Nama : Eko Putra Nugraha

NIM : 1103213212

Analisis Simulasi Gazebo

- Greedy Best-First Search (GBFS)
  Algoritma GBFS beroperasi dengan prinsip mendekati target secara langsung menggunakan estimasi jarak terpendek antara posisi saat ini dan tujuan. Dalam konteks Path Searching, GBFS seringkali menghasilkan waktu pemrosesan yang cepat karena lebih memprioritaskan jarak terdekat ke tujuan. Namun, kelemahan utama GBFS adalah ketidakefisienan dalam mencari jalur optimal secara keseluruhan, sebab ia tidak selalu mempertimbangkan jarak terpendek total, melainkan hanya jarak terdekat ke tujuan dari posisi saat ini. Trajectory Optimization dalam GBFS juga cenderung kurang optimal karena mengabaikan jalur terpendek yang mungkin memerlukan sedikit belokan atau perubahan arah yang tidak ideal. Dalam hasil animasi, GBFS sering menunjukkan jalur yang "langsung" tetapi sering kali mengabaikan optimalisasi bentuk lintasan, menyebabkan jalur terlihat kurang efisien.

- Dijkstra
  Algoritma Dijkstra, di sisi lain, beroperasi dengan menjelajahi semua jalur secara bertahap untuk memastikan bahwa jalur terpendek ditemukan. Untuk Path Searching, algoritma ini menawarkan hasil optimal, meskipun waktu pemrosesannya cenderung lebih lama karena evaluasi menyeluruh di setiap titik. Algoritma Dijkstra mempertimbangkan setiap node dan secara sistematis memilih jalur dengan jarak paling pendek ke semua titik. Pada Trajectory Optimization, Dijkstra memberikan jalur yang sangat efisien dan konsisten, menghasilkan lintasan paling pendek yang mungkin. Dari segi animasi, Dijkstra menunjukkan jalur yang sistematis dan mendetail, di mana lintasan bergerak secara bertahap namun dengan hasil akhir yang paling optimal.

- A*
  Algoritma A* menggabungkan pendekatan heuristik GBFS dan optimalisasi menyeluruh dari Dijkstra. Dalam Path Searching, A* menawarkan kecepatan dan efisiensi yang lebih tinggi dibandingkan Dijkstra dengan mempertimbangkan estimasi jarak ke tujuan dan jarak yang telah ditempuh. A* memberikan hasil yang optimal dalam waktu yang relatif cepat. Trajectory Optimization dalam A* menghasilkan lintasan yang lebih mulus, hampir setara dengan Dijkstra namun dengan waktu pencarian yang lebih cepat, membuatnya menjadi pilihan populer dalam aplikasi perencanaan jalur. Hasil animasi menunjukkan bahwa A* memberikan lintasan yang lancar tanpa berbelok secara berlebihan, menciptakan jalur yang optimal dengan waktu pencarian yang efisien.

Analisis File Python

A* Planner, algoritma ini menggunakan pendekatan pencarian berbasis graf dengan bantuan fungsi heuristik untuk menemukan jalur terpendek dari titik awal ke titik tujuan di atas sebuah grid. Fungsi utama a_star melakukan proses iteratif untuk mencari jalur optimal dengan mempertimbangkan biaya aktual dari titik awal hingga node saat ini dan menambahkan estimasi biaya ke titik tujuan berdasarkan jarak Manhattan sebagai heuristiknya. Algoritma ini terus memilih node dengan nilai f terendah (gabungan antara biaya aktual dan estimasi) hingga menemukan jalur yang sesuai. Outputnya berupa jalur yang menghubungkan titik awal ke tujuan, atau jalur kosong jika tidak ditemukan rute yang memungkinkan.

Cell Decomposition, menyediakan implementasi awal dari metode decomposisi sel. Pendekatan ini membagi area yang dituju menjadi beberapa sel bebas yang dapat dilalui dengan mempertimbangkan rintangan yang ada. Fungsi cell_decomposition memecah area bebas dari rintangan menjadi beberapa sel yang saling berhubungan. Pada file ini, create_cells digunakan untuk mendefinisikan cell-cell dasar berdasarkan posisi rintangan, sementara find_path_through_cells menentukan jalur yang menghubungkan cell-cell tersebut dari titik awal ke titik tujuan. Implementasi yang sederhana ini bekerja dengan cell yang didefinisikan secara statis dan dapat dikembangkan lebih lanjut agar lebih fleksibel dalam menangani berbagai konfigurasi rintangan.

algoritma Dijkstra digunakan untuk mencari jalur terpendek pada graf berbobot dengan pendekatan yang sangat berbeda. Algoritma ini tidak menggunakan heuristik, melainkan memanfaatkan struktur data priority queue untuk selalu memilih node dengan biaya terendah pada setiap langkahnya. Dalam fungsi dijkstra, algoritma mengevaluasi tetangga dari setiap node dan memperbarui jalur jika ditemukan biaya yang lebih rendah daripada sebelumnya. Algoritma ini cocok untuk graf berbobot tanpa arah dan menghasilkan jalur optimal dengan total biaya dari titik awal ke titik tujuan, atau kembali dengan nilai "tak terhingga" jika tidak ada jalur yang memungkinkan.

Secara keseluruhan, ketiga algoritma ini masing-masing menawarkan metode yang sesuai untuk konteks yang berbeda: A* Planner efektif untuk grid dua dimensi dengan penggunaan heuristik, Cell Decomposition cocok untuk area dengan rintangan yang bisa dibagi menjadi cell-cell, dan Dijkstra efisien untuk graf berbobot dengan kebutuhan jalur terpendek tanpa memerlukan estimasi tambahan.