https://jurnal.pustakagalerimandiri.co.id/index.php/pustakaai
DOI : https://doi.org/10.55382/jurnalpustakaai.v6i1.1619

JURNAL PUSTAKA AI
JURNAL PUSAT AKSES KAJIAN TEKNOLOGI ARTIFICIAL INTELLIGENCE
Pustaka Galeri Mandiri

Vol. 6 No. 1 (2026) 134 – 152
E ISSN: 2809-4069

Simulasi Navigasi Robot Labirin Menggunakan Algoritma Depth-First Search pada Platform Webots

Muhammad Fajri[1], Putri Sakinah[2]
[12]Program Studi Informatika, Universitas Adzkia
[1]mfajri.tech@gmail.com [2]putrisakinah@adzkia.ac.id

Abstract

Autonomous Mobile Robots (AMRs) are a branch of robotics that enable robots to move and navigate autonomously without direct human intervention. One of the important components in the development of such robots is path planning that refers to the robot's ability to determine its route. This study aims to evaluate the capability of the Depth-First Search (DFS) algorithm in assisting a robot to navigate and solve a maze environment without prior mapping. The DFS algorithm is implemented on an e-puck robot using the Webots simulator. The robot is tested to find the goal point in five different maze scenarios. The simulation results show that the DFS algorithm is able to guide the robot to successfully reach the goal in all tested maze environments. In its implementation, the DFS algorithm applies two different exploration strategies. They are right-priority and left-priority strategies. Based on the experimental results from the five tested scenarios, the right-priority strategy achieves a faster overall completion time with a difference of 610.78 seconds compare to the left-priority strategy.

Keywords: Depth-First Search, Path Planning, Autonomous Mobile Robot, Maze Robot, Webots.

Abstrak

Autonomous Mobile Robot (AMR) merupakan cabang robotika yang memungkinkan robot berjalan dan menavigasi diri tanpa intervensi secara langsung dari manusia. Salah satu komponen penting dalam pengembangan robot ini ialah *path planning*, yaitu kemampuan robot dalam menentukan jalur yang akan dilaluinya. Penelitian ini bertujuan untuk menguji kemampuan algoritma *Depth-First Search* (DFS) dalam membantu robot menavigasi diri menyelesaikan arena labirin tanpa pemetaan awal. Algoritma DFS diimplementasikan dalam robot e-puck menggunakan simulator Webots. Robot akan diuji untuk menemukan titik akhir dari lima skenario arena labirin yang berbeda. Hasil simulasi menunjukkan bahwa algoritma DFS mampu membantu robot dalam menemukan titik akhir seluruh arena labirin yang diuji. Dalam implementasinya, algoritma DFS dapat menggunakan dua strategi penelusuran yang berbeda, yaitu strategi prioritas kanan dan prioritas kiri. Berdasarkan hasil pengujian dari kelima skenario yang digunakan, strategi prioritas kanan unggul 610,78 detik lebih cepat dalam menyelesaikan seluruh labirin dibandingkan strategi prioritas kiri.

Kata kunci: *Depth-First Search*, *Path Planning*, *Autonomous Mobile Robot*, Robot Labirin, Webots.

Submitted : 20-03-2026 | Reviewed : 25-03-2026 | Accepted : 30-03-2026

1. Pendahuluan

Perkembangan teknologi robotika saat ini semakin pesat dan telah digunakan dalam berbagai bidang, salah satunya di bidang industri. Kemajuan ini tak hanya membuat robot menjadi pilihan yang efisien, tetapi juga murah dan aman untuk digunakan. Hal ini membuat banyak perusahaan mulai beralih dari tenaga manusia ke otomatisasi berbasis robot untuk meningkatkan daya produksi dan menekan biaya operasional [1]. Meskipun tidak sepenuhnya dapat menggantikan peran menusia, robot mampu menyelesaikan tugas secara berulang dengan waktu yang lebih singkat dan efisien [2].

Salah satu jenis robot yang banyak digunakan di bidang industri pada saat ini adalah *Autonomous Mobile Robot* (AMR). AMR merupakan robot cerdas yang mampu bergerak dan melakukan tugasnya secara mandiri tanpa intervensi secara langsung dari manusia [3]. Sebelum digunakan, AMR telah diprogram terlebih dahulu, yang biasanya melibatkan proses pemetaan lingkungan untuk mengembangkan pola navigasi. Robot ini juga dilengkapi dengan sensor dan kecerdasan buatan untuk membantu proses navigasi dan menghindari rintangan [4]. Kemampuannya dalam beradaptasi pada lingkungan inilah yang membuat AMR dapat digunakan untuk melakukan pekerjaan yang membutuhkan konsistensi tinggi dan berulang, seperti memindahkan barang, mengangkut material dari satu titik ke titik lain, serta melakukan proses distribusi di area industri [5]. Tantangan besar dalam membangun sistem AMR adalah memahami lingkungan yang kompleks dan tidak terstruktur [6]. Kemampuan navigasi pada AMR menjadi aspek penting untuk memastikan robot bergerak dan melakukan pekerjaannya dengan aman dan efisien di lingkungan kerja yang kompleks.

Salah satu komponen penting dalam sistem navigasi AMR adalah perencanaan jalur (*path planning*), yaitu proses menentukan jalur yang akan dilalui oleh robot untuk mencapai tujuan [7]. Dalam memetakan jalur yang akan dilalui, *path planning* tidak hanya mempertimbangkan aspek jarak tempuh terpendek, tetapi juga harus mempertimbangkan aspek keamanan, efisiensi energi, keterbatasan medan, dan dinamika lingkungan sekitar [8]. Hal inilah yang menjadikan *path planning* sebagai tolak ukur utama dalam menilai kecerdasan sistem robot otonom.

Permasalahan *path planning* dapat dimodelkan sebagai permasalahan berbasis graf, yang telah banyak dikaji dalam berbagai konteks. Salah satu penelitian membahas tentang simulasi pengiriman air mineral galon dengan multi-depot menggunakan algoritma *Hill Climbing* dan A* [9]. Penelitian tersebut menunjukkan bahwa algoritma pencarian dapat digunakan secara efektif dalam menentukan rute terbaik berdasarkan rute terpendek dan waktu tercepat. Namun, penelitian ini bersifat simulasi komputasional dan tidak melibatkan robot fisik maupun proses eksplorasi lingkungan di dalamnya.

Dalam konteks penelitian ini, permasalahan *path planning* dapat dianalogikan dengan kasus pencarian titik akhir pada labirin, di mana robot harus mengeksplorasi labirin untuk menemukan titik akhir yang tepat. Hal serupa telah dilakukan oleh Adryady dkk. dalam penelitiannya yang berjudul "Implementasi Algoritma *Path Planning* dan *Mapping* Arena pada *Mobile Robot*", di mana *path planning* dapat membantu robot dalam menemukan jalur terpendek dan tercepat untuk mencapai tujuannya [10]. Penelitian yang semisal juga dilakukan oleh Sabril dkk. dengan judul "Perbandingan Waktu Tempuh *Mobile Robot* dalam Arena Labirin dengan Algoritma Tangan Kiri dan Algoritma Tangan Kanan." Penelitian ini menunjukkan bahwa algoritma tangan kanan lebih unggul dibandingkan algoritma tangan kiri dari segi waktu tempuh secara keseluruhan [11]. Namun, pada kedua penelitian tersebut, *mapping* arena labirin telah diberikan kepada robot secara rinci terlebih dahulu, sehingga robot dapat melakukan perhitungan dan analisis jalur sebelum mulai bergerak. Proses ini berbeda dengan penelitian yang dilakukan saat ini, di mana robot akan melakukan eksplorasi secara mandiri untuk menemukan jalur yang tepat untuk mencapai tujuannya.

Pada kasus eksplorasi jalur tanpa *mapping* arena, proses navigasi robot sangat bergantung pada algoritma eksplorasi yang digunakan. Terdapat dua algoritma yang sangat fundamental dalam *graph theory* dan *computer science*, yaitu *Breadth-First Search* (BFS) dan *Depth-First Search* (DFS) [12]. Beberapa penelitian menunjukkan bahwa BFS dan DFS memiliki karakteristik yang berbeda dalam berbagai aplikasi. Pada penelitian di bidang web *crawler*, BFS mampu melakukan eksplorasi data yang lebih luas dan mendalam, namun memerlukan waktu eksekusi yang lebih panjang dibandingkan DFS yang lebih cepat dengan jumlah data yang lebih terbatas [13]. Temuan serupa juga diperoleh pada studi navigasi berbasis graf di lingkungan kampus, di mana DFS memiliki keunggulan dalam kecepatan eksekusi dan penggunaan memori, sementara BFS lebih konsisten dalam menemukan solusi optimal [14]. Hal ini menunjukkan bahwa tidak ada algoritma yang memiliki keunggulan secara mutlak, melainkan pemilihan algoritma harus disesuaikan dengan kebutuhan dan karakteristik permasalahan yang akan diselesaikan.

Dalam konteks navigasi robot labirin tanpa pemetaan awal, kecepatan eksplorasi dan kemampuan menelusuri jalur secara bertahap menjadi aspek penting. Oleh karena itu, penelitian ini menggunakan algoritma DFS sebagai metode pencarian jalur yang akan membantu robot untuk menemukan titik akhir labirin.

Submitted : 20-03-2026 | Reviewed : 25-03-2026 | Accepted : 30-03-2026

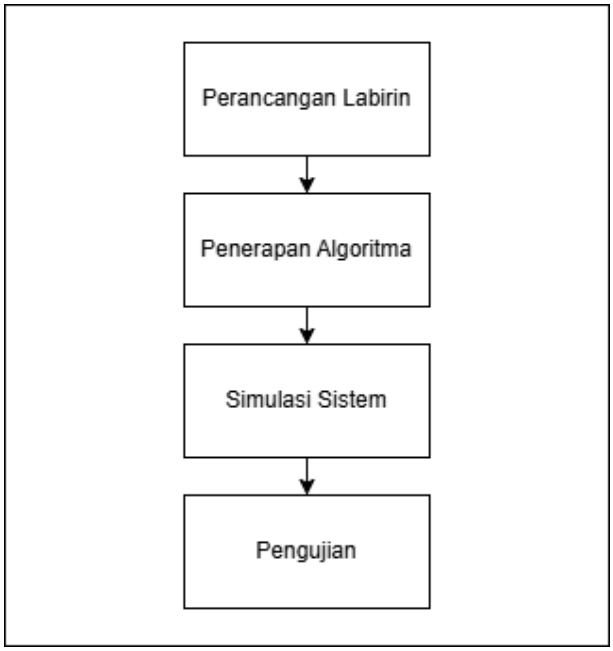
DFS bekerja dengan melakukan eksplorasi sedalam mungkin di dalam graf selama masih memungkinkan. DFS mengeksplorasi sisi (*edge*) yang keluar dari simpul yang paling baru ditemukan, selama simpul tersebut masih memiliki sisi yang belum di eksplorasi. Setelah semua sisi yang terhubung dengan simpul telah dieksplorasi, akan dilakukan proses pencarian mundur (*backtracking*) untuk melakukan eksplorasi pada sisi-sisi lain yang belum dicoba dari simpul sebelumnya [15].

Mengacu pada konsep dasar algoritma DFS, penelitian ini mengimplementasikan DFS sebagai metode eksplorasi dan navigasi robot di lingkungan labirin. Penelitian dilakukan dengan menggunakan pendekatan simulasi menggunakan simulator Webots, yang mencakup tahapan perancangan sistem, implementasi algoritma, simulasi sistem, serta pengujian DFS dalam navigasi robot. Fokus utama penelitian ini ialah menganalisis kemampuan DFS dalam membantu robot menavigasi diri melakukan eksplorasi buta pada labirin untuk mendapatkan titik akhir tanpa membutuhkan pemetaan awal secara eksplisit.

2. Metode Penelitian

Penelitian ini dilakukan dengan menggunakan metode eksperimen berbasis simulasi. Di dalam dunia robotika, penggunaan simulasi merupakan langkah penting dalam pengembangan robot sebelum dilakukan implementasi secara langsung pada dunia nyata [16]. Metode ini dipilih karena penelitian difokuskan pada proses perancangan dan pengujian algoritma *Depth-First Search* (DFS) dalam sistem navigasi robot e-puck pada simulator Webots.

Langkah awal penelitian adalah membuat diagram alur penelitian. Diagram alur penelitian berisi tahapan-tahapan yang dilakukan selama penelitian, yang dirancang untuk memastikan sistem yang dibangun dapat berfungsi dan bekerja sesuai dengan tujuan penelitian [17]. Tahapan penelitian dimulai dengan proses perancangan labirin, implementasi algoritma, simulasi sistem, hingga pengujian hasil simulasi untuk menganalisis efektivitas algoritma dalam menemukan titik akhir dari labirin.

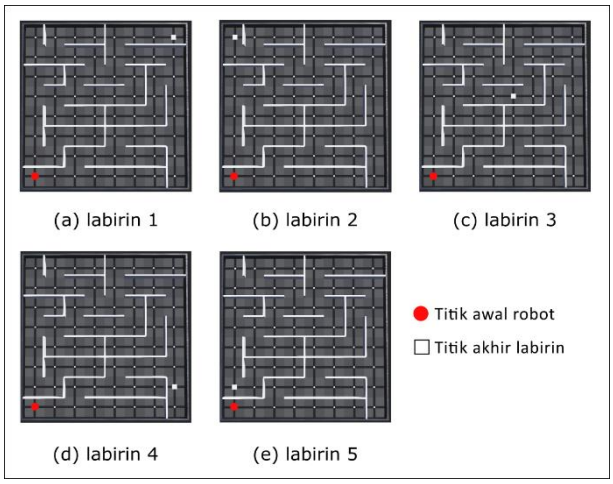


Gambar 1. Diagram Alur Penelitian

2.1. Perancangan Labirin

Arena dirancang berbentuk persegi, dengan panjang sisi 1 m. Arena dibagi menjadi 8 baris dan kolom dengan panjang sisi 12,5 cm. Terdapat *checkpoint* berupa kotak kecil berwarna putih di bagian tengah setiap sel, yang mana setiap *checkpoint* saling terhubung satu sama lain dengan garis hitam yang berada di tengah sel. Dinding labirin dibuat menggunakan objek dinding (*wall*) yang telah disediakan pada simulator Webots, dengan tinggi 10 cm. Pada penelitian ini digunakan lima skenario labirin yang berbeda untuk mengetahui kemampuan robot dalam beradaptasi pada tantangan yang berbeda. Perbedaan antar kelima labirin ini terletak pada titik akhir yang harus dicapai robot untuk menyelesaikan labirin.

Submitted : 20-03-2026 | Reviewed : 25-03-2026 | Accepted : 30-03-2026



Gambar 2. Arena labirin pada simulator Webots

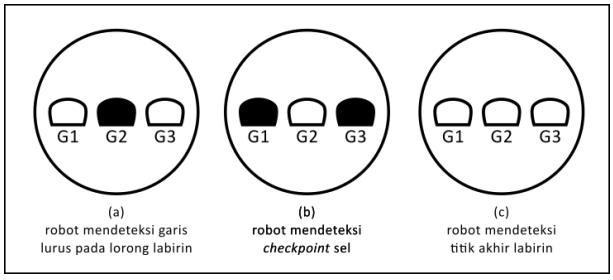
Dalam penelitian ini, robot sama sekali tidak memiliki pengetahuan tentang arena labirin. Robot akan melakukan eksplorasi mandiri dari titik awal (titik merah) hingga menemukan titik akhir labirin. Dalam hal ini robot akan memaksimalkan penggunaan sensor Inframerah dan Ground untuk mendeteksi halangan dan mengetahui posisinya dalam arena.

2.2. Penerapan Algoritma

Pada penelitian ini, algoritma *Depth-First Search* (DFS) digunakan sebagai metode pencarian jalur untuk membantu robot menelusuri labirin hingga mencapai titik akhir. Proses penelusuran dilakukan dengan cara robot bergerak dari satu sel ke sel berikutnya berdasarkan hasil pembacaan sensor terhadap lantai labirin.

Untuk membantu proses navigasi, robot menggunakan tiga buah sensor Ground untuk mendeteksi posisi robot pada sel, yang masing-masing diberi nama G1, G2, dan G3. Terdapat tiga kondisi berbeda berdasarkan hasil pembacaan sensor Ground pada robot: 1) robot mendeteksi garis tengah labirin dan akan berjalan maju ketika sensor G2 bernilai *false* dan sensor lainnya bernilai *true*, 2) robot mendeteksi *checkpoint* ketika sensor G2 bernilai *true* dan sensor lainnya bernilai *false*, 3) robot akan mendeteksi titik akhir labirin ketika ketiga sensor bernilai *true* (dapat dilihat pada Gambar 2).

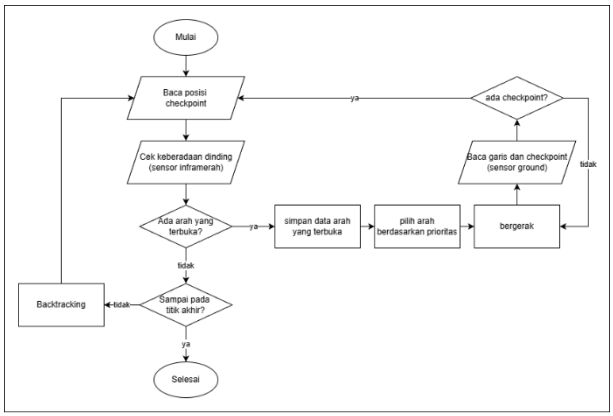
Setiap kali robot menemukan *checkpoint* baru, maka diasumsikan bahwa robot telah berpindah pada sel berikutnya. Pergerakan robot akan dibantu dengan garis hitam yang berada tepat di tengah lorong labirin. Sensor Ground akan mendeteksi garis hitam, dan memastikan bahwa robot selalu berada pada garis tersebut. Untuk meningkatkan kendali robot, kontrol PID digunakan untuk memperbaiki setiap *error* antara robot dan garis hitam arena. Pada penelitian ini, digunakan parameter PID sebagai berikut: Kp bernilai 0,1, Ki bernilai 0,000005, dan Kd bernilai 0,015. Nilai-nilai ini dipilih berdasarkan pengaturan dan penyesuaian yang telah dilakukan yang dapat memberikan kinerja maksimal pada robot dalam mengikuti garis dan mendeteksi *checkpoint* pada setiap sel.



Gambar 3. Kondisi sensor Ground saat menentukan posisi robot berdasarkan lantai labirin

Setiap kali robot berada pada *checkpoint* sel, robot akan berhenti sejenak dan melakukan pengecekan terhadap keberadaan dinding di sekelilingnya menggunakan sensor Inframerah. Berdasarkan hasil pembacaan sensor ini, robot akan menyimpan informasi sel di sekitarnya yang tidak terhalang dinding ke dalam struktur data tumpukan (*stack*). Selanjutnya robot akan memilih arah penelusuran berikutnya sesuai dengan data sel yang tersimpan pada *stack* berdasarkan prioritas arah penelusuran, yaitu kiri → depan → kanan atau kanan → depan → kiri. Apabila robot mencapai jalur buntu yang bukan merupakan titik akhir, maka robot akan melakukan proses *backtracking*, yaitu kembali ke posisi sebelumnya dan melanjutkan penelusuran ke jalur lain yang belum dilewati. Proses ini terus berulang hingga robot mencapai titik akhir labirin. Untuk lebih jelasnya, alur kerja algoritma DFS dapat dilihat pada Gambar 4.

Submitted : 20-03-2026 | Reviewed : 25-03-2026 | Accepted : 30-03-2026



Gambar 4. Alur kerja algoritma DFS pada robot

Robot dinyatakan berhasil menyelesaikan arena labirin apabila robot mampu mencapai titik akhir tanpa mengalami kegagalan navigasi. Kegagalan navigasi didefinisikan sebagai kondisi ketika robot tidak mampu melakukan proses *backtracking* saat menemukan jalur buntu, keluar dari garis tengah lorong labirin, sensor Ground gagal mendeteksi titik akhir labirin, atau robot tidak mampu melanjutkan penelusuran akibat kesalahan pembacaan sensor lainnya. Selain itu, simulasi dianggap berhasil apabila algoritma DFS mampu menavigasi robot dalam melakukan *backtracking* ketika diperlukan dan melanjutkan proses penelusuran hingga titik akhir labirin berhasil ditemukan.

2.3. Simulasi Sistem

Setelah semua komponen sistem dirancang dan algoritma diimplementasikan, tahap selanjutnya yaitu simulasi sistem menggunakan aplikasi simulator Webots. Simulasi dilakukan dengan menggunakan robot e-puck. Robot e-puck merupakan miniatur robot yang bersifat *open-source* yang dikembangkan oleh EPFL dan ditujukan sebagai media pembelajaran robotika [18]. Robot ini dipilih karena memiliki struktur yang sederhana dan memiliki sensor yang lengkap, seperti sensor Inframerah, Giroskop, dan sensor Ground yang dibutuhkan untuk membantu proses navigasi robot.

Tabel 1. Karakteristik robot e-puck pada simulator Webots

Karakteristik	Nilai
Diameter	71 mm
Tinggi	50 mm
Radius roda	20,5 mm
Berat	0,16 kg
Kecepatan maksimal	0,25 m/s
Kecepatan rotasi roda maksimal	6,28 rad/s
Perangkat/sensor pendukung	Motor, sensor posisi roda, sensor inframerah, sensor cahaya, LED, Kamera, Akselerometer, Giroskop, sensor Ground, Speaker

Bahasa pemrograman yang digunakan adalah Python, yang ditulis pada *controller file* dan dijalankan secara langsung pada aplikasi Webots. *Controller file* inilah yang akan digunakan oleh robot e-puck dalam mengolah data sensor serta mengambil keputusan. Simulasi dilakukan pada arena labirin berbentuk persegi dengan panjang sisi 1 meter, yang dibagi menjadi 64 sel dengan ukuran sisi masing-masing 12,5 cm. Robot akan bergerak secara otomatis menelusuri setiap sel berdasarkan hasil pembacaan sensor Inframerah dan sensor Ground hingga menemukan titik akhir labirin.

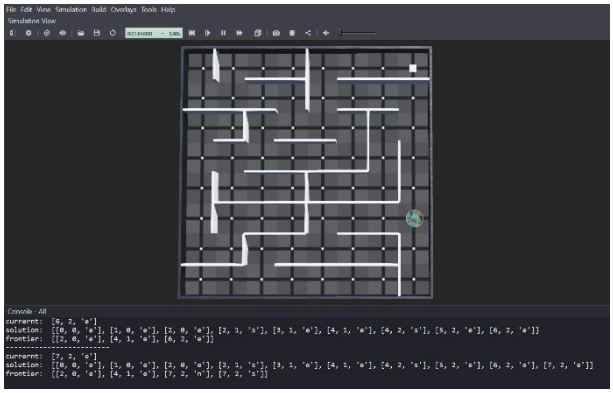
2.4. Pengujian

Pengujian dilakukan untuk mengetahui kemampuan robot dan efektifitas algoritma DFS dalam mengeksplorasi dan menemukan titik akhir labirin. Pada tahap ini robot akan diuji dengan lima skenario labirin yang berbeda dan dua prioritas eksplorasi yang berbeda (prioritas kiri dan prioritas kanan). Pengujian dianggap berhasil apabila robot mampu menyelesaikan seluruh arena labirin yang digunakan dalam tahap pengujian, dengan cara menemukan titik akhir labirin berdasarkan hasil pembacaan sensor Ground seperti yang telah digambarkan pada Gambar 3. Analisis hasil pengujian dijelaskan pada bagian Hasil dan Pembahasan.

Submitted : 20-03-2026 | Reviewed : 25-03-2026 | Accepted : 30-03-2026

3. Hasil dan Pembahasan

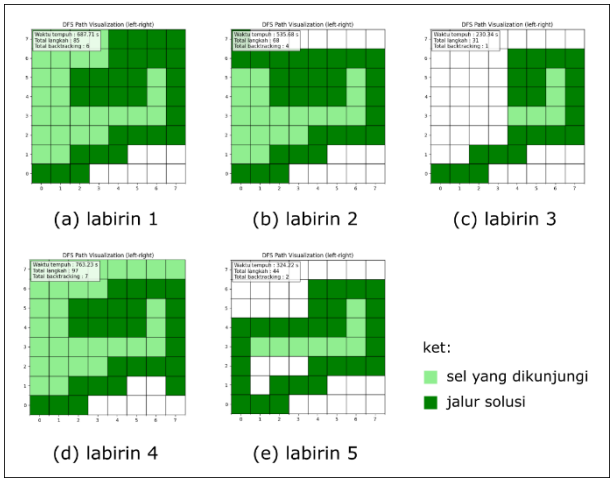
Bagian ini membahas hasil pengujian algoritma *Depth-First Search* (DFS) pada robot otonom dalam melakukan eksplorasi menemukan titik akhir pada kelima skenario labirin yang berbeda. Setiap skenario diuji dengan dua strategi penelusuran yang berbeda, yaitu prioritas kiri dan prioritas kanan, untuk melihat pengaruh arah prioritas navigasi robot terhadap performa pencarian jalur.



Gambar 5. Tampilan simulator Webots saat menjalankan simulasi

Dari seluruh pengujian yang dilakukan, algoritma DFS terbukti mampu membantu robot dalam menyelesaikan labirin. Robot melakukan eksplorasi labirin dari satu sel menuju sel lainnya secara bertahap. Pada setiap sel (*checkpoint*), robot akan melakukan pengecekan jalur yang akan ditempuh selanjutnya menggunakan sensor Inframerah. Jika ditemukan percabangan, maka robot akan menyimpan semua data percabangan, dan melanjutkan penelusuran ke salah satu cabang berdasarkan prioritas penelusuran. Jika ditemukan jalan buntu, maka robot akan melakukan proses *backtracking*, yaitu kembali ke percabangan terakhir dan melanjutkan penelusuran ke cabang lainnya. Proses ini dilakukan secara terus menerus hingga robot menemukan titik akhir labirin.

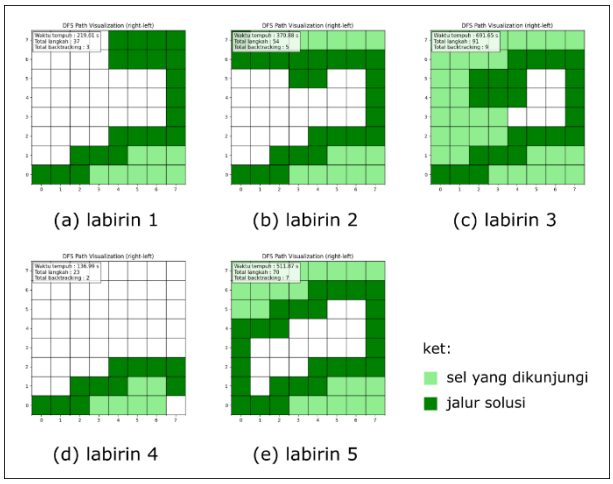
Pada strategi prioritas kiri, robot akan mengambil jalur paling kiri terlebih dahulu sesuai dengan prioritas navigasi (kiri → depan → kanan) saat menemukan percabangan. Hasil penelusuran robot menggunakan strategi prioritas kiri dapat dilihat pada Gambar 6.



Gambar 6. Visualisasi jalur eksplorasi robot menggunakan DFS prioritas kiri

Visualisasi yang berbeda ditunjukkan oleh penelusuran dengan strategi prioritas kanan. Pada strategi ini, robot akan memprioritaskan jalur sebelah kanan terlebih dahulu jika menemukan percabangan. Hasil penelusuran robot menggunakan strategi prioritas kanan dapat dilihat pada Gambar 7.

Submitted : 20-03-2026 | Reviewed : 25-03-2026 | Accepted : 30-03-2026



Gambar 7. Visualisasi jalur eksplorasi robot menggunakan DFS prioritas kanan

Pemilihan prioritas navigasi pada algoritma DFS dapat memengaruhi performa robot dalam menyelesaikan labirin. Secara umum, strategi prioritas kiri lebih sering digunakan pada algoritma ini. Namun, pemilihan prioritas navigasi dapat disesuaikan dengan kebutuhan dan jenis tantangan yang akan dihadapi untuk mendapatkan hasil yang maksimal. Perbandingan hasil penelusuran robot dengan kedua strategi dapat dilihat pada Tabel 2.

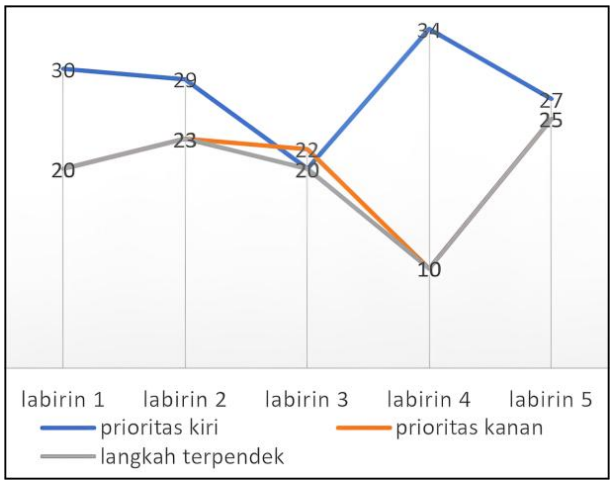
Tabel 2. Perbandingan strategi prioritas DFS pada robot

Labirin	Parameter	Prioritas kiri	Prioritas kanan
Labirin 1	Waktu tempuh (s)	687,71	219,01
	Total langkah	85	37
	Langkah solusi	30	20
	Backtracking	6	3
Labirin 2	Waktu tempuh (s)	535,68	370,88
	Total langkah	68	54
	Langkah solusi	29	23
	Backtracking	4	5
Labirin 3	Waktu tempuh (s)	230,34	691,65
	Total langkah	31	91
	Langkah solusi	20	22
	Backtracking	1	9
Labirin 4	Waktu tempuh (s)	763,23	136,99
	Total langkah	97	23
	Langkah solusi	34	10
	Backtracking	7	2
Labirin 5	Waktu tempuh (s)	324,22	511,87
	Total langkah	44	70
	Langkah solusi	27	25
	Backtracking	2	7

Merujuk pada data yang ditampilkan pada Tabel 2, terdapat perbedaan waktu tempuh robot dalam menyelesaikan labirin yang sama berdasarkan strategi prioritas navigasi yang digunakan. Prioritas kiri cenderung unggul pada labirin 3 dan 5, sedangkan prioritas kanan unggul pada labirin 1, 2, dan 4. Temuan ini sebanding dengan total langkah eksplorasi yang dilakukan oleh robot saat menyelesaikan labirin. Namun, hasil yang berbeda ditemukan pada langkah solusi yang dihasilkan, dimana strategi prioritas kanan menghasilkan langkah solusi yang lebih pendek pada empat dari lima skenario labirin yang diuji.

Langkah solusi merepresentasikan jalur akhir yang menghubungkan robot dari titik awal menuju titik akhir labirin setelah seluruh eksplorasi dan *backtracking* dilakukan. Pada algoritma *Depth-First Search* (DFS), langkah solusi yang dihasilkan sangat dipengaruhi oleh strategi prioritas penelusuran yang digunakan. Hal ini disebabkan karena DFS selalu mengeksplorasi jalur terakhir yang tersimpan pada struktur data tumpukan (*stack*) hingga mencapai jalur buntu atau titik tujuan eksplorasi. Oleh karena itu, pemilihan prioritas penelusuran dapat menghasilkan langkah solusi yang berbeda, meskipun lingkungan labirin yang digunakan tetap sama. Analisis langkah solusi menunjukkan bahwa panjang langkah eksplorasi DFS tidak selalu berbanding lurus terhadap langkah solusi yang dihasilkan.

Submitted : 20-03-2026 | Reviewed : 25-03-2026 | Accepted : 30-03-2026



Gambar 8. Grafik perbandingan jumlah langkah solusi labirin

Gambar 8 menunjukkan perbandingan antara langkah solusi yang dihasilkan DFS dengan langkah optimal terpendek yang dapat dilakukan untuk mencapai titik akhir labirin. Berdasarkan hasil analisis yang dilakukan, langkah solusi yang dihasilkan oleh algoritma DFS dengan strategi prioritas kanan memiliki panjang yang sama dengan jalur terpendek yang dapat ditempuh untuk mencapai titik akhir labirin pada empat dari lima skenario labirin yang diuji.

Secara teoritis, algoritma *Depth-First Search* (DFS) memiliki kompleksitas waktu sebesar *O(V+E)*, di mana *V* menyatakan jumlah simpul (*vertex)* dan *E* menyatakan jumlah sisi (*edge*) pada labirin. Dalam konteks navigasi robot labirin, simpul merepresentasikan sel atau *checkpoint* yang dapat dicapai robot, dan sisi merepresentasikan jalur yang menghubungkan antar sel tersebut. Dengan demikian, kompleksitas waktu DFS pada robot labirin berbanding lurus dengan jumlah sel dan banyaknya koneksi antar sel yang harus diperiksa oleh robot selama menjalankan proses penelusuran.

Sementara itu, kompleksitas ruang DFS bernilai *O(V)* yang berkaitan dengan kebutuhan memori untuk menyimpan data-data sel yang telah dikunjungi, serta data tumpukan (*stack*) yang digunakan selama proses navigasi. Pada implementasi sistem ini, ruang memori digunakan untuk menyimpan data *checkpoint* yang telah dilewati dan arah penelusuran yang akan dieksplorasi. Pada sistem navigasi robot yang digunakan, jumlah sel pada labirin masih terbatas sehingga kebutuhan memori DFS masih berada dalam batas yang dapat ditangani oleh sistem kontrol robot. Selain itu strategi prioritas navigasi yang digunakan (kiri atau kanan) memengaruhi urutan eksplorasi sel dan langkah solusi yang dihasilkan, namun tidak mengubah kompleksitas waktu dan ruang dalam algoritma DFS.

Tabel 3. Perbandingan hasil eksplorasi robot berdasarkan strategi navigasi yang digunakan

Parameter	Prioritas kiri	Prioritas kanan	Selisih
Waktu tempuh (s)	2541,18	1930,4	610,78
Total langkah	325	275	50
Langkah solusi	140	100	40
Backtracking	31	26	5

Dari keseluruhan hasil analisis yang dilakukan, strategi prioritas kanan jauh lebih unggul daripada prioritas kiri dari segi waktu tempuh, total langkah, serta *backtracking* yang dilakukan. Prioritas kanan unggul 610,78 detik untuk waktu tempuh, 50 langkah untuk total langkah eksplorasi, 40 langkah untuk total langkah solusi, dan 5 kali backtracking dibandingkan prioritas kiri secara keseluruhan. Meskipun demikian, penelitian ini tidak secara eksplisit menganalisis karakteristik topologi labirin yang digunakan secara matematis. Oleh karena itu, keunggulan strategi prioritas kanan tidak di klaim sebagai sifat umum algoritma DFS, melainkan sebagai hasil empiris yang muncul pada struktur labirin tertentu. Temuan ini mengindikasikan bahwa strategi prioritas penelusuran yang digunakan memiliki pengaruh yang signifikan terhadap performa navigasi robot, yang sangat bergantung pada karakteristik lingkungan yang digunakan.

4. Kesimpulan

Pada penelitian ini, algoritma *Depth-First Search* (DFS) terbukti efektif dalam membantu robot menavigasi diri dan menyelesaikan seluruh skenario labirin yang diberikan. Robot diuji dengan lima skenario labirin yang berbeda dan dua strategi prioritas navigasi, yaitu prioritas kiri dan prioritas kanan. Hasil pengujian menunjukkan kedua strategi ini mampu membantu robot dalam menyelesaikan semua skenario labirin yang diberikan.

Submitted : 20-03-2026 | Reviewed : 25-03-2026 | Accepted : 30-03-2026

Strategi navigasi prioritas kiri menghasilkan waktu eksplorasi yang lebih panjang dibandingkan prioritas kanan, dengan selisih waktu penyelesaian seluruh labirin sebesar 610,78 detik. Selain itu, empat dari lima langkah solusi yang dihasilkan oleh strategi prioritas kanan merupakan langkah optimal yang dapat ditempuh oleh robot untuk mencapai titik akhir labirin.

Meskipun demikian, penelitian ini memiliki beberapa keterbatasan. Pengujian sepenuhnya dilakukan pada lingkungan simulasi yang menggunakan arena labirin berbentuk persegi dengan kondisi yang terkontrol. Selain itu, penelitian ini hanya menggunakan satu jenis robot dan belum mempertimbangkan faktor-faktor dunia nyata seperti *noise* sensor, gangguan fisik nyata, serta ketidakteraturan permukaan. Oleh karena itu, hasil penelitian ini belum dapat sepenuhnya merepresentasikan performa algoritma DFS pada lingkungan nyata yang kompleks.

Dari sisi implikasi praktis, hasil penelitian ini menunjukkan bahwa algoritma DFS dapat dimanfaatkan sebagai metode eksplorasi dasar bagi robot otonom pada lingkungan yang belum diketahui. Dalam konteks penerapan di dunia nyata, pendekatan ini berpotensi digunakan sebagai solusi navigasi robot otonom, seperti robot gudang atau robot inspeksi, ketika informasi lingkungan belum tersedia atau tidak lengkap. Algoritma DFS memungkinkan robot melakukan eksplorasi lingkungan secara mandiri dengan mekanisme *backtracking* sederhana tanpa memerlukan pemetaan awal. Untuk pengembangan lebih lanjut, penelitian selanjutnya dapat diarahkan pada pengujian di lingkungan nyata, penggunaan skenario labirin yang lebih kompleks, serta penerapan pada jenis robot yang berbeda. Selain itu, integrasi DFS dengan algoritma lain diharapkan dapat meningkatkan efisiensi dan keandalan navigasi robot pada lingkungan yang dinamis.

Daftar Rujukan

[1] I. Palčič dan J. Prester, "Effect of Usage of Industrial Robots on Quality, Labor Productivity, Exports and Environment," *Sustainability*, vol. 16, no. 18, hlm. 8098, Jan 2024, doi: 10.3390/su16188098.

[2] E. C. Routray dan E. U. K. Sahoo, "AN OVERVIEW REPORT ON NEW ERA OF ROBOTICS STATISTICS & SYSTEMS," *INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT*, vol. 8, no. 10, hlm. d539–d544, Okt 2023.

[3] F. R. Saputra, R. Hadiazzaka, S. R. S. Silalahi, dan A. S. Priambodo, "IMPLEMENTASI LOGIKA FUZZY DALAM PENGHINDARAN HAMBATAN ROBOT E-PUCK PADA LINGKUNGAN STATIS," *JEIS: Jurnal Elektro dan Informatika Swadharma*, vol. 5, no. 1, hlm. 55–65, Jan 2025, doi: 10.56486/jeis.vol5no1.557.

[4] Gayatri M. Ghodke dan Nilima Prakash Jajoo, "Latest Innovation in Robotics," *IJARSCT*, hlm. 223–228, Mar 2024, doi: 10.48175/IJARSCT-15740.

[5] I. Kubasáková, J. Kubáňová, D. Benčo, dan N. Fábryová, "Application of Autonomous Mobile Robot as a Substitute for Human Factor in Order to Increase Efficiency and Safety in a Company," *Applied Sciences*, vol. 14, no. 13, hlm. 5859, Jul 2024, doi: 10.3390/app14135859.

[6] L. Wijayathunga, A. Rassau, dan D. Chai, "Challenges and Solutions for Autonomous Ground Robot Scene Understanding and Navigation in Unstructured Outdoor Environments: A Review," *Applied Sciences*, vol. 13, no. 17, hlm. 9877, Agu 2023, doi: 10.3390/app13179877.

[7] R. Amirullah, A. Rusdina, dan D. Darlis, "Implementasi Sistem Path Planning Dan Routing Untuk Mobile Robot Berbasis Visible Light Communication," *eProceedings of Engineering*, vol. 8, no. 5, Okt 2021, Diakses: 2 April 2026. [Daring]. Tersedia pada: https://openlibrarypublications.telkomuniversity.ac.id/index.php/engineering/article/view/15584

[8] A. Jayadi, P. A. Pratomo, dan E. R. Dalimunthe, "Implementasi Algoritma A-Star untuk Perencanaan Jalur Robot Menggunakan Deteksi Warna Berbasis Kamera," *Blend Sains J. Teknik*, vol. 4, no. 1, hlm. 116–127, Jul 2025, doi: 10.56211/blendsains.v4i1.987.

[9] M. D. Fauzi, G. Hajar, D. N. Rachmaniar, dan M. H. Z. al Faroby, "Simulasi Pengiriman Air Mineral Galon dengan Multi Depot Menggunakan Hill Climbing dan Algoritma A*," *Jurnal Pustaka AI (Pusat Akses Kajian Teknologi Artificial Intelligence)*, vol. 5, no. 2, hlm. 316–323, 2025, doi: 10.55382/jurnalpustakaai.v5i2.1170.

[10] D. F. Adryady, A. Prasetyo, H. P. Shatyaziamawan, dan A. S. Priambodo, "Implementasi Algoritma Path Planning dan Mapping Arena pada Mobile Robot," *Jurnal Teknik Elektro dan Komputer TRIAC*, vol. 9, no. 2, hlm. 41–45, Agu 2022, doi: 10.21107/triac.v9i2.13215.

[11] A. Sabril dan N. Abdal, "PERBANDINGAN WAKTU TEMPUH MOBILE ROBOT DALAM ARENA LABIRIN DENGAN ALGORITMA TANGAN KIRI DAN ALGORITMA TANGAN KANAN," *Jurnal Media Elektrik*, vol. 17, hlm. 97, Agu 2020, doi: 10.26858/metrik.v17i3.14961.

[12] R. Scheffler, "On the recognition of search trees generated by BFS and DFS," *Theoretical Computer Science*, vol. 936, hlm. 116–128, Nov 2022, doi: 10.1016/j.tcs.2022.09.018.

[13] A. Mustaqim, D. B. Dinova, M. S. Fadhilah, R. Seivany, B. Prasetiyo, dan M. A. Muslim, "Optimizing the Implementation of the BFS and DFS algorithms using the web crawler method on the kumparan site," *J. Soft Comput. Explor.*, vol. 5, no. 2, hlm. 200–206, Jul 2024, doi: 10.52465/joscex.v5i2.309.

[14] A. Putra, A. Maulana, S. Febrianti, N. Camelia, S. Hutagalung, dan A. Khudori, "Comparison of Breadth-First Search (BFS) and Depth-First Search (DFS) Algorithms for Shortest Search in Campus Labyrinth," *Journal of Enhanced Studies in Informatics and Computer Applications*, vol. 2, hlm. 43–51, Jul 2025, doi: 10.47794/jesica.v2i2.14.

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, dan Clifford Stein, *Introduction to Algorithms*, 2 ed. United States: The MIT Press, 2001.

[16] F. Kurniawan, A. Saddiq, dan C. Saragih, "Analisis Kinerja Algoritma Path Planning Klasik dan Heuristik pada Robot Bergerak dalam Lingkungan Dinamis," *Seminar Nasional Teknik Elektro*, vol. 4, hlm. 129–136, Sep 2025, doi: 10.46962/snte.25.081.

[17] A. Audy, H. Mudia, dan A. Nofrianto, "Sistem Pemantauan Keluaran Generator Set Berbasis IoT," *Jurnal Pustaka AI (Pusat Akses Kajian Teknologi Artificial Intelligence)*, vol. 5, no. 3, hlm. 462–470, Des 2025, doi: 10.55382/jurnalpustakaai.v5i3.1389.

[18] S. R. Divanny, "PENERAPAN OPENCV DAN FUZZY LOGIC CONTROLLER UNTUK LINE FOLLOWER BERBASIS KAMERA PADA SIMULASI ROBOT E-PUCK DI WEBOTS," *JITET*, vol. 12, no. 3, Agu 2024, doi: 10.23960/jitet.v12i3.4718.