BENCHMARKING ALGORITMA PERENCANAAN GERAK BERBASIS SAMPLING PADA ROBOT UR5 DENGAN ROS MOVEIT DAN GAZEBO

Disusun Oleh:

N a m a : Octaviano Haposan Badar Ginanjar

NIM : 21523220

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2025

HALAMAN PENGESAHAN DOSEN PEMBIMBING

BENCHMARKING ALGORITMA PERENCANAAN GERAK BERBASIS SAMPLING PADA ROBOT UR5 DENGAN ROS MOVEIT DAN GAZEBO

TUGAS AKHIR

Disusun Oleh:

N a m a : Octaviano Haposan Badar Ginanjar
NIM : 21523220

Yogyakarta, 5 November 2025
Pembimbing,

(Ir. Chandra Kusuma Dewa, S.Kom., M.Cs., Ph.D.)

HALAMAN PENGESAHAN DOSEN PENGUJI

BENCHMARKING ALGORITMA PERENCANAAN GERAK BERBASIS SAMPLING PADA ROBOT UR5 DENGAN ROS MOVEIT DAN GAZEBO

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 5 November 2025

Tim Penguji

Chandra Kusuma Dewa S.Kom., M.Cs, Ph.D. _____

Anggota 1

Irving Vitra Paputungan, S.T., M.Sc. 01/12/2025

Anggota 2

Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng.,PhD.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Octaviano Haposan Badar Ginanjar
NIM	:	21523220

Tugas akhir dengan judul:

BENCHMARKING ALGORITMA PERENCANAAN GERAK BERBASIS SAMPLING PADA ROBOT UR5 DENGAN ROS MOVEIT DAN GAZEBO

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 22 Oktober 2025

Octaviano Haposan Badar Ginanjar

HALAMAN PERSEMBAHAN

Assalamualaikum Warahmatullahi Wabarakatuh.

Alhamdulillahi rabbil 'alamin, segala puji bagi Allah SWT yang telah melimpahkan rahmat, hidayah, dan kemudahan sehingga penulis dapat menyelesaikan tugas akhir ini dengan lancar. Shalawat serta salam senantiasa tercurah kepada Rasulullah Muhammad SAW yang telah membawa umat manusia dari kegelapan menuju cahaya ilmu pengetahuan. Tugas akhir ini penulis persembahkan kepada berbagai pihak yang telah berperan penting dalam perjalanan akademik penulis dan membentuk penulis menjadi pribadi yang lebih baik, kritis, dan bijaksana

Pertama, penulis persembahkan kepada kedua orang tua tercinta yang telah memberikan kasih sayang, doa, dukungan, dan pengorbanan yang tiada henti. Setiap tetes keringat dan air mata kalian adalah sumber kekuatan terbesar dalam hidup penulis. Terima kasih telah menjadi cahaya di setiap kegelapan dan pelabuhan di setiap badai kehidupan.

Kedua, penulis persembahkan kepada Intan, partner penulis yang selalu setia memberikan semangat.Terima kasih telah menjadi pendengar terbaik di setiap keluh kesah, dan menjadi penyemangat ketika penulis hampir menyerah. Kehadiranmu adalah berkah yang selalu penulis syukuri.

Ketiga, penulis persembahkan kepada saudari kandung yang telah menjadi penghibur di kala jenuh. Kalian adalah bagian tak terpisahkan dari perjalanan hidup penulis.

Keempat, penulis persembahkan kepada Program Studi Informatika Fakultas Teknologi Industri Universitas Islam Indonesia yang telah memberikan wadah untuk menuntut ilmu dan mengembangkan diri. Terima kasih atas segala fasilitas, bimbingan, dan kesempatan yang telah diberikan sehingga penulis dapat tumbuh menjadi pribadi yang lebih percaya diri untuk menghadapi tantangan masa depan.

Semoga tugas akhir ini dapat memberikan manfaat bagi banyak pihak dan menjadi amal jariyah bagi penulis. Segala kekurangan dalam tugas akhir ini sepenuhnya menjadi tanggung jawab penulis.

Wassalamualaikum Warahmatullahi Wabarakatuh.

HALAMAN MOTO

”Barangsiapa yang bersabar dan memaafkan, sesungguhnya (perbuatan) yang demikian itu termasuk perbuatan yang mulia”

- Asy-Syura:43

“Kematian adalah harapan bagi sebagian orang, kelegaan bagi banyak orang, dan akhir dari segalanya. Ia membebaskan sang budak, mengantarkan orang buangan pulang, dan menempatkan semua manusia pada derajat yang sama, sedemikian rupa sehingga hidup itu sendiri akan menjadi hukuman tanpa kematian.”

-Lucius Annaeus Seneca

KATA PENGANTAR

Alhamdulillahi rabbil ‘alamin. Segala puji bagi Allah SWT tuhan seluruh alam semesta, tuhan yang maha adil, tuhan yang maha mengetahui, yang senantiasa selalu melimpahkan rahmat, hidayah, karunia dan Ridha-Nya sehingga penulis dapat dengan lancar menuliskan tugas akhir ini sebagai penutup perjalanan akademiknya di Yogyakarta.Shalawat serta salam senantiasa dihaturkan kepada Nabi Muhammad SAW yang telah membawa umat islam dari jaman jahilliyah kepada jaman yang terang benderang.

Atas izin Allah SWT penulis dapat menyelesaikan penelitian dan tugas akhir dengan judul “*Benchmarking* Algoritma Perencanaan Gerak Berbasis Sampling pada Robot UR5 dengan ROS Moveit dan Gazebo” sebagai salah satu syarat kelulusan dalam jenjang S1 Informatika Universitas Islam Indonesia. Dalam proses ini penulis tidak lepas dari arahan serta dukungan dari berbagai pihak. Penulis mengucapkan terima kasih serta rasa hormat sebesar-besarnya kepada semua pihak yang telah membantu. Dengan segala hormat dan ketulusan, penulis ingin mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa Allah SWT, sumber segala petunjuk dan pengetahuan. Dari-Nya kita berasal, dan kepada-Nya pula kita akan kembali.
2. Kedua Orang tua yang penulis cintai dan hormati mama reni dan papa gigin, yang selalu memberikan dukungan, doa dan kasih sayang yang tak pernah padam, dengan segala pengorbanan yang mereka berikan menjadi sumber kekuatan bagi penulis untuk menyelesaikan perjalanan akademiknya.
3. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
4. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Jurusan Informatika Universitas Islam Indonesia.
5. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
6. Bapak Chandra Kusuma Dewa S.Kom., M.Cs., Ph.D., dosen pembimbing tugas akhir, tanpa arahan dan panduannya penulis tidak akan menyelesaikan tugas akhir ini.
7. Partner penulis, Intan, pemberi semangat, dukungan, simpati, dan apresiasi, rasa terima kasih penulis akan selalu mengiringi.
8. Saudara penulis, euis dan uli yang selalu memberikan semangat dan penghibur penulis
9. Keluarga besar alm. bapak olih solihin dan alm. bapak mulia sibarani, atas dukungan yang telah diberikan, terutama kepada sepupu penulis koko ryan yang telah menjadi wadah penampung pikiran selama dalam pengerjaan skripsi.

10. Grup Penuh Berkah Melimpah, Vahri, Fikri, Wisnu, Rafli, Agung, Fadhlan, Angga, Ridzqwan, Maulana, Luqman, Bima, Wildan, dan Ahsan sahabat perjuangan yang menemani penulis dari awal perjalanan kuliah sampai akhir.
11. Teman seperjuangan SMA kepada burhan dan michelle terimakasih banyak atas dukungan dan sudah menjadi tempat cerita serta mendukung dari awal pendidikan.

Penulis menyadari masih banyak kekurangan yang ada di dalam tugas akhir ini, oleh karena itu penulis mengapresiasi segala kritik dan saran yang dapat menyempurnakan penulisan ilmiah lain kedepannya. Penulis berharap tugas akhir ini dapat memberikan manfaat dan kontribusi yang baik, juga dapat memberikan wawasan dan inspirasi kepada pembaca yang mengkaji topik serupa di masa mendatang.

Yogyakarta, 15 Oktober 2025

Octaviano Haposan Badar Ginanjar

SARI

Teknologi robotika telah berkembang pesat dengan lebih dari 2,7 juta robot industri beroperasi hingga 2020. Robot manipulator *pick-and-place* menjadi komponen vital dalam efisiensi manufaktur. Keberhasilan implementasinya tidak hanya bergantung pada perangkat keras, tetapi juga algoritma perencanaan gerak (*motion planning*) yang merancang lintasan optimal dengan mempertimbangkan kendala lingkungan dan keterbatasan mekanis. Pemilihan algoritma yang tepat secara signifikan memengaruhi waktu penyelesaian tugas, konsumsi energi, ketepatan navigasi, dan umur operasional robot.

Penelitian ini bertujuan untuk melakukan evaluasi komprehensif terhadap 19 algoritma perencanaan gerak berbasis sampling yang tersedia dalam OMPL)\ pada robot manipulator UR5 6 derajat kebebasan dalam konteks tugas penyortiran balok warna. Algoritma yang dievaluasi mencakup SBL, EST, LBKPIECE, BKPIECE, KPIECE, RRT, RRTConnect, RRT*, TRRT, PRM, PRM*, FMT, BFMT, PDST, STRIDE, BiTRRT, LBTRRT, ProjEST, dan BiEST.

Metodologi penelitian mencakup perancangan model robot dan lingkungan simulasi dirancang untuk meniru skenario penyortiran industri dengan tiga meja sortir berwarna dalam jangkauan kerja robot. Data yang dikumpulkan mencakup delapan metrik performa: waktu operasi rata-rata, total waktu, waktu tercepat, waktu terlambat, *total Cartesian distance*, *total joint distance*, penggunaan memori minimum, dan maksimum. Setelah agregasi data, dilakukan analisis koefisien variasi untuk mengidentifikasi metrik dengan daya diskriminasi terbaik, dan normalisasi Min-Max untuk memungkinkan perbandingan objektif antar algoritma.

Hasil penelitian menunjukkan temuan signifikan dalam beberapa dimensi performa. PDST mencatat waktu tercepat, diikuti TRRT dan RRT. FMT menunjukkan konsistensi tertinggi sekaligus efisiensi lintasan terbaik, memvalidasi jaminan teoretisnya.

Temuan kritis mengungkap degradasi performa sistematis algoritma *bi-directional* pada lingkungan sederhana. BFMT menunjukkan penurunan drastis dibandingkan FMT, juga algoritma dengan jaminan *asymptotic optimality* (RRT*, PRM*) menunjukkan *trade-off* tidak menguntungkan.

Kata kunci: ROS, MoveIt, UR5, algoritma perencanaan gerak, *sampling-based planning*, Gazebo.

GLOSARIUM

Asymptotic Optimality	Jaminan teoretis bahwa algoritma akan konvergen menuju solusi optimal seiring dengan bertambahnya jumlah sampel atau iterasi.
Configuration Space (C-space)	Ruang yang merepresentasikan semua kemungkinan konfigurasi robot, di mana untuk robot dengan n derajat kebebasan.
Coefficient of Variation (CV)	Ukuran variabilitas relatif yang dihitung sebagai rasio standar deviasi terhadap nilai rata-rata, dinyatakan dalam persentase.
Degrees of Freedom (DOF)	Jumlah parameter independen yang menentukan konfigurasi robot; dalam penelitian ini menggunakan robot 6-DOF.
End-effector	Bagian akhir dari robot manipulator yang berinteraksi dengan lingkungan, dalam penelitian ini adalah *gripper* untuk mengambil balok.
Joint	Sambungan atau persendian yang menghubungkan dua *link* pada robot dan memungkinkan gerakan relatif di antaranya.
Joint Space	Ruang konfigurasi yang didefinisikan oleh sudut atau posisi dari semua *joint* robot.
Link	Segmen kaku pada struktur robot yang dihubungkan oleh joint.
Logical Camera	Sensor simulasi yang memberikan informasi posisi dan orientasi objek secara langsung tanpa pemrosesan citra.
Motion Planning	Proses komputasi untuk merancang lintasan gerak robot dari konfigurasi awal ke konfigurasi tujuan dengan mempertimbangkan kendala lingkungan.
Node	Unit eksekusi independen dalam ROS yang berkomunikasi dengan Unit eksekusi independen lain melalui topik, layanan, atau parameter.
Pick-and-Place	Tugas robotik yang melibatkan pengambilan objek dari satu lokasi dan menempatkannya di lokasi lain.
Sampling-based Planning	Pendekatan *motion planning* yang mengambil sampel konfigurasi secara acak untuk membangun representasi diskrit dari ruang bebas.
Trajectory	Lintasan pergerakan robot yang mencakup posisi, kecepatan, dan akselerasi sepanjang waktu.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi Robot adalah buah hasil dari cabang interdisipliner dari teknik dan ilmu yang mencakup teknik mesin, teknik elektronik & listrik, ilmu komputer, dan bidang lainnya. Robotika berkaitan dengan desain, konstruksi, pengoperasian, dan penggunaan robot, serta sistem komputer untuk pengendalian, umpan balik sensori, dan pemrosesan informasi (Ghadge et al., 2018) . Teknologi tersebut diterapkan dalam pengembangan mesin yang dirancang untuk membantu, mendukung, atau bahkan sepenuhnya menggantikan peran manusia dalam melaksanakan berbagai jenis tugas, mulai dari tugas sederhana hingga yang kompleks. Robot yang dikembangkan dapat memiliki berbagai bentuk dan desain, tergantung pada fungsi dan tujuan penggunaannya. Sebagian besar robot dirancang menyerupai bagian tubuh manusia, seperti lengan, kaki, atau tangan, sementara beberapa robot memiliki bentuk yang menyerupai tubuh manusia secara keseluruhan. Pendekatan ini dipilih karena desain antropomorfik mempermudah robot untuk beradaptasi dan berintegrasi dengan lingkungan kerja manusia, serta memungkinkan robot untuk menangani tugas-tugas yang sebelumnya hanya bisa dilakukan oleh manusia. Selain itu, bentuk robot yang menyerupai manusia juga membantu meningkatkan interaksi antara manusia dan robot, Dengan demikian, pengembangan robot tidak hanya bertujuan untuk meningkatkan efisiensi, tetapi juga untuk menciptakan teknologi yang dapat dengan mudah diterima dan digunakan (Ghadge et al., 2018).

Robot memainkan peran yang semakin penting dalam kehidupan modern dan telah menjadi komponen integral dalam berbagai sektor masyarakat (Garrett et al., 2025). Perkembangan teknologi yang pesat telah mendorong perluasan penerapan robot di berbagai bidang, mulai dari eksplorasi luar angkasa, layanan kesehatan, pertanian, militer, hingga lini produksi dalam industri manufaktur (Sandakalum & Ang, 2022). Salah satu contoh nyata dari perkembangan ini adalah implementasi robot dalam proses manufaktur industri. Hingga tahun 2020, tercatat lebih dari 2,7 juta robot industri telah digunakan di berbagai pabrik di seluruh dunia (International Federation of Robotics, 2022), menunjukkan pertumbuhan signifikan dalam adopsi teknologi robotika di sektor produksi. Angka ini mencerminkan meningkatnya kepercayaan industri terhadap kemampuan robot dalam meningkatkan produktivitas dan mengurangi biaya operasional.

Salah satu jenis robot yang sering digunakan dalam industri manufaktur adalah robot penyusun atau *pick-and-place* robot (Gomes et al., 2022) . Robot ini dirancang menyerupai lengan manusia lengkap dengan persendian untuk menangani tugas-tugas yang membutuhkan

kecepatan, presisi, dan konsistensi, seperti memindahkan, menyusun, atau mengatur komponen pada jalur produksi. Tidak hanya itu *pick-and-place* robot dapat menyelesaikan suatu tugas pada lingkungan dan kondisi yang keras dan berbahaya (S. Liu & Liu, 2021). Dengan kemampuannya untuk bekerja secara otomatis, robot penyusun dapat mengurangi kesalahan manusia dan meningkatkan efisiensi dalam proses manufaktur secara signifikan. Namun demikian, keberhasilan implementasi robot dalam lingkungan industri tidak hanya bergantung pada perangkat keras (*hardware*) seperti aktuator dan sensor, tetapi juga pada kecanggihan perangkat lunak (*software*), terutama dalam aspek perencanaan gerak atau *motion planning*.

Motion planning merupakan komponen fundamental dalam sistem kendali robot, yang berperan dalam merancang lintasan gerak dari titik awal menuju tujuan akhir dengan mempertimbangkan berbagai kendala yang mungkin dihadapi. Kendala-kendala tersebut dapat berupa kondisi lingkungan yang dinamis dan keterbatasan mekanis dari struktur robot. Sejak awal mula perkembangan robotika, sudah ada penelitian terhadap perencanaan gerak yang melibatkan pengembangan algoritma untuk menentukan jalur yang harus dijalankan robot untuk mencapai suatu tujuan (Garrett et al., 2025). Permasalahan perencanaan gerak pertama yang diangkat adalah memindahkan robot dari satu konfigurasi posisi ke konfigurasi posisi lain tanpa bertabrakan dengan objek di dunia nyata. Permasalahan tersebut diformulasikan oleh penelitian Lozano-Pérez dan Wesley pada tahun 1979 sebagai pencarian perencanaan gerak melalui ruang konfigurasi robot (ruang dengan dimensi yang mewakili sendi-sendi yang dapat dikontrol dari robot) dan telah menjadi fokus dari banyak pengembangan algoritma perencanaan gerak (Garrett et al., 2025). Oleh karena itu, penelitian ini akan memberikan kontribusi tambahan pada diskusi mengenai topik *motion planning* pada robot, dengan fokus untuk mengevaluasi dan mengidentifikasi algoritma perencanaan gerak yang paling efisien dalam membantu robot mencapai tujuannya secara optimal.

Dengan algoritma perencanaan gerak yang tepat, robot dapat menghindari rintangan, menyesuaikan pergerakan secara *real-time*, dan menyelesaikan tugasnya dengan efisien. Selebihnya, pemilihan algoritma *motion planning* yang sesuai akan sangat memengaruhi performa keseluruhan sistem robot, termasuk dalam hal waktu penyelesaian tugas, konsumsi energi (Alam et al., 2023) , ketepatan navigasi, dan bahkan umur operasional perangkat. Oleh karena itu, evaluasi dan penerapan algoritma *motion planning* yang handal dan sesuai dengan kebutuhan aplikasi merupakan faktor kunci dalam meningkatkan performa keseluruhan sistem robotika.

Dalam beberapa tahun terakhir, berbagai studi *benchmarking* telah dilakukan untuk mengevaluasi performa algoritma *motion planning* pada sistem robotika. Sebagian besar

penelitian tersebut berfokus pada skenario lingkungan yang kompleks dan penuh rintangan (*cluttered environments*), seperti penelitian oleh Zhang et al. (2025) yang meninjau algoritma berbasis sampling dalam ruang matematikal abstrak berdimensi tinggi, penelitian Orthey et al. (2025) yang menguji algoritma dalam 24 skenario berbeda termasuk eksperimen manipulasi dan jalur sempit, serta penelitian (Yang et al., 2023) yang melakukan studi komprehensif tentang performa *motion planner* dalam lingkungan berantakan (*cluttered environments*) dengan tiga tingkat kompleksitas yang berbeda. Penelitian-penelitian tersebut memberikan kontribusi penting dalam memahami perilaku algoritma pada kondisi lingkungan yang menantang, di mana robot harus menavigasi melalui ruang kerja yang penuh dengan rintangan statis maupun dinamis.

Meskipun penelitian tentang *motion planning* di lingkungan kompleks telah berkembang pesat, terdapat kesenjangan penelitian (*research gap*) yang signifikan pada evaluasi algoritma untuk tugas-tugas manipulasi sederhana di lingkungan yang relatif tidak berhalangan (*uncluttered environments*). Penelitian ini dirancang untuk menambah gambaran komprehensif mengenai karakteristik performa algoritma *motion planning* dengan mengevaluasi perilaku mereka pada kondisi lingkungan yang berbeda dari studi-studi sebelumnya. Untuk membangun pemahaman yang utuh tentang algoritma *motion planning*, diperlukan evaluasi sistematis di berbagai spektrum kompleksitas lingkungan, mulai dari lingkungan sederhana hingga kompleks. Karakteristik performa algoritma *motion planning* dapat berbeda secara signifikan ketika diaplikasikan pada lingkungan dengan tingkat kompleksitas yang berbeda. Algoritma yang dirancang untuk mengatasi kompleksitas tinggi mungkin menunjukkan perilaku yang berbeda ketika diterapkan pada tugas yang lebih *straightforward*, sementara algoritma yang lebih sederhana mungkin dapat mengungkapkan efisiensi optimal pada kondisi tertentu. Oleh karena itu, penelitian ini bertujuan memberikan kontribusi akademis berupa *benchmark* awal untuk lingkungan sederhana, sehingga melengkapi body of knowledge yang sudah ada dan memberikan gambaran yang lebih komprehensif tentang perilaku algoritma *motion planning* di berbagai kondisi operasional.

Penelitian ini dirancang untuk mengisi kesenjangan tersebut dengan melakukan evaluasi komprehensif terhadap 19 algoritma *motion planning* berbasis sampling pada tugas manipulasi sederhana, yaitu penyusunan balok warna menggunakan robot lengan UR5 dalam lingkungan simulasi tanpa dinamika rintangan kompleks. Berbeda dengan penelitian-penelitian sebelumnya yang menekankan kemampuan algoritma dalam mengatasi kompleksitas spasial tinggi, penelitian ini berfokus pada pengukuran efisiensi temporal, konsistensi performa, dan penggunaan sumber daya komputasional ketika algoritma diaplikasikan pada operasi *pick-and-*

place. Pendekatan ini bertujuan untuk memberikan *baseline benchmark* yang dapat menjadi referensi bagi pengembangan sistem robotika pada aplikasi-aplikasi industri yang tidak memerlukan kompleksitas navigasi tinggi, namun tetap menuntut reliabilitas dan efisiensi operasional. Dengan demikian, hasil penelitian ini diharapkan dapat memberikan wawasan komplementer terhadap studi-studi *benchmarking* yang telah ada, serta membantu praktisi dalam membuat keputusan yang lebih informed dalam pemilihan algoritma *motion planning* berdasarkan karakteristik lingkungan kerja dan kompleksitas tugas yang dihadapi.

Berdasarkan konteks yang telah dipaparkan mengenai pentingnya evaluasi algoritma *motion planning* untuk tugas manipulasi sederhana, penelitian ini bertujuan untuk mengidentifikasi algoritma robot penyusun balok warna yang paling efisien dan optimal dalam lingkungan kerja terstruktur. Penelitian ini akan merancang dan mengembangkan simulasi menggunakan platform *Robot Operating System* (ROS) untuk mengevaluasi kinerja 19 algoritma perencanaan gerak dalam menyusun balok berdasarkan warna. Evaluasi akan dilakukan dengan menggunakan metrik kuantitatif yang komprehensif, mencakup efisiensi waktu, konsistensi performa, efisiensi jalur, dan penggunaan sumber daya komputasional. Hasil penelitian ini diharapkan dapat memberikan rekomendasi praktis yang dapat diterapkan pada sistem robot penyusun di lingkungan manufaktur atau aplikasi serupa yang memiliki karakteristik tugas dengan kompleksitas rintangan rendah, sehingga implementasi teknologi robotika dapat dilakukan secara lebih efisien dan efektif.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini adalah:

a) Bagaimana perancangan simulasi robot dalam perangkat lunak simulasi Gazebo ?
b) Bagaimana karakteristik performa 19 algoritma *motion planning* berbasis sampling ketika diaplikasikan pada tugas manipulasi sederhana dalam lingkungan tidak berhalangan ?
c) Algoritma perencanaan gerak apa yang memberikan performa terbaik dalam hal kecepatan, konsistensi, dan efisiensi untuk tugas penyusunan balok berdasarkan warna?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

a) Penelitian dilakukan dalam lingkungan simulasi menggunakan Gazebo 11.11.0 dengan ROS Noetic pada sistem operasi Ubuntu 20.04 LTS, tanpa implementasi pada robot fisik.

b) Robot yang digunakan adalah model UR5 dengan 6 derajat kebebasan (6-DOF) yang dirancang oleh MTE-ROBOTIC-LAB.
c) Lingkungan kerja yang digunakan berupa simulasi tiga dimensi statis tanpa dinamika lingkungan (misal: suhu, angin dan gesekan) ataupun dinamika rintangan.
d) Tugas manipulasi terbatas pada penyortiran 6 balok berwarna (2 merah, 2 hijau, 2 biru) ke dalam 3 meja sortir berdasarkan warna.
e) Deteksi objek menggunakan sensor *Logical camera* yang memberikan informasi posisi dan orientasi secara langsung tanpa pemrosesan *computer vision*.
f) Penelitian membandingkan beberapa algoritma *motion planning* berbasis *sampling*
g) Evaluasi algoritma terbatas pada 19 algoritma berbasis sampling yang tersedia dalam *Open Motion Planning Library* (OMPL) yang terintegrasi dengan MoveIt..

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah:

a) Merancang dan mengembangkan simulasi robot penyusun balok warna yang dilengkapi dengan algoritma *motion planning*.
b) Melakukan evaluasi kuantitatif terhadap performa setiap algoritma berdasarkan beberapa metrik.
c) Mengidentifikasi algoritma perencanaan gerak yang paling optimal untuk tugas penyusunan objek dalam konteks aplikasi robotika industri.
d) Memberikan *baseline benchmark* untuk performa algoritma *motion planning* pada lingkungan tidak berhalangan (*uncluttered environments*) sebagai komplemen terhadap penelitian-penelitian yang fokus pada lingkungan kompleks.
e) Memberikan rekomendasi strategi implementasi algoritma *motion planning* yang dapat diterapkan pada sistem robot penyusun di lingkungan manufaktur atau aplikasi serupa.

1.5 Manfaat Penelitian

Manfaat penelitian ini adalah:

a) Memberikan kontribusi akademis berupa baseline benchmark untuk evaluasi algoritma motion planning pada lingkungan tidak berhalangan (uncluttered environments), melengkapi body of knowledge yang telah fokus pada lingkungan kompleks.
b) Menyediakan data komparatif komprehensif tentang karakteristik performa 19 algoritma motion planning berbasis sampling dalam konteks tugas manipulasi

sederhana, yang dapat menjadi referensi bagi peneliti dalam memahami perilaku algoritma di berbagai spektrum kompleksitas lingkungan.

c) Memberikan panduan pemilihan algoritma yang tepat bagi praktisi dan peneliti berdasarkan karakteristik lingkungan kerja dan kompleksitas tugas, dengan pertimbangan *trade-off* antara efisiensi temporal, konsistensi, dan penggunaan sumber daya komputasional..

d) Menyediakan landasan awal bagi pengembangan lebih lanjut, termasuk integrasi dengan sistem penglihatan komputer atau pembelajaran mesin untuk otomatisasi tingkat lanjut.

1.6 Metodologi Penelitian

Penelitian ini memiliki rancangan :

PRA-PELAKSANAAN

Tahapan ini mencakup proses identifikasi masalah dan analisis metodologi. Pada tahap identifikasi masalah, peneliti menyoroti kebutuhan akan penerapan robot di berbagai bidang, khususnya di sektor industri yang menuntut efisiensi tinggi dalam pelaksanaan tugas-tugas otomatisasi. Salah satu tantangan utama adalah menentukan algoritma perencanaan gerak (*motion planning*) yang paling tepat dan optimal untuk mendukung kinerja robot. Jawaban terhadap permasalahan tersebut diwujudkan melalui pengembangan dan evaluasi algoritma *motion planning* yang efektif untuk robot lengan 6-derajat kebebasan (6-DoF) dalam menyelesaikan tugas penyusunan balok berdasarkan warna. Oleh karena itu, penelitian ini memiliki potensi strategis dalam mengidentifikasi algoritma perencanaan gerak yang paling sesuai untuk kebutuhan presisi dan efisiensi tinggi.

PELAKSANAAN

Pada tahap ini, pelaksanaan penelitian dibagi menjadi dua bagian utama, yaitu tahap pengembangan dan tahap pengujian. Tahap pengembangan mencakup serangkaian proses awal seperti persiapan lingkungan *Robot Operating System* (ROS), perancangan model robot menggunakan *Unified Robot Description Format* (URDF), konfigurasi lingkungan simulasi menggunakan Gazebo, serta implementasi perencanaan lintasan robot. Setelah seluruh komponen pengembangan selesai dan sistem simulasi siap digunakan, penelitian dilanjutkan ke tahap pengujian. Pada tahap ini, dilakukan evaluasi terhadap performa algoritma perencanaan gerak dalam lingkungan simulasi yang telah disiapkan sebelumnya, dengan fokus pada pengujian efektivitas dan efisiensi robot dalam menjalankan tugas penyusunan balok berdasarkan warna.

PASCA PELAKSANAAN

Setelah algoritma selesai diuji, tahapan selanjutnya adalah melakukan evaluasi dan analisis terhadap data hasil pengujian. Proses ini bertujuan untuk menilai kinerja masing-masing algoritma berdasarkan parameter yang telah ditentukan, seperti efisiensi waktu, akurasi lintasan, dan tingkat keberhasilan tugas penyusunan. Hasil evaluasi dan analisis tersebut akan menjadi dasar dalam menarik kesimpulan, sekaligus menjawab rumusan masalah yang telah dirumuskan pada tahap awal penelitian.

1.7 Sistematika Penulisan

Struktur penulisan dan penyusunan laporan tugas akhir ini disusun dalam beberapa bab yang memberikan gambaran lengkap mengenai masalah yang diangkat serta solusi yang ditawarkan. Adapun sistematika penulisan ini terdiri dari 5 bab:

BAB I PENDAHULUAN

Bab ini mencakup pembahasan mengenai latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, serta sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini membahas tinjauan terhadap penelitian sebelumnya yang relevan dengan rancangan dan implementasi yang dilakukan, serta teori dasar yang mendukung sistem implementasi algoritma perencanaan gerak pada robot penyusun.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan tahapan dan metodologi yang digunakan dalam pengembangan sistem robot penyusun balok warna berbasis ROS. Penelitian mencakup perancangan model robot, konfigurasi kontrol dan perencanaan gerak, serta integrasi sensor *logical camera*. Lingkungan simulasi dibangun dalam Gazebo dan diuji melalui arsitektur sistem yang modular. Seluruh sistem dirancang untuk memungkinkan evaluasi performa algoritma *motion planning* secara efisien dan terukur.

BAB IV HASIL DAN PEMBAHASAN

Bab ini memaparkan hasil dari algoritma dan pembelajaran robot penyusun, termasuk implementasi dan analisis hasil berdasarkan metodologi yang diterapkan dalam simulasi robot.

BAB V KESIMPULAN DAN SARAN

Bab terakhir ini berisi kesimpulan dari penelitian yang telah dilakukan serta saran untuk pengembangan lebih lanjut berdasarkan temuan dalam tugas akhir.

BAB II
KAJIAN PUSTAKA

2.1 Implementasi Robot Penyusun

Tujuan utama dari robot penyusun adalah untuk memberikan kemampuan pada robot agar dapat menyusun dan mengorganisasi objek berdasarkan karakteristik atau fitur yang dimiliki oleh objek tersebut. Selain itu, robot ini juga dirancang untuk mampu merencanakan pergerakan atau trayektori secara efisien dalam menyelesaikan tugas penyusunan. Proses ini didukung oleh masukan data dari sensor, seperti kamera, yang memberikan informasi visual mengenai objek yang akan diatur. Kebutuhan akan robot penyusun dapat ditemukan dalam berbagai proses industri seperti industri agrikultur dan pemenuhan pemesanan *E-commerce* (McKenzie et al., 2017). Penerapan robot penyusun yang efisien dalam proses industri memungkinkan fleksibilitas yang lebih tinggi dalam proses manufaktur dengan biaya rendah. Oleh karena itu, pelaksanaan tugas secara efektif dan efisien menjadi suatu keharusan dalam implementasi teknologi robot ini.

Dengan berkembangnya kebutuhan akan robot penyusun industri dapat terlihat juga perkembangan pengembangan robot menggunakan *framework Robot Operating System* (ROS) dimana algoritma perencanaan jalur yang efektif merupakan faktor yang krusial (Wei & Ren, 2018).

2.2 Robot Operating System (ROS)

ROS telah menjadi *framework* sistem yang umum digunakan dalam penelitian aplikasi robotika. ROS menyediakan platform aplikasi sumber terbuka untuk mengembangkan berbagai jenis aplikasi robot, mulai dari robot industri hingga robot layanan. ROS pertama kali dikembangkan pada tahun 2007 oleh *Stanford Artificial Intelligence Laboratory* dan sejak itu telah berkembang menjadi salah satu platform yang paling banyak digunakan di bidang robotika, menawarkan berbagai perangkat lunak dan konvensi yang membantu pengembang merancang sistem robot yang kompleks.

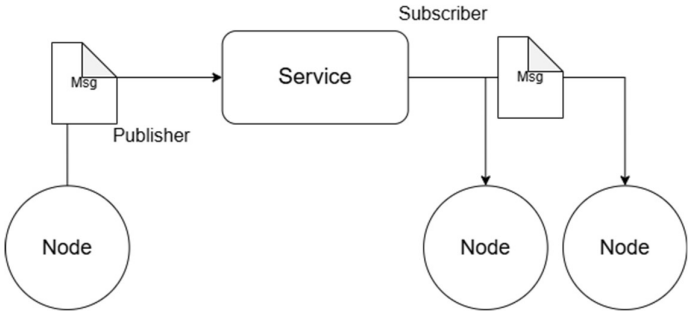
Pada intinya, ROS menyediakan kerangka kerja yang fleksibel dan modular yang memisahkan berbagai komponen sistem robot, memungkinkan mereka untuk bekerja secara independen namun berkomunikasi dengan lancar. ROS berfungsi sebagai lapisan middleware antara perangkat keras robot dan aplikasi perangkat lunak, menyediakan fungsionalitas utama seperti abstraksi perangkat keras, kontrol perangkat, pengiriman pesan, dan komunikasi antar sistem.

ROS menawarkan layanan-layanan yang khusus untuk robot. Layanan-layanan ini meliputi:

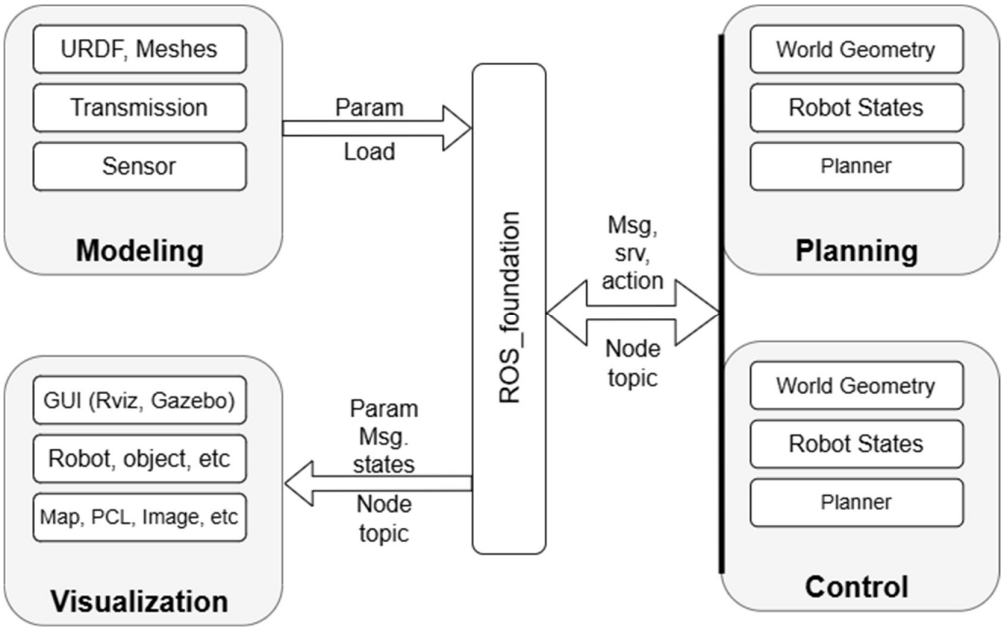
1. Abstraksi *hardware*: ROS menyediakan lapisan abstraksi yang menyederhanakan interaksi dengan berbagai komponen perangkat keras, seperti sensor, aktuator, dan kamera, memungkinkan pengembang untuk lebih fokus pada logika aplikasi daripada rincian spesifik perangkat keras.
2. Kontrol Perangkat Tingkat Rendah: ROS mencakup antarmuka untuk mengontrol perangkat keras pada tingkat rendah, memudahkan integrasi dengan berbagai platform *hardware* dan sensor.
3. Mekanisme Komunikasi: ROS menggunakan pengiriman pesan untuk komunikasi antar-proses, di mana berbagai *node* (atau komponen) dari sistem robot saling berkomunikasi dengan mengirim dan menerima pesan. Struktur ini memungkinkan integrasi yang mudah dari berbagai modul perangkat lunak dan sensor, bahkan jika mereka berjalan pada mesin yang terpisah.
4. Manajemen Paket: ROS menawarkan sistem manajemen paket yang kuat, yang memungkinkan pengembang untuk dengan mudah berbagi dan menggunakan kembali kode. Ekosistem ROS kaya dengan pustaka, alat, dan penggerak (*drivers*), sehingga mempermudah integrasi solusi yang sudah ada ke dalam proyek robotika baru.
5. Alat Pengembangan: ROS juga menyediakan serangkaian alat pengembangan untuk membantu dalam simulasi, *debugging*, dan visualisasi. Misalnya, RViz digunakan untuk visualisasi data sensor dan status robot dalam 3D.

Semua fungsi utama ROS dibagi menjadi beberapa bagian kecil yang disebut *node*, yang saling berkomunikasi melalui pesan (*messages*). Pesan-pesan ini pada dasarnya adalah struktur data yang berisi bidang-bidang bertipe tertentu. Sebuah node mengirimkan pesan dengan menerbitkannya ke topik (*topic*) tertentu, di mana topik memiliki nama yang digunakan untuk mengidentifikasi isi pesan. Oleh karena itu, *node* lain yang membutuhkan data tertentu harus berlangganan (*subscribe*) ke topik yang relevan.

Untuk satu topik, mungkin ada beberapa penerbit (*publisher*) dan pelanggan (*subscriber*) yang berjalan secara bersamaan, serta satu *node* yang dapat menerbitkan atau berlangganan ke beberapa topik sekaligus. Selain itu, melalui layanan (*services*) yang terdiri dari dua struktur pesan, yaitu *request* dan *reply*, permintaan dan tanggapan dapat dilakukan. Klien menggunakan layanan dengan mengirimkan *request* dan menunggu *response*. Pada Gambar 2.1, ilustrasi komunikasi pesan dalam ROS ditampilkan.(Agha et al., 2021)

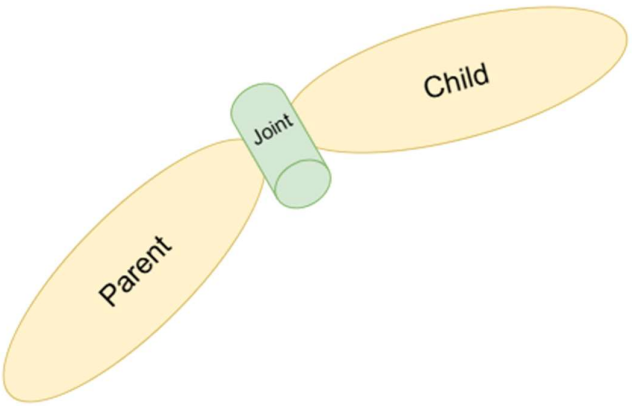


Gambar 2. 1 Visualisasi komunikasi dalam ROS



Gambar 2. 2 Arsitektur simulasi robotik dan hubungan module

Dalam ROS, untuk merepresentasikan robot dan modelnya, URDF diperlukan. URDF digunakan untuk mendeskripsikan semua elemen yang dibutuhkan, seperti sambungan (*joints*), sensor, tautan (*links*) dan manipulator seperti penjepit (*gripper*), agar seragam di ROS. File URDF pada dasarnya menggunakan format XML atau Xacro. URDF hanya dapat menunjukkan spesifikasi kinematik dan dinamik dari satu robot saja. Oleh karena itu, agar URDF dapat berfungsi dengan benar di simulator Gazebo, diperlukan penambahan *tag* simulasi khusus yang mencakup pose robot, inersia, *collision*, gesekan, dan properti lainnya Agha et al. (2021).



Gambar 2. 3 Visualisasi URDF hubungan antar 2 link dengan *joint revolute*

2.3 Gazebo

Gazebo adalah sebuah *software* simulasi 3D canggih yang dirancang untuk memodelkan robot, sensor, dan objek dalam berbagai lingkungan. Platform ini dikenal karena kemampuannya menghasilkan umpan balik sensor yang realistis serta interaksi fisik yang akurat, berkat integrasi simulasi *physic* benda kaku yang akurat. Dengan fitur-fitur ini, Gazebo memungkinkan pengguna untuk menciptakan lingkungan simulasi yang mendekati kondisi dunia nyata, sehingga menjadi alat yang ideal untuk pengujian dan validasi dalam penelitian robotika.

Salah satu keunggulan utama Gazebo adalah kemampuannya untuk menyimulasikan robot dan objek secara bersamaan dalam dunia tiga dimensi, menciptakan lingkungan yang kompleks dan dinamis. Platform ini juga memungkinkan pengembang menjalankan kode yang dirancang untuk robot fisik pada model virtual, memberikan kemudahan dalam menguji algoritma sebelum diterapkan pada robot sebenarnya.

Selain itu, arsitektur modular Gazebo mencakup *library* simulasi *physic*, antarmuka pengguna, komunikasi data, dan *rendering* grafis. Kombinasi ini memberikan fleksibilitas tinggi bagi para peneliti untuk mengembangkan eksperimen yang spesifik sesuai kebutuhan. Gazebo juga mendukung integrasi dengan *third party library*.

Sebagai alat yang digunakan secara luas dalam komunitas robotika, Gazebo telah membantu banyak peneliti dalam mengembangkan, menguji, dan mengoptimalkan berbagai teknologi robotika. Popularitasnya terus meningkat karena kemampuannya untuk memberikan simulasi yang realistis dan efisien, menjadikannya salah satu pilihan utama dalam penelitian berbasis simulasi.

2.4 Algoritma Perencanaan Gerak

Algoritma perencanaan gerak (*motion planning*) merupakan komponen fundamental dalam sistem manipulasi robot yang bertanggung jawab merancang lintasan optimal dari konfigurasi awal ke konfigurasi tujuan. Perencanaan ini harus mempertimbangkan berbagai kendala termasuk *obstacle avoidance*, batasan kinematik robot, dan kriteria optimasi seperti waktu minimum atau energi minimum.

Dalam konteks robot manipulator, *motion planning* bekerja dalam *configuration space* (*C-space*) yang merepresentasikan semua kemungkinan konfigurasi robot. Untuk robot dengan n derajat kebebasan, *C-space* memiliki dimensi n. Tujuan algoritma adalah menemukan jalur dalam *free C-space* yang menghubungkan konfigurasi awal dengan konfigurasi tujuan tanpa melanggar *constraint* apapun.. Penelitian telah menunjukkan bahwa ini merupakan salah satu cara paling efisien untuk menyelesaikan masalah dengan jumlah derajat kebebasan (DOF) yang besar (Orthey et al., 2025).

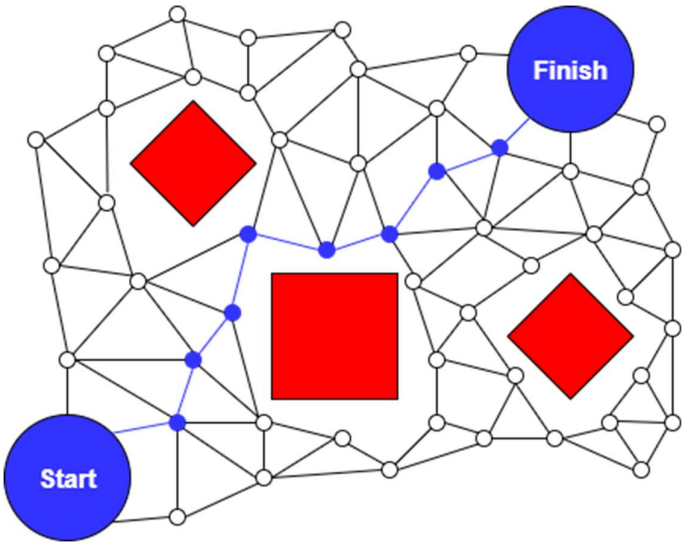
Pendekatan *sampling-based motion planning* telah terbukti sebagai salah satu metode paling efisien untuk menyelesaikan masalah dengan dimensi tinggi. Berbeda dengan pendekatan deterministik yang mencoba mengeksplorasi seluruh *C-space*, metode *sampling-based* mengambil sampel konfigurasi secara acak untuk membangun representasi diskrit dari *free space*. Pendekatan ini memiliki karakteristik *probabilistically complete*, artinya probabilitas menemukan solusi mendekati 1 seiring dengan bertambahnya jumlah sampel (Orthey et al., 2025). *Library* utama yang digunakan dalam penelitian ini adalah OMPL (*Open Motion Planning Library*), yang telah diintegrasikan dengan MoveIt!.

2.4.1 Jenis Algoritma OMPL (Open Motion Planning Library)

OMPL menyediakan berbagai algoritma, seperti:

a. PRM (*Probabilistic Roadmap*)

Diperkenalkan pada tahun 1996 oleh Lydia E. Kavraki, algoritma ini terdiri dari dua fase, yaitu fase *learning* dan fase *query*. Pada fase *learning*, sebuah *roadmap* (peta jalan) dibangun dengan cara mengambil sampel acak dari ruang konfigurasi, kemudian menyaring sampel yang bebas dari rintangan, serta menghubungkannya untuk membentuk sebuah graf. Selanjutnya, pada fase *query*, titik awal dan titik tujuan konfigurasi robot akan dihubungkan ke graf tersebut, lalu algoritma akan mencari jalur terpendek yang memungkinkan untuk mencapai solusi yang optimal. Namun ada kekurangan dari metode ini, yaitu metode ini tidak memiliki strategi optimasi jalur lokal, yang menyebabkan redundansi pembuatan jalur (Orthey et al., 2025).



Gambar 2. 4 Visualisasi algoritma PRM

b. PRM Star (PRM*)

PRM dengan strategi "*star*" yang secara otomatis menghitung jumlah tetangga yang akan dihubungkan berdasarkan cakupan ruang, menjamin optimalitas solusi. diperkenalkan untuk menyempurnakan PRM dasar dengan menjamin optimalitas asimtotik sambil mempertahankan kelengkapan probabilistik (Zhang et al., 2025).

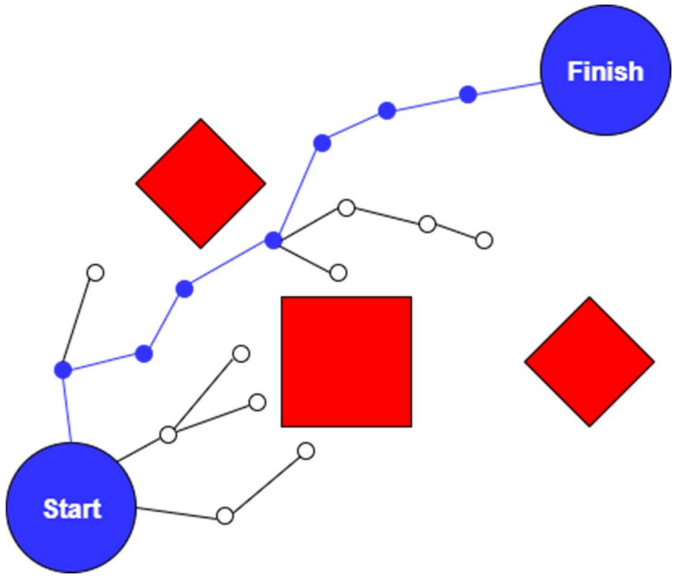
c. LazyPRM

Varian dari algoritma PRM yang dirancang untuk meminimalkan jumlah pemeriksaan tabrakan yang mahal, yang sangat membantu dalam lingkungan berdimensi tinggi atau berantakan. LazyPRM tidak memvalidasi semua *node* (titik) untuk mendeteksi tabrakan selama fase konstruksi peta jalan (seperti pada PRM standar), LazyPRM menunda pemeriksaan ini hingga jalur potensial ditemukan selama fase query (Orthey et al., 2025).

d. RRT (*Rapidly-exploring Random Tree*)

Diperkenalkan pada tahun 1998, RRT dikembangkan sebagai solusi untuk meningkatkan efisiensi dalam perencanaan gerak robot. Pada saat itu, algoritma yang populer seperti PRM masih menghadapi kesulitan dalam menangani skenario di mana robot berada di lingkungan yang dinamis atau berubah-ubah. Berbeda dengan PRM, RRT membangun jalur dengan memulai dari konfigurasi awal, kemudian secara iteratif mengambil sampel titik acak dan menghubungkannya ke *node* terdekat dalam pohon. Pemilihan titik acak ini memiliki bias terhadap area konfigurasi yang belum

dijelajahi, sehingga pohon akan terus berkembang ke arah ruang yang belum dieksplorasi, membentuk struktur bercabang seperti pohon (Zhang et al., 2025).



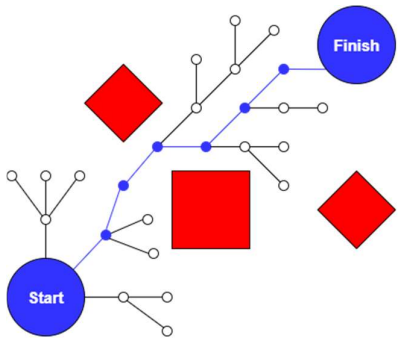
Gambar 2. 5 Visualisasi algoritma RRT

e. RRT-Connect

Algoritma RRT-Connect merupakan hasil perkembangan algoritma RRT yang mempercepat proses pencarian solusi dengan menghubungkan dua pohon. Algoritma ini memperkenalkan mekanisme ekspansi pohon ganda yang secara simultan mengembangkan pohon acak dengan titik awal dan titik tujuan sebagai *node* awal (Zhang et al., 2025).

f. RRT Star (RRT*)

Menambahkan strategi optimasi dua langkah yang mencakup '*rewire*' dan '*random relink*', di mana *rewire* berarti setelah sebuah node baru ditambahkan ke pohon, node induk untuk node tersebut dipilih ulang agar nilai biaya dari jalur yang dihasilkan menjadi lebih kecil, dan *random relink* berarti setelah proses *rewiring*, *node-node* di sekitar *node* baru akan disambungkan kembali ke *node* baru tersebut (Zhang et al., 2025).



Gambar 2. 6 Visualisasi algoritma RRT*

g. T-RRT (*Transition-based Rapidly-exploring Random Tree*)

Algoritma T-RRT adalah varian lain dari RRT yang mempertimbangkan "biaya" pada setiap posisi (*state*) untuk mencari jalur dengan biaya rendah, yaitu jalur yang mengikuti area yang aman atau mudah dilalui di dalam *costmap* ruang konfigurasi. T-RRT menggunakan *transition test* untuk memutuskan apakah suatu posisi baru layak ditambahkan ke dalam jalur atau tidak (Zhang et al., 2025).

h. BITRRT(*Bi-directional Transition-based Rapidly-exploring Random Trees*)

BITRRT adalah varian dari TRRT dimana pembuatan jalur tidak hanya terjadi di titik awal namun juga di titik tujuan secara bersamaan, saat kedua jalur bertemu maka jalur yang efisien telah ditemukan (Zhang et al., 2025).

i. LBTRRT (*Lower Bound Tree Rapidly-Exploring Random Tree*)

LBTRRT adalah varian dari RRT* yang menambahkan pohon batas bawah (*lower bound tree*) yang dibangun secara paralel dengan pohon utama. Pada RRT*, setiap iterasi berusaha menemukan jalur yang semakin mendekati optimal melalui proses *rewiring*. Pada LBTRRT, pohon batas bawah digunakan untuk menyimpan estimasi biaya minimum dari setiap *node* ke tujuan berdasarkan jarak bebas hambatan (*obstacle-free heuristic*). Informasi ini memungkinkan algoritma untuk memangkas pencarian di area yang sudah dapat dipastikan tidak akan menghasilkan jalur yang lebih baik dari solusi terbaik yang telah ditemukan, sehingga proses pencarian menjadi lebih efisien. Algoritma ini bekerja dengan membangun dua pohon, yaitu pohon utama seperti pada RRT* yang mencari jalur valid, dan pohon *lower bound* yang mencatat estimasi biaya minimum ke tujuan. Pada setiap langkah, algoritma melakukan sampling titik baru di ruang konfigurasi, menghitung batas bawah biaya dari titik tersebut ke tujuan, lalu membandingkannya dengan solusi terbaik saat ini. Jika hasil penjumlahan batas bawah biaya dan biaya dari titik awal ke titik tersebut lebih besar

daripada solusi terbaik yang sudah ditemukan, titik tersebut akan diabaikan (*pruning*). Jika tidak, titik tersebut dimasukkan ke pohon utama dan dilakukan proses *rewiring* seperti pada RRT*. Langkah ini diulang hingga waktu pencarian habis atau iterasi selesai (Tarbouriech & Suleiman, 2020).

j. EST (*Expansive-Spaces Tree*)

Algoritma EST menggunakan pendekatan *single-query tree-based planning* yang menumbuhkan pohon dari konfigurasi awal secara acak dengan menggunakan fungsi sampling hingga mencapai tujuan. EST dirancang khusus untuk ruang konfigurasi ekspansif, di mana setiap titik dapat "melihat" sebagian besar ruang bebas. Algoritma ini termasuk dalam kategori *tree-based planners* yang cocok untuk masalah *single-query* dan menjadi salah satu pelopor *sampling-based planning* pada era *sampling-advent* (1990-1999). Hsu, Latombe, dan Motwani membuktikan EST efektif untuk menjelajahi ruang yang luas dan kompleks, namun kinerjanya dapat menurun pada ruang konfigurasi berdimensi tinggi karena estimasi kepadatan menjadi mahal dan kurang akurat .

k. SBL (*Single-query Bi-directional Lazy collision checking*)

Algoritma SBL menggunakan pendekatan *bi-directional* dengan *lazy collision checking*, yang mengurangi waktu perencanaan dengan faktor besar dan memungkinkan penanganan masalah perencanaan yang lebih sulit, termasuk masalah multi-robot dalam lingkungan yang kompleks secara geometris. Penelitian Sanches dan Latombe membuktikan SBL menunjukkan performa terbaik dalam menemukan jalur terpendek dengan standar deviasi yang lebih tinggi (Naranjo-Campos et al., 2024).

l. ProjEST (*Projected Expansive-Spaces Tree*)

ProjEST adalah varian dari algoritma EST yang menjadi solusi untuk keterbatasan EST pada ruang konfigurasi dengan dimensi yang tinggi. Seperti EST, projEST membangun pohon dari konfigurasi awal, tetapi perbedaan utamanya terletak pada proses estimasi kepadatan. projEST memproyeksikan konfigurasi robot ke ruang berdimensi lebih rendah sebelum menghitung kepadatan, sehingga perhitungan menjadi lebih efisien. *Node* yang berada pada area dengan kepadatan rendah di ruang proyeksi akan diprioritaskan untuk dikembangkan. Dengan cara ini, projEST mempertahankan sifat eksploratif EST, namun lebih cepat dan lebih efisien untuk robot dengan banyak derajat kebebasan atau ruang dengan dimensi tinggi.

m. KPIECE (*Kinematic Planning by Interior-Exterior Cell Exploration*)

KPIECE adalah planner *kinodynamic* yang menggunakan diskretisasi berbasis grid *multi-level* untuk memperkirakan cakupan ruang keadaan. Estimasi cakupan membantu *planner* mendeteksi area yang kurang dieksplorasi dalam ruang keadaan (Silva et al., 2022).

n. BKPIECE (*Bi-directional* KPIECE)

BKPIECE adalah varian *bi-directional* dari KPIECE yang menunjukkan performa baik dengan tingkat keberhasilan tinggi di berbagai skenario, meskipun waktu perencanaan lebih lama dibandingkan algoritma lainnya (Silva et al., 2022).

o. LBKPIECE (*Lazy Bi-directional* KPIECE)

LBKPIECE adalah varian *bi-directional* dari KPIECE yang menggunakan satu level diskretisasi dengan *lazy collision checking*. Grid diimposisikan pada proyeksi ruang keadaan, dengan preferensi diberikan pada boundary dari grid yang telah dieksplorasi (Silva et al., 2022).

p. STRIDE (*Search Tree with Resolution Independent Density Estimation*)

STRIDE adalah algoritma perencanaan gerak berbasis pohon yang dirancang untuk ruang konfigurasi berdimensi tinggi, terutama yang memiliki banyak celah sempit. STRIDE bekerja dengan membangun pohon dari konfigurasi awal robot dan menggunakan estimasi kepadatan untuk menentukan area mana yang perlu dijelajahi. Tidak seperti algoritma lain yang menggunakan grid tetap atau proyeksi eksplisit, STRIDE memanfaatkan struktur data GNAT untuk memperkirakan kepadatan secara adaptif tanpa ketergantungan pada resolusi tertentu. Konfigurasi yang berada di area dengan kepadatan rendah akan diprioritaskan untuk dikembangkan, sehingga eksplorasi menjadi lebih efisien dan menyeluruh.

q. FMT (*Fast Marching Tree*)

FMT dirancang untuk mencari jalur yang optimal secara *asymptotically*, artinya semakin banyak sampel yang diambil, semakin mendekati jalur terbaik yang dapat ditemukan — seperti halnya pada algoritma PRM dan RRT. Algoritma ini bekerja dengan mengambil sejumlah sampel dari ruang konfigurasi yang bebas rintangan, namun berbeda dari PRM yang menyambungkan semua sampel atau RRT yang menumbuhkan cabang layaknya pohon, FMT membangun jalur berdasarkan perhitungan *cost* dari titik awal menuju titik-titik sampel yang memiliki *cost* terendah. Proses ini dilakukan secara iteratif hingga titik tujuan tercapai atau seluruh sampel

telah dieksplorasi yang dapat memperoleh solusi yang layak dengan waktu komputasi yang lebih singkat dan biaya jalur yang lebih rendah (J. Liu et al., 2025).

r. BFMT (*Bidirectional Asymptotically Optimal Fast Marching Tree*)

Algoritma BFMT adalah varian dari algoritma FMT dimana pembuatan jalur tidak hanya terjadi di titik awal namun juga di titik tujuan secara bersamaan, saat kedua jalur bertemu maka jalur yang efisien telah ditemukan.

2.5 *Forward Kinematics*, *Inverse Kinematics*, dan CHOMP dalam MoveIt

Forward Kinematics (FK) dan *Inverse Kinematics* (IK) merupakan konsep dasar yang digunakan secara luas dalam sistem perencanaan gerak robot manipulator. Dalam MoveIt, FK digunakan untuk menghitung posisi dan orientasi *end-effector* (gripper) berdasarkan nilai sudut tiap *joint*, sedangkan IK digunakan untuk menentukan konfigurasi joint yang sesuai agar end-effector dapat mencapai target posisi dan orientasi di ruang kartesian. Selain itu, MoveIt juga mendukung algoritma optimasi lintasan seperti CHOMP (Covariant Hamiltonian Optimization for Motion Planning) untuk memperhalus jalur yang dihasilkan oleh algoritma perencanaan berbasis sampling.

Forward Kinematics (FK) merupakan proses transformasi dari ruang joint menuju ruang kartesian. Masukan pada FK berupa nilai sudut atau translasi setiap joint, sedangkan keluarannya adalah pose end-effector dalam ruang Kartesian. Jika sebuah robot manipulator memiliki n buah joint dengan konfigurasi $\theta = [\theta_1,\ \theta_2,\ \cdots, \theta_n]^T$, maka pose end-effector dapat diperoleh dengan mengalikan matriks transformasi homogen tiap link menggunakan parameter Denavit–Hartenberg (DH):

$$T_0^n(\theta) = A_1(\theta_1) \cdot A_2(\theta_2) \cdot \cdots \cdot A_n(\theta_n) \tag{1}$$

Dengan

$$A_i(\theta_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos a_i & \sin\theta_i \sin a_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos a_i & -\cos\theta_i \sin a_i & a_i \sin\theta_i \\ 0 & \sin a_i & \cos a_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Hasil $T_0^n(\theta)$ memberikan posisi end-effector $p = [x, y, z]^T$. Dalam MoveIt, perhitungan FK dipanggil secara berulang untuk memvalidasi konfigurasi robot, melakukan pemeriksaan tabrakan, serta memverifikasi jalur yang dihasilkan algoritma perencanaan.

Inverse Kinematics (IK) adalah proses sebaliknya, yaitu menentukan konfigurasi joint yang sesuai agar end-effector mencapai pose target T_{target}. IK bisa memiliki banyak solusi, tidak ada solusi, atau memerlukan metode numerik. Salah satu metode numerik yang umum

digunakan, termasuk dalam KDL (*Kinematics and Dynamics Library*) yang menjadi *solver default* MoveIt, adalah pendekatan *Damped Least Squares* (DLS). Secara iteratif, update konfigurasi joint dilakukan dengan:

$$\Delta\theta = J^T(JJ^T + \lambda^2 I)^{-1}\Delta x \quad (3)$$

Dengan J adalah matriks Jacobian, Δx adalah error posisi dan orientasi end-effector, serta λ adalah faktor peredam (damping factor) untuk meningkatkan stabilitas pada kondisi mendekati singularitas. Iterasi dihentikan apabila error posisi dan orientasi lebih kecil dari toleransi yang ditentukan atau jumlah iterasi maksimum tercapai.

Setelah MoveIt memperoleh konfigurasi tujuan dari IK, proses perencanaan lintasan dilakukan di ruang joint oleh algoritma sampling-based planner dari OMPL, seperti RRT, PRM, KPIECE, dan varian lainnya. Pada tahap ini, FK kembali digunakan secara berulang untuk memproyeksikan konfigurasi joint ke ruang kartesian dalam rangka melakukan collision checking terhadap lingkungan.

Untuk menghasilkan lintasan yang lebih halus, MoveIt menyediakan modul optimasi CHOMP. Algoritma ini meminimalkan fungsi biaya yang terdiri dari dua komponen, yaitu smoothness cost untuk memastikan lintasan tetap halus dan bebas dari perubahan kecepatan mendadak, serta obstacle cost untuk menjaga jarak aman terhadap objek di lingkungan.

Dengan demikian, integrasi FK, IK, sampling-based motion planners, serta algoritma optimasi lintasan seperti CHOMP menjadikan MoveIt sebagai kerangka kerja yang kuat untuk penelitian ini, khususnya dalam melakukan benchmark terhadap 19 algoritma perencanaan gerak berbasis sampling pada robot UR5.

2.6 Penelitian Terdahulu

Semakin meningkatnya kebutuhan akan robot industri yang mampu beroperasi secara kolaboratif dengan algoritma yang efisien telah mendorong munculnya berbagai penelitian terkait. Hal ini mencerminkan tingginya minat untuk mengembangkan teknologi robotika yang dapat memenuhi tuntutan industri modern. Berikut adalah penelitian yang telah dilakukan dahulu yang relevan terhadap penelitian ini:

Tabel 2. 1 Penelitian Terdahulu

No	Judul	Penulis	Deskripsi	Algoritma yang diuji	Robot/lingkungan
1	Motion Planning for Robotics: A Review for	Liding Zhang, Kuanqi Cai, Zewei Sun, Zhenshan Bing, Chaoqun	Bertujuan untuk meninjau kembali berbagai algoritma *motion planning* berbasis *sampling* dengan studi	RRT, RRT*, RRT#, RRTConnect, LazyPRM*	Algoritma diuji di R^n *spaces* (lingkungan matematikal abstrak untuk menguji

No	Judul	Penulis	Deskripsi	Algoritma yang diuji	Robot/lingkungan
	Sampling-based Planners	Wang, Luis Figueredo, Sami Haddadin	literatur serta memberikan analisis mendalam mengenai desain, tantangan, pengembangan, implementasi, dan evaluasi dari algoritma-algoritma tersebut.	, BIT*, ABIT*, AIT*, EIRM*, EIT*	skalabilitas dan efisiensi algoritma dalam ruang yang memiliki dimensi yang tinggi) lalu algoritma diuji dalam tugas manipulasi robot *single-arm* dan *dual-arms*
2	Multi-criteria metric to evaluate motion planners for underwater intervention	Renato Silva, Aníbal Matos, Andry Maykol Pinto	Mengusulkan metrik pengujian algoritma *motion planning* yang bernama NEMU untuk mengevaluasi algoritma *motion planning* dalam robot manipulasi yang berada di lingkungan dalam air	BiEST, EST, ProjEST, TRRT, BiTRRT, RRT, SBL, BKPIECE, PDST, RRTConnect, STRIDE, RRT*, PRM*, PRM, FMT, FMT*, LBTRRT, LazyPRM, LazyPRM*, SPARS, SPARS2, KPIECE, SPARSEWO, LBKPIECE	Robot dengan 6 derajat kebebasan dalam skenario lingkungan dalam air
3	Sampling-Based Motion Planning: A Comparative Review	Andreas Orthey, Constantinos Chamzas, Lydia E. Kavraki	Menyajikan panduan komprehensif mengenai algoritma-algoritma *motion planning* berbasis sampling, yang mencakup sejarah perkembangan, prinsip dasar, serta pembahasan	RRT-Connect, PRM*, RRT*, KPIECE, FMT, EST, TRRT, BIT*	Algoritma-algoritma *motion planning* diuji dalam 24 skenario yang melibatkan berbagai jenis robot sesuai dengan kebutuhan masing-masing skenario. Skenario tersebut terdiri dari 6 eksperimen

No	Judul	Penulis	Deskripsi	Algoritma yang diuji	Robot/lingkungan
			mendetail mengenai karakteristik dan properti algoritma. Selain itu, penelitian ini juga melakukan evaluasi terhadap sejumlah algoritma dengan mengujikannya pada 24 permasalahan perencanaan		klasik *motion planning*, 6 eksperimen manipulasi, 6 eksperimen *constraint-extension*, serta 6 eksperimen jalur sempit.
4	Benchmarking of robot arm motion planning in cluttered environments	Shibao Yang, Pengcheng Liu, Nick Pears	Penelitian ini mempresentasikan studi komprehensif tentang performa berbagai motion planner dalam lingkungan berantakan yang berfokus pada efisiensi waktu, ketahanan, dan senstivitas parameter.	RRT, RRTConnect,RRTstar, TRRT, EST, SBL, KPIECE, BKPIECE, LBKPIECE, PDST, STRIDE, FMT	Eksperimen dalam penelitian ini dilakukan menggunakan tiga jenis robot, yaitu Franka, UR5, dan Kuka, yang dipilih untuk merepresentasikan variasi konfigurasi manipulator modern. Pengujian dilakukan pada tiga skenario lingkungan berantakan (*cluttered environments*)

Dari penelitian terdahulu, dapat disimpulkan bahwa sebagian besar studi telah menggunakan platform ROS dan simulator Gazebo secara efektif untuk menguji dan meninjau algoritma-algoritma motion planning berbasis sampling. Berbeda dengan penelitian-penelitian terdahulu, penelitian ini akan menguji 19 algoritma motion planning, yaitu: SBL, EST, LBKPIECE, BKPIECE, KPIECE, RRT, RRTConnect, RRT*, TRRT, PRM, PRM*, FMT, BFMT, PDST, STRIDE, BiTRRT, LBTRRT, dan ProjEST. Seluruh algoritma tersebut akan diuji pada robot MTE Lab Robotic yang menggunakan dasar dari robot UR5. Skenario yang digunakan juga berbeda, yaitu robot akan melakukan tugas penyusunan balok-balok warna dengan bantuan *Logical camera* sebagai sensor persepsi. Dengan demikian, penelitian ini tidak hanya membandingkan kinerja algoritma pada lingkungan simulasi, tetapi juga menekankan pada penerapan langsung untuk tugas manipulasi spesifik yang bersifat aplikatif. Pendekatan

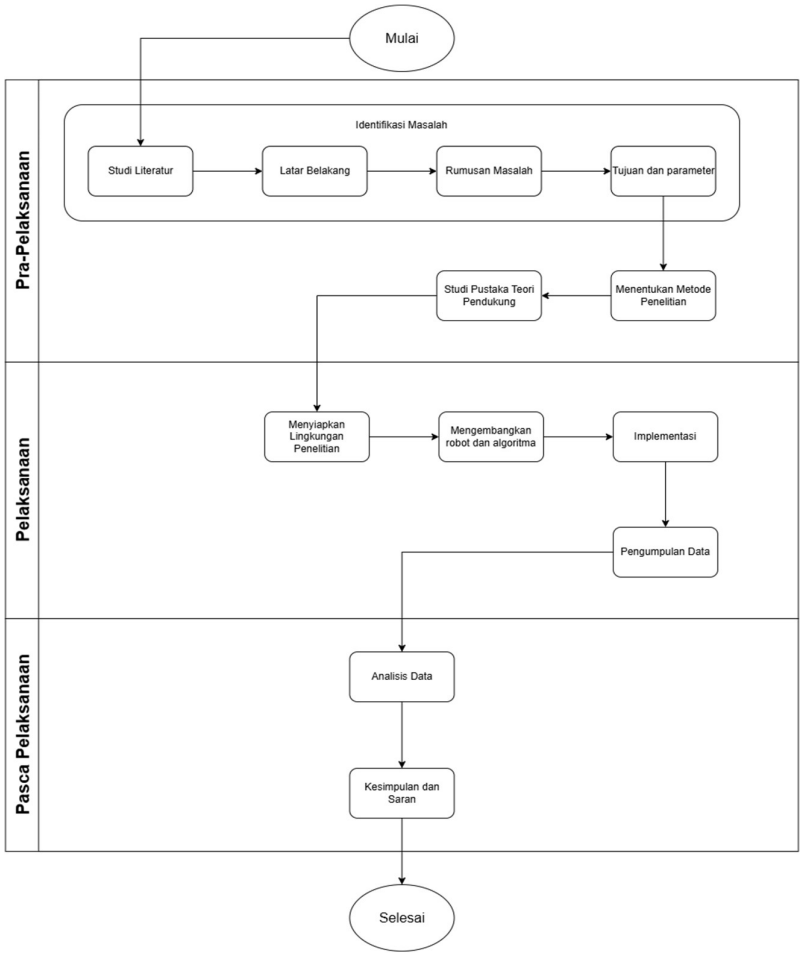
ini diharapkan memberikan solusi yang efisien, akurat, dan dapat direplikasi untuk skenario industri berbasis simulasi dan robotika nyata.

Untuk memperkuat validitas dan kontekstualisasi temuan penelitian ini, dilakukan analisis komparatif dengan studi *benchmarking* oleh Yang et al. (2023) yang berjudul "*Benchmarking of Robot Arm Motion Planning in Cluttered Environments*". Pemilihan studi ini sebagai referensi komparatif. Studi Yang et al. (2023)secara eksplisit mengevaluasi performa algoritma motion planning dalam lingkungan dengan berbagai tingkat clutter, mulai dari sederhana hingga kompleks. Hal ini memberikan spektrum perbandingan yang ideal untuk penelitian ini yang fokus pada lingkungan sederhana tanpa clutter. Dengan membandingkan kedua ekstrem spektrum kompleksitas lingkungan, dapat diperoleh pemahaman komprehensif tentang bagaimana karakteristik algoritma berubah seiring perubahan kompleksitas lingkungan.

Bab III
METODE PENELITIAN

3.1 Tahapan Penelitian

Penelitian ini terdiri dari beberapa tahapan utama yang melibatkan pengembangan algoritma *motion planning* untuk mengendalikan robot penyusun balok warna. Penelitian akan dilakukan dengan cara mengumpulkan data kuantitatif sebagai indikator keberhasilan pengembangan robot dan perancangan algoritma, yang mencakup metrik waktu, konsistensi, efisiensi dan daya komputasional untuk menilai efisiensi operasi robot pada siklus pengambilan dan penyusunan balok.



Gambar 3. 1 Alur metodologi penelitian

Pengembangan dilakukan menggunakan platform Robot Operating System (ROS) Noetic menggunakan robot rancangan MTE-ROBOTIC-LAB yang berbasis UR5 dengan dukungan perangkat lunak simulasi dan visualisasi RViz serta Gazebo, dan untuk perencanaan pergerakan robot digunakan package MoveIt.

3.2 Spesifikasi Penelitian

Dalam penelitian ini, peneliti menggunakan sejumlah perangkat dengan spesifikasi tertentu guna mendukung proses pelaksanaan penelitian. Adapun perangkat yang digunakan memiliki spesifikasi sebagai berikut:

- Ubuntu ≥ 20.04 LTS
- CUDA ≥ 12.7
- Rosdistro ≥ Noetic
- ros-noetic-catkin ≥ 0.8.10
- ros-noetic-moveit ≥ 1.17.3
- ros-noetic-roscpp ≥ 1.17.0
- ros-noetic-rospy ≥ 1.17.0
- Gazebo multi-robot simulator ≥ 11.11.0

3.3 Persiapan ROS

Dalam pengembangan sistem robotika, ROS berperan sebagai kerangka kerja utama yang memungkinkan komputer untuk mengendalikan berbagai komponen robot. Untuk menciptakan lingkungan simulasi virtual, pengguna perlu menginstal sejumlah paket ROS melalui terminal pada sistem operasi Ubuntu. Proses ini mencakup serangkaian tahapan yang dirancang agar robot dapat dikembangkan dengan kemampuan layaknya lengan untuk melakukan *operasi pick and place*. Rangkaian proses tersebut dijabarkan secara visual dalam Gambar 3.2

Gambar 3. 2 Langkah pengembangan *robot pick* dan *place* pada ROS

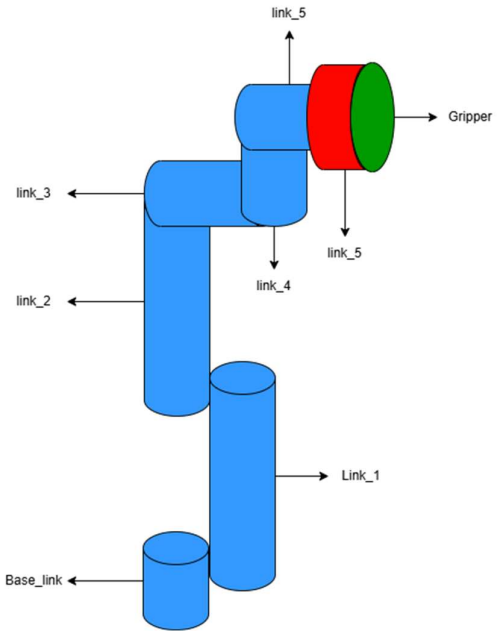
Dalam upaya mengembangkan robot yang mampu meniru proses berpikir dan perilaku lengan manusia dalam menjalankan tugas penyusunan, dibutuhkan sistem manipulasi cerdas yang memungkinkan robot berinteraksi secara dinamis dengan lingkungannya. Agar hal tersebut tercapai, robot perlu dibekali dengan kemampuan bergerak, merespons, dan mengambil keputusan layaknya lengan manusia. Sehingga untuk mewujudkan hal tersebut, langkah fundamental yang paling awal dilakukan adalah membangun sebuah kerangka kerja berbasis ROS. ROS berperan sebagai sistem operasi robotik yang mengatur komunikasi antar komponen seperti sensor, aktuator, kamera, hingga algoritma penyusun. Penjelasan mengenai langkah-langkah dalam ilustrasi ialah:

a. *Static and dynamic* URDF

Dalam ROS (*Robot Operating System*), pengembangan model dan sistem robot disimpan dalam format URDF (*Unified Robot Description Format*). URDF adalah file

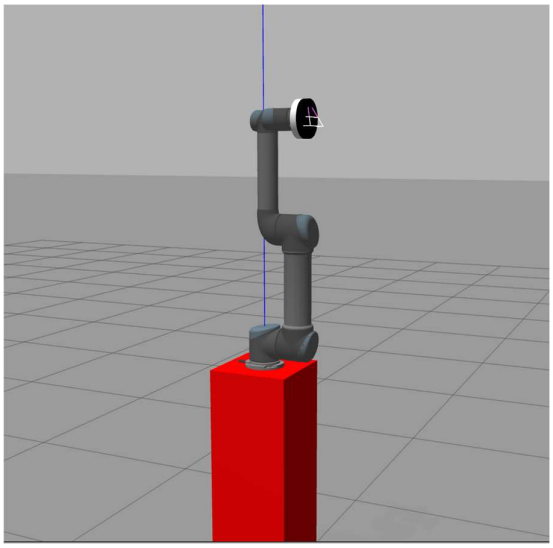
XML yang memuat seluruh informasi fisik robot, termasuk *link, joint, inertia*, hingga sensor-sensor yang ada pada robot. Selanjutnya, ROS akan membaca dan mengenali struktur robot melalui file URDF, yang kemudian dapat divisualisasikan secara grafis menggunakan RViz untuk memudahkan pemantauan dan analisis komponen serta gerakan robot.

Pada penelitian ini, peneliti mengembangkan model lengan robot dengan 6 sumbu kebebasan, yaitu lengan robot yang mampu bergerak pada 6 sumbu berbeda. Hal ini memungkinkan robot melakukan gerakan yang kompleks, sehingga mampu menyelesaikan tugas-tugas rumit (Khalil, 2019), dalam penelitian ini peneliti menggunakan robot 6 sumbu dari *package* Universal_robot yaitu UR5. Dalam kasus ini, robot dirancang untuk menyusun objek (balok) pada suatu area tertentu.



Gambar 3. 3 Diagram struktur robot UR5

Seluruh komponen URDF dalam proyek robot disusun secara modular dan efisien ke dalam satu berkas XACRO. XACRO (XML *Macros*) adalah sebuah bahasa berbasis XML yang dirancang khusus untuk menyederhanakan dan mempercepat penulisan *file* URDF yang kompleks dan berulang. Dengan menggunakan XACRO, peneliti dapat memanfaatkan fitur seperti variabel, fungsi makro, dan parameterisasi yang memungkinkan fleksibilitas tinggi dalam merancang struktur fisik robot secara dinamis.



Gambar 3. 4 Visualisasi robot UR5 dalam Gazebo

b. *Controls, Transmission,* dan *interface*

Setelah merancang bentuk fisik robot *pick and place* UR5, langkah selanjutnya adalah membangun fungsi gerak dari aktuator lengan dan *gripper* agar robot mampu melakukan gerakan manipulatif sesuai dengan informasi yang diberikan sensor kamera. Untuk mencapai tujuan tersebut, *transmission* digunakan untuk menghubungkan spesifikasi *joint* robot dengan aktuator yang memperbolehkan *joint* untuk bergerak. Dalam implementasi penelitian ini, peneliti mengkonfigurasikan setiap *joint* dengan elemen *transmission* sederhana yang memetakan pergerakan *joint* kepada masing-masing aktuator dengan rasio reduksi mekanis 1:1.

```
<transmission name="ur5_shoulder_pan_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="ur5_shoulder_pan_joint">

<hardwareInterface>hardware_interface/PositionJointInterface</hardw
areInterface>
  </joint>
  <actuator name="ur5_shoulder_pan_motor">

<hardwareInterface>hardware_interface/PositionJointInterface</hardw
areInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
```

Gambar 3. 5 Konfigurasi *transmission* salah satu *joint*

Untuk mengatur pergerakan robot peneliti mengimplementasikan arsitektur kontrol berlapis yang terdiri dari 2 komponen yaitu *Joint State Controller dan Trajectory Controller*. *Joint State Controller* merupakan komponen non-pengendali yang

mempublikasikan keadaan semua *joint* pada kecepatan 50Hz, menyediakan masukan yang penting sistem kontrol dan komponen pembantu seperti visualisasi.

```
joint_state_controller:
  type: joint_state_controller/JointStateController
  publish_rate: 50
```

Gambar 3. 6 *Joint State Controller*

Komponen selanjutnya yang digunakan untuk mengatur pergerakan robot adalah *Trajectory Controller*. Komponen ini berfungsi untuk mengoordinasikan pergerakan *multi-joint* dengan menerima titik-titik arah lintasan (*trajectory waypoints*) dan melakukan interpolasi secara halus di antara titik-titik tersebut.

Pengontrol ini menyediakan kemampuan penting dalam menjalankan gerakan yang kompleks dan terkoordinasi di seluruh *joint* secara bersamaan, yang merupakan syarat utama untuk memastikan operasi *pick and place* berjalan dengan lancar dan presisi.

```
ur5_1_controller:
  type: "position_controllers/JointTrajectoryController"
  joints:     [ur5_shoulder_pan_joint,     ur5_shoulder_lift_joint,
ur5_elbow_joint,
           ur5_wrist_1_joint, ur5_wrist_2_joint, ur5_wrist_3_joint]
```

Gambar 3. 7 *Trajectory Controller*

Komponen selanjutnya yang kita butuhkan pada langkah ini adalah komponen yang dapat mengatur dan menginterpretasikan perintah dari sistem kontrol yang akan dieksekusi oleh aktuator robot.

```
hardware_interface:
  joints:
    - ur5_shoulder_pan_joint
    - ur5_shoulder_lift_joint
    - ur5_elbow_joint
    - ur5_wrist_1_joint
    - ur5_wrist_2_joint
    - ur5_wrist_3_joint
    - gripper_base_to_gripper
  sim_control_mode: 1  # 0: position, 1: velocity
```

Gambar 3. 8 Komponen interface

Pada kontrol simulasi, peneliti memutuskan menggunakan mode kontrol berdasarkan *velocity* (kecepatan). Keputusan ini peneliti ambil berdasarkan kebutuhan akan dinamika pergerakan yang realistis dalam simulasi.

c. *Motion planning*

Setelah seluruh struktur robot dimodelkan dan sistem kontrol diatur, tahap selanjutnya dalam pengembangan sistem pick and place adalah merancang perencanaan gerak (*motion planning*). Perencanaan gerak dalam penelitian ini dibantu oleh MoveIt, yaitu sebuah framework dalam ROS yang dirancang khusus untuk memfasilitasi perencanaan lintasan dan kontrol inverse kinematics (IK).

Hal pertama yang harus kita lakukan adalah mendefinisikan sebuah planning group yang terdiri dari beberapa *joint*, hal ini dilakukan dengan tujuan memperbolehkan solver IK dan motion planner untuk membuat lintasan kepada beberapa *joint* secara bersamaan.

```
<group name="ur5_1_planning_group">
    <joint name="ur5_shoulder_pan_joint" />
    <joint name="ur5_shoulder_lift_joint" />
    <joint name="ur5_elbow_joint" />
    <joint name="ur5_wrist_1_joint" />
    <joint name="ur5_wrist_2_joint" />
    <joint name="ur5_wrist_3_joint" />
</group>
```

Gambar 3. 9 *planning* grup robot UR5

Dalam *motion planning* robot, *inverse kinematic* adalah proses mengalkulasikan sudut *joint* yang dibutuhkan untuk mencapai posisi dan orientasi *end-effector*. Namun MoveIt berkerja dalam ruang Cartesian 3 dimensi, oleh karena itu posisi dan orientasi end-effector dalam ruang Cartesian harus diubah menjadi sudut setiap *joint*. Oleh karena itu dibutuhkan IK *Solver*. Pada penelitian ini peneliti memilih *plugin Kinematics and Dynamics Library* (KDL) karena KDL menggunakan metode numerik untuk mengkomputasikan konfigurasi *joint*.

```
ur5_1_planning_group:
  kinematics_solver: kdl_kinematics_plugin/KDLKinematicsPlugin
  kinematics_solver_search_resolution: 0.005
  kinematics_solver_timeout: 0.005
```

Gambar 3. 10 Konfigurasi IK Solver

Selanjutnya, peneliti mengonfigurasikan algoritma-algoritma berbasis sampling untuk digunakan dalam *motion planning*. Pada MoveIt, peneliti menggunakan OMPL (*Open Motion planning Library*) yang menyediakan berbagai algoritma *motion planning* termasuk algoritma berbasis sampling.

```
ur5_1_planning_group:
  default_planner_config: None
  planner_configs:
    - RRT
    - RRTConnect
    - RRTstar
    - TRRT
    - PRM
    - PRMstar
    - FMT
    - BFMT
    - PDST
    - STRIDE
    - BiTRRT
    - LBTRRT
```

Gambar 3. 11 konfigurasi algoritma berbasis sampling

Setelah algoritma menghasilkan perencanaan lintasan, peneliti juga mengoptimasi perencanaan lintasan dengan *Covariant Hamiltonian Optimization for Motion planning* atau CHOMP. CHOMP adalah algoritma berulang yang akan meningkatkan hasil lintasan dari algoritma *motion planning*.

```
planning_time_limit: 10.0
max_iterations: 200
max_iterations_after_collision_free: 5
smoothness_cost_weight: 0.1
obstacle_cost_weight: 1.0
learning_rate: 0.01
```

Gambar 3. 12 Parameter CHOMP

Langkah pengoptimalan ini memastikan bahwa pergerakan robot tidak hanya layak tetapi juga mulus dan efisien, meminimalkan waktu pengoperasian.

d. *Logical camera Sensor*

Pada penelitian ini, peneliti mengimplementasikan sensor Logical Camera. Sensor *Logical camera* merupakan pendekatan baru untuk deteksi objek dalam simulasi robotika, yang secara mendasar berbeda dari metode *computer vision* tradisional. Daripada memproses data gambar mentah, kamera logis menyediakan identifikasi objek langsung dan estimasi pose melalui analisis geometris lingkungan simulasi. Pemilihan implementasi *Logical camera* dilakukan karena penelitian ini tidak berfokus pada *computer vision*, serta penggunaan sensor ini yang tidak memerlukan pemrosesan citra mentah dapat mengurangi beban komputasional secara signifikan. *Logical camera* beroperasi dengan mendefinisikan sebuah *frustum* penglihatan

(volume deteksi berbentuk piramida) yang memiliki parameter yang dapat dikonfigurasi.

```
<logical_camera>
  <near>0.06</near>
  <far>0.09</far>
  <horizontal_fov>1.1</horizontal_fov>
  <aspect_ratio>1.5</aspect_ratio>
</logical_camera>
```

Gambar 3. 13 Konfigurasi Sensor *Logical Camera*

Setiap objek yang berpotongan dengan *frustum* ini akan terdeteksi secara otomatis, dan posisinya dihitung relatif terhadap *frame* kamera. Metode ini menghilangkan beban komputasi yang terkait dengan pemrosesan citra, sekaligus menyediakan data lokasi objek yang presisi. Dengan demikian, sensor *Logical camera* tidak menghasilkan citra visual, melainkan data posisi dan orientasi dari objek-objek model (balok berwarna) di lingkungan simulasi.

```
pose {
  position {
    x: 0.88208281979670877
    y: 0.21404034088957063
    z: 0.94417799846205708
  }
  orientation {
    x: -8.4804316650129874e-05
    y: 0.70715593406976585
    z: 0.707057549672211
    w: -0.00031491216823218397
  }
}
```

Gambar 3. 14 *Output* sensor *logical camera*

Hasil dari *output* sensor akan dibawa oleh *node publisher*, selanjutnya komponen penting yang membutuhkan *output Logical camera* akan *subscribe* (berlangganan) kepada *node* tersebut.

Penelitian ini menggunakan pendekatan klasifikasi berbasis jenis model, di mana balok warna dibedakan melalui pengenal model yang telah ditentukan sebelumnya, bukan melalui analisis fitur visual. Metode ini memastikan pengenalan objek yang deterministik dengan tingkat kesalahan nol (*zero false positives*), yang sangat penting untuk tugas manipulasi robotik yang andal.

Sistem klasifikasi ini menggunakan konvensi penamaan sebagai berikut:

- Balok merah: package_r1, package_r2
- Balok Hijau: package_g1, package_g2
- Balok Biru: package_b1, package_b2

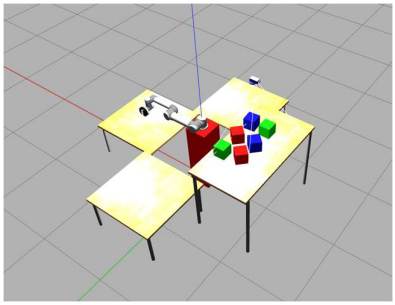
3.4 Lingkungan Simulasi

Penelitian ini dilakukan menggunakan *framework* ROS yang terintegrasi dengan simulator fisika Gazebo, sehingga menyediakan lingkungan simulasi robotik yang komprehensif. Gazebo dipilih karena kemampuannya dalam menyediakan mesin fisika yang andal, pemodelan sensor yang realistis, serta integrasi yang mulus dengan *middleware* ROS, sehingga memungkinkan representasi yang akurat terhadap tugas manipulasi robotik di dunia nyata.

Lingkungan simulasi dikonfigurasi dengan pemecah fisika *Open Dynamics Engine* (ODE), yang beroperasi dengan ukuran langkah maksimum sebesar 0.001 detik, faktor waktu nyata (real-*time factor*) sebesar 1.0, dan laju pembaruan (*update rate*) sebesar 1000 Hz. Parameter-parameter ini dipilih secara khusus untuk menyeimbangkan efisiensi komputasi dengan akurasi simulasi, guna memastikan perhitungan fisika yang stabil selama menjalankan tugas manipulasi yang kompleks.

Lingkungan simulasi dirancang untuk meniru skenario penyortiran industri dengan komponen utama sebagai berikut:

a. Permukaan Kerja: Model meja ditempatkan pada koordinat (-0.8, 0, 0) dengan rotasi 90 derajat terhadap sumbu-z, memberikan permukaan kerja yang stabil untuk tugas manipulasi objek.
b. Infrastruktur Penyortiran: Tiga meja sortir berwarna ditempatkan secara strategis dalam jangkauan kerja robot.
c. Sistem Persepsi: Sensor kamera logika diintegrasikan pada posisi (-0.8, 0, 2.0) dengan sudut 90 derajat, menyediakan pengamatan dari atas terhadap area kerja. Konfigurasi sensor ini memungkinkan pemantauan yang menyeluruh terhadap posisi dan kondisi objek selama proses penyortiran.



Gambar 3. 15 Lingkungan simulasi dalam gazebo

3.5 Metode Pengumpulan dan Analisis Data

3.5.1 Desain eksperimen

Penelitian ini menerapkan pendekatan eksperimental yang terstruktur untuk memastikan validitas dan reliabilitas data yang dikumpulkan. Proses penyortiran enam balok dilakukan sebanyak sepuluh kali untuk setiap algoritma perencanaan gerak yang diuji. Pemilihan jumlah iterasi ini didasarkan pada pertimbangan untuk memperoleh data yang cukup representatif dalam menghitung tingkat reliabilitas masing-masing algoritma, sekaligus mempertimbangkan efisiensi waktu komputasi yang diperlukan.

Setiap iterasi dimulai dengan konfigurasi lingkungan yang identik, dimana enam balok berwarna ditempatkan secara acak pada area kerja robot. Konsistensi kondisi awal ini sangat penting untuk memastikan bahwa perbedaan performa yang diamati berasal dari karakteristik algoritma itu sendiri, bukan dari variasi kondisi lingkungan. Robot UR5 kemudian melakukan tugas penyortiran dengan memindahkan setiap balok ke meja sortir yang sesuai berdasarkan warnanya, menggunakan algoritma perencanaan gerak yang telah ditentukan untuk iterasi tersebut.

Pengujian dilakukan dalam lingkungan simulasi Gazebo yang telah dikonfigurasi dengan komponen utama sebagai berikut. Robot manipulator UR5 ditempatkan pada posisi tetap dengan jangkauan kerja yang mencakup seluruh area penyortiran. Permukaan kerja berupa model meja yang ditempatkan pada koordinat minus nol koma delapan, nol, nol dengan rotasi sembilan puluh derajat terhadap sumbu z, memberikan landasan stabil untuk operasi manipulasi objek.

Tiga meja ditempatkan secara strategis dalam jangkauan kerja robot, masing-masing ditujukan untuk menampung balok dengan warna tertentu. Sistem persepsi menggunakan sensor kamera logika yang diintegrasikan pada sudut sembilan puluh derajat, menyediakan pengamatan dari atas terhadap area kerja. Konfigurasi sensor ini memungkinkan robot untuk mendeteksi posisi dan warna setiap balok secara akurat sebelum melakukan operasi *pick-and-place*.

3.5.2 Data Metrik

Proses dalam mengeksekusi setiap iterasi, terminal akan mengeluarkan berbagai metrik performa yang penting untuk evaluasi algoritma. Metrik waktu operasi mencakup rata-rata waktu yang diperlukan untuk satu operasi pick-and-place, total waktu yang dibutuhkan untuk menyelesaikan penyortiran seluruh enam balok, waktu operasi tercepat dalam satu iterasi, dan waktu operasi terlambat. Metrik temporal ini memberikan indikasi langsung tentang efisiensi algoritma dalam merencanakan dan mengeksekusi gerakan robot.

Metrik jarak perjalanan mengukur total jarak yang ditempuh oleh end-effector robot dalam ruang Cartesian dan ruang joint. Jarak Cartesian mengindikasikan panjang lintasan fisik yang dilalui oleh gripper robot di dunia tiga dimensi, sementara jarak joint space mengukur total perubahan sudut pada semua joint robot. Kedua metrik ini memberikan perspektif berbeda tentang efisiensi lintasan yang dihasilkan oleh algoritma perencanaan gerak.

Metrik penggunaan memori mencatat konsumsi memori minimum dan maksimum selama eksekusi tugas. Informasi ini penting untuk mengevaluasi beban komputasional yang diperlukan oleh setiap algoritma, yang menjadi pertimbangan krusial dalam implementasi sistem robot yang memiliki keterbatasan sumber daya komputasi.

3.5.3 Proses Agregasi Data

Agregasi data dilakukan dengan menghitung ukuran statistik deskriptif untuk setiap metrik yang dikumpulkan, Hal ini dilakukan untuk menyederhanakan 190 data dari 10 data per algoritma, kita hanya menganalisis 19 data saja. Untuk setiap algoritma dilakukan perhitungan sebagai berikut:

a. Nilai Rata-rata (Mean): Merepresentasikan performa tipikal dari algoritma dalam sepuluh iterasi pengujian. Nilai rata-rata memberikan indikasi umum tentang kecenderungan pusat dari distribusi data.
b. Nilai Median: Merepresentasikan nilai tengah dari distribusi data.
c. Nilai Minimum dan Maksimum: Menunjukkan rentang performa dari algoritma, dari kondisi terbaik hingga terburuk yang diamati selama pengujian.
d. Standar Deviasi (Standard Deviation - σ): Mengukur dispersi atau variabilitas data dari nilai rata-rata. Standar deviasi yang lebih rendah mengindikasikan konsistensi performa yang lebih baik.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}} \quad (4)$$

Dimana x_i adalah masing-masing data, μ adalah nilai rata-rata data dan N jumlah data

3.5.4 Normalisasi Data

Setelah data selesai diagregasikan, data metrik akan dinormalisasikan. Penelitian ini menggunakan metode normalisasi Min-Max untuk mentransformasi seluruh metrik ke dalam skala [0, 1]. Metode ini dipilih karena:

a. Mempertahankan distribusi relatif data
b. Mudah diinterpretasikan
c. Tidak terpengaruh oleh outlier ekstrem seperti Z-score normalization
d. Umum digunakan dalam sistem multi-criteria decision making

$$x_{norm} = 1 - \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (5)$$

Rumus ini adalah rumus normalisasi dimana semakin rendah nilai metrik maka semakin baik (*lower is better*).

3.5.5 Koefisien variasi

Dari delapan metrik performa yang dikumpulkan, tidak semua metrik memiliki daya diskriminasi (discriminatory power) yang sama dalam membedakan performa antar algoritma. Oleh karena itu, diperlukan proses seleksi sistematis untuk mengidentifikasi metrik yang paling efektif untuk evaluasi komparatif.

Metrik harus menunjukkan variabilitas substansial antar algoritma untuk dapat membedakan performa secara meaningful. Koefisien variasi (CV) digunakan sebagai ukuran normalized variability yang memungkinkan perbandingan objektif antar metrik dengan skala berbeda.

Koefisien Variasi Dihitung sebagai rasio standar deviasi terhadap nilai rata-rata, dinyatakan dalam persentase. CV memberikan ukuran variabilitas relatif yang memungkinkan perbandingan konsistensi antar metrik dengan skala berbeda.

$$CV = \frac{\sigma}{\mu} \times 100\% \quad (6)$$

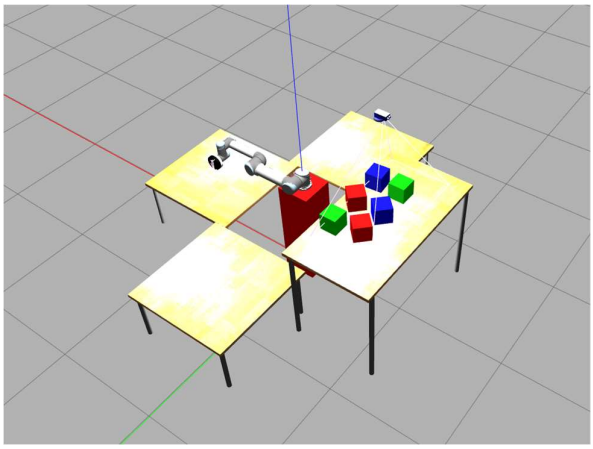
dimana σ adalah standar deviasi dan μ adalah mean.

- Metrik dikategorikan berdasarkan CV:
  a. CV < 5%: Variabilitas rendah (low discrimination) - metrik cenderung uniform antar algoritma, kurang efektif sebagai pembeda
  b. 5% ≤ CV < 15%: Variabilitas sedang (moderate discrimination) - metrik menunjukkan perbedaan yang detectable namun tidak dominan
  c. CV ≥ 15%: Variabilitas tinggi (high discrimination) - metrik menunjukkan perbedaan substansial yang dapat membedakan performa secara signifikan

BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

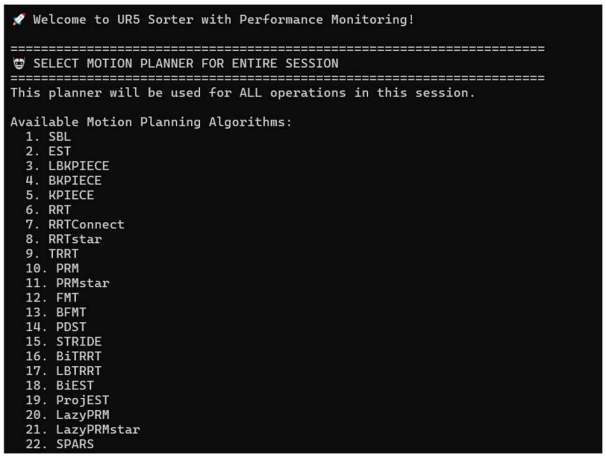
Setelah model robot dan lingkungan di Gazebo selesai diimplementasikan, langkah selanjutnya adalah menjalankan simulasi robot serta melakukan ekstraksi data. Bagian ini menjelaskan secara rinci proses pengumpulan data yang dilakukan untuk mengevaluasi performa algoritma perencanaan gerak dalam tugas penyortiran balok warna.



Gambar 4. 1 Robot dalam lingkungan simulasi

4.2 Proses Eksekusi Simulasi

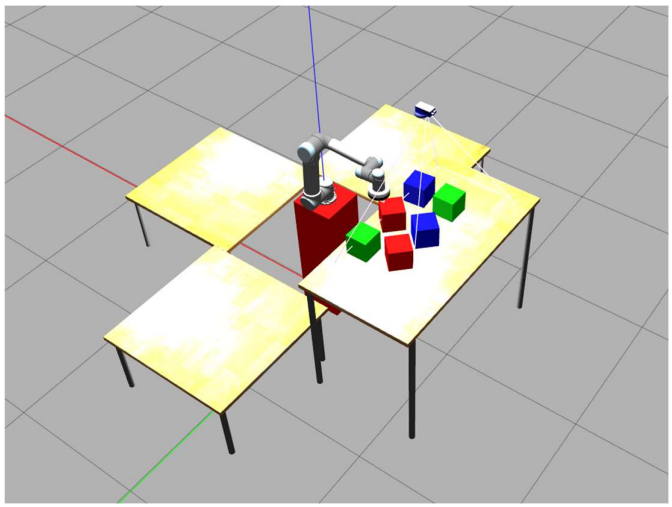
Proses simulasi dimulai dengan inisialisasi sistem melalui *file* peluncuran yang mengaktifkan semua komponen secara berurutan. Robot pertama kali bergerak ke pose *home*, yaitu konfigurasi awal yang telah ditentukan sebelumnya, sebagai titik referensi untuk semua operasi selanjutnya. Pose ini memastikan bahwa robot memulai setiap iterasi dari kondisi yang sama, mengurangi variabilitas yang tidak diinginkan dalam pengukuran.



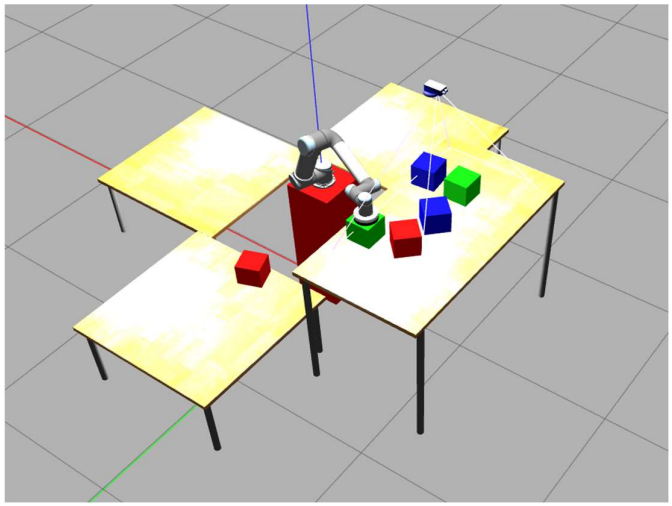
Gambar 4. 2 Menentukan algoritma perencanaan gerak

Setelah mencapai pose *home*, sensor kamera logika mulai mendeteksi balok-balok yang berada dalam area kerja. Data posisi dan orientasi dari setiap balok dikirimkan melalui topik ROS yang kemudian diproses oleh *node* pengontrol utama. Sistem klasifikasi berbasis jenis model mengidentifikasi warna setiap balok berdasarkan pengenal model yang telah ditentukan, tanpa memerlukan pemrosesan citra kompleks.

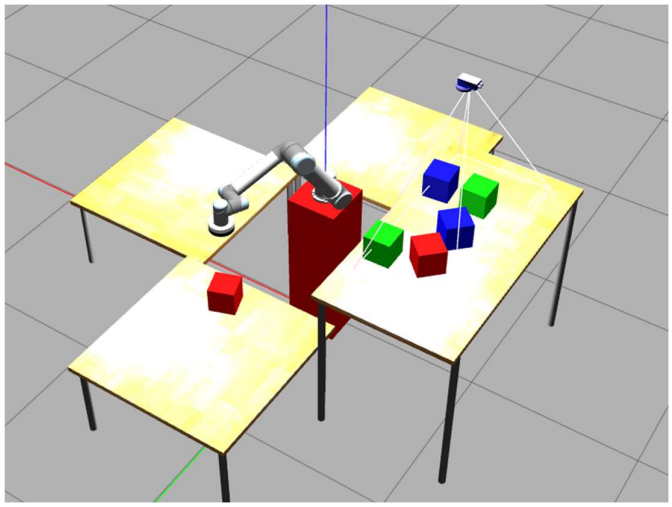
Robot kemudian melakukan operasi pick-and-place secara berurutan untuk setiap balok yang terdeteksi. Algoritma perencanaan gerak yang sedang diuji bertanggung jawab untuk menghasilkan lintasan yang aman dan efisien dari posisi saat ini menuju lokasi balok, kemudian ke meja sortir yang sesuai, dan akhirnya kembali untuk mengambil balok berikutnya. Proses ini berlanjut hingga semua enam balok berhasil disortir ke meja yang tepat.



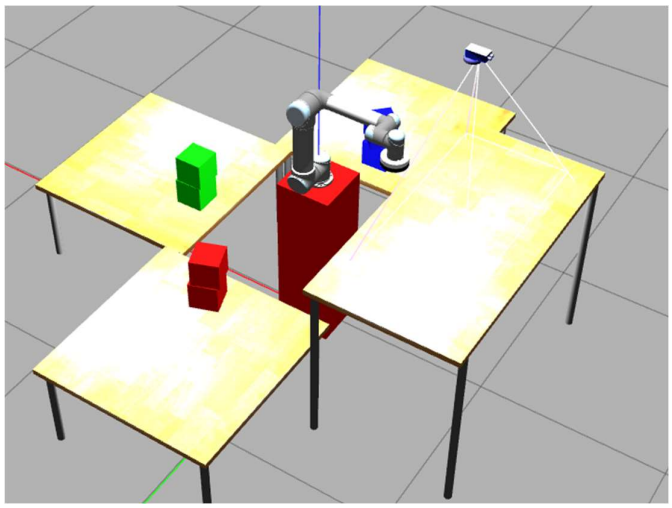
Gambar 4. 3 robot dalam pose “Home”



Gambar 4. 4 Robot mengidentifikasi dan mengangkat balok



Gambar 4. 5 Robot mulai menyortir balok



Gambar 4. 6 Robot selesai menyortir balok-balok

```
Final Performance Summary:
============================================================
Memory Usage: 115.25 MB
Virtual Memory: 1706.03 MB
CPU Usage: 0.0%
Average Operation Time: 19.368 seconds
Total Operations: 6
Peak Memory: 114.87 MB
Min Memory: 114.25 MB
Path Length Statistics (Cartesian):
  Total Distance: 15.2878 m
  Average per Motion: 0.4023 m
  Max Single Motion: 1.5323 m
  Min Single Motion: 0.0006 m
Path Length Statistics (Joint Space):
  Total Distance: 70.2542 rad
  Average per Motion: 1.8488 rad
  Max Single Motion: 5.8992 rad
  Min Single Motion: 0.0066 rad
Motion Planning Algorithm Usage:
  SBL: 39 times (100.0%)
============================================================
============================================================
Memory Usage: 115.25 MB
Virtual Memory: 1706.03 MB
CPU Usage: 0.0%
Average Operation Time: 19.368 seconds
Total Operations: 6
Peak Memory: 115.25 MB
Min Memory: 114.25 MB
Motion Planning Algorithm Usage:
  SBL: 39 times (100.0%)
============================================================
Total execution time: 116.209 seconds
Fastest operation: 15.935 seconds
Slowest operation: 24.654 seconds
Final Motion Planner Usage Summary:
  SBL: 39 times (100.0%)
Total Cartesian Distance Traveled: 15.2878 meters
Total Joint Space Distance Traveled: 70.2542 radians
```

Gambar 4. 7 Hasil output setelah penyortiran selesai

Tabel 4. 1 Output data algoritma BFMT

name	avg op time	total time	fastest	slowest	total cartersian dist	total joint dist	min memory	max memory
1	22.284	133.707	16.816	32.578	15.2004	70.4395	113.6	115.1
2	24.797	148.781	18.476	31.722	15.1816	76.6913	113.42	114.92
3	22.003	132.016	17.334	30.138	15.1746	70.5832	113.33	114.71
4	18.984	113.905	16.753	23.951	15.2233	63.6416	113.39	115.02
5	23.105	138.631	18.505	28.774	15.2367	74.545	113.85	115.22
6	18.922	113.53	17.751	21.369	15.1678	63.8982	113.96	115.34
7	19.734	118.406	16.974	27.282	15.1533	66.8403	113.89	115.14
8	20.889	125.332	18.088	23.962	15.2087	67.9219	113.55	115.05
9	23.371	140.227	18.186	29.316	15.1943	73.1723	113.1	114.73
10	24.673	148.037	19.952	30.158	15.2349	77.0733	114.13	115.26

Data yang dikumpulkan dari setiap iterasi disimpan dalam format terstruktur yang memudahkan analisis selanjutnya. Tabel empat koma delapan menunjukkan contoh struktur data untuk sepuluh iterasi dari salah satu algoritma yang diuji. Setiap baris merepresentasikan satu iterasi lengkap, dengan kolom yang mencatat semua metrik yang telah disebutkan sebelumnya.

Format data ini memungkinkan agregasi dan analisis statistik yang efisien. Nilai rata-rata, standar deviasi, dan ukuran statistik lainnya dapat dengan mudah dihitung untuk setiap algoritma, memberikan gambaran komprehensif tentang performa dan konsistensi masing-masing algoritma dalam menyelesaikan tugas penyortiran balok warna. Setelah pengumpulan data untuk semua sembilan belas algoritma selesai, tahap selanjutnya adalah melakukan agregasi dan normalisasi data untuk memungkinkan perbandingan yang objektif dan sistematis antar algoritma.

4.3 Agregasi Data

Setelah menjalankan simulasi sebanyak sepuluh kali untuk setiap algoritma perencanaan gerak, tahap selanjutnya adalah melakukan agregasi data dari seluruh 19 algoritma yang diuji. Proses ini bertujuan untuk menghasilkan representasi statistik yang komprehensif dari performa masing-masing algoritma dan memungkinkan perbandingan yang objektif antar algoritma. Hasil dari agregasi data dapat dilihat pada Tabel 4.2 Hasil agregasi data.

Tabel 4. 2 Hasil agregasi data

	avg op time		total time		fastest		slowest		total cartersian dist		total joint dist		min memory		max memory	
	Mean ± SD	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**
bfmt	21.8762 ± 2.072	9.472	131.257 ± 12.432	9.472	17.8835 ± 0.933	5.22	27.925 ± 3.514	12.585	15.1975 ± 0.027	0.178	70.4806 ± 4.644	6.59	113.622 ± 0.309	0.272	115.049 ± 0.201	0.175
lbkpiece	22.242 ± 2.183	9.816	133.452 ± 13.099	9.816	18.3798 ± 0.491	2.671	28.167 ± 5.195	18.442	15.2329 ± 0.051	0.334	67.5394 ± 5.025	7.44	113.715 ± 0.230	0.203	115.076 ± 0.292	0.254
biest	16.9814 ± 1.585	9.334	101.889 ± 9.511	9.334	14.4832 ± 0.395	2.728	21.195 ± 5.094	24.033	15.2072 ± 0.035	0.232	63.9072 ± 3.905	6.111	113.715 ± 0.313	0.275	114.99 ± 0.179	0.156
bitrrt	17.1734 ± 1.501	8.742	103.040 ± 9.008	8.742	14.9724 ± 0.354	2.365	21.5951 ± 4.766	22.071	15.1952 ± 0.025	0.163	64.2970 ± 4.072	6.333	113.776 ± 0.337	0.297	114.9001 ± 0.171	0.149
bkpiece	18.9634 ± 1.260	6.645	113.780 ± 7.560	6.645	16.3885 ± 0.647	3.947	25.1365 ± 4.398	17.495	15.2485 ± 0.118	0.777	64.8141 ± 2.449	3.779	113.555 ± 0.240	0.211	114.992 ± 0.231	0.201
est	17.4768 ± 1.238	7.083	104.861 ± 7.426	7.081	15.0649 ± 0.526	3.49	23.2673 ± 4.881	20.979	15.2076 ± 0.031	0.207	64.3171 ± 2.372	3.688	113.7 ± 0.323	0.284	114.863 ± 0.185	0.161
fmt	19.3341 ± 0.306	1.581	116.004 ± 1.834	1.581	18.0453 ± 0.389	2.158	21.1631 ± 1.359	6.421	15.2175 ± 0.118	0.774	60.9592 ± 0.744	1.22	113.579 ± 0.274	0.241	115.141 ± 0.370	0.322
kpiece	17.1871 ± 0.749	4.359	103.1233 ± 4.495	4.359	15.2483 ± 0.447	2.928	22.564 ± 2.898	12.842	15.1956 ± 0.036	0.234	63.7253 ± 1.578	2.476	113.799 ± 0.244	0.214	114.888 ± 0.222	0.193
lbtrrt	47.9764 ± 2.101	4.378	287.858 ± 12.603	4.378	44.866 ± 0.431	0.96	53.1337 ± 3.938	7.412	15.1781 ± 0.015	0.099	67.0897 ± 6.216	9.265	113.511 ± 0.284	0.25	115.074 ± 0.284	0.247
pdst	16.4194 ± 1.061	6.464	98.516 ± 6.368	6.464	14.6716 ± 0.407	2.772	19.4184 ± 4.101	21.121	15.1875 ± 0.026	0.17	62.3157 ± 2.255	3.619	113.797 ± 0.238	0.209	114.821 ± 0.255	0.222
prm	17.0488 ± 1.787	10.482	102.293 ± 10.723	10.482	14.785 ± 0.316	2.135	21.613 ± 5.318	24.605	15.1803 ± 0.017	0.111	64.0762 ± 4.061	6.338	113.9831 ± 0.378	0.332	115.2806 ± 0.661	0.574

	avg op time		total time		fastest		slowest		total cartersian dist		total joint dist		min memory		max memory	
	Mean ± SD	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**	**Mean ± SD**	**CV**
prmstar	45.9172 ± 0.376	0.818	275.503 ± 2.255	0.818	44.5491 ± 0.267	0.6	47.9369 ± 1.728	3.604	15.1710 ± 0.019	0.127	61.1677 ± 0.901	1.473	113.817 ± 0.309	0.271	115.093 ± 0.190	0.165
projest	17.4903 ± 1.658	9.48	104.941 ± 9.948	9.479	14.9642 ± 0.378	2.529	22.6532 ± 5.930	26.175	15.1796 ± 0.023	0.154	64.0582 ± 4.150	6.478	113.789 ± 0.401	0.352	115.138 ± 0.217	0.188
rrt	16.839 ± 0.698	4.144	101.034 ± 4.186	4.144	15.0005 ± 0.385	2.565	21.7062 ± 3.453	15.907	15.1825 ± 0.027	0.179	62.4526 ± 1.807	2.893	113.74 ± 0.342	0.301	114.867 ± 0.213	0.186
rrtconnect	17.8557 ± 1.610	9.016	108.673 ± 8.301	7.639	14.8552 ± 0.950	6.398	23.2697 ± 2.788	11.982	15.2758 ± 0.238	1.557	67.3784 ± 3.308	4.909	113.907 ± 0.430	0.377	115.006 ± 0.178	0.155
rrtstar	46.6242 ± 0.759	1.627	279.745 ± 4.551	1.627	44.7461 ± 0.299	0.669	51.5201 ± 3.879	7.53	15.1660 ± 0.031	0.202	63.1845 ± 2.024	3.204	113.678 ± 0.284	0.25	114.914 ± 0.224	0.195
sbl	17.1826 ± 0.628	3.652	103.095 ± 3.766	3.653	15.3328 ± 0.563	3.674	21.3473 ± 3.922	18.372	15.2233 ± 0.043	0.28	62.8735 ± 2.173	3.456	113.801 ± 0.328	0.289	115.103 ± 0.294	0.256
stride	17.4984 ± 1.028	5.874	104.990 ± 6.167	5.874	14.914 ± 0.544	3.649	23.4785 ± 4.157	17.706	15.2082 ± 0.068	0.45	63.5940 ± 1.792	2.818	113.729 ± 0.282	0.248	114.967 ± 0.083	0.072
trrt	16.6483 ± 1.520	9.133	101.302 ± 7.648	7.55	13.8023 ± 2.706	19.604	22.2135 ± 6.916	31.136	15.3033 ± 0.353	2.31	63.0377 ± 3.110	4.934	113.699 ± 0.302	0.266	114.898 ± 0.175	0.152

4.3.1. Hasil analisis Agregasi Data

Dari data agregasi pada Tabel 4.2 , dilakukan evaluasi mendalam terhadap karakteristik performa setiap metrik dan algoritma. Analisis ini menggunakan koefisien variasi (CV) sebagai ukuran utama untuk menentukan daya diskriminasi setiap metrik dalam membedakan performa antar algoritma.

a. Metrik dengan Variasi Rendah (CV < 5%):
   - *Total Cartesian Distance*: Menunjukkan koefisien variasi yang sangat rendah (CV rata-rata 0.23%), mengindikasikan bahwa seluruh algoritma menghasilkan panjang lintasan Cartesian yang hampir identik (berkisar 15.17-15.30 unit). Hal ini menunjukkan bahwa metrik ini kurang efektif sebagai pembeda antar algoritma. Hal ini wajar karena semua algoritma *motion planning* berbasis sampling berusaha menemukan path dari *start* ke *goal* yang *feasible*. Dalam *Cartesian space*, jarak geometris antara dua titik relatif tetap, sehingga variasi minimal.
   - *Minimum Memory*: Memiliki koefisien variasi rendah (CV rata-rata 1.92%), dengan rentang penggunaan memori 103.98-113.91 MB. Variasi yang kecil ini menunjukkan bahwa perbedaan konsumsi memori minimum antar algoritma tidak signifikan.

b. Metrik dengan Variasi Sedang (5% < CV < 15%):
   - *Total Joint Distance*: Menunjukkan koefisien variasi sedang (CV rata-rata 3.58%), dengan rentang nilai dari 60.96 unit (fmt) hingga 70.48 unit (bfmt), menghasilkan perbedaan absolut 9.52 unit atau sekitar 15.6% dari nilai minimum. Perbedaan 15.6% antar algoritma menunjukkan bahwa metrik ini cukup signifikan sebagai pembeda performa, terutama untuk aplikasi yang memprioritaskan efisiensi energi atau *life cycle* komponen mekanis. *Joint distance* yang lebih pendek mengindikasikan gerakan yang lebih efisien dalam ruang konfigurasi robot, potensi pengurangan keausan mekanis pada joint, konsumsi energi yang lebih rendah, danjalur yang lebih halus dalam *joint space*. Variasi ini terjadi karena meskipun semua algoritma mencapai goal yang sama dalam *Cartesian space*, strategi *sampling* dan eksplorasi yang berbeda menghasilkan *trajectory* dengan kompleksitas pergerakan joint yang berbeda-beda.

c. Metrik dengan Variasi Tinggi (CV > 15%):

- Metrik Waktu Eksekusi: *Average operation time, total time, fastest, dan slowest* menunjukkan koefisien variasi tinggi (CV: 38-53%), mengindikasikan bahwa metrik-metrik ini merupakan pembeda yang kuat antar algoritma. Perbedaan waktu eksekusi antar algoritma dapat mencapai 140-154% dari nilai rata-rata.

4.4 Normalisasi dan Evaluasi Algoritma

Sebelum melakukan pemeringkatan algoritma, dilakukan seleksi metrik yang akan digunakan berdasarkan hasil analisis koefisien variasi dan korelasi pada bagian sebelumnya. Metrik yang dipilih adalah *average operation time*, standar deviasi (sebagai ukuran konsistensi), *total joint distance*, dan *max memory*.

4.4.1 Justifikasi pemilihan metrik

Sebagaimana diidentifikasi oleh "*Evaluating Motion Planning Performance Workshop*" (2022), tidak ada standarisasi universal untuk metrik evaluasi dan *threshold* klasifikasi dalam *motion planning* (Constantinos Chamzas et al., 2022) dikarenakan bidang motion planning tidak memiliki dataset dan metrik performa yang terstandarisasi. Dikarenakan hal itu, penelitian ini mengadopsi pendekatan data-driven dan *context-specific* dengan justifikasi sebagai berikut:

a. *Average Operation Time* (dipilih sebagai representasi metrik waktu)

Average operation time dipilih sebagai metrik utama untuk mengukur kecepatan eksekusi dengan pertimbangan sebagai berikut:

- Koefisien variasi tinggi (CV: 47.11%): Menunjukkan daya diskriminasi yang kuat dalam membedakan performa antar algoritma, dengan rentang nilai dari 16.42 ms hingga 47.98 ms (perbedaan hingga 192%).
- Format standar dalam *benchmark*: Merupakan metrik yang umum digunakan dalam literatur *motion planning*, memudahkan perbandingan dengan penelitian lain.
- *Scale-independent*: Nilai yang ternormalisasi per operasi tidak bergantung pada jumlah iterasi yang dijalankan, sehingga lebih umum untuk berbagai konfigurasi eksperimen.
- Menghindari redundansi: Analisis korelasi menunjukkan bahwa semua metrik waktu (*average operation time, total time, fastest, slowest*) memiliki korelasi sangat kuat, sehingga pemilihan satu metrik sudah cukup merepresentasikan dimensi kecepatan eksekusi untuk menghindari *double-counting*.

b. Standar Deviasi (Konsistensi)

Standar deviasi dari *average operation time* digunakan sebagai ukuran konsistensi performa:

- Mengukur dimensi berbeda: Tidak berkorelasi dengan average time, sehingga memberikan informasi independen tentang prediktabilitas dan *reliability* algoritma.
- Penting untuk aplikasi praktis: Algoritma dengan standar deviasi rendah memberikan performa yang lebih dapat diprediksi, krusial untuk sistem *real-time* dan *safety-critical.*

c. Total Joint Distance

Pertimbangan dalam memilih *Total joint distance* dipilih, yaitu:

- Variasi signifikan (15.6%): Rentang dari 60.96 hingga 70.48 unit menunjukkan perbedaan substansial antar algoritma.
- Relevansi praktis: Metrik ini mencerminkan efisiensi gerakan dalam *joint space*, yang berimplikasi pada konsumsi energi, keausan mekanis, dan *smoothness trajectory*.

d. Max Memory

Max memory consumption dipertahankan dengan bobot rendah karena:

- Pertimbangan *hardware*: Meskipun variasi rendah perbedaan antara algoritma seperti dapat menjadi relevan untuk sistem *embedded* atau *resource-constrained*.
- Faktor pelengkap: Memberikan gambaran lengkap tentang profil resource usage algoritma.

➢ Metrik yang tidak digunakan:

a. Total Cartesian Distance

Total cartesian distance tidak dipilih karena koefisien variasi sangat rendah (CV: 0.23%), dengan rentang hanya 0.13 unit (15.17-15.30), mengindikasikan bahwa semua algoritma menghasilkan panjang lintasan Cartesian yang hampir identik. Hal ini menjadikan metrik ini tidak efektif sebagai pembeda performa.

b. Minimum Memory

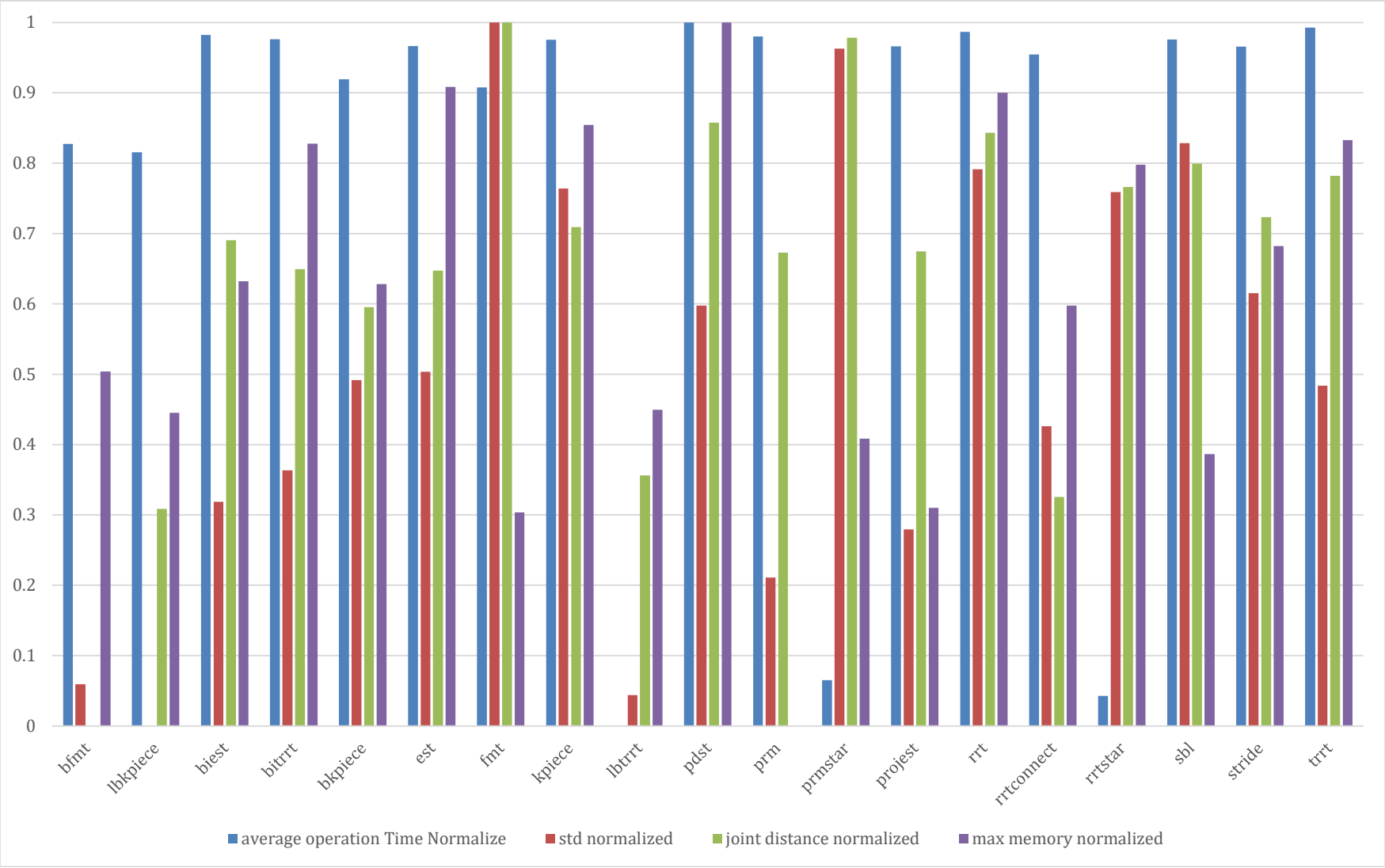
Minimum memory tidak dipilih karena memiliki korelasi tinggi dengan max memory (r = 0.666) dan koefisien variasi yang rendah (CV: 1.92%), sehingga redundan dan tidak memberikan informasi tambahan yang signifikan.

4.4.2 Normalisasi data

Metrik-metrik yang dipilih memiliki skala pengukuran yang berbeda-beda: *average operation time* dalam satuan detik (s), *memory* dalam *megabyte* (MB), *joint distance* dalam unit jarak, dan standar deviasi dalam detik (s). Perbedaan skala ini menyebabkan metrik dengan nilai absolut lebih besar dalam representasi hasil.

Tabel 4. 3 Tabel Hasil Normalisasi

Algorithm	Avg Op Time (s)	Std Dev (s)	Joint Distance (unit)	Max Memory (MB)	Avg Op Time Normalized	Std Dev Normalized	Joint Dist Normalized	Memory Normalized
bfmt	21.876	2.072	70.481	115.049	0.827	0.059	0.000	0.504
biest	16.981	1.585	63.907	114.990	0.982	0.319	0.690	0.632
bitrrt	17.173	1.501	64.297	114.900	0.976	0.363	0.649	0.828
bkpiece	18.963	1.260	64.814	114.992	0.919	0.492	0.595	0.628
est	17.477	1.238	64.317	114.863	0.967	0.504	0.647	0.909
fmt	19.334	0.306	60.959	115.141	0.908	1.000	1.000	0.304
kpiece	17.187	0.749	63.725	114.888	0.976	0.764	0.710	0.854
lbkpiece	22.242	2.183	67.540	115.076	0.816	0.000	0.309	0.445
lbtrrt	47.976	2.101	67.090	115.074	0.000	0.044	0.356	0.450
pdst	16.419	1.061	62.316	114.821	1.000	0.598	0.858	1.000
prm	17.049	1.787	64.076	115.281	0.980	0.211	0.673	0.000
prmstar	45.917	0.376	61.168	115.093	0.065	0.963	0.978	0.408
projest	17.490	1.658	64.058	115.138	0.966	0.280	0.675	0.310
rrt	16.839	0.698	62.453	114.867	0.987	0.791	0.843	0.900
rrtconnect	17.856	1.610	67.379	115.006	0.955	0.305	0.326	0.598
rrtstar	46.624	0.759	63.185	114.914	0.043	0.759	0.766	0.798
sbl	17.183	0.628	62.874	115.103	0.976	0.829	0.799	0.386
stride	17.498	1.028	63.594	114.967	0.966	0.615	0.723	0.682
trrt	16.648	1.521	63.038	114.898	0.993	0.353	0.782	0.833



Gambar 4. 8 Visualisasi hasil data normalisasi

4.4.3 Evaluasi Algoritma

Berdasarkan hasil data dari normalisasi, maka metrik-metrik komparatif algoritma pencernaan gerak berbasis sampling akan dilanjutkan dengan mengevaluasi algoritma, dengan mempertimbangkan parameter, antara lain:

a. Analisis Performa Kecepatan

Berdasarkan metrik rata-rata waktu operasi, performa algoritma dapat dikategorikan ke dalam kelompok berbeda yang menunjukkan rentang kecepatan yang cukup signifikan.

Kelompok pertama adalah kelompok ultra-cepat dengan rentang waktu antara 16,4 hingga 17,5 detik, mencakup sebelas algoritma atau sekitar 58% dari total algoritma yang diuji. PDST mencatat waktu tercepat absolut, yaitu 16,42 detik, yang menetapkan *baseline* optimal bagi *throughput* sistem. Dalam kelompok ini, TRRT mengikuti dengan selisih sangat tipis hanya 0,23 detik lebih lambat (16,65 detik), diikuti oleh RRT dengan 16,84 detik, yang menunjukkan performa sangat kompetitif dengan selisih hanya 0,42 detik dari algoritma tercepat. Algoritma BiEST, PRM, BiTRRT, SBL, KPIECE, EST, ProjEST, dan STRIDE juga menunjukkan performa yang sangat kompetitif dengan waktu eksekusi tidak melebihi 17,5 detik.

Interpretasi terhadap kelompok ultra-cepat ini menunjukkan bahwa untuk tugas *pick-and-place* sederhana tanpa rintangan kompleks, mayoritas *sampling-based planner* memberikan *throughput* yang sangat baik dan sebanding. Perbedaan maksimum dalam kelompok ini hanya 1,08 detik atau sekitar 6,6%, yang dalam konteks industri dengan target siklus waktu 15–20 detik dapat dianggap tidak signifikan secara praktis. Hal ini mengindikasikan bahwa pemilihan algoritma dalam kelompok ini sebaiknya lebih dipertimbangkan berdasarkan faktor lain seperti konsistensi atau efisiensi lintasan, bukan semata-mata kecepatan.

Kelompok ketiga adalah kelompok lambat, dengan rentang waktu antara 45,9 hingga 48,0 detik, yang menunjukkan penalti performa yang sangat besar. PRM* tercatat pada 45,92 detik, menjadikannya 2,8 kali lebih lambat dibandingkan algoritma tercepat. RRT* dengan 46,62 detik menunjukkan bahwa *asymptotic optimality* memerlukan operasi *rewiring* yang sangat intensif secara komputasional. LBTRRT dengan 47,98 detik menjadi algoritma paling lambat, dengan *overhead* pohon *lower bound* yang signifikan—hampir 2,9 kali lebih lambat dibandingkan PDST.

Interpretasi terhadap kelompok lambat ini mengonfirmasi *trade-off* fundamental yang banyak dibahas dalam literatur *motion planning*, yaitu antara kualitas solusi dan

kecepatan komputasional. Algoritma dengan jaminan *optimality* menunjukkan penalti performa yang drastis, berkisar antara 180% hingga 192% lebih lambat dibandingkan algoritma tercepat. Untuk *task pick-and-place* sederhana seperti yang digunakan dalam penelitian ini, jaminan *optimality* tampaknya tidak layak dipertahankan mengingat tingginya biaya waktu komputasi yang harus dibayar. Hasil ini memberikan wawasan penting bahwa pemilihan algoritma harus mempertimbangkan kompleksitas tugas dan lingkungan, karena manfaat optimality guarantee mungkin tidak sepadan dengan biaya komputasi pada skenario yang relatif sederhana.

b. Analisis Konsistensi

Analisis berdasarkan standar deviasi rata-rata waktu operasi mengungkapkan variasi performa yang sangat luas antar algoritma, menunjukkan bahwa konsistensi merupakan dimensi performa yang berbeda dari rata-rata waktu operasi.

Kelompok sangat konsisten dengan standar deviasi kurang dari 5 detik mencakup enam algoritma yang menunjukkan stabilitas luar biasa. FMT memimpin dengan standar deviasi hanya 1,83 detik, menghasilkan *coefficient of variat*ion sekitar 9,5%, yang merupakan tingkat konsistensi luar biasa untuk *sampling-based planner*. PRM* mengikuti dengan 2,26 detik, menunjukkan perilaku konvergensi yang stabil pada *planner* yang optimal. SBL dengan 3,77 detik memperlihatkan bahwa pendekatan lazy bi-directional memberikan perilaku yang sangat *predictable*. RRT dengan 4,19 detik menunjukkan konsistensi yang mengesankan untuk *tree-based planner* yang memiliki unsur stokastik bawaan. KPIECE (4,50 detik) dan RRT* (4,55 detik) melengkapi kelompok ini.

Interpretasi dari kelompok sangat konsisten ini menunjukkan adanya keberagaman pendekatan algoritmik yang mampu menghasilkan konsistensi tinggi. Kelompok ini mencakup *asymptotically optimal planners* (FMT, PRM*, RRT*), *grid-based planners* (KPIECE), pendekatan *lazy collision checking* (SBL), serta *classic tree-based planner* (RRT). Tidak ada satu paradigma algoritmik yang secara dominan unggul dalam hal konsistensi, yang mengindikasikan bahwa konsistensi lebih bergantung pada detail implementasi dan penyesuaian parameter daripada pada pendekatan algoritmik dasarnya. Yang menarik adalah performa RRT, dengan standar deviasi hanya 4,19 detik atau sekitar 25% dari nilai rata-ratanya, menunjukkan stabilitas yang luar biasa untuk algoritma berbasis pohon dengan sifat stokastik yang tinggi.

Kelompok sangat bervariasi dengan standar deviasi di atas 9 detik mencakup enam algoritma dengan perilaku konvergensi yang sulit diprediksi. BiEST (9,51 detik) menunjukkan peningkatan variabilitas sebesar 28% dibandingkan EST versi *single-direction*. ProjEST (9,95 detik) memperlihatkan bahwa proyeksi ke ruang berdimensi lebih rendah menambah ketidakpastian dalam konvergensi. PRM (10,72 detik) menunjukkan bahwa konstruksi *roadmap* sangat bervariasi pada tugas sederhana. BFMT (12,43 detik) menjadi algoritma paling bervariasi, dengan peningkatan 578% dibandingkan FMT standar. LBTRRT (12,60 detik) dan LBKPIECE (13,10 detik) juga menunjukkan perilaku konvergensi yang sangat tidak stabil.

Pola yang sangat jelas dari analisis ini adalah bahwa varian bi-directional secara konsisten lebih bervariasi dibandingkan versi *single-query-nya*. BiEST menunjukkan peningkatan variabilitas 28% dibandingkan EST, BFMT 578% dibandingkan FMT, dan BKPIECE 68% dibandingkan KPIECE. Hal ini mengindikasikan bahwa koordinasi dua pohon dalam lingkungan sederhana tanpa *narrow passages* waktu pertemuan dua pohon sangat bergantung pada *random sampling*, yang menyebabkan variansi besar dalam waktu eksekusi total. Temuan ini penting bagi aplikasi yang memerlukan perilaku prediktif, seperti *collaborative robotics* atau *safety-critical systems*, di mana variabilitas tinggi dapat menjadi faktor risiko yang tidak dapat diterima.

c. Analisis Efisiensi

Analisis berdasarkan *total joint distance* menunjukkan variasi sebesar 15,6% antara algoritma paling efisien dan paling tidak efisien, yang merupakan rentang cukup signifikan serta memberikan *discriminatory power* yang baik bagi metrik ini.

Kelompok sangat efisien dengan rentang 60,9 hingga 63,0 unit mencakup enam algoritma dengan kualitas lintasan yang sangat baik. FMT memimpin dengan 60,96 unit, menetapkan *baseline* optimal yang memvalidasi jaminan teoretisnya sebagai perencana *asymptotically optimal*. PRM* sangat dekat dengan 61,17 unit, hanya 0,21 unit lebih panjang dari optimal. PDST (62,32 unit) sebagai algoritma tercepat juga menunjukkan efisiensi luar biasa, hanya 2,2% lebih panjang dari optimal. RRT (62,45 unit) memperlihatkan keseimbangan sangat baik antara kecepatan dan efisiensi, dengan selisih hanya 2,4% dari optimal. SBL (62,87 unit) dan TRRT (63,04 unit) juga menunjukkan efisiensi lintasan yang tinggi.

Interpretasi dari kelompok ini menunjukkan bahwa FMT mendominasi dalam efisiensi lintasan, sesuai dengan fondasi teoretisnya. Namun, temuan yang paling

menarik dan penting secara praktis adalah bahwa PDST dan RRT—dua algoritma tercepat—juga sangat efisien dengan panjang lintasan hanya sekitar 2% lebih panjang dari optimal. Hal ini menunjukkan bahwa pada lingkungan sederhana tanpa rintangan kompleks, tidak terdapat *trade-off* signifikan antara kecepatan dan kualitas lintasan bagi algoritma tersebut. Dengan demikian, PDST dan RRT menjadi pilihan sangat menarik untuk aplikasi praktis yang membutuhkan kecepatan sekaligus efisiensi lintasan.

Kelompok kurang efisien dengan rentang 67,1 hingga 70,5 unit mencakup empat algoritma dengan lintasan yang relatif lebih panjang. LBTRRT (67,09 unit) menunjukkan bahwa *lower bound pruning* tidak menghasilkan lintasan lebih pendek meskipun dengan *computational overhead* yang tinggi. RRTConnect (67,38 unit) memperlihatkan bahwa strategi koneksi *bi-directional* dapat menghasilkan lintasan yang lebih panjang akibat titik sambung yang kurang optimal. LBKPIECE (67,54 unit) menjadi yang paling tidak efisien di antara varian *lazy bi-directional*, sementara BFMT (70,48 unit) menunjukkan performa paling buruk, dengan lintasan 15,6% lebih panjang dibandingkan FMT.

Anomali pada BFMT ini sangat menarik dan memberikan wawasan penting mengenai keterbatasan pendekatan *bi-directional* pada lingkungan sederhana. Temuan bahwa BFMT menghasilkan lintasan jauh lebih panjang dibandingkan FMT menunjukkan bahwa koordinasi dua pohon dalam kondisi tanpa rintangan kompleks justru *counterproductive*. *Overhead* koordinasi dan kemungkinan titik sambung yang suboptimal menghapus seluruh keuntungan teoretis dari pencarian dua arah. Temuan ini menjadi pelajaran penting bahwa perluasan algoritmik yang secara teoretis menguntungkan pada lingkungan kompleks dapat justru merugikan pada skenario sederhana.

d. Analisis Penggunaan Memori

Analisis berdasarkan *maximum memory consumption* menghasilkan temuan yang berbeda dibandingkan metrik-metrik lainnya, di mana variasi antar algoritma sangat kecil dan dapat diabaikan. Distribusi penggunaan memori menunjukkan bahwa PDST menggunakan memori paling rendah, yaitu 114,82 MB, sedangkan PRM menggunakan memori paling tinggi, yaitu 115,28 MB. Rentang totalnya hanya 0,46 MB atau sekitar 0,4% dari nilai rata-rata, dengan standard deviation hanya sekitar 0,12 MB, yang sangat kecil dalam konteks sistem komputasi modern.

Interpretasi dari hasil ini sangat jelas — metrik ini tidak memiliki *discriminatory power* yang berarti untuk membedakan performa antar algoritma. Seluruh 19 algoritma beroperasi dalam rentang memori yang sangat sempit, mengindikasikan bahwa pada *task pick-and-place* sederhana dengan robot 6-DOF, seluruh sampling-based planner memiliki jejak kaki memori yang hampir sama, terlepas dari pendekatan algoritmik yang digunakan. Penggunaan memori terutama dipengaruhi oleh kompleksitas model robot dan representasi lingkungan perencanaan, bukan oleh algoritma perencananya. Perbedaan kecil yang teramati—di mana PDST paling rendah dan PRM paling tinggi—kemungkinan disebabkan oleh perbedaan *data structure*, dengan PRM yang menyimpan struktur graf permanen sehingga membutuhkan sedikit lebih banyak memori dibandingkan struktur pohon yang dapat dipangkas.

Implikasi praktis dari temuan ini adalah bahwa penggunaan memori tidak seharusnya menjadi faktor utama dalam pemilihan algoritma untuk aplikasi ini. Bahkan sistem *embedded* dengan kapasitas RAM sedang (lebih dari 200 MB) dapat menjalankan seluruh algoritma yang diuji tanpa kendala memori. Perbedaan 0,46 MB antara algoritma paling efisien dan paling boros memori sangatlah kecil dalam konteks sistem robotika modern yang umumnya memiliki RAM dalam skala gigabita. Oleh karena itu, pemilihan algoritma sebaiknya difokuskan pada dimensi performa lain seperti kecepatan, konsistensi, dan efisiensi lintasan, karena konsumsi memori tidak memberikan diferensiasi yang bermakna antar algoritma.

e. Temuan Kunci

Penelitian ini mengungkapkan beberapa temuan kunci yang berkontribusi signifikan terhadap pemahaman performa algoritma *motion planning* dalam tugas manipulasi sederhana. Temuan pertama adalah kegagalan paradigma *bi-directional* pada lingkungan yang sederhana, yang merupakan pola sistematis yang diamati pada berbagai keluarga algoritma. Perbandingan antara BFMT dan FMT menunjukkan di mana BFMT memiliki waktu eksekusi 13% lebih lambat, konsistensi 578% lebih buruk, serta jalur yang 15,6% lebih panjang — lebih buruk pada hampir semua aspek. Perbandingan antara BKPIECE dan KPIECE menunjukkan hasil negatif dengan waktu eksekusi 10% lebih lambat, konsistensi 68% lebih buruk, dan jalur 1,7% lebih panjang. LBKPIECE tercatat sebagai algoritma paling lambat. Sementara itu, perbandingan antara RRTConnect dan RRT secara mengejutkan menunjukkan bahwa RRTConnect memiliki waktu eksekusi yang lebih lama, yaitu 17,86 detik dibandingkan 16,84 detik pada RRT, dengan efisiensi yang lebih rendah.

Penjelasan teoretis terhadap kinerja buruk yang sistematis dari pendekatan *bi-directional* adalah bahwa algoritma ini secara fundamental dirancang untuk lingkungan dengan *narrow passages* atau konfigurasi hambatan yang kompleks. Dalam skenario *pick-and-place* sederhana yang diuji dalam penelitian ini, beban koordinasi antara dua pohon tidak terkompensasi oleh konvergensi yang lebih cepat sebagaimana biasanya diharapkan. Titik koneksi antara dua pohon sering kali bersifat suboptimal, terutama terlihat pada kasus BFMT, di mana strategi koneksi menghasilkan jalur keseluruhan yang jauh lebih panjang. Selain itu, kondisi race dalam pertumbuhan pohon menciptakan variabilitas yang tinggi karena waktu ketika kedua pohon saling bertemu menjadi sangat stokastik.

Temuan kedua menunjukkan bahwa *asymptotic optimality* tidak dapat dibenarkan untuk tugas-tugas yang bersifat sederhana. Algoritma RRT* dan PRM* menunjukkan penurunan kinerja yang signifikan, dengan waktu eksekusi 177–169% lebih lambat hanya untuk peningkatan kualitas jalur yang minimal, yaitu 1,2–4,5% jalur yang lebih pendek. Perbandingan antara RRT* dan RRT menunjukkan hasil yang sangat mencolok — RRT* memerlukan waktu 177% lebih lama hanya untuk menghasilkan jalur 1,2% lebih pendek, yang merepresentasikan nilai efisiensi yang sangat rendah. PRM* dibandingkan dengan PRM memang sedikit lebih baik, dengan waktu eksekusi 169% lebih lambat untuk jalur 4,5% lebih pendek, namun tetap menunjukkan *trade-off* yang sangat tidak menguntungkan.

Interpretasi dari temuan ini adalah bahwa dalam lingkungan sederhana, non-*optimal planners* seperti RRT dan PDST secara empiris telah mampu menghasilkan jalur yang hampir optimal. Proses *rewiring* yang ekstensif pada RRT*, yang secara teoretis menjamin *asymptotic optimality*, pada kenyataannya merupakan perhitungan yang sebagian besar terbuang karena jalur awal yang dihasilkan sudah memiliki kualitas tinggi. Mekanisme seperti *temperature annealing* atau *cost-based* sampling pada varian algoritma yang lebih kompleks tidak memberikan manfaat yang cukup besar untuk membenarkan beban komputasi tambahan ketika lingkungan tidak memiliki permukaan biaya (*cost surface*) yang kompleks atau *narrow passages* yang memerlukan navigasi cermat.

Implikasi praktis dari hasil ini sangat jelas — algoritma dengan sifat asymptotically optimal tidak direkomendasikan untuk tugas pick-and-place sederhana, kecuali jika optimalitas jalur dapat dibuktikan sebagai persyaratan yang benar-benar kritis, yang jarang terjadi dalam aplikasi industri nyata. Untuk sebagian besar tugas

manipulasi seperti pada sistem otomasi gudang, lini perakitan, atau operasi pengemasan, jalur yang hampir optimal dari fast planners sudah sepenuhnya memadai dan jauh lebih disukai karena efisiensi waktu yang dihasilkan.

4.5 Validasi Temuan melalui Studi Komparatif

Penelitian studi menurut Yang et al. (2023) melakukan evaluasi komprehensif terhadap 12 algoritma *motion planning* pada tiga robot manipulator (Franka, UR5, dan Kuka) dalam lingkungan *berclutter* dengan tiga tingkat kompleksitas berbeda. Penelitian tersebut menggunakan MotionBenchmaker tool untuk mengevaluasi performa algoritma dalam skenario yang menantang, mulai dari lingkungan dengan rintangan tanpa batasan spasial hingga lingkungan tertutup seperti *drawer environment* (lingkungan seperti laci) yang sangat terbatas.

Perbandingan dengan studi Yang et al. memberikan konteks penting untuk memahami karakteristik performa algoritma dalam lingkungan sederhana versus kompleks. Studi tersebut menunjukkan degradasi performa yang signifikan ketika kompleksitas lingkungan meningkat dari skenario terbuka ke lingkungan tertutup. Temuan ini memvalidasi pentingnya fokus penelitian ini pada lingkungan sederhana tanpa *clutter*, mengingat karakteristik performa algoritma berubah secara fundamental berdasarkan tingkat kompleksitas lingkungan.

Dalam lingkungan rintangan tanpa batasan spasial (Skenario 1 Yang et al.), algoritma RRTConnect memiliki performa yang terbaik, sejalan dengan temuan penelitian ini yang menunjukkan Algoritma *asymptotically complete* seperti PDST dan RRT, mencapai performa optimal. Namun, ketika kompleksitas meningkat dengan batasan spasial (Skenario 2-3), terjadi degradasi performa yang tidak seragam antar algoritma, mendukung kesimpulan bahwa tidak ada algoritma universal terbaik untuk semua kondisi lingkungan.

Temuan paling signifikan dari analisis komparatif ini adalah mengenai algoritma *bi-directional*. Penelitian ini menemukan bahwa BFMT menunjukkan konsistensi 578% lebih buruk dibandingkan FMT dalam lingkungan sederhana. Namun, studi Yang et al. menunjukkan bahwa dalam lingkungan dengan rintangan, algoritma *bi-directional* seperti BKPIECE mempertahankan success rate yang tinggi menunjukkan performa yang baik meskipun dengan waktu *planning* lebih lama.

Perbedaan ini mengindikasikan bahwa performa buruk algoritma *bi-directional* dalam penelitian ini kemungkinan bersifat spesifik untuk lingkungan sederhana. Dalam konteks tugas *pick-and-place* sederhana tanpa *narrow passages* atau *clutter* yang signifikan, *overhead* komputasi dari strategi bi-directional tidak memberikan keuntungan dan justru mengurangi

efisiensi. Sebaliknya, dalam lingkungan kompleks dengan batasan spasial, pendekatan *bi-directional* menjadi menguntungkan untuk menemukan jalur yang dapat ditempuh yang menghubungkan konfigurasi awal dan tujuan dari kedua arah.

Kedua, studi juga menunjukkan pola *trade-off* yang konsisten antara *optimality* dan efisiensi komputasi. Algoritma *asymptotically optimal* seperti RRT* dan PRM* memberikan kualitas jalur lebih baik tetapi dengan biaya komputasi signifikan. Dalam studi Yang et al., RRT* hanya menemukan solusi *low-cost* dalam 1 dari 6 skenario manipulasi. Hal ini memperkuat temuan penelitian ini bahwa *overhead* komputasi untuk jaminan *optimality* atau pencarian jalur yang terbaik tidak dapat dibenarkan secara praktis. Penalti waktu eksekusi sebesar 180-192% yang ditemukan dalam penelitian ini untuk algoritma optimal menunjukkan bahwa *trade-off* ini menjadi semakin tidak menguntungkan dalam lingkungan sederhana di mana jalur *near-optimal* sudah dapat ditemukan dengan cepat oleh algoritma non-optimal

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini berhasil melakukan *benchmarking* evaluasi komprehensif terhadap 19 algoritma perencanaan gerak berbasis sampling untuk robot manipulator UR5 dalam tugas penyusunan balok warna menggunakan platform ROS Noetic dan simulator Gazebo dalam lingkungan sederhana tanpa rintangan. Penelitian ini mengisi *research gap* yang signifikan dalam literatur motion planning, mengingat sebagian besar studi yang ada seperti Yang et al., (2023) yang berfokus pada skenario *cluttered environment* dengan rintangan kompleks. Dengan menyediakan baseline benchmark pada simple environment, penelitian ini memberikan perspektif komplementer yang penting untuk pemahaman komprehensif tentang karakteristik performa algoritma *motion planning*.

Hasil penelitian mengungkap beberapa temuan, algoritma yang optimal pada lingkungan kompleks tidak selalu menunjukkan performa superior pada lingkungan sederhana, bahkan beberapa mengalami degradasi signifikan. Temuan ini membuktikan bahwa pemilihan algoritma *motion planning* harus sangat dependen dengan kebutuhan, bukan hanya bergantung pada jaminan teoretis atau performa pada satu jenis lingkungan tertentu. Temuan utama penelitian dapat dirangkum sebagai berikut:

a. Pengembangan Simulasi Robot Penyusun

   Penelitian ini berhasil dirancang dan dikembangkan sistem simulasi robot penyusun balok warna yang terintegrasi dengan 19 algoritma *motion planning* berbasis *sampling*. Sistem ini menggunakan arsitektur modular dengan integrasi sempurna antara Gazebo, MoveIt!, dan ROS control, memungkinkan evaluasi sistematis terhadap berbagai algoritma dalam lingkungan yang terkontrol.

b. *Benchmarking* Algoritma dan Variabilitas Performa

   Evaluasi menyeluruh terhadap 19 algoritma telah dilakukan dengan 10 iterasi per algoritma, menghasilkan total 190 dataset. Hasil *benchmarking* menunjukkan PDST menghasilkan performa terbaik dalam hal kecepatan dengan waktu rata-rata 16.42 detik per operasi, diikuti oleh TRRT (16.65 detik) dan RRT (16.84 detik). Selanjutnya, FMT unggul dalam efisiensi lintasan dengan *total joint distance* terpendek (60.96 unit) dan konsistensi tertinggi (standar deviasi 0.306 detik), memvalidasi jaminan teoretisnya sebagai algoritma *asymptotically optimal.*

c. Efektivitas Metrik Evaluasi

Analisis koefisien variasi mengungkapkan bahwa tidak semua metrik memiliki daya diskriminasi yang sama. Total Cartesian *distance* dan *Minimum Memory* menunjukkan variasi minimal (CV: 0.23%), mengindikasikan bahwa semua algoritma menghasilkan jalur dengan panjang geometris yang hampir identik. Sebaliknya, *total joint distance* menunjukkan variasi signifikan (15.6%), menjadikannya metrik yang lebih efektif untuk membedakan efisiensi algoritma. Temuan ini memberikan wawasan penting bahwa evaluasi algoritma *motion planning* harus mempertimbangkan metrik di *joint space*, bukan hanya ruang Cartesian.

d. Temuan Kritis tentang Paradigma *Bi-directional*

Penelitian ini mengungkap temuan penting bahwa algoritma *bi-directional* menunjukkan performa yang secara sistematis lebih buruk dibandingkan varian *single-direction*-nya pada lingkungan sederhana tanpa rintangan kompleks. BFMT menunjukkan degradasi yang paling drastis dengan waktu eksekusi 13% lebih lambat, konsistensi 578% lebih buruk, dan jalur 15.6% lebih panjang dibandingkan FMT. Pola serupa juga teramati pada varian lainnya, di mana BKPIECE dan LBKPIECE menunjukkan penurunan performa signifikan dibandingkan KPIECE. Temuan ini berkontradiksi langsung dengan hasil (Yang et al., 2023) yang menunjukkan superioritas algoritma *bi-directional* pada *cluttered environment*. Perbedaan ini mengkonfirmasi bahwa *overhead* koordinasi *Bi-directional* tidak dapat dibenarkan pada lingkungan sederhana namun berkerja dengan performa yang baik pada lingkungan yang penuh rintangan dan batasan spasial. Temuan ini memberikan wawasan penting bahwa perluasan algoritmik yang secara teoretis menguntungkan pada lingkungan kompleks dapat justru kontraproduktif pada skenario sederhana.

e. Evaluasi *Trade-off Asymptotic Optimality*

Algoritma dengan jaminan *asymptotic optimality* seperti RRT*, PRM*, dan LBTRRT menunjukkan *trade-off* yang sangat tidak menguntungkan untuk tugas sederhana. RRT* memerlukan waktu 177% lebih lama hanya untuk menghasilkan jalur 1.2% lebih pendek dibandingkan RRT, sementara PRM* membutuhkan waktu 169% lebih lama untuk menghasilkan jalur yang hanya 4.5% lebih pendek dibandingkan PRM. Pada Penelitian Yang et al., (2023) Algoritma *Asymptotic Optimality* memiliki performa lintasan yang bagus pada lingkungan kompleks dan penuh rintangan, namun dengan *trade-off* metrik waktu yang tidak optimal. `Hasil ini menunjukkan bahwa pada lingkungan sederhana, algoritma non-optimal seperti RRT dan PDST secara empiris telah mampu menghasilkan jalur yang hampir optimal, sehingga proses *rewiring* yang

ekstensif pada algoritma optimal menjadi perhitungan yang sebagian besar terbuang. Temuan ini menegaskan bahwa *overhead* komputasi untuk mencapai jaminan teoretis *optimality* tidak dapat dibenarkan secara praktis dalam konteks tugas *pick-and-place* sederhana.

Secara keseluruhan, penelitian ini memberikan kontribusi signifikan dalam memahami karakteristik performa algoritma *motion planning* berbasis sampling untuk aplikasi robotika industri, khususnya dalam konteks tugas manipulasi sederhana. Hasil penelitian menunjukkan bahwa pemilihan algoritma yang tepat dapat meningkatkan efisiensi operasional hingga 192%, menjadikan *benchmarking* sistematis sebagai langkah krusial dalam desain sistem robotika.

5.2 Saran

Harapannya, penelitian ini dapat menjadi kontribusi bagi komunitas ilmiah dan industri robotika, khususnya dalam pengembangan sistem robot manipulator untuk *aplikasi pick-and-place*. Dengan menyajikan evaluasi komprehensif terhadap 19 algoritma *motion planning* berbasis sampling, penelitian ini diharapkan menjadi landasan yang kokoh bagi studi lanjutan dalam bidang perencanaan gerak robot. Keberhasilan penelitian ini juga diharapkan menginspirasi peneliti lain untuk mengeksplorasi aspek-aspek baru dalam optimisasi performa algoritma *motion planning* untuk berbagai aplikasi robotika industri. Selain itu, diharapkan bahwa hasil penelitian ini dapat menjadi rujukan bagi praktisi dan pengembang sistem robotika dalam memilih algoritma *motion planning* yang paling sesuai dengan kebutuhan aplikasi spesifik mereka. Dengan memahami *trade-off* antara kecepatan, konsistensi, efisiensi lintasan, dan penggunaan memori, industri dapat meningkatkan produktivitas dan efisiensi operasional sistem robot mereka.

Harapan dari penelitian lanjutan, disarankan beberapa hal berikut:

a. Implementasi pada robot fisik untuk validasi performa algoritma dalam kondisi nyata dengan mempertimbangkan faktor dinamika, sensor *noise*, dan ketidakpastian lingkungan.
b. Perluasan kompleksitas lingkungan kerja dengan menambahkan rintangan statis dan dinamis, skenario *multi-robot*, serta evaluasi dalam lingkungan dengan celah sempit (*narrow passages*).
c. Integrasi dengan sistem visi komputer menggunakan RGB-D *camera* atau kamera stereo untuk deteksi objek yang lebih realistis, termasuk penanganan kondisi pencahayaan bervariasi dan oklusi.
d. Aplikasi pada tugas manipulasi yang lebih kompleks seperti operasi *assembly* dengan pose *constraints* presisi, manipulasi objek *deformable*, atau *dual-arm coordination*.

e. Studi kasus industri spesifik melalui kolaborasi dengan manufaktur untuk evaluasi performa dalam konteks produksi nyata dengan analisis *Return on Investment* (ROI).

Dengan menjadikan penelitian ini sebagai referensi yang andal, diharapkan peneliti, akademisi, dan praktisi di masa mendatang dapat mengembangkan sistem robotika yang lebih inovatif, efisien, dan berkontribusi pada kemajuan otomasi industri secara global. Penelitian ini juga diharapkan dapat mempercepat adopsi teknologi robotika dalam berbagai sektor industri, khususnya manufaktur, logistik, dan layanan.

DAFTAR PUSTAKA

Agha, R. A. A., Mahdi, Z. H., Sefer, M. N., & Hamarash, I. (2021). A ROS-Gazebo Interface for the Katana Robotic Arm Manipulation. *UKH Journal of Science and Engineering*, *5*(1), 26–37. https://doi.org/10.25079/ukhjse.v5n1y2021.pp26-37

Alam, M. M., Nishi, T., Liu, Z., & Fujiwara, T. (2023). A Novel Sampling-Based Optimal Motion Planning Algorithm for Energy-Efficient Robotic Pick and Place. *Energies*, *16*(19). https://doi.org/10.3390/en16196910

Constantinos Chamzas, Zachary Kingston, Khen Elimelech, Wil Thomason, Carlos Quintero-Peña, Rahul Shome, & Lydia E. Kavraki. (2022, October 23). *Evaluating Motion Planning Performance: Metrics, Tools, Datasets, and Experimental Design,*. IEEE/RSJ IROS 2022 Workshop.

Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., & Lozano-Pérez, T. (2025). Integrated Task and Motion Planning. *Robotics, and Autonomous Systems Annu. Rev. Control Robot. Auton. Syst*, *58*, 50. https://doi.org/10.1146/annurev-control-091420

Ghadge, K., More, S., Gaikwad, P., & Chillal, S. (2018). Robotic Arm for Pick and Place Application. *International Journal of Mechanical Engineering and Technology (IJMET*, *9*(1), 125–133. http://www.iaeme.com/IJMET/index.asp125http://www.iaeme.com/IJMET/issues.asp?JType=IJMET&VType=9&IType=1http://www.iaeme.com/IJMET/issues.asp?JType=IJMET&VType=9&IType=1

Gomes, N. M., Martins, F. N., Lima, J., & Wörtche, H. (2022). Reinforcement Learning for Collaborative Robots Pick-and-Place Applications: A Case Study †. *Automation*, *3*(1), 223–241. https://doi.org/10.3390/automation3010011

International Federation of Robotics. (2022, November 20). *World Robotic Reports 2020*.

Khalil, W. (2019). *Modeling and Control of Manipulators-Part I: Geometric and Kinematic Models*. https://inria.hal.science/cel-02129939v1

Liu, J., Zhang, Y., Bian, S., Song, R., Zhang, C. H., Wei, Y., & Hua, C. (2025). P-RT-BFMT: A Prediction-Based Real-Time Bidirectional Fast Marching Tree for Robot Motion Planning in Dynamic Environments. *IEEE Transactions on Industrial Electronics*. https://doi.org/10.1109/TIE.2025.3574520

Liu, S., & Liu, P. (2021). Benchmarking and optimization of robot motion planning with motion planning pipeline. *The International Journal of Advanced Manufacturing Technology*, *118*. https://doi.org/10.1007/s00170-021-07985-5/Published

McKenzie, R. M., Barraclough, T. W., & Stokes, A. A. (2017). Integrating soft robotics with the robot operating system: A hybrid pick and place arm. *Frontiers Robotics AI*, *4*(AUG). https://doi.org/10.3389/frobt.2017.00039

Naranjo-Campos, F. J., Victores, J. G., & Balaguer, C. (2024). *Method for Bottle Opening with a Dual-Arm Robot*. https://doi.org/10.3390/biomimetics

Orthey, A., Chamzas, C., & Kavraki, L. E. (2025). Sampling-Based Motion Planning: A Comparative Review. *Robotics, and Autonomous Systems Annu. Rev. Control Robot. Auton. Syst. 2024*, *12*, 46. https://doi.org/10.1146/annurev-control-061623

Sandakalum, T., & Ang, M. H. (2022). Motion Planning for Mobile Manipulators—A Systematic Review. In *Machines* (Vol. 10, Issue 2). MDPI. https://doi.org/10.3390/machines10020097

Silva, R., Matos, A., & Pinto, A. M. (2022). Multi-criteria metric to evaluate motion planners for underwater intervention. *Autonomous Robots*, *46*(8), 971–983. https://doi.org/10.1007/s10514-022-10060-x

Tarbouriech, S., & Suleiman, W. (2020). Bi-objective Motion Planning Approach for Safe Motions: Application to a Collaborative Robot. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *99*(1), 45–63. https://doi.org/10.1007/s10846-019-01110-1

Wei, K., & Ren, B. (2018). A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm. *Sensors (Switzerland)*, *18*(2). https://doi.org/10.3390/s18020571

Yang, S., Liu, P., & Pears, N. (2023). Benchmarking of Robot Arm Motion Planning in Cluttered Environments. *ICAC 2023 - 28th International Conference on Automation and Computing*. https://doi.org/10.1109/ICAC57885.2023.10275283

Zhang, L., Cai, K., Sun, Z., Bing, Z., Wang, C., Figueredo, L., Haddadin, S., & Knoll, A. (2025). Motion planning for robotics: A review for sampling-based planners. In *Biomimetic Intelligence and Robotics* (Vol. 5, Issue 1). Elsevier B.V. https://doi.org/10.1016/j.birob.2024.100207

LAMPIRAN

1. Hasil data pengujian Algoritma BFMT

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	22.284	133.707	16.816	32.578	15.2004	70.4395	113.6	115.1
2	24.797	148.781	18.476	31.722	15.1816	76.6913	113.42	114.92
3	22.003	132.016	17.334	30.138	15.1746	70.5832	113.33	114.71
4	18.984	113.905	16.753	23.951	15.2233	63.6416	113.39	115.02
5	23.105	138.631	18.505	28.774	15.2367	74.545	113.85	115.22
6	18.922	113.53	17.751	21.369	15.1678	63.8982	113.96	115.34
7	19.734	118.406	16.974	27.282	15.1533	66.8403	113.89	115.14
8	20.889	125.332	18.088	23.962	15.2087	67.9219	113.55	115.05
9	23.371	140.227	18.186	29.316	15.1943	73.1723	113.1	114.73
10	24.673	148.037	19.952	30.158	15.2349	77.0733	114.13	115.26

2. Hasil data pengujian Algoritma LBKPIECE

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	20.806	124.835	18.098	28.483	15.21	66.4804	113.62	115.62
2	19.112	114.673	17.709	20.069	15.1975	60.7307	113.66	114.53
3	25.592	153.553	17.92	38.219	15.247	72.0721	113.79	114.79
4	24.61	147.659	18.588	30.545	15.2476	76.2848	113.55	115.05
5	24.241	145.445	18.559	31.496	15.3674	72.8574	113.38	115.12
6	19.407	116.439	18.948	19.99	15.1665	60.8264	114.06	115.31
7	21.031	126.188	18.726	25.676	15.2227	64.3268	113.53	114.9
8	24.162	144.974	17.692	31.679	15.2169	69.32	113.96	115.34
9	20.935	125.611	19.203	27.192	15.2441	63.2605	113.53	115.15
10	22.524	135.146	18.355	28.321	15.2072	69.2358	114.07	114.95

3. Hasil data pengujian Algoritma BiEST

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	15.563	93.38	15.138	16.295	15.1976	60.7488	114.05	115.18
2	15.814	94.883	14.04	16.744	15.1933	60.8641	113.61	115.11
3	18.818	112.91	14.594	27.661	15.2042	67.9912	113.77	114.77
4	17.605	105.632	14.808	27.46	15.2194	66.5898	113.52	115.14
5	16.097	96.584	14.025	20.152	15.1902	61.0478	113.25	114.75
6	18.307	109.844	14.724	25.293	15.158	68.6569	113.18	114.81
7	20.296	121.775	14.029	28.484	15.2958	70.8994	114.04	114.79
8	15.97	95.82	14.87	17.323	15.1856	60.7315	113.85	115.1
9	15.743	94.456	14.049	16.444	15.1766	60.8411	114.15	115.02
10	15.601	93.606	14.555	16.094	15.1865	60.7015	113.73	115.23

4. Hasil data pengujian Algoritma BiTRRT

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	19.854	119.124	14.686	28.4	15.1573	71.072	113.52	115.02
2	15.451	92.706	14.266	16.193	15.1782	60.7012	113.54	114.79
3	17.742	106.454	15.191	24.192	15.2048	67.9262	113.71	114.71
4	19.765	118.588	14.844	30.02	15.2017	71.5858	113.31	114.81
5	15.593	93.559	14.868	16	15.2322	60.9002	114.16	114.79
6	16.912	101.473	15.045	20.59	15.208	62.9325	114.13	114.63
7	15.994	95.961	14.973	17.194	15.1808	60.8024	113.6	115.1
8	16.975	101.85	14.964	22.352	15.2386	63.2396	114.06	114.93
9	16.036	96.216	15.171	17.396	15.1944	60.7852	114.31	115.171
10	17.412	104.471	15.716	23.614	15.1692	63.0253	113.42	115.05

5. Hasil data pengujian Algoritma BKPIECE

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	18.682	112.09	15.915	26.155	15.1931	62.8965	114.16	115.16
2	20.119	120.717	16.946	32.903	15.2158	67.0021	113.7	114.7
3	20.573	123.439	16.781	26.609	15.2279	65.4541	113.19	114.81
4	20.16	120.961	17.403	28.487	15.1596	69.0942	113.41	114.91
5	18.334	110.005	16.761	22.493	15.3074	63.5714	113.52	114.89
6	17.621	105.728	16.421	19.058	15.2428	62.0269	113.55	115.18
7	17.567	105.403	14.948	21.446	15.5833	63.9345	113.4	115.27
8	16.851	101.105	15.889	18.518	15.1963	60.8881	113.54	115.29
9	20.187	121.119	16.417	26.713	15.1721	67.2074	113.48	115.11
10	19.54	117.238	16.404	28.983	15.1872	66.0662	113.6	114.6

6. Hasil data pengujian Algoritma EST.

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	18.891	113.346	15.249	25.403	15.1421	64.8244	113.54	115.04
2	16.136	96.819	14.719	17.015	15.246	61.0012	113.42	114.67
3	17.231	103.386	15.364	21.217	15.2312	64.5052	113.25	114.63
4	17.72	106.318	15.044	25.793	15.2354	68.1141	113.94	114.94
5	19.686	118.114	15.525	31.208	15.2387	65.4275	113.38	115
6	17.245	103.47	14.179	24.425	15.1755	65.7661	113.61	114.86
7	17.42	104.522	15.482	24.405	15.201	66.0221	113.92	115.05
8	16.019	96.115	14.901	17.156	15.2175	61.0173	113.63	114.63
9	18.659	111.952	15.906	29.385	15.1821	65.6058	113.93	114.68
10	15.761	94.567	14.28	16.666	15.2067	60.8874	114.38	115.13

7. Hasil data pengujian Algoritma FMT.

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	19.115	114.689	18.526	19.974	15.1577	60.6054	113.87	114.99
2	19.381	116.288	17.817	20.762	15.1777	60.684	113.81	115.06
3	19.059	114.353	17.857	20.725	15.1573	60.6446	113.15	115.02
4	19.715	118.29	18.805	20.685	15.1687	60.5749	113.36	115.11
5	19.182	115.094	17.715	20.121	15.16	60.6467	113.66	115.04
6	19.291	115.746	17.988	22.153	15.5537	60.6632	113.53	116.15
7	18.934	113.604	17.668	20.556	15.1688	60.6569	113.14	114.64
8	20.022	120.131	18.207	24.769	15.1405	63.1424	113.55	115.18
9	19.274	115.642	18.332	20.259	15.2184	60.8441	113.76	115.26
10	19.368	116.206	17.538	21.627	15.2722	61.1304	113.96	114.96

8. Hasil data pengujian Algoritma KPIECE

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	17.142	102.851	15.569	22.304	15.1941	63.1907	113.42	114.67
2	15.772	94.634	14.862	16.452	15.2074	60.8627	114.32	115.07
3	17.243	103.46	15.013	23.67	15.1442	66.9578	113.77	115.02
4	17.453	104.718	14.905	23.122	15.209	63.2595	113.96	115.21
5	17.146	102.877	15.776	21.316	15.1956	63.2759	113.98	114.98
6	18.871	113.227	15.751	26.074	15.2823	65.3781	113.82	114.7
7	16.715	100.291	15.638	21.25	15.1922	63.2281	113.8	115.05
8	17.736	106.418	14.514	27.73	15.1939	64.9777	113.5	114.88
9	16.94	101.638	15.643	20.931	15.1503	63.2987	113.79	114.42
10	16.853	101.119	14.812	22.791	15.1876	62.8242	113.63	114.88

9. Hasil data pengujian Algoritma LBTRRT

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	52.123	312.736	45.439	57.498	15.1871	79.0887	113.47	114.97
2	47.944	287.666	44.817	58.137	15.1728	64.9993	113.19	115.06
3	45.924	275.542	44.453	46.888	15.1946	61.0843	113.14	114.77
4	46.547	279.282	44.656	51.533	15.1898	63.3851	113.95	115.2
5	45.39	272.34	44.121	46.189	15.1589	60.9139	114.02	114.52
6	47.048	282.285	45.084	52.706	15.1541	63.5778	113.33	115.08
7	46.54	279.238	44.707	51.681	15.172	63.4114	113.6	115.6
8	51.189	307.133	45.562	56.272	15.1832	78.1333	113.54	115.04
9	48.78	292.68	45.222	55.671	15.2026	67.9978	113.61	115.11
10	48.279	289.676	44.599	54.762	15.1665	68.3063	113.26	115.39

10. Hasil data pengujian Algoritma PDST

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	15.882	95.29	14.863	16.323	15.1782	60.7296	114.3	115.05
2	15.618	93.708	14.561	16.618	15.1595	60.6784	113.51	114.51
3	16.631	99.784	14.534	22.334	15.1677	63.0487	113.88	115.13
4	16.271	97.624	14.533	18.682	15.1774	61.9372	113.93	114.43
5	19.18	115.081	15.305	29.581	15.2343	68.1341	113.74	114.86
6	15.623	93.741	14.787	16.241	15.1823	60.8285	113.5	114.75
7	17.393	104.357	15.315	22.897	15.15	64.3296	114.03	115.15
8	16.142	96.851	14.639	17.235	15.197	60.8077	113.58	114.58
9	15.846	95.076	14.263	16.398	15.2224	60.76	113.82	115.07
10	15.608	93.646	13.916	17.875	15.2062	61.9033	113.68	114.68

11. Hasil data pengujian Algoritma PRM

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	15.598	93.585	14.738	16.515	15.2027	60.7492	113.58	115.08
2	15.705	94.231	14.506	16.779	15.147	60.7134	113.93	115.05
3	17.685	106.108	15.099	28.407	15.2011	66.7404	113.41	114.79
4	15.843	95.059	14.871	17.216	15.1872	60.7946	114.871	117.216
5	16.551	99.304	14.711	20.518	15.1655	63.1847	114.04	115.29
6	17.351	104.105	14.372	27.406	15.1827	64.9148	114.1	115.23
7	20.984	125.907	15.493	28.074	15.1965	72.455	114.25	115.13
8	15.709	94.253	14.648	16.91	15.1824	60.7572	114.02	114.89
9	19.541	117.246	14.48	28.015	15.1729	69.6596	113.83	114.96
10	15.521	93.129	14.932	16.29	15.1656	60.7936	113.8	115.17

12. Hasil data pengujian Algoritma PRMStar

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	46.078	276.468	44.707	49.361	15.1577	60.6846	113.82	114.7
2	45.581	273.487	44.837	46.476	15.1898	60.651	114.13	115.26
3	45.749	274.492	44.35	46.633	15.1596	60.7053	113.57	115.07
4	46.445	278.672	44.377	49.847	15.142	62.674	113.35	115.1
5	45.802	274.812	44.56	47.934	15.1946	61.1724	114.38	115.25
6	45.524	273.143	44.156	46.457	15.145	60.7006	113.79	115.29
7	45.586	273.514	44.487	46.52	15.2016	60.624	114.17	115.05
8	46.738	280.425	45.136	51.805	15.1765	63.1829	113.79	115.04
9	45.86	275.161	44.512	47.195	15.1695	60.6146	113.47	114.85
10	45.809	274.853	44.369	47.141	15.1744	60.6678	113.7	115.32

13. Hasil data pengujian Algoritma PROJEST

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	15.969	95.813	14.66	17.163	15.212	61.0081	113.13	115.01
2	16.378	98.267	14.219	18.793	15.1599	61.8466	114.22	115.34
3	16.5	99	14.761	19.151	15.1822	62.1561	113.89	115.64
4	16.435	98.609	15.304	19.44	15.1222	60.7258	114.14	115.01
5	16.706	100.239	15.203	19.77	15.1764	60.9236	113.19	115.06
6	20.858	125.145	15.69	27.688	15.1879	73.69	113.79	114.79
7	19.455	116.729	14.915	28.225	15.1797	69.8023	113.43	115.05
8	17.39	104.34	15.086	21.999	15.1998	62.9695	114.32	115.19
9	19.337	116.019	14.998	36.595	15.1808	65.503	114.07	115.2
10	15.875	95.248	14.806	17.708	15.1955	61.9576	113.71	115.09

14. Hasil data pengujian Algoritma RRT

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	15.585	93.51	14.587	17.168	15.225	60.8874	113.62	115
2	15.81	94.86	14.748	16.755	15.2128	60.9277	114.37	115
3	16.628	99.767	14.8	20.905	15.1376	60.6969	113.54	115.17
4	16.946	101.677	15.393	22.177	15.2011	63.2641	114.24	114.99
5	17.639	105.835	14.543	26.891	15.1692	61.2044	113.91	114.91
6	17.023	102.137	15.121	21.381	15.1717	63.1501	113.21	114.59
7	17.86	107.158	14.595	27.985	15.1419	66.4769	113.41	114.41
8	16.652	99.914	15.651	19.591	15.1932	60.731	113.66	114.91
9	16.757	100.54	15.084	21.092	15.1762	63.196	113.87	114.74
10	17.49	104.939	15.483	23.117	15.1963	63.9923	113.57	114.95

15. Hasil data pengujian Algoritma RRTConnect

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	15.585	93.51	14.587	17.168	15.225	60.8874	113.62	115
2	15.81	94.86	14.748	16.755	15.2128	60.9277	114.37	115
3	16.628	99.767	14.8	20.905	15.1376	60.6969	113.54	115.17
4	16.946	101.677	15.393	22.177	15.2011	63.2641	114.24	114.99
5	17.639	105.835	14.543	26.891	15.1692	61.2044	113.91	114.91
6	17.023	102.137	15.121	21.381	15.1717	63.1501	113.21	114.59
7	17.86	107.158	14.595	27.985	15.1419	66.4769	113.41	114.41
8	16.652	99.914	15.651	19.591	15.1932	60.731	113.66	114.91
9	16.757	100.54	15.084	21.092	15.1762	63.196	113.87	114.74
10	17.49	104.939	15.483	23.117	15.1963	63.9923	113.57	114.95

16. Hasil data pengujian Algoritma RRTStar

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	46.262	277.573	44.463	51.416	15.1542	62.6686	113.82	114.82
2	46.417	278.504	44.798	50.475	15.1595	63.1825	113.75	115.25
3	45.767	274.605	44.452	47.656	15.2073	60.7612	113.52	115.14
4	45.849	275.095	44.798	46.885	15.1285	60.6701	113.65	114.65
5	47.635	285.812	45.067	56.842	15.2322	65.5444	113.14	115.02
6	46.673	280.036	45.258	51.309	15.1584	63.1167	113.84	115.21
7	47.448	284.687	44.956	55.192	15.1288	66.5878	113.22	114.59
8	46.67	280.019	44.869	50.679	15.1505	63.1332	113.88	114.88
9	47.879	287.272	44.228	58.246	15.1686	65.471	113.93	114.68
10	45.642	273.852	44.572	46.501	15.1724	60.7097	114.03	114.9

17. Hasil data pengujian Algoritma SBL

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	16.442	98.652	15.278	17.808	15.3002	61.2295	113.82	114.95
2	16.639	99.833	15.42	18.148	15.1564	60.8018	113.5	115
3	16.898	101.385	16.176	18.131	15.2315	61.0137	114	114.88
4	18.168	109.01	15.784	26.124	15.199	66.2615	113.8	115.05
5	17.112	102.674	14.407	20.578	15.2038	60.8418	113.68	114.81
6	17.771	106.625	15.349	23.66	15.1928	64.2841	114.62	115.89
7	18.095	108.567	14.683	28.8	15.2918	65.1939	113.72	115.34
8	17.543	105.257	15.064	24.592	15.2257	65.9743	113.48	114.98
9	16.576	99.454	16.204	17.132	15.2388	60.9582	113.95	115.07
10	16.582	99.49	14.963	18.5	15.1938	62.1767	113.44	115.06

18. Hasil data pengujian Algoritma Stride

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	18.344	110.064	14.831	24.575	15.19	65.7475	114.23	114.98
2	17.109	102.652	14.784	23.862	15.1461	62.9056	113.61	115.11
3	17.772	106.631	14.199	28.572	15.1468	65.0398	113.42	114.92
4	16.058	96.348	14.651	18.634	15.1567	61.9268	114.1	115.1
5	16.863	101.177	14.834	19.279	15.356	62.2991	113.55	114.92
6	19.166	114.997	14.933	30.918	15.3095	63.549	113.79	114.92
7	15.677	94.062	14.47	16.821	15.1988	60.7756	113.68	114.93
8	18.357	110.142	16.366	25.515	15.2384	67.1	113.67	114.92
9	17.592	105.552	14.948	22.787	15.1793	63.1357	113.96	114.84
10	18.046	108.278	15.124	23.822	15.1608	63.4616	113.28	115.03

19. Hasil data pengujian Algoritma TRRT

name	avg op tim	total time	fastest	slowest	total carte	total joint	min memo	max memory
1	15.589	93.535	14.684	16.23	15.1386	60.7501	113.29	114.79
2	15.71	94.258	14.797	16.353	15.2147	60.8809	113.8	115.05
3	14.13	98.908	5.824	16.681	16.361	64.0862	113.71	115.21
4	18.531	111.186	15.214	26.514	15.1762	69.0563	113.43	114.55
5	17.172	103.034	13.947	26.192	15.1769	60.667	113.85	114.85
6	19.24	115.437	14.059	38.515	15.2241	64.8054	113.94	114.94
7	17.734	106.404	14.975	24.855	15.1642	67.9942	114.18	115.05
8	17.343	104.056	15.546	23.982	15.2213	60.7586	114.03	114.9
9	15.612	93.672	14.91	16.465	15.1862	60.666	113.24	114.74
10	15.422	92.532	14.067	16.348	15.1706	60.7129	113.52	114.9