Implementasi Algoritma *Path Planning A** Pada *Base Station* Robot Sepak Bola Beroda

Luluk Indah Safitri[1], Julian Sahertian[2], Danang Wahyu Widodo[3]
[1,2,3]Teknik Informatika, Fakultas Teknik, Universitas Nusantara PGRI Kediri
E-mail: [1]*luluk.safitri89@gmail.com*, [2]*juliansahertian@unpkediri.ac.id*,
[3]*danangwahyuwidodo@unpkediri.ac.id*
Coresponden Author: *juliansahertian@unpkediri.ac.id*

Diterima Redaksi: 05 Juli 2023 Revisi Akhir: 24 Oktober 2023 Diterbitkan Online: 02 November 2023

Abstrak – *Kontes Robot Sepak Bola Indonesia Beroda (KRSBIB) merupakan kompetisi robotika yang melibatkan keterampilan dalam penyusunan strategi pada robot, sistem navigasi robot, serta perencanaan jalur pada robot. Penelitian ini membahas implementasi algoritma A* pada base station tim robotik Abimanyu Universitas Nusantara PGRI Kediri sebagai algoritma perencanaan jalur, path planning diterapkan secara simulasi tanpa melibatkan pengiriman data ke robot. Pada penerapan algoritma, lapangan terlebih dahulu dibagi menjadi beberapa grid yang merepresentasikan node yang dapat dilalui oleh setiap algoritma. Penelitian yang dilakukan mendapatkan hasil bahwa algoritma A* mencapai waktu eksekusi yang lebih cepat pada map dengan ukuran grid 32px dengan kondisi apapun, pada map Empty 32px mendapatkan waktu eksekusi 0.091 s lebih cepat dibandingkan map Empty 8px dengan panjang path 0.1 cm lebih pendek, pada map Simple 32px mendapatkan waktu eksekusi 0.542 s lebih cepat dibanding map simple 8px dengan panjang path 0.2 cm lebih pendek, dan selisih paling besar yakni pada tipe map Surround Target 32px yang memiliki waktu eksekusi 1.653 s lebih cepat dan panjang path 0.5 cm lebih pendek dari map Surround Target 8px, hal ini akan mempengaruhi kinerja komputasi apabila diterapkan pada kondisi lingkungan yang dinamis*.

Kata Kunci *— Algoritma A*, Base station, Perencanaan jalur, Simulator*

Abstract – *The Wheeled Soccer Robot Competition in Indonesia is a robotics competition involving skills in strategizing on robots, robot navigation systems, and path planning on robots. This research discusses the implementation of the A* algorithm on a base station of the Abimanyu robotic team of Universitas Nusantara PGRI Kediri as a path planning algorithm, path planning is applied in simulation without involving sending data to the robot. In the application of the algorithm, the field is first divided into several grids that represent nodes that can be traversed by each algorithm. The research conducted obtained the results that the A* algorithm achieves faster execution time on maps with a grid size of 32px under any conditions, on Empty 32px maps get an execution time of 0.091 s faster than the Empty 8px maps with a path length of 0.1 cm shorter, on Simple 32px maps get an execution time of 0.542 s faster than the simple 8px map with a path length of 0.2 cm shorter, and the biggest difference is in the Surround Target 32px map type which has an execution time of 1.653 s faster and a path length of 0.5 cm shorter than the Surround Target 8px map, this will affect computational performance when applied to dynamic environmental conditions.*

Keywords *— A* Algorithm, Base station, Path planning, Simulator*

1. PENDAHULUAN

Kontes Robot Indonesia merupakan kompetisi yang diadakan oleh Pusat Prestasi Nasional (puspresnas) setiap tahunnya dan diikuti oleh mahasiswa dari seluruh perguruan tinggi di Indonesia. Pada tahun 2022, KRI masih diadakan secara *online* pada tingkat wilayah, dan diadakan secara *offline* pada tingkat nasional. Peserta yang berhasil lolos ke babak nasional akan mempertandingkan robotnya secara langsung pada lapangan pertandingan. KRI mempertandingkan enam divisi, salah satunya yang masih diikuti oleh Universitas Nusantara PGRI Kediri hingga saat ini merupakan divisi Kontes Robot Sepak Bola (KRSBI) Beroda [1].

Tim Abimanyu merupakan tim dari komunitas robotik Universitas Nusantara PGRI Kediri yang mengikuti divisi KRSBI Beroda tersebut. Pada kompetisi ini robot digerakkan secara autonomus melalui *base*

station yang berfungsi untuk mengirimkan perintah dan melakukan monitoring posisi robot [2] sehingga diperlukan strategi pada robot untuk mencetak gol. Pergerakan robot untuk mencetak gol dan bergerak menuju titik tujuan tentunya akan menghadapi halangan berupa robot lawan. Oleh karena itu perlu adanya algoritma perencanaan jalur (*path planning*) yang optimal dan sesuai dengan kondisi robot agar dapat menghindari halangan, serta diperlukan sebuah simulator sebagai simulasi pergerakan robot dan untuk mengetahui panjang *path* serta waktu eksekusi yang diperlukan dari masing-masing algoritma *path planning.*

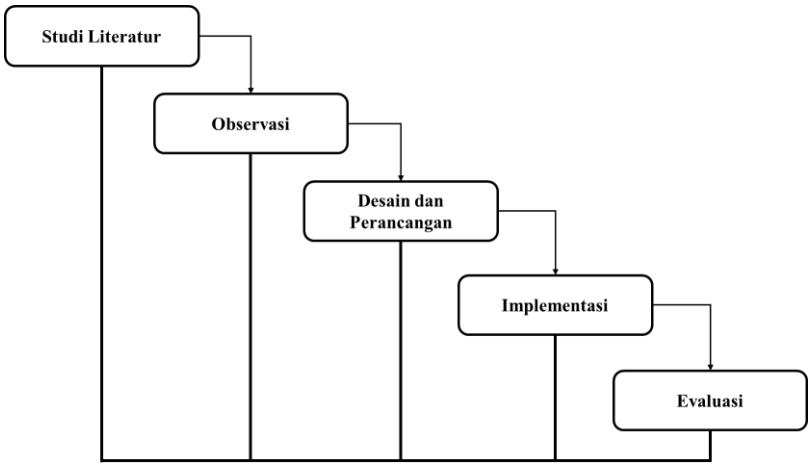
Path planning merupakan suatu metode yang berfungsi untuk melakukan perencanaan jalur dari titik awal hingga titik akhir baik pada lingkungan yang statis maupun dinamis [3]. Algoritma yang umum digunakan dalam path planning diantaranya adalah Algoritma *A** dan Algoritma *Dijkstra*. Berdasarkan hasil penelitian sebelumnya dari kedua algoritma tersebut yang paling efektif dalam *path planning* adalah Algoritma *A** [4]. Algoritma *A** juga dapat menghindari objek halangan dan menuju titik *finish* secara optimal pada *map* dengan resolusi yang tinggi [5]. Dalam melakukan perancangan visualisasi pada *base station*, diperlukan skala untuk menentukan perbandingan antara lapangan sesungguhnya dengan yang ada pada visualisasi [6]. Pada penelitian ini diterapkan algoritma *A** untuk perencanaan jalur robot sepak bola beroda.

Dalam menentukan jalur, diperlukan metode pengukuran jarak. Metode pengukuran jarak yang paling sering digunakan diantaranya *euclidean distance* dan *manhattan distance*. *Euclidean distance* merupakan fungsi pengukuran jarak yang umum digunakan pada K-NN dengan menerapkan pembobotan fitur [7], sedangkan *manhattan distance* menghitung perbedaan absolut dari kedua titik koordinat dari dua buah objek yang diketahui. Pada algoritma *A*, manhattan distance* bekerja lebih baik dibandingkan *euclidean distance* dikarenakan perhitungannya yang lebih sederhana [8] sehingga *path planning* pada penelitian ini menerapkan perhitungan jarak menggunakan *manhattan distance*.

2. METODE PENELITIAN

2.1. Langkah Penelitian

Metode yang digunakan dalam implementasi ini adalah metode *waterfall*. Metode *waterfall* atau yang sering disebut *classic life cycle* merupakan metode pada pengembangan *software* yang menerapkan pendekatan sistematis dan berurutan [9]. Menurut Satiansyah dan Oktaviana, metode Waterfall adalah pendekatan sistematis dan berurutan dengan model klasik terhadap tingkat kemajuan sistem dalam semua analisis, desain, kode, pengujian, dan pemeliharaan [10].



Gambar 1. Metode waterfall

Gambar 1 merupakan tahapan metode *waterfall* yang diterapkan pada penelitian ini. Studi literatur dilakukan untuk merumuskan masalah serta analisa sistem, observasi dilakukan untuk menganalisa kebutuhan robot secara langsung agar dapat memberikan manfaat terhadap mitra penelitian, Desain dan perancangan dilakukan untuk menetapkan algoritma dan perancangan sistem, kemudian dilakukan implementasi dan evaluasi hasil.

2.2. Desain Sistem

Pembuatan perangkat lunak memerlukan suatu desain dan perencanaan untuk mempermudah pengerjaan perangkat lunak kedepannya [11]. Desain sistem yang dibuat meliputi ukuran lapangan yang menyesuaikan aturan perlombaan, virtual grid untuk algoritma *path planning*, pemaparan rancangan algoritma, dan pencarian nilai

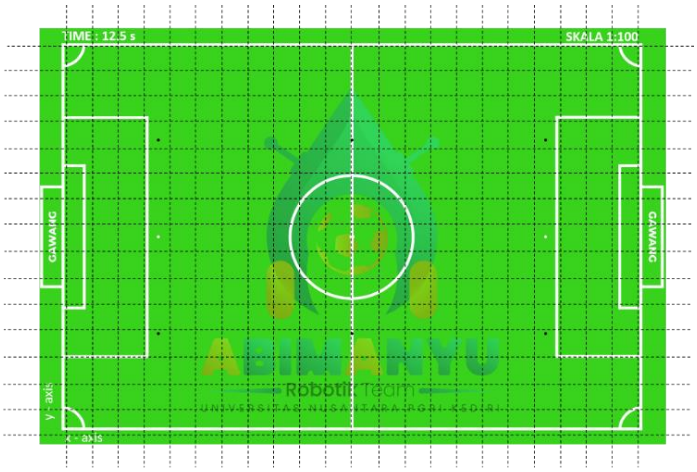
heuristik. Simulator yang diperlukan pada penelitian ini dirancang dengan bahasa pemrograman *python* dengan modul *pygame* yang berperan untuk menampilkan simulasi *path planning* pada simulator yang diusulkan [12]. *Pygame* merupakan salah satu module pada *Python* yang dirancang untuk membuat *video game*, modul ini disediakan secara gratis dan dibangun di atas *Simple DirectMedia Layer Library* (SDL) yang memudahkan dalam hal akses elemen suara maupun visual [13] sehingga mampu mempermudah visualisasi *path planning* pada robot sepak bola beroda. Penggunaan bahasa pemrograman *python* didasarkan pada base station milik tim Abimanyu yang sebelumnya telah dikembangkan pada beberapa penelitian [14], [15], sehingga penelitian ini tidak merubah fitur lainnya yang ada pada *base station* terdahulu. *Python* juga merupakan bahasa pemrograman yang banyak dipelajari hingga saat ini, terutama bagi pemula karena sifatnya yang lintas platform, yakni dapat digunakan pada *Windows, Linux*, hingga *MacOS* [16], sehingga jika *base station* perlu diterapkan pada perangkat dengan sistem operasi yang berbeda, *user* tidak kesulitan dalam melakukan instalasi.

2.2.1. Skala Lapangan

Pada pertandingan KRSBI Beroda yang diadakan secara luring pada tahun 2022 menerapkan ukuran lapangan asli yang memiliki panjang 12 meter dan lebar 8 meter. Penerapan area lapangan pada simulator menggunakan skala 1:100 yang berarti menerapkan ukuran 742 x 495 px pada pemrograman *python*.

2.2.2. Ukuran Virtual *Grid*

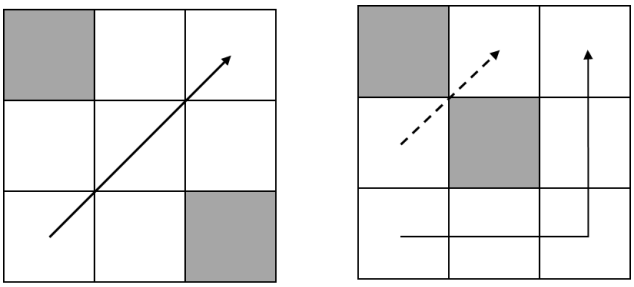
Berdasarkan skala lapangan yang telah ditentukan, didapatkan ukuran robot setelah diskala yaitu 32px, sehingga ukuran virtual *grid* juga menyesuaikan, akan tetapi untuk mengetahui efektifitas algoritma dalam beberapa kondisi, diperlukan .ukuran *grid* yang lebih kecil yaitu 8px. Gambar 2 merupakan penggambaran virtual grid yang akan diterapkan.



Gambar 2. Virtual *grid* pada lapangan

2.2.3. *Path Planning*

Pada perencanaan jalur (*path planning*) umumnya akan terdapat beberapa *obstacle* yang mengharuskan robot untuk menghindari *obstacle* tersebut serta menemukan jalur paling optimal [17]. Pada umumnya algoritma *path planning* dapat melakukan pencarian ke 8 arah, namun dalam penelitian ini, arah digonal hanya diterapkan jika *grid* yang dilewati tidak memiliki *obstacle* di arah vertikal dan horizontal. Pertimbangan arah perluasan algoritma secara diagonal diperlukan untuk memperhatikan *safety area* pada robot. Jika pada arah vertikal dan horizontal tidak terdapat *obstacle*, maka algoritma dapat melakukan pencarian secara diagonal seperti pada Gambar 3 (a) tetapi jika pada arah vertikal dan horizontal terdapat *obstacle*, maka pencarian algoritma hanya dilakukan secara 4 arah (vertikal dan horizontal). Pada Gambar 3 (b) garis putus-putus adalah grid yang tidak dapat dilalui.



Gambar 3. Pemilihan jalur diagonal

2.2.4. Algoritma *A**

Algoritma *A** merupakan salah satu algoritma *path planning* yang merupakan pengembangan dari algoritma *Dijkstra* dan merupakan bentuk modern dari algoritma *Breadth-First Search* (BFS) [18]. Dalam pencarian jalur yang lebih baik dan efisien, algoritma *A** menerapkan metode pencarian heuristik, yaitu metode yang memanfaatkan fungsi perhitungan biaya dari satu titik ke titik lainnya, dan melakukan perhitungan nilai pada setiap titik koordinat di sekitarnya, sehingga dapat menemukan jalur terpendek yang dapat dilalui. Algoritma *A** menggunakan persamaan yang dituliskan sebagai berikut:

$$f(n) = g(n) + h(n) \quad \text{..... (1)}$$

Pada persamaan (1), *f(n)* merupakan estimasi harga terkecil untuk solusi jalur sepanjang *n*, *g(n)* merupakan jarak dari titik start ke titik *n*, dan *h(n)* merupakan estimasi jarak dari titik *n* ke titik goal.

2.2.5. *Manhattan Distance*

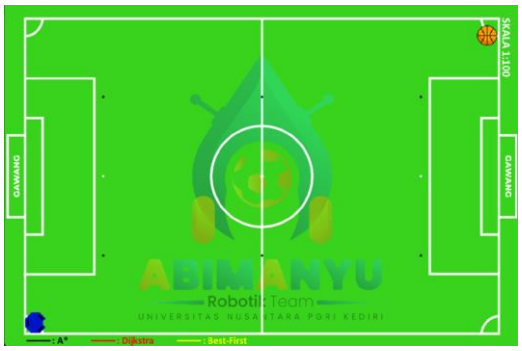
Untuk menghitung pencarian heuristik pada algoritma *A** digunakan *manhattan distance* yang dinotasikan dalam persamaan (2). Pada algoritma *A**, *manhattan distance* bekerja lebih baik dibandingkan *Euclidean distance* dikarenakan perhitungannya yang lebih sederhana [8].

$$D(u,v) = |x_1 - x_2| + |y_1 - y_2| \quad \text{.....(2)}$$

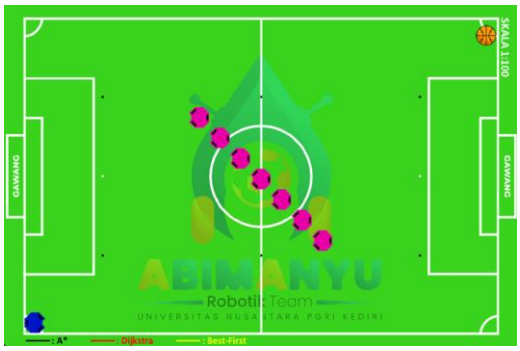
Manhattan distance menghitung perbedaan absolut dari kedua titik koordinat dari dua buah objek yang diketahui. Sebagai contoh, jika diketahui titik *u* memiliki koordinat *(x1, y1)* dan titik *v* dengan koordinat *(x2, y2)*, maka perhitungan jarak antara kedua titik tersebut dapat digambarkan menggunakan rumus *manhattan distance* pada persamaan (2).

3. HASIL DAN PEMBAHASAN

Pengujian dilakukan pada 3 kondisi *map* dengan ukuran *grid* yang berbeda di setiap kondisinya. Pengujian pertama yaitu pada *empty map* seperti pada gambar 4 untuk mengetahui perbandingan bentuk *trajectory* (lintasan) pada ukuran *grid* yang berbeda. Pengujian kedua yaitu pengujian *simple map* dengan kondisi *map* pada gambar 5, yaitu *map* dengan obstacle diagonal di tengah lapangan. Pengujian ketiga yaitu *surround target*, merupakan kondisi dimana *obstacle* mengitari *target* dengan satu sisi yang terbuka seperti pada gambar 6. Pada akhir pengujian akan dibahas mengenai efektifitas algoritma pada kondisi *map* tertentu.



Gambar 4. Kondisi *empty map*

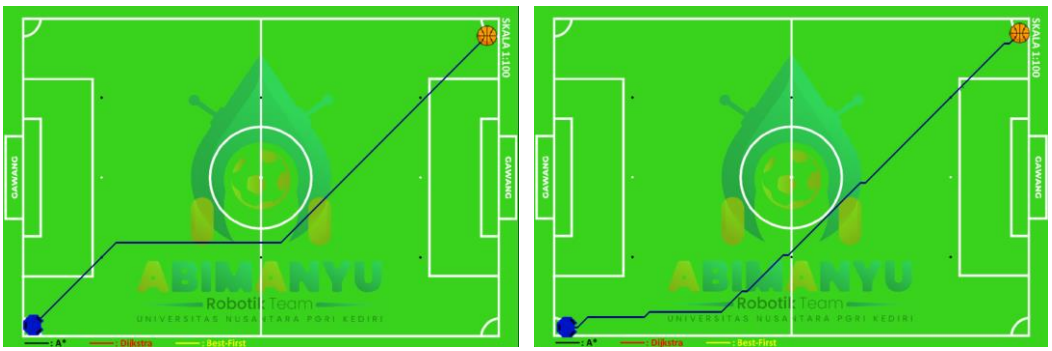


Gambar 5. Kondisi *simple map*

Gambar 6. Kondisi *surround target*

3.1. *Empty Map*

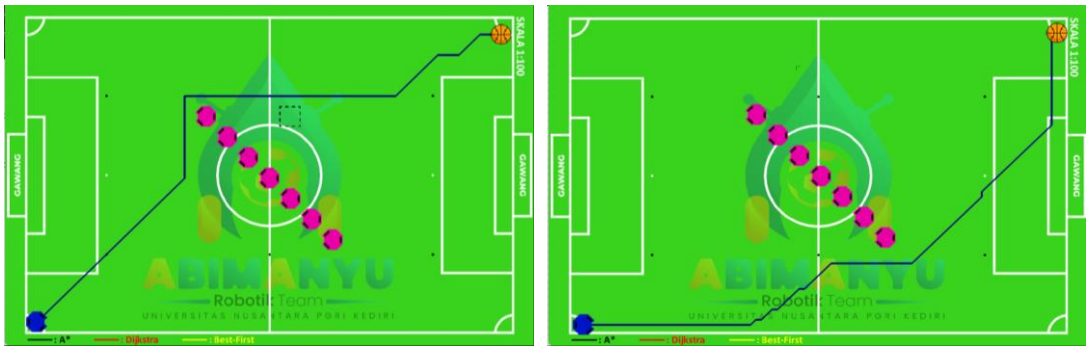
Pada pengujian *empty map* didapatkan perbandingan *trajectory* pada gambar 7, jarak yang ditempuh menghasilkan panjang yang tidak berbeda jauh, akan tetapi menghasilkan terlalu banyak belokan pada ukuran *grid* 8px.



Gambar 7. (a) *Empty map* 32px (b) *Empty map* 8px

3.2. *Simple Map*

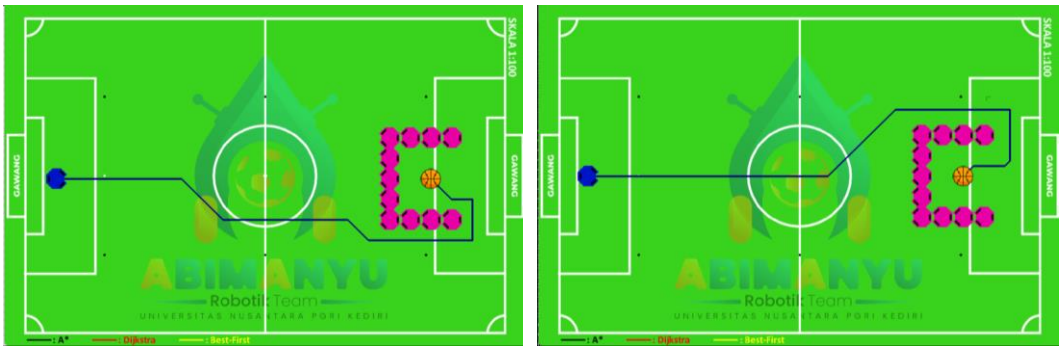
Pengujian berikutnya dilakukan pada *simple map* dengan ukuran *grid* 32px dan 8px, gambar 8 merupakan perbandingan *trajectory* yang dihasilkan ketika algoritma menghadapi *obstacle* yang diletakkan secara diagonal di tengah lapangan. Sama seperti pengujan sebelumnya, *trajectory* yang dihasilkan pada *map* berukuran *grid* 8px memiliki terlalu banyak belokan yang tidak perlu.



Gambar 8. (a) *Simple map* 32px (b) *Simple map* 8px

3.3. *Surround Target*

Pengujian di akhir dilakukan dengan penempatan *obstacle* yang mengelilingi *target* dengan satu sisi yang terbuka, hasil yang didapatkan tertera pada gambar 9. Pada pengujian ini algoritma *A** memiliki waktu eksekusi yang lebih lambat meskipun jarak tempuhnya lebih pendek.



Gambar 9. (a) *Surround target* 32px (b) *Surround target* 8px

3.4. Analisa Hasil

Berdasarkan ketiga skenario pengujian, didapatkan rata-rata waktu eksekusi dan panjang *path* yang dihasilkan algoritma pada tabel 1. Perolehan waktu eksekusi dan panjang path dibandingkan dengan ukuran grid yang berbeda pada tipe map yang sama, hal ini untuk mempermudah perolehan kesimpulan akhir pada penelitian.

Tabel 1. Hasil pengujian pada seluruh skenario

Tipe Map	Ukuran grid	Waktu Eksekusi (s)	Panjang *Path* (cm)
Empty	32px	0.032	14.4
	8px	0.123	14.3
Simple	32px	0.036	15.6
	8px	0.578	15.8
Surround Target	32px	0.045	13.3
	8px	1.698	13.8

Pada tabel 1 dapat diketahui bahwa *path planning* pada algoritma *A** menghabiskan waktu eksekusi yang lebih banyak pada ukuran *grid* 8px dibandingkan 32px, hal ini dikarenakan pada ukuran *grid* 8px terdapat lebih banyak *grid* yang perlu dieksekusi. Sedangkan berdasarkan tipe *map*, kondisi *surround target* memakan waktu eksekusi paling lama dibandingkan *empty map* dan *simple map* meskipun jarak tempuhnya lebih pendek, hal ini karena algoritma *A** perlu melakukan lebih banyak perluasan ke *grid* lainnya dan menempuh jalur yang lebih panjang.

Perbedaan ukuran *grid* dan kondisi lapangan berpengaruh cukup besar pada kinerja algoritma *A**, hal ini dapat diketahui dari selisih waktu yang didapatkan pada masing-masing tipe *map*. Pada tipe map *Empty* memiliki selisih waktu 0.091 s dengan selisih panjang *path* 0.1 cm, pada tipe *map Simple* memiliki selisih waktu eksekusi sebesar 0.542 s dengan selisih panjang *path* 0.2 cm, selisih terbanyak terjadi pada tipe *map surround target* yakni sebesar 1.653 s dengan selisih panjang *path* 0.5 cm. Ukuran *grid* 32px memakan waktu eksekusi yang lebih singkat

dan jarak yang lebih pendek, hal ini dapat membantu meringankan kinerja robot agar lebih baik dan lebih cepat untuk menemukan jalur paling optimal.

4. SIMPULAN

Tipe *map* bertujuan untuk mengetahui bagaimana kinerja algoritma jika dihadapkan dengan kondisi musuh tertentu, yang mana kondisi musuh tidak dapat diprediksi ataupun ditentukan dalam perlombaan. Pengujian dengan ukuran *grid* yang berbeda bertujuan untuk mengetahui pada ukuran *grid* mana kinerja algoritma *A** menjadi lebih optimal. Berdasarkan hasil analisis yang telah dijabarkan, ukuran grid 32px mendapatkan hasil yang lebih baik pada tipe *map* apapun, perbedaan yang paling terlihat ada pada tipe *map surround target* yang memiliki selisih waktu 1.653 s lebih cepat pada ukuran *grid* 32px, dan jarak tempuh 0.5 cm lebih pendek, sehingga penerapan *map* dengan ukuran grid 32px lebih diutamakan untuk diterapkan pada simulator agar path planning dapat berjalan dengan lebih optimal sekalipun pada kondisi lapangan yang dinamis.

5. SARAN

Dari penelitian dan pengujian yang telah dilakukan, dapat diambil beberapa saran guna pengembangan dan kesempurnaan pada penelitian berikutnya dengan topik yang serupa, antara lain (a) Pengembangan algoritma untuk perencanaan jalur yang menghasilkan lintasan lebih *smooth* agar tidak menghabiskan terlalu banyak energi ketika diterapkan pada robot, (b) Pengembangan algoritma agar dapat diterapkan pada lingkungan yang dinamis.(c) Pengembangan simulator agar dapat menerapkan hasil *path planning* langsung pada robot.

DAFTAR PUSTAKA

[1] Pusat Prestasi Nasional, "PANDUAN KONTES ROBOT INDONESIA (KRI) TAHUN 2022," 2022.

[2] N. M. Figueiredo, A. J. R. Neves, N. Lau, A. Pereira, dan G. Corrente, "Control and Monitoring of a Robotic Soccer Team: The Base Station Application," 2009. [Daring]. Tersedia pada: http://www.ieeta.pt/atri/cambada

[3] A. Gasparetto, P. Boscariol, A. Lanzutti, dan R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," dalam *Motion and Operation Planning of Robotic Systems*, vol. 29, Kluwer Academic Publishers, 2015, hlm. 3–27. doi: 10.1007/978-3-319-14705-5_1.

[4] A. W. R. Ramadhan dan D. Udjulawa, "Perbandingan Algoritma Dijkstra dan Algoritma A Star Pada Permainan Pac-Man," 2020.

[5] N. Setyawan, D. Nur Fajar, dan K. Hidayat, "PERENCANAAN JALUR ROBOT SEPAK BOLA UMMIROS MENGGUNAKAN ALGORITMA A*," dalam *Seminar Nasional Teknologi dan Rekayasa (SENTRA)*, Kota Malang, 2019, hlm. 52–57.

[6] V. Indrarta, S. Tri Rasmana, dan I. Puspasari, "PERANCANGAN VISUALISASI POSISI ROBOT PADA PENGENDALI ROBOT SEPAK BOLA BERODA," *Journal of Control and Network Systems*, vol. 8, no. 2, hlm. 52–60, 2019, [Daring]. Tersedia pada: http://jurnal.stikom.edu/index.php/jcone

[7] M. J. Vikri dan R. Rohmah, "Penerapan Fungsi Exponential Pada Pembobotan Fungsi Jarak Euclidean Algoritma K-Nearest Neighbor," *Generation Journal*, vol. 6, no. 2, hlm. 98–105, 2022.

[8] S. K. Sharma dan S. Kumar, "COMPARATIVE ANALYSIS OF MANHATTAN AND EUCLIDEAN DISTANCE METRICS USING A* ALGORITHM 1 2," 2016. [Daring]. Tersedia pada: http://wiki.gamegardens.com/Path_Finding_Tut

[9] T. Sanubari, C. Prianto, dan N. Riza, *Odol (one desa one product unggulan online) penerapan metode Naive Bayes pada pengembangan aplikasi e-commerce menggunakan Codeigniter*. Bandung: Kreatif, 2020.

[10] A. Rohman, R. Y. Perkasa, A. S. Hidaytullah, dan M. G. Rohman, "Implementasi Metode Waterfall Pada Rancang Bangun Sistem Pengarsipan Surat Berbasis Web," *Generation Journal*, vol. 6, no. 2, hlm. 134–143.

[11] A. W. M. Fauzi dan A. Momon, "Sistem Informasi Pada UMKM Kedai Jalinan Coffe," *Generation Journal*, vol. 7, no. 2, hlm. 126–132, 2023.

[12] R. F. Waliulu, A. Hendriawan, S. Supardi, dan A. Pramono, "PROTOTYPE DETEKSI OBJEK MENGGUNAKAN RASPBERRY PI MELALUI MODUL SENSOR ULTRASONIK HC-SR04," 2021. [Daring]. Tersedia pada: www.ojs.poltekpelsorong.ac.id

[13] S. Kelly, *Python, PyGame and Raspberry Pi Game Development*. Canada: Apress, 2016. Diakses: 26 November 2022. [Daring]. Tersedia pada: https://www.google.co.id/books/edition/Python_PyGame_and_Raspberry_Pi_Game_Deve/RovJDQAAQBAJ?hl=id&gbpv=1&dq=python+pygame&pg=PA59&printsec=frontcover

[14] W. Firmansyah, "IMPLEMENTASI SISTEM MULTITHREADING DENGAN MENGGUNAKAN PROTOKOL TCP (TRANSMISSION CONTROL PROTOCOL) PADA BASESTATION ROBOT SEPAK BOLA BERODA," Universitas Nusantara PGRI Kediri, Kediri, 2022.

[15] W. Firmansyah, J. Sahertian, dan J. Sulaksono, "Implementasi Fitur Manual Keyboard Menggunakan Header Pada Basestation Robot Sepak Bola Beroda Abimanyu," dalam *Seminar Nasional Inovasi Teknologi*, Kediri, Jul 2022, hlm. 247–252.

[16] Jubilee Enterprise, *Python untuk Membuat Game hingga Face Detector*. PT Elex Media Komputindo, 2020.

[17] N. A. Shiltagh dan L. D. Jalal, "Path Planning of Intelligent Mobile Robot Using Modified Genetic Algorithm 32," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 3, no. 2, hlm. 31–36, 2013.

[18] G. Datta Lohith, C. N. Sujatha, C. Pamuleti, K. Mahesh, dan S. Sai Charan, "PATH ROUTING ON A MAP AND SIMULATION USING A STAR ALGORITHM."