Nama : Ketut Satria Wibisana
NIM : 1103213148
Kelas : TK-45-G09

Analisis Tiga Algoritma Perencanaan Jalur dan ROS Motion Planning

1. **Analisis Tiga Algoritma Perencanaan Jalur**
   - **Algoritma Dijkstra**

     Kode Python pada `dijkstra_planner.py` ini menerapkan algoritma Dijkstra, yang terkenal sebagai salah satu algoritma pencarian jalur terpendek. Algoritma ini berfungsi untuk menemukan jalur dengan biaya paling rendah antara dua titik dalam suatu graf. Cara kerja algoritma ini dimulai dari node awal yang ditentukan, lalu memanfaatkan struktur data antrian prioritas (atau min-heap) untuk menelusuri node tetangga satu per satu. Setiap kali algoritma akan memilih node dengan biaya total terkecil yang telah terakumulasi untuk diperiksa lebih lanjut. Setiap kali algoritma bergerak ke node tetangga, ia akan membandingkan jarak total dari node awal menuju node tersebut. Jika ditemukan jalur yang memiliki jarak lebih pendek daripada sebelumnya, algoritma memperbarui jarak minimum untuk node tersebut. Algoritma Dijkstra ini juga memiliki mekanisme pencatatan terhadap node-node yang sudah dikunjungi, sehingga tidak perlu meninjau kembali node yang sama, menghemat waktu dan langkah perhitungan. Algoritma Dijkstra ini akan terus berlanjut sampai node tujuan tercapai, pada titik mana ia akan berhenti dan mengembalikan informasi mengenai biaya minimum yang diperlukan serta rute atau jalur yang harus ditempuh untuk mencapai tujuan tersebut. Dalam penerapannya pada `dijkstra_planner.py`, graf direpresentasikan menggunakan daftar ketetanggaan, di mana algoritma menemukan jalur terpendek dari node 'A' ke node 'D'. Hasil akhirnya menunjukkan bahwa biaya total untuk mencapai node 'D' dari 'A' adalah 4, dengan rute yang ditempuh adalah ['A', 'B', 'C', 'D'].

   - **Algoritma A***

     Script Python dalam `a_star_planner.py` mengimplementasikan algoritma pencarian jalur A*, yang merupakan modifikasi dari algoritma Dijkstra. Algoritma A* dirancang untuk menemukan jalur dengan biaya terendah dari satu titik ke titik lainnya dalam sebuah graf, namun dengan tambahan komponen heuristik yang memperkirakan jarak yang masih harus ditempuh menuju tujuan. Tambahan heuristik ini memungkinkan A* untuk lebih cerdas dalam memilih jalur yang lebih mungkin mencapai tujuan dengan lebih cepat. Algoritma A* menggunakan dua nilai utama dalam perhitungannya. Yang pertama adalah nilai `g`, yaitu total biaya aktual yang diperlukan untuk bergerak dari titik awal ke node tertentu. Nilai kedua adalah `f`, yang merupakan kombinasi dari `g` dan nilai estimasi heuristik untuk jarak ke tujuan yang tersisa. Dalam kasus ini, heuristik dihitung menggunakan jarak Manhattan, yang memberikan perkiraan seberapa jauh node tersebut dari

tujuan secara horizontal dan vertikal. Dengan cara ini, algoritma A* mampu memperhitungkan baik jarak yang sudah ditempuh maupun prediksi jarak yang tersisa. Proses kerja A* dimulai dari node awal dan memilih node berikutnya berdasarkan nilai `f` yang paling rendah, yaitu node dengan kombinasi biaya terendah untuk sampai ke sana serta prediksi jarak paling dekat ke tujuan. Dengan pendekatan ini, A* berupaya menyeimbangkan antara eksplorasi jalur terpendek dan upaya untuk mencapai tujuan secepat mungkin. Setiap kali algoritma menemukan node dengan nilai `f` yang lebih rendah daripada nilai sebelumnya untuk node yang sama, ia memperbarui nilai tersebut dan melanjutkan pencarian dari node tersebut. Pada skenario dalam script ini, graf direpresentasikan sebagai grid 2D di mana algoritma A* menghitung jalur dari posisi pojok kiri atas hingga ke pojok kanan bawah, sambil menghindari rintangan yang ada pada grid tersebut. Dengan menggunakan mekanisme ini, algoritma A* berhasil menemukan jalur optimal yang tidak hanya menghindari hambatan, tetapi juga mengarahkan menuju tujuan dengan jalur yang paling efisien.

- **Algoritma Cell Decomposition**

  Script Python pada `cell_decomposition.py` ini mengimplementasikan metode dasar cell decomposition, yang merupakan teknik populer dalam perencanaan jalur di bidang robotika. Metode cell decomposition membagi ruang yang memiliki rintangan menjadi sejumlah sel-sel kecil atau area-area bebas rintangan, yang memungkinkan perencanaan jalur menjadi lebih sederhana. Teknik ini sangat berguna karena memungkinkan robot atau objek lain bergerak secara efisien di ruang yang kompleks dengan cara yang lebih terstruktur. Script ini berfokus pada dua fungsi utama yang mendukung proses cell decomposition, yaitu `create_cells` dan `find_path_through_cells`. Fungsi `create_cells` bertanggung jawab untuk membentuk atau menghasilkan sel-sel berdasarkan letak dan ukuran rintangan di dalam ruang. Dengan membagi ruang menjadi bagian-bagian yang bebas dari rintangan, fungsi ini memudahkan dalam menentukan area mana saja yang bisa dilalui. Fungsi kedua, `find_path_through_cells`, bertujuan untuk menemukan jalur sederhana dari titik awal menuju titik tujuan dengan melewati sel-sel yang telah didefinisikan sebagai area bebas rintangan. Fungsi ini mencari jalur terpendek atau paling langsung di antara sel-sel, sehingga memungkinkan pergerakan yang efektif antara dua titik tanpa harus menabrak rintangan.Meskipun implementasi dalam script ini masih tergolong dasar dan tidak mencakup logika pembentukan sel atau navigasi yang kompleks, script ini berfungsi sebagai fondasi untuk membangun sistem perencanaan jalur berbasis cell decomposition yang lebih canggih. Misalnya, tahap-tahap pengembangan berikutnya dapat mencakup optimalisasi pembagian sel, deteksi rintangan yang lebih akurat, dan penambahan logika navigasi yang lebih adaptif terhadap perubahan lingkungan. Dengan kerangka dasar ini, script dapat menghasilkan output yang menunjukkan jalur sederhana yang menghubungkan titik awal ke titik tujuan, sehingga memberikan ilustrasi dasar cara cell decomposition bekerja dalam perencanaan jalur.

2. Analisis ROS Motion Planning

ROS (Robot Operating System) menyediakan kerangka kerja yang esensial untuk proses perencanaan gerakan (motion planning) dalam robotika, yang bertujuan mengarahkan robot dari posisi awal menuju tujuan sambil menghindari rintangan, serta memastikan bahwa batasan kinematik dan dinamik robot tetap terpenuhi. Salah satu alat utama dalam ROS untuk mendukung perencanaan gerakan adalah MoveIt!, yang secara luas diterapkan pada lengan robotik dengan berbagai derajat kebebasan (DOF). Dengan MoveIt!, robot mampu merencanakan gerakan yang aman dan efisien melalui berbagai algoritma pencarian jalur seperti Rapidly-exploring Random Tree (RRT), Probabilistic Roadmap (PRM), dan A*. Algoritma ini berfungsi menemukan jalur optimal yang tidak hanya cepat tetapi juga aman dari tabrakan. Setelah jalur awal ditemukan, MoveIt! melanjutkan dengan tahap perencanaan lintasan yang bertujuan menghaluskan jalur tersebut sehingga robot dapat bergerak dengan gerakan yang lembut, mempertimbangkan variabel seperti kecepatan maksimum, percepatan, dan karakteristik dinamis lainnya. Untuk memastikan robot tidak bertabrakan dengan objek di sekitar, ROS menggunakan alat pemeriksaan tabrakan seperti Flexible Collision Library (FCL), yang menganalisis area sekitar robot untuk mendeteksi kemungkinan tabrakan dan menghindarinya. Khusus untuk lengan robot, perencanaan gerakan mencakup proses yang disebut kinematika invers. Dalam kinematika invers, sistem menghitung posisi sendi-sendi tertentu yang diperlukan agar ujung lengan robot mencapai lokasi yang diinginkan di ruang kerja, memperhitungkan berbagai kendala agar pergerakan tetap berada dalam batas aman. Di sisi lain, untuk robot bergerak seperti robot mobile atau otonom, elemen penting lainnya adalah kemampuan lokalisasi dan pemetaan. Metode Simultaneous Localization and Mapping (SLAM) memungkinkan robot untuk memetakan lingkungannya secara simultan sambil melokalisasi posisinya di dalam peta tersebut, sehingga robot dapat menavigasi di lingkungan yang belum dikenalnya. Seluruh proses ini ditutup dengan tahap pengendalian gerakan, di mana kontroler seperti Proportional-Integral-Derivative (PID) controllers diterapkan untuk menerjemahkan lintasan yang direncanakan ke dalam perintah yang dieksekusi oleh aktuator robot. Penggunaan kontroler ini memastikan robot bergerak sesuai rencana dengan tingkat kehalusan dan presisi yang tinggi, serta mampu beradaptasi terhadap perubahan kondisi atau gangguan yang mungkin terjadi selama pergerakan. Dengan komponen-komponen ini, ROS memberikan dukungan komprehensif bagi perencanaan gerakan yang andal dalam robotika, memungkinkan robot bergerak dengan aman, efisien, dan presisi dalam beragam lingkungan.