Algoritma Djikstra : Algoritma Dijkstra adalah algoritma yang digunakan untuk menemukan jalur terpendek dari satu titik ke titik lainnya dalam graf berbobot, di mana bobot (atau jarak) antara titik-titik tersebut adalah non-negatif.

- Inisialisasi Antrian Prioritas: Menggunakan heapq untuk menyimpan node dengan prioritas biaya terendah.
- Pengulangan dan Pengecekan Node: Algoritma berjalan selama masih ada node dalam antrian, dengan memeriksa apakah node sudah dikunjungi.
- Penambahan Jalur dan Biaya: Node yang dikunjungi ditambahkan ke jalur, dan total biaya diperbarui untuk setiap node tetangga.
- Pengecekan Tujuan: Jika node tujuan tercapai, algoritma akan mengembalikan jalur dan biaya totalnya.
- Output: Menampilkan jalur terpendek dan total biaya dari titik awal ke tujuan.

Algoritma A* : Algoritma A* (A-star) adalah algoritma pencarian jalur yang digunakan untuk menemukan jalur terpendek dari satu titik awal ke titik tujuan dalam graf berbobot, dengan mempertimbangkan biaya perjalanan dan estimasi jarak ke tujuan.

- Fungsi Heuristic: Menggunakan jarak Manhattan sebagai perkiraan biaya dari node saat ini ke tujuan.
- Open Set dan f-score: Open set menyimpan node yang akan dievaluasi dengan prioritas biaya f = g + h, di mana g adalah biaya dari start ke node saat ini, dan h adalah nilai heuristic.
- Proses Pencarian: Untuk setiap node, algoritma memeriksa tetangga yang dapat diakses (tidak ada rintangan) dan memperbarui nilai g_score dan f_score.
- Rekonstruksi Jalur: Jika node tujuan tercapai, algoritma membangun jalur dari start ke goal dengan menelusuri came_from.
- Output: Menampilkan jalur yang ditemukan atau pesan bahwa tidak ada jalur.

Algoritma Cell Decomposition : Algoritma cell decomposition adalah metode dalam bidang perencanaan jalur (path planning) yang digunakan untuk memecah atau membagi ruang kerja robot menjadi bagian-bagian yang lebih kecil atau sel, sehingga memungkinkan robot merencanakan jalur dari titik awal ke titik tujuan dengan cara yang lebih sederhana.

- Inisialisasi Antrian: Menggunakan antrian untuk menyimpan cell yang akan dievaluasi. Mulai dari titik awal, menyimpan jalur yang ditempuh sejauh ini.
- Pengulangan Pencarian Jalur: Algoritma memeriksa cell saat ini dan tetangganya (atas, bawah, kiri, kanan). Jika cell tetangga berada dalam batas grid dan tidak memiliki rintangan, algoritma menambahkannya ke antrian.
- Pengecekan Tujuan: Jika mencapai cell tujuan, algoritma mengembalikan jalur yang ditempuh.
- Output: Menampilkan jalur dari titik awal ke tujuan atau pesan bahwa tidak ada jalur.

ROS Motion Planning : ROS Motion Planning adalah suatu masalah komputasional yang berfungsi untuk mencari sekuensi konfigurasi yang valid untuk menggerakan robot dari lokasi awal ke tujuan akhir. Biasanya sudah termasuk *Path Searching* dan *Trajectory Optimization.* Dalam simulasi dari github kita bisa menggunakan 2D Nav Goal untuk menentukan tujuan akhir yang kita inginkan robot untuk berjalan.

GBFS (Greedy Best-First Search): Algoritma pencarian yang memilih jalur berdasarkan heuristik terdekat ke tujuan, tetapi tidak mempertimbangkan biaya keseluruhan.

Dijkstra: Algoritma pencarian jalur terpendek yang menjamin menemukan jalur optimal dengan mempertimbangkan semua kemungkinan jalur.

A*: Menggunakan kombinasi biaya dari jalur yang telah dilalui dan estimasi biaya ke tujuan, mengoptimalkan pencarian dengan mempertimbangkan baik jarak yang telah ditempuh maupun perkiraan biaya.

JPS (Jump Point Search): Peningkatan dari A untuk grid, yang mengurangi jumlah node yang dieksplorasi dengan melompati node yang tidak relevan.

D: Algoritma dinamis yang memperbarui jalur saat lingkungan berubah, memungkinkan penyesuaian jalur secara real-time.

LPA (Lifelong Planning A): Versi Dijkstra dan A yang mampu memperbaharui rencana jalur secara efisien ketika ada perubahan dalam lingkungan.

D Lite: Versi lebih efisien dari D yang dirancang untuk memperbaharui jalur dengan lebih cepat saat perubahan kecil terjadi.

Voronoi: Menggunakan diagram Voronoi untuk menentukan jalur, menjamin jarak maksimum dari batas halangan, sehingga menciptakan jalur yang aman.

Theta*: Mengoptimalkan jalur yang dihasilkan oleh A dengan menghilangkan sudut tajam, menghasilkan jalur yang lebih halus.

Lazy Theta*: Varian Theta yang menunda proses optimasi jalur sampai jalur awal ditemukan, sehingga mempercepat pencarian.

S-Theta*: Menggabungkan ide dari Theta dan Lazy Theta, menyeimbangkan antara efisiensi pencarian dan kualitas jalur.

Hybrid A*: Menggunakan A* dengan model kendaraan untuk memperhitungkan batasan kinematik, meningkatkan keakuratan jalur untuk kendaraan nyata.

RRT (Rapidly-exploring Random Tree): Algoritma pencarian jalur yang membangun pohon pencarian secara acak, efektif dalam ruang berdimensi tinggi.

RRT*: Versi optimal dari RRT yang memperbaiki jalur yang ditemukan secara bertahap untuk mencapai solusi yang lebih baik.

Informed RRT: Menggunakan heuristik untuk membimbing pencarian RRT, mempercepat konvergensi menuju solusi optimal.

RRT-Connect: Mempercepat pencarian jalur dengan membangun dua RRT secara bersamaan yang mencoba menghubungkan satu sama lain.

ACO (Ant Colony Optimization): Menggunakan pendekatan berbasis koloni semut untuk menemukan jalur optimal dengan memanfaatkan feromon sebagai sinyal untuk jalur yang lebih baik.

GA (Genetic Algorithm): Menggunakan mekanisme evolusi untuk mencari solusi optimal dengan menggabungkan dan memodifikasi jalur yang ada.

PSO (Particle Swarm Optimization): Menggunakan kelompok partikel yang bergerak dalam ruang pencarian untuk menemukan solusi optimal, terinspirasi oleh perilaku sosial burung.

PID (Proportional-Integral-Derivative): Kontroler yang digunakan untuk mengendalikan sistem, mengatur respons berdasarkan kesalahan saat ini, akumulasi kesalahan, dan perubahan kesalahan.

LQR (Linear Quadratic Regulator): Metode optimal untuk kontrol sistem linier yang meminimalkan fungsi biaya yang melibatkan kontrol dan keadaan sistem.

DWA (Dynamic Window Approach): Pendekatan kontrol yang mempertimbangkan kecepatan dan akurasi untuk menentukan jalur yang aman bagi robot.

APF (Artificial Potential Fields): Menggunakan gaya tarik dan tolak untuk mengarahkan robot menuju tujuan sambil menghindari halangan.

RPP (Rapidly-exploring Random Polygons): Memperluas RRT untuk menangani lingkungan poligon kompleks dengan lebih efisien.

TEB (Timed Elastic Band): Mengoptimalkan jalur dengan mempertimbangkan waktu dan elastisitas, cocok untuk navigasi dinamis.

MPC (Model Predictive Control): Metode kontrol yang menggunakan model dinamis sistem untuk merencanakan kontrol optimal di masa depan berdasarkan prediksi.

Lattice: Pendekatan yang membagi ruang pencarian menjadi grid, memfasilitasi pencarian jalur yang lebih efisien dengan mempertimbangkan batasan kendaraan.