ANALISIS NO 1 DAN NO 2

- Analisis 1 implementasi 3 algoritma perencanaan jalur
  1. Dijkstra Algorithm
     Algoritma ini menemukan jalur terpendek dari satu titik ke titik tujuan dalam graf dengan bobot non-negatif, melalui penjelajahan node dengan jarak terpendek. Kelebihan Efisien pada graf dengan bobot non-negatif dan banyak digunakan dalam aplikasi seperti penentuan rute. Kekurangan Tidak bisa menangani bobot negatif dan tidak menggunakan heuristic, sehingga bisa lambat. Pada Kode Python menggunakan `networkx` dan `matplotlib` untuk membuat graf dan memvisualisasikan jalur terpendek.
  2. A* Algorithm
     Untuk mempercepat pencarian dalam graf, algoritma ini menggunakan heuristic dan jalur terpendek. Algoritma ini sangat cocok untuk grid dengan rintangan. Metode ini Lebih cepat dari Dijkstra dalam beberapa kasus karena heuristic-nya. Cocok untuk lingkungan dengan banyak rintangan. Namun Performa sangat bergantung pada kualitas heuristic. Implementasi bisa lebih rumit karena perlu mengelola set terbuka dan tertutup. Pada Kode Python menggunakan `pygame` untuk visualisasi interaktif, memungkinkan pengguna menentukan titik awal, tujuan, dan rintangan.
  3. Cell Decomposition
     memecah area kompleks menjadi sel-sel kecil untuk mencari jalur aman di antara sel yang bebas rintangan. Cocok untuk robotika yang bekerja di lingkungan yang sulit dan berbahaya. Jalur aman lebih mudah ditemukan dengan cell. Namun dalam membagi ruang menjadi cell membutuhkan lebih banyak pemrosesan, terutama jika luasnya besar atau kompleks. Implementasi kode Python menggunakan "matplotlib" untuk menampilkan sel dan rintangan. Kemudian, untuk menemukan jalur terpendek, letakkan A* di dalam sel yang telah didekomposisi.

- Analisis 2 terhadap simulasi berbagai metode perencanaan jalur dan optimasi trajektori di ROS Motion Planning, yang digunakan dalam simulasi robot pada RViz dan Gazebo di lingkungan virtual. Analisis ini mencakup tiga algoritma perencanaan jalur utama dan hasil dari beberapa metode lain berdasarkan animasi dari setiap planner, termasuk metode Path Searching dan Trajectory Optimization.
  1. Path Searching Algorithms
     - Greedy Best-First Search (GBFS): Focus pada jalur yang tampak paling optimal menuju target tanpa menjamin solusi optimal secara global. Dalam penerapan ROS, GBFS menghasilkan jalur yang cepat tetapi sering kali kurang efisien dan dapat mengalami hambatan ketika melewati area yang sulit.
     - Dijkstra: Menghitung setiap node tanpa memperhatikan arah tujuan, algoritma ini menjamin solusi terbaik. Jalan yang sangat efisien dalam hal biaya ditunjukkan oleh simulasi Djikstra pada ROS, tetapi membutuhkan waktu pemrosesan yang lebih lama, terutama untuk peta besar.

- Algoritma A* menggabungkan pendekatan optimal Dijkstra dengan heuristik yang mempercepat jalur menuju target. Pada ROS, A* menunjukkan hasil jalur yang lebih efisien dan cepat dibandingkan Dijkstra, namun kualitas jalur bergantung pada heuristik yang digunakan.

2. Advanced Path Planning Algorithms
   - Jump Point Search (JPS): Mempercepat pencarian pada grid dengan mengurangi node yang perlu dievaluasi. Dalam ROS, JPS menghasilkan jalur yang cepat pada grid-map, namun hanya efektif pada peta grid reguler.
   - D dan D Lite: Dirancang untuk lingkungan yang sebagian tidak diketahui. Dalam simulasi ROS, kedua algoritma ini menyesuaikan jalur secara dinamis saat mendapatkan informasi baru, membuatnya cocok untuk lingkungan yang berubah.
   - Hybrid A*: Algoritma ini mengatasi keterbatasan jalur grid dengan mempertimbangkan orientasi kendaraan. Pada ROS, Hybrid A* cocok untuk kendaraan dengan batasan arah, seperti mobil otonom.
   - Rapidly-exploring Random Tree (RRT) dan RRT: Algoritma sampling-based ini sangat cepat, terutama dalam skenario dengan banyak hambatan. RRT menunjukkan efektivitas dalam menjelajahi area yang luas, namun jalur yang dihasilkan sering kali tidak optimal tanpa post-processing.
   - Informed RRT: Mengoptimalkan jalur yang dihasilkan dengan fokus pada area tertentu. Hasil simulasi pada ROS menunjukkan jalur yang lebih optimal dibandingkan RRT standar.
3. Trajectory Optimization Techniques
   - Polynomial, Bezier, dan Cubic Spline**: Digunakan untuk membuat jalur yang halus. Dalam ROS, ketiga metode ini menunjukkan jalur yang mulus dan menghindari perubahan sudut tajam.
   - Dubins dan Reeds-Shepp Curves: Dirancang untuk kendaraan dengan batasan gerak seperti mobil yang tidak bisa bergerak mundur (Dubins) atau bisa bergerak maju-mundur (Reeds-Shepp). Dalam simulasi ROS, kurva ini menghasilkan jalur yang optimal sesuai dengan batasan kendaraan.
   - Local Planning and Collision Avoidance
   - Dynamic Window Approach (DWA)**: Metode ini menggunakan jendela dinamis untuk menghindari tabrakan dengan cara memprediksi gerakan. DWA pada ROS menunjukkan kemampuan robot untuk merespons hambatan dalam waktu nyata.
   - Artificial Potential Field (APF)**: Menggunakan medan potensial untuk mengarahkan robot. Hasil simulasi menunjukkan APF efektif dalam lingkungan terbuka, tetapi dapat mengalami jebakan di antara hambatan.
4. Performance Analysis Based on Animations

   Kinerja masing-masing algoritma berbeda, seperti yang ditunjukkan oleh animasi di ROS. A* menunjukkan jalur optimal secara cepat, sedangkan RRT menemukan jalur awal lebih cepat, tetapi seringkali membutuhkan perbaikan untuk mencapai jalur optimal. JPS dan Hybrid A* memberikan kombinasi terbaik untuk grid dan kendaraan berarah, sedangkan D* dan D* Lite memungkinkan penyesuaian jalur dinamis.