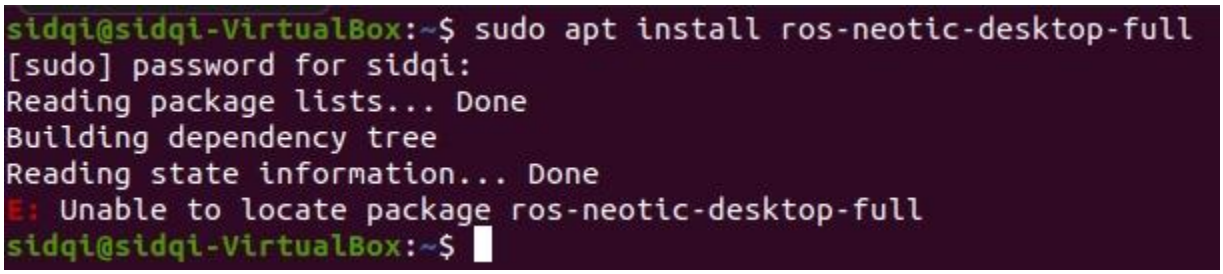
Muhammad Sidqi Nabhan
1103200179

Terdapat error ketika menginstal ros , jadi saya hanya akan menganalisis paparan tugas yang di berikan asisten




## Analisis Implementasi Algoritma

### 1. Dijkstra's Algorithm

Algoritma Dijkstra adalah salah satu metode pencarian jalur terpendek yang telah lama digunakan dalam berbagai aplikasi, termasuk perencanaan jalur untuk robot otonom. Kekuatan utama dari algoritma Dijkstra adalah kemampuannya untuk menjamin jalur terpendek, tetapi kelemahannya terletak pada kompleksitas komputasi yang tinggi ketika diterapkan pada graf yang besar, karena Dijkstra tidak memiliki heuristik.

Pada simulasi, Dijkstra akan mencoba semua kemungkinan jalur yang dapat diambil dari titik awal ke titik tujuan, menghitung total biaya untuk setiap jalur, dan akhirnya memilih jalur dengan biaya terendah. Pada grid dengan rintangan, Dijkstra membutuhkan waktu yang cukup lama untuk memproses seluruh graf, khususnya jika gridnya besar dan jumlah rintangan tinggi. Algoritma ini paling efektif digunakan ketika ada kepastian bahwa jalur yang dihasilkan harus benar-benar optimal tanpa memperhatikan efisiensi waktu komputasi. Dengan demikian, untuk robot yang bergerak di lingkungan yang cukup statis, Dijkstra akan berguna, namun kurang efisien pada lingkungan yang dinamis atau besar.

### 2. *A Algorithm**

Algoritma A* menggabungkan aspek eksplorasi dari Dijkstra dengan penambahan heuristik untuk mempercepat pencarian jalur. Algoritma ini menggunakan fungsi heuristic untuk memberikan perkiraan jarak tersisa ke tujuan, yang membantunya untuk menghindari pemeriksaan terhadap node-node yang tidak relevan. Pada lingkungan grid yang berisi rintangan, A* bekerja lebih cepat dibandingkan dengan

Dijkstra, karena ia tidak mengevaluasi semua simpul, melainkan hanya simpul-simpul yang mendekatkan robot ke arah tujuan.

A* ideal diterapkan pada perencanaan jalur robot dalam lingkungan yang dinamis, di mana waktu komputasi penting dan efisiensi diperlukan. Misalnya, dalam lingkungan dengan banyak rintangan seperti gudang yang penuh barang atau ruang kerja industri, A* dapat memberikan respons jalur yang lebih cepat sambil tetap memastikan bahwa jalur tersebut optimal atau mendekati optimal. Hal ini sangat menguntungkan dalam aplikasi seperti ROS Motion Planning, di mana algoritma A* memberikan hasil yang mendekati optimal dalam waktu yang lebih singkat dibandingkan Dijkstra.

**3. Cell Decomposition**

Metode Cell Decomposition memecah ruang pencarian menjadi bagian-bagian yang lebih kecil (cell) dan berusaha untuk mencari jalur aman yang melewati cell-cell bebas rintangan. Cell Decomposition sering digunakan pada perencanaan jalur robot di lingkungan yang luas dengan rintangan besar atau tidak beraturan. Metode ini cukup efisien karena memungkinkan pemetaan lingkungan yang kompleks menjadi bentuk grid yang lebih sederhana.

Dalam penerapannya, jika robot bergerak di dalam lingkungan yang besar, metode ini mengurangi ruang pencarian dan memungkinkan pencarian jalur yang lebih cepat. Namun, kekurangannya adalah metode ini tidak selalu menghasilkan jalur yang paling optimal, terutama jika resolusi grid terlalu kasar. Untuk aplikasi di ROS, Cell Decomposition sangat berguna dalam perencanaan jalur dasar dan bisa dikombinasikan dengan metode pencarian lain seperti Dijkstra atau A* untuk meningkatkan ketepatan jalur dalam ruang yang sudah terpecah menjadi cell.

Implementasi ROS Motion Planning yang mengintegrasikan algoritma-algoritma ini akan membantu dalam melakukan perencanaan jalur dan optimasi trajektori berdasarkan kondisi rintangan. Tutorial dari ROS pada Link yang sudah di tautkan  dapat membantu menyimulasikan algoritma dan mendapatkan gambaran lebih jelas tentang efisiensi tiap algoritma dalam lingkungan dinamis.