INSECT: Jurnal Teknik Informatika
INFORMATICS AND SECURITY e-ISSN. 2614-431X

Penerapan Algoritma *A-Star* sebagai Metode *Pathfinding* pada *Game* Lokasi Logika Cerdas untuk *Non-Player Character*

Kevin Gibran*[1], Rudi Heriansyah[2], Zaid Romegar Mair[3]
[1,2,3]Program Studi Teknik Informatika, Universitas Indo Global Mandiri
E-mail: *[1]kevgib46@gmail.com, [2]rudi@uigm.ac.id, [3]zaidromegar@uigm.ac.id

Abstrak

Perkembangan game simulasi modern menuntut sistem navigasi Non-Player Character (NPC) yang efisien dan adaptif dalam lingkungan virtual yang kompleks. Algoritma A-Star (A) banyak digunakan dalam pathfinding karena mengombinasikan biaya aktual dan estimasi heuristic untuk menghasilkan jalur optimal. Namun, performanya dipengaruhi oleh jenis heuristic yang digunakan. Penelitian ini bertujuan mengimplementasikan A* pada sistem navigasi NPC berbasis Roblox Studio serta menganalisis kinerja Manhattan Distance dan Euclidean Distance. Metode yang digunakan adalah eksperimen komputasional pada tiga tingkat kompleksitas map (25×25, 50×50, dan 75×75 node) dengan dua mode pergerakan, yaitu 4-arah dan 8-arah. Parameter evaluasi meliputi waktu pencarian, panjang jalur, dan tingkat optimalitas. Hasil menunjukkan bahwa Manhattan Distance lebih efisien dalam waktu pencarian, terutama pada map Level 3 mode 8-arah (1.220,1 ms dibandingkan 6.990,5 ms pada Euclidean Distance). Sebaliknya, Euclidean Distance menghasilkan jalur lebih pendek dan optimal pada mode 8-arah (99 node dibandingkan 106 node pada Manhattan Distance). Temuan ini menunjukkan adanya trade-off, yaitu tidak bisa mendapatkan semuanya sekaligus secara maksimal antara efisiensi waktu dan kualitas jalur. Penelitian ini memberikan dasar empiris dalam pemilihan heuristic yang sesuai untuk sistem navigasi NPC tiga dimensi.*

Kata kunci— *Game, A-Star, Pathfinding, Non-Player Character, Heuristic*

1. PENDAHULUAN

Perkembangan *game* digital modern tidak lagi terbatas pada aspek hiburan, tetapi telah berkembang menjadi media simulasi, edukasi, dan eksperimen teknologi interaktif [1]. Perkembangan teknologi juga memberikan peluang besar dalam peningkatan kualitas pembelajaran berbasis digital dan interaktif [2]. Lingkungan simulasi *game* saat ini dikembangkan menggunakan berbagai *platform* [3] seperti *Roblox*, *Edublock*, dan *platform* berbasis *engine* lainnya yang mendukung integrasi kecerdasan buatan dalam lingkungan virtual.

Seiring berkembangnya teknologi komputasi, kecerdasan buatan (*Artificial Intelligence*) semakin banyak dimanfaatkan untuk membangun sistem yang mampu melakukan analisis dan pengambilan keputusan secara otomatis. Berbagai penelitian menunjukkan bahwa penerapan algoritma dalam sistem berbasis kecerdasan buatan mampu membantu proses analisis data dan pengenalan pola secara sistematis. Penelitian oleh Kanaka dkk. (2024), menggunakan algoritma *Decision Tree* dan *Support Vector Machine* untuk membantu proses pengambilan keputusan dalam pemilihan mahasiswa penerima program KIP-K [4]. Selain itu, penelitian yang dilakukan oleh Mair dkk. (2022), memanfaatkan algoritma *Convolutional Neural Network* (CNN) untuk menganalisis pola tulisan tangan dalam mengidentifikasi bakat anak serta karakteristik kepribadian berdasarkan pola penulisan [5], [6]. Dalam konteks pengembangan *game*, kecerdasan

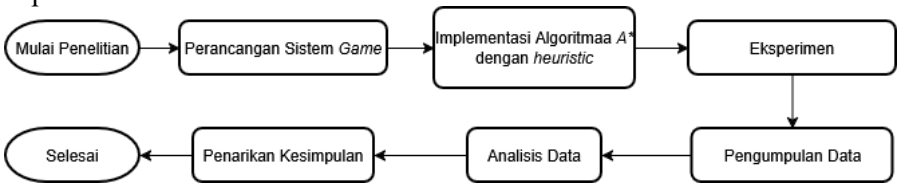
buatan digunakan untuk menciptakan perilaku karakter virtual yang lebih realistis, salah satunya melalui pengembangan sistem navigasi pada *Non-Player Character* (NPC).

Salah satu komponen penting dalam navigasi NPC adalah sistem *pathfinding*, yaitu mekanisme pencarian jalur dari titik awal menuju tujuan secara efisien dalam lingkungan virtual yang kompleks [7]. Salah satu algoritma yang paling banyak digunakan dalam *pathfinding* adalah *A-Star* (A*), karena kemampuannya mengombinasikan biaya aktual dan estimasi *heuristic* untuk menemukan jalur optimal [8]. Beberapa penelitian menunjukkan bahwa algoritma A* memiliki performa yang lebih efisien dibandingkan algoritma lain seperti *Dijkstra dan Breadth-First Search* (BFS) dalam proses pencarian jalur pada lingkungan *game* [9], [10]. Kinerja algoritma A* sendiri sangat dipengaruhi oleh fungsi *heuristic* yang digunakan dalam proses estimasi jarak menuju tujuan. Dua fungsi *heuristic* yang umum digunakan dalam sistem berbasis *grid* adalah *Manhattan Distance* dan *Euclidean Distance*. *Manhattan Distance* memiliki kompleksitas perhitungan yang lebih sederhana karena hanya melibatkan operasi aritmatika dasar [11], sedangkan *Euclidean Distance* merepresentasikan jarak geometris langsung antara dua titik sehingga lebih sesuai digunakan pada sistem yang mengizinkan pergerakan *diagonal*.

Berbagai penelitian sebelumnya telah menerapkan algoritma A* dalam sistem *pathfinding* pada lingkungan *game* maupun sistem navigasi digital. Penelitian oleh Agung dkk. (2022) mengimplementasikan algoritma A* untuk menentukan jalur karakter secara otomatis dalam lingkungan permainan [7]. Rizki dkk. (2025) juga menunjukkan bahwa algoritma A* memiliki performa pencarian jalur yang lebih efisien dibandingkan algoritma *Dijkstra* pada sistem navigasi [8], sedangkan Affandi dkk. (2022) melakukan perbandingan beberapa algoritma *pathfinding* dalam navigasi karakter virtual untuk memperoleh jalur optimal [9]. Meskipun demikian, sebagian besar penelitian tersebut masih berfokus pada perbandingan antar algoritma atau implementasi *pathfinding* secara umum. Kajian yang secara khusus menganalisis pengaruh variasi fungsi *heuristic* terhadap performa algoritma A* pada lingkungan simulasi tiga dimensi masih relatif terbatas, khususnya pada *platform Roblox Studio*. Oleh karena itu, penelitian ini bertujuan untuk mengimplementasikan algoritma A* pada sistem navigasi NPC berbasis *Roblox Studio* serta menganalisis pengaruh penggunaan *heuristic Manhattan Distance* dan *Euclidean Distance* terhadap kinerja *pathfinding* pada tiga tingkat kompleksitas *map* dengan dua mode pergerakan, yaitu 4-arah dan 8-arah, berdasarkan parameter waktu pencarian jalur, panjang jalur, dan tingkat optimalitas jalur.

2. METODE PENELITIAN

Penelitian ini dilaksanakan melalui beberapa tahapan sistematis yang dirancang untuk menguji performa algoritma *A-Star* (A*) dalam sistem navigasi *Non-Player Character* (NPC) berbasis *Roblox Studio*. Tahapan penelitian dimulai dari perancangan sistem *game* sebagai lingkungan simulasi, implementasi algoritma A* dengan dua *heuristic* yang diuji, pelaksanaan eksperimen terkontrol, pengumpulan data berbasis *logging* otomatis, analisis data secara deskriptif, dan diakhiri dengan penarikan kesimpulan berdasarkan hasil empiris. Seperti ditunjukkan pada Gambar 1.



Gambar 1 Diagram Alur Penelitian

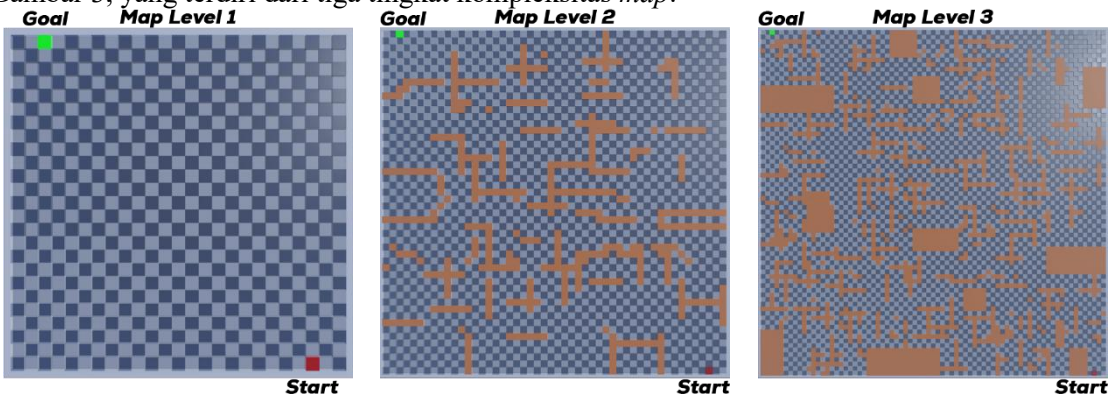
2.1 Perancangan Sistem Game

Perancangan sistem *game* pada penelitian ini dilakukan dengan membangun lingkungan simulasi tiga dimensi berbasis *grid* menggunakan *Roblox* Studio, yaitu *platform* pengembangan game yang memungkinkan pembuatan lingkungan virtual interaktif serta integrasi logika pemrograman untuk simulasi karakter digital [12]. Sistem dirancang untuk merepresentasikan

ruang navigasi NPC secara terstruktur, sehingga memudahkan implementasi algoritma *A** dan pengujian fungsi *heuristic*. NPC yang digunakan dalam penelitian ini berupa karakter *humanoid* standar *Roblox* yang bertindak sebagai agen otonom dalam eksperimen. Karakter tersebut tidak dikendalikan oleh pemain, melainkan bergerak secara otomatis [9] berdasarkan jalur yang dihasilkan oleh algoritma A*. Tampilan NPC dalam lingkungan simulasi ditunjukkan pada Gambar 2.

Gambar 2 NPC

Lingkungan navigasi dalam penelitian ini dibangun menggunakan sistem *grid* persegi yang merepresentasikan *node-node* pencarian jalur. Desain *map* pengujian ditunjukkan pada Gambar 3, yang terdiri dari tiga tingkat kompleksitas *map*.



Gambar 3 Desain *Map Level* 1, *Level* 2, dan *Level* 3

Pada *map level* 1, *map* dirancang menggunakan *grid* berukuran 25 × 25 *node* tanpa rintangan sebagai *baseline* pengujian. Struktur *grid* yang bersih memungkinkan NPC bergerak langsung dari titik awal yang ditandai warna merah menuju titik tujuan berwarna hijau tanpa hambatan, sehingga jalur yang dihasilkan merepresentasikan kondisi ideal.

Pada *map level* 2, *map* dikembangkan dengan ukuran 50 × 50 *node* dan ditambahkan rintangan statis pada beberapa titik dalam *grid* yang ditandai dengan warna coklat. Keberadaan rintangan ini memaksa algoritma A* untuk melakukan eksplorasi jalur alternatif dalam proses pencarian jalur menuju tujuan.

Sementara itu, *map level* 3 dirancang dengan tingkat kompleksitas tertinggi menggunakan *grid* 75 × 75 *node* dengan jumlah rintangan yang lebih padat dan variasi bentuk hambatan yang lebih beragam. Konfigurasi ini menghasilkan ruang pencarian yang lebih luas sehingga meningkatkan beban komputasi algoritma dalam menentukan jalur optimal.

*2. 2 Implementasi Algoritma A**

Algoritma A* merupakan algoritma pencarian jalur berbasis graf yang menggabungkan biaya aktual dari *node* awal dengan estimasi jarak menuju tujuan menggunakan fungsi *heuristic*. Algoritma ini banyak digunakan dalam sistem navigasi dan *pathfinding* karena kemampuannya menemukan jalur optimal secara efisien [13]. Algoritma A* diimplementasikan menggunakan fungsi evaluasi pada persamaan (1):

$$F(n) = G(n) + H(n) \quad (1)$$

Dimana:

f(n): nilai total evaluasi dari *node* n

g(n): biaya riil (*cost*) dari *node* awal ke *node* n

h(n): estimasi biaya terpendek (*heuristic*) dari *node* n ke tujuan

Dua jenis *heuristic* yang diuji adalah *Manhattan Distance* dan *Euclidean Distance*. *Manhattan Distance* digunakan untuk merepresentasikan pergerakan *orthogonal* (4-arah), sedangkan *Euclidean Distance* merepresentasikan estimasi jarak garis lurus antar *node* dalam ruang *Euclidean* sehingga sering digunakan pada algoritma pencarian jalur yang mengizinkan pergerakan *diagonal* [14].Implementasi dilakukan melalui bahasa pemrograman *Luau*, yaitu bahasa skrip yang digunakan dalam *Roblox Studio* untuk membangun logika permainan dan perilaku karakter dalam lingkungan virtual, serta diintegrasikan dengan sistem *logging* untuk pencatatan data eksperimen secara otomatis [15].

2.3 Eksperimen

Eksperimen dilakukan menggunakan pendekatan kuantitatif berbasis simulasi komputasional untuk mengevaluasi kinerja algoritma A* dalam sistem navigasi NPC. Pengujian dilaksanakan pada tiga tingkat kompleksitas *map*. Pada masing-masing *level*, algoritma A* dijalankan menggunakan dua jenis *heuristic*, yaitu *Manhattan Distance* dan *Euclidean Distance*, serta dua mode pergerakan NPC, yaitu 4-arah (*orthogonal*) dan 8-arah (*diagonal*).

Evaluasi kinerja dilakukan berdasarkan tiga indikator utama, yaitu waktu pencarian jalur dalam milidetik (*ms*), panjang jalur yang ditempuh dalam jumlah *node*, serta tingkat optimalitas jalur dalam persentase. Tingkat optimalitas jalur dihitung berdasarkan perbandingan antara panjang jalur yang dihasilkan algoritma dengan panjang jalur optimal secara teoritis. Persentase optimalitas dihitung menggunakan persamaan (2):

$$Tingkat\ Optimalitas\ Jalur\ (\%) = \left(\frac{Panjang\ Jalur\ Optimal}{Panjang\ Jalur\ Aktual}\right) \times 100 \quad (2)$$

Dimana:

Panjang Jalur Optimal: jumlah *node* minimum dari *start* ke *goal.*

Panjang Jalur Aktual: jumlah *node* yang benar-benar dilalui oleh NPC pada hasil eksperimen.

Seluruh pengujian dilakukan dalam lingkungan simulasi yang terkontrol dengan konfigurasi sistem yang konsisten, sehingga hasil yang diperoleh merepresentasikan karakteristik performa masing-masing *heuristic* pada setiap tingkat kompleksitas *map*.

2.4 Pengumpulan Data

Data eksperimen dikumpulkan secara otomatis melalui mekanisme *logging* internal pada *Roblox Studio*. Sistem mencatat durasi pencarian jalur sejak algoritma dijalankan hingga jalur ditemukan, jumlah *node* yang dilalui oleh NPC, serta nilai optimalitas jalur berdasarkan perbandingan antara jalur aktual dan jalur optimal secara teoritis. Data hasil pengujian kemudian diekstraksi dan disusun dalam format tabel terstruktur untuk memudahkan proses analisis lanjutan.

2.5 Analisis Data

Analisis data dilakukan menggunakan teknik statistik deskriptif untuk membandingkan performa kedua *heuristic* pada setiap *level map* dan mode pergerakan. Nilai rata-rata waktu pencarian, panjang jalur, dan persentase optimalitas dihitung untuk masing-masing skenario. Hasil analisis divisualisasikan dalam bentuk tabel dan grafik guna mengidentifikasi pola perbedaan kinerja antara *Manhattan Distance* dan *Euclidean Distance* dalam berbagai kondisi kompleksitas lingkungan.

2.6 Penarikan Kesimpulan

Kesimpulan penelitian ditarik berdasarkan hasil analisis empiris terhadap seluruh parameter evaluasi. Interpretasi hasil dilakukan dengan mengacu pada tujuan penelitian, yaitu mengevaluasi kinerja algoritma A* menggunakan dua *heuristic* berbeda dalam sistem navigasi NPC berbasis *Roblox Studio*. Temuan penelitian digunakan untuk memberikan rekomendasi pemilihan *heuristic* yang sesuai dengan karakteristik lingkungan dan kebutuhan sistem navigasi.

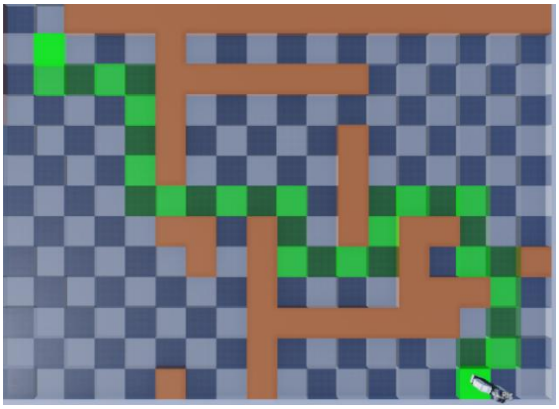
3. HASIL DAN PEMBAHASAN

3.1 Implementasi Pergerakan Non-Player Character (NPC)

Implementasi pergerakan *NPC* bertujuan untuk merealisasikan hasil pencarian jalur algoritma *A-Star* (A*) ke dalam simulasi visual berbasis *grid* pada *Roblox Studio*. Setelah algoritma menentukan urutan *node* dari titik awal (*start*) menuju titik tujuan (*goal*), NPC bergerak secara otomatis mengikuti jalur tersebut secara berurutan hingga mencapai target akhir.

Pada Gambar 4 memperlihatkan visualisasi pergerakan NPC yang mengikuti jalur hasil perhitungan algoritma. Untuk memperjelas proses evaluasi antar *node*, Gambar 5 menampilkan distribusi nilai G, H, dan F pada setiap *node grid*. Pada visualisasi tersebut, warna merah menunjukkan titik awal, hijau gelap menunjukkan titik tujuan, coklat merepresentasikan rintangan (*obstacle*), biru menunjukkan *node* yang masuk dalam proses eksplorasi (*open list*), dan hijau terang menandakan jalur akhir dengan biaya total terendah yang berhasil dipilih algoritma.

Pada area *grid* tanpa rintangan, terlihat bahwa beberapa *node* memiliki nilai evaluasi *F* yang identik. Kondisi ini menghasilkan lebih dari satu kandidat *node* dengan prioritas yang sama pada *open list*. Fenomena ini dikenal sebagai *symmetric paths*, yaitu kondisi ketika beberapa jalur alternatif memiliki total biaya yang setara menuju tujuan. Karena algoritma A* tidak memiliki aturan tambahan untuk memprioritaskan arah geometris tertentu saat terjadi nilai evaluasi yang sama, pemilihan *node* dilakukan berdasarkan urutan pemrosesan pada struktur data antrian. Akibatnya, algoritma secara bergantian memilih pergerakan pada sumbu horizontal dan vertikal.



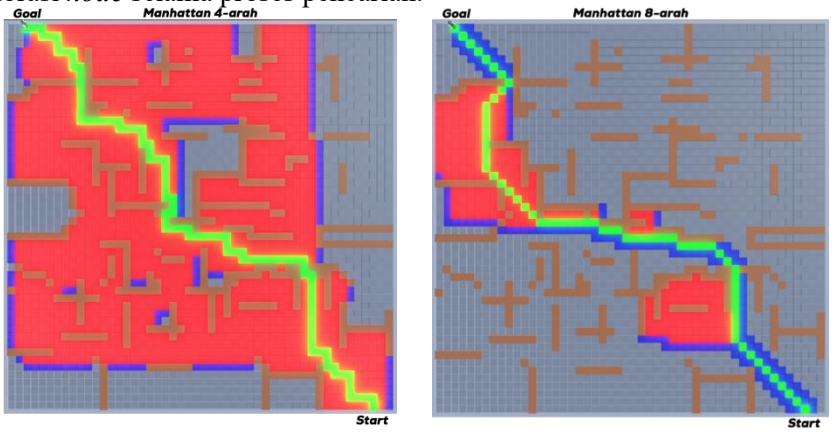
Gambar 4 Pergerakan NPC

412,-100 G=8 H=108 F=116	408,-100 G=4 H=104 F=108																
412,-104 G = 4 H=104 F=108	408,-104 Posisi Start	404,-104 G=4 H=96 F=100	400,-104 G=8 H=92 F=100	396,-104 G=12 H=88 F=100		388,-104 G=124 H=80 F=204	384,-104 G=120 H=76 F=196	380,-104 G=116 H=72 F=188	376,-104 G=112 H=68 F=180	372,-104 G=108 H=64 F=172	368,-104 G=104 H=60 F=164	364,-104 G=100 H=56 F=156	360,-104 G=104 H=52 F=156	356,-104 G=108 H=48 F=156	352,-104 G=112 H=44 F=156	348,-104 G=116 H=48 F=164	344,-104 G=120 H=52 F=172
412,-108 G=8 H=100 F=108	408,-108 G=4 H=96 F=100	404,-108 G=8 H=92 F=100	400,-108 G=12 H=88 F=100	396,-108 G=16 H=84 F=100								364,-108 G=96 H=52 F=148	360,-108 G=100 H=48 F=148	356,-108 G=104 H=44 F=148	352,-108 G=108 H=40 F=148	348,-108 G=112 H=44 F=156	344,-108 G=116 H=48 F=164
412,-112 G = 12 H=96 F=108	408,-112 G=8 H=92 F=100	404,-112 G=12 H=88 F=100	400,-112 G=16 H=84 F=100	396,-112 G=20 H=80 F=100		388,-112 G=52 H=72 F=124	384,-112 G=56 H=68 F=124	380,-112 G=60 H=64 F=124	376,-112 G=64 H=60 F=124	372,-112 G=68 H=56 F=124	368,-112 G=72 H=52 F=124	364,-112 G=92 H=48 F=140	360,-112 G=96 H=44 F=140	356,-112 G=100 H=40 F=140	352,-112 G=104 H=36 F=140	348,-112 G=108 H=40 F=148	344,-112 G=112 H=44 F=156
412,-116 G = 16 H=92 F=108	408,-116 G=12 H=88 F=100	404,-116 G=16 H=84 F=100	400,-116 G=20 H=80 F=100	396,-116 G=24 H=76 F=100		388,-116 G=48 H=68 F=116	384,-116 G=52 H=64 F=116	380,-116 G=56 H=60 F=116	376,-116 G=60 H=56 F=116	372,-116 G=64 H=52 F=116		364,-116 G=88 H=44 F=132	360,-116 G=92 H=40 F=132	356,-116 G=96 H=36 F=132	352,-116 G=100 H=32 F=132	348,-116 G=104 H=36 F=140	344,-116 G=108 H=40 F=148
412,-120 G = 20 H=88 F=108	408,-120 G=16 H=84 F=100	404,-120 G=20 H=80 F=100	400,-120 G=24 H=76 F=100	396,-120 G=28 H=72 F=100		388,-120 G=44 H=64 F=108	384,-120 G=48 H=60 F=108	380,-120 G=52 H=56 F=108	376,-120 G=56 H=52 F=108	372,-120 G=60 H=48 F=108		364,-120 G=84 H=40 F=124	360,-120 G=88 H=36 F=124	356,-120 G=92 H=32 F=124	352,-120 G=96 H=28 F=124	348,-120 G=100 H=32 F=132	344,-120 G=104 H=36 F=140
412,-124 G = 24 H=84 F=108	408,-124 G=20 H=80 F=100	404,-124 G=24 H=76 F=100	400,-124 G=28 H=72 F=100	396,-124 G=32 H=68 F=100	392,-124 G=36 H=64 F=100	388,-124 G=40 H=60 F=100	384,-124 G=44 H=56 F=100	380,-124 G=48 H=52 F=100	376,-124 G=52 H=48 F=100	372,-124 G=56 H=44 F=100		364,-124 G=80 H=36 F=116	360,-124 G=84 H=32 F=116	356,-124 G=88 H=28 F=116	352,-124 G=92 H=24 F=116	348,-124 G=96 H=28 F=124	344,-124 G=100 H=32 F=132
412,-128 G = 28 H=80 F=108	408,-128 G=24 H=76 F=100	404,-128 G=28 H=72 F=100	400,-128 G=32 H=68 F=100	396,-128 G=36 H=64 F=100			384,-128 G=48 H=52 F=100		376,-128 G=56 H=44 F=100	372,-128 G=60 H=40 F=100		364,-128 G=76 H=32 F=108			352,-128 G=96 H=20 F=116	348,-128 G=100 H=24 F=124	344,-128 G=104 H=28 F=132
412,-132 G = 32 H=76 F=108	408,-132 G=28 H=72 F=100	404,-132 G=32 H=68 F=100	400,-132 G=36 H=64 F=100	396,-132 G=40 H=60 F=100	392,-132 G=44 H=56 F=100		384,-132 G=52 H=48 F=100		376,-132 G=60 H=40 F=100	372,-132 G=64 H=36 F=100	368,-132 G=68 H=32 F=100	364,-132 G=72 H=28 F=100		356,-132 G=104 H=20 F=124	352,-132 G=100 H=16 F=116	348,-132 G=104 H=20 F=124	
412,-136 G = 36 H=72 F=108	408,-136 G=32 H=68 F=100	404,-136 G=36 H=64 F=100	400,-136 G=40 H=60 F=100	396,-136 G=44 H=56 F=100	392,-136 G=48 H=52 F=100	388,-136 G=52 H=48 F=100	384,-136 G=56 H=44 F=100		376,-136 G=64 H=36 F=100	372,-136 G=68 H=32 F=100	368,-136 G=72 H=28 F=100	364,-136 G=76 H=24 F=100				348,-136 G=108 H=16 F=124	344,-136 G=112 H=20 F=132
412,-140 G = 40 H=68 F=108	408,-140 G=36 H=64 F=100	404,-140 G=40 H=60 F=100	400,-140 G=44 H=56 F=100	396,-140 G=48 H=52 F=100	392,-140 G=52 H=48 F=100	388,-140 G=56 H=44 F=100	384,-140 G=60 H=40 F=100								352,-140 G=116 H=8 F=124	348,-140 G=112 H=12 F=124	344,-140 G=116 H=16 F=132
412,-144 G = 44 H=64 F=108	408,-144 G=40 H=60 F=100	404,-144 G=44 H=56 F=100	400,-144 G=48 H=52 F=100	396,-144 G=52 H=48 F=100	392,-140 G=56 H=44 F=100	388,-144 G=60 H=40 F=100	384,-144 G=64 H=36 F=100		376,-144 G=144 H=26 F=170	372,-144 G=140 H=22 F=164	368,-144 G=136 H=20 F=156	364,-144 G=132 H=16 F=148	360,-144 G=128 H=12 F=140	356,-144 G=124 H=8 F=132	352,-144 G=120 H=4 F=124	348,-144 G=116 H=8 F=124	344,-144 G=120 H=12 F=132
412,-148 G = 48 H=60 F=108	408,-148 G=44 H=56 F=100	404,-148 G=48 H=52 F=100	400,-148 G=52 H=48 F=100	396,-148 G=56 H=44 F=100		388,-148 G=64 H=36 F=100	384,-148 G=68 H=32 F=100		376,-148 G=148 H=30 F=178	372,-148 G=144 H=26 F=170	368,-148 G=140 H=24 F=164	364,-148 G=136 H=20 F=156	360,-148 G=132 H=16 F=148	356,-148 G=128 H=4 F=132	352,-148 G=124 H=0 F=124	348,-148 G=120 H=4 F=124	344,-148 G=124 H=8 F=132

Gambar 5 Perhitungan antar *Node* Algoritma A*

3.2 Implementasi Jalur Manhattan Distance

Implementasi *heuristic Manhattan Distance* menghasilkan jalur berdasarkan perhitungan selisih absolut koordinat horizontal dan vertikal ($|d_x| + |d_y|$). Pada visualisasi hasil implementasi, proses kerja algoritma A* tidak hanya ditunjukkan melalui jalur akhir, tetapi juga melalui eksplorasi *node* selama proses pencarian.



Gambar 6 Hasil Jalur Algoritma A* Menggunakan *Heuristic Manhattan Distance* 4-arah dan 8-arah

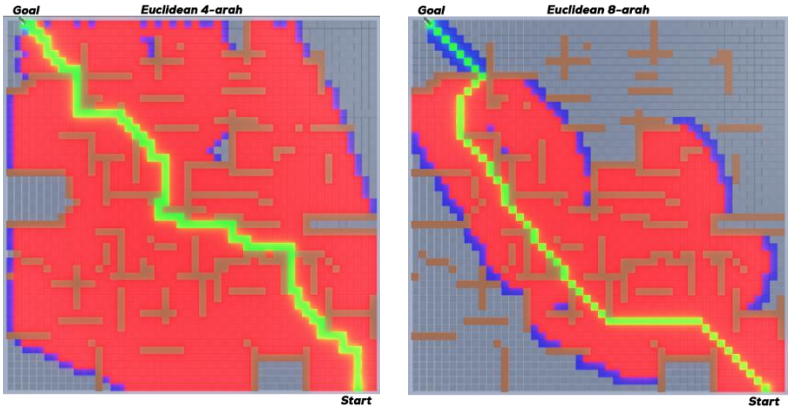
Pada Gambar 6 ditunjukkan visualisasi mode pergerakan 4-arah dan 8-arah. Warna merah merepresentasikan *closed list*, yaitu *node* yang telah dievaluasi oleh algoritma, sedangkan warna biru menunjukkan *open list*, yaitu *node* yang masih berada dalam antrian evaluasi. Jalur berwarna hijau *neon* merupakan lintasan akhir yang ditemukan dari titik start menuju *goal* dengan nilai evaluasi terbaik.

Pada mode 4-arah, jalur yang dihasilkan mengikuti pola *orthogonal* sesuai struktur *grid*. Eksplorasi *node* menyebar secara horizontal dan vertikal karena karakteristik *Manhattan Distance* yang merepresentasikan pergerakan siku-siku, sehingga jalur yang dihasilkan tetap optimal.

Pada mode 8-arah, NPC dapat bergerak secara *diagonal* sehingga jalur yang terbentuk menjadi lebih fleksibel dibandingkan mode 4-arah. Meskipun *Manhattan Distance* tidak sepenuhnya sesuai untuk pergerakan *diagonal*, algoritma A* tetap mampu menemukan jalur yang valid menuju tujuan.

3.3 Implementasi Jalur Euclidean Distance

Implementasi *heuristic Euclidean Distance* dilakukan dengan menghitung estimasi jarak garis lurus antar dua titik menggunakan pendekatan geometris $\sqrt{(dx^2 + dy^2)}$. Visualisasi proses pencarian ditampilkan melalui distribusi *node* selama algoritma berjalan, di mana warna merah menunjukkan *closed list node* yang telah dievaluasi, warna biru menunjukkan *open list node* yang masih dalam antrian evaluasi, dan jalur hijau neon merepresentasikan lintasan akhir yang berhasil ditemukan.



Gambar 7 Hasil Jalur Algoritma A* Menggunakan *Heuristic Euclidean Distance* 4-arah dan 8-arah

Pada Gambar 7 yaitu mode 4-arah dan 8-arah, terlihat bahwa distribusi *closed list* menyebar dalam pola yang relatif membulat dan terfokus mengarah ke *goal*. Meskipun NPC di pergerakan 4-arah hanya diperbolehkan bergerak secara horizontal dan vertikal, estimasi jarak *Euclidean Distance* tetap mengarah pada garis lurus menuju tujuan. Namun, karena pergerakan *diagonal* tidak diperbolehkan, jalur akhir tetap mengikuti pola *orthogonal* meskipun estimasi jaraknya berbasis geometri.

Pada mode 8-arah, terlihat bahwa jalur akhir membentuk lintasan *diagonal* yang lebih halus dan mendekati garis lurus geometris dari start menuju *goal*. Distribusi *closed list* juga tampak lebih terfokus mengikuti arah *diagonal* tersebut. Hal ini menunjukkan bahwa *Euclidean Distance* bekerja secara lebih representatif ketika sistem mengizinkan *diagonal movement*, karena estimasi *heuristic* sejalan dengan biaya aktual pergerakan.

*3.4 Hasil Pengujian Algoritma A**

Rekapitulasi hasil pengujian algoritma A* ditunjukkan pada Tabel 1 berikut.

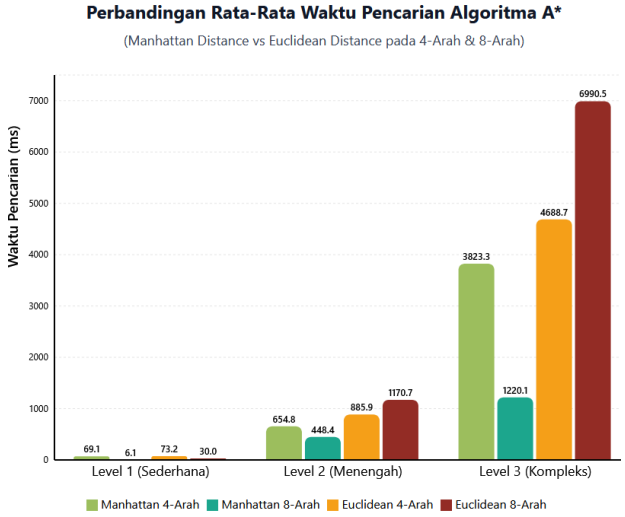
Tabel 1 Rekapitulasi hasil rata-rata pengujian algoritma *A**

Level Map	Jenis Heuristic	Mode Pergerakan	Rata-rata Waktu (*ms*)	Panjang Jalur (*node*)	Tingkat Optimalitas
Level 1	Manhattan Distance	4-arah	69,1	43	100%
		8-arah	6,1	24	100%
	Euclidean Distance	4-arah	73,2	43	100%
		8-arah	30,0	24	100%
Level 2	Manhattan Distance	4-arah	654,8	97	97,93%
		8-arah	448,4	70	90,00%
	Euclidean Distance	4-arah	885,9	97	97,93%
		8-arah	1.179,7	63	100%
Level 3	Manhattan Distance	4-arah	3.823,3	153	94,74%
		8-arah	1.220,1	106	93,39%
	Euclidean Distance	4-arah	4.688,7	153	94,74%
		8-arah	6.990,5	99	100%

Hasil menunjukkan peningkatan waktu pencarian seiring bertambahnya kompleksitas *map*, serta perbedaan performa antara kedua *heuristic* pada mode pergerakan yang berbeda.

3.5 Analisis Hasil Waktu Pencarian

Analisis waktu pencarian dilakukan untuk mengevaluasi efisiensi komputasi algoritma A* pada berbagai tingkat kompleksitas *map* dan mode pergerakan. Grafik pada Gambar 9 menunjukkan perbandingan rata-rata waktu pencarian antara *Manhattan Distance* dan *Euclidean Distance* pada mode 4-arah dan 8-arah.



Gambar 9 Grafik rata-rata waktu pencarian algoritma A*

Secara umum, waktu pencarian meningkat signifikan seiring bertambahnya kompleksitas *map* dari *Level* 1 (sederhana) hingga *Level* 3 (kompleks). Pada *Level* 1, seluruh konfigurasi menunjukkan waktu yang relatif rendah, dengan *Manhattan Distance* 8-arah sebagai yang tercepat yaitu 6,1 *ms*. Hal ini menunjukkan bahwa pada lingkungan sederhana dengan ruang

pencarian terbatas, perbedaan kompleksitas perhitungan *heuristic* belum memberikan dampak yang signifikan terhadap waktu komputasi.

Namun, pada *Level* 2 dan terutama *Level* 3, perbedaan performa menjadi semakin jelas. *Manhattan Distance* secara konsisten menghasilkan waktu pencarian yang lebih rendah dibandingkan *Euclidean Distance* pada kedua mode pergerakan. Perbedaan paling mencolok terlihat pada *Level* 3 mode 8-arah, di mana *Manhattan Distance* membutuhkan 1.220,1 *ms*, sementara *Euclidean Distance* mencapai 6.990,5 *ms*. Kenaikan waktu pada *Euclidean Distancec* hampir enam kali lebih besar dibanding *Manhattan Distance* dalam konfigurasi yang sama.

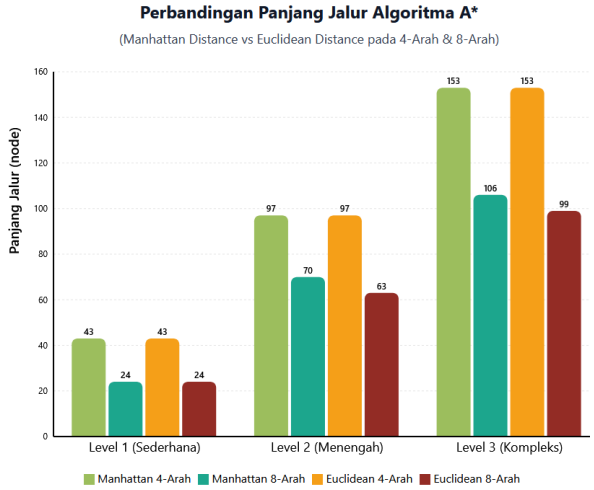
Fenomena ini dapat dijelaskan dari sisi kompleksitas komputasi *heuristic*. *Manhattan Distance* hanya melibatkan operasi aritmatika sederhana seperti penjumlahan dan nilai absolut, sedangkan *Euclidean Distance* memerlukan operasi akar kuadrat yang lebih berat secara komputasional. Pada *map* dengan jumlah *node* yang besar, perbedaan kompleksitas operasi ini terakumulasi dan menghasilkan selisih waktu yang signifikan.

Selain itu, mode 8-arah cenderung menghasilkan waktu yang lebih rendah dibandingkan 4-arah pada *Manhattan Distance* karena ruang solusi menjadi lebih fleksibel sehingga jumlah node yang dievaluasi lebih sedikit. Sebaliknya, pada *Euclidean Distance*, peningkatan jumlah tetangga dari 4 menjadi 8 dikombinasikan dengan operasi akar kuadrat menyebabkan lonjakan beban komputasi yang drastis pada *map* kompleks.

Dengan demikian, dari perspektif efisiensi waktu pencarian, *Manhattan Distance* terbukti lebih unggul terutama pada lingkungan dengan kompleksitas tinggi. Hasil ini menunjukkan adanya *trade-off* antara efisiensi komputasi dan kualitas estimasi jarak, yang akan dibahas lebih lanjut pada analisis optimalitas jalur.

3.6 Analisis Panjang Jalur dan Optimalitas

Analisis berikut mengevaluasi kualitas jalur yang dihasilkan algoritma A* berdasarkan panjang jalur (*node*) dan tingkat optimalitas (*%*). Perbandingan kedua parameter tersebut ditunjukkan pada Gambar 9 dan Gambar 10.



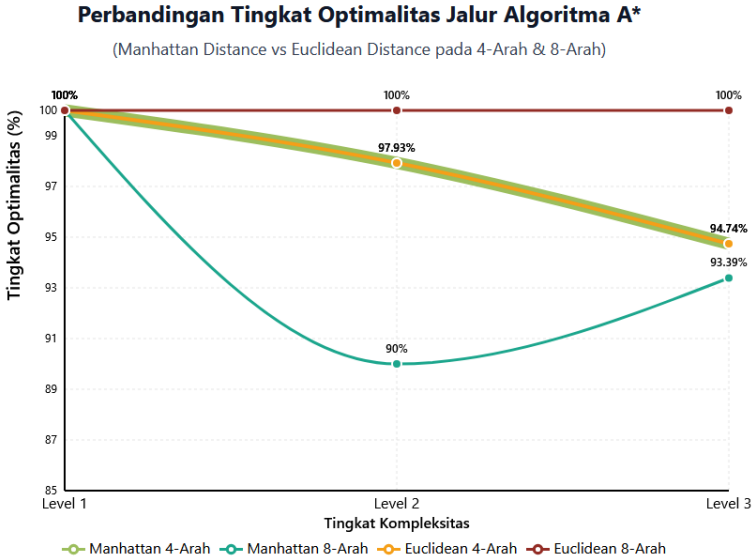
Gambar 9 Grafik perbandingan panjang jalur algoritma A*

Grafik pada Gambar 9 menunjukkan bahwa panjang jalur meningkat seiring bertambahnya kompleksitas *map*. Pada *Level* 1, seluruh konfigurasi menghasilkan jalur yang relatif pendek, dengan nilai 43 *node* pada mode 4-arah dan 24 *node* pada mode 8-arah. Hal ini menunjukkan bahwa pemberian opsi *diagonal* secara langsung memangkas jumlah langkah yang diperlukan untuk mencapai tujuan.

Pada *Level* 2 dan *Level* 3, perbedaan mulai terlihat. Mode 8-arah secara konsisten menghasilkan jalur lebih pendek dibanding 4-arah pada kedua *heuristic*. Hal ini disebabkan karena *diagonal movement* memungkinkan *NPC* mendekati tujuan dengan lintasan yang lebih langsung.

Euclidean Distance pada mode 8-arah menghasilkan jalur terpendek pada seluruh tingkat kompleksitas, dengan 63 *node* pada *Level* 2 dan 99 *node* pada *Level* 3. Sebaliknya, *Manhattan*

Distance menghasilkan jalur sedikit lebih panjang pada mode 8-arah karena estimasi jaraknya tidak sepenuhnya merepresentasikan jarak *diagonal* secara geometris.



Gambar 10 Grafik perbandingan optimalitas jalur algoritma A*

Grafik pada Gambar 10 menunjukkan bahwa *Euclidean Distance* pada mode 8-arah mempertahankan tingkat optimalitas 100% pada seluruh *level*. Hal ini menunjukkan bahwa *heuristic* tersebut bersifat *admissible* dan konsisten dalam konteks pergerakan *diagonal*, sehingga mampu menemukan jalur terpendek secara matematis.

Sebaliknya, *Manhattan Distance* mengalami penurunan optimalitas pada *Level* 2 dan *Level* 3, terutama pada mode 8-arah yang turun hingga 90% pada *Level* 2 dan 93,39% pada *Level* 3. Penurunan ini terjadi karena *Manhattan Distance* cenderung melebih-lebihkan estimasi jarak *diagonal*, sehingga jalur yang dipilih tidak selalu merupakan lintasan terpendek secara geometris.

Pada mode 4-arah, kedua *heuristic* menunjukkan tingkat optimalitas yang relatif sama. Hal ini disebabkan karena model pergerakan sesuai dengan asumsi *Manhattan Distance*, sehingga tidak terjadi ketidak seimbangan estimasi jarak.

4. KESIMPULAN

Penelitian ini menunjukkan bahwa algoritma *A-Star* (*A**) berhasil diimplementasikan secara efektif pada sistem navigasi *NPC* berbasis *Roblox Studio*, dengan perbedaan karakteristik performa antara *heuristic Manhattan Distance* dan *Euclidean Distance*. *Manhattan Distance* terbukti unggul dari sisi efisiensi waktu komputasi, terutama pada *map level* 3 dengan mode 8-arah, di mana waktu pencarian mencapai 1.220,1 *ms* dibandingkan 6.990,5 *ms* pada *Euclidean Distance*. Sebaliknya, *Euclidean Distance* unggul dalam kualitas jalur, menghasilkan lintasan lebih pendek dan tingkat optimalitas 100% pada pergerakan *diagonal* (99 *node* dibandingkan 106 *node* pada *Manhattan Distance*). Temuan ini menegaskan adanya *trade-off*, yaitu tidak bisa mendapatkan semuanya sekaligus secara maksimal antara efisiensi waktu dan kualitas jalur, sehingga pemilihan *heuristic* perlu disesuaikan dengan kebutuhan sistem, apakah berorientasi pada kecepatan atau akurasi lintasan.

5. SARAN

Untuk penelitian selanjutnya, disarankan melakukan pengujian pada lingkungan yang lebih dinamis dengan rintangan bergerak untuk mengevaluasi kemampuan adaptasi algoritma *pathfinding* secara *real-time*. Selain itu, penelitian selanjutnya juga dapat mengeksplorasi penggunaan *adaptive heuristic* atau *weighted heuristic* untuk meningkatkan efisiensi pencarian

jalur, serta melakukan perbandingan dengan algoritma *pathfinding* lainnya guna memperoleh pemahaman yang lebih komprehensif dalam pengembangan sistem navigasi NPC pada lingkungan *game* yang lebih kompleks.

DAFTAR PUSTAKA

[1] U. Wilhelmsson, W. Wang, R. Zhang, and M. Toftedahl, "Shift from game-as-a-product to game-as-a-service research trends," *Serv. Oriented Comput. Appl.*, vol. 16, no. 2, pp. 79–81, 2025, doi: 10.1007/s11761-022-00335-7.

[2] Z. R. Mair, "Strategi Pendekatan Konsep Pemrograman Dasar Komputer Dengan Game Dan Storytelling," vol. 3, no. 3, pp. 229–236, 2023, doi: 10.23960/jpkmt.v3i3.99.

[3] Z. R. Mair, *Coding Asyik For Kids*. Yogyakarta: Deepublish, 2024.

[4] N. A. S. Kanaka, R. Heriansyah, and S. Puspasari, "TIN : Terapan Informatika Nusantara Perbandingan Algoritma Decision Tree dan Support Vector Machine Dalam Pemilihan Calon Mahasiswa Penerima KIP-K TIN : Terapan Informatika Nusantara," vol. 4, no. 9, pp. 613–619, 2024, doi: 10.47065/tin.v4i9.4902.

[5] Z. R. Mair, M. H. Irfani, and D. Marcelina, "Kajian Bakat Anak Melalui Pola Tulisan Tangan dengan Algoritma Convolutional Neural Network .," vol. 10, no. 2, pp. 66–71, 2022.

[6] Z. R. Mair, W. Cholil, E. Yulianti, D. Marcelina, Theresiawati, and I. N. Isnainiyah, "Convolutional Neural Network Analysis on Handwriting Patterns and Its Relationship to Personality : A Systematical Review," pp. 308–312, 2023.

[7] E. G. Agung, D. Eridani, and A. Fauzi, "Implementasi Metode Pathfinding dengan Algoritma A * pada Game Rogue-like menggunakan Unity," vol. 7, no. December, pp. 81–94, 2022, doi: 10.34818/indojc.2022.7.3.677.

[8] A. C. Rizki, K. R. Athallah, S. M. Siddiq, and Z. Kusmanidar, "Performance Analysis of Dijkstra and A-star Algorithm in Maritime Navigation Pathfinding," vol. 2, no. 2, pp. 31–50, 2025.

[9] U. Affandi, S. Iskandar, N. M. Diah, M. Ismail, and A. Abdullah, "Comparing the efficiency of Pathfinding Algorithms for NPCs in platform games," *J. Posit. Sch. Psychol.*, vol. 6, no. 3, pp. 8434–8441, 2022.

[10] R. N. Sarbini, I. Ahmad, R. O. Bura, and L. Simbolon, "Comparative Analysis Of Pathfinding Artificial Intelligence Using Dijkstra And A * Algorithms Based On Rpg Maker Mv," 2020.

[11] R. P. Cahyani and R. Heriansyah, "Identification of Ginger Varieties Using Manhattan Distance on Image Pixel Vectors and Histograms," vol. 8, no. 2, 2025, doi: 10.32877/bt.v8i2.3019.

[12] F. S. Esen, H. Tinmaz, and M. Singh, *Metaverse: Technologies, Opportunities and Threats*, 1st ed. in Studies in Big Data. Beach Road: Springer Nature Singapore, 2023. [Online]. Available: https://books.google.co.id/books?id=iMfcEAAAQBAJ

[13] Y. Zhang, "ASL-DWA : An Improved A-Star Algorithm for Indoor Cleaning Robots," *IEEE Access*, vol. 10, no. August, pp. 99498–99515, 2022, doi: 10.1109/ACCESS.2022.3206356.

[14] R. Mussabayev, "Optimizing Euclidean Distance Computation," pp. 1–36, 2024.

[15] M. Kiepe, *Mastering Roblox Coding: The unofficial guide to leveling up your Roblox scripting skills and building games using Luau programming*, 1st ed. Birmingham: Packt Publishing, 2022. [Online]. Available: https://books.google.co.id/books?id=YG1_EAAAQBAJ