Aplikasi Algoritma RRT-Star untuk Simulasi Perencanaan Jalur Robot di Lingkungan Dalam Ruangan

Karol Yangqian Poetracahya - 13523093[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523093@std.stei.itb.ac.id [1]karolyangqian14@gmail.com

***Abstrak*— Perencanaan jalur dalam robotika merupakan proses krusial yang memungkinkan robot bertransisi dari satu keadaan ke keadaan lainnya secara efisien, baik untuk manipulasi objek maupun pergerakan (navigasi). Rapidly-Exploring Random Tree Star (RRT*) adalah salah satu algoritma berbasis *sampling* yang dirancang untuk perencanaan jalur pada ruang pencarian kontinu. Algoritma ini memiliki keunggulan dalam menangani ruang keadaan berdimensi tinggi, sehingga menjadi solusi yang andal untuk perencanaan jalur robot dalam lingkungan statis. Makalah ini mendemonstrasikan dan menganalisis performa algoritma RRT* dalam perencanaan jalur melalui simulasi robot menggunakan ROS2 dan Gazebo.**

***Kata Kunci*—Perencanaan Jalur, Robotika, RRT*, Tree.**

I. PENDAHULUAN

Dalam robotika, perencanaan jalur atau lintasan merupakan bagian dari perencanaan pergerakan yang berfokus pada pencarian solusi berbasis geometri untuk menemukan jalur yang bebas tumbukan, tanpa memperhitungkan dinamika atau durasi pergerakan [1]. Proses ini memungkinkan robot untuk bernavigasi pada ruang 2D atau 3D. Perencanaan jalur memainkan peran penting dalam berbagai aplikasi, mulai dari pengoptimalan waktu pengiriman barang dalam logistik manufaktur hingga menggerakkan mekanisme manipulasi objek dengan derajat kebebasan (*degree of freedom* atau DOF) yang tinggi.

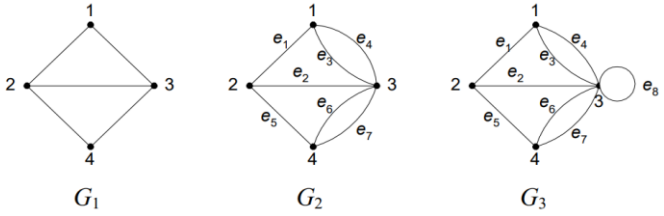
Untuk menangani situasi yang kompleks, diperlukan algoritma yang andal guna memberikan solusi jalur yang optimal dan efisien. Salah satu kelompok algoritma perencanaan jalur yang populer adalah algoritma berbasis graf, seperti Dijkstra dan A*. Namun, algoritma berbasis graf hanya efektif dalam lingkungan berbasis kisi-kisi (*grid*). Sementara itu, banyak kasus dalam robotika membutuhkan solusi jalur di ruang pencarian kontinu, seperti navigasi robot dalam ruangan, *unmanned aerial vehicles* (UAV) yang harus menghindari rintangan di udara, atau pergerakan lengan robot dengan 3 DOF. Untuk menjawab kebutuhan tersebut, dikembangkan algoritma berbasis *sampling*, seperti Probabilistic RoadMap Star (PRM*) dan RRT*.

Algoritma berbasis *sampling* memiliki keunggulan dalam merencanakan jalur secara andal untuk kasus dengan ruang keadaan berdimensi tinggi. Selain itu, algoritma ini lebih efisien waktu dalam lingkup pencarian yang luas karena jarak antar simpul keadaan tidak terbatas pada *grid* atau kisi-kisi, seperti halnya pada algoritma berbasis graf. Hal ini menjadikan algoritma berbasis *sampling* unggul dalam berbagai aplikasi dan sangat umum digunakan dalam perencanaan jalur di dunia industri.

Makalah ini secara khusus berfokus pada salah satu algoritma berbasis *sampling* yang juga berbasis pohon, yaitu RRT*. Algoritma ini pertama kali diusulkan pada tahun 2011 oleh Karaman dan Frazzoli dalam penelitian berjudul *Sampling-based Algorithms for Optimal Motion Planning*. RRT* adalah pengembangan dari algoritma Rapidly-Exploring Random Tree (RRT) yang belum mampu memberikan solusi jalur optimal. Dalam makalah ini, performa algoritma RRT* dianalisis dan diimplementasikan melalui simulasi robot menggunakan ROS2 dan Gazebo. Sisa dari struktur makalah ini meliputi Bab II Landasan Teori yang membahas konsep-konsep yang mendasari penelitian ini, Bab III Metodologi yang merincikan langkah-langkah kerja penelitian dan konfigurasi simulasi, Bab IV Hasil dan Analisis yang memaparkan hasil implementasi simulasi serta analisis performa algoritma, dan terakhir Bab V Kesimpulan dan Saran yang merangkum temuan penelitian serta memberikan rekomendasi untuk pengembangan di masa mendatang.

II. LANDASAN TEORI

A. Graf

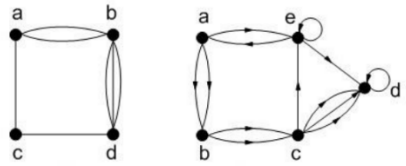


(a) Graf sederhana (b) Graf tak-sederhana ganda (c) Graf tak-sederhana semu

Gambar 1. Jenis-jenis graf berdasarkan ada tidaknya sisi ganda atau gelang: (a) graf sederhana, (b) graf tak-sederhana ganda, dan (c) graf tak-sederhana semu
Sumber: informatika.stei.itb.ac.id/~rinaldi.munir/

Graf didefinisikan sebagai $G = (V, E)$ dengan V adalah himpunan tidak kosong dari simpul-simpul dan E adalah

himpunan sisi yang menghubungkan sepasang simpul [10]. Berdasarkan ada atau tidaknya sisi ganda atau gelang, graf digolongkan menjadi graf sederhana dan tak-sederhana seperti pada. Graf sederhana tidak memiliki gelang maupun sisi ganda. Graf tak-sederhana dapat dibedakan lagi menjadi dua jenis, yakni graf ganda yang mengandung sisi ganda serta graf semu (*pseudo-graph*) yang mengandung sisi gelang. Graf juga dapat digolongkan berdasarkan orientasi arah pada sisi, yakni graf tak-berarah (*undirected graph*) yang sisi-sisinya tidak memiliki orientasi arah serta graf berarah (*directed graph*) yang sisi-sisinya memiliki arah.



(a) Graf tak-berarah (b) Graf berarah

Gambar 2. Jenis-jenis graf berdasarkan orientasi arah sisi
Sumber: informatika.stei.itb.ac.id/~rinaldi.munir/

Berikut adalah beberapa terminologi yang mendeskripsikan sebuah graf:

1. **Lintasan (*path*)**
   Lintasan dengan panjang n adalah barisan selang-seling simpul-simpul dan sisi-sisi yang menghubungkan simpul awal v_0 dan simpul tujuan v_n. Lintasan tersebut ditulis sebagai $v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n$ dengan $e_0 = (v_0, v_1), \ldots, e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf. Pada graf sederhana, lintasan dapat dinyatakan tanpa menuliskan sisi-sisinya.
2. **Siklus (*cycle*) atau sirkuit (*cicuit*)**
   Siklus atau sirkuit adalah lintasan yang berawal dan berakhir di simpul yang sama.
3. **Keterhubungan (*connected*)**
   Sepasang simpul disebut terhubung apabila terdapat lintasan yang menghubungkan keduanya. Sebuah graf adalah terhubung (*connected* graph) jika setiap pasang simpul di dalamnya terhubung. Jika syarat tidak dipenuhi, maka graf tersebut tak terhubung (*disconnected graph*).
4. **Upagraf (*subgraph*)**
   Graf $G_1 = (V_1, E_1)$ adalah upagraf dari $G = (V, E)$ apabila $V_1 \subseteq V$ dan $E_1 \subseteq E$.
5. **Graf berbobot (*weighted graph*)**
   Jika setiap sisi pada sebuah graf diberikan harga, maka graf tersebut menjadi graf berbobot.

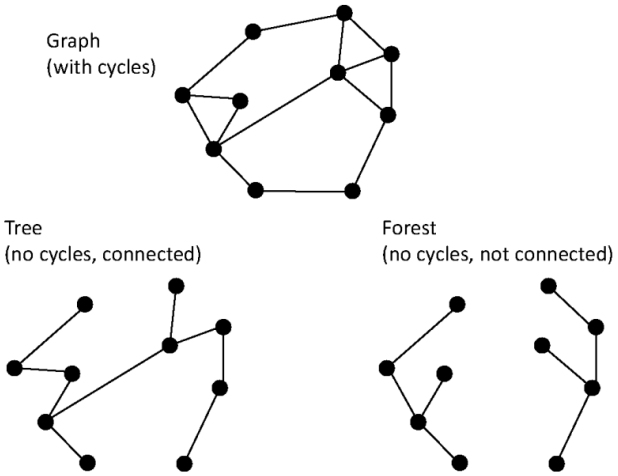
B. Pohon

Pohon didefinisikan sebagai sebuah graf yang tak-berarah, terhubung, dan tidak mengandung sirkuit atau siklus [11]. Misalkan $G = (V, E)$ merupakan graf tak-berarah sederhana dan memiliki jumlah simpul n. Dengan demikian, semua pernyataan di bawah ini ekuivalen:

1. G adalah pohon.
2. Setiap pasang simpul dalam G hanya terhubung oleh sebuah lintasan.
3. G terhubung dan memiliki $n - 1$ sisi.
4. G tidak mengandung sirkuit dan memiliki $n - 1$ sisi.
5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf hanya akan membentuk sebuah sirkuit.
6. G terhubung dan semua sisinya merupakan jembatan.

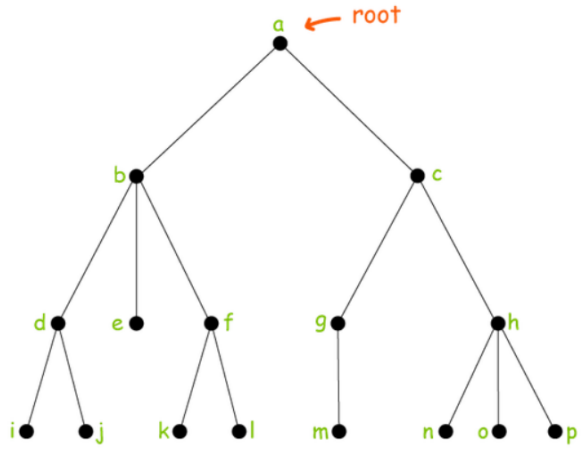
[11]

Pernyataan di atas merupakan teorema yang juga dapat dikatakan sebagai definisi lain sebuah pohon. Kumpulan pohon yang saling lepas disebut sebagai hutan.



Gambar 3. Perbandingan antara graf, pohon, dan hutan
Sumber: informatika.stei.itb.ac.id/~rinaldi.munir/

Pohon yang salah satu simpulnya diperlakukan sebagai akar (*root*) dan sisi-sisinya diberikan arah sehingga membentuk sebuah graf berarah disebut sebagai pohon berakar (*rooted tree*). Umumnya, arah pada sisi-sisi pohon berakar tidak perlu divisualisasikan secara eksplisit sebab arahnya sudah terdefinisi dengan jelas, yakni selalu berasal dari orangtua menuju anak.



Gambar 4. Pohon berakar
Sumber: informatika.stei.itb.ac.id/~rinaldi.munir/

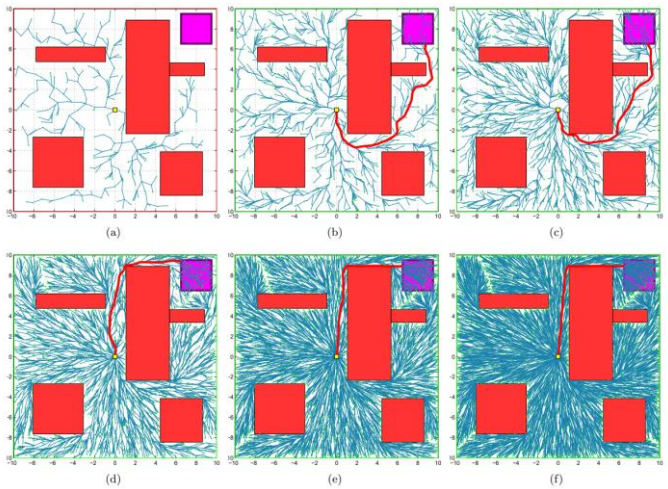
Beberapa terminologi berikut digunakan untuk mendeskripsikan sebuah pohon berakar:

1. **Anak (*child*) dan orangtua (*parent*)**
   Anak adalah simpul yang ditunjuk oleh simpul orangtuanya, menjauh dari akar. Pada Gambar XX., simpul g dan h adalah anak dari c. Sebaliknya, c adalah orangtua dari g dan h.

2. **Lintasan (*path*)**
   Lintasan adalah serangkaian simpul dan sisi yang menghubungkan sepasang simpul. Namun, karena pohon tergolong sebagai graf sederhana, sisi-sisi yang membentuk lintasan tidak perlu dituliskan. Pada Gambar XX., lintasan dari a ke j adalah a, b, d, j dengan panjang lintasan 3.
3. **Saudara kandung (*sibling*)**
   Sebuah simpul dikatakan sebagai saudara kandung dari simpul lain apabila kedua simpul tersebut memiliki orangtua yang sama.
4. **Upapohon (*subtree*)**
   Upapohon adalah pohon berakar yang merupakan upagraf dari pohon berakar utamanya.
5. **Derajat (*degree*)**
   Derajat dari sebuah simpul adalah jumlah anak dari simpul tersebut.
6. **Daun (*leaf*)**
   Daun adalah simpul yang tidak memiliki anak atau memiliki derajat nol.
7. **Simpul dalam (*internal nodes*)**
   Simpul dalam adalah simpul yang memiliki anak (simpul yang bukan daun).
8. **Aras (*level*) atau tingkat**
   Aras sebuah simpul adalah panjang lintasan yang menghubungkannya dengan simpul akar.
9. **Tinggi (*height*) atau kedalaman (*depth*)**
   Tinggi atau kedalaman dari sebuah pohon berakar didefinisikan sebagai aras maksimum dari pohon tersebut.

C. Algoritma Rapidly-Exploring Random Tree Star

Algoritma RRT* adalah salah satu algoritma berbasis *sampling* yang digunakan untuk perencanaan jalur pada ruang pencarian kontinu. RRT* dikembangkan sebagai pembaharuan dari algoritma RRT dengan kemampuan untuk memberikan solusi jalur yang optimal secara asimtotik, yaitu jalur yang mendekati optimal seiring bertambahnya jumlah sampel. Algoritma ini memiliki kompleksitas waktu pemrosesan $O(n \log n)$ dan waktu query $O(n)$ [5].



Gambar 5. Visualisasi algoritma RRT*
Sumber: arxiv.org/pdf/1105.1186

RRT* membangun struktur data berbasis pohon berakar yang eksploratif. Simpul-simpul dalam pohon mewakili keadaan robot seperti koordinat global (x, y, z) dan sisi-sisinya merepresentasikan transisi dari suatu keadaan ke keadaan lainnya. Akar pohon mewakili keadaan awal dari robot dan keadaan tujuan dalam perencanaan jalur adalah sebuah simpul dengan koordinat yang memenuhi syarat keadaan tujuan, biasanya berupa radius lingkaran dalam 2D atau bola dalam 3D. Objektif dari algoritma ini adalah membangun sebuah pohon yang mengandung lintasan optimal (memiliki total biaya seminimal mungkin) yang menghubungkan keadaan awal dan keadaan tujuan. Algoritma ini menggunakan pendekatan *sampling* acak untuk mengeksplorasi ruang pencarian yang berdimensi tinggi, dengan tetap mempertimbangkan kendala lingkungan seperti rintangan dan batasan ruang. Proses *sampling* ini memperluas ukuran pohon secara iteratif sehingga pada iterasi yang mendekati tak-hingga, pohon akan mengisi seluruh ruang pencarian.

Tahap yang membedakan RRT* dengan RRT adalah proses *rewiring* atau penyusunan ulang koneksi simpul. Penyusunan ulang koneksi ini memperbaiki struktur pohon dengan cara menghubungkan simpul baru ke simpul terdekat dalam radius tertentu di sekitarnya dengan jalur yang menghasilkan biaya minimum. Proses ini menjadikan RRT* optimal secara asimtotik. Berikut adalah tahapan-tahapan umum dari algoritma RRT*:

1. ***Sampling***: mengambil sampel acak dari ruang pencarian
2. ***Nearest Neighbor*:** menentukan simpul terdekat di dalam pohon terhadap sampel yang diambil
3. ***Steering*:** memperluas pohon dari simpul terdekat menuju sampel, menghasilkan simpul baru jika memungkinkan
4. ***Collision Checking***: memastikan bahwa cabang yang dihasilkan bebas dari tumbukan
5. ***Rewiring*:** menghubungkan simpul baru ke simpul terdekat dalam radius tertentu di sekitarnya dengan jalur yang menghasilkan biaya minimum

Algorithm 6: RRT*

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
  if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
    $V \leftarrow V \cup \{x_{\text{new}}\}$;
    $x_{\min} \leftarrow x_{\text{nearest}}$; $c_{\min} \leftarrow \texttt{Cost}(x_{\text{nearest}}) + c(\texttt{Line}(x_{\text{nearest}}, x_{\text{new}}))$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Connect along a minimum-cost path
      if $\texttt{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}})) < c_{\min}$ **then**
        $x_{\min} \leftarrow x_{\text{near}}$; $c_{\min} \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$
    $E \leftarrow E \cup \{(x_{\min}, x_{\text{new}})\}$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Rewire the tree
      if $\texttt{CollisionFree}(x_{\text{new}}, x_{\text{near}}) \wedge \texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}})) < \texttt{Cost}(x_{\text{near}})$ **then** $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
      $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$
return $G = (V, E)$;

Gambar 6. *Pseudocode* algoritma RRT*
Sumber: arxiv.org/pdf/1105.1186

III. METODOLOGI

A. Inisialisasi Lingkungan dan Plugin

Penelitian ini menggunakan implementasi algoritma RRT*

yang telah dipublikasikan di Github (tautan repositori terlampir). Program dijalankan dalam sistem operasi Linux Ubuntu 22.04 dengan ROS2 Humble. Simulasi dilakukan menggunakan model robot “burger” miliki Turtlebot3 yang tersedia secara sumber terbuka. Turtlebot3 memanfaatkan pustaka Navigation2 untuk fitur navigasinya. Secara bawaan, Navigation2 menggunakan algoritma Behavior Tree (BT) untuk perencanaan jalur. Agar algoritma RRT* dapat digunakan, parameter dalam file “burger.yaml” di pustaka turtlebot3_navigation diganti sesuai Gambar 7. agar file implementasi RRT* dikenali sebagai sebuah plugin oleh Navigation2 dan dapat dijalankan. Setelah workspace ROS2 yang berisi package “turtlebot3_simulation” dan “TurtleBot-RRT-Star” di-build, simulasi sudah bisa dijalankan.

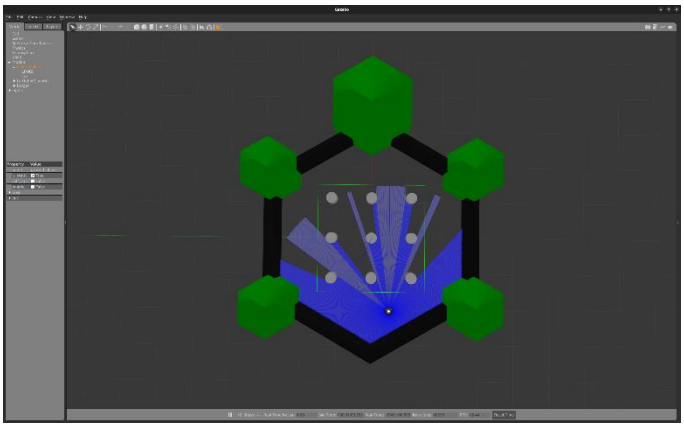
```
# GridBased:
#   plugin: 'nav2_navfn_planner/NavfnPlanner'
#   tolerance: 0.5
#   use_astar: false
#   allow_unknown: true

planner_plugin_types: ['nav2_rrtstar_planner::RRTStar'] # For Foxy and earlier
planner_plugin_ids: ['GridBased'] # For Foxy and earlier
plugins: ['GridBased'] # For Galactic and later
use_sim_time: True
GridBased:
  plugin: nav2_rrtstar_planner/RRTStar # For Galactic and later
  interpolation_resolution: 0.01
```

Gambar 7. Perubahan parameter dalam file “burger.yaml”

B. Simulasi Gazebo

Simulasi dilakukan dalam dunia Gazebo yang telah dibuat menggunakan model Turtlebot3. Dunia ini memiliki tembok pembatas berbentuk heksagon, dengan sembilan rintangan berbentuk silinder yang tersusun secara teratur di dalamnya. Saat inisialisasi, robot ditempatkan pada koordinat origin relatif terhadap world frame, yang terletak di sekitar tenggara dari rintangan. Orientasi robot menghadap ke utara, seperti yang digambarkan pada Gambar 8. Model robot yang digunakan adalah tipe "burger", yang merupakan robot dengan konfigurasi *differential drive* beroda dua. Robot ini dilengkapi dengan sensor lidar 360 derajat yang dapat memindai lingkungan sekitarnya untuk mendeteksi objek dan rintangan di sekitar area simulasi.

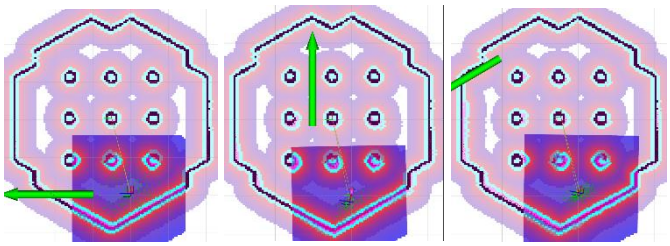


Gambar 8. Konfigurasi awal dunia simulasi

Peta dunia simulasi harus dipindai dan dibangkitkan terlebih dahulu agar robot dapat melakukan navigasi. Proses ini dilakukan menggunakan fungsi Simultaneous Localization and Mapping (SLAM) yang tersedia pada Turtlebot3. Saat fungsi ini dijalankan, robot digerakkan secara manual menggunakan tombol w, a, s, dan d melalui fitur teleop untuk mengelilingi dunia simulasi serta melewati rintangan yang ada. Hal ini memungkinkan SLAM untuk melakukan pemetaan lingkungan secara menyeluruh. Setelah data peta selesai dikumpulkan, peta diekspor ke dalam format file dengan ekstensi .pgm untuk gambar peta dan .yaml untuk parameter pendukung peta.

Setelah peta berhasil dibuat, proses navigasi dan perencanaan jalur dapat dimulai. Peta yang telah dihasilkan dimuat dan divisualisasikan menggunakan aplikasi RVIZ. Agar robot dapat menavigasi peta, ia perlu mengetahui posisinya relatif terhadap peta tersebut. Untuk itu, fungsi lokalisasi diaktifkan menggunakan algoritma Adaptive Monte Carlo Localization (AMCL) dengan memberikan estimasi titik posisi awal robot pada peta secara manual. Setelah lokalisasi berhasil dilakukan, perencanaan jalur dapat dilakukan dengan menetapkan titik tujuan pada peta. Algoritma RRT* kemudian akan melakukan perhitungan jalur untuk mencapai titik tersebut.

C. Pengambilan Data



(a) Tujuan I (b) Tujuan II (c) Tujuan III

Gambar 9. Tiga titik tujuan yang digunakan untuk pengujian perencanaan jalur

Jumlah maksimum iterasi pengambilan sampel simpul pada algoritma RRT* divariasikan menjadi 1000, 2000, dan 5000 iterasi. Variasi ini dilakukan untuk mengevaluasi performa algoritma dalam menghadapi beban komputasi yang berbeda. Pada setiap variasi iterasi, pengujian dilakukan terhadap tiga titik tujuan yang ditunjukkan pada Gambar 9. Variasi titik tujuan ini dirancang untuk menguji kemampuan algoritma dalam menentukan jalur menuju target dengan tingkat kesulitan yang berbeda, di mana tujuan I, II, dan III memiliki jarak yang semakin jauh secara berurutan. Tujuan I hanya berada dekat di sebelah titik awal, tujuan II berada di antara rintangan, dan tujuan III berada di ujung yang berlawanan dari titik awal.

Pada setiap percobaan, waktu pemrosesan perencanaan jalur dicatat untuk mengevaluasi efisiensi algoritma. Waktu yang dievaluasi hanya untuk proses perencanaan jalur saja, tidak mencakup proses lain seperti pergerakan robot menuju titik tujuan. Selain itu, keberhasilan algoritma dalam menemukan jalur yang valid untuk mencapai tujuan juga diuji. Data hasil pengujian ini diharapkan memberikan wawasan mengenai pengaruh jumlah iterasi terhadap performa RRT*, baik dari segi kecepatan pemrosesan maupun keakuratan jalur yang dihasilkan.

IV. HASIL DAN PEMBAHASAN

Tabel 1. Percobaan untuk 1000 iterasi

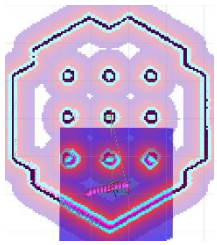
Percobaan	Tujuan	Waktu (s)	Berhasil
1.1	I	5	Ya
1.2	II	-	Tidak
1.3	III	-	Tidak

Tabel 2. Percobaan untuk 2000 iterasi

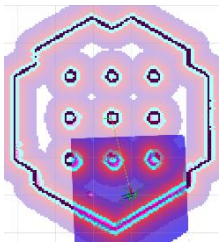
Percobaan	Tujuan	Waktu (s)	Berhasil
2.1	I	9	Ya
2.2	II	-	Tidak
2.3	III	-	Tidak

Tabel 3. Percobaan untuk 5000 iterasi

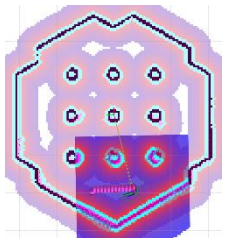
Percobaan	Tujuan	Waktu (s)	Berhasil
3.1	I	18	Ya
3.2	II	25	Ya
3.3	III	-	Tidak



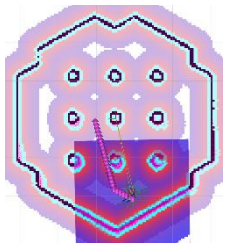
(a) Jalur hasil percobaan 1.1



(b) Jalur hasil percobaan 2.1



(c) Jalur hasil percobaan 3.1



(b) Jalur hasil percobaan 3.2

Gambar 10. Jalur yang terbentuk pada percobaan dengan perencanaan jalur yang berhasil

Pada penelitian ini, hasil simulasi perencanaan jalur menggunakan algoritma RRT* menunjukkan performa yang kurang memuaskan, dengan tingkat keberhasilan hanya 44,44% dari total 9 percobaan. Jalur berhasil ditemukan pada tujuan I dalam semua iterasi (1000, 2000, dan 5000), sementara tujuan II hanya berhasil pada iterasi 5000, dan tujuan III tidak berhasil sama sekali. Pada percobaan yang jalur tidak berhasil ditemukan, waktu tidak dapat dicatat sebab program berjalan terus menerus tanpa henti dan tidak memberikan hasil.

Salah satu tantangan yang dihadapi algoritma berbasis *sampling* adalah penentuan sampel simpul pohon dalam peta. Karena RRT* menentukan sampel secara acak, pembentukan pohon menjadi tidak terarah. Untuk jumlah iterasi kecil, algoritma mungkin tidak dapat menemukan titik tujuan selama pencariannya. RRT* memerlukan jumlah iterasi yang cukup besar untuk dapat menemukan jalur menuju tujuan. Namun, hal ini berarti beban komputasi juga meningkat, seperti yang ditunjukkan pada percobaan 3.1 dan 3.2 di mana algoritma ini memerlukan 18 dan 25 detik secara berturut-turut untuk melakukan kalkulasi jalur, jauh lebih besar dibandingkan percobaan 1.1 dan 2.1.

Faktor lain yang menyebabkan kegagalan adalah konfigurasi parameter yang tidak optimal. Algoritma RRT* memiliki beberapa parameter seperti radius pengambilan sampel, radius *rewiring*, dan jumlah iterasi. Parameter yang melibatkan radius dapat memengaruhi distribusi *sampling* sehingga algoritma menjadi tidak optimal dalam melakukan eksplorasi. Penelitian ini belum melakukan optimasi terhadap penentuan parameter-parameter tersebut sehingga menimbulkan performa yang kurang memuaskan.

Lingkungan pencarian yang kompleks juga menjadi tantangan dalam perencanaan jalur. Rintangan yang banyak seperti dalam dunia simulasi Gazebo ini dapat menyulitkan algoritma dalam melakukan eksplorasi dalam ruang pencarian. Adanya rintangan menyebabkan pengambilan sampel menjadi terhalang sehingga jarak titik yang baru dari titik sebelumnya menjadi tidak maksimal untuk pencarian. Masalah mendasar dalam perencanaan jalur ini belum berhasil diselesaikan dalam penelitian ini.

Di luar masalah perencanaan jalur, terjadi sebuah fenomena pada percobaan 3.2 di mana algoritma telah berhasil membentuk jalur ke tujuan, namun robot tiba-tiba berhenti di tengah perjalanannya. Berbagai forum robotika daring menyebutkan bahwa ini terjadi karena terdapat beberapa titik keadaan atau *pose* di sepanjang jalur yang berada di luar radius maksimum terhadap robot sehingga robot tidak dapat bergerak menuju titik titik tersebut. Kesalahan ini perlu diteliti lebih lanjut dan diperbaiki untuk penelitian selanjutnya yang melibatkan simulasi perencanaan jalur. Terdapat kesalahan juga yakni pada percobaan 2.1, jalur sudah terbentuk namun belum tervisualisasi dengan baik oleh simulator. Diperlukan penelusuran lebih lanjut terkait konfigurasi untuk memperbaiki hal ini.

V. KESIMPULAN DAN SARAN

Evaluasi terhadap simulasi perencanaan jalur menggunakan algoritma RRT* dalam dunia Gazebo menunjukkan bahwa algoritma ini memiliki performa yang belum andal dengan persentase keberhasilan 44,44% untuk seluruh percobaan. Algoritma hanya mampu menentukan jalur untuk titik tujuan dengan jarak yang relatif dekat dengan titik awal. Selain itu, hasil penelitian juga menunjukkan bahwa terdapat efek yang signifikan dari parameter jumlah iterasi terhadap performa sistem. Algoritma masih belum optimal dalam menangani beban komputasi yang besar. Ini menjadi masalah karena RRT* akan memberikan jalur yang semakin optimal seiring bertambahnya jumlah iterasi, yang mana hal ini masih belum bisa ditangani dengan baik. Waktu pemrosesan perencanaan jalur pun masih belum layak agar sistem ini dapat diimplementasikan di lingkungan dan medan yang nyata.

Walau terdapat faktor-faktor yang telah terukur dan diindikasi menjadi penyebab buruknya performa, masih perlu dilakukan penelitian lebih lanjut untuk mengetahui parameter-parameter lain yang berpotensi menjadi penyebabnya, seperti konfigurasi simulasi yang belum tepat serta parameter algoritma yang belum

dioptimalisasi. Parameter yang perlu diteliti meliputi radius pengambilan sampel dan *rewiring*. Untuk melakukan observasi terhadap parameter ini, dapat dilakukan visualisasi yang lebih sederhana seperti pembangkitan grafik koordinat 2D untuk mempermudah analisis. Berbagai kendala mengenai penggunaan ROS2 dan Gazebo yang berpotensi memberikan masalah juga dapat ditelusuri. Simulasi dapat dilakukan dalam dunia Gazebo yang berbeda untuk membandingkan performa di berbagai medan.

RRT* secara murni adalah algoritma yang relatif tertinggal. Saat ini, terdapat berbagai pengembangan dan optimalisasi terhadap algoritma ini seperti RRT*-Connect dan Informed RRT*. Pustaka Navigation2 sendiri telah menggunakan algoritma Behavior Tree yang sudah andal dalam melakukan perencanaan jalur di medan yang bervariasi. Penelitian selanjutnya sebaiknya berfokus pada analisis dan pengembangan algoritma yang lebih maju ini untuk menemukan teknik yang lebih efisien dalam perencanaan jalur.

VI. Lampiran

1. Tautan repository Github simulasi oleh penulis: https://github.com/karolyangqian/RRT-Star-Makalah-IF1220-Matematika-Diskrit
2. Tautan video makalah di Youtube: https://youtu.be/Oif8ixBnzYI
3. Tautan repository Github algoritma RRT*: https://github.com/mmcza/TurtleBot-RRT-Star

VII. Ucapan Terima Kasih

Penulis mengucapkan terima kasih yang sedalam-dalamnya kepada berbagai pihak:

1. Tuhan Yang Maha Esa,
2. orangtua penulis,
3. Dosen pengampu mata kuliah IF1220 Matematika Diskrit dan asisten Lab IRK,
4. pemilik repository Github TurtleBot-RRT-Star,
5. pengembang Turtlebot3, ROS2, dan Gazebo,
6. teman-teman penulis, dan
7. pihak-pihak lain

yang telah mendukung penulis selama proses pembelajaran dan memungkinkan penelitian serta penulisan makalah ini dilaksanakan.

Daftar Pustaka

[1] E. Brandberg, P. Engelking, Y. Jiang, N. Kumar, R. Mbagna-Nanko, R. Narasimhan, A. Rai, S. Shaik, V. R. R. Varikuti, and M. Yu, "Robot Path Planning and Application in Manufacturing Logistics," in *Advanced Robotics for Manufacturing*, Clemson University.

[2] H. Choset and G. Wagner, *Robotic Motion Planning: RRT's*, lecture notes, Robotic Motion Planning, Carnegie Mellon University, Pittsburgh, PA, Fall 2010.

[3] S. Luo, M. Zhang, Y. Zhuang, C. Ma, and Q. Li, 'A survey of path planning of industrial robots based on rapidly exploring random trees', Frontiers in Neurorobotics, vol. 17, 2023.

[4] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, 'Path Planning and Trajectory Planning Algorithms: A General Overview', in *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, G. Carbone and F. Gomez-Bravo, Eds. Cham: Springer International Publishing, 2015, pp. 3–27.

[5] S. Karaman and E. Frazzoli, 'Sampling-based Algorithms for Optimal Motion Planning', *arXiv [cs.RO]*. 2011.

[6] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, I. Ahmedy and I. Ali, "Informed RRT*-Connect: An Asymptotically Optimal Single-Query Path Planning Method," in *IEEE Access*, vol. 8, pp. 19842-19852, 2020, doi: 10.1109/ACCESS.2020.2969316.

[7] A. Lonklang and J. Botzheim, "Mobile Robot Path Planning for Unknown Static Obstacle Avoidance by Improved RRT* Algorithm," 2024 10th International Conference on Automation, Robotics and Applications (ICARA), Athens, Greece, 2024, pp. 155-159, doi: 10.1109/ICARA60736.2024.10553042.

[8] Z. Du and S. Liu, "Asymptotical RRT-Based Path Planning for Mobile Robots in Dynamic Environments," 2018 37th Chinese Control Conference (CCC), Wuhan, China, 2018, pp. 5281-5286, doi: 10.23919/ChiCC.2018.8482649.

[9] M. R. T. Dale, "Shapes of Graphs: Trees to Triangles," in Applying Graph Theory in Ecological Research, Cambridge: Cambridge University Press, 2017, pp. 37–53

[10] R. Munir, *Graf (Bag.1)*, lecture notes, IF1220 Matematika Diskrit, Institut Teknologi Bandung, Bandung, Jawa Barat, 2024.

[11] R. Munir, *Pohon (Bag.1)*, lecture notes, IF1220 Matematika Diskrit, Institut Teknologi Bandung, Bandung, Jawa Barat, 2024.

[12] R. Munir, *Pohon (Bag.2)*, lecture notes, IF1220 Matematika Diskrit, Institut Teknologi Bandung, Bandung, Jawa Barat, 2024.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Januari 2025

Karol Yangqian Poetracahya
13523093