara roda-roda di keempat sudut robot. Jarak antar roda ini mempengaruhi stabilitas dan kemampuan manuver robot. Dan tanpa panah orange pada setiap roda menunjukkan arah vektor gaya dorong yang dihasilkan oleh masing-masing roda, karena rod aini memiliki mekanisme khusus, roda bisa bergerak kearah yang berbeda dari sumbu normal putaran roda. Dan setiap roda bekerja sama untuk menghasilkan gaya gabungan yang memungkinkan robot bergerak ke berbagai arah.

Penelitian ini menggunakan persamaan (1) kinematika terbalik (*inverse*) dan persamaan (2) kinematika maju (*forward*) untuk mendukung perencanaan jalur dengan algoritma Dijkstra pada robot mecanum.

Persamaan (1) inverse kinematik digunakan untuk menentukan kecepatan motor roda yang diperlukan untuk mencapai posisi atau orientasi tertentu ($\dot{\theta}_{FL}$, $\dot{\theta}_{FR}$, $\dot{\theta}_{BL}$, $\dot{\theta}_{BR}$) berdasarkan kecepatan linear (Vx, Vy) dan kecepatan angular (ω) yang diinginkan. Persamaan (2) kinematika maju digunakan untuk menentukan posisi dan orientasi robot berdasarkan kecepatan motor dan konfigurasi roda. Dengan mengetahui kecepatan roda, dapat dihitung kecepatan robot ke arah depan (Vx), ke samping (Vy), dan rotasional (ω).

Untuk bergerak maju, keempat roda harus bergerak ke depan dengan kecepatan yang sama. Untuk bergerak ke kiri, roda 1 dan 4 bergerak ke belakang, sedangkan roda 2 dan 3 berhenti. Untuk berputar searah jarum jam, roda 1 dan 3 maju, sementara roda 2 dan 4 mundur. Jika roda 1 dan 3 bergerak lebih cepat daripada roda 2 dan 4, robot akan bergerak membentuk lingkaran searah jarum jam [20].



Gambar. 3. Konfigurasi dan Resultan Gaya Robot Mecanum.

$$\begin{bmatrix}\dot{\theta}_{FL}\\ \dot{\theta}_{FR}\\ \dot{\theta}_{BL}\\ \dot{\theta}_{BR}\end{bmatrix} = \frac{1}{2\pi r}\begin{bmatrix}1 & -1 & -(d+e)/2\\ 1 & 1 & (d+e)/2\\ 1 & 1 & -(d+e)/2\\ 1 & -1 & (d+e)/2\end{bmatrix}\begin{bmatrix}V_x\\ V_y\\ \omega\end{bmatrix} \quad (1)$$

$$\begin{bmatrix}V_x\\ V_y\\ \omega\end{bmatrix} = 2\pi r\begin{bmatrix}\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}\\ -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4}\\ -\frac{1}{2(d+e)} & \frac{1}{2(d+e)} & -\frac{1}{2(d+e)} & \frac{1}{2(d+e)}\end{bmatrix} \quad (2)$$

Keterangan:
$\dot{\theta}_{FL}$: Kecepatan roda depan-kiri atau front-left (rps)
$\dot{\theta}_{FR}$: Kecepatan roda depan-kanan atau front-right (rps)
$\dot{\theta}_{BL}$: Kecepatan roda belakang-kiri atau Back-Left (rps)
$\dot{\theta}_{BR}$: kecepatan roda belakang-kanan atau Back-Right (rps)
r : Radius roda (m)
d : Jarak roda kiri ke roda kanan (m)
e : Jarak roda depan ke roda belakang (m)
Vx : Kecepatan robot dalam arah maju (m/s)
Vy : kecepatan robot dalam arah samping (m/s)
ω : Kecepatan rotasional robot (rad/s)

III. HASIL DAN ANALISA

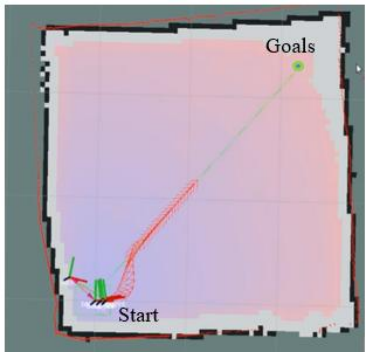
Gambar. 4. Lapangan Uji

Dalam penelitian ini, algoritma Dijkstra diterapkan untuk merencanakan jalur terpendek pada lapangan uji (gambar 4) berukuran 3.3 m x 3.3 m. Lapangan uji tersebut berisi beberapa rintangan yang ditempatkan secara acak untuk mensimulasikan kondisi nyata. Pengujian dilakukan pada 5 kondisi dengan posisi halangan yang berbeda, namun dengan titik awal yang sama (x = 0, y = 0) dan tujuan yang sama (x = 2.4, y = 2.3). Langkah pengujian meliputi:

1) Inisialisasi lingkungan dengan menyiapkan peta grid dan rintangan.
2) Inisialisasi algoritma dengan menentukan titik awal dan tujuan, lalu menjalankan algoritma untuk menentukan jalur terpendek.
3) Mengukur panjang jalur, waktu tempuh, dan mencatat keberhasilan dalam menghindari rintangan.
4) Menganalisis data yang diperoleh.
5) Memvisualisasikan hasil jalur.

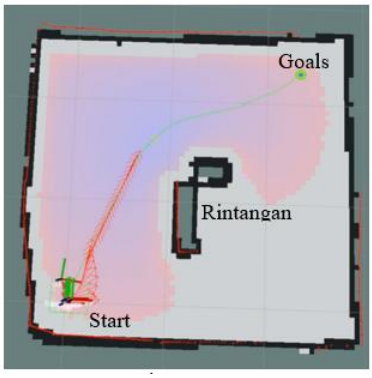
A. Pengujian Pertama Peta 1

Pada gambar 5, hasil pengujian robot mecanum di Peta 1 tanpa rintangan menunjukkan bahwa robot menempuh jarak 3.21 m dalam 14.96 detik dengan kecepatan rata-rata 0.2 m/s. Kecepatan ini dihitung dari pembagian jarak total dengan waktu total. Hasil ini mencerminkan kemampuan robot untuk menavigasi lingkungan sederhana dengan efisiensi dan stabilitas yang baik, serta dapat digunakan sebagai *baseline* untuk membandingkan performa robot dikondisi yang lebih menantang.



Gambar. 5. Peta 1.

B. Pengujian Kedua Peta 2



Gambar. 6. Peta 2

Pada pengujian kedua di peta 2 (Gambar 6), yang memiliki rintangan tambahan, robot mecanum menempuh jarak 3.38 meter dalam 16.08 detik dengan kecepatan rata-rata 0.2 m/s. Meskipun jarak dan waktu tempuh lebih panjang dibandingkan Peta 1, robot tetap mencapai tujuan dengan kecepatan konstan. Hasil ini menunjukkan kemampuan robot untuk beradaptasi dengan rintangan di lingkungan yang lebih kompleks.

C. Pengujian ketiga Peta 3

Pada pengujian ketiga (Gambar 7) di peta 3, yang memiliki lebih banyak rintangan, robot mecanum menempuh jarak 4.14 meter dalam 22.13 sekon dengan kecepatan rata-rata 0.2 m/s. Jarak dan waktu tempuh lebih panjang dibandingkan peta 1 dan 2, menunjukkan kompleksitas rintangan mempengaruhi jalur yang diambil robot. Hasil ini menunjukkan bahwa algoritma *path planning* berhasil mengarahkan robot melalui jalur yang aman meskipun lebih panjang, dan robot tetap dapat beradaptasi dengan lingkungan yang lebih kompleks sambil mempertahankan kecepatan yang konsisten.

Gambar. 7. Peta 3

D. Pengujian Keempat Peta 4

Gambar. 8. Peta 4.

Pada pengujian keempat (Gambar 8) di peta 4, robot mecanum menempuh jarak 4.07 meter dalam 19.12 detik dengan kecepatan rata-rata 0.2 m/s. Meskipun jarak hampir sama dengan peta 3, waktu tempuh lebih singkat, menunjukkan rintangan di peta 4 lebih mudah dihindari. Hasil ini menunjukkan kemampuan robot untuk beradaptasi dengan konfigurasi rintangan yang berbeda sambil mempertahankan kecepatan yang konsisten.

E. Pengujian kelima Peta 5

Pada pengujian kelima (Gambar 9) di peta 5, robot mecanum menempuh jarak 4.29 meter dalam 22.85 detik dengan kecepatan rata-rata 0.2 m/s. Meskipun posisi awal dan tujuan sama dengan pengujian sebelumnya, rintangan diatur berbeda, menambah tantangan. Jarak dan waktu tempuh lebih panjang dibandingkan peta 4, menunjukkan kemampuan robot untuk beradaptasi dengan rintangan baru dan mempertahankan kecepatan yang konsisten, serta efektivitas algoritma *path planning* dalam menemukan jalur optimal. Berikut ini hasil ringkasan dari pengujian-pengujian yang dilakukan dengan menggunakan lima data peta dapat dilihat pada Tabel 3.

Gambar. 9. di Peta 5

Dari tabel 3, pada peta 1 tanpa rintangan, robot menempuh jarak 3.21m dalam waktu 14.96 detik. Posisi akhir robot sedikit berbeda dari target dengan error 0.07 m pada sumbu x dan 0.05 m pada sumbu y. Kecepatan rata-rata adalah 0.21 m/s. Pada peta 2 dengan rintangan tambahan, robot menempuh jarak 3.38 m dalam 16.08 detik. Error terhadap posisi akhir adalah 0.08 m pada sumbu x dan 0.05 m pada sumbu y. Kecepatan rata-rata tetap 0.21 m/s. Pada peta 3 dengan lebih banyak rintangan, robot menempuh jarak 4.14 m dalam 22.13 detik. Error terhadap posisi akhir adalah 0.07 m pada sumbu x dan 0.06 m pada sumbu y. Kecepatan rata-rata menurun menjadi 0.18 m/s. Pada peta 4, robot menempuh jarak 4.07 m dalam 19.12 detik. Error terhadap posisi akhir adalah 0.08 m pada sumbu x dan 0.06 m pada sumbu y. Kecepatan rata-rata kembali menjadi 0.21 m/s dan yang terakhir pada peta 5 dengan rintangan baru, robot menempuh jarak 4.20 m dalam 22.85 detik. Error terhadap posisi akhir adalah 0.05 m pada sumbu x dan 0.04 m pada sumbu y. Kecepatan rata-rata adalah 0.18 m/s.

TABLE 3
HASIL PENGUJIAN TERHADAP PETA 1-5

Keterangan	Peta 1	Peta 2	Peta 3	Peta 4	Peta 5
Waktu Eksekusi (s)	14.96	16.08	22.13	19.12	22.85
Jarak (m)	3.21	3.38	4.14	4.07	4.2
Posisi X (m)	2.33	2.32	2.33	2.32	2.35
Posisi Y (m)	2.35	2.35	2.36	2.36	2.34
Error posisi X	0.07	0.08	0.07	0.08	0.05
Error posisi Y	0.05	0.05	0.06	0.06	0.04
Rata- rata vel. (m/s)	0.21	0.21	0.21	0.21	0.18
Rata – Rata kecepatan Robot					0.19

IV. KESIMPULAN

Penelitian ini meneliti efektivitas algoritma Dijkstra dalam perencanaan jalur untuk robot mecanum. Hasil penelitian menunjukkan bahwa algoritma Dijkstra sangat efektif dalam menentukan jalur terpendek dari titik awal ke tujuan. Robot yang menggunakan algoritma ini mampu mencapai kecepatan rata-rata 0.19 m/s, dengan jarak tempuh terpendek sebesar 3.2 meter dalam waktu 14.96 detik, dan jarak terjauh sebesar 4.2 meter dalam waktu 22.85 detik. Kecepatan rata-rata robot dalam semua uji adalah 0.18 m/s. Selain itu, error posisi rata-rata robot adalah 0.6% pada sumbu X dan 0.5% pada sumbu Y yang menunjukkan tingkat akurasi yang tinggi. Algoritma Dijkstra juga terbukti mampu mempertahankan kinerja yang stabil meskipun menghadapi rintangan tambahan di sepanjang jalur. Hal ini menunjukkan kemampuan algoritma dalam beradaptasi dalam berbagai kondisi lingkungan yang berbeda-beda.

REFERENCES

[1] C. Wang and J. Mao, "Summary of AGV Path Planning," in *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, IEEE, Oct. 2019, pp. 332–335. doi: 10.1109/EITCE47263.2019.9094825.

[2] Y. Li, Z. Huang, and Y. Xie, "Path planning of mobile robot based on improved genetic algorithm," *Proc. - 2020 3rd Int. Conf. Electron Device Mech. Eng. ICEDME 2020*, pp. 691–695, 2020, doi: 10.1109/ICEDME50972.2020.00163.

[3] M. N. A. Wahab, C. M. Lee, M. F. Akbar, and F. H. Hassan, "Path Planning for Mobile Robot Navigation in Unknown Indoor Environments Using Hybrid PSOFS Algorithm," *IEEE Access*, vol. 8, pp. 161805–161815, 2020, doi: 10.1109/ACCESS.2020.3021605.

[4] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid Path Planning Algorithm Based on Membrane Pseudo-Bacterial Potential Field for Autonomous Mobile Robots," *IEEE Access*, vol. 7, pp. 156787–156803, 2019, doi: 10.1109/ACCESS.2019.2949835.

[5] H. Zhao, Z. Nie, F. Zhou, and S. Lu, "A Compound Path Planning Algorithm for Mobile Robots," *Proc. 2021 IEEE Int. Conf. Power Electron. Comput. Appl. ICPECA 2021*, pp. 541–545, 2021, doi: 10.1109/ICPECA51329.2021.9362724.

[6] Y. Jiang *et al.*, "Path Planning for Mobile Robots Based on Improved RRT Algorithm," *ICARM 2022 - 2022 7th IEEE Int. Conf. Adv. Robot. Mechatronics*, pp. 793–798, 2022, doi: 10.1109/ICARM54641.2022.9959538.

[7] W. Andriani, "Perbandingan Algoritma Dijkstra dan Algoritma Floyd-Warshall Penentuan Jalur Lintasan Terpendek Stasiun Tegal Menuju Hotel," *BATIRSI-Bahari Tek. Inform. dan …*, vol. 4, no. 2, pp. 1–8, 2021, [Online]. Available: https://e-journal.stmik-tegal.ac.id/index.php/batirsi/article/view/42

[8] A. Alyasin, E. I. Abbas, and S. D. Hasan, "An Efficient Optimal Path Finding for Mobile Robot Based on Dijkstra Method," *4th Sci. Int. Conf. Najaf, SICN 2019*, pp. 11–14, 2019, doi: 10.1109/SICN47020.2019.9019345.

[9] Q. Liu, H. Xu, L. Wang, J. Chen, Y. Li, and L. Xu, "Application of Dijkstra algorithm in path planning for geomagnetic navigation," *Proc. IEEE Sens. Array Multichannel Signal Process. Work.*, vol. 2020-June, pp. 19–22, 2020, doi: 10.1109/SAM48682.2020.9104382.

[10] Y. Xu, Y. Zhao, and S. L. Zhao, "Grid Line Tracking Omnidirectional Robot Based on Visible Light Sensor and Mecanum Wheel PID Control," *Proc. - 2021 7th Int. Symp. Mechatronics Ind. Informatics, ISMII 2021*, pp. 57–60, 2021, doi: 10.1109/ISMII52409.2021.00019.

[11] G. Rahmatillah and B. Suprianto, "Sistem Pengendalian Kecepatan Motor DC Pada Prototipe Lift Menggunakan Kontroler Pi Berbasis Arduino," *J. Tek. Elektro*, vol. 9, no. 2, pp. 269–276, 2020.

[12] M. Fikri and M. Rivai, "Sistem Penghindar Halangan dengan Metode Lidar pada Unmanned Surface Vehicle," *J. Tek. ITS*, vol. 8, no. 2, pp. 127–132, 2020, doi: 10.12962/j23373539.v8i2.43153.

[13] E. C. Orozco-Magdaleno, F. Gomez-Bravo, E. Castillo-Castaneda, and G. Carbone, "Evaluation of Locomotion Performances for a Mecanum-Wheeled Hybrid Hexapod Robot," *IEEE/ASME Trans. Mechatronics*, vol. 26, no. 3, pp. 1657–1667, 2021, doi: 10.1109/TMECH.2020.3027259.

[14] L. Qiong, C. Xudong, H. Jizhuang, and M. Ruihao, "Research on robot path planning method based on tangent intersection method," *Proc. - 2020 Int. Conf. Inf. Sci. Parallel Distrib. Syst. ISPDS 2020*, pp. 272–276, 2020, doi: 10.1109/ISPDS51347.2020.00063.

[15] H. Huang *et al.*, "Dynamic path planning based on improved D∗ algorithms of Gaode map," *Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019*, no. Itnec, pp. 1121–1124, 2019, doi: 10.1109/ITNEC.2019.8729438.

[16] Y. Ding, H. Ma, and S. Li, "Path Planning of Omnidirectional Mobile Vehicle Based on Road Condition," *Proc. 2019 IEEE Int. Conf. Mechatronics Autom. ICMA 2019*, pp. 1425–1429, 2019, doi: 10.1109/ICMA.2019.8816402.

[17] Z. Nie and H. Zhao, "Research on Robot Path Planning Based on Dijkstra and Ant Colony Optimization," in *2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, IEEE, Nov. 2019, pp. 222–226. doi: 10.1109/ICIIBMS46890.2019.8991502.

[18] H. Liu, L. Huang, and H. Ye, "Autonomous path planning strategy for water-Air amphibious vehicle based on improved A* algorithm," *IEEE Int. Conf. Ind. Informatics*, vol. 2020-July, pp. 812–816, 2020, doi: 10.1109/INDIN45582.2020.9442090.

[19] Wu J, Ma X, Peng T, Wang H. An Improved Timed Elastic Band (TEB) Algorithm of Autonomous Ground Vehicle (AGV) in Complex Environment. Sensors. 2021; 21(24):8312. https://doi.org/10.3390/s21248312

[20] F. Fahmizal, A. Priyatmoko, and A. Mayub, "Implementasi Kinematika Trajectory Lingkaran pada Robot Roda Mecanum," *J. List. Instrumentasi dan Elektron. Terap.*, vol. 3, no. 1, pp. 25–30, 2022, doi: 10.22146/juliet.v3i1.74760.