4TM00 Robot Motion Planning and Control
Assignment 1 - Safe Robot Teleoperation & Reactive Robot Navigation

Group 5
Devan Sedmak[1](2234238) and Matteo Petris[2](2234203)

***Abstract*— In this assignment we develop two safety mechanisms for the RoboCyl robot: a teleoperation system that modifies user commands to avoid collisions and a reactive navigation algorithm for obstacle avoidance using ROS2. Both approaches were evaluated in dynamic environments, with limitations and improvements discussed.**

I. Safe Robot Teleoperation

A. Introduction

In the first part of the assignment, we will design a ROS node that receives original teleoperation control commands from a user as input and minimally modifies these commands using the robot's scan measurements and pose to ensure collision-free teleoperation around obstacles. In other words, if the requested control command is safe, the robot should execute it as is; otherwise, the command should be minimally modified or set to zero in the worst-case scenario to avoid collisions.

B. Design

The robot in use is RoboCyl. It is a fully-actuated velocity-controlled robot with a circular body shape of radius $\rho = 0.2\,\text{m}$. The model is described are as follows:

- State Variable

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{(2D Position)}$$

- Equation of Motion

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = \mathbf{u} \quad \text{(Velocity Control)}$$

- Body Shape $\quad (\rho \geq 0 : \text{Body Radius})$

$$B(\mathbf{x}, \rho) = \left\{ \mathbf{y} \in \mathbb{R}^2 \;\middle|\; \|\mathbf{y} - \mathbf{x}\| \leq \rho \right\}$$

The robot is inside a closed rectangular environment bounded by walls. Inside this environment there are some cilinders. The obstalces are the walls and the cilinders. The robot can only detect some points on the surface of these obstacles. These points will be called point onstacles. Using the scan, the robot can detect the point obstacles around it with a minumum range of 0.2 m and a maximum range of 3.5 m, with an angular resolution of 1.0 degreees. The point obstacles and the distance to a point obstacle with respect to the robot's position x are defined as follows:

- Point obstacles $\quad (p_i \in \mathbb{R}^2$: Point obstacle position)

$$\mathbf{P} = (p_1, \ldots, p_m) \in \mathbb{R}^{m \times 2}$$

where m is the number of detected points by the scan.
- Distance to point obstacle p_i

$$d_i(\mathbf{x}) = \|\mathbf{x} - p_i\|$$

- Distance to point obstacle p_i gradient

$$\nabla d_i(\mathbf{x}) = \frac{\mathbf{x} - p_i}{\|\mathbf{x} - p_i\|}$$

Firstly, we need to identify the Local Nearest Points with respect to the robot's position. We can use the following algorithm:

i) (Initialize) Start with the empty nearest neighbor set N_P,

$$N_p \leftarrow \varnothing$$

ii) (Identify) Find the closest point of the obstacle set to the robot's position,

$$\bar{\mathbf{p}} = \arg\min_{\mathbf{p} \in \mathbf{P}} \|\mathbf{p} - \mathbf{x}\|$$

iii) (Update) Add the nearest point to the nearest neighbor set and eliminate all obstacle points on the other side of the hyperplane,

$$N_P \leftarrow N_P \cup \{\bar{\mathbf{p}}\}$$
$$\mathbf{P} \leftarrow \{\mathbf{p} \in \mathbf{P} \mid (\mathbf{p} - \mathbf{x})^T (\bar{\mathbf{p}} - \mathbf{x}) \leq 0\}$$

iv) (Terminate) If $\mathbf{P} \neq \varnothing$, then go to step ii. Otherwise, terminate with the nearest neighbor set N_P.

The result of the alghoritm can be seen is the Fig. 1.

The robot needs to be repulsed by the obstacles in a way that it does not collide with them. To do that, we will directly design a repulsive field. If a point $p_i \in N_P$ is distant

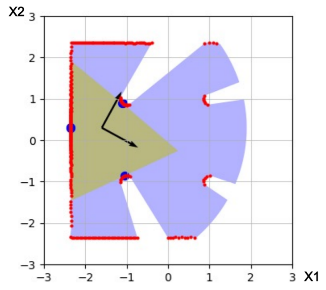

Fig. 1. The blue dots represent the local nearest points

from the robot's position for more than 2ρ, then the robot should execute every command inputed by the user without modifying it. Otherwise, if it is distant between ρ and 2ρ it should repulse the robot linearly with respect to $d_i(x)$. The maximum repulsion should be when the robot's center is distant ρ from the point p_i. The maximum module of the field should be equal to the maximum module of the user's velocity. The user can input a maximum of 1.0 m/s velocity in the x_1 and in x_2 direction. This means that the maximum module of the velocity is $\sqrt{1.0^2 + 1.0^2} = \sqrt{2} \approx 1.4$. In this way even if the user inputs the maximum velocity when the robot is close to the obstacle, the field will repulse it and the robot will not crash. Formalizing this design, the module of the repulsive field U_i should follow the following rule based on the distances from the center of the robot a point $p_i \in N_P$:

$$||U_i(x)|| = \begin{cases} 1.4 & d_i(x) \leq \rho \\ -\frac{1.4}{\rho} d_i(x) + 2.8 & \rho < d_i(x) < 2\rho \\ 0 & d_i(x) \geq 2\rho \end{cases}$$

The graph of the function can be seen in the Fig. 2.

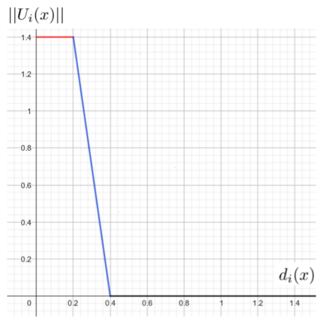

Fig. 2. Module of $U_i(x)$ respect to the distance $d_i(x)$ form the point p_i

The repulsive field $U_i(x)$ will have the module as outlined before and the direction given by the gradient of the distance from the point p_i to the robot's center. So we have:

$$U_i(x) = \nabla d_i(x) ||U_i(x)||$$

The total repulsive field U will now be the sum of all these $U_i(x)$:

$$U(x) = \sum_{i=1}^{|N_P|} U_i(x)$$

The $U(x)$ that we have found refers to the space x_1, x_2 of the environment. But the velocity that is given to the robot must be oriented with x'_1, x'_2 referring to the orientation of the robot in the environment. We can find the input $U'(x)$ to send to the robot, knowing α the orientation of the robot with respect to x_1, x_2 and using the following formula:

$$U'(x) = \begin{bmatrix} cos(\alpha) & sin(\alpha) \\ -sin(\alpha) & cos(\alpha) \end{bmatrix} U(x)$$

Now we just need to modify the user input in the following way:

$$u = u_{user} + U'(x)$$

In this way the user input is modified only when it is necessary to mantain the teleoperation safe.

C. Implementation

We implemented our design in ROS2. We designed a ROS node, that subscribes to the $scan, pose$ and $cmd\_vel\_key\_raw$ topics. This node, called $safe\_twist\_teleop$, then modifies the input from $cmd\_vel\_key\_raw$, based on $scan$ and $pose$. In the end the new velocity is published to the topic cmd_vel_key. The RQT graph can be seen in the Fig. 3.

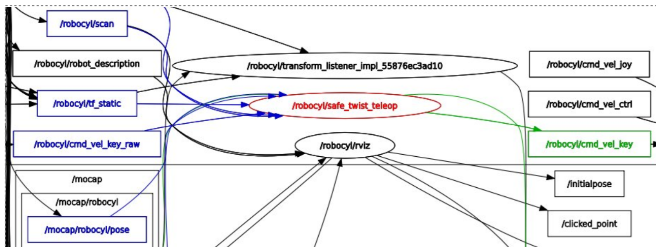

Fig. 3. RQT graph of the ROS2 implementation

D. Limitations

The robot, implemented based on our design, exhibits some limitations. Specifically, when the user input sets the robot to full speed (1.0 m/s) and it approaches a corner, the robot becomes stuck at that point.
Another limitation of our design is that it does not account for the robot's velocity and direction. Instead, it focuses solely on the robot's position relative to the nearest obstacles.

E. Conclusion

This simple design allows the robot to avoid collisions with obstacles. However, as previously mentioned, this approach has some limitations. A more refined design could retain the same simplicity while also considering the robot's velocity and direction. With an improved approach, it would be possible to adjust the user's commands minimally, enabling the robot to avoid obstacles more effectively.

II. Safe Reative Navigation

A. Introduction

In the second part of the assignment, we will design a ROS node that receives a goal position and navigates towards it while avoiding sensed obstacles. This will be achieved using the robot's scan measurements and pose. The objective is to use the instantaneous scan measurements as effectively as possible to move closer to the goal position without collisions.

B. Design

The robot in use is RoboCyl. It is a fully-actuated velocity-controlled robot with a circular body shape of radius $\rho = 0.2\,\mathrm{m}$. The model is described are as follows:

- State Variable

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{(2D Position)}$$

- Equation of Motion

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = \mathbf{u} \quad \text{(Velocity Control)}$$

- Body Shape $\quad (\rho \geq 0 : \text{Body Radius})$

$$B(\mathbf{x}, \rho) = \left\{ \mathbf{y} \in \mathbb{R}^2 \;\middle|\; \|\mathbf{y} - \mathbf{x}\| \leq \rho \right\}$$

The robot is inside a closed rectangular environment bounded by walls. Inside this environment there are some cilinders as can be seen in the Fig. 4

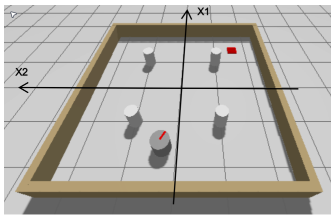

Fig. 4. The robot inside the environment. The red square is the goal

The obstalces are the walls and the cilinders. The robot can only detect some points on the surface of these obstacles. These points will be called point onstacles. Using the scan, the robot can detect the point obstacles around it with a minumum range of 0.2 m and a maximum range of 3.5 m, with an angular resolution of 1.0 degreees. The point obstacles and the distance to a point obstacle with respect to the robot's position x are defined as follows:

- Point obstacles $\quad (p_i \in \mathbb{R}^2$: Point obstacle position)

$$\mathbf{P} = (p_1, \ldots, p_m) \in \mathbb{R}^{m \times 2}$$

- Distance to point obstacle p_i

$$d_i(\mathbf{x}) = \|\mathbf{x} - p_i\|^2$$

- Distance t point obstacle p_i gradient

$$\nabla d_i(\mathbf{x}) = 2(\mathbf{x} - p_i)$$

We also have a goal $x_{goal} \in \mathbb{R}^2$ where the robot needs to go. The challenge is that a direct path to the goal may be obstructed by an obstacle, requiring the robot to navigate around it. We can solve this problem by constructing the Local Safe corridor around the robot's position. To do that we firtsly need to find the Local Nearest Points N_P around the robot's position as we did before in the Safe Robot Teleoperation part.

Now we can define the Local Safe Corridor:

$$SC_P(x) = \left\{ \mathbf{y} \in \mathbb{R}^2 \mid (\mathbf{y} - \mathbf{x})^T (\mathbf{p} - \mathbf{x}) \leq 0 \,\forall\, \mathbf{p} \in N_P(c) \right\}$$

$SC_P(x)$ is constructed using the intersection of all half-planes that pass through the N_P and are perpendicular to the distance vector.

After that we need to define the Local Free Space, defined as:

$$LF_P(x) = \left\{ \mathbf{y} \in \mathbb{R}^2 \mid B(\mathbf{y}, \rho) \subseteq SC_P(x) \right\}$$

Firstly, we construct N'_P, which is obtained by shifting the points in N_P by a distance of ρ along the direction of the distance vector. The $LF_P(x)$ is then defined as the intersection of all half-planes that pass through the points in N'_P and are perpendicular to the distance vector.

Now if the goal x_{goal} is inside the Local Free Space, we will use directly the goal. Otherwise we will project the goal x_{goal} on the boundary of the Local Safe Corridor. This means we need to find the point on the boundary that minimizes the distance from the goal to the Local Free Space. So we now have a new goal that changes with respect to where the robot is located.

$$x^* = \begin{cases} x_{goal} & \text{if } x_{goal} \in LF(x) \\ x'_{goal} & \text{if } x_{goal} \notin LF(x) \end{cases}$$

The plot of the Safe Local Corridor, the goal and the projected goal can be seen in Fig. 5.

The idea is that the goal should be the center of an attractive field, so where the field is equal to 0. The obstacles should be the centers of a repulsive field, so where the filed is equal to 0. The attractive field is constructed using the gradient of the attractive goal potential described by Squared Euclidean Distance. So we have:

$$V_{\text{att}}(x) = ||x - x^*||^2$$

$$\nabla V_{\text{att}}(x) = 2(x - x^*)$$

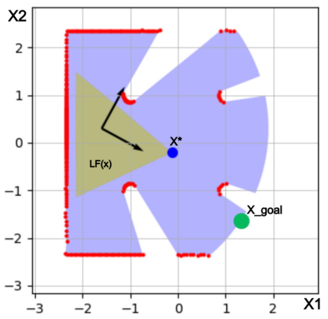

Fig. 5. Plot of the Local Free Space, the goal and the projected goal

The repulsive field is constructed using the gradient of the Bounded repulsive potential. The repulsive potential and repulsive gradient are respectevely:

$$V_{\text{rep}}(x) = \frac{1}{\prod_{i=1}^{m} s_{k_i}(d_i(x))}$$

$$\nabla V_{\text{rep}}(x) = -V_{\text{rep}}(x) \sum_{1=1} m \frac{s'_{k_i}(d_i(x))}{s_{k_i}(d_i(x))} \nabla d_i(x)$$

where $s_k(x)$ is the exponential sigmoid, so we have:

$$s_k(x) = tanh(kx)$$

$$s_k(x)' = k(1 - tanh(kx)^2)$$

After some experimentation we choose the threshold decay rate $k = 3$. We combine the repulsive and attractive potencial using the Multiplicative Navigation Potencial:

$$V(x) = V_{\text{att}}(x) * V_{\text{rep}}(x)$$

$$\nabla V(x) = \nabla V_{\text{att}}(x) * V_{\text{rep}}(x) + V_{\text{att}}(x) * \nabla V_{\text{rep}}(x)$$

The Multiplicative Navigation Gradient field will look like in Fig. 6.

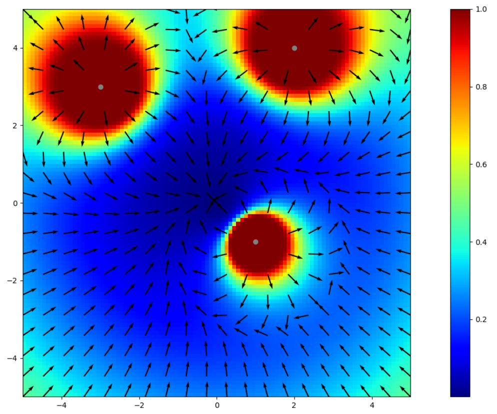

Fig. 6. The red zone is the repulsive field of the obstacles. The blue zone in the center is the attractive field of the projection of the goal

Now we need to transform ∇V in $\nabla V'$, referring to the orientation of the robot in the environment, as we did in the Safe Robot Teleoperation. Finally, we must choose an appropriate control gain g and set the control input u equal to:

$$u = -g \nabla V'$$

After experimentation, we selected $g = 0.1$

C. Implementation

We implemented our design in ROS2. We designed a ROS node, that subscribes to the $scan, pose$ and $goal\_pose$ topics. This node, called $safe\_reactive\_navigation$ sets the output, based on $scan$, $pose$ and $goal\_pose$. In the end the velocity is published to the topic cmd_vel_ctrl. In this case the $twist\_teleop\_key\_2D$ enters directly in the $twist\_mux$ node. Since the keyboard has the priority, the user can still teleoperate the robot. The RQT graph can be seen in the Fig. 7.

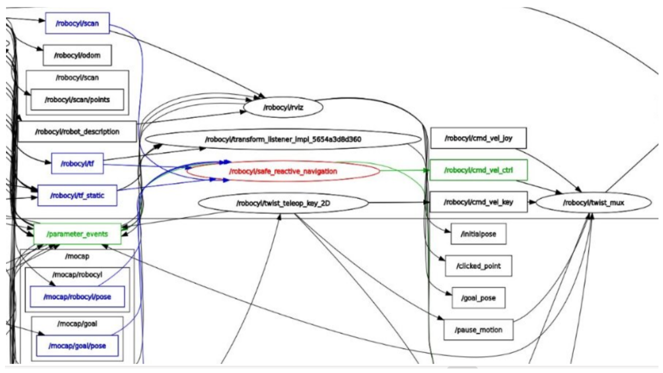

Fig. 7. RQT graph of the ROS2 implementation

D. Limitations

The robot implemented based on our design exhibits certain limitations. Specifically, when the robot approaches an obstacle in a direction closely aligned with the distance vector to the obstacle, it tends to become stuck for a short period of time.

Another limitation is that when the robot is far from both obstacles and the goal, its movement becomes unnecessarily slow.

E. Conclusion

This simple design enables the robot to avoid collisions with obstacles while reaching its goal. However, as noted earlier, this approach has some limitations. An alternative approach could involve following the level curves of $V(x) = V_{\text{att}}(x) * V_{\text{rep}}(x)$, allowing the robot to reach its goal without getting stuck near obstacles. Additionally, it would be beneficial to consider that when the robot is far from both obstacles and the goal, it could move at full velocity to improve efficiency

References

[1] Ö. Arslan, Robot Motion Planning and Control (4TM00) - Course material, Eindhoven University of Technology, 2024.