Perencanaan Jalur Robot dalam Labirin Berbasis Grid dengan Algoritma A* pada ROS 2

Karol Yangqian Poetracahya - 13523093
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: karolyangqian14@gmail.com , 13523093@std.stei.itb.ac.id

***Abstrak*—Perencanaan jalur adalah sebuah persoalan krusial di bdiang robotika dan otomasi untuk memenuhi kebutuhan industri di era modern ini. Dengan kebutuhan yang kian bervariasi, sistem otomasi membutuhkan metode untuk beradaptasi pada lingkungan yang kompleks untuk menyelesaikan tugasnya dan mampu menangani berbagai kasus secara umum. Sistem perencanaan jalur memungkinkan robot untuk melakukan berbagai tugas umum seperti bernavigasi di dalam sebuah lingkungan hingga memanipulasi objek dengan efisien apabila menggunakan algoritma yang tepat. Salah satu algoritma perencanaan jalur yang umum digunakan adalah A*. Dalam studi ini, dieksekusi sebuah simulasi perencanaan jalur robot menggunakan ROS 2 dan Gazebo dan divisualisasikan dengan RViz2.**

Kata Kunci—algoritma A*; perencanaan jalur; ROS 2; labirin; Gazebo

I. PENDAHULUAN

Di era modern ini, penerapan sistem robotika dan otomasi pada berbagai bidang industri dituntut untuk lebih otonom, fleksibel, dan memiliki kemampuan belajar untuk menghadapi tantangan seperti kustomisasi massal dan variabilitas produk. Globalisasi dan persaingan pasar menuntut kemampuan adaptasi pada sistem-sistem manufaktur, produksi, atau layanan lain dalam menangani permintaan dan permasalahan yang bervariasi. Sistem robot konvensional yang diprogram untuk melakukan aksi secara sekuensial unggul dalam tugas repetitif dan terstruktur. Namun, dalam skenario yang kompleks dan dinamis, pendekatan seperti ini gagal untuk memberikan solusi yang efisien dan menguntungkan bagi bisnis. Sistem produksi konvensional kini kian beralih menuju *cyber-physical systems* (CPS) untuk meningkatkan produktivitas secara keseluruhan dan menawarkan solusi bagi permasalahan lain seperti dampak lingkungan dan sistem yang *human-centered* dengan cara menggabungkan robot, sensor, dan perangkat heterogen lainnya untuk berpartisipasi dan bekerja sama untuk memberikan hasil yang diharapkan. [1]

Salah satu permasalahan dalam otomasi adalah perencanaan jalur. Proses ini melibatkan algoritma yang efisien untuk menentukan jalur yang optimal untuk mencapai suatu kondisi tujuan dengan mempertimbangkan berbagai parameter seperti keseimbangan eksplorasi, identifikasi dan penghindaran rintangan yang biasanya berkaitan dengan keamanan, optimalisasi rute, efisiensi pencarian, dan lain-lain. Tang, G., dkk [2] membahas tentang penerapan perencanaan jalur dalam lingkungan pelabuhan untuk otomasi logistik pergudangan. Agen yang digunakan adalah sebuah *Autonomous Guided Vehicle* (AGV) yang dapat melakukan pengereman, berbelok, memilih jalan, dan operasi lain tanpa intervensi manusia. Beberapa kebutuhan yang harus dipenuhi oleh sebuah AGV dalam kasus ini yang melibatkan perencanaan jalur meliputi penanganan dan transportasi barang hingga mencari tempat pengisian daya dengan cepat.

Kasus lain yang dibahas oleh Bonci, A., dkk [1] melibatkan lengan robotik dengan 6 derajat kebebasan (*degree of freedom,* DOF) untuk memanipulasi sekumpulan benda di depannya. Sistem ini merupakan kolaborasi antara sensor kamera Intel RealSense Model D435i untuk persepsi dan manipulator yang terdiri dari lengan 6 DOF dan *gripper*. Salah satu subsistem—*Planning and Control Layer*—memiliki tanggung jawab untuk merencanakan perilaku, lintasan, dan pergerakan lengan robotik dari titik awal menuju *pose* tujuan untuk manipulasi objek. ROS2 digunakan sebagai *middleware* sistem dan menjadi opsi *framework* yang terpilih karena menawarkan sejumlah API dan kakas grafis, seperti MoveIt 2 dan RViz2, untuk membantu menyelesaikan masalah perencanaan pergerakan yang standar. Pustaka perencanaan pergerakan umum yang kompatibel dengan ROS2 adalah Open Motion Planning Library (OMPL) yang memiliki sejumlah implementasi algoritma *sampling-based* mutakhir seperti RRT (Rapidly Exploring Random Tree), PRM (Probabilistic Roadmap Method), dan masih banyak lagi. Pada studi referensi ini, perencanaan lintasan diimplementasikan dengan konfigurasi dasar MoveIt 2 yang melibatkan detektor tumbukan FCL dan algoritma perencana pergerakan RRTConnect dari OMPL lalu divisualisasikan dalam RViz2, memungkinkan perencanaan lintasan ditampilkan secara grafis sebelum dieksekusi pada robot.

Pada studi ini, sebuah robot dengan sistem kendali yang diimplementasikan dengan ROS 2 disimulasikan dalam simulator grafis Gazebo untuk bernavigasi dalam labirin virtual dengan memanfaatkan algoritma A* sebagai strategi perencanaan jalur. Gazebo memungkinkan simulasi fisik robot yang realistis untuk pengujian perilaku dalam dunia virtual sebelum dieksekusi di dunia nyata. Selain itu, Gazebo juga telah memiliki sekumpulan API pada ROS 2, meliputi *spawner* dan *server*, yang mempermudah integrasi dan implementasi menggunakan berbagai kakas yang disediakan dalam ROS 2. Fokus dari studi ini adalah pengimplementasian algoritma A*

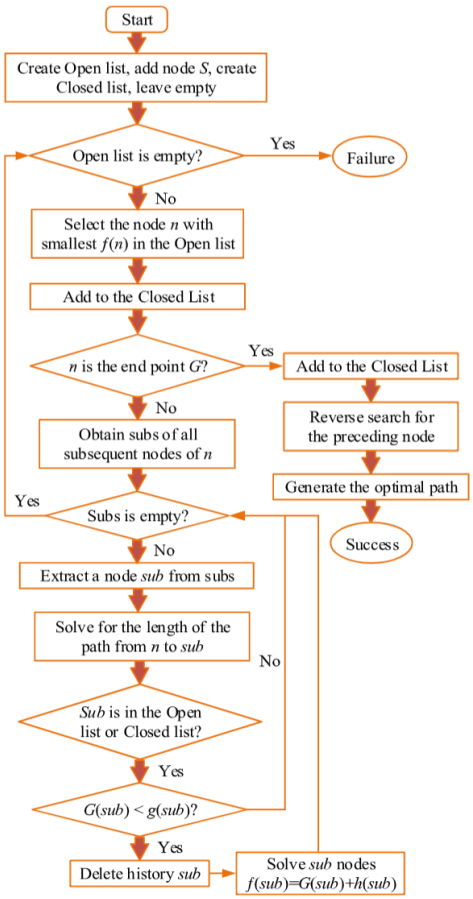
dalam perencanaan jalur. Pembangkitan labirin berbasis *occupancy grid* memanfaatkan pustaka dari kreator eksternal (terlampir) dan model robot kendali diferensial diambil dari proyek sumber terbuka Turtlebot3 yang menyediakan sejumlah model robot dengan pustaka dan antarmuka yang standar dan terintegrasi serta dapat dieksekusi dengan sederhana. Analisis hasil studi dilakukan secara kualitatif dengan mengobservasi visualisasi grafis jalur yang dibangkitkan algoritma.

II. Landasan Teori

A. *Algoritma A* untuk Pencarian Rute*

Algoritma A* adalah algoritma pencarian graf yang digunakan untuk mencari rute terpendek dari sebuah titik awal berupa menuju simpul tujuan akhir. Algoritma ini digagas oleh Peter Hart, Nils Nelson, dan Bertram Raphael pada 1968 sebagai pengembangan dari algoritma Dijkstra. A* menggabungkan pendekatan *greedy* menggunakan heuristik yang diterapkan pada algoritma Greedy Best-First Search dengan pendekatan nilai *cost* dari simpul awal menuju simpul saat ini yang diterapkan pada algoritma Dijsktra dan Uniform Cost Search.

$$f(x) = g(x) + h(x) \quad (1)$$



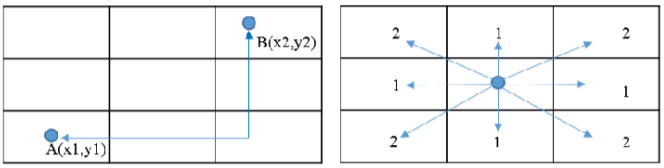
Gambar 1. Bagan Alir Algoritma A* [3]
Sumber: http://dx.doi.org/10.1109/ACCESS.2022.3179397

Bagian utama dari algoritma A* terdapat pada perhitungan fungsi biaya atau *cost* total yang dinyatakan dalam persamaan (1). Biaya total pada simpul x dinyatakan sebagai $f(x)$ yang diekspresikan sebagai jumlah dari biaya untuk mencapai simpul tersebut dari simpul awal (dilambangkan sebagai $g(x)$) dengan estimasi heuristik biaya dari simpul tersebut ke simpul tujuan (dilambangkan sebagai $h(x)$). Beberapa implementasi dari $h(x)$ meliputi Euclidean Distance, Manhattan Distance, dan Chebyshev Distance.

B. *Jarak Manhattan*

Jarak Manhattan adalah sebuah parameter heuristik yang dapat digunakan pada algoritma pencarian rute berbasis heuristik seperti A*. Jarak Manhattan dari dua buah titik didefinisikan sebagai jumlah dari selisih mutlak komponen koordinat kartesian kedua titik.

$$manhattan(a,b) = |x_a - x_b| + |y_a - y_b| \quad (2)$$

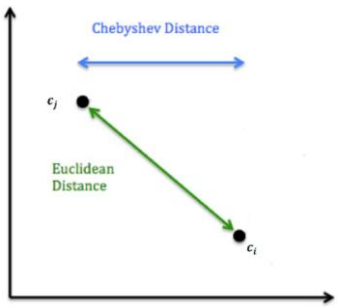


Gambar 2. Representasi Graf Jarak Manhattan [4]
Sumber: http://dx.doi.org/10.1145/3338472.3338490

C. *Jarak Chebyshev*

Metode lain dalam penghitungan heuristik untuk algoritma pencarian rute adalah Jarak Chebyshev. Parameter ini dikalkulasi dengan mencari maksimum dari mutlak selisih setiap komponen koordinat kartesian kedua titik. Perhitungan jarak ini mengasumsikan bahwa bergerak secara diagonal pada bidang berbasis grid memiliki biaya yang sama dengan bergerak secara paralel terhadap sumbu kartesian.

$$chebyshev(a,b) = \max(|x_a - x_b|, |y_a - y_b|) \quad (3)$$



Gambar 3. Visualisasi Jarak Chebyshev Dibandingkan dengan Jarak Euclidean [4]
Sumber: http://dx.doi.org/10.22075/ijnaa.2022.5784

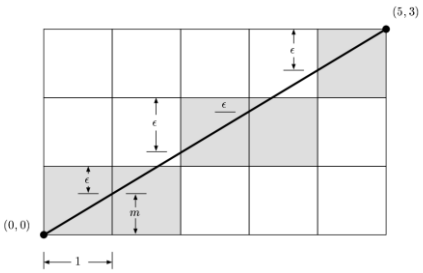
Variasi lain dari Jarak Chebyshev adalah dengan tetap memperhitungkan biaya secara Euclidean untuk pergerakan diagonal namun kalkulasi biaya total tetap mematuhi

mekanisme pergerakan layaknya ratu pada permainan catur seperti yang diasumsikan pada kalkulasi Jarak Chebyshev. Dengan demikian, fungsi heuristik variasi ini dinyatakan sebagai (4).

$$dist(a,b) = \max(|x_a - x_b|, |y_a - y_b|) + (\sqrt{2} - 1) * \min(|x_a - x_b|, |y_a - y_b|) \quad (4)$$

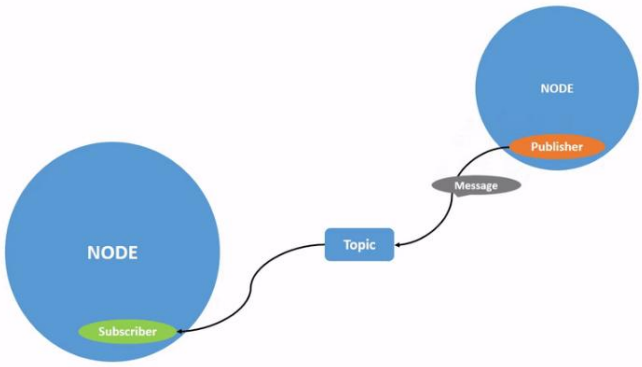
D. Algoritma Garis Bresenham

Algoritma Garis Bresenham digunakan untuk menggambar sebuah garis pada layar digital yang berbasis pixel. Motivasi utama dari algoritma ini adalah untuk menentukan sekumpulan pixel pada layar atau sel pada bidang grid yang dapat memvisualisasikan sebuah garis yang direpresentasikan sebagai dua buah titik koordinat pada bidang tersebut. Dalam kasus implementasi perencanaan jalur, algoritma ini dapat digunakan untuk mekanisme *path smoothing* dengan cara mencari simpul terjauh yang dapat dilihat dari simpul saat ini berdasarkan Algoritma Garis Bresenham lalu membuang simpul-simpul di antaranya untuk mengefisienkan jalur.



Gambar 4. Visualisasi Algoritma Garis Bresenham [5]
Sumber: https://www.cs.put.poznan.pl/swilk/pmwiki/uploads/Dydaktyka/bresenham-int.pdf

E. Arsitektur ROS 2



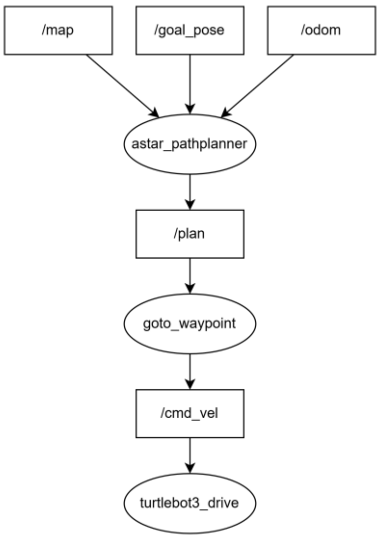
Gambar 5. Model *Publisher & Subscriber* pada ROS 2

Sumber: https://docs.ros.org/

ROS 2 menyediakan berbagai pustaka untuk mengimplementasikan *middleware* komunikasi antarkomponen dalam sebuah sistem robotika. Pola desain arsitektur yang diusung oleh ROS 2 menerapkan skema *publisher* dan *subscriber*. Sebuah proses direpresentasikan sebagai *node* dan sebuah data *share memory* direpresentasikan sebagai *topic*. *Node* dapat mengumumkan (*publish*) dan berlangganan (*subscribe*) ke sejumlah topik untuk berkomunikasi dengan *node* lain. Desain arsitektur perangkat lunak ini memastikan modularitas sistem dan memungkinkan pengisolasian pada setiap subsistem yang ada.

III. Desain dan Implementasi

A. Desain Sistem Robot dan Simulasi



Gambar 6. Diagram *Publisher & Subscriber* Sistem Robot

Dalam studi ini, digunakan robot terintegrasi yang disediakan oleh Turtlebot3 untuk kerangka dasar sistem aktuator dan kinematika gerak. Robot mengusung model pergerakan diferensial yang terdiri atas dua buah roda di sisi kiri dan kanan *base*. Robot dapat melakukan lokomosi dengan bergerak maju dan mundur serta berotasi pada sumbu putar yang terletak di dalam *base*. Sistem Turtlebot3 yang digunakan hanya terbatas pada lapisan kendali aktuator dengan antarmuka perintah kecepatan relatif robot yakni topik */cmd_vel*.

Di atas lapisan kendali aktuator, diimplementasikan lapisan kendali lokomosi robot dan lapisan perencanaan jalur sebagai dua subsistem yang berbeda. Skema alur data mulai dari input titik tujuan hingga mencapai kendali aktuator adalah sebagai berikut:

1. Pengguna memberikan titik tujuan berupa pesan Pose ke topik */goal_pose* melalui kakas grafis RViz2.
2. *Node* perencanaan jalur menerima titik tujuan beserta data OccupancyGrid dari *map server* melalui topik */map* dan lokasi robot saat ini melalui topik */odom* yang diumumkan oleh Gazebo.
3. *Node* perencanaan jalur mencari rute ke titik tujuan dari lokasi robot saat ini menggunakan algoritma A* lalu mengumumkan pesan Path ke topik */plan*. Path adalah pesan yang terdiri atas beberapa Pose yang merepresentasikan jalur dari titik awal ke tujuan.

4. Subsistem kendali lokomosi menerima Path dari topik */plan* lalu mengeksekusi *loop* yang menggerakkan robot mengikuti jalur yang diberikan dengan cara memberikan perintah berupa pesan Twist ke topik */cmd_vel* secara berkala. *Loop* dieksekusi hingga robot mencapai Pose terakhir dalam Path.
5. Subsistem kendali aktuator menerima pesan dari topik */cmd_vel* lalu melakukan kalkulasi invers kinematika untuk menggerakkan aktuator roda sesuai perintah kecepatan relatif robot yang diberikan.

Robot diinstansiasikan dalam sebuah dunia kosong di Gazebo. Labirin yang menghadirkan peta rintangan bagi robot diinstansiasikan secara virtual melalui topik */map* yang diumumkan oleh *node map_server*. Dengan demikian, walau tidak terdapat rintangan fisik dalam dunia Gazebo, subsistem perencanaan jalur tetap dapat mengidentifikasi rintangan labirin berupa pesan OccupancyGrid. Peta OccupancGrid labirin dibangkitkan dengan program pembangkit OccupancyGrid eksternal (sumber terlampir) yang menghasilkan *file* Portable Gray Map (PGM) beserta konfigurasinya dalam *file* YAML. *File* ini dibaca oleh *map_server* untuk menghasilkan pesan OccupancyGrid yang dapat digunakan dalam siklus ROS 2. Peta dan jalur yang dihasilkan beserta posisi global robot divisualisasikan dalam kakas grafis RViz2.

Sistem robot dan program ini dirancang untuk memenuhi beberapa kebutuhan berikut:

1. Sistem mampu membangkitkan jalur dari posisi awal robot menuju titik posisi akhir yang diberikan pengguna menggunakan algoritma A*.
2. Robot mampu berlokomosi dari posisi awal menuju posisi akhir dengan mengikuti jalur yang dibangkitkan.
3. Jalur dan peta labirin divisualisasikan secara grafis agar dapat dilihat oleh pengguna.

B. Subsistem Kendali Lokomosi

Subsistem kendali lokomosi diimplementasikan sebagai *node goto_waypoint* yang berlangganan ke topik */plan* untuk menerima jalur yang telah direncanakan lalu menggerakkan robot mengikuti jalur yang diberikan. Algoritma penelusuran jalur diimplementasikan sebagai *closed loop* dengan *feedback* proporsional berupa sudut simpang robot terhadap garis yang menghubungkan robot dengan Pose berikutnya pada jalur yang sedang ditempuh. Apabila sudut simpang terlalu besar, *node goto_waypoint* akan memerintahkan robot mengurangi sudut simpan dengan cara memperbesar nilai mutlak kecepatan sudut. Robot dinyatakan sudah mencapai sebuah Pose dalam jalur apabila sudah memasuki radius tertentu terhadap Pose yang dituju. Ketika robot sudah mencapai Pose terakhir dalam jalur, *closed loop* diselesaikan.

C. Subsistem Perencanaan Jalur

Subsistem perencanaan jalur diimplementasikan sebagai *node astar_pathplanner* yang melakukan langganan ke tiga topik, yakni */map*, */goal_pose*, dan */odom*, lalu mengumumkan hasil perencanaan jalur ke topik */plan*. *Node* ini bekerja berdasarkan *trigger* kemunculan pesan pada topik */goal_pose* dari masukan pengguna. Untuk melakukan pencarian rute berdasarkan input dari ketiga topik, diimplementasikan kelas Astar yang dikomposisikan ke dalam *node astar_pathplanner*. Ketika terdapat pesan titik tujuan yang masuk, *node* ini memiliki tanggung jawab untuk meneruskan input tersebut ke fungsi pencarian rute pada instansiasi objek Astar yang dimilikinya. Pengembalian fungsi ini adalah jalur dari titik awal ke tujuan berupa pesan Path yang kemudian diumumkan ke topik */plan*.

*D. Implementasi Algoritma A**

```
struct AStarNode
{
  int x, y;           // Grid coordinates
  double f, g, h;     // A* costs
  AStarNode* parent; // Parent node for path
reconstruction

  AStarNode(int x = 0, int y = 0, double g =
0.0, double h = 0.0, AStarNode* parent =
nullptr)
    : x(x), y(y), g(g), h(h), f(g + h),
parent(parent) {}

  bool operator>(const AStarNode& other) const
{
    return f > other.f;
  }
};
```

Dibentuk sebuah kelas bernama Astar yang khusus untuk menangani pencarian rute menggunakan algoritma A*. Kelas ini menggunakan pendefinisian tersendiri mengenai simpul untuk kebutuhan algoritma, yakni sebuah *struct* yang memiliki integer x dan y sebagai posisi simpul, *double* f, g, dan h sebagai nilai biaya total, biaya dari simpul awal, dan nilai parameter heuristik simpul, konstruktor, dan operator perbandingan.

```
// Coordinate conversion functions
bool worldToGrid(double world_x, double
world_y, int& grid_x, int& grid_y) const;
void gridToWorld(int grid_x, int grid_y,
double& world_x, double& world_y) const;

// Path processing
nav_msgs::msg::Path gridPathToWorldPath(
const std::vector<std::pair<int, int>>&
grid_path,
const std::string& frame_id) const;
std::vector<std::pair<int, int>> smoothPath(
const std::vector<std::pair<int, int>>& path)
const;
```

Karena kelas ini memiliki pendefinsian simpul dan graf peta yang berbeda dengan tipe pesan standar pada ROS 2, dibuat implementasi fungsi untuk melakukan transformasi nilai-nilai dalam representasi menurut ROS 2 menuju representasi menurut kelas ini. Beberapa fungsi tersebut meliputi fungsi transformasi koordinat simulasi menuju koordinat grid dan sebaliknya yang menggunakan perhitungan skala serta fungsi transformasi jalur

pada representasi grid menjadi pesan Path yang kompatibel dengan ROS 2.

```
std::priority_queue<AStarNode,
std::vector<AStarNode>,
std::greater<AStarNode>> open_set;

std::unordered_map<std::pair<int, int>, bool,
NodeHash> closed_set;

std::unordered_map<std::pair<int, int>, double,
NodeHash> g_costs;

std::unordered_map<std::pair<int, int>,
AStarNode*, NodeHash> all_nodes;
```

Setalah mendapatkan koordinat hasil transformasi, pencarian rute dilakukan sebagai tanggung jawab dari fungsi utama kelas yakni findPath(). Dalam fungsi ini, digunakan tipe-tipe dari pustaka standar C++ untuk merepresentasikan komponen-komponen dari algoritma A*, seperti *std::priority_queue* untuk *open list/set* dan *std::unordered_map* untuk *closed list/set*, memoisasi biaya g, dan penyimpanan setiap simpul dalam graf peta.

```
if (current.x == goal_x && current.y == goal_y)
{
    std::vector<std::pair<int, int>> raw_path =
reconstructPath(all_nodes[current_pos]);

    path = smoothPath(raw_path);

    for (auto& node_pair : all_nodes) {
    delete node_pair.second;
    }

    return path;
}
```

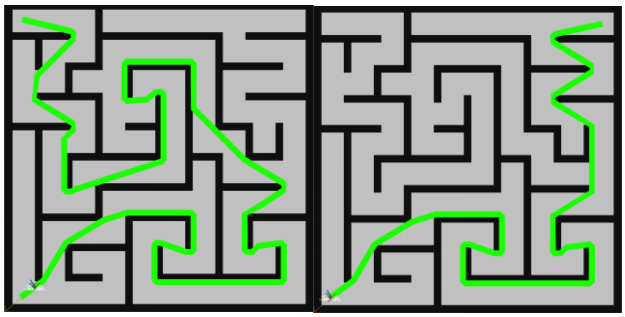
Melanjutkan proses setelah titik tujuan ditemukan menggunakan algoritma A*, rute dikonstruksi ulang dengan fungsi reconstructPath() yang memanfaatkan struktur data simpul yang membentuk *linked list* di mana setiap simpul menunjuk ke simpul orangtuanya. Struktur data ini membuat proses rekonstruksi rute menjadi efisien. Setelah rute direkonstruksi, rute dihaluskan dengan algoritma yang membuang simpul-simpul redundan dengan cara mencari simpul terjauh yang dapat dilihat dari simpul saat ini menurut Algoritma Garis Bresenham.

```
// Convert grid path to world path
nav_msgs::msg::Path world_path =
astar_.gridPathToWorldPath(grid_path,
msg->header.frame_id);
world_path.header.stamp = this->now();

// Publish the path
plan_pub_->publish(world_path);

RCLCPP_INFO(this->get_logger(), "Path published
with %zu waypoints", world_path.poses.size());
```

Jalur halus dalam representasi grid yang telah ditemukan menggunakan algoritma A* ini kemudian ditransformasikan kembali dalam representasi tipe Path. Transformasi ini dilakukan menggunakan fungsi gridPathToWorldPath() yang dipanggil dalam kode implementasi *node astar_pathplanner*. Jalur ini kemudian diumumkan ke topic */plan* sesuai rancangan skema alur data sistem.

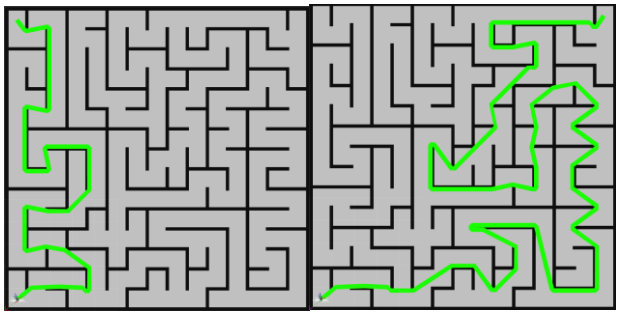
IV. Hasil dan Pembahasan

Jalur yang dibangkitkan sistem perencanaan jalur divisualisasikan pada RVIz2 sebagai rute berwarna hijau terang di atas peta labirin. Segmen berwarna hitam pada labirin menandakan tembok (*occupied cell* dalam OccupancyGrid) yang tidak bisa dilewati robot, sedangkan segmen berwarna abu-abu adalah sel bebas yang dapat dilewati robot.



(a) (b)

Gambar 7. Hasil Perencanaan Jalur pada Labirin 10x10 dengan Titik Tujuan di (a) Kiri Atas dan (b) Kanan Atas Labirin



(a) (b)

Gambar 8. Hasil Perencanaan Jalur pada Labirin 15x15 dengan Titik Tujuan di (a) Kiri Atas dan (b) Kanan Atas Labirin

Secara umum, kebutuhan yang terdapat dalam rancangan sistem robot dan program telah terpenuhi dengan baik. Subsistem perencanaan jalur berhasil membangkitkan jalur yang mencapai posisi tujuan dari posisi awal robot dan tervisualisasi dengan baik. Ketika didemonstrasikan, robot juga mampu mencapai posisi tujuan dengan *tuning* parameter kecepatan yang benar.

Karena bentuk labirin yang kompleks, penilaian efisiensi dan efektivitas jalur yang dibangkitkan membutuhkan studi yang lebih lanjut. Di samping itu, hasil visualisasi jalur menggambarkan ciri khas jalur yang dibangkitkan oleh algoritma A*. Terdapat dua karakteristik yang muncul pada dua kasus berbeda mengenai lorong labirin. Pertama, pada kasus di

mana jalur terbentuk pada bagian labirin yang membentuk lorong dan jalur harus berbelok di akhir lorong dengan arah yang sama dengan arah belok ketika memasuki lorong (misalnya, jalur memasuki lorong dengan berbelo ke kiri dan harus keluar dengan berbelok ke kiri lagi), maka jalur akan menempel berhimpitan dengan tembok. Pada kasus kedua, yakni ketika arah belok keluar lorong berbeda dengan arah masuk (misalnya, jalur memasuki lorong dengan berbelok ke kiri dan harus keluar dari lorong dengan berbelok ke kanan), maka jalur akan membentuk garis lurus yang secara efisien menghubungkan sudut jalan masuk dan keluar lorong tersebut. Ini adalah efek yang dihadirkan dengan adanya parameter biaya dari titik awal (g) dalam penghitungan biaya total simpul. Fungsi biaya ini memberikan efek pembangkitan jalur yang tegas dan efisien seperti karakteristik optimalitas algoritma Uniform Cost Search.

Karakteristik lain dari jalur yang dibangkitkan adalah pada kasus jalur sudah mencapai ruang terbuka pada labirin dan tidak ada penghalang lagi untuk mencapai posisi akhir. Pada kondisi ini, jalur yang terbentuk secara tegas dan efisien langsung berekspansi menuju tujuan. Efek ini ditimbulkan oleh fungsi heuristik yang mengkalkulasi estimasi jarak simpul saat ini ke simpul tujuan dan menjadikannya parameter untuk memilih simpul terbaik berikutnya untuk ditelusuri.

V. KESIMPULAN

Sistem yang diimplementasikan dalam studi ini telah memenuhi setiap kebutuhan dalam rancangan dan telah menggambarkan karakteristik dari algoritma A* dalam perencanaan jalur robot dalam labirin. Subsistem perencanaan jalur mampu membangkitkan jalur berdasarkan input posisi tujuan dari pengguna serta input informasi peta dan posisi awal robot dari sistem. Jalur kemudian dapat diterima oleh subsistem kendali lokomosi untuk menggerakkan robot ke posisi tujuan dengan mengikuti jalur yang dibangkitkan.

Untuk pengembangan lebih lanjut, sistem dapat diperbaiki dengan menambahkan sistem lokalisasi yang memungkinkan robot mengetahui lokasi global pada dunia nyata berdasarkan akuisisi data sensor. Fitur ini memungkinkan pengembangan perencanaan jalur pada lingkungan yang dinamis dan robot menjadi adaptif terhadap lingkungannya.

TAUTAN VIDEO DI YOUTUBE

https://youtu.be/M2Xfe0WTDPQ

TAUTAN REPOSITORI GITHUB

https://github.com/karolyangqian/stimabot_ros

TAUTAN REPOSITORI GITHUB KREATOR PEMBANGKIT LABIRIN

https://github.com/Geckostya/maze_generator

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 IF2211 Strategi Algoritma dan asisten,
4. Pemilik repositori Github maze_generator,
5. Pengembang Turtlebot3, ROS2, dan Gazebo,
6. Teman-teman anggota tim Garudago Kru 15 yang telah mengajari ROS 2 kepada penulis,
7. Teman-teman penulis lainnya, dan
8. Pihak-pihak lain

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

DAFTAR PUSTAKA

[1] A. Bonci, F. Gaudeni, M. C. Giannini, and S. Longhi, "Robot Operating System 2 (ROS2)-Based Frameworks for Increasing Robot Autonomy: A Survey," *Appl. Sci.*, vol. 13, no. 23, p. 12796, Nov. 2023. doi: 10.3390/app132312796.

[2] G. Tang, C. Tang, C. Claramunt, X. Hu and P. Zhou, "Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment," in *IEEE Access*, vol. 9, pp. 59196-59210, 2021, doi: 10.1109/ACCESS.2021.3070054.

[3] Y. Li *et al.*, 'A Mobile Robot Path Planning Algorithm Based on Improved A* Algorithm and Dynamic Window Approach', *IEEE Access*, vol. 10, pp. 1–1, 01 2022.

[4] L. Ali and S. Adel, 'A local density-based outlier detection method for high dimension data', *The International Journal of Nonlinear Analysis and Applications (IJNAA)*, vol. 13, 10 2021.

[5] K. I. Joy, *On-Line Computer Graphics Notes: Bresenham's Algorithm*, Visualization and Graphics Research Group, Department of Computer Science, University of California, Davis. [Online]. Available: https://www.cs.put.poznan.pl/swilk/pmwiki/uploads/Dydaktyka/bresenham-int.pdf

[6] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, 'Path Planning and Trajectory Planning Algorithms: A General Overview', *Mechanisms and Machine Science*, vol. 29, pp. 3–27, 03 2015.

[7] M. Piras, "Path planning strategies for autonomous vehicles," M.S. thesis, Dept. Control Comput. Eng., Politecnico di Torino, Turin, Italy, 2024.

[8] T.-V. Dang, 'Optimization Hybrid Path Planning based on A-star Algorithm combining with DWA', *MM Science Journal*, vol. 10, pp. 7551–7555, 10 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, 24 Juni 2025

Karol Yangqian Poetracahya