Nama : Irman Prayista
NIM : 1103210094
Kelas : TK-45-02

Hasil Analisis Simulasi

a. Dijkstra Planner

Algoritma Dijkstra dalam kode ini digunakan untuk mencari jalur terpendek dalam graf yang memiliki bobot. Dengan menggunakan antrean prioritas dari modul heapq, algoritma ini memeriksa node dengan biaya terendah terlebih dahulu dan memperbarui jarak terpendek ke tetangga node saat ini. Contoh graf menunjukkan algoritma ini menemukan jalur terpendek dari node 'A' ke 'D', memberikan total biaya dan urutan node yang dilalui. Dijkstra sangat berguna dalam aplikasi, seperti pemrograman graf, sistem navigasi, dan pengoptimalan jaringan.

b. A Star Planner

Algoritma A* untuk mencari jalur terpendek dalam grid yang berisi jalan bebas dan halangan. Fungsi get_neighbors menemukan tetangga yang bisa dijangkau dari node saat ini, dengan mempertimbangkan arah gerakan yang valid dan menghindari halangan. Algoritma A* mengelola dua daftar, yaitu open_list untuk node yang perlu diperiksa dan closed_list untuk node yang sudah diperiksa. Dengan menghitung biaya dari titik awal dan menambahkan estimasi jarak ke tujuan, algoritma ini dapat menemukan jalur terpendek dengan cepat. Grid berfungsi dalam lingkungan sederhana untuk mencapai tujuan dari titik awal sambil menghindari halangan.

c. Cell Decomposition

Metode *cell decomposition* untuk mencari jalur dari titik awal ke tujuan dengan mempertimbangkan rintangan. Fungsi cell_decomposition untuk membuat sel berdasarkan posisi rintangan dan kemudian mencari jalur antara titik awal dan tujuan. Fungsi tersebut hanya memberikan contoh sel dan mengembalikan jalur langsung tanpa memeriksa rintangan. Ini memberikan kerangka dasar yang dapat dikembangkan lebih lanjut untuk menghitung sel yang lebih tepat dan menemukan jalur yang sebenarnya di antara rintangan.

d. ROS Motion Planning

ros_motion_planning terdapat berbagai algoritma perencanaan jalur, seperti *Greedy Best-First Search* (GBFS), Dijkstra, dan A*. GBFS memprioritaskan jalur berdasarkan heuristik yang mengarahkan pencarian ke tujuan dengan cepat, tetapi tidak menjamin

jalur optimal. Dijkstra menjamin jalur terpendek dengan memeriksa semua kemungkinan jalur dari titik awal ke tujuan. A* menggabungkan pendekatan keduanya dengan memanfaatkan heuristik dan biaya untuk menemukan jalur yang optimal dan efisien. Tutorial ini menjelaskan cara mengimplementasikan algoritma ini dalam aplikasi nyata, serta contoh penggunaannya dalam konteks robotika.