Analisis Perbandingan UCS, A*, dan RRT* pada Simulasi Pencarian Jalur Robot Sepak Bola Humanoid di Lapangan Berhalangan

Muchammad Yuda Tri Ananda - 24060124110142
Departemen Informatika
Universitas Diponegoro
Semarang, Indonesia
myudak@students.undip.ac.id

Abstract—Penelitian ini membandingkan Brute Force berbasis waypoint, kelompok Uniform Cost Search (UCS), Greedy Best First Search (GBFS), A*, dan Rapidly-exploring Random Tree Star (RRT*) pada pencarian jalur robot sepak bola humanoid. Eksperimen utama menggunakan empat skenario, sepuluh seed peta yang dipastikan feasible, dan tiga pengulangan waktu per run. Analisis tambahan menguji enam resolusi grid serta enam tingkat kepadatan hambatan. Hasil menunjukkan bahwa A* selalu mempertahankan biaya UCS pada grid dengan ekspansi lebih terarah, GBFS paling cepat tetapi menghasilkan detour lebih besar, Brute Force kehilangan reliabilitas ketika hambatan bertambah, dan RRT* menyediakan lintasan kontinu dengan biaya sampling yang lebih variatif. Regresi empiris menunjukkan runtime UCS dan A* tumbuh mendekati linear terhadap jumlah sel pada rentang pengujian.

Keywords—Brute Force, UCS, Greedy Best First Search, A*, RRT*, pencarian jalur, robot humanoid, analisis algoritma

I. PENDAHULUAN

Robot sepak bola humanoid membutuhkan kemampuan memilih lintasan secara cepat karena lingkungan pertandingan dipenuhi objek yang dapat menghalangi gerak. Dalam tahap perencanaan gerak, robot perlu menentukan rute dari posisi awal menuju target sambil menghindari robot lain, batas area, atau zona yang tidak boleh dilalui.

Pencarian jalur relevan untuk Analisis dan Strategi Algoritma karena satu masalah dapat diselesaikan dengan pendekatan yang sangat berbeda. Pada makalah ini, algoritma yang dibandingkan disusun menjadi tiga pilihan utama: Brute Force, kelompok UCS/GBFS/A*, dan RRT*. Pengelompokan UCS, GBFS, dan A* mengikuti butir tugas yang menempatkan ketiganya sebagai satu strategi pencarian berbasis frontier.

Penelitian ini memiliki tiga tujuan: (1) membandingkan kualitas, reliabilitas, waktu, dan beban processed kelima metode pada peta yang sama; (2) mengukur skalabilitas empiris UCS, GBFS, dan A* ketika resolusi grid meningkat; serta (3) menilai perubahan robustness seluruh metode ketika kepadatan hambatan bertambah.

II. BATASAN MASALAH

Penelitian dibatasi pada simulasi dua dimensi. Lapangan direpresentasikan sebagai bidang berukuran 60 x 40 satuan, sedangkan posisi awal dan posisi target dianggap sudah diketahui. Hambatan dimodelkan sebagai lingkaran dan persegi panjang yang tidak boleh dilewati lintasan.

Brute Force menggunakan enumerasi kandidat waypoint dan pemeriksaan garis bebas hambatan. UCS, GBFS, dan A* menggunakan grid delapan arah, yaitu atas, bawah, kiri, kanan, dan empat arah diagonal. Biaya gerak lurus bernilai 1, sedangkan biaya gerak diagonal bernilai akar dua. Pada RRT*, posisi robot direpresentasikan sebagai titik kontinu.

Aspek dinamika robot seperti keseimbangan, percepatan servo, gaya gesek kaki, sensor kamera, dan ketidakpastian lokalisasi tidak dimodelkan secara detail. Oleh karena itu, eksperimen ini lebih tepat disebut sebagai simulasi perencanaan jalur, bukan simulasi robot humanoid lengkap.

Secara formal, masukan masalah adalah ukuran lapangan, posisi awal robot, posisi tujuan, dan himpunan hambatan. Hambatan direpresentasikan sebagai lingkaran atau persegi panjang, sedangkan posisi start dan goal berada pada ruang bebas. Pada model grid, ruang dibagi menjadi sel diskret; pada model kontinu, posisi dianggap sebagai koordinat real yang dapat dihubungkan oleh segmen garis.

Keluaran yang diharapkan adalah lintasan valid dari start ke goal. Lintasan valid tidak boleh berpotongan dengan hambatan dan harus tetap berada di dalam batas lapangan. Fungsi objektif yang digunakan adalah meminimalkan biaya lintasan, yaitu jumlah jarak antar titik atau simpul berurutan. Dengan formulasi ini, perbedaan algoritma dapat dibandingkan melalui keberhasilan menemukan jalur, biaya rute, waktu komputasi, dan jumlah processed.

III. Dasar Teori

A. Pemodelan Pencarian Jalur

Masalah pencarian jalur dapat dimodelkan sebagai pencarian lintasan dari titik awal ke titik tujuan pada ruang bebas hambatan. Dalam representasi grid, setiap sel bebas menjadi simpul dan perpindahan antar sel menjadi sisi. Dalam representasi waypoint atau ruang kontinu, simpul dapat berupa titik kandidat yang dihubungkan jika segmen garis antar titik tidak menabrak hambatan.

$$C(P) = \Sigma\, d(v_i, v_{i+1}),\ 0 \le i < k \qquad (1)$$

Persamaan (1) menunjukkan bahwa biaya lintasan merupakan penjumlahan jarak antar simpul atau titik berurutan. Untuk robot humanoid, biaya ini dapat dimaknai sebagai pendekatan terhadap jarak tempuh yang harus dilalui sebelum modul kontrol gerak menjalankan langkah fisik.

B. Brute Force Berbasis Waypoint

Brute Force mencoba semua kombinasi kandidat solusi dalam batas tertentu. Pada eksperimen ini, kandidat waypoint diambil dari koridor start-goal, titik sekitar hambatan, dan area celah dinding. Program kemudian mengevaluasi setiap urutan hingga tiga waypoint dan memilih rute bebas hambatan dengan biaya terkecil. Pendekatan ini mudah dipahami, tetapi tidak lengkap jika solusi membutuhkan waypoint di luar kandidat atau lebih banyak titik antara.

C. Kelompok UCS, Greedy Best First Search, dan A*

UCS, GBFS, dan A* sama-sama menggunakan frontier berbasis priority queue, tetapi berbeda pada fungsi prioritasnya. UCS memilih simpul dengan biaya aktual terkecil dari start. GBFS memilih simpul yang secara heuristik paling dekat ke tujuan. A* menggabungkan biaya aktual dan estimasi sisa biaya sehingga dapat lebih terarah tanpa kehilangan optimalitas ketika heuristiknya admissible [1], [2].

$$p\_UCS(n) = g(n),\ p\_GBFS(n) = h(n),\ p\_A*(n) = g(n) + h(n) \qquad (2)$$

Heuristik yang digunakan adalah jarak Euclidean karena gerakan grid mengizinkan arah diagonal. UCS digunakan sebagai baseline optimal pada graf berbobot non-negatif, GBFS digunakan untuk melihat dampak keputusan yang sangat agresif mengikuti arah goal, dan A* digunakan sebagai kompromi antara optimalitas dan efisiensi.

D. Rapidly-exploring Random Tree Star

RRT* adalah pengembangan dari Rapidly-exploring Random Tree yang menumbuhkan pohon pencarian melalui sampling acak pada ruang konfigurasi. Setiap sampel dihubungkan ke simpul terdekat jika lintasan bebas hambatan, kemudian dilakukan rewiring agar struktur pohon mendekati lintasan berbiaya lebih kecil [3], [4].

E. Metrik Evaluasi

Lima metrik digunakan. Tingkat keberhasilan menyatakan persentase run yang menemukan jalur; biaya lintasan menyatakan panjang rute; detour ratio membagi biaya dengan jarak lurus start-goal agar kualitas dapat dibandingkan lintas representasi; waktu eksekusi dinyatakan dalam milidetik; dan processed mencatat kandidat waypoint, simpul grid, atau node sampling. Processed hanya dibandingkan langsung pada UCS, GBFS, dan A* karena ketiganya memakai satuan yang sama.

F. Kompleksitas dan Karakter Implementasi

Brute Force memiliki kompleksitas yang dipengaruhi jumlah kandidat m dan batas kedalaman d, yaitu mendekati jumlah permutasi ΣP(m,d). Pada eksperimen ini d dibatasi menjadi tiga agar masih dapat diuji. UCS, GBFS, dan A* pada priority queue memiliki kompleksitas teoretis mendekati O((V + E) log V), dengan V sebagai jumlah sel bebas dan E sebagai jumlah hubungan antar sel.

RRT* memiliki karakter berbeda karena biaya komputasi dipengaruhi jumlah sampel dan proses pencarian tetangga untuk rewiring. Implementasi sederhana yang digunakan masih melakukan pencarian tetangga secara linear, sehingga biaya dapat mendekati $O(n^2)$ terhadap jumlah node sampling.

TABLE I. Ringkasan Kompleksitas Lima Algoritma

Algoritma	Prioritas	Kompleksitas	Catatan
Bruteforce	waypoint	$O(\Sigma P(m,d) \cdot L)$	Batas d=3
UCS	g(n)	$O\big((V+E)\log V\big)$	Optimal
GBFS	h(n)	$O\big((V+E)\log V\big)$	Tidak optimal
A*	g(n) + h(n)	$O\big((V+E)\log V\big)$	Optimal*
RRT*	Sampling	$O(n)^2$	Kontinu

IV. Metodologi

A. Desain Simulasi

Eksperimen dibuat menggunakan program Python. Peta simulasi berukuran 60 x 40 satuan dengan titik awal di sisi kiri lapangan dan titik tujuan di sisi kanan lapangan. Setiap peta berisi hambatan berbentuk lingkaran dan persegi panjang. Empat tingkat skenario digunakan agar algoritma diuji pada kondisi mudah hingga padat.

B. Lingkungan Eksperimen

Eksperimen dijalankan pada satu perangkat yang sama agar perbandingan waktu lebih konsisten. Program dijalankan melalui uv dengan CPython 3.12.13 pada Windows 11 dan memori sekitar 16 GB. Pengukuran waktu menggunakan time.perf_counter sehingga nilai yang dicatat adalah durasi eksekusi fungsi algoritma dalam milidetik.

Eksperimen utama memakai grid 60 x 40 dan sepuluh independent map seed per skenario. Generator melakukan resampling deterministik sampai A* memastikan start dan goal terhubung. Setiap runtime diukur tiga kali; median per peta kemudian diringkas dengan rata-rata dan simpangan baku antarseed. Data mentah dan ringkasan disimpan sebagai CSV.

TABLE II. SKENARIO PENGUJIAN

Algoritma	Hambatan	Tujuan uji
Mudah	5 lingkaran	Hambatan jarang
Sedang	10 lingkaran + 2 dinding	Ada celah lintasan
Sulit	2 dinding + 8 lingkaran	Perlu belokan jauh
Padat	19 lingkaran + 2 dinding	Ruang bebas sempit

C. Langkah Eksperimen

Setiap skenario diuji pada sepuluh peta feasible. Selain eksperimen utama, scaling grid memakai resolusi 30 x 20 sampai 120 x 80 dengan geometri ternormalisasi yang sama. Sweep kepadatan memakai 4, 8, 12, 16, 20, dan 24 hambatan nested per seed, sehingga setiap tingkat mempertahankan seluruh hambatan dari tingkat sebelumnya.

Pada Brute Force, program menyusun kandidat waypoint, memeriksa visibilitas antar titik, lalu mengevaluasi semua urutan kandidat sampai batas tiga waypoint. Pada UCS, GBFS, dan A*, frontier diimplementasikan menggunakan priority queue dengan fungsi prioritas masing-masing. Pada RRT*, parameter utama yang digunakan adalah jumlah iterasi maksimum, step size, radius rewiring, dan peluang sampling langsung ke goal.

D. Parameter Algoritma

TABLE III. PARAMETER ALGORITMA

Algoritma	Parameter
Bruteforce	Waypoints = 3; kandidat = 24 kandidat maks. 24, kedalaman waypoint maks. 3
UCS	$p(n) = g(n)$; grid 8 arah grid 60x40, 8 arah, biaya lurus 1 dan diagonal √2
GBFS	$p(n) = h(n)$; euclidean grid 60x40, h(n)=jarak euclidean ke goal
A*	$p(n) = g(n) + h(n)$; euclideangrid 60x40, f(n)=g(n)+h(n), h(n)=euclidean
RRT*	Iter = 650/950; step = 2,5; radius = 4,5max_iter 650 (padat 950), step 2,5, radius 4,5, goal bias 0,12

Parameter Brute Force dibatasi agar eksperimen tetap selesai dalam waktu wajar. Parameter RRT* dipilih melalui percobaan awal agar algoritma memiliki peluang menemukan jalur pada skenario sulit, tetapi tetap dapat selesai dalam waktu yang sebanding dengan metode lain. Pada skenario padat, jumlah iterasi maksimum RRT* dinaikkan karena ruang bebas lebih sempit.

E. Rancangan Pseudocode Ringkas

Secara umum, Brute Force membangun daftar kandidat titik, lalu mencoba rute start-waypoint-goal secara menyeluruh sampai batas kedalaman. UCS, GBFS, dan A* menggunakan kerangka program yang sama: memasukkan simpul awal ke priority queue, mengambil simpul dengan prioritas terkecil, memeriksa goal, lalu memasukkan tetangga valid dengan prioritas sesuai algoritma.

Untuk RRT*, program mengulang proses sampling titik acak, mencari simpul terdekat pada pohon, membuat titik baru sejauh step size, memeriksa tabrakan, memilih parent terbaik dari tetangga dalam radius tertentu, lalu melakukan rewiring jika jalur melalui titik baru menghasilkan biaya lebih rendah.

V. HASIL DAN PEMBAHASAN

A. Hasil Kuantitatif

Tabel IV merangkum empat skenario dari sepuluh seed. Biaya dihitung hanya dari run yang berhasil, sehingga harus dibaca bersama tingkat keberhasilan. Waktu merupakan median dari tiga eksekusi per peta sebelum dirata-ratakan antarseed.

TABLE IV. RINGKASAN BIAYA DAN WAKTU EKSPERIMEN

Sken.	Algoritma	Ok	Biaya	Ms
Mudah	Brute Force	100%	53.36	137.1
Mudah	UCS	100%	54.74	300.3
Mudah	GBFS	100%	55.30	8.3
Mudah	A*	100%	54.74	38.0
Mudah	RRT*	100%	57.35	302.5
Sedang	Brute Force	80%	55.70	168.1
Sedang	UCS	100%	55.40	540.1
Sedang	GBFS	100%	56.38	18.7
Sedang	A*	100%	55.40	89.2
Sedang	RRT*	100%	57.35	469.9
Sulit	Brute Force	20%	65.37	108.9
Sulit	UCS	100%	70.35	433.9
Sulit	GBFS	100%	75.88	29.1
Sulit	A*	100%	70.35	266.0
Sulit	RRT*	90%	82.27	232.0
Padat	Brute Force	30%	70.43	91.8
Padat	UCS	100%	57.06	674.0
Padat	GBFS	100%	60.44	20.1
Padat	A*	100%	57.06	146.4
Padat	RRT*	100%	59.17	974.2

Selain biaya dan waktu, jumlah processed penting karena menunjukkan seberapa banyak kandidat solusi, simpul grid, atau node sampling yang harus diperiksa oleh algoritma. Karena satuan processed berbeda antarrepresentasi, angka pada Tabel V dibaca sebagai indikator beban kerja internal setiap algoritma, bukan sebagai satuan fisik yang sepenuhnya identik.

TABLE V. RINGKASAN JUMLAH PROCESSED

Sken.	Algoritma	Processed
Mudah	Brute Force	12721
Mudah	UCS	2036.2
Mudah	GBFS	54.5

Sken.	Algoritma	Processed
Mudah	A*	238.1
Mudah	RRT*	615.1
Sedang	Brute Force	12721
Sedang	UCS	1817.2
Sedang	GBFS	78.8
Sedang	A*	307.1
Sedang	RRT*	559.2
Sulit	Brute Force	12721
Sulit	UCS	1796.8
Sulit	GBFS	130.8
Sulit	A*	1030.8
Sulit	RRT*	389.7
Padat	Brute Force	12721
Padat	UCS	1743.3
Padat	GBFS	66
Padat	A*	373.6
Padat	RRT*	768.9

UCS memproses simpul lebih banyak karena eksplorasinya melebar berdasarkan biaya aktual. A* menekan jumlah processed dengan memanfaatkan heuristik Euclidean, sedangkan GBFS paling sedikit memproses simpul tetapi tidak menjamin biaya minimum. Brute Force selalu mengevaluasi 12.721 kandidat rute pada konfigurasi batas yang digunakan, sementara processed RRT* bergantung pada jumlah node sampling yang berhasil ditambahkan.

Pada skenario mudah, UCS dan A* sama-sama berhasil 100% dengan biaya 54,74. A* memproses rata-rata 238,1 simpul dibanding 2036,2 pada UCS. GBFS paling cepat pada 8,3 ms, tetapi biayanya 55,30. Brute Force berhasil 100% dan RRT* 100%; detour ratio keduanya masing-masing 1,007 dan 1,082.

Pada skenario sedang, UCS dan A* sama-sama berhasil 100% dengan biaya 55,40. A* memproses rata-rata 307,1 simpul dibanding 1817,2 pada UCS. GBFS paling cepat pada 18,7 ms, tetapi biayanya 56,38. Brute Force berhasil 80% dan RRT* 100%; detour ratio keduanya masing-masing 1,051 dan 1,082.

Pada skenario sulit, UCS dan A* sama-sama berhasil 100% dengan biaya 70,35. A* memproses rata-rata 1030,8 simpul dibanding 1796,8 pada UCS. GBFS paling cepat pada 29,1 ms, tetapi biayanya 75,88. Brute Force berhasil 20% dan RRT* 90%; detour ratio keduanya masing-masing 1,233 dan 1,552.

Pada skenario padat, UCS dan A* sama-sama berhasil 100% dengan biaya 57,06. A* memproses rata-rata 373,6 simpul dibanding 1743,3 pada UCS. GBFS paling cepat pada 20,1 ms, tetapi biayanya 60,44. Brute Force berhasil 30% dan RRT* 100%; detour ratio keduanya masing-masing 1,329 dan 1,116.

Secara umum, hasil menunjukkan bahwa Brute Force cocok sebagai baseline konseptual tetapi tidak kuat untuk peta kompleks. GBFS unggul dari sisi waktu dan jumlah processed, tetapi kualitas jalurnya tidak selalu baik. A* menjadi metode grid paling seimbang karena mempertahankan optimalitas UCS dengan ekspansi lebih sedikit. RRT* relevan untuk ruang kontinu, tetapi performanya lebih sensitif terhadap sampling dan parameter.

Gambar 1 memisahkan hasil setiap metode pada skenario Sulit seed 4 agar bentuk lintasan tidak saling menutupi. UCS dan A* menghasilkan rute grid berbiaya sama, GBFS lebih agresif menuju goal, Brute Force menghubungkan waypoint kontinu, dan RRT* membentuk rute dari pohon sampling.

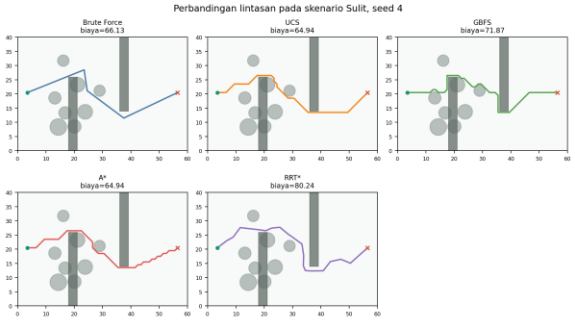

Fig. 1. Perbandingan lintasan Brute Force, UCS, GBFS, A*, dan RRT* pada skenario sulit.

Gambar 2 menunjukkan waktu median rata-rata dengan batang galat simpangan baku antarseed. GBFS menjadi metode grid tercepat pada seluruh skenario. Variasi yang tetap terlihat menegaskan bahwa runtime perlu dibaca sebagai distribusi antarpeta, bukan satu pengukuran tunggal.

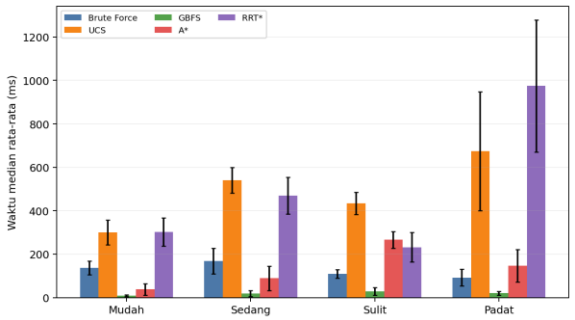

Fig. 2. Waktu eksekusi rata-rata dan simpangan baku pada setiap skenario.

Gambar 3 membandingkan processed UCS, GBFS, dan A*. A* mengurangi processed terhadap UCS sebesar Mudah 88,3%, Sedang 83,1%, Sulit 42,6%, Padat 78,6%. Pengurangan tersebut tetap menghasilkan biaya dan keberhasilan yang samadengan UCS.

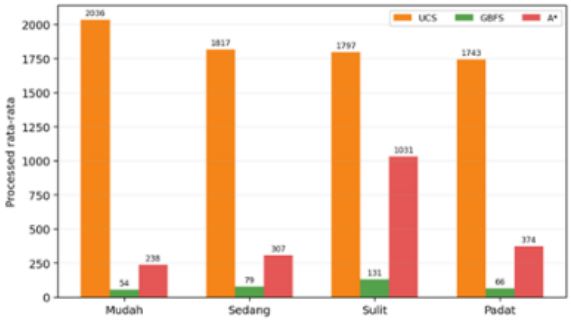

Fig. 3. Jumlah processed rata-rata UCS, GBFS, dan A* pada setiap skenario.

Gambar 4 mengevaluasi scaling enam resolusi. Eksponen runtime empiris terhadap jumlah sel adalah UCS 1,01 (R^2=1,000), GBFS 0,68 (R^2=0,996), dan A* 0,99 (R^2=0,999). Pada rentang ini, runtime dan processed UCS/A* meningkat hampir linear terhadap jumlah sel, sedangkan GBFS tumbuh lebih lambat karena frontier yang dieksplorasi tetap sempit.

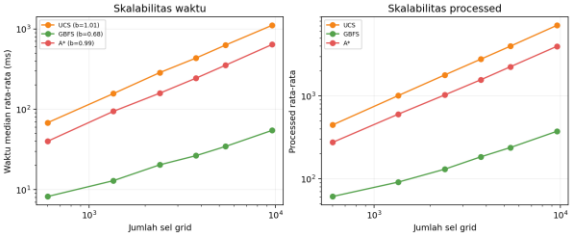

Fig. 4. Skalabilitas waktu dan processed UCS, GBFS, dan A* terhadap jumlah sel grid.

Gambar 5 menunjukkan dampak kepadatan pada peta yang seluruhnya feasible. Saat hambatan meningkat dari 4 menjadi 24, keberhasilan Brute Force turun dari 100% menjadi 50%, sedangkan UCS, GBFS, A*, dan RRT* tetap 100%. Pada 24 hambatan, detour ratio A*/UCS adalah 1,069, GBFS 1,108, dan RRT* 1,134. Dengan demikian, kecepatan GBFS dibayar dengan rute yang lebih panjang, sedangkan keterbatasan kandidat waypoint menjadi sumber utama kegagalan Brute Force.

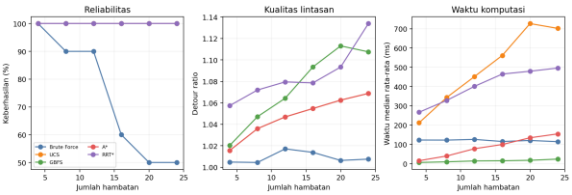

Fig. 5. Keberhasilan, detour ratio, dan waktu terhadap peningkatan jumlah hambatan.

B. *Analisis Perbandingan Algoritma*

Brute Force memiliki nilai pedagogis karena memperlihatkan cara paling langsung untuk mencoba kandidat solusi. Namun, pendekatan ini cepat kehilangan kelengkapan ketika jumlah kandidat dibatasi dan cepat kehilangan efisiensi ketika batasnya diperbesar. Dalam konteks robot sepak bola, Brute Force lebih cocok sebagai baseline kecil, bukan strategi utama.

Kelompok UCS, GBFS, dan A* memperlihatkan spektrum frontier yang konsisten pada eksperimen utama dan scaling. UCS paling menyeluruh, GBFS paling cepat tetapi menghasilkan detour lebih besar, dan A* mempertahankan biaya UCS dengan eksplorasi lebih kecil. Pada skenario Sulit, A* masih memproses 1030,8 simpul dibanding 1796,8 pada UCS karena dinding memaksa detour yang kurang terwakili oleh jarak Euclidean.

RRT* memiliki keunggulan konseptual karena tidak harus membagi ruang menjadi grid. Pendekatan ini relevan untuk perencanaan gerak kontinu dan dapat dikembangkan untuk mempertimbangkan batasan robotik lebih nyata, termasuk melalui varian informed sampling seperti Informed RRT* [7]. Kekurangannya adalah performa lebih sulit diprediksi karena bergantung pada jumlah sampel, goal bias, dan konfigurasi hambatan.

C. *Implikasi untuk Robot Sepak Bola Humanoid*

Jika sistem robot membutuhkan keputusan cepat pada peta grid lokal, A* merupakan pilihan paling kuat karena kompromi antara kualitas solusi dan efisiensi komputasi. GBFS dapat digunakan ketika waktu sangat sempit dan sedikit penurunan kualitas jalur masih dapat diterima. UCS berguna sebagai pembanding optimal, Brute Force sebagai baseline, dan RRT* untuk pengembangan menuju ruang gerak kontinu.

D. *Validasi dan Keterbatasan Eksperimen*

Validasi dilakukan dengan memeriksa bahwa jalur tidak memotong area hambatan. Pada Brute Force dan RRT*, setiap segmen garis diuji terhadap hambatan. Pada UCS, GBFS, dan A*, validasi dilakukan melalui pemeriksaan sel tetangga dan garis antar pusat sel agar gerak diagonal tidak melewati sudut hambatan.

Keterbatasan utama eksperimen adalah bentuk robot direduksi menjadi titik. Dalam robot humanoid nyata, tubuh robot memiliki lebar, postur, dan stabilitas berjalan yang perlu diperhitungkan. Selain itu, peta bersifat statis, sedangkan pertandingan robot sepak bola memiliki objek dinamis dan ketidakpastian sensor.

Sebagai bagian dari validasi, perbandingan ini tidak sepenuhnya identik karena setiap pendekatan memakai representasi ruang yang berbeda. UCS, GBFS, dan A* bekerja pada grid diskret; Brute Force bekerja pada kandidat waypoint dan segmen garis; sedangkan RRT* bekerja melalui sampling di ruang kontinu. Akibatnya, biaya lintasan A* dapat disebut optimal pada representasi grid, tetapi tidak otomatis menjadi optimum global pada ruang kontinu.

Validitas diperkuat dengan sepuluh seed, peta yang dipastikan feasible, dan tiga pengulangan timing. Namun, regresi scaling hanya menggambarkan rentang 600-9.600 sel dan tidak menggantikan analisis asimtotik. Implementasi Python, parameter waypoint/RRT*, serta bentuk robot titik tetap membatasi generalisasi. Seluruh algoritma dijalankan pada peta, perangkat, dan validator tabrakan yang sama.

VI. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan eksperimen utama, scaling, dan density sweep, A* menjadi metode grid paling seimbang. A* mempertahankan biaya dan keberhasilan UCS pada semua peta feasible, tetapi memproses lebih sedikit simpul. Pertumbuhan runtime empiris A* juga mendekati linear terhadap jumlah sel pada rentang uji.

GBFS konsisten paling cepat, tetapi detour meningkat ketika hambatan bertambah. Brute Force paling cepat kehilangan reliabilitas karena kandidat waypoint terbatas. RRT* tetap relevan untuk ruang kontinu, meskipun waktu dan kualitasnya dipengaruhi sampling.

Dengan demikian, untuk kasus lapangan berhalangan yang masih berbasis grid, A* lebih direkomendasikan sebagai strategi utama. Untuk penelitian robotika lanjutan yang mempertimbangkan ruang gerak kontinu dan batasan kinematika, RRT* tetap relevan sebagai dasar pengembangan.

B. Saran

Penelitian selanjutnya dapat menambahkan hambatan dinamis agar lebih mendekati kondisi pertandingan robot sepak bola. Eksperimen juga dapat diperluas dengan variasi algoritma seperti Theta* [8], D* Lite [9], atau Hybrid A* [10] yang lebih dekat dengan kebutuhan navigasi robot bergerak. Pada sisi Brute Force, kandidat waypoint dapat dibuat adaptif menggunakan visibility graph agar baseline lebih kuat. Pada sisi RRT*, tuning parameter dapat dilakukan secara lebih sistematis dengan menguji variasi step size, radius rewiring, goal bias, jumlah iterasi, dan informed sampling [7].

LAMPIRAN

Link kode program: https://github.com/myudak/Project-ASA-Analisis-Perbandingan-Path.git

Link video penjelasan: https://youtu.be/6qeD83UTYG8

Link website interaktif dan simulasi: https://myudak.github.io/Project-ASA-Analisis-Perbandingan-Path/

REFERENCES

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, vol. 1, pp. 269–271, 1959.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

[3] S. M. LaValle, Planning Algorithms. Cambridge, U.K.: Cambridge University Press, 2006.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," The International Journal of Robotics Research, vol. 30, no. 7, pp. 846–894, 2011.

[5] P. Corke, Robotics, Vision and Control: Fundamental Algorithms in MATLAB, 2nd ed. Cham, Switzerland: Springer, 2017.

[6] M. Bennewitz, S. Behnke, and W. Burgard, "Optimizing humanoid walking paths using body motion primitives," in Proc. IEEE International Conference on Robotics and Automation, 2005, pp. 1273–1278.

[7] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 2997–3004.

[8] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," Journal of Artificial Intelligence Research, vol. 39, pp. 533–579, 2010.

[9] S. Koenig and M. Likhachev, "D* Lite," in Proc. AAAI Conference on Artificial Intelligence, 2002, pp. 476–483.

[10] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," in Proc. AAAI Workshop on Search in Artificial Intelligence and Robotics, 2008.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya susun merupakan hasil karya saya sendiri dan tidak merupakan saduran, terjemahan, maupun plagiasi dari karya orang lain. Seluruh sumber yang digunakan telah dicantumkan secara benar sesuai dengan kaidah penulisan ilmiah.

Terkait penggunaan teknologi kecerdasan buatan (Artificial Intelligence), saya menyatakan bahwa (hapus salah satu):

☐ Saya **tidak menggunakan** alat bantu AI dalam proses penyusunan makalah ini.

☑ Saya **menggunakan** alat bantu AI sebagai pendukung dalam penyusunan makalah ini dengan rincian sebagai berikut:

- Nama alat AI: OpenAI Codex.
- Tujuan penggunaan: membantu revisi struktur algoritma, perapian narasi, dan penyelarasan hasil eksperimen dengan kode.
- Ruang lingkup penggunaan: penyusunan teks makalah, bantuan pembaruan kode eksperimen, dan perapian tabel/visualisasi; seluruh isi, analisis, dan keputusan akhir tetap diverifikasi.

Penggunaan AI, apabila ada, hanya bersifat sebagai alat bantu dan tidak menggantikan peran saya sebagai penulis utama. Seluruh isi, analisis, serta kesimpulan dalam makalah ini tetap merupakan tanggung jawab saya sepenuhnya.

Semarang, 9 Mei 2026

Muchammad Yuda Tri Ananda
24060124110142