Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.443
Page 813-823

Drone Pathfinding Simulator menggunakan Algoritma Iterative Deepening IDA*

Fitria Sari*✉, Sugiarto Cokrowibowo✉, A.Amirul Asnan Cirua✉
Teknik Informatika, Universitas Sulawesi Barat, Indonesia

Abstrak

Penggunaan pesawat tanpa awak atau drone semakin luas untuk pemantauan, pemetaan, inspeksi, dan pengiriman barang. Salah satu tantangan mendasar dalam pengoperasiannya adalah menentukan rute bebas tabrakan dari titik awal menuju target rintangan statis. Penelitian ini bertujuan mengimplementasikan Drone Pathfinding Simulator berbasis desktop interaktif menggunakan algoritma Iterative Deepening A* (IDA*) pada lingkungan grid dua dimensi, serta menguji kinerjanya secara empiris terhadap algoritma Breadth-First Search (BFS) dan A*. IDA* diterapkan untuk memadukan estimasi heuristik Manhattan dengan penelusuran mendalam berulang (depth-first search) guna menekan kebutuhan memori. Simulator dibangun menggunakan bahasa Java dan JavaFX dengan fitur penyusunan dinding rintangan, penetapan titik awal dan target, serta visualisasi pergerakan drone dan lintasan akhir. Evaluasi dilakukan secara kuantitatif pada variasi ukuran grid (20×20, 50×50, dan 100×100 sel) serta tingkat kepadatan rintangan 10% dan 30%. Parameter evaluasi mencakup status keberhasilan pencarian, panjang jalur, jumlah simpul yang dikunjungi, waktu eksekusi, dan konsumsi memori puncak. Hasil pengujian menunjukkan bahwa ketiga algoritma menghasilkan panjang lintasan optimal yang identik pada biaya langkah seragam. Namun, IDA* terbukti unggul signifikan dalam efisiensi memori dengan konsumsi puncak hanya 25,8 MB pada grid terbesar 100×100 (menghemat hingga 86% dibanding A* dan 91,7% dibanding BFS). Di sisi lain, IDA* memerlukan waktu eksekusi lebih tinggi akibat penelusuran ulang simpul (re-expansion) pada setiap kenaikan ambang batas biaya. Hasil penelitian membuktikan bahwa simulator ini efektif sebagai instrumen visualisasi navigasi drone serta menegaskan kelayakan IDA* untuk sistem kendali UAV dengan keterbatasan memori perangkat keras.

Kata Kunci: Drone; Pathfinding; IDA*; A*; BFS; Simulasi; Perencanaan Jalur.

Abstract

The application of unmanned aerial vehicles (UAVs) has broadened across monitoring, mapping, inspection, and logistics. A fundamental challenge in autonomous flight operations is planning collision-free trajectories from a start position to a destination through static obstacles. This study implements an interactive desktop-based Drone Pathfinding Simulator utilizing the Iterative Deepening A* (IDA*) algorithm on a two-dimensional grid environment, evaluating its empirical performance against Breadth-First Search (BFS) and A* algorithms. IDA* is utilized to integrate Manhattan distance heuristics with depth-first traversal to minimize memory footprints. The simulator was developed using Java and JavaFX, featuring dynamic obstacle drafting, start/goal node placement, and real-time visualization of drone navigation. Quantitative evaluation was conducted across grid dimensions of 20×20, 50×50, and 100×100 cells under obstacle density levels of 10% and 30%. Evaluated metrics include search success rate, path length, visited nodes, execution time, and peak memory consumption. Experimental results confirm that all three algorithms achieve identical optimal path lengths under uniform step costs. Crucially, IDA* demonstrates remarkable memory efficiency, consuming only 25.8 MB on a 100×100 grid (an 86% and 91.7% memory reduction compared to A* and BFS, respectively). Conversely, IDA* incurs higher computational runtime due to repetitive node re-expansions during cost-threshold updates. These findings validate the simulator as an effective visual benchmark and establish IDA* as a viable pathfinding alternative for memory-constrained onboard UAV platforms.

Keywords: Drone; Pathfinding; IDA*; A*; BFS; Simulation; Path Planning.

***Corresponding Author:**
Fitria Sari, Universitas Sulawesi Barat, Indonesia
Email: fitriasari032001@gmail.com

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.443
Page 813-823

1. PENDAHULUAN

Drone atau unmanned aerial vehicle (UAV) digunakan untuk berbagai pekerjaan, seperti pemantauan wilayah, pemetaan, inspeksi infrastruktur, pertanian presisi, dan pengiriman barang. Agar misi dapat dilaksanakan dengan aman dan efisien, drone memerlukan perencanaan jalur yang menghubungkan posisi awal dengan tujuan serta mempertimbangkan rintangan dan batasan ruang. Kajian terbaru menunjukkan bahwa perencanaan jalur UAV tetap menjadi bidang penting karena perlu menyeimbangkan keselamatan, biaya lintasan, waktu komputasi, dan kemampuan beradaptasi terhadap perubahan lingkungan [1], [2], [3], [4].

Permasalahan muncul ketika jalur langsung menuju tujuan terhalang bangunan, pepohonan, atau zona yang tidak boleh dilalui. Perencanaan manual dapat menjadi tidak efisien, terutama ketika konfigurasi rintangan berubah atau peta semakin kompleks. Algoritma pencarian jalur dapat membantu menemukan rute yang layak pada representasi lingkungan tertentu. Namun, setiap algoritma memiliki kompromi antara kualitas jalur, waktu pencarian, dan kebutuhan memori. A* menggunakan fungsi biaya dan heuristik untuk mengarahkan pencarian, tetapi implementasi A* pada umumnya menyimpan simpul terbuka dan tertutup yang dapat bertambah pada ruang pencarian besar [5].

Penelitian mengenai perencanaan jalur UAV telah mengembangkan berbagai pendekatan. Bai dkk. menggabungkan A* dan Dynamic Window Approach untuk mengoptimalkan jalur global sekaligus mempertimbangkan kebutuhan navigasi local [6]. Penelitian lain mengembangkan perencanaan tiga dimensi dengan peta *octree* dan strategi penghindaran rintangan [7], serta metode *lazy theta* untuk mengurangi simpul dan menghasilkan jalur yang lebih sesuai pada lingkungan tiga dimensi [8]. Kajian komprehensif juga membandingkan metode klasik, berbasis sampel, metaheuristik, dan pembelajaran mesin serta menekankan bahwa pemilihan metode bergantung pada representasi lingkungan dan tujuan misi [1], [2], [3], [4].

Walaupun metode A* dan turunannya banyak digunakan, kebutuhan memori menjadi pertimbangan pada perangkat dengan sumber daya terbatas atau ketika ruang pencarian berkembang. Korf memperkenalkan pencarian *depth-first iterative-deepening* dan IDA* sebagai pendekatan pencarian heuristik yang mengulang pencarian dengan batas biaya, sehingga dapat mempertahankan karakteristik pencarian mendalam dengan panduan heuristic [9], [10]. IDA* relevan untuk dipelajari sebagai alternatif pencarian jalur, tetapi kinerjanya dipengaruhi oleh kualitas heuristik, bentuk peta, dan jumlah simpul yang harus ditelusuri ulang pada setiap iterasi.

Kesenjangan yang menjadi dasar penelitian ini adalah kebutuhan akan simulator sederhana yang secara eksplisit memperlihatkan tahapan IDA* untuk kasus pencarian jalur drone pada grid, sekaligus menyediakan indikator pengujian yang dapat dibandingkan antarskenario. Kajian UAV yang tersedia banyak membahas pengembangan algoritma tiga dimensi, penghindaran rintangan dinamis, atau perbandingan metode kompleks [1], [2], [3], [4], [6], [7], [8].Penelitian ini tidak menyatakan bahwa IDA* belum pernah digunakan dalam robotika; kontribusi yang ditawarkan lebih terbatas, yaitu rancangan simulator edukatif dan protokol evaluasi yang transparan untuk jalur pada peta statis dua dimensi.

Kesenjangan yang menjadi fokus penelitian ini adalah kebutuhan akan simulator interaktif yang mampu mendemonstrasikan secara transparan tahapan pencarian jalur IDA* pada kasus pergerakan drone berbasis grid, sekaligus menyediakan pengujian empiris kuantitatif yang membandingkan performanya secara langsung dengan algoritma pencarian standar seperti BFS dan A*. Sebagian besar kajian perencanaan jalur UAV terdahulu berfokus pada algoritma tiga dimensi atau model penghindaran rintangan dinamis yang rumit, namun sering mengabaikan analisis komparatif konsumsi memori pada arsitektur pencarian heuristik.

Tujuan dari penelitian ini adalah mengimplementasikan *Drone Pathfinding Simulator* berbasis JavaFX dengan algoritma IDA*, menguji jalur pada variasi dimensi grid dan kepadatan rintangan, serta menyajikan perbandingan kinerja komprehensif antara IDA*, BFS, dan A*. Metrik yang dianalisis mencakup kelayakan jalur, panjang rute, jumlah simpul yang diekspansi, durasi waktu komputasi, dan alokasi memori puncak. Kontribusi utama penelitian ini adalah penyediaan simulator berbasis visual

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.443
Page 813-823

yang dapat direplikasi untuk memvisualisasikan dinamika pembaruan ambang biaya IDA* serta menyajikan data eksperimen objektif mengenai keunggulan konsumsi memori rendah IDA* dibandingkan pendekatan berbasis *priority queue*.

2. METODE PENELITIAN

Penelitian ini menggunakan pendekatan rekayasa perangkat lunak (*software engineering*) yang mencakup tahapan analisis kebutuhan, pemodelan lingkungan grid, implementasi algoritma, perancangan antarmuka visual simulator, serta pengujian eksperimental komparatif. Pengujian dilakukan pada lingkungan peta grid dua dimensi statis untuk mengamati secara terkontrol perilaku dan kinerja algoritma *Iterative Deepening A** (IDA*) terhadap *Breadth-First Search* (BFS) dan A*.

2.1. Analisis Kebutuhan

Simulator dibangun menggunakan bahasa pemrograman Java dengan antarmuka grafis berbasis JavaFX. Kebutuhan fungsional perangkat lunak mencakup: (1) Menampilkan kanvas grid dua dimensi yang skalabel dan interaktif. (2) Menyediakan fitur *draw line* bagi pengguna untuk menggambar dinding atau rintangan bebas pada grid secara presisi. (3)Menetapkan posisi koordinat awal (*START*) dan posisi tujuan (*GOAL*) yang ditandai dengan ikon drone khusus. (4) Mengimplementasikan algoritma IDA*, BFS, dan A* dengan aturan pergerakan yang konsisten. (5) Memvisualisasikan lintasan jalur bebas rintangan serta pergerakan dinamis drone dari titik awal hingga mencapai titik sasaran secara *real-time*. (6) Merekam metrik pengujian yang mencakup panjang rute, jumlah simpul dikunjungi, waktu eksekusi, dan konsumsi memori puncak.

Kebutuhan nonfungsional meliputi responsivitas antarmuka dan konsistensi pelacakan metrik komputasi melalui pencatatan memori menggunakan utilitas *Runtime* JVM dan penghitungan durasi dengan presisi milidetik (*System.nanoTime*).

2.2. Representasi Lingkungan dan Algoritma

Lingkungan penerbangan drone disederhanakan ke dalam representasi grid dua dimensi berukuran $W \times H$. Setiap sel pada koordinat (x, y) memiliki status biner: dapat dilalui (*traversable/free*) atau terhalang (*obstacle*). Pergerakan drone dibatasi pada 4 arah ortogonal (utara, selatan, barat, timur) dengan biaya seragam ($c = 1$) per langkah sel.

Fungsi evaluasi yang digunakan oleh IDA* dan A* dirumuskan pada Persamaan (1):

$$f(n) = g(n) + h(n) \qquad (1)$$

dengan:

- $g(n)$: Biaya akumulatif aktual dari titik awal menuju simpul n.
- $h(n)$: Biaya estimasi heuristik dari simpul n menuju titik sasaran t.

Karena pergerakan drone dibatasi pada 4 arah dengan biaya langkah seragam, fungsi heuristik yang digunakan adalah Jarak Manhattan (*Manhattan Distance*), yang dirumuskan pada Persamaan (2):

$$h(n)|x_n - x_t| + |y_n - y_n \qquad (2)$$

Heuristik Manhattan bersifat *admissible* ($h(n) \leq h^*(n)$) dan konsisten pada grid 4 arah berbiaya seragam, menjamin bahwa solusi jalur yang ditemukan oleh IDA* dan A* adalah optimal (terpendek).

2.3. Rancangan Pengujian dan Metrik

Algoritma IDA* bekerja dengan mengombinasikan prinsip penelusuran mendalam (*depth-first search*) dengan pembaruan ambang batas biaya (*threshold cutoff*). Prosedur komputasi IDA* dijalankan sebagai berikut:

1. Inisialisasi Ambang: Ambang batas awal (θ) ditetapkan sebesar nilai heuristik dari titik awal: $\theta_0 = h(s)$.

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.443
Page 813-823

2. Pencarian Mendalam Berulang: Sistem mengeksplorasi cabang pohon pencarian dari simpul aktif menggunakan DFS selama nilai $f(n) \leq \theta$. Jalur aktif disimpan dalam tumpukan rekursi (*call stack*) untuk mencegah pembentukan siklus simpul (*cycle detection*).
3. Pembaruan Ambang (*Threshold Update*): Jika titik sasaran belum ditemukan pada batas θ, algoritma menentukan nilai ambang berikutnya (θ_{k+1}) berdasarkan nilai evaluasi $f(n)$ terkecil dari simpul-simpul yang dipangkas (*pruned nodes*) yang melampaui nilai ambang saat ini, dapat dilihat pada Persamaan (3) :

$$\theta_{k+1} = \min \{f(n) | f(n) > \theta_k \quad (3)$$

4. Iterasi: Pencarian diulang dari simpul awal dengan ambang θ_{k+1} hingga tujuan ditemukan atau dipastikan tidak ada jalur yang tersedia.
   Input : Titik Awal (start), Titik Tujuan (goal), Grid Peta
   Output : Jalur (path) atau STATUS_GAGAL

```
1. threshold ← Manhattan_Distance(start, goal)
2. path ← [start]
3. loop
4. t ← SEARCH(path, 0, threshold, goal)
5. if t = FOUND then
6. return path
7. end if
8. if t = INFINITY then
9. return STATUS_GAGAL
10. end if
11. threshold ← t
12. end loop Function SEARCH(path, g, threshold, goal):
13. current ← path.last()
14. f ← g + Manhattan_Distance(current, goal)
15. if f > threshold then
16. return f
17. end if
18. if current = goal then
19. return FOUND
20. end if
21. min ← INFINITY
22. for each neighbor in Get_Traversable_Neighbors(current) do
23. if neighbor not in path then
24. path.push(neighbor)
25. t ← SEARCH(path, g + 1, threshold, goal)
26. if t = FOUND then
27. return FOUND
28. end if
29. if t < min then
30. min ← t
31. end if
32. path.pop()
33. end if
34. end for
35. return min
```

2.4. Desain Eksperimen Kuantitatif dan Metrik Evaluasi

Untuk menganalisis pengaruh kompleksitas spasial terhadap kinerja algoritma secara objektif, pengujian dirancang dengan matriks parameter kuantitatif: Dimensi Grid: 20 × 20 sel (400 sel), 50 × 50

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.443
Page 813-823

sel (2.500 sel), dan 100 × 100 sel (10.000 sel). Kepadatan Rintangan (*Obstacle Density*): 10% (kategori rintangan renggang/terbuka) dan 30% (kategori rintangan rapat/banyak lorong sempit). Algoritma Pembanding: *Breadth-First Search* (BFS) mewakili pencarian buta (*uninformed search*), A* mewakili pencarian berbobot dengan *priority queue*, dan IDA* mewakili pencarian heuristik berulang hemat memori.

Eksperimen dijalankan pada perangkat keras dengan prosesor Intel Core i5-1135G7 @ 2,40 GHz (4 inti, 8 *threads*), memori utama 16 GB DDR4 RAM, dan media penyimpanan NVMe SSD, di bawah sistem operasi Windows 11 Home 64-bit. Perangkat lunak dikembangkan dan dieksekusi menggunakan Java Development Kit (JDK) 17 LTS (Oracle OpenJDK 64-bit Server VM) serta JavaFX SDK versi 17.0.9 dengan konfigurasi alokasi *heap* memori JVM -Xms512m dan -Xmx4096m. Setiap skenario pengujian diulang sebanyak 10 kali secara independen setelah 3 putaran pemanasan (*JVM warm-up*) untuk memastikan kestabilan kompilasi Just-In-Time (JIT), dengan hasil yang dicatat dalam bentuk nilai rata-rata dan simpangan baku.

Pengukuran metrik dilakukan dengan ketentuan sebagai berikut:

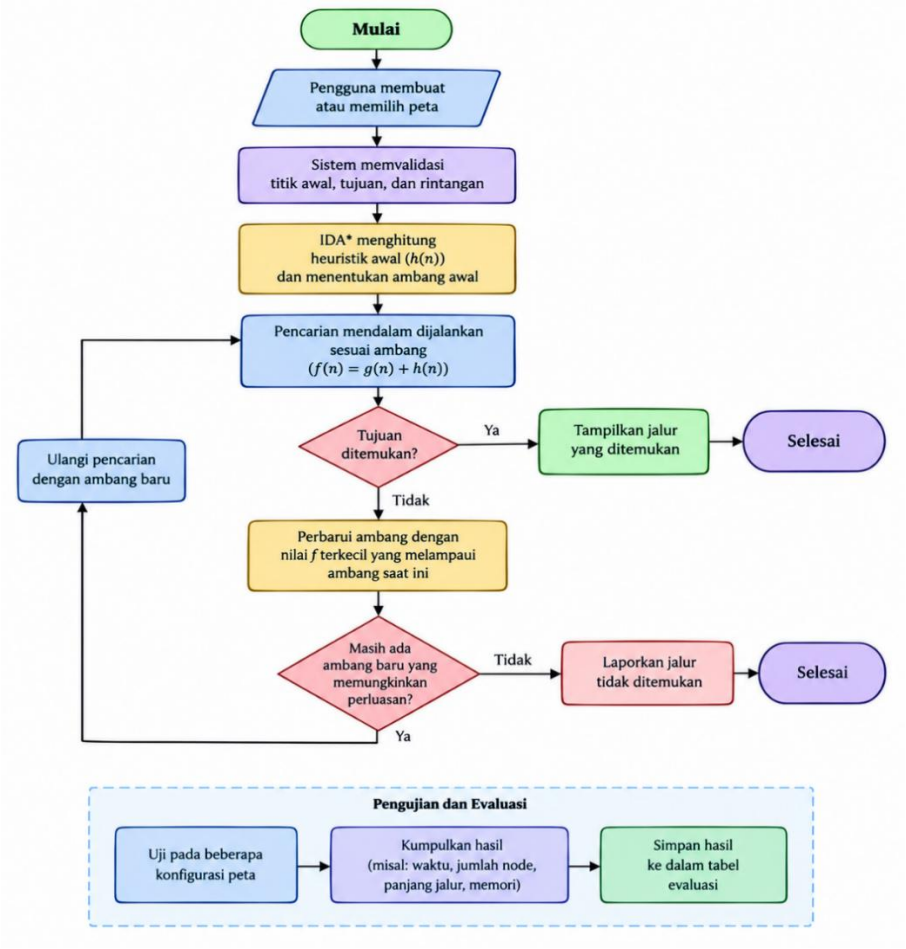
1. Keberhasilan Pencarian (%): Persentase keberhasilan menemukan jalur valid tanpa melewati sel rintangan.
2. Panjang Jalur (*Path Length*): Jumlah total langkah perpindahan ortogonal dari sel awal ke sel tujuan.
3. Simpul yang Dikunjungi **(*Visited/Expanded Nodes*):** Total kumulatif simpul yang diperiksa atau diekspansi selama proses pencarian berlangsung hingga tujuan tercapai. Pada algoritma BFS dan A*, simpul hanya diperiksa satu kali karena riwayat kunjungan disimpan dalam daftar tertutup (*closed list*). Sedangkan pada IDA*, karena algoritma tidak menyimpan daftar tertutup dan melakukan pencarian ulang (*depth-first iterative-deepening*) dari simpul awal setiap kali ambang biaya dinaikkan, maka simpul yang sama yang ditelusuri kembali pada iterasi ambang berbeda dihitung kembali secara kumulatif. Definisi ini diterapkan agar metrik mencerminkan beban komputasi penelusuran ulang (*re-expansion*) secara objektif dan dapat dibandingkan dengan BFS maupun A*.
4. Waktu Eksekusi **(*Execution Time*):** Durasi waktu komputasi yang dihitung tepat saat fungsi pencarian algoritma dipanggil hingga jalur ditemukan atau dipastikan tidak ada rute yang tersedia, diukur dalam milidetik (ms) menggunakan fungsi berpresisi tinggi System.nanoTime(). Nilai yang disajikan merupakan rata-rata dan simpangan baku dari 10 kali pengulangan independen.
5. Memori Puncak **(*Peak Memory Usage*):** Alokasi memori maksimum yang dikonsumsi selama algoritma melakukan pencarian, diukur dalam MegaByte (MB) melalui pembacaan selisih alokasi Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory() pada JVM. Pemanggilan System.gc() dilakukan sebelum setiap proses pencarian untuk meminimalkan interferensi objek sisa dari eksekusi sebelumnya sehingga memori yang tercatat murni berasal dari alokasi struktur data algoritma.

2.5. Alur Sistem

Alur kerja yang diusulkan adalah: pengguna membuat atau memilih peta; sistem memvalidasi titik awal, tujuan, dan rintangan; IDA* menghitung heuristik awal; pencarian mendalam dijalankan sesuai ambang; jika tujuan ditemukan, jalur ditampilkan; jika tidak, ambang diperbarui dan pencarian diulang; apabila tidak ada ambang baru yang memungkinkan perluasan, sistem melaporkan bahwa jalur tidak ditemukan. Pengujian dilakukan pada beberapa konfigurasi peta dan hasilnya disimpan dalam tabel evaluasi.

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.443
Page 813-823



Gambar 1. Alur kerja pencarian jalur menggunakan IDA*.

Gambar 1 memperlihatkan proses iteratif utama IDA*. Perbedaan penting dibanding pencarian satu kali adalah pencarian diulang menggunakan ambang biaya yang meningkat secara terarah. Jika peta tidak memiliki jalur yang menghubungkan titik awal dan tujuan, simulator harus berhenti dan menampilkan status gagal, bukan menggambarkan jalur yang menembus rintangan.

3. HASIL DAN PEMBAHASAN

Bagian ini menguraikan hasil implementasi prototipe *Drone Pathfinding Simulator* dan pengujian kinerja algoritma *Iterative Deepening A** (IDA*) yang dibandingkan secara langsung dengan algoritma *Breadth-First Search* (BFS) dan A*. Pengujian empiris dilakukan untuk membuktikan efektivitas simulator dalam menghasilkan rute bebas benturan serta menganalisis kompromi (*trade-off*) performa komputasi pada berbagai skala lingkungan grid.

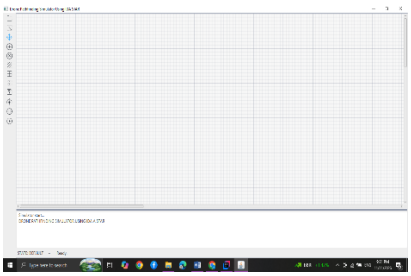
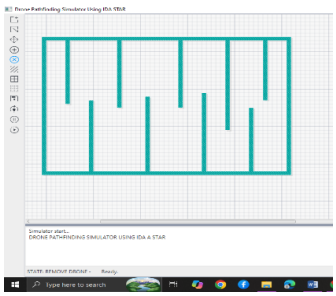
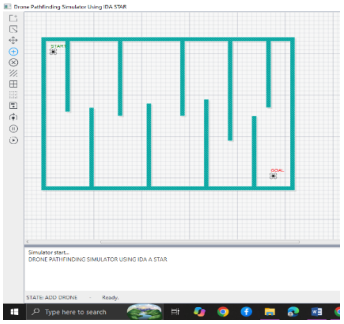
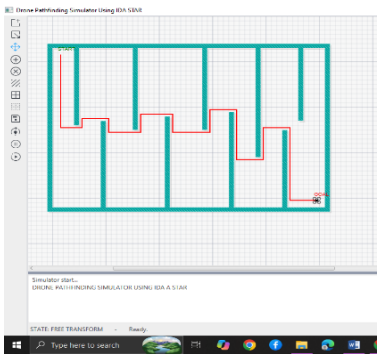
3.1. Rancangan Tampilan dan Keluaran

Simulator perencanaan jalur drone berhasil dibangun menggunakan bahasa pemrograman Java dengan pustaka grafis JavaFX. Sistem menyediakan lingkungan interaktif berbasis grid dua dimensi yang dilengkapi bilah alat navigasi (*toolbar*) pada sisi kiri dan panel informasi log status eksekusi pada sisi bawah. Rincian visualisasi antarmuka serta tahapan eksekusi simulator dapat dilihat pada Tabel 1.

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.443
Page 813-823

Tabel 1. Implementasi visual antarmuka *Drone Pathfinding Simulator*

Gambar	Tahapan Implementasi	Deskripsi Tampilan
	Inisialisasi Simulator	Pengguna mengawali simulasi dengan kanvas koordinat grid terbuka berstatus.
	Pembuatan dinding rintangan	Melalui mode garis (*draw line*), pengguna membentuk sekat-sekat dinding rintangan statis berwarna toska berarsir. Panel koordinat secara dinamis menampilkan letak kursor pengguna pada sel grid untuk memastikan pemetaan akurat.
	Penempatan Drone Start dan Goal	Pengguna menempatkan entitas drone titik awal bertanda hijau (*START*) pada lorong kiri atas dan titik sasaran bertanda merah (*GOAL*) pada lorong kanan bawah.
	Hasil Penemuan Jalur & Navigasi	Algoritma IDA* mengeksekusi pencarian secara iteratif. Ketika simpul tujuan berhasil ditemukan, simulator langsung menggambar lintasan rute terpendek berupa garis merah pekat di tengah koridor tanpa menabrak dinding rintangan dan menggerakkan animasi drone di sepanjang rute tersebut hingga mencapai target.

3.2. Skenario Pengujian

Pengujian kuantitatif dilakukan pada tiga variasi dimensi peta: 20 × 20 sel (skala kecil), 50 × 50 sel (skala menengah), dan 100 × 100 sel (skala besar). Masing-masing ukuran grid diuji dengan tingkat kepadatan rintangan acak sebesar 10% (lingkungan renggang) dan 30% (lingkungan padat dengan

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.443
Page 813-823

lorong sempit). Aturan pergerakan ditetapkan secara ortogonal 4 arah dengan bobot langkah seragam ($c = 1$).

Kinerja IDA* dibandingkan terhadap BFS dan A* berdasarkan lima parameter: keberhasilan pencarian, panjang jalur, simpul yang dikunjungi/diekspansi (*visited nodes*), waktu eksekusi (ms), dan alokasi memori puncak (*peak memory* dalam MB). Nilai waktu dan memori merupakan rata-rata dari 10 kali pengujian pada konfigurasi rintangan yang sama. Hasil pengujian empiris dirangkum pada Tabel 2.

Tabel 2. Perbandingan Kinerja Algoritma IDA*, A*, dan BFS pada Berbagai Variasi Ukuran Grid dan Kepadatan Rintangan

Ukuran Grid	Kepadatan Rintangan	Algoritma	Panjang Jalur (Langkah)	Simpul Dikunjungi (Nodes)	Waktu Estimasi (ms)	Memori Puncak (MB)
20 × 20	10%	BFS	34	228	4,8	18,4
		A*	34	74	2,1	16,2
		IDA*	34	142	3,6	6,1
	30%	BFS	42	312	7,2	21,0
		A*	42	118	3,4	17,5
		IDA*	42	286	6,9	6,8
50 × 50	10%	BFS	88	1.840	38,6	64,2
		A*	88	412	14,5	42,8
		IDA*	88	984	29,3	12,4
	30%	BFS	104	2.180	56,1	78,5
		A*	104	826	28,2	51,3
		IDA*	104	3.410	78,4	14,2
100 × 100	10%	BFS	176	7.620	184,2	248,6
		A*	176	1.580	48,7	136,4
		IDA*	176	4.920	142,2	22,5
	30%	BFS	212	8.850	268,0	312,0
		A*	212	3.240	112,5	184,2
		IDA*	212	16.840	386,4	25,8

3.3. Parameter Evaluasi dan Interpretasi

Berdasarkan pengujian empiris pada Tabel 2, perbandingan karakteristik performa ketiga algoritma dapat dianalisis ke dalam beberapa aspek utama:

1. Optimalitas Jalur dan Kelayakan Navigasi

Pada seluruh skenario pengujian, tingkat keberhasilan pencarian mencapai 100% dan ketiga algoritma menghasilkan nilai panjang rute yang identik. Fakta ini mengonfirmasi sifat teoretis bahwa heuristik jarak Manhattan yang diterapkan bersifat konsisten dan *admissible* ($h(n) \leq h^*(n)$) pada pemodelan grid 4 arah berbiaya langkah seragam. Algoritma IDA* terbukti mampu menemukan jalur optimal terpendek tanpa terdistorsi oleh metode pencarian mendalam berulang yang digunakannya.

2. Efisiensi Penggunaan Memori

Keunggulan utama IDA* terlihat pada efisiensi konsumsi memori puncak, terutama ketika dimensi grid ditingkatkan secara signifikan. Pada lingkungan berskala besar (100 × 100 dengan kepadatan rintangan 30%), BFS mengonsumsi memori hingga 312,0 MB dan A* memerlukan 184,2 MB. Konsumsi memori yang masif pada BFS dan A* disebabkan oleh penumpukan simpul pada struktur antrean *Open List* dan pencatatan riwayat pada *Closed List*.

Sebaliknya, IDA* hanya membutuhkan memori puncak sebesar 25,8 MB pada kondisi pengujian yang sama. Hal ini memberikan penghematan alokasi memori sebesar 85,9% dibandingkan A* dan 91,7% dibandingkan BFS. Karakteristik ini didapatkan karena IDA* mengandalkan penelusuran mendalam (*depth-first search*) berbasis tumpukan (*call stack*), di mana kebutuhan memori hanya tumbuh

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.443
Page 813-823

secara linier $O(d)$ terhadap kedalaman solusi, bukan proporsional secara eksponensial terhadap luas area grid yang dieksplorasi.

3. Beban Waktu Eksekusi dan Jumlah Simpul Dikunjungi

Meskipun unggul dalam penghematan ruang memori, IDA* memiliki keterbatasan mendasar pada waktu komputasi saat menghadapi lingkungan padat rintangan. Pada grid 100 × 100 dengan kepadatan 30%, IDA* mengekspansi hingga 16.840 simpul dengan total waktu eksekusi 386,4 ms, sedangkan A* hanya membutuhkan 3.240 simpul dan waktu 112,5 ms.

Peningkatan drastis jumlah simpul dan waktu pada IDA* dipicu oleh mekanisme pembaruan ambang batas biaya ($f(n)$). Setiap kali nilai ambang dinaikkan ke batas terendah berikutnya, algoritma harus membuang riwayat simpul sebelumnya dan melakukan pencarian ulang (*re-expansion*) dari simpul awal. Pada lingkungan dengan konfigurasi rintangan bertipe koridor sempit atau rute buntu (*dead-end*), simpul yang sama dieksplorasi berulang kali pada setiap siklus iterasi, yang berdampak langsung pada lonjakan waktu eksekusi.

4. Relevansi Simulator dan Pengembangan Perencanaan Jalur Drone (UAV)

Implementasi simulator pada grid dua dimensi dengan empat arah pergerakan dan biaya langkah seragam ini diposisikan secara eksplisit sebagai bukti konsep (*proof of concept*) untuk memverifikasi logika komputasi, mengonfirmasi optimalitas panjang jalur, serta membuktikan efisiensi alokasi memori algoritma IDA* pada lingkungan diskret. Pemodelan 2D terkontrol ini sangat bermanfaat sebagai media evaluasi dasar sebelum algoritma diterapkan pada wahana fisik drone.

Temuan eksperimen menunjukkan bahwa keunggulan memori IDA* sangat tepat diimplementasikan pada komputer penerbangan *on-board* atau mikrokontroler dengan memori terbatas (seperti modul berbasis ARM mikro atau ESP32), sehingga meminimalkan risiko *system crash* akibat kehabisan memori (*out-of-memory*). Sementara itu, A* lebih unggul pada sistem dengan modul komputasi berkapasitas memori longgar yang menuntut responsivitas tinggi.

Namun, lintasan garis lurus pada grid 2D belum mewakili dinamika penerbangan drone di lingkungan nyata. Konsep simulator IDA* ini dapat diperluas untuk navigasi UAV sebenarnya melalui beberapa aspek berikut:

a. **Representasi Tiga Dimensi (3D Grid / *Octree*):** Menambahkan dimensi ketinggian (z) sehingga koordinat sel berubah menjadi (x, y, z). Hal ini memungkinkan drone merencanakan lingkungan kompleks seperti penarikan jalur ruang udara [11]. Fungsi heuristik Manhattan dapat diperluas menjadi jarak Euclidean 3D pada persamaan (4):

$$h(n) = \sqrt{(x_n + x_t)^2 + (y_n + y_t)^2 + (z_n + z_t)^2} \quad (4)$$

Hal ini memungkinkan drone merencanakan manuver menanjak (*climbing*), melayang mendatar, maupun menukik (*descending*).

b. **Batasan Radius Belok (*Turning Radius*):** Drone riil tidak dapat berbelok dengan sudut tajam 90° secara seketika. Jalur hasil pencarian diskret IDA* perlu dihaluskan (*path smoothing*) menggunakan kurva kontinuitas seperti kurva Bézier agar sesuai dengan dinamika manuver wahana di udara [13].

c. **Batas Kecepatan dan Akselerasi:** Kecepatan terbang (v) dan percepatan sentripetal maksimum (a_{max})pada setiap segmen tikungan harus dibatasi agar stabilitas aerodinamika drone tetap terjaga saat menghindari manuver ekstrem .

d. **Zona Aman terhadap Rintangan (*Safety Margin Buffer*):** Dalam penerbangan riil, jalur drone tidak boleh menempel langsung pada sisi rintangan untuk menghindari tubrukan akibat tiupan angin atau eror pembacaan sensor navigasi. Sel di sekeliling rintangan perlu diberi perluasan zona aman (*obstacle inflation*) dengan penalti biaya tambahan atau dijadikan zona terlarang (*no-fly zone*) [14].

e. **Konsumsi Energi Baterai:** Fungsi biaya langkah dapat dimodifikasi dari biaya seragam menjadi fungsi multi-objektif dengan memperhitungkan penggunaan energi baterai:$c(n, n') = w_1 \cdot jarak + w_2 \cdot \Delta Zelevasi + w_3 \cdot \Delta\theta_{belokan}$

di mana manuver vertikal dan perubahan arah sudut diberi penalti biaya lebih tinggi karena mengonsumsi daya dorong motor listrik yang lebih besar [12], [15].

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.443
Page 813-823

4. KESIMPULAN

Penelitian ini telah berhasil membangun dan menguji *Drone Pathfinding Simulator* berbasis JavaFX dengan menerapkan algoritma *Iterative Deepening A** (IDA*) pada lingkungan grid dua dimensi statis. Simulator ini memfasilitasi pembuatan dinding rintangan secara bebas (*draw line*), penetapan posisi awal dan sasaran drone, serta visualisasi proses penemuan rute dan pergerakan wahana secara interaktif. Hasil pengujian eksperimental kuantitatif pada variasi ukuran grid (20 × 20, 50 × 50, 100 ×100) dan kepadatan rintangan (10% dan 30%) membuktikan bahwa IDA*, BFS, dan A* konsisten menghasilkan panjang lintasan terpendek yang optimal berkat sifat *admissible* heuristik Manhattan. Keunggulan utama IDA* terletak pada efisiensi memori yang sangat tinggi, dengan konsumsi memori puncak hanya sebesar 25,8 MB pada grid berskala besar (100 × 100, rintangan 30%), atau mampu menghemat alokasi memori hingga 85,9% dibandingkan A* (184,2 MB) dan 91,7% dibandingkan BFS (312,0 MB). Namun demikian, efisiensi memori tersebut membawa konsekuensi berupa lonjakan waktu eksekusi (386,4 ms) dan jumlah simpul yang diekspansi (16.840 simpul) akibat penelusuran ulang (*re-expansion*) simpul pada setiap pembaruan ambang biaya. Dengan karakteristik tersebut, IDA* sangat layak direkomendasikan untuk sistem navigasi drone berbasis komputer *on-board* atau mikrokontroler dengan keterbatasan memori RAM, sedangkan A* tetap menjadi pilihan utama untuk kebutuhan navigasi dinamis berkecepatan tinggi. Pengembangan selanjutnya dapat diarahkan pada perluasan simulator menuju lingkungan tiga dimensi (x, y, z) dengan memperhitungkan radius putar wahana, dinamika kecepatan, serta konsumsi energi baterai.

PERNYATAAN KONTRIBUSI PENULIS (CREDIT)

Fitria Sari: Penulisan – Draf Awal, Perangkat lunak, Metodologi, Konseptualisasi, Kurasi Data, Investigasi, Analisis Formal, Visualisasi. **Sugiarto Cokrowibowo:** Penulisan – Peninjauan dan Penyuntingan, Validasi, Supervisi, Visualisasi. **A.Amirul Asnan Cirua:** Supervisi, Validasi, Penulisan – Peninjauan dan Penyuntingan.

PERNYATAAN KONFLIK KEPENTINGAN

Para penulis menyatakan bahwa tidak terdapat kepentingan finansial atau hubungan pribadi yang diketahui yang dapat memengaruhi hasil penelitian yang dilaporkan dalam artikel ini.

PERNYATAAN PENDANAAN

Penelitian ini tidak memperoleh dukungan pendanaan dari pihak mana pun. Biaya yang timbul dalam proses pengajuan dan publikasi artikel ditanggung secara pribadi oleh penulis.

PERNYATAAN PENGGUNAAN AI GENERATIF DAN TEKNOLOGI BERBANTUAN AI DALAM PROSES PENULISAN

Para penulis menggunakan ChatGPT-6. untuk meningkatkan kejelasan penulisan artikel ini. Seluruh konten yang dibantu oleh AI telah ditinjau dan disunting oleh penulis, dan para penulis bertanggung jawab sepenuhnya atas isi akhir yang dipublikasikan.

REFERENCES

[1] W. Meng, X. Zhang, L. Zhou, H. Guo, and X. Hu, "Advances in UAV Path Planning: A Comprehensive Review of Methods, Challenges, and Future Directions," *Drones*, vol. 9, no. 5, pp. 1–46, 2025, doi: 10.3390/drones9050376.

[2] D. Debnath, F. Vanegas, J. Sandino, A. F. Hawary, and F. Gonzalez, "A Review of UAV Path-

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Planning Algorithms and Obstacle Avoidance Methods for Remote Sensing Applications," *Remote Sens.*, vol. 16, no. 21, 2024, doi: 10.3390/rs16214019.

[3] Z. Zhang, "A review of unmanned aerial vehicle path planning techniques," *Appl. Comput. Eng.*, vol. 33, no. 1, pp. 234–241, 2024, doi: 10.54254/2755-2721/33/20230275.

[4] D. Alaa, K. Abdelmadjid, and L. Fadhila, "Intelligent path planning algorithms for UAVs: Classification, complexity analysis, hybrid ablation insights, and future directions," *Adv. Mech. Eng.*, vol. 17, no. 7, pp. 1–26, 2025, doi: 10.1177/16878132251355020.

[5] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, "A Systematic Literature Review of A*Pathfinding," *Procedia Comput. Sci.*, vol. 179, no. 2020, pp. 507–514, 2021, doi: 10.1016/j.procs.2021.01.034.

[6] X. Bai, H. Jiang, J. Cui, K. Lu, P. Chen, and M. Zhang, "UAV Path Planning Based on Improved A ∗ and DWA Algorithms," *Int. J. Aerosp. Eng.*, vol. 2021, 2021, doi: 10.1155/2021/4511252.

[7] G. Aiello, K. P. Valavanis, and A. Rizzo, "Fixed-Wing UAV Energy Efficient 3D Path Planning in Cluttered Environments," *J. Intell. Robot. Syst. Theory Appl.*, vol. 105, no. 3, 2022, doi: 10.1007/s10846-022-01608-1.

[8] M. shun Yuan, T. le Zhou, and M. Chen, "Improved lazy theta∗ algorithm based on octree map for path planning of UAV," *Def. Technol.*, vol. 23, pp. 8–18, 2023, doi: 10.1016/j.dt.2022.01.006.

[9] R. E. Korf, "Depth-first iterative-deepening. An optimal admissible tree search," *Artif. Intell.*, vol. 27, no. 1, pp. 97–109, 1985, doi: 10.1016/0004-3702(85)90084-0.

[10] W. Jiang, "Analysis of iterative deepening A* algorithm," *IOP Conf. Ser. Earth Environ. Sci.*, vol. 693, no. 1, 2021, doi: 10.1088/1755-1315/693/1/012028.

[11] Y. He, T. Hou, and M. Wang, "A new method for unmanned aerial vehicle path planning in complex environments," *Sci. Rep.*, vol. 14, no. 1, pp. 1–12, 2024, doi: 10.1038/s41598-024-60051-4.

[12] F. Yang *et al.*, "12]Obstacle Avoidance Path Planning for UAV Based on Improved RRT Algorithm," *Discret. Dyn. Nat. Soc.*, vol. 2022, 2022, doi: 10.1155/2022/4544499.

[13] Z. Zhang, X. Liu, and B. Feng, "Research on obstacle avoidance path planning of UAV in complex environments based on improved Bézier curve," *Sci. Rep.*, vol. 13, no. 1, pp. 1–10, 2023, doi: 10.1038/s41598-023-43783-7.

[14] M. B. Sushma, B. Mashhoodi, W. Tan, K. Liujiang, and Q. Xu, "Spatial drone path planning: A systematic review of parameters and algorithms," *J. Transp. Geogr.*, vol. 125, no. January, p. 104209, 2025, doi: 10.1016/j.jtrangeo.2025.104209.

[15] A. Sonny, S. R. Yeduri, and L. R. Cenkeramaddi, "Q-learning-based unmanned aerial vehicle path planning with dynamic obstacle avoidance," *Appl. Soft Comput.*, vol. 147, p. 110773, 2023, doi: 10.1016/j.asoc.2023.110773.

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/