Robótica Cognitiva – 2022/2023

Autonomous Mobile Robot for Desinfection

Midterm Submission

Duarte Cruz 2017264087
Gonçalo Arsénio 2017246034
Pedro Gomes 2018298280

1 Introdução

Para este projeto foi nos proposto a realização de um robô móvel autónomo para desinfeção, ou seja, um robô que navegue autonomamente por um mapa, construído *a priori*, dos espaços em que vai efetuar a desinfeção, tendo de ser capaz de navegar de forma segura perante a presença de obstáculos estáticos e dinâmicos. Quando este chega a uma sala que tem de desinfetar, pré-definida pelo utilizador, este faz *scan* do QR code que se encontra na entrada da sala, que vai conter informações tais como, a potência da lâmpada, a energia necessária para eliminar os vírus e onde o robô deve parar em cada sala, para assegurar que a desinfeção é feita corretamente.

2 Backround

2.1 Bringup_explorer

Permite ao robô construir autónomamente um mapa do ambiente em que se encontra, ao mesmo tempo é corrido o *explorer_node* que vai fazer a leitura dos códigos QR.

slam_gmapping

O nó *gmapping* recebe informação do *sensor_msgs/LaserScan* através de mensagens e constrói um mapa (nav_msgs/OccupancyGrid). O mapa é extraído através de um tópico ROS.

explore

Quando o nó está em execução, o robô explorará o ambiente em seu redor até que nenhuma fronteira seja encontrada. Vai enviar comandos de movimento para o nó *move_base* através do tópico **/move_base/goal** e vai subscrever o tópico **/move_base/status**.

move_base

Este nó permite fazer o controlo da navegação da plataforma móvel. Este é o nó que subscreve ao tópico **/move_base/goal**, que indica o objetivo que robô deve atingir no mundo, envia os comandos de velocidade da plataforma e fornece informações sobre *status* das metas que são enviadas para a ação move_base através do tópico **/move_base/status**.

gazebo

O nó **gazebo** envolve o simulador. O Gazebo simula um mundo, tal como definido num ficheiro .world. Este contém tudo sobre o mundo no gazebo, desde obstáculos, a robôs e outros objetos.

robot_state_publisher

O **robot state publisher** subscreve ao tópico **joint_states**, com informação da posição das juntas, e publica em tf a *pose* 3D de cada *link*.

explorer_node

Desenvolvemos um nó ROS que chamamos *explorer_node*. É suposto que ele a partir dos dados que recebe dos tópicos aos quais está subscrito tenha capacidade para guardar os pontos dos códigos QR numa base de dados.

2.2 Bringup

A plataforma efetua a desinfeção com base nas instruções obtidas através dos códigos QR.

map_server

Nó responsável por disponibilizar o mapa ao nó **move_base**.

amcl

Este nó é responsável pela localização da plataforma móvel no mapa. É um sistema de localização probabilística que implementa o algoritmo Adaptativo de Monte Carlo. O nó **amcl** lê um mapa, leituras de Lidar e mensagens **tf** para estimar a pose da plataforma móvel.

barcode_reader_node

Subscreve o tópico **/camera/rgb/image_raw**, com as imagens para fazer o *scan* do código QR, e publica a informação desses códigos detetados.

guide_node

Desenvolvemos um nó ROS que chamamos *guide_node*. É suposto que ele a partir dos dados que recebe dos tópicos **/barcode** e **/move/base/status** simule a desinfeção.

3 Explorer Node

No **explorer_node.cpp** está contido o código que nos vai permitir usar a câmara do robô para detetar *keypoints* nos códigos QR, para que com esses ele calcule o ponto médio e assim consiga aplicar um transformada que vai indicar o ponto à frente desse mesmo código. Todos os pontos dos códigos encontrados durante o mapeamento são guardados numa base de dados e mais tarde utilizados pelo *guide node*.

3.1 barcodeCallback()

Esta função vai receber mensagens *string* do tópico subscrito */barcode*, que contêm a leitura que é feita do código. É efetuada a leitura da primeira linha da mensagem do código QR, que vai conter o id da sala.

3.2 cameraCallback()

Vai subscrever o tópico */camera/rgb/image_raw/compressed*, convertendo a mensagem que recebe como *input* para uma imagem. Dentro desta função é usada uma *flag* para o algorítmo saber se está algum código novo a ser lido ou não, em caso negativo simplesmente mostra o *frame* da imagem, em caso afirmativo através do *SimpleBlobDetector* vão ser calculados os *keypoints* desse *frame*, que vão ser guardados e mostrados na imagem.

3.3 pointcloudCallback()

Vai ter como parâmetro de entrada a mensagem que recebe do tópico */camera/depth/points*, e fazendo uso dos *keypoints* detetados anteriormente e da *pointcloud* vai ser calculado um ponto médio do código QR, é então chamada a função *calculatePoseWrtMap()* que vai fazer o cálculo da transformação de forma a saber qual o ponto que plataforma tem de estar para fazer a leitura do código QR. Obtendo então este ponto e o id da sala, do respetivo código, obtido através função *barcodeCallback()*, essa informação é guardada num ficheiro. Para garantir que no ficheiro apenas é guardada uma vez a informação de cada sala faz-se uso da função *emplace()*.

4 Guide Node

Tendo então os pontos objetivo que a plataforma tem de atingir de forma a conseguir ler os códigos QR, a plataforma vai conseguir extrair as informações necessárias para desinfetar cada sala. Com esta informação o robô é capaz de simular a desinfeção com sucesso.

Este ficheiro contém 3 funções: *barcodeCallback()*, *moveToQRCode()* e *moveTo()* que vão definir o comportamento do nosso robô.

4.1 moveToQRCode()

A função *moveToQRCode()*, tal como o nome indica, manda pontos para a platarforma, para que este se dirija a um código QR e assim consiga fazer a sua leitura, esta faz uso do *MoveBaseClient* que é utilizado para podermos enviar

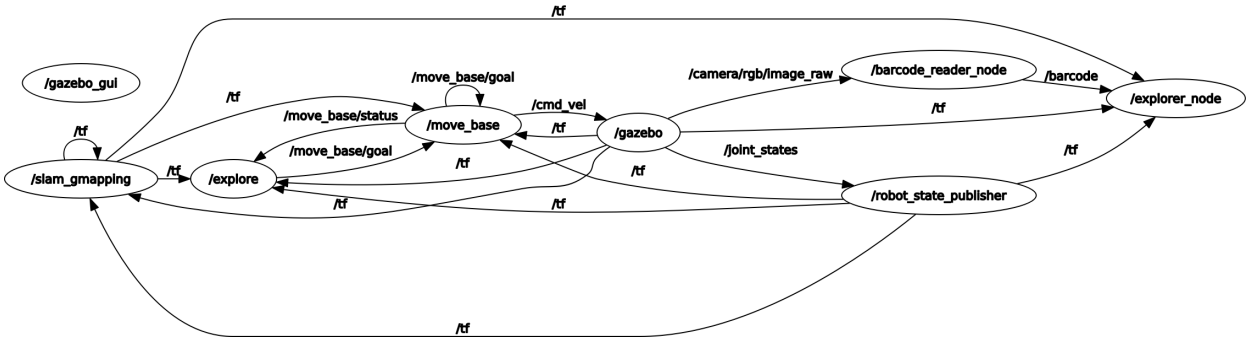

Figure 1: Diagrama dos nós ROS presentes num ambiente de simulação - **bringup_explorer.launch**

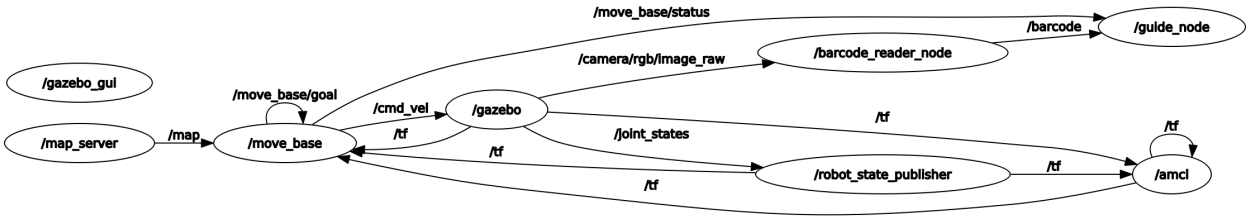

Figure 2: Diagrama dos nós ROS presentes num ambiente de simulação - **bringup.launch**

comandos de posição para o *move_base* e receber um *callback* de objetivo atingido. Vai ter como input os pontos guardados na base de dados *qr-code_database*, construída no *explorer node*. Assim que o *goal* é atingido vai ativar uma *flag* que vai ativar a leitura do código.

4.2 barcodeCallback()

Esta função vai receber mensagens *string* do tópico subscrito */barcode*, que contêm a leitura que é feita do código. Através desta leitura é retirada toda a informação que é contida no código, para conseguir simular a desinfeção.

Cada sala tem um código único com os valores necessários para a energia e potência da luz UV, que vai ser diferente consoante o tamanho das salas, e os *waypoints* a ser atingidos, onde a plataforma vai ativar a lâmpada UV durante um determinado tempo. Estes pontos vão ser os parâmetros de entrada da função *moveTo()*;

4.3 moveTo()

O *moveTo()* apresenta um algorítmo semelhante ao *moveToQRCode()*, fazendo também uso do *MoveBaseClient* para enviar os *waypoints* para a plataforma e quando recebe a informação que esse mesmo ponto é atingido simula o *turn on/off* da luz UV.

Quando os *waypoints* forem todos percorridos o robô dá a desinfeção como terminada e desloca-se ao código seguinte.

5 Simulação

O algorítmo foi testado fazendo uso do mapa 1r5, da pasta *rc_simul_worlds* fornecida pelo Professor.

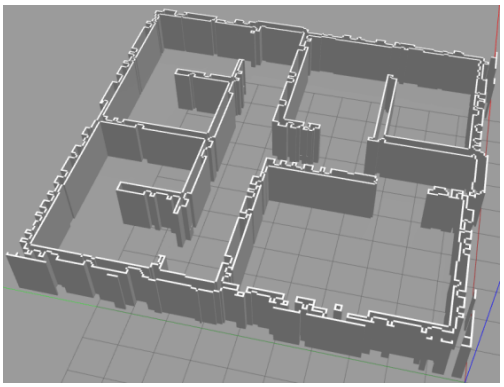

Figure 3: Mapa gerado no gazebo

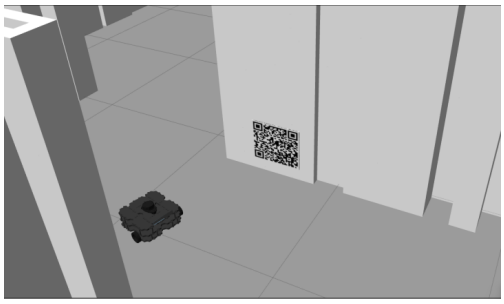

Figure 4: QR code gerado no gazebo

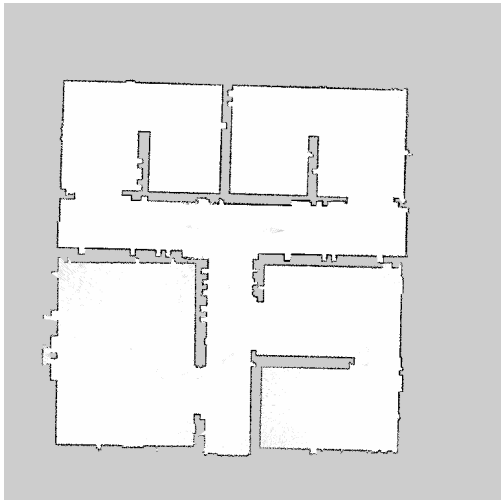

Figure 5: Mapa obtido pelo explorer

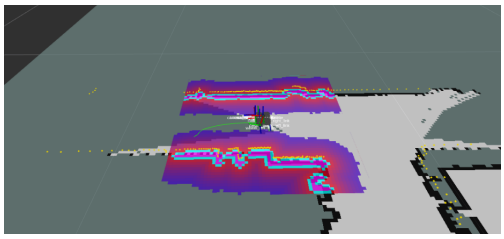

Figure 6: Informação dos sensores no rviz

6 Conclusão

Como pode ser confirmada através dos videos, o robô executa com sucesso o mapeamento do ambiente e deteção dos códigos QR.

No entanto, a segunda parte do *guide node* não se encontrai ainda completa. A plataforma tem a capacidade de se dirigir ao código QR para fazer a sua leitura e executar os *waypoints* lidos no entanto quando tenta passar ao seguinte QR code dá um erro que ainda não fomos capazes de resolver (Clearing both costmaps outside a square (3.00m) large centered on the robot).