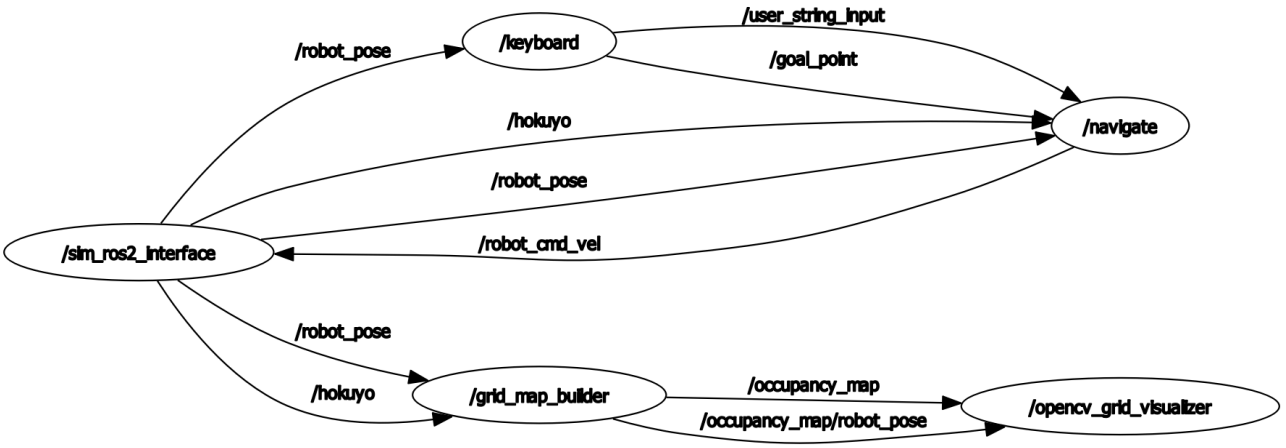
Laporan Praktikum 4 Robotika

ROS2 & CoppeliaSim

Immanuel - 2006463162

Pradipta Davi Valendra - 2006462664

1. **Blok diagram node**



2. **GitHub Link:**
   https://github.com/Hzzkygcs/Robotik-Ros2/tree/tugas-praktikum-ros2
3. **Navigation Package**
   a. **Navigate**

```
[INFO] [1715585201.881976900] [navigate]: Goal is reachable without obstacle around
[INFO] [1715585297.795933700] [navigate]: Goal is reachable without obstacle around
[INFO] [1715585362.081438200] [navigate]: Goal is reachable without obstacle around
[INFO] [1715585377.089834800] [navigate]: Goal is reachable without obstacle around
```

Subscribes:

- Hokuyo (Laser scan)
- Robot pose
- Goal point

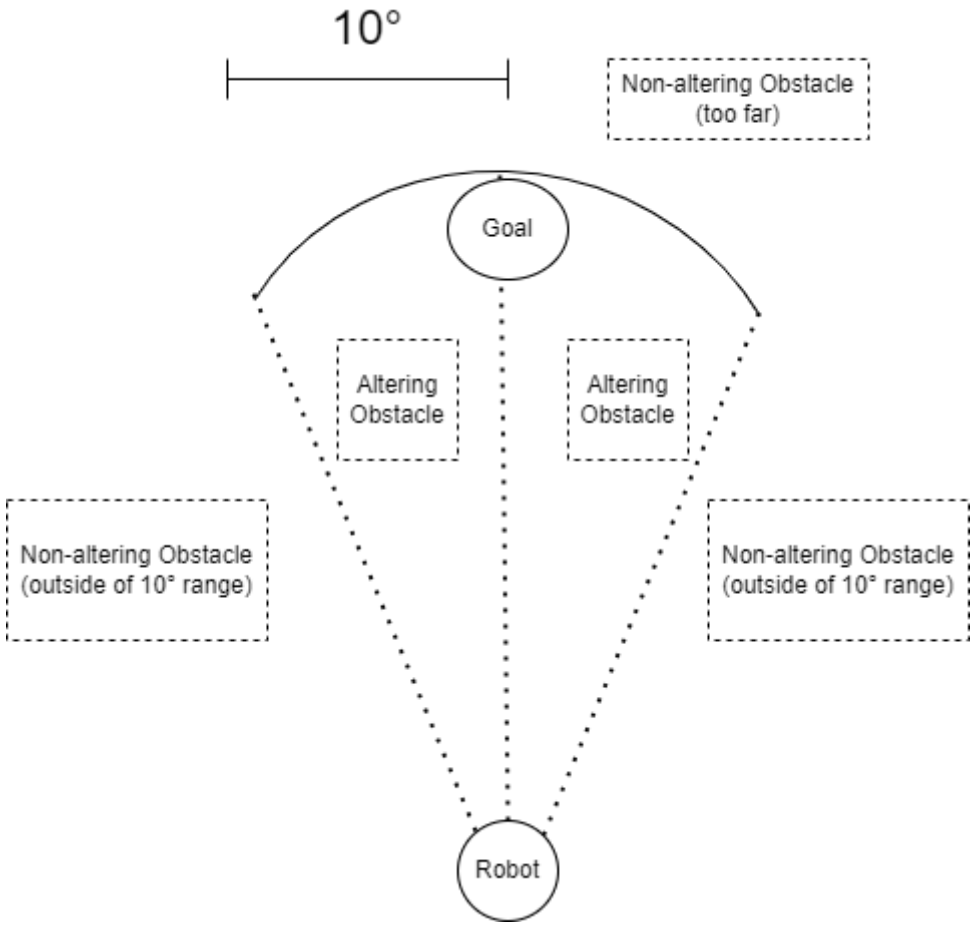
Publishes:

- Robot cmd vel

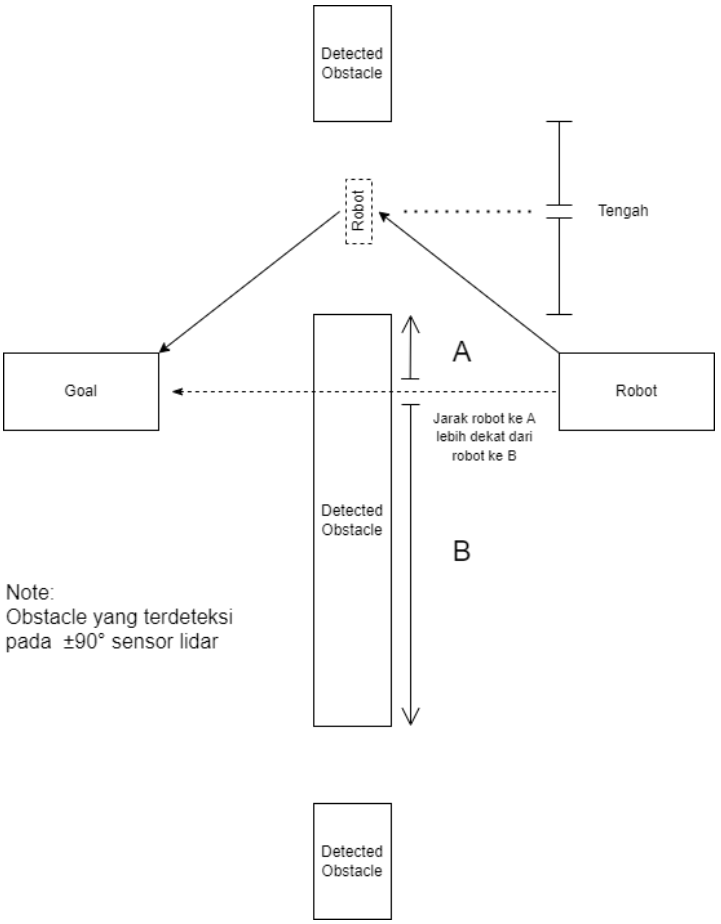
Node Navigate memiliki fungsi utama untuk menggerakan robot. Node ini menerima data dari beberapa topik, yaitu 'goal point' (sebagai tujuan), 'robot pose' (untuk mengetahui posisi robot saat ini), dan 'hokuyo' (sebagai sensor LIDAR/raycast penghindar tembok). Setelah memproses data tersebut, node ini akan mempublikasikan perintah pergerakan robot ke topik 'robot cmd vel', yakni besar kecepatan bergerak dan kecepatan angular dari robot kepada CoppeliaSim.

Algoritma:

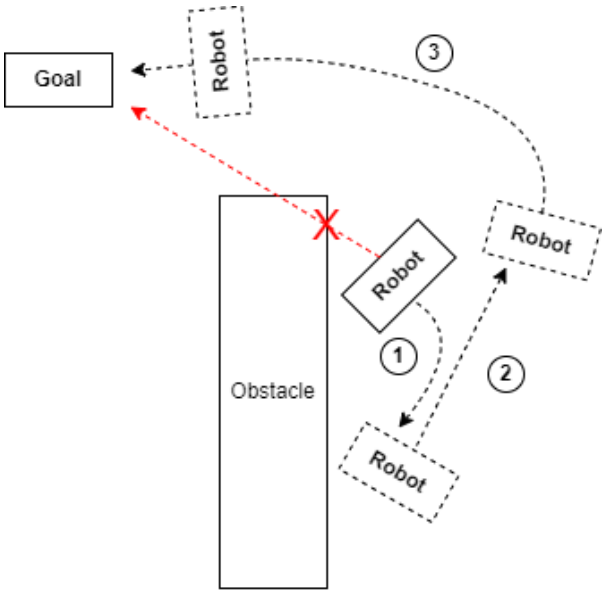
- Pergerakan ke goal point
  Jika tidak terdapat sebuah rintangan antara goal-point dan robot, maka robot akan secara langsung bergerak menuju goal point tanpa ada algoritma khusus.
  Jika terdapat sebuah rintangan yang berada pada ±10° disekitar goal-point dan rintangan tersebut berada ditengah-tengah goal-point dan robot seperti yang dapat dilihat pada Gambar 1 , maka robot akan mencoba jalan kepada goal point dengan gabungan algoritma obstacle avoidance.
- Obstacle avoidance
  Menggunakan topic dari hokuyo (Laser Scan), robot akan berusaha menghindari *obstacle* seperti pada Gambar 2. jika salah satu raycast robot terlalu dekat sebuah rintangan (sekitar 0.3 unit), robot akan memundurkan diri, kemudian maju menjauhi tembok, dan kemudian kembali maju menuju goal-point, seperti yang dapat dilihat pada Gambar 3.



Gambar 1. contoh *altering-obstacle*, yakni *obstacle* yang dapat memengaruhi algoritma Navigation untuk beralih dari *simple algorithm* menjadi yang lebih kompleks

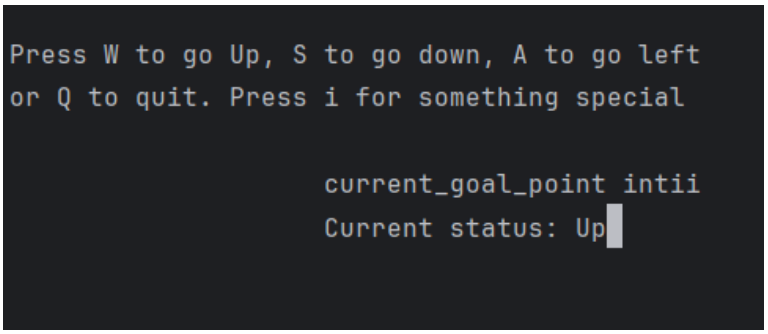


Gambar 2. Algoritma *obstacle avoidance* yang akan mencari ruang kosong (non-obstacle) terdekat untuk menuju goal



Gambar 3. Algoritma mundur untuk mundur dan menghindari obstacle

b. Keyboard



Subscribes:

- Robot pose

Publishes:

- Goal point

Node Keyboard mengendalikan pergerakan robot melalui input keyboard pengguna. Node ini mempublikasikan 'goal point' (tujuan robot) berdasarkan input pengguna dan posisi robot saat ini yang diterima dari topik 'robot pose'.

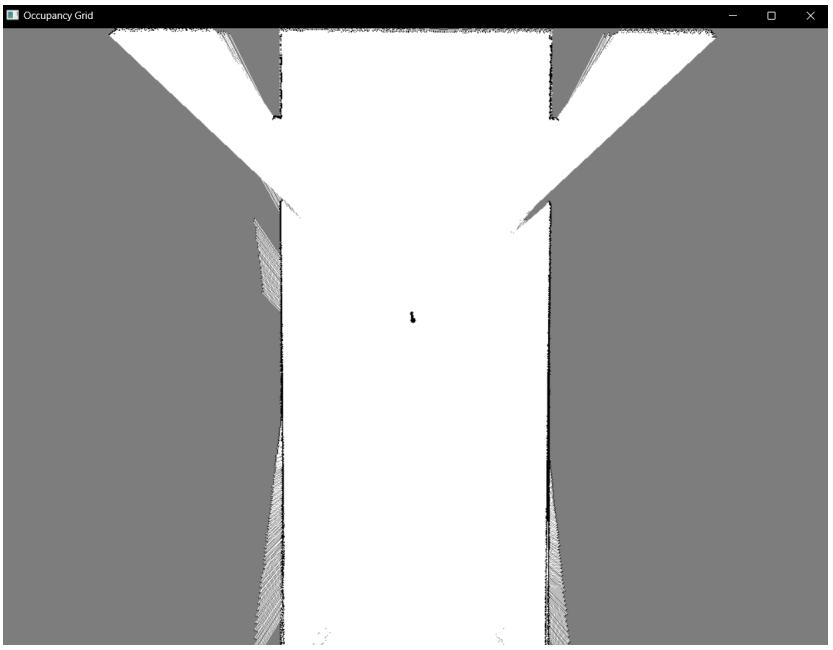
Algoritma:

- Node mendengarkan input user pada keyboard setiap 0.2 detik
- Input 'up' (W) akan menambah axis Y kepada goal point saat ini.
- Input 'down' (S) akan mengurangi axis Y kepada goal point saat ini.
- Input 'left' (A) akan mengurangi axis X kepada goal point saat ini.
- Input 'right' (D) akan menambah axis X pada goal point saat ini.
- Input 'manual input' (i) akan mengizinkan pengguna untuk menulis “x,y” sehingga robot akan bergerak ke koordinat (x,y) tersebut, relatif terhadap posisi robot saat ini.
- Input 'quit' (Q) akan menghentikan program keyboard ini.
- Jika tidak terdapat input selama 0.2 detik maka goal point akan berubah sesuai dengan posisi robot untuk berhenti.

Catatan:

- Program keyboard hanya akan menerima input setelah mendapatkan informasi posisi robot dari CoppeliaSim, sehingga perlu menjalankan simulasi pada CoppeliaSim terlebih dahulu

4. Visualization Package
    a. grid_visualizer



Subscribes:

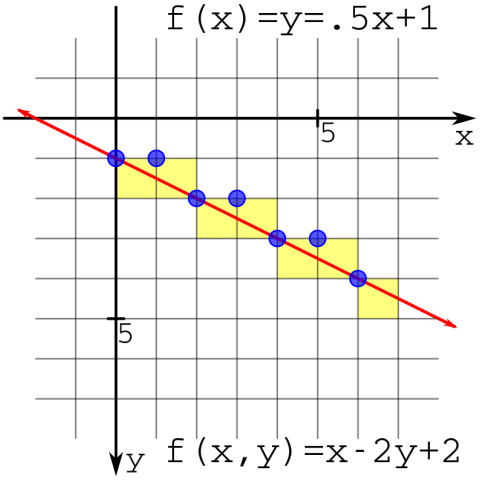
- Occupancy map
- Robot pose (From Occupancy Map)

Node grid visualizer menampilkan visualisasi area yang dijelajahi robot menggunakan GUI OpenCV. Node ini menerima input dari topik 'robot pose', yakni orientasi robot dan posisi robot (posisi pada koordinat peta, bukan koordinat yang sebenarnya), serta 'occupancy map' (peta eksplorasi). Peta tersebut ditampilkan dengan warna putih (jalur yang sudah dilihat dan memiliki probabilitas tinggi tidak ada *obstacle*), hitam (probabilitas tinggi yang diduga *obstacle*), dan abu-abu (area belum dijelajahi).

Algoritma:

- Node grid visualizer membuka window baru sebagai hasil visualisasi grafik dari data yang didapatkan dari topic occupancy grid.
- Setiap value < threshold dari occupancy grid akan dipetakan menjadi pixel putih yang menandakan area yang sudah dijelajah dan memiliki probabilitas terdapat *obstacle* yang *rendah*.
- Setiap value -1 dari occupancy grid akan dipetakan menjadi pixel abu-abu yang menandakan area yang belum dijelajah.
- Setiap value > threshold dari occupancy grid akan dipetakan menjadi pixel hitam yang menandakan terdapat probabilitas tinggi berupa tembok sehingga area tersebut tidak dapat dilewati.
- Threshold yang dipilih pada percobaan ini adalah 30

5. Mapping Package
    a. Mapping



Subscribes:
- Robot pose
- Hokuyo

Publishes:
- Occupancy Map
- Robot Pose (from Occupancy Map)

Node mapping memiliki tujuan untuk memproses data backend dari CoppeliaSim, yaitu data dari sensor-sensor robot (topik 'hokuyo') dan posisi robot (topik 'robot pose'), agar dapat membentuk gambaran peta dari ruangan dan kemudian divisualisasikan oleh node 'grid visualizer'. Data dari topik 'hokuyo' berupa hasil pemindaian laser (LIDAR) yang menggambarkan lingkungan sekitar robot, sedangkan data dari topik 'robot pose' menginformasikan orientasi dan posisi robot di dalam lingkungan simulasi CoppeliaSim.

Algoritma:
- Node mapping menggunakan data dari topic hokuyo (laser scan) dan memetakan setiap area yang terdeteksi oleh raycast dan petakan kepada koordinat x,y yang sesuai pada grid occupancy.
- Untuk setiap garis raycast individu, disimpan koordinat x,y ujung garis tersebut dan angle garis tersebut relatif dengan peta keseluruhan dan orientasi robot.
- Dengan algoritma bresenham, untuk setiap garis-garis *raycast* akan dihitung semua koordinat x,y ditengah-tengah kedua ujung *raycast* tersebut dan dipetakan kepada topic grid occupancy sebagai area dengan probabilitas rendah terdapat *obstacle* (white area). Sementara pada ujung *raycast* yang menyentuh *obstacle* itu akan dipetakan pada GridOccupancy sebagai probabilitas tinggi terdapat *obstacle* (black area).

Improvisasi Algoritma:

- Merubah occupancy grid map menjadi multi-layer. Multilayer ini memiliki beberapa lapisan, di mana pada lapisan pertama merupakan OccupancyGrid seperti biasanya, sedangkan lapisan berikutnya memiliki jumlah *pixels* yang lebih sedikit karena setiap 10x10 *pixels* pada *layer* saat ini akan digabung menjadi 1 buah *pixel* pada *layer* berikutnya. Meskipun hal ini menghasilkan OccupancyGrid dengan resolusi yang lebih rendah pada layer tersebut, hal ini dapat meningkatkan performa A*. Pendekatan ini menyeimbangkan antara kebutuhan untuk memperoleh performa A* yang terjangkau tetapi juga tetap memenuhi kebutuhan untuk menjalankan A* secara presisi.