Optimasi Penghitungan Jalur Terpendek pada Large-scale Road Network Menggunakan Algoritma A*

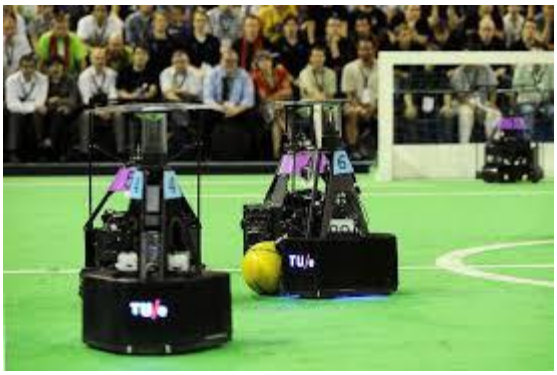
Owen Tobias Sinurat - 13522131
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522131@std.stei.itb.ac.id

***Abstrak*— Penentuan jalur terpendek dalam jaringan jalan berskala besar merupakan tantangan yang signifikan dalam bidang ilmu komputer dan rekayasa transportasi. Algoritma A* (A-star) merupakan salah satu metode pencarian jalur yang efektif dan efisien, terutama karena kemampuannya untuk menggabungkan pendekatan heuristik dengan pencarian jalur klasik. Makalah ini membahas optimasi penghitungan jalur terpendek pada jaringan jalan berskala besar menggunakan algoritma A*. Kami memperkenalkan beberapa teknik optimasi, termasuk penggunaan heuristik yang lebih akurat, manajemen data yang efisien, dan teknik penyaringan simpul yang tidak relevan. Eksperimen dilakukan pada dataset jaringan jalan besar untuk mengevaluasi kinerja algoritma A* yang telah dioptimasi. Hasilnya menunjukkan peningkatan signifikan dalam kecepatan pencarian jalur tanpa mengorbankan akurasi. Implementasi ini diharapkan dapat memberikan kontribusi pada pengembangan sistem navigasi dan manajemen lalu lintas yang lebih cerdas dan responsif.**

Keywords—A*; astar; algoritma; robotika; robot sepakbola; robocup; path-planning; rute; heuristik.

I. PENDAHULUAN

Dalam permainan sepakbola, ada banyak aspek yang diperlukan untuk meraih kemenangan, seperti operan, gerakan, tendangan, pertahanan, serangan, dan lainnya. Hal itu pun berlaku pada robot sepakbola beroda.



Gambar 1 Kompetisi RoboCup Middle Size League

Mobilitas adalah hal terpenting dalam permainan sepakbola yang dimainkan oleh robot beroda. Kenapa? Karena tidak seperti manusia, robot tidak memiliki kecerdasan sendiri sehingga memerlukan perintah untuk menentukan arah gerak selanjutnya.

Salah satu algoritma yang paling banyak digunakan untuk menyelesaikan masalah ini adalah Algoritma A* (A-star). Algoritma A* terkenal karena kemampuannya menggabungkan pendekatan heuristik dengan pencarian jalur klasik, yang membuatnya lebih efisien dibandingkan metode lain seperti Dijkstra, terutama pada jaringan besar dan kompleks. Namun, ketika diterapkan pada jaringan jalan berskala besar, Algoritma A* menghadapi berbagai tantangan, seperti kompleksitas komputasi yang tinggi dan kebutuhan akan manajemen data yang efisien. Untuk mengatasi tantangan ini, diperlukan optimasi lebih lanjut agar algoritma dapat berfungsi dengan baik dalam konteks jaringan jalan yang sangat besar.

Makalah ini bertujuan untuk mengeksplorasi dan mengimplementasikan berbagai teknik optimasi pada Algoritma A* untuk meningkatkan kinerjanya dalam mencari jalur terpendek sambil menghindari *collision*. Beberapa teknik yang dibahas meliputi penggunaan heuristik yang lebih akurat, strategi penyimpanan data yang efisien, dan metode penyaringan simpul yang tidak relevan. Dengan mengoptimalkan aspek-aspek ini, diharapkan Algoritma A* dapat memberikan solusi yang lebih cepat dan tetap akurat dalam aplikasi robot sepakbola beroda. Struktur makalah ini dimulai dengan tinjauan literatur terkait Algoritma A* dan berbagai teknik optimasi yang telah diterapkan sebelumnya. Selanjutnya, metode dan pendekatan yang diusulkan dalam penelitian ini akan dijelaskan secara detail. Bagian hasil dan diskusi akan menyajikan evaluasi kinerja dari algoritma yang telah dioptimasi, diikuti dengan kesimpulan yang merangkum temuan utama serta saran untuk penelitian lebih lanjut.

II. DASAR TEORI

A. *Algoritma A**

Algoritma A* adalah salah satu algoritma pencarian jalur yang paling populer dan efektif yang digunakan dalam berbagai aplikasi mulai dari game komputer hingga robotika dan navigasi. Algoritma ini merupakan pengembangan dari algoritma Dijkstra, yang menemukan jalur terpendek antara dua titik dalam graf, dengan menambahkan fungsi heuristik untuk memperkirakan jarak yang tersisa ke tujuan. Dengan

kombinasi ini, A* mampu mengoptimalkan pencarian jalur dengan lebih efisien, baik dalam hal waktu maupun sumber daya komputasi. Algoritma A* bekerja berdasarkan prinsip pencarian graf. Graf terdiri dari simpul-simpul (nodes) yang saling terhubung oleh sisi-sisi (edges). Setiap sisi memiliki bobot yang menunjukkan biaya atau jarak antara dua simpul yang terhubung. Algoritma A* menggunakan dua fungsi utama, yaitu fungsi g(n) yang mengukur biata dari titik awal ke simpul n dan fungsi h(n) yang memperkirakan biaya dari simpul n ke tujuan. Fungsi evaluasi, f(n), untuk algoritma pencarian A* adalah sebagai berikut:

$$f(n) = g(n) + h(n)$$

Algoritma ini merupakan penggabungan algoritma Uniform Cost (UCS) dan Greedy Best-First Search (GBFS). UCS mendapatkan nilai f(n) menggunakan fungsi g(n) sedangkan GBFS mendapatkan nilai f(n) menggunakan fungsi h(n). Keberhasilan algoritma A* sangat bergantung pada pemilihan fungsi heuristik yang tepat. Fungsi heuristik haru admissible, artinya tidak boleh melebihi biaya sebenarnya dari simpul saat ini ke tujuan. Beberapa contoh fungsi heuristik yang umum digunakan meliputi:

• Manhattan Distance, Digunakan untuk grid yang memungkinkan pergerakan hanya dalam arah vertikal atau horizontal.

$$h(n) = |xgoal - xcurrent| + |ygoal - ycurrent|$$

• Euclidean Distance, Digunakan untuk pergerakan di ruang yang memungkinkan pergerakan diagonal.

$$h(n) = \sqrt{(xgoal - xcurrent)2 + (ygoal - ycurrent)2}$$

Pemilihan fungsi heuristik yang tepat dapat mempercepat pencarian jalur dan mengurangi beban komputasi. Algoritma A* secara garis besar dapat dijelaskan seperti berikut:

1. Masukkan node awal ke daftar node terbuka (open)
2. Loop langkah-langkah dibawah ini:
    a. Cari node (n) dengan nilai f(n) yang paling kecil dalam open list, dan node ini sekarang menjadi current node.
    b. Keluarkan current node dari daftar node terbuka dan masukkan ke open list
    c. Untuk setiap tetangga dari current node jika tidak dapat dilalui atau sudah ada dalam close list , abaikan. Jika belum ada di open list buat current node parent dari node tetangga ini, simpan nilai f, g, dan h dari node ini. Jika sudah ada di daftar node terbuka cek apabila node tetangga ini lebih baik menggunakan nilai g sebagai ukuran. Jika lebih baik ganti parent dari node ini di opentlist menjadi current node, lalu kalkulasikan ulang nilai g dan fdari node ini.
    d. Hentikan looping jika node tujuan telah ditambah ke daftar node terbuka yang berarti rute ditemukan dan jika belum menemukan node akhir (tujuan) sementara openlist kosong atau berarti tidak ada rute.

B. Perencanaan Rute

Perencanaan rute (route planning) adalah proses menentukan jalur terbaik dari satu titik ke titik lain, yang dapat mencakup beberapa perhentian atau tujuan. Dalam konteks robot sepakbola beroda, perencanaan rute ditetapkan dengan penentuan jalur paling efisien dan aman dari tabrakan dengan objek apapun. Beberapa hal yang dipertimbangkan seperti posisi robot musuh, posisi robot kawan, posisi bola, posisi robot di lapangan, dan estimasi pergerakan robot lawan akan membantu pengambilan keputusan robot untuk menentukan rute terbaik.

Langkah-langkah perencanaan rute:

1. Inisialisasi dan Pemahaman Lingkungan

    Proses ini melibatkan pengumpulan data dari berbagai sensor, termasuk sensor ultrasonik, kamera, dan IMU untuk mendapatkan informasi tentang posisi robot, bola, rekan satu tim, dan lawan. Data ini digunakan untuk membangun representasi peta lapangan sepakbola yang mencakup posisi tetap seperti gawang dan posisi dinamis seperti bola dan robot lainnya. Lokalisasi dilakukan dengan algoritma pemrosesan citra dan data sensor untuk memperkirakan posisi dan orientasi robot secara akurat.

2. Perumusan Masalah dan Penerapan Algoritma Perencanaan Jalur

    Pada tahap ini, titik awal dan tujuan rute ditentukan, seperti posisi awal robot dan lokasi bola atau posisi strategis di lapangan. Selain itu, identifikasi hambatan dilakukan untuk mengenali area yang tidak dapat dilalui, termasuk posisi robot lawan dan batas lapangan. Untuk menemukan jalur terpendek, Algoritma A* diterapkan dengan menggunakan fungsi biaya total

    $$f(n)=g(n)+h(n)$$

    di mana g(n) adalah biaya aktual dari titik awal ke simpul n dan h(n) adalah estimasi biaya dari simpul n ke tujuan.

3. Langkah Optimasi Jalur dan Penghindaran Hambatan Dinamis

    Optimasi heuristik dilakukan untuk menyesuaikan fungsi estimasi biaya agar lebih sesuai dengan kondisi lapangan sepakbola, seperti mempertimbangkan kecepatan rata-rata robot dan faktor dinamis lainnya. Selain itu, integrasi metode penghindaran hambatan dinamis sangat penting untuk memastikan jalur tetap optimal dan aman dari gangguan seperti robot lawan yang bergerak.

4. Kontrol Gerakan dan Penyesuaian Real-time

    Pada tahap ini, algoritma kontrol digunakan untuk mengatur kecepatan dan arah gerak robot sesuai dengan jalur yang telah direncanakan. Kontrol umpan balik (feedback control) diterapkan untuk menyesuaikan gerakan robot secara real-time berdasarkan data sensor terbaru, sehingga robot dapat

tetap berada di jalur optimal meskipun terdapat perubahan kondisi lapangan.

C. *Robot Sepakbola Beroda*

Robot sepakbola beroda adalah jenis robot otonom yang dirancang untuk bermain sepakbola dalam lingkungan yang terstruktur seperti pertandingan RoboCup. Robot ini dilengkapi dengan roda untuk mobilitas, sensor untuk persepsi lingkungan, dan algoritma untuk pengambilan keputusan serta strategi permainan. Robot sepakbola beroda biasanya memiliki beberapa komponen utama:

1. Sistem Mobilitas: Sistem ini meliputi roda, motor, dan mekanisme penggerak yang memungkinkan robot bergerak dengan cepat dan akurat di atas lapangan. Ada dua jenis utama konfigurasi roda yang digunakan:
    a. Roda Diferensial: Menggunakan dua roda penggerak independen dan roda penyeimbang, memungkinkan gerakan linier dan rotasi.
    b. Roda Omni-Directional: Menggunakan roda omni atau mecanum yang memungkinkan gerakan dalam semua arah tanpa perlu rotasi.
2. Sistem Sensor: Sensor digunakan untuk mendapatkan informasi dari lingkungan sekitar. Sensor yang umum digunakan termasuk:
    a. Sensor Ultrasonik: Untuk mendeteksi jarak ke objek di sekitar.
    b. Kamera: Untuk pengenalan objek seperti bola dan pemain lawan.
    c. IMU (Inertial Measurement Unit): Untuk mengetahui orientasi dan gerakan robot.
3. Sistem Pengolahan dan Pengambilan Keputusan: Komputer onboard atau mikrokontroler yang memproses data dari sensor dan membuat keputusan taktis serta strategis. Algoritma kecerdasan buatan (AI) sering digunakan dalam pengambilan keputusan ini.

III. IMPLEMENTASI DAN PENGUJIAN

A. *Identifikasi Masalah*

Robot sepakbola beroda tentunya akan berpindah tempat saat bermain. Hal ini menjadi aspek krusial dalam memenangkan permainan. Namun, dalam perpindahan dari satu titik ke titik lain tidaklah semudah itu, mengingat adanya objek lain yang berada di lapangan, seperti robot kawan, robot lawan, dan bola. Permasalahan ini dapat diselesaikan dengan menggunakan algoritma A* untuk mencapai titik tujuan dengan jalur tercepat tanpa menabrak objek lain *(obstacle).*

B. *Pemetaan Masalah ke Algoritma A**

Berikut adalah pemetaan masalah ke algoritma A*:

1. Lokasi robot sekarang sebagai titik awal.
    - Inisialisasi Posisi: Posisi awal robot ditentukan berdasarkan data sensor yang mengidentifikasi koordinat (x, y) robot di lapangan.
    - Titik Awal (Start Node): Node yang merepresentasikan posisi awal robot akan diinisialisasi sebagai titik awal dalam Algoritma A*.
2. Jarak riil sebagai berat sisi dan nilai g(n)
    - Berat Sisi (Edge Weight): Setiap gerakan dari satu node ke node lainnya memiliki berat yang dihitung berdasarkan jarak riil antara dua titik tersebut. Ini menjadi nilai g(n), yaitu biaya sebenarnya dari titik awal ke node saat ini.
    - Perhitungan Jarak: Gunakan metrik Euclidean atau Manhattan untuk menghitung jarak antara dua titik dalam grid. Misalnya, jika menggunakan Euclidean distance, jaraknya adalah (x2−x1)2+(y2−y1)2\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}(x2−x1)2+(y2−y1)2.
3. Jarak titik ke tujuan sebagai heuristik h(n)
    - Heuristik (Heuristic): Fungsi heuristik h(n) digunakan untuk memperkirakan biaya terendah dari node saat ini ke node tujuan. Ini sering kali menggunakan jarak Euclidean atau Manhattan antara node saat ini dan node tujuan.
    - Estimasi Biaya: Fungsi heuristik harus dirancang untuk tidak melebihi biaya terendah aktual, sehingga Algoritma A* tetap optimal dan lengkap.
4. Membangun graf dengan informasi yang sudah didapatkan
    - Representasi Grid: Lapangan sepakbola diwakili oleh grid 2D di mana setiap sel di grid merepresentasikan posisi yang dapat dicapai oleh robot.
    - Node dan Edge: Setiap sel grid adalah node, dan tepi (edge) antara node dihasilkan berdasarkan kemampuan robot untuk bergerak dari satu sel ke sel lainnya. Setiap gerakan diperhitungkan dengan berat yang dihitung sebelumnya.

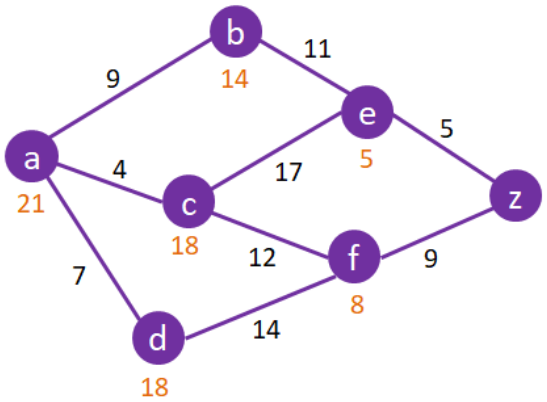
- Hindari Hambatan: Hambatan seperti robot lawan atau batas lapangan tidak dimasukkan sebagai node yang dapat dilalui. Node-node yang berada di antara hambatan juga tidak dihubungkan oleh edge.

5. Menghubungkan titik-titik dan menghindari *obstacle*

   - Tetangga Valid (Valid Neighbors): Untuk setiap node, tetangga yang valid ditentukan sebagai node yang dapat dicapai tanpa melewati hambatan. Periksa apakah ada obstacle di antara node saat ini dan tetangga potensial sebelum menghubungkan mereka.

   - Penghindaran Hambatan (Obstacle Avoidance): Hambatan dinamis seperti pergerakan robot lawan dipantau secara real-time dan digunakan untuk memperbarui node dan edge di grid.

   - Update Graf Dinamis: Setiap perubahan pada posisi hambatan menyebabkan pembaruan graf, memastikan bahwa Algoritma A* selalu bekerja dengan informasi terbaru.



Gambar 2 Contoh Representasi Titik pada Graf

C. Pseudocode

```
function A*(start, goal)
    openSet := priority queue containing start
    cameFrom := empty map

    gScore := map with default value of Infinity
    gScore[start] := 0

    fScore := map with default value of Infinity
    fScore[start] := heuristic(start, goal)

    while openSet is not empty
        current := node in openSet with lowest fScore value
        if current == goal
            return reconstruct_path(cameFrom, current)

        remove current from openSet
        for each neighbor of current
            tentative_gScore := gScore[current] + distance(current, neighbor)

            if tentative_gScore < gScore[neighbor]
                cameFrom[neighbor] := current
                gScore[neighbor] := tentative_gScore
                fScore[neighbor] := gScore[neighbor] + heuristic(neighbor, goal)
                if neighbor not in openSet
                    add neighbor to openSet

    return failure

function reconstruct_path(cameFrom, current)
    total_path := [current]
    while current in cameFrom
        current := cameFrom[current]
        total_path.prepend(current)
    return total_path
```

Berikut adalah penjelasan masing-masing aspek pada pseudocode tersebut.

a) Fungsi A(start, goal)*

1. Inisialisasi openSet:

   openSet adalah priority queue (antrian prioritas) yang menyimpan node-node yang akan dievaluasi. Pada awalnya, openSet hanya berisi node start.

2. Inisialisasi cameFrom:

   cameFrom adalah peta kosong yang melacak asal usul setiap node, yaitu node mana yang datang sebelum node ini dalam jalur yang optimal.

3. Inisialisasi gScore:

   gScore adalah peta yang menyimpan biaya aktual dari node start ke setiap node. Semua nilai awalnya diatur ke Infinity, kecuali untuk node start yang diatur ke 0.

4. Inisialisasi fScore:

   fScore adalah peta yang menyimpan estimasi biaya total dari node start ke node goal melalui node tertentu. Semua nilai awalnya diatur ke Infinity, kecuali untuk node start yang dihitung menggunakan fungsi heuristik terhadap node goal.

5. While Loop (openSet tidak kosong):

   Selama openSet tidak kosong, algoritma akan terus mengevaluasi node-node.

6. Mengambil Node dengan Nilai fScore Terendah:

   current adalah node dalam openSet dengan nilai fScore terendah. Ini adalah node yang paling menjanjikan untuk jalur terpendek.

7. Memeriksa Jika Node Saat Ini adalah Tujuan:

   Jika current adalah node goal, maka jalur ditemukan dan fungsi reconstruct_path dipanggil untuk membangun kembali jalur dari cameFrom.

8. Menghapus Node Saat Ini dari openSet:

   Node current dihapus dari openSet karena sudah dievaluasi.

9. Evaluasi Tetangga:

   Untuk setiap neighbor dari node current, algoritma menghitung tentative_gScore, yaitu biaya sementara dari node start ke neighbor melalui current.

10. Memperbarui Jalur Optimal Jika Ditemukan Jalur yang Lebih Baik:

    Jika tentative_gScore lebih rendah dari gScore yang tercatat untuk neighbor, ini berarti jalur baru ke neighbor lebih baik:

    - cameFrom[neighbor] diperbarui untuk menunjukkan bahwa neighbor datang dari current.
    - gScore[neighbor] diperbarui ke tentative_gScore.
    - fScore[neighbor] dihitung sebagai penjumlahan dari gScore[neighbor] dan heuristik dari neighbor ke goal.
    - Jika neighbor belum ada di openSet, maka neighbor ditambahkan ke openSet.

11. Mengembalikan Kegagalan Jika openSet Kosong dan Tujuan Tidak Tercapai:

    Jika openSet menjadi kosong dan tujuan belum tercapai, algoritma mengembalikan kegagalan.

b) Fungsi reconstruct_path(cameFrom, current)

1. Inisialisasi Jalur:

   total_path adalah daftar yang pada awalnya hanya berisi current.

2. Melacak Kembali Asal Usul Jalur:

   Selama current ada dalam cameFrom, current diperbarui ke node yang datang sebelumnya, dan node ini ditambahkan ke total_path.

3. Mengembalikan Jalur:

   Jalur yang dibangun di total_path dikembalikan sebagai hasil akhir.

IV. HASIL DAN DISKUSI

Setelah eksperimen yang dilakukan penulis, didapatkan hasil sebagai berikut (garis biru muda adalah trajectory planning dari robot 4 ke titik tujuan).

Gambar 3 Hasil Trajectory Planning

Dapat dilihat bahwa titik yang ingin dituju oleh robot 4 terhalang oleh robot 3. Sehingga robot 4 perlu memutar agar tidak menabrak robot 3. Rute yang dihasilkan dengan algoritma A* adalah seperti pada gambar. Dapat dilihat bahwa rute memutari robot 3 dengan bentuk seperti bagian dari oktagon.

Pertama, robot berhasil membangun peta lapangan yang akurat berdasarkan data sensor, memungkinkan identifikasi posisi bola dan robot lain secara real-time. Algoritma A* kemudian diaplikasikan untuk menghitung jalur terpendek dari posisi awal robot ke bola. Hasilnya menunjukkan bahwa Algoritma A* mampu menemukan jalur yang optimal dengan

meminimalkan waktu tempuh, bahkan dalam skenario yang kompleks dengan banyak hambatan dinamis. Dalam hal ini, fungsi heuristik yang dioptimasi memberikan kontribusi besar, dengan estimasi biaya yang lebih akurat sesuai dengan kondisi nyata lapangan, seperti mempertimbangkan kecepatan rata-rata robot dan perubahan posisi bola serta lawan.

Selain itu, algoritma menunjukkan kemampuan yang kuat dalam penghindaran hambatan. Ketika robot lawan bergerak secara dinamis di lapangan, Algoritma A* secara efektif menyesuaikan jalur untuk menghindari tabrakan, memastikan robot tetap berada di jalur optimal menuju bola. Penyesuaian real-time menggunakan kontrol umpan balik terbukti efektif dalam mempertahankan akurasi gerakan robot sesuai dengan jalur yang direncanakan, bahkan ketika kondisi lapangan berubah secara tiba-tiba.

Gambar 4 Node yang mengelilingi robot 3

Gambar tersebut menunjukkan posisi-posisi node yang mengelilingi robot 3. Oktagon yang dibuat memiliki cukup *space* bagi robot untuk menghindari terjadinya *collision* karena representasi titik tidak proporsional dengan ukuran robot, sehingga perlu diberikan space antar titik untuk memastikan jarak antar titik lebih dari ukuran setengah robot.

Dari segi efisiensi, eksperimen menunjukkan bahwa Algoritma A* mampu melakukan perhitungan jalur dengan waktu komputasi yang rendah, memungkinkan robot untuk merespons perubahan kondisi lapangan dengan cepat. Ini sangat penting dalam konteks pertandingan RoboCup di mana keputusan cepat dan adaptasi terhadap situasi yang berubah adalah kunci keberhasilan.

V. KESIMPULAN

Algoritma A* merupakan salah satu metode pencarian jalur terpendek yang efektif dan efisien, khususnya dalam konteks perencanaan rute untuk robot sepakbola beroda dalam kompetisi RoboCup. Algoritma ini menggabungkan kekuatan dari algoritma Dijkstra dengan fungsi heuristik untuk memperkirakan jarak ke tujuan, sehingga memungkinkan pencarian jalur terpendek yang lebih cepat dan lebih optimal.

Dalam penerapannya pada robot sepakbola beroda, A* dimulai dengan menginisialisasi posisi awal robot sebagai titik awal dan menghitung jarak riil sebagai berat sisi. Jarak ini digunakan untuk menghitung biaya aktual (g(n)) dari titik awal ke node saat ini. Selain itu, jarak dari node saat ini ke tujuan digunakan sebagai heuristik (h(n)) untuk memperkirakan biaya terendah ke tujuan. Algoritma ini membangun graf berdasarkan informasi yang ada, memastikan bahwa setiap node terhubung kecuali terdapat hambatan di antara node-node tersebut.

Proses pencarian melibatkan pemilihan node dengan nilai f(n) terendah dari antrian prioritas, dimana f(n) adalah penjumlahan dari g(n) dan h(n). Jika node yang dipilih adalah tujuan, algoritma akan membangun kembali jalur optimal dari node tersebut ke titik awal. Jika tidak, algoritma terus mengevaluasi tetangga dari node saat ini, memperbarui nilai g(n) dan f(n) serta menambahkannya ke dalam antrian prioritas jika jalur yang lebih baik ditemukan.

Eksperimen dengan Algoritma A* menunjukkan bahwa algoritma ini mampu menemukan jalur terpendek secara efektif dalam lingkungan dinamis seperti lapangan RoboCup, dengan memperhatikan hambatan statis maupun dinamis. Implementasi yang menggunakan grid 2D sebagai representasi lapangan memungkinkan perhitungan yang mudah dan cepat. Dengan mempertimbangkan semua faktor ini, Algoritma A* terbukti sebagai alat yang sangat berguna untuk perencanaan rute dalam aplikasi robotika, khususnya untuk robot sepakbola beroda dalam kompetisi RoboCup.

UCAPAN TERIMA KASIH

Ucapan terima kasih penulis berikan kepada Tuhan Yang Maha Esa atas berkah-Nya yang telah memungkinkan penulisan makalah ini dapat diselesaikan dengan sempurna dan sesuai waktu yang diinginkan. Penulis juga ingin mengucapkan terima kasih yang sebesar-besarnya kepada para dosen pengampu Mata Kuliah Strategi Algoritma, khususnya Bapak Monterico Adrian, S.T., M.T. dan Bapak Ir. Rila Mandala, M.Eng., Ph.D, yang telah memberikan bimbingan, ilmu, dan inspirasi selama proses belajar. Selain itu, penulis juga ingin menyampaikan rasa terima kasih kepada asisten mata kuliah, teman-teman seangkatan, serta keluarga penulis yang telah memberikan bantuan dan dukungan yang begitu luar biasa selama proses berjalannya kuliah dan pembuatan makalah ini.

Penulis ingin berterima kasih juga kepada tim KRSBI-B ITB yaitu Dagozilla, yang menjadi sumber ide makalah ini

dan juga membantu dalam eksperimen yang penulis lakukan, makalah ini tidak akan ada tanpa mereka.

Penulis juga ingin mengucapkan terima kasih kepada semua sumber referensi dan literatur yang telah menjadi landasan utama dalam penulisan makalah ini. Setiap kontribusi dari sumber-sumber tersebut sangat berharga dalam memperkaya isi dan kualitas penelitian ini. Akhir kata, penulis berharap makalah ini dapat memberikan manfaat dan kontribusi positif bagi pengembangan ilmu pengetahuan, khususnya dalam bidang optimasi rute dan algoritma pencarian.

REFERENSI

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use G. Eason

[1] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] 101Computing, “A* Search Algorithm.” Diakses dari https://www.101computing.net/a-star-search-algorithm/ , pada 12 Juni 2024.

[3] R. Munir, "Penentuan Rute (Route/Path Planning) (Bag.1)." Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/RoutePlanning-Bagian1-2021.pdf , pada 11 Juni 2024.

[4] R. Munir, "Penentuan Rute (Route/Path Planning) (Bag.2)." Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/RoutePlanning-Bagian2-2021.pdf , pada 11 Juni 2024.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

Owen Tobias Sinurat
13522131