PENERAPAN ALGORITMA A* DALAM NAVIGASI PERGERAKAN ROBOT OTONOM

Muhammad Althariq Fairuz - 13522027
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): althariqfairuz273@gmail.com

***Abstract*— Robot otonom adalah sistem robotik yang mampu bergerak dan melakukan serangkaian tugas tanpa intervensi manusia secara langsung. Teknologi ini sering diuji dalam kompetisi robot otonom, robot-robot berlomba untuk menyelesaikan tugas-tugas kompleks dalam lingkungan yang dinamis. Kompetisi ini tidak hanya mengukur kecepatan dan ketepatan robot, tetapi juga kemampuan mereka dalam menavigasi rute yang optimal. Salah satu algoritma navigasi yang sering digunakan dalam robot otonom adalah algoritma A* (A-star). Algoritma A* (A-star) adalah algoritma yang menggabungkan kelebihan dari algoritma Uniform-Cost Search (UCS) dan algoritma pencarian Greedy Best-First Search (GBFS) dengan menggunakan fungsi heuristik untuk menentukan jalur terpendek yang optimal dari titik awal ke tujuan.**

Keywords—Robot Otonom, Navigasi, Algoritma A*, Heuristik, Optimal.

I. PENDAHULUAN

Kemajuan teknologi di era digital telah memberikan dampak yang signifikan terhadap berbagai aspek kehidupan manusia. Salah satu perkembangan yang menarik perhatian adalah kemajuan dalam bidang robotika, khususnya robot otonom. Robot otonom adalah sistem robotik yang mampu bergerak dan melakukan tugas tanpa intervensi manusia secara langsung. Robot-robot ini semakin banyak digunakan dalam berbagai aplikasi, mulai dari industri manufaktur hingga eksplorasi ruang angkasa, serta dalam kompetisi robot otonom yang menantang kemampuan mereka dalam navigasi dan penyelesaian tugas-tugas kompleks.

Gambar 1.1 Perlombaan Kontes Robot Indonesia (KRI)

(Sumber: KRI 2023 - Kontes Robot Indonesia)

Kompetisi robot otonom sering kali menguji kemampuan robot dalam menavigasi lingkungan yang dinamis dan tidak terstruktur. Tantangan utama dalam kompetisi ini adalah bagaimana robot dapat menemukan jalur terpendek dan paling efisien untuk mencapai tujuan mereka, menghindari rintangan, dan beradaptasi dengan perubahan lingkungan secara real-time.

Gambar 1.2 Robot pencari korban gempa

(Sumber: Robot Pencari Korban Gempa di Jepang)

Selain kompetisi, robot otonom juga digunakan untuk mencari korban di daerah bencana alam, seperti gempa bumi, banjir, atau kebakaran hutan. Mereka dapat menjelajahi reruntuhan atau area berbahaya yang sulit dijangkau oleh manusia, mencari tanda-tanda kehidupan, dan memberikan bantuan pertama.

Untuk mencapai tujuan ini, robot harus dilengkapi dengan algoritma navigasi yang efektif. Salah satu algoritma navigasi yang paling efektif dan populer adalah algoritma A* (A-star). Algoritma A* adalah algoritma pencarian jalur yang menggabungkan kelebihan dari algoritma Uniform-Cost Search dan algoritma pencarian Greedy Best-First Search. Dengan menggabungkan kelebihan dari dua algoritma ini, Algoritma A* dapat menentukan jalur terpendek yang optimal dari titik awal ke tujuan dengan efisiensi yang tinggi. Pada robot otonom, penerapan algoritma A* sangat penting karena memungkinkan robot untuk melakukan navigasi yang efisien dalam lingkungan yang bisa diprediksi.

II. TEORI DASAR

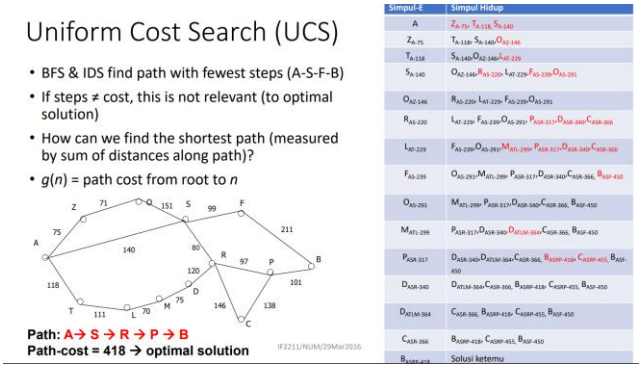
Pencarian rute adalah proses mengidentifikasi langkah-langkah atau urutan tindakan yang diperlukan untuk mencapai tujuan tertentu. Ada dua metode yang umum digunakan, yaitu *informed search* dan *uninformed search*.

Pencarian *informed search* menggunakan informasi tambahan tentang masalah atau rute yang tersedia untuk memandu proses pencarian. Algoritma yang masuk dalam kategori *informed search* menggunakan nilai heuristik atau fungsi evaluasi yang memperkirakan biaya atau sisa jarak ke tujuan. Heuristik ini memberikan petunjuk tentang arah yang lebih efisien untuk mencapai suatu solusi. Contoh algoritmanya adalah A* (A-star) dan Greedy Best-First Search (GBFS). Algoritma A* menggunakan fungsi evaluasi (f(n)) yang menggabungkan biaya langkah yang telah diambil (g(n)) dengan perkiraan sisa biaya (h(n)) untuk mencapai tujuan.

Uninformed search, juga dikenal sebagai pencarian buta, tidak menggunakan informasi tambahan tentang masalah atau ruang pencarian. Algoritma ini hanya mengandalkan struktur ruang pencarian yang tersedia untuk mengeksplorasi semua langkah yang mungkin dilakukan secara sistematis. Contoh algoritmanya adalah breadth-first search (BFS), depth-first (DFS), dan uniform-cost search (UCS). BFS berfokus pada menjelajahi semua simpul pada kedalaman yang sama terlebih dahulu, sedangkan DFS menggunakan pendekatan yang berbeda, yaitu menjelajahi sebanyak mungkin simpul pada satu rute sebelum pindah ke rute berikutnya, dan UCS memperhitungkan biaya langkah-langkah saat menjelajahi ruang pencarian dan mengekspan simpul dengan biaya terendah pada simpul hidup yang tersedia.

A. Algoritma Uniform-Cost Search (UCS)

Uniform Cost Search (UCS) adalah algoritma pencarian yang mirip dengan Breadth-First Search (BFS), tetapi dengan perbedaan UCS mempertimbangkan biaya jalur (path cost) saat menentukan simpul mana yang akan dieksplorasi berikutnya. UCS selalu memperluas simpul dengan biaya kumulatif terendah dari simpul awal dengan menggunakan *priority queue* [1].

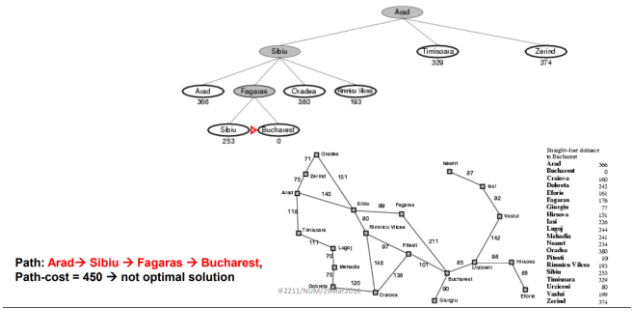


Gambar 2.1 Contoh penggunaan Algoritma UCS

(Sumber: Route Planning - Bagian 1)

B. Algoritma Greedy Best-First Search (GBFS)

Greedy Best-First Search (GBFS) adalah algoritma pencarian yang memprioritaskan eksplorasi simpul berdasarkan nilai heuristik yang menunjukkan seberapa dekat simpul tersebut dengan tujuan. Simpul dieksplorasi sesuai dengan aturan *priority queue*, yaitu simpul dengan nilai heuristik terendah memiliki prioritas tertinggi. Algoritma ini tidak selalu menghasilkan solusi yang optimal [1].

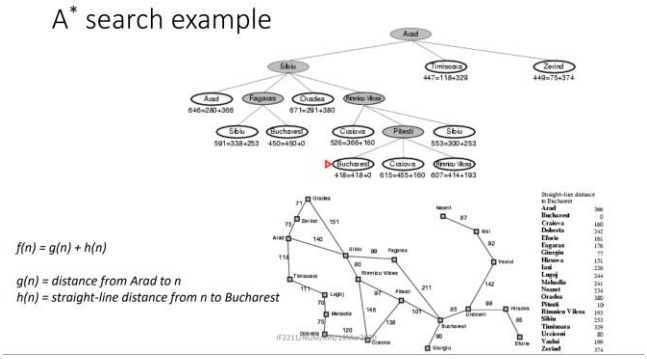


Gambar 2.2 Contoh penggunaan Algoritma GBFS

(Sumber: Route Planning - Bagian 1)

C. Algoritma A* (A-Star)

A* adalah algoritma pencarian yang menggabungkan pendekatan UCS dan GBFS dengan menggunakan fungsi biaya total f(n) = g(n) + h(n), dengan g(n) adalah biaya kumulatif dari simpul awal ke simpul n, dan h(n) adalah nilai heuristik yang memperkirakan biaya dari simpul n ke tujuan [2].



Gambar 2.3 Contoh penggunaan Algoritma A*

(Sumber: Route Planning - Bagian 2)

D. Fungsi Heuristik

Heuristik berasal dari kata Yunani "heuriskein" yang berarti mencari atau menemukan. Dalam dunia pemrograman, heuristik adalah pendekatan yang digunakan dalam pengembangan algoritma atau teknik pemecahan masalah berdasarkan uji coba atau aturan praktis lainnya. Metode-metode ini bertujuan untuk menemukan solusi yang efisien dan efektif walaupun tidak selalu sempurna atau optimal. Pendekatan ini sangat berguna untuk masalah-masalah yang terlalu kompleks untuk dipecahkan menggunakan algoritma deterministik tradisional atau ketika solusi yang tepat tidak praktis karena batasan waktu [3].

Fungsi Heuristik haruslah *admissible*, artinya adalah nilai heuristik tersebut tidak melebih-lebihkan biaya untuk mencapai tujuan dari simpul mana pun dalam graf. Dengan kata lain, estimasi biaya yang diberikan oleh fungsi heuristik h(n) selalu kurang dari atau sama dengan biaya aktual terendah untuk mencapai tujuan.

E. Robot Otonom

Robot otonom pada dasarnya adalah sebuah mesin yang dilengkapi dengan sensor, prosesor, dan aktuator sehingga membuatnya mampu untuk melihat sekelilingnya, memproses informasi, dan mengambil tindakan tanpa campur tangan manusia [4].

Proses navigasi merupakan salah satu aspek penting pada robot. Navigasi merujuk pada kemampuan robot untuk bergerak dari satu lokasi ke lokasi lainnya dengan aman dan efisien, menghindari rintangan, dan mencapai tujuan yang ditetapkan.

Beberapa fitur umum dari perencanaamn navigasi robot adalah [5]:

1. Memberikan Jalur dari Posisi Awal ke Posisi Tujuan

   Algoritma navigasi robot harus mampu memberikan serangkaian langkah atau posisi yang harus diambil oleh robot untuk mencapai tujuan dari posisi awalnya.

2. Jalur yang Bebas dari Rintangan

   Jalur yang dihasilkan harus bebas dengan rintangan yang ada di lingkungan sekitarnya. Hal ini memastikan bahwa robot dapat bergerak dengan aman tanpa menabrak apapun di sepanjang jalurnya.

3. Optimasi Heuristik

   Biasanya, jalur yang dihasilkan oleh algoritma harus dioptimalkan sesuai dengan heuristik tertentu. Misalnya, jarak tempuh yang minimal atau waktu yang minimal untuk mencapai tujuan.

4. Jalur yang Dijelajahi oleh Robot

   Jalur yang dihasilkan harus dapat dilalui oleh robot, memperhitungkan dinamikanya seperti kecepatan maksimum dan kemampuan lainnya.

Gambar 2.4 Robot pengantar pesanan

(Sumber: Self-driving robots will start making deliveries in Europe)

Algoritma navigasi seperti A* yang telah dibuat akan dihubungkan ke robot otonom dengan melibatkan integrasi algoritma ke dalam perangkat lunak kontrol robot. Algoritma tersebut diimplementasikan dalam sistem kontrol robot sehingga robot dapat menggunakan informasi dari algoritma untuk merencanakan jalur dan mengambil keputusan navigasi saat bergerak.

Misalnya, dalam robot otonom yang bergerak di sebuah grid, algoritma A* dapat diimplementasikan dalam perangkat lunak kontrol robot untuk merencanakan jalur dari lokasi saat ini ke tujuan. Robot akan menggunakan informasi dari algoritma A* untuk menavigasi melalui lingkungan, menghindari rintangan, dan mencapai tujuan yang ditetapkan.

III. ANALISIS DAN PEMBAHASAN

A. Visualisasi Denah Penjelajahan Robot

Untuk memvisualisasikan denah penjelajahan robot beserta rintangannya, akan digunakan Bahasa Pemrograman Python dengan bantuan *library* matplotlib untuk memvisualisasikan denah penjelajahan, gerakan, serta rute yang telah ditempuh robot selama perjalanannya.

AStar.py

```python
# Generate arena with obstacles (1) and free space (0), obstacle probability is 0.2 by default
def generate_arena(rows: int, cols: int, obstacle_prob: float = 0.2) -> np.ndarray:
    arena: np.ndarray = np.zeros((rows, cols), dtype=int)
    for i in range(rows):
        for j in range(cols):
            if np.random.rand() < obstacle_prob:
                arena[i, j] = 1  # Mark as obstacle
    return arena

```

Snipped

Gambar 3.1 Impelementasi pembuatan denah robot

(Sumber: Dokumentasi penulis)

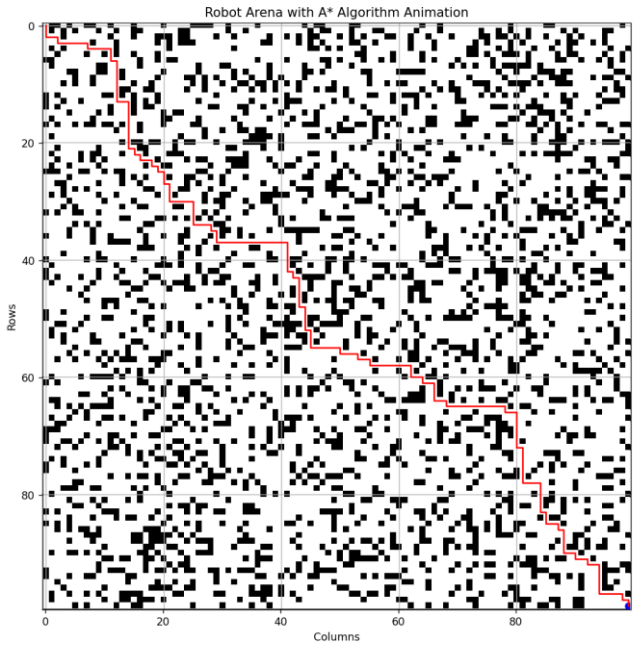
AStar.py

```python
# Animate the path found by A* search
def animate_path(arena: np.ndarray, path: list[tuple[int, int]]):
    fig, ax = plt.subplots(figsize=(10, 10))
    ax.imshow(arena, cmap='Greys', origin='upper')

    robot, = ax.plot([], [], 'bo')  # Robot position
    trail, = ax.plot([], [], 'r-')  # Robot trail

    def init():
        robot.set_data([], [])
        trail.set_data([], [])
        return robot, trail

    def update(frame):
        x_data, y_data = trail.get_data()
        x_data = np.append(x_data, frame[1])
        y_data = np.append(y_data, frame[0])
        robot.set_data(frame[1], frame[0])
        trail.set_data(x_data, y_data)
        return robot, trail

    ani = animation.FuncAnimation(
        fig, update, frames=path, init_func=init, blit=True, repeat=False)

    plt.title('Robot Arena with A* Algorithm Animation')
    plt.xlabel('Columns')
    plt.ylabel('Rows')
    plt.grid(True)
    plt.show()
```

Snipped

Gambar 3.2 Impelementasi visualisasi denah robot

(Sumber: Dokumentasi penulis)



Gambar 3.3 Visualisasi denah penjelajahan robot

(Sumber: Dokumentasi penulis)

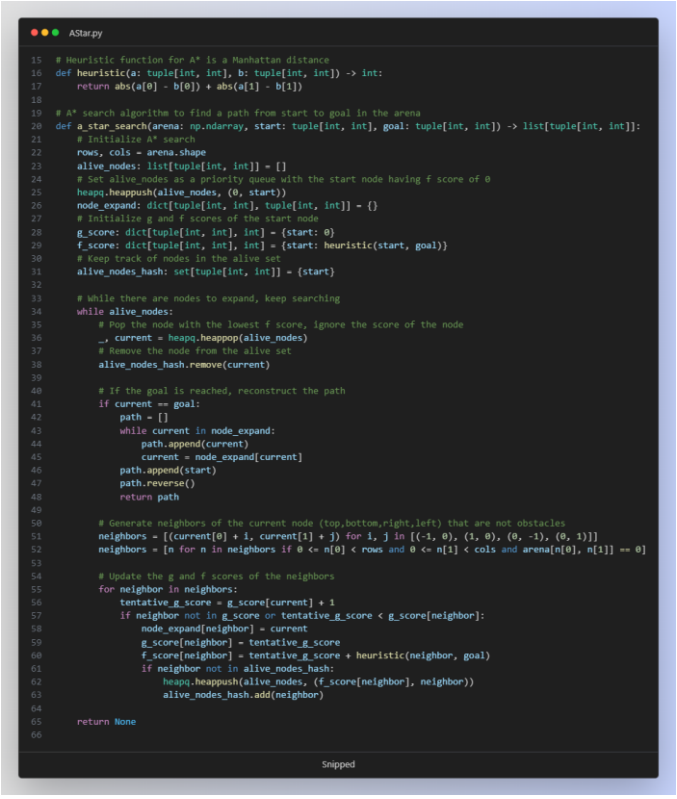
Denah yang dibangkitkan merupakan denah dengan ukuran 50x50, ukuran denah bisa diganti sesuai dengan keinginan. Rintangan pada denah ditandai dengan daerah yang diwarnai dengan warna hitam, rintangan dibuat dengan probabilitas kemunculan 0.2, probabilitas kemunculan rintangan nantinya bisa diatur sesuai keinginan. Lintasan terpendek yang mungkin ditempuh oleh robot ditandai dengan garis berwarna merah dan posisi robot ditandai dengan titik berwarna biru. Perhatikan bahwa ada kemungkinan rute tidak ditemukan karena rintangan yang dihasilkan acak/tidak *fixed*, menyesuaikan dengan kondisi di dunia nyata.

B. Penerapan Algoritma A* dalam Mencari Rute Terpendek

Algoritma A* digunakan untuk menentukan rute terpendek dari suatu tempat ke tempat lainnya. Algoritma A* akan menghasilkan solusi yang optimum. Hal tersebut karena dalam prosesnya, algoritma A* akan mengevaluasi beberapa jalur yang mungkin dilewati dan akan menghasikan solusi dengan rute terpendek yang dihasilkan dari penjumlahan biaya kumulatif dari simpul awal ke tujuan dengan fungsi heuristiknya. Rumus perhitungan Algoritma A* adalah:

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

Dengan g(n) merupakan biaya untuk mencapai titik saat ini dari titik awal. Dalam implementasi ini, setiap langkah ke tetangga (atas, bawah, kiri, kanan) memiliki biaya tetap sebesar 1, sedangkan h(n) adalah nilai heuristik. Heuristik yang digunakan adalah Manhattan Distance antara titik saat ini dengan titik tujuan.



Gambar 3.3 Impelementasi nilai heuristik Manhattan Distance dan Algoritma A*

(Sumber: Dokumentasi penulis)

Langkah-langkah penyelesaiannya adalah sebagai berikut:

1. Inisialisasi

   Mulai dengan simpul awal (start) dan simpul tujuan (goal).

2. Inisialisasi Nilai G dan F

   Tentukan nilai g untuk simpul awal, yaitu biaya dari start ke simpul saat ini. Biasanya nilai g untuk simpul awal adalah 0. Tentukan juga nilai f untuk simpul awal, yaitu jumlah dari g dan nilai heuristik sehingga f = g + h.

3. Ekspansi Simpul

   Mulai dengan simpul awal dan tambahkan ke simpul hidup. Ambil simpul dengan nilai f terkecil dari simpul hidup sebagai simpul saat ini. Periksa apakah simpul saat ini adalah simpul tujuan. Jika iya, jalur telah ditemukan dan proses selesai.

4. Evaluasi Tetangga

   Untuk setiap tetangga dari simpul saat ini, hitung nilai g baru. Jika nilai g baru lebih kecil dari nilai g saat ini dari tetangga tersebut, update nilai g, f, dan tambahkan tetangga tersebut ke simpul hidup. Simpan simpul saat ini ke dalam daftar yang telah

dievaluasi/simpul yang telah diekspan agar tidak dievaluasi lagi.

5. Ulangi

   Ulangi proses dari langkah 3 sampai tidak ada simpul hidup yang tersisa atau simpul tujuan telah ditemukan. Jika simpul hidup kosong, berarti tidak ada jalur yang memungkinkan dari start ke goal, dan pencarian dihentikan.

6. Rekonstruksi Jalur

   Jika jalur telah ditemukan, rekonstruksi jalur dari simpul tujuan ke simpul awal dengan mengikuti kembali arah dari setiap simpul.

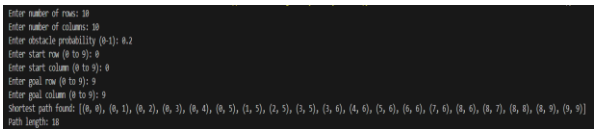
7. Jalur yang dihasilkan merupakan jalur terpendek dari titik awal ke tujuan.

IV. PENGUJIAN DAN EVALUASI

Pengujian dilakukan dengan menguji program dengan beberapa kasus uji sederhana. Hasil yang diharapkan adalah program mampu memberikan lintasan terpendek yang mungkin dalam suatu denah. Hasil pengujian, seperti lintasan yang dilewati dan jumlah lintasan yang dilewati, akan ditampilkan pada terminal.

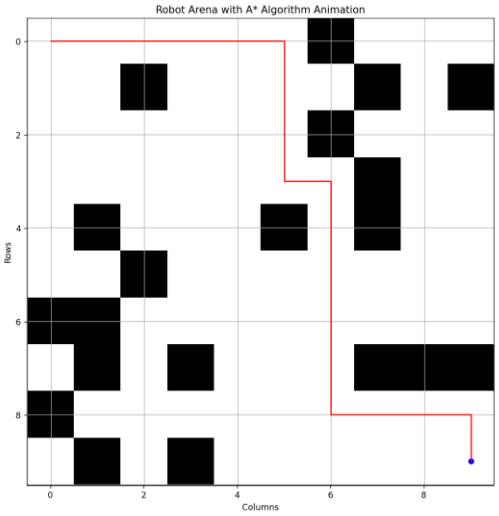
A. Pengujian

1. Pengujian pertama



Gambar 4.1 Hasil pengujian pertama

(Sumber: Dokumentasi penulis)



Gambar 4.2 Visualisasi pengujian pertama

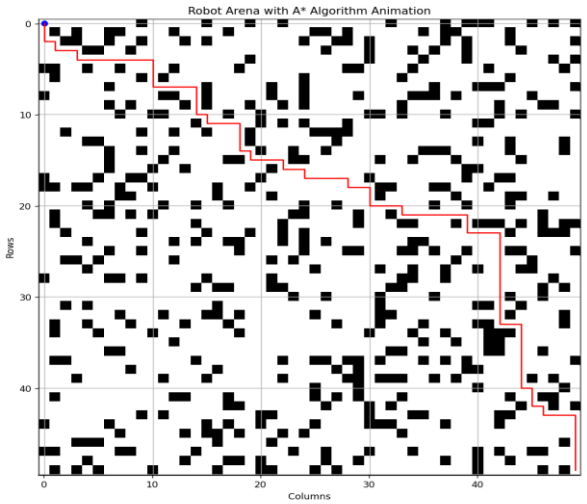
(Sumber: Dokumentasi penulis)

Dari hasil pengujian pertama, terlihat bahwa program mampu menemukan lintasan terpendek yang mungkin tanpa menabrak rintangan yang ditandai dengan daerah berwarna hitam.

2. Pengujian kedua

Gambar 4.3 Hasil pengujian kedua

(Sumber: Dokumentasi penulis)



Gambar 4.4 Visualisasi pengujian kedua

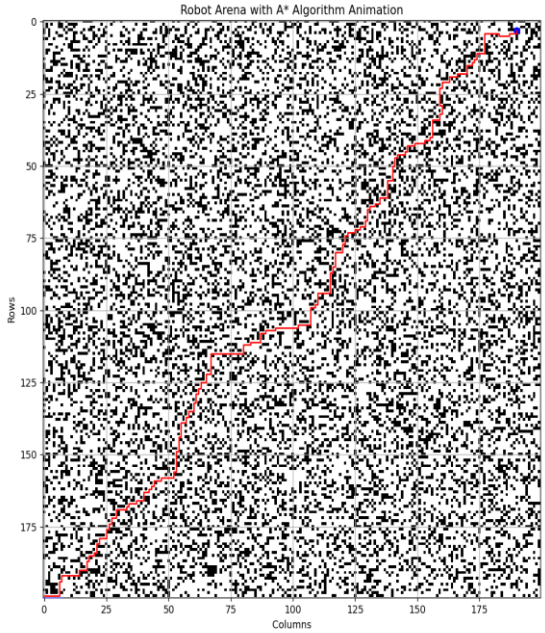
(Sumber: Dokumentasi penulis)

Dari hasil pengujian kedua, program juga mampu menghasilkan lintasan minimum pada denah yang sedikit lebih ekstrim dari pengujian sebelumnya.

3. Pengujian ketiga

Gambar 4.5 Hasil pengujian ketiga

(Sumber: Dokumentasi penulis)



Gambar 4.6 Visualisasi pengujian ketiga

(Sumber: Dokumentasi penulis)

Perhatikan bahwa pada pengujian ketiga, rintangan yang dihasilkan jauh lebih banyak karena probabilitasnya dinaikkan sebanyak 10% dan juga ukuran denah diperbesar menjadi 200x200 dengan titik awal pada baris 199 kolom 0 dan titik tujuan pada baris 3 kolom 190. Program masih mampu menghasilkan rute terpendek terlepas dari ketidakpastian denah yang akan dilaluinya.

B. Evaluasi

Terdapat beberapa evaluasi untuk kasus pengujian, terutama untuk pengujian pertama karena lintasan yang dihasilkan masih terlalu sederhana sehingga algoritma dapat menyelesaikannya dengan sangat cepat dan tidak bisa dijadikan patokan apakah algoritma sudah benar atau belum. Pada kasus ketiga, proses visualisasi sedikit memakan waktu karena rute yang ditempuh cukup panjang.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Dari implementasi algoritma A* pada kode di atas, dapat disimpulkan bahwa Algoritma A* mampu menemukan jalur terpendek dari titik awal ke titik tujuan dalam sebuah denah yang berisi rintangan. Beberapa poin penting dari hasil implementasi ini adalah:

1. Efektivitas Algoritma A*

   Algoritma A* terbukti efektif dalam mencari jalur terpendek dengan menggunakan kombinasi dari g-score (biaya dari awal ke titik saat ini) dan heuristik (perkiraan biaya dari titik saat ini ke tujuan).

2. Visualisasi Jalur

   Dengan menggunakan matplotlib, proses pencarian serta jalur yang ditemukan oleh algoritma dapat divisualisasikan sehingga memudahkan dalam memahami hasil dari pencarian jalur.

3. Penanganan Rintangan

   Denah yang menghasilkan rintangan secara acak membuat pengujian algoritma dalam berbagai skenario menjadi akurat, memastikan bahwa algoritma *robust* terhadap berbagai konfigurasi rintangan.

B. Saran

Untuk memperbaiki dan memperluas implementasi algoritma ini, beberapa saran yang dapat dipertimbangkan adalah:

1. Implementasi Algoritma Lain

   Membandingkan hasil A* dengan algoritma pencarian jalur lainnya, seperti Dijkstra atau Greedy Best-First Search untuk memahami kelebihan dan kekurangan masing-masing algoritma.

2. Optimasi Kode

   Memastikan kode sudah optimal dalam hal kecepatan dan efisiensi memori, terutama jika digunakan untuk denah yang lebih besar atau dalam *real-time*.

3. GUI untuk Pengguna

   Mengembangkan antarmuka pengguna grafis (GUI) yang memungkinkan pengguna untuk menggambar arena mereka sendiri, menempatkan titk awal dan tujuan, serta melihat hasil pencarian jalur secara interaktif.

VI. UCAPAN TERIMA KASIH

Puji dan syukur saya panjatkan kepada Tuhan Yang Maha Esa karena atas berkat rahmat dan karunia-Nya sehingga saya dapat menyelesaikan penulisan makalah ini dan implementasi kode program untuk penyelesaian makalah ini. Ucapan terima kasih saya sampaikan kepada keluarga saya serta kepada dosen pengampu kelas 01 mata kuliah Strategi Algoritma tahun 2024, yaitu Bapak Rinaldi Munir, atas pengajarannya selama satu semester ini sehingga saya dapat menyelesaikan pembuatan makalah ini dengan baik.

LINK REPOSITORI

AStar-Algorithm-For-Autonomous-Robot/AStar.py at main · AlthariqFairuz/AStar-Algorithm-For-Autonomous-Robot (github.com)

LINK YOUTUBE

Pemanfaatan Algoritma A* dalam Navigasi Robot Otonom

REFRENSI

[1] Munir, Rinaldi. "Penentuan Rute (Route Planning) Bagian 1." Institut Teknologi Bandung, 2020-2021.
Penetuan Rute (Route Planning) Bagian 1. Diakses pada 10 Juni 2024.

[2] Munir, Rinaldi. "Penentuan Rute (Route Planning) Bagian 2." Institut Teknologi Bandung, 2020-2021.
Penetuan Rute (Route Planning) Bagian 2. Diakses pada 10 Juni 2024.

[3] DevX. “Heuristic Programming”.
Heuristic Programming - Glossary (devx.com). Diakses pada 10 Juni 2024.

[4] Andre, Dave. “What is an Autonomous Robot?” .
What Is an Autonomous Robot?. Diakses pada 10 Juni 2024.

[5] ERC Handbook. ”Path Planning in Robotics”.
Introduction to Path Planning in Robotics - ERC Handbook (erc-bpgc.github.io). Diakses pada 10 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

Muhamamd Althariq Fairuz
13522027