Luís Pedro Esteves Sarmento

Navegação Assistida e Semi-Autónoma da Plataforma ROBONUC

Luís Pedro Esteves Sarmento

Navegação Assistida e Semi-Autónoma da Plataforma ROBONUC

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob a orientação científica de Vítor Manuel Ferreira dos Santos, Professor Associado com Agregação do Departamento de Engenharia Mecânica da Universidade de Aveiro

o júri / the jury

presidente / president

Professor Doutor Miguel Armando Riem de Oliveira
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Doutor Paulo Jorge Sequeira Gonçalves
Professor Coordenador do Instituto Politécnico de Castelo Branco (arguente principal)

Professor Doutor Vítor Manuel Ferreira dos Santos
Professor Associado com Agregação da Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Dedico este espaço para agradecer a todos que me acompanharam durante estes anos. Aos meus pais, irmãos e família pelo o apoio e carinho indispensável para atingir os meus objetivos.

Aos meus colegas do LAR que estão sempre prontos a auxiliar nas alturas de maior dificuldade. Em particular queria agradecer ao Bernardo Lourenço, à Joana Mota e ao eng. Rui Heitor que tiveram a paciência para me ajudar, desde as coisas mais simples aos problemas mais complexos.

Ao professor Vítor Santos, pelo acompanhamento, orientação, apoio e conhecimento transmitido ao longo do semestre.

Um particular obrigado a todos os meus amigos por tornarem esta experiência académica única e inesquecível e contribuírem para o meu crescimento pessoal.

Keywords Robuter II, manipulação móvel, ROS, LIDAR, hodometria, SLAM.

Resumo

A manipulação móvel é uma área com uma importância crescente a nível industrial que permite a execução de tarefas complexas em ambientes dinâmicos.

A presente dissertação insere-se no projeto ROBONUC que é constituído por uma plataforma móvel e por um manipulador FANUC. Tem por objetivo a implementação de um sistema de controlo remoto para a plataforma, de modo que esta tenha autonomia suficiente para corrigir a trajetória evitando colisões.

O trabalho está a ser desenvolvido em ambiente *Robotic Operating System* (ROS), que apresenta uma estrutura modular. Este ambiente permite a integração de programas desenvolvidos por terceiros, assim como a distribuição de processos em diversas máquinas.

A primeira tarefa consistiu em implementar a hodometria, ou seja, determinar o deslocamento da plataforma recorrendo à informação proveniente dos *encoders* das rodas. Foi introduzido o HectorSlam, um programa desenvolvido em ROS, que permite realizar a construção do mapa do espaço envolvente recorrendo ao laser Hokuyo instalado na plataforma.

Para garantir a segurança foram definidas zonas de risco de colisão, que são variáveis em função da velocidade da plataforma. Foi considerado existir risco de colisão se um dos dois lasers Hokuyo, presentes na plataforma, detetar um objeto na zona de risco. Desenvolveu-se um programa ROS que permite a navegação segura e autónoma da plataforma. Caso seja determinado risco de colisão o modo autónomo é acionado, permitindo o contorno do obstáculo.

A realização de testes experimentais foi essencial para realizar a calibração da hodometria, assim como a calibração das áreas de risco.

Keywords Robuter II, mobile manipulation, ROS, LIDAR, Odometry, SLAM.

Abstract

Mobile manipulation is a field with increasing importance in the industry which allows to execute complex tasks in dynamic environment.
The current dissertation is inserted in the ROBONUC project which is composed by a mobile platform and a FANUC manipulator. The objective of this work is to implement a remote control system on the platform in such a way that allows it to navigate with enough autonomy to correct the trajectory imposed by the user in order to avoid a collision.
The work is being developed in a modular structured environment called Robotic Operating System (ROS). This software allows to integrate third party programs as well as the distribution of multiple processes in different machines.
The first task consisted of the implementation of the odometry or in other words, the calculation of the distance travelled by the platform using the information provided by the wheel encoders. HectorSLAM was introduced, an open source program developed in ROS, that allows to reconstruct the map of the involving environment using the Hokuyo laser installed on the platform.
To ensure safety, several collision risk areas were defined that are variable in size depending on the platform's velocity. There is the existence of collision risk if one of the two Hokuyo lasers on the platform detects an objective in the risk area. An open source ROS program that allows a safely autonomous navigation was implemented in the platform. In case that a rick of collision is detected the autonomous mode is activated allows to avoid the obstacle.
The execution of experimental tests was essencial to calibrate the odometry, as well as adjusting the risk areas.

Conteúdo

Lista de Figuras **iii**

Lista de Tabelas **v**

1 Introdução **1**
1.1 Contexto . . . 1
1.2 Enquadramento e Motivações . . . 1
1.3 Plataforma ROBONUC . . . 2
1.4 Trabalhos Relacionados . . . 3
1.5 Estrutura da Dissertação . . . 6

2 Plataforma Móvel do ROBONUC **7**
2.1 Locomoção . . . 7
2.2 Unidades de Processamento . . . 8
2.2.1 CPU1 - Mini PC Cubi . . . 8
2.2.2 CPU2 - Arduino Leonardo ETH . . . 8
2.2.3 CPU3 - Arduino Micro . . . 10
2.3 Sensores . . . 10
2.4 Comunicação . . . 12
2.5 Software . . . 12
2.6 *JoyStick* . . . 14

3 Localização **17**
3.1 Hodometria . . . 17
3.1.1 Cinemática da Plataforma . . . 17
3.1.2 *Encoders* . . . 19
Leitura dos Encoders . . . 19
3.1.3 Cálculo da Hodometria . . . 20
3.1.4 Nodo ROS . . . 22
3.2 Mapeamento . . . 23
3.2.1 Sistema LIDAR . . . 23
3.2.2 Leitura de Múltiplos Dados Laser . . . 23
3.2.3 Técnicas de Mapeamento . . . 24
3.2.4 HectorSLAM . . . 25
Sistema de Coordenadas . . . 25
Criação do Mapa . . . 26
Mapeamento com Múltiplos Lasers . . . 27

4 Navegação **29**
4.1 Sistema de Controlo Remoto . . . 29
4.1.1 ROS Distribuído . . . 30
4.2 Navegação Assistida . . . 33
4.2.1 Modo Automático . . . 33
4.2.2 Modo semi-Automático . . . 34
4.2.3 Distância de Paragem . . . 35
4.2.4 Zonas de Risco . . . 37
Zona 1 . . . 39
Zona 2 . . . 40
Zona 3 . . . 41
4.2.5 Nodos ROS . . . 42
Modo Autónomo . . . 43
Leitura do Laser . . . 43
Leitura do *Joystick* . . . 43
Controlo da Velocidade da Plataforma . . . 43
Transformações . . . 43
Mapeamento . . . 46

5 Testes e Resultados **49**
5.1 Hodometria . . . 49
5.2 HectorSLAM . . . 51
5.3 Segurança . . . 51
5.4 Navegação semi-Automática . . . 52

6 Conclusões e Trabalho Futuro **55**
6.1 Conclusões . . . 55
6.2 Trabalho Futuro . . . 56

Referências **57**

Lista de Figuras

1.1 Plataforma ROBONUC: modelo e sistema real. 2
1.2 FANUC LR Mate 200iD. 2
1.3 Robuter II. 3
1.4 Estrutura do manipulador móvel RobuTER/ULM [8]. 4
1.5 Gráfico do campo artificial de vetores de magnitude [11]. 4
1.6 Meio envolvente original $E(t)$ (zona mais escura) e meio envolvente expandido $\hat{E}(t)$ (zona cinza mais clara) [13]. 5
1.7 Ilustração da função binária $M(\alpha, t)$ [13]. 6
1.8 Disco $C(t)$ de diâmetro d_{sen} [13]. 6

2.1 Sistema de tração da plataforma móvel ROBONUC. 7
2.2 Acoplamentos da roda [4]. 8
2.3 CPU1 - MSI Cubi 2 [14]. 9
2.4 CPU2 - Arduino Leonardo ETH [15]. 9
2.5 CPU3 - Arduino Micro [16]. 10
2.6 Hokuyo URG-04LX-UG01 [17]. 11
2.7 Hokuyo UTM-30LX [17]. 12
2.8 Router Asus RT-AC51U [18]. 13
2.9 Diagrama de comunicações da plataforma. 13
2.10 Diagrama dos nodos envolvidos na operação remota da plataforma [4]. 14
2.11 Controlador e recetor Microsoft XBox 360. 15

3.1 Cinemática do robô com locomoção diferencial [21]. 18
3.2 Ondas geradas pelos canais A e B de um encoder [22]. CW - Sentido dos ponteiros do relógio. CCW - Sentido contrário aos ponteiros do relógio. . . . 19
3.3 Diagrama de leitura dos *encoders*. 20
3.4 Tópicos subscritos pelo nodo `odom_node`. 23
3.5 Representação gráfica dos dados laser em RViz. 24
3.6 Mapas obtidos em Ambiente real [28]. 25
3.7 Sistemas de coordenadas necessários para o HectorSLAM. 25
3.8 Sistemas de coordenadas necessários para o HectorSLAM. 26
3.9 Ocorrência de erros aquando a utilização do HectorSLAM. 27
3.10 Criação do mapa com dois lasers. Laser 1 a vermelho e laser 2 a verde. Representação aérea. 28

4.1 Botões do comando XBOX 360. A função de cada botão encontra-se descrita na tabela 4.1. 29

4.2 RViz apresentado por defeito para o controlo remoto. 31
4.3 *Webcam* adicionada à plataforma. 32
4.4 Diagrama de funcionamento do modo semi-automático. 34
4.5 Deslocamento da plataforma e zonas de risco de colisão para uma velocidade linear positiva e velocidade angular não nula. 35
4.6 Deslocamento da plataforma e zonas de risco de colisão para uma velocidade linear negativa e velocidade angular não nula. 35
4.7 Áreas definidas como zonas de risco para velocidade linear positiva. Variável em função de $\vec{v}$. Zona colorida a amarelo - Área 1a. Zona colorida a verde - Área 2a. Zona colorida a vermelho - Área 3a. 37
4.8 Áreas definidas como zonas de risco para velocidade linear negativa. Variável em função de $\vec{v}$. Zona colorida a amarelo - Área 1b. Zona colorida a verde - Área 2b. Zona colorida a vermelho - Área 3b. 38
4.9 Representação gráfica da nomenclatura utilizada. 39
4.10 Zona de risco número 1. 40
4.11 Zona de risco 3a. 42
4.12 Diagrama de funcionamento do modo automático e semi-manual. 44
4.13 Diagrama do modo automático. 45
4.14 Diagrama de leitura do laser. 46
4.15 Diagrama de leitura do comando XBox. 46
4.16 Diagrama de controlo da velocidade da plataforma. 46
4.17 Diagrama das transformações da plataforma. 47
4.18 Diagrama do mapeamento. 48

5.1 Percurso de 5m em linha reta registado com hodometria. 50
5.2 Percurso com inicio e final no mesmo local registado com hodometria. 50
5.3 Áreas do teste de colisão. 51
5.4 Áreas do teste de colisão. Nomeado de A) a I), representando cada letra o momento em que há alteração do modo de funcionamento. 53
5.5 Áreas do teste de colisão. Nomeado de A) a F), representando cada letra o momento em que há alteração do modo de funcionamento. 54

Lista de Tabelas

2.1 Características do CPU1 [4] 9
2.2 Características do CPU2 [15]. 10
2.3 Características do CPU3 [16]. 11
2.4 Características do laser Hokuyo URG-04LX-UG01 [17]. 11
2.5 Características do laser Hokuyo UTM-30LX [17]. 12

4.1 Descrição da função dos botões do controlador XBox. 30
4.2 Condições de velocidade linear e angular para a utilização das áreas 1a, 2a, 3a, 1b, 2b e 3b. Ver figuras 4.7 e 4.8. 38
4.3 Nomenclatura utilizada para determinar a área de risco. 39

5.1 Distância determinada por hodometria para raio de rodas de 12mm e 12.8mm. 50
5.2 Distância de paragem entre a plataforma e um obstáculo. 52

Capítulo 1

Introdução

1.1 Contexto

A manipulação móvel é uma área da robótica de grande crescimento em que o objetivo final é a execução de tarefas complexas em ambientes não estruturados e dinâmicos e onde a interação humana pode ser necessária [1]. Esta área é particularmente promissora em diversas aplicações industriais e de serviços, incluindo montagens, inspeções ou trabalhos em ambientes perigosos [2].

A autonomia do robô está diretamente ligada ao grau de previsibilidade da tarefa e do ambiente. É expectável robôs que executam tarefas em ambiente previsível sejam totalmente automatizados. Pelo contrário, robôs que operem em ambientes com elevada variação, um objetivo é a aplicação da teleoperação com a decisão humana, em que o robô atua como uma extensão do operador [3].

Diversas aplicações na indústria são repetitivas e requerem um elevado nível de concentração e destreza manual. Adicionalmente, é frequente o operador ser responsável pela monitorização do local assim como pela deteção e prevenção de perigos. Este operador encontra-se mais propício a desgaste, o que tem sido associado a acidentes graves. Deste modo, a introdução de sistemas de segurança e de auxílio ao utilizador levam a uma redução da concentração necessária, uma vez que diminui a exigência para o sucesso da tarefa.

1.2 Enquadramento e Motivações

A plataforma ROBONUC, representada na figura 1.1, foi desenvolvida pela integração de um robô móvel com um manipulador. Neste momento inicial do desenvolvimento dispõe essencialmente das unidades principais e infraestruturas suficientes para operação básica. Para poder fazer missões mais complexas, o sistema carece da instalação da unidade de hodometria e de alguns nodos ROS adicionais para a navegação.

Por outro lado, a plataforma deve poder ser teleoperada com segurança através da rede informática, e deve ter autonomia suficiente para efetuar algumas manobras sem intervenção humana. Isso inclui a capacidade de corrigir pontualmente trajetórias que possam, de um momento para o outro, tornar-se arriscadas pela operação humana, cumprindo assim a função de assistência na teleoperação, que é o primeiro passo para a navegação autónoma; em última instância, a plataforma deve parar se houver colisão iminente ou não houver espaço de manobra para continuar o movimento. Para se poder fazer a teleoperação e controlo remoto

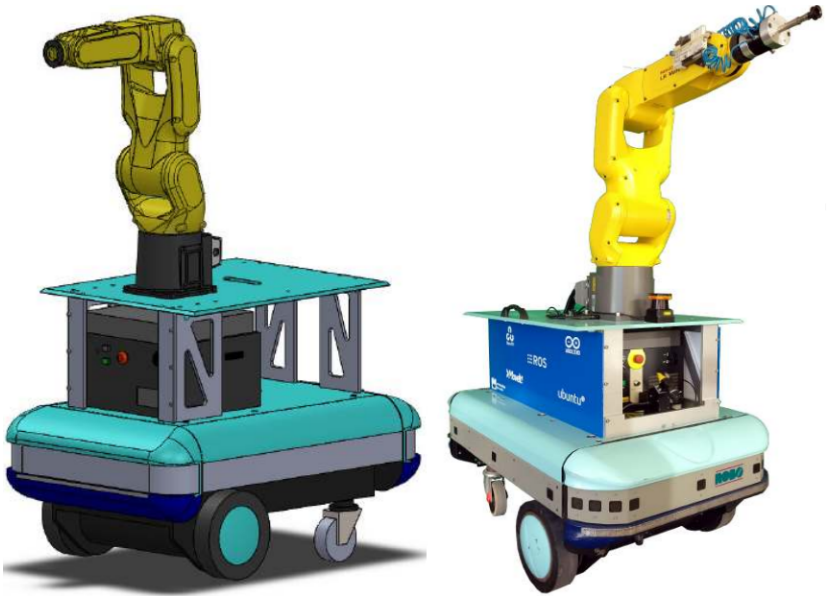


Figura 1.1: Plataforma ROBONUC: modelo e sistema real.

da plataforma é necessário desenvolver uma estação de teleoperação que poderá incluir um *joystick*, *gamepad* ou interface similar, ligada a um terminal de monitorização do estado do sistema e do espaço de navegação detetado pelo sistema à sua volta. Os sensores LIDAR instalados a bordo são usados para este processo de mapeamento do espaço de navegação e assistência à teleoperação. Toda a infraestrutura de software deverá estar sob suporte ROS.

1.3 Plataforma ROBONUC

A plataforma ROBONUC é constituída por uma plataforma móvel, o Robuter II recuperado, e um manipulador, o FANUC LR Mate 200iD.

O manipulador FANUC, representado na figura 1.2, é um robô industrial, versátil, ideal para operações de montagem, empacotamento, *pick and place*, testes e também para atividades educacionais e escolares. O manipulador possui 6 graus de liberdade, um alcance de 717mm e uma carga máxima de 7kg.

Figura 1.2: FANUC LR Mate 200iD.

A plataforma móvel era originalmente o Robuter II, representado na figura 1.3. Esta

sofreu diversas alterações em relação à plataforma original. Inicialmente pesava cerca de 150kg e a sua carga máxima transportável cerca de 120kg. No entanto, a sua massa foi reduzida em cerca de 5%, pelo que permite o transporte de uma carga superior. Os seus 24 sensores de ultrassons foram substituídos por sistemas LIDAR, mais precisos, e com menor complexidade. A linguagem de programação utilizada, ALBATROS, foi substituída por ROS (*Robotic Operating System*). A linguagem ROS dispõe de um elevado número de ferramentas e bibliotecas para aplicações para robôs. É um sistema *open-source* e promove a partilha e reutilização de condigo. Tem ainda possibilidade executar programas independentemente e integrá-los no sistema, sendo permitida a comunicação entre eles.

Figura 1.3: Robuter II.

Os processos em ROS são chamados de "nodos". A comunicação entre nodos faz-se publicando/subscrevendo a um dado "tópico". Um tópico contém uma mensagem enviada por um ou vários nodos publicadores e pode ser lida pelos nodos que subscrevem esse tópico.

1.4 Trabalhos Relacionados

A plataforma Robutter II presente no LAR (Laboratório de Automação e Robótica) do departamento de Engenharia Mecânica da Universidade de Aveiro sofreu diversas alterações no sentido de se adequar à tarefa de manipulação móvel. Estas alterações foram levadas a cabo por Bruno Vieira [4] e por Vítor Silva [5].

O trabalho de Bruno Vieira centrou-se na reconversão da plataforma, uma vez que parte do hardware não se encontrava funcional e o método de programação encontrava-se desatualizado e pouco prático [4].

O trabalho de Vítor Silva centrou-se na integração do manipulador FANUC na plataforma, que permite realizar a manipulação de objetos. É de salientar a adição dos lasers Hokuyo para detetar obstáculos do espaço envolvente [5].

Relativamente à manipulação móvel, recorrendo à plataforma Robuter, foi desenvolvido no *Centre de Développement des Technologies Avancées* (CDTA) um método de criação de rotas livres de colisões para um manipulador móvel complexo conhecendo o meio envolvente com obstáculos, recorrendo a um campo de potencial artificial [6]. A plataforma utilizada é um Robuter, semelhante ao do ROBONUC, com 24 sensores ultrassónicos e um laser frontal [7] para que a plataforma evite obstáculos e se localize no seu meio envolvente. Sobre a plataforma, encontra-se instalado um manipulador ultra leve (ULM) com 6 graus de liberdade, apresentando características semelhantes ao FANUC presente no ROBONUC. A rota gerada faz a ligação da posição inicial da plataforma à posição pretendida da garra do manipulador. Na figura 1.4 é possível observar a plataforma desenvolvida.

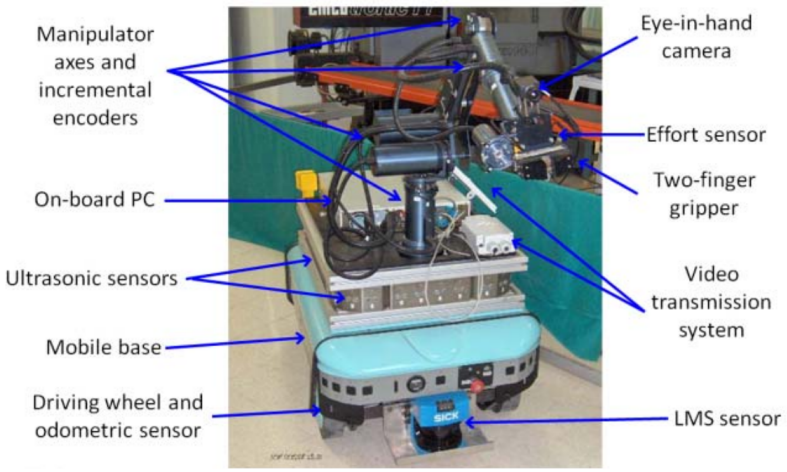


Figura 1.4: Estrutura do manipulador móvel RobuTER/ULM [8].

Para realizar a navegação é definido um campo de potencial. Este campo é definido de forma a que o potencial do objetivo, a posição final pretendida, seja mínimo e obstáculos e paredes sejam definidos como máximo, como mostrado na figura 1.5 [9] [10].

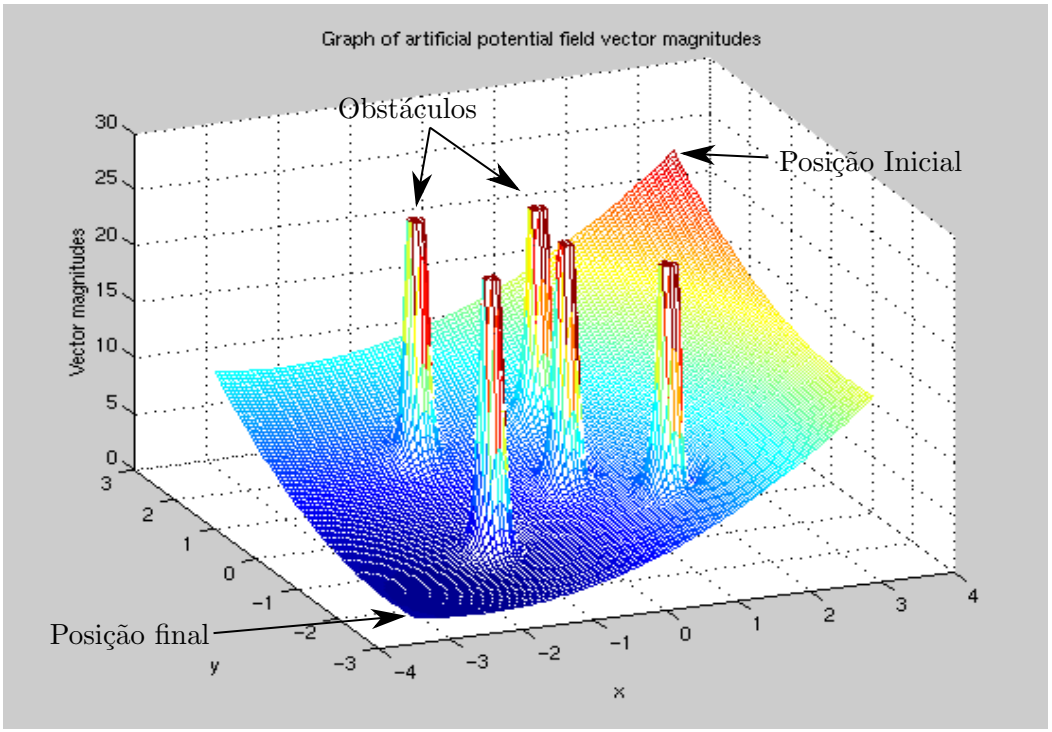


Figura 1.5: Gráfico do campo artificial de vetores de magnitude [11].

Na área de navegação semi-autónoma, foi desenvolvido, por Andrey V. Savkin e Chao Wang [12], um método de navegação assistida de veículos semí-autónomos livre de colisões

em ambientes dinâmicos com obstáculos móveis e estáticos. O sistema proposto consiste num veículo semi-autónomo guiado por um operador humano e por um navegador automático reativo, sendo que o bloco de navegação automática reativa retira o controlo do operador humano em situações em que exista risco de colisão com obstáculos. Em [12] é realizada uma análise matemática do sistema proposto.

O meio envolvente é definido como um conjunto variável no tempo $E(t)$ que é desconhecido pelo sistema de navegação do veículo. É definida uma distância $d_{safe} > 0$ sendo que o objetivo do sistema de navegação autónoma é manter uma distância $d(t)$ superior a $d_{safe} > 0$ de obstáculos. É então definido um meio envolvente expandido considerando a distância de segurança, como é possível observar na figura 1.6.

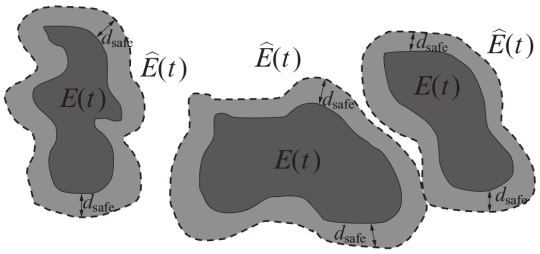


Figura 1.6: Meio envolvente original $E(t)$ (zona mais escura) e meio envolvente expandido $\hat{E}(t)$ (zona cinza mais clara) [13].

Foi introduzido uma equação binária $M(\alpha, t) \in \{0, 1\}$ definida para todo o $t \geq 0$ e todo o $\alpha \in [\theta(t) - \frac{\pi}{2}, \theta(t) + \frac{\pi}{2}]$, em que θ é a direção de deslocamento do robô, da seguinte forma:

- $M(\alpha, t) = 1$ se o raio emitido a um dado tempo t na direção α atinja o meio envolvente expandido $\hat{E}(t)$ num ponto p cuja distância d_p desde o robô, na posição $r(t)$, não exceda

$$d_p(t) := ||r(t) - p|| \leq d_{sen} \cos \alpha \tag{1.1}$$

- $M(\alpha, t) = 0$ de outra forma.

Na figura 1.7 é possível observar o valor de $M(\alpha, t)$ (figura 1.7b) para um meio envolvente expandido $\hat{E}$ (representado na figura 1.7a).

A condição (1.1) pode ser interpretada geometricamente como sendo um disco $C(t)$ de diâmetro d_{sen} centrado no ponto O que se encontra em frente do robô a uma distância $\frac{d_{sen}}{2}$, como é possível observar na figura 1.8.

É através da função $M(\alpha, t)$ que o robô obtém informação do meio envolvente. Caso se tenha $M(\theta, t) = 1$, é porque existe um obstáculo na direção do movimento e o movimento é comandado pelo sistema automático. Caso contrário, o movimento é definido pelo operador humano. Na eventualidade de existir risco de colisão, o sistema automático define uma direção de deslocamento numa direção livre, próxima da direção θ. No caso do exemplo representado na figura 1.7 o sistema automático iria tentar direcionar o robô numa direção com um ângulo inferior a θ_2.

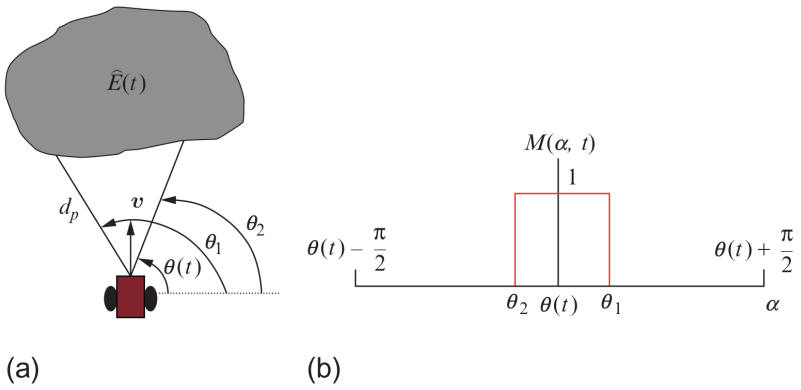


Figura 1.7: Ilustração da função binária $M(\alpha, t)$ [13].

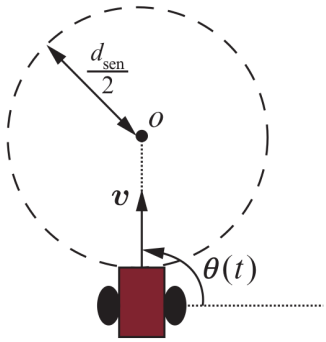


Figura 1.8: Disco $C(t)$ de diâmetro d_{sen} [13].

1.5 Estrutura da Dissertação

A presente dissertação encontra-se dividida em seis capítulos.

O primeiro é reservado à introdução, onde é feito um breve enquadramento do trabalho desenvolvido.

No segundo capítulo são descritas as principais características da plataforma ROBONUC, que apresenta diversas diferenças relativamente à plataforma original, o Robuter II.

No capítulo 3 é iniciada a apresentação do trabalho desenvolvido. São relatadas as técnicas utilizadas para efetuar a localização da plataforma, dividida em 2 partes, para cada técnica utilizada. A primeira parte descreve a implementação da hodometria e a segunda o mapeamento.

No quarto capítulo expõe-se o sistema de navegação da plataforma. O capítulo está dividido em duas partes, sendo a primeira reservada para a interação do utilizador com a plataforma. É apresentado o sistema remoto onde é realizada essa interação. Na segunda parte apresentam-se as técnicas desenvolvidas para efetuar uma navegação autónoma/semi-autónoma.

No capítulo 5 são apresentados os testes e os resultados da navegação semi-autónoma.

Por fim, no capítulo 6 efetuam-se conclusões do trabalho desenvolvido e sugestões para trabalhos futuros.

Capítulo 2

Plataforma Móvel do ROBONUC

A plataforma móvel do ROBONUC sofreu diversas alterações. Pouco mais do que o chassis pertence à plataforma original, o Robutter II. Este capítulo tem como propósito documentar as características mais relevantes da plataforma após as alterações. Para uma descrição mais detalhada podem ser consultados os documentos [4] e [5].

2.1 Locomoção

O sistema de tração não sofreu alterações, mantendo-se o sistema que se encontrava originalmente no Robutter II. A plataforma possuí dois motores elétricos diretamente acoplados às rodas dianteira da plataforma. O controlo dos motores é realizado separadamente, sendo a direção da plataforma controlada pela velocidade de cada motor. Duas rodas giratórias são utilizadas para oferecer estabilidade à plataforma. Na figura 2.1 é possível observar as rodas de tração, de maior raio, e as rodas giratórias, de menor raio.

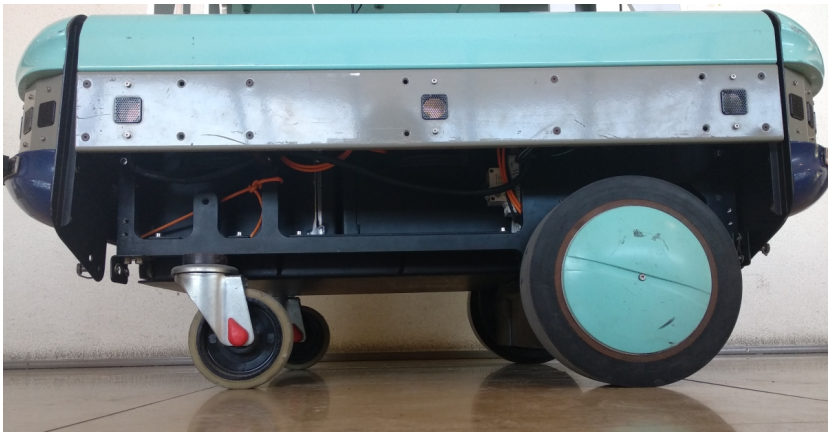


Figura 2.1: Sistema de tração da plataforma móvel ROBONUC.

Um *encoder* e um travão eletromagnético estão acoplados a cada motor. O sistema constituído por estes três elementos pode ser observado na figura 2.2.

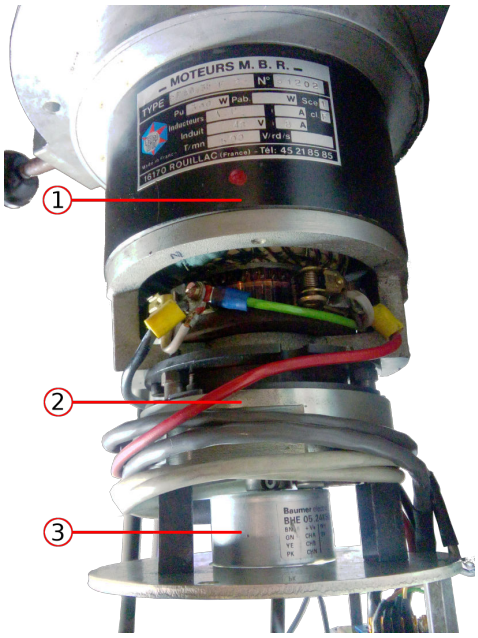


Figura 2.2: Acoplamentos da roda [4].

Legenda da figura 2.2:

1. **Motor:** Íman permanente CC 300W 48V
2. ***Encoder*:** Baumer BHE 05.25K500 de três canais com 8024 pulsos por rotação. No entanto apenas dois canais são utilizados.
3. **Travões:** Travões eletromagnéticos KEB 48V.

2.2 Unidades de Processamento

A unidade de processamento original - VersaModular Eurocard bus (VMEbus), Motorola 68020, 16MHz - foi substituída por três unidades de processamento designadas por CPU1, CPU2 e CPU3, porque o seu método de programação encontrava-se desatualizado e pouco prático. Cada uma destas três unidades de processamento têm funções distintas e serão apresentadas na secção 2.5.

2.2.1 CPU1 - Mini PC Cubi

O CPU1 trata-se de um computador MSI Cubi 2 012BEU i5-7200U. O computador pode ser observado na figura 2.3. As suas especificações encontram-se descritas na tabela 2.1.

2.2.2 CPU2 - Arduino Leonardo ETH

O CPU2 trata-se de um micro-controlador Arduino Leonardo ETH que pode ser observado na figura 2.4. As suas características encontram-se descritas na tabela 2.2.

Figura 2.3: CPU1 - MSI Cubi 2 [14].

Especificações	Detalhes
Chipset	Intel® H110
Processador	Intel® Core^{TM} i5-7200U Dual-Core, 2.5 GHz with Turbo up to 3.1 GHz, 3 MB Cache
Armazenamento	SSD 2.5" Kingston V300 120 GB MLC SATA
Memória	RAM SO-DIMM Crucial Ballistic 8GB DDR4-2400MHz
Gráficos	Intel® HD Graphics 620
Network	Intel® Dual Band Wireless-AC 3168 (M.2 2230) Realtek RTL8111H GigaLan 10/100/1000
Alimentação	19V input, 65W
Dimensões	115 X 111 X 35 [mm]

Tabela 2.1: Características do CPU1 [4]

Figura 2.4: CPU2 - Arduino Leonardo ETH [15].

Especificações	Detalhes
Processador	802.3 10/100 Mbit/s
Micro-controlador	ATmega32u4
Arquitetura	AVR
Tensão de Operação	5V
Memória flash	32 KB das quais 4 KB usadas pelo *bootloader*
SRAM	2.5Kb
Velocidade do Relógio	16 MHz
Pinos Analógicos I/O	12
Pinos Digitais I/O	20
Canais PWM	7
EEPROM	1 KB
Interface	RS232, SPI, I^2C, Ethernet

Tabela 2.2: Características do CPU2 [15].

2.2.3 CPU3 - Arduino Micro

O CPU3 trata-se de um micro-controlador Arduino Micro que pode ser observado na figura 2.5. As suas características encontram-se descritas na tabela 2.3.

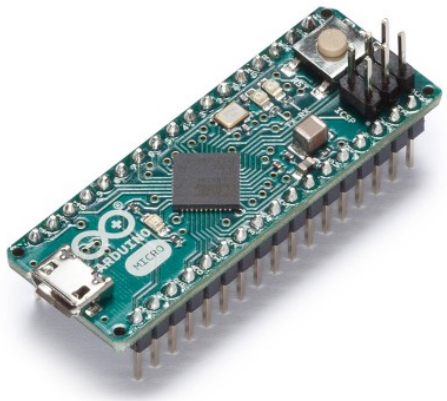


Figura 2.5: CPU3 - Arduino Micro [16].

2.3 Sensores

Para além dos *encoders* referidos anteriormente, a plataforma móvel ROBONUC possui outros sensores de auxilio à navegação: dois lasers *rangefinder* (LRF).

O primeiro trata-se do laser Hokuyo URG-04LX-UG01. É possível observar o laser na figura 2.6 e as suas características na tabela 2.4.

O segundo trata-se do laser Hokuyo UTM-30LX. É possível observar o laser na figura 2.7 e as suas características na tabela 2.5.

Especificações	Detalhes
Micro-controlador	ATmega32U4
Tensão de Operação	5V
Memória flash	32 KB das quais 4 KB usadas pelo *bootloader*
SRAM	2.5Kb
Velocidade do Relógio	16 MHz
Pinos Analógicos I/O	12
Pinos Digitais I/O	20
Canais PWM	7
EEPROM	1 KB
Interface	RS232, SPI, I^2C

Tabela 2.3: Características do CPU3 [16].

Figura 2.6: Hokuyo URG-04LX-UG01 [17].

Especificações	Detalhes
Fonte de Alimentação	5VCC±5%
Fonte Luminosa	Díodo laser semicondutor (λ=785nm), classe 1 de segurança
Área de Medição	20 a 5600mm, 240^o
Precisão	60 a 1,000mm: ±30mm 1,000 a 4,095mm: ±3%
Resolução Angular	Ângulo de passo: $\approx 0.36^o$
Tempo de Leitura	100ms/scan

Tabela 2.4: Características do laser Hokuyo URG-04LX-UG01 [17].

Figura 2.7: Hokuyo UTM-30LX [17].

Especificações	Detalhes
Fonte de Alimentação	12VCC±10%
Fonte Luminosa	Díodo laser semicondutor (λ=905nm), classe 1 de segurança
Área de Medição	0.1 a 30m, 270^o
Precisão	0.1 a 10m: ±30mm
	10 a 30mm: ±50mm
Resolução Angular	Ângulo de passo: $\approx 0.25^o$
Tempo de Leitura	25ms/scan

Tabela 2.5: Características do laser Hokuyo UTM-30LX [17].

2.4 Comunicação

Para efetuar comunicação entre os CPUs foi instalado um *router* na plataforma, ilustrado na figura 2.8. O router permite criar uma rede wi-fi sendo possível aceder e controlar a plataforma remotamente por *Secure Shell* (SSH). A comunicação entre o CPU1 e CPU2 é estabelecida via Ethernet. Para transferir informação entre o CPU2 e CPU3 é utilizado o protocolo I^2C. Na figura 2.9 encontra-se o diagrama das ligações implementadas na plataforma.

2.5 Software

O CPU3 é um contador de pulsos dos *encoders*, que envia periodicamente essa informação para o CPU2.

O CPU2 é responsável pelo controlo da velocidade da plataforma. Este recebe informação do CPU1 relativamente à velocidade pretendida em cada uma das rodas, enviado o sinal correto para as *drives* CC dos motores. A informação dos *encoders*, proveniente do CPU3, é

Figura 2.8: Router Asus RT-AC51U [18].

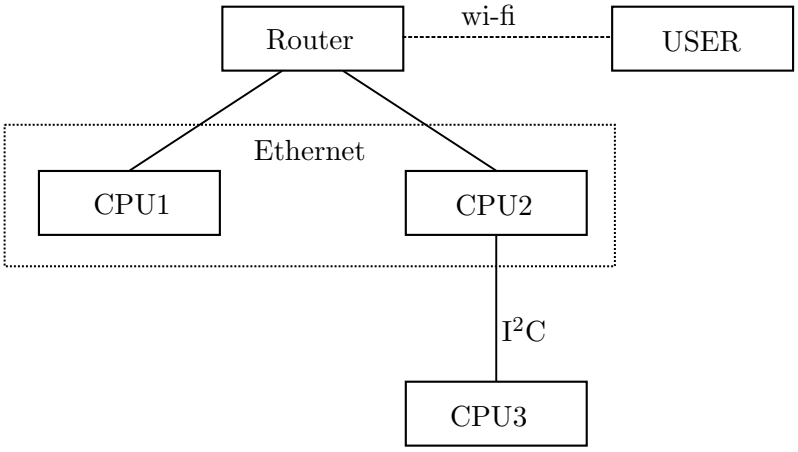


Figura 2.9: Diagrama de comunicações da plataforma.

utilizada para criar um controlador PID para regular corretamente a velocidade.

O sistema operativo utilizado no CPU1 é o Ubuntu 16.04 LTS. Este possui o ROS instalado. A principal função desta unidade é determinar a velocidade de cada um dos motores. Esta velocidade pode ser determinada em função da informação proveniente de um *joystick* controlado pelo utilizador. Para o fazer, são utilizados diversos nodos ROS, que interpretam a informação proveniente de um comando XBox, convertendo em mensagens no formato adequado para enviar para o CPU2. É possível observar o diagrama com os nodos envolvidos no controlo remoto com o comando XBox na figura 2.10.

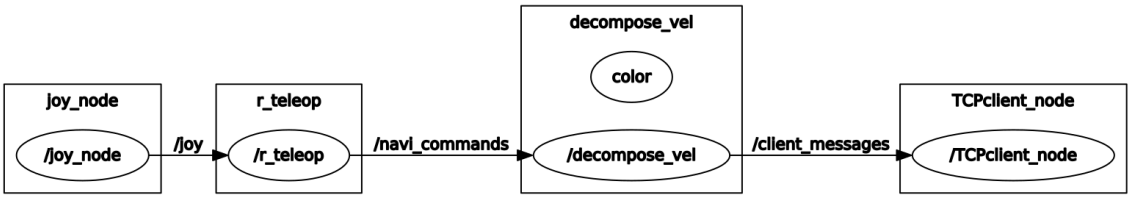


Figura 2.10: Diagrama dos nodos envolvidos na operação remota da plataforma [4].

Assim, o utilizador pode controlar a plataforma, bastando conectar à rede wi-fi da mesma, aceder ao CPU1 por SSH e executar o ficheiro **robonuc_teleop.launch**. Os comandos necessários para o fazer são os seguintes:

- ssh -x robuter@192.168.0.123
- roslaunch r_platform robonuc_teleop.launch

2.6 *JoyStick*

A plataforma pode ser controlada manualmente através de um comando sem fios XBox 360 (figura 2.11). Este comando utiliza um recetor sem fios Microsoft XBox 360 com um alcance de 9m [19]. O comando é alimentado por duas pilhas AA. Este comando foi selecionado por já ser utilizado anteriormente na plataforma e por permitir um controlo preciso e intuitivo.

Figura 2.11: Controlador e recetor Microsoft XBox 360.

Capítulo 3

Localização

O conhecimento da posição exata de um veículo é um problema central em aplicações de robótica móvel. Existem diversas técnicas utilizadas para efetuar a localização de uma plataforma, como a hodometria, a navegação inercial, a utilização de bússolas magnéticas, sistemas de posicionamento global (GPS), correspondência de modelos, entre outras [20].

Tendo em conta os equipamentos instalados na plataforma, os *encoders* e os lasers, é pertinente explorar duas das técnicas anteriormente mencionadas, a hodometria e a correspondência de modelos.

3.1 Hodometria

A hodometria é uma técnica que permite determinar o deslocamento de um robô conhecendo o deslocamento das rodas. Cada nova medida de deslocamento das rodas é convertida numa distância e direção. Conhecendo o tempo entre duas medições pode ser determinada a velocidade.

Esta técnica apresenta algumas desvantagens que, se não forem consideradas, podem invalidar a sua utilização:

- a definição do deslocamento é feita pela aproximação de um movimento curvilíneo da plataforma em segmentos de reta, pelo que será tanto mais precisa quanto maior for a frequência da leitura do deslocamento das rodas;
- eventuais deslizamentos da roda não são contabilizados pelos sensores;
- uma vez que o deslocamento é determinado relativamente ao ponto de inicial, erros provenientes dos pontos indicados anteriormente são cumulativos.

Devido ao erro acumulado, esta técnica é mais adequada para pequenas distâncias, sendo particularmente útil para determinar trajetórias onde há necessidade de realizar um movimento autónomo para evitar uma potencial colisão.

3.1.1 Cinemática da Plataforma

A locomoção da plataforma é realizada por duas rodas. Cada uma destas rodas possui um motor elétrico acoplado. Assim, é possível realizar o controlo individual da velocidade de cada uma das rodas. A direção do movimento pode ser definida controlando a velocidade de

cada uma das rodas, tendo em conta que conferir velocidades distintas às rodas resulta num movimento curvilíneo. A plataforma obtém estabilidade recorrendo a duas rodas giratórias livres.

Na figura 3.1 é possível observar a relação da velocidade de cada uma das rodas, Vr e Vl, e as velocidades linear e angular da plataforma.

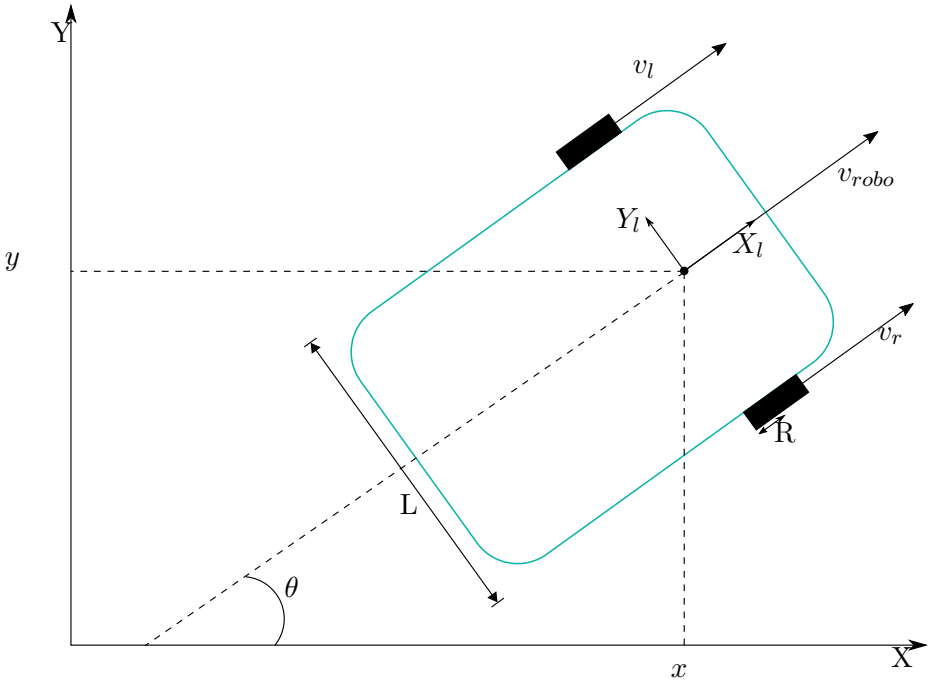


Figura 3.1: Cinemática do robô com locomoção diferencial [21].

$$\begin{cases} \dot{x} = \frac{R}{2}(v_r + v_l)\cos\theta \\ \dot{y} = \frac{R}{2}(v_r + v_l)\sin\theta \\ \dot{\theta} = \frac{R}{L}(v_r - v_l) \end{cases} \tag{3.1}$$

O sistema de equações (3.1) define a variação de posição ao longo do tempo para um referencial estático. Para facilitar a determinação da hodometria é adicionado um segundo o referencial local que se desloca com a plataforma, tendo o eixo xx a mesma orientação da plataforma e sendo, consequentemente, θ nulo. Uma vez que os dados obtidos das rodas correspondem a rotações entre leituras e não velocidades, é vantajoso analisar variações de rotação para definir a trajetória. No entanto, é determinado o tempo decorrido entre duas leituras consecutivas para ser calculada a velocidade da plataforma. Atendendo a estas condições, o sistema de equações resultante (3.2) define o deslocamento da plataforma.

$$\begin{cases} \Delta x_l = \frac{R}{2}(v_r dt + v_l dt) \\ \Delta y_l = 0 \\ \Delta \theta_l = \frac{R}{L}(v_r dt - v_l dt) \end{cases} \tag{3.2}$$

Para voltar ao referencial global foi utilizado o sistema de equações (3.3).

$$\begin{cases} \Delta x = \Delta x_l \cdot \cos(\theta) - \Delta y_l \cdot \sin(\theta) \\ \Delta y = \Delta x_l \cdot \sin(\theta) + \Delta y_l \cdot \cos(\theta) \\ \Delta \theta = \Delta \theta_l \end{cases} \tag{3.3}$$

3.1.2 *Encoders*

O *encoder* tem um disco com furos que tem a mesma velocidade angular da roda. Um LED incide luz sobre o disco e, quando passa nos furos um sensor ótico, deteta e emite um pulso elétrico. Estes pulsos são lidos e interpretados pelo Arduino Micro. Quando a roda realiza uma revolução completa o *encoder* instalado emite 8024 pulsos. Deste modo, é possível fazer a relação entre um pulso e o número de rotações de uma roda. A equação (3.4) permite determinar o número de rotações da roda direita quando o *encoder* emite um dado número de pulsos P.

$$\Delta r = \frac{P}{8024} 2\pi \tag{3.4}$$

Leitura dos Encoders

Cada encoder possui dois canais, A e B. Na figura 3.2 é possível observar as ondas geradas por cada um desses canais e o sentido em que são geradas dependendo da direção de rotação, no sentido dos ponteiros do relógio (CW) ou no sentido contrário (CCW). Assim, sempre que é recebido um novo pulso no canal A é realizada uma leitura do sinal B. Se o sinal for positivo a rotação da roda está a ser realizada no sentido dos ponteiros do relógio, sendo a velocidade da roda positiva, e é incrementado o contador de pulsos. Caso contrário é subtraído 1 pulso ao contador.

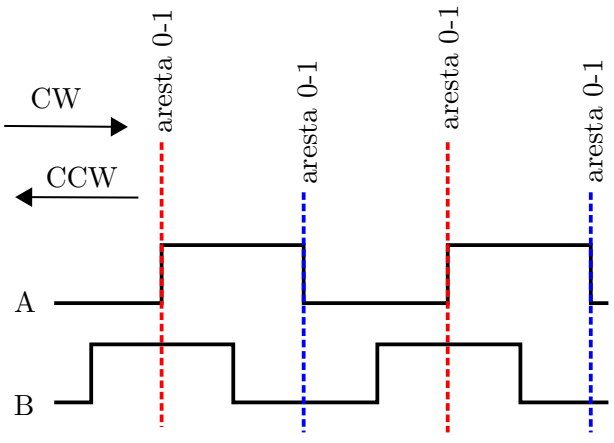


Figura 3.2: Ondas geradas pelos canais A e B de um encoder [22]. CW - Sentido dos ponteiros do relógio. CCW - Sentido contrário aos ponteiros do relógio.

A leitura dos *encoder* é realizada pelo Arduino Micro (CPU3), que realiza a leitura dos canais A e B dos dois *encoders* e a contagem dos pulsos. Este valor é enviado para o Arduino

Leonardo Ethernet (CPU2) a uma frequência de 20Hz. O CPU2 utiliza a leitura dos *encoders* para efetuar o controlo da velocidade definida pelo utilizador. Em seguida é enviada a informação para o CPU1, sendo então possível efetuar o cálculo da hodometria. Na figura 3.3 encontra-se o diagrama de leitura dos *encoders*.

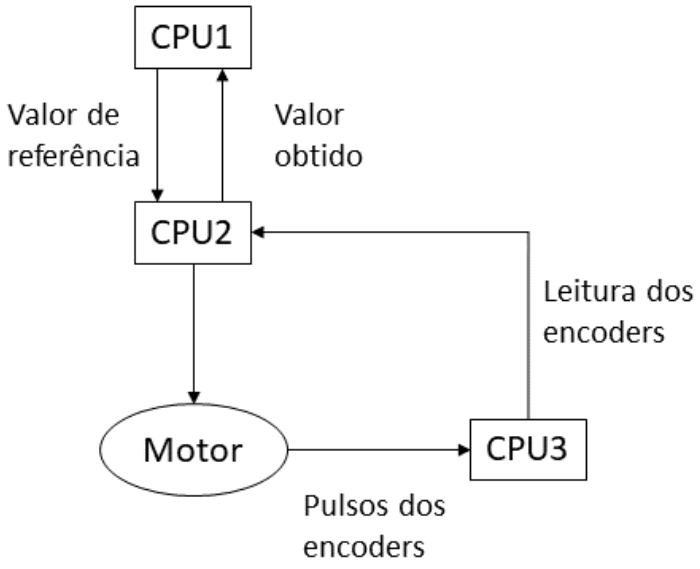


Figura 3.3: Diagrama de leitura dos *encoders*.

3.1.3 Cálculo da Hodometria

Idealmente, o cálculo da hodometria deverá ser realizado no sistema de mais baixo nível de modo a evitar potenciais perdas de informação e uma determinação do deslocamento menos precisa. Foi então considerada a possibilidade de efetuar a determinação da hodometria no Arduino Leonardo Ethernet (CPU2). No entanto, foi optado por efetuar este cálculo em ambiente ROS, no CPU1, por um conjunto de razões:

- o ROS possui ferramentas que facilitam a determinação da hodometria e a sua publicação [23];
- foi verificado que a receção da leitura dos *encoders* é realizada a 20Hz, não havendo perda de informação;
- o cálculo com números decimais (variáveis do tipo *float* ou *double*) necessário para a determinação da hodometria é significativamente mais moroso que o cálculo com números inteiros (*integer*) [24].

Como foi demonstrado que o envio da informação dos *encoders* é estável, não existe desvantagens em realizar a determinação da hodometria no CPU1. No entanto, devido às limitações de *hardware* do Arduino, a adição do cálculo da hodometria poderia levar a que

este não fosse capaz de efetuar os cálculos do controlo da velocidade das rodas a uma frequência de 20Hz.

No algoritmo 3.1 encontra-se um excerto do código em C++ utilizado para determinar e publicar a hodometria.

Algoritmo 3.1: Publicador de Hodometria

```
1  void OdometryPublisher::pidCallback(const comm_tcp::pid::ConstPtr &msg) {
2
3    int pulse1 = msg->encoder_read1; //Numero de pulsos da roda direita
4    int pulse2 = msg->encoder_read2; //Numero de pulsos da roda esquerda
5
6    ros::Time current_time = ros::Time::now();
7
8    /* _params e uma estrutura que contem a o numero de pulsos por rotacao da
9    roda (p_turn), frequencia de leitura dos encoders (msg_freq),
10   largura da plataforma (length) e o raio das rodas (radius)*/
11
12   // velocidade da roda direita e esquerda
13   float xR = (float)pulse1 / _params.p_turn * 2.0f * M_PI;
14   float xL = (float)pulse2 / _params.p_turn * 2.0f * M_PI;
15
16   //sistema de equacoes 3.2
17   float dx = _params.radius / 2.0f * (xR + xL);
18   float dy = 0.0;
19   float dth = _params.radius / _params.length * (xR - xL);
20
21   // sistema de equacoes 3.3
22   double delta_x = dx * cos(_position.th) - dy * sin(_position.th);
23   double delta_y = dx * sin(_position.th) + dy * cos(_position.th);
24   double delta_th = dth;
25
26   // determinacao do tempo entre leituras e da velocidade
27   double dt = (current_time - _last_time).toSec();
28   double vx = delta_x / dt;
29   double vy = delta_y / dt;
30   double vth = delta_th / dt;
31
32   _last_time = current_time;
33
34   // atualizacao da posicao da plataforma
35   _position.x += delta_x;
36   _position.y += delta_y;
37   _position.th += delta_th;
38
39   if (_listener.frameExists("/scanmatcher_frame")) { // verificacao se
         existe referencial do hectorSALM
40
41     if (getDistance()) { /* getDistance() e uma funcao que retorna 1 se a
           distancia entre o referencial da hodometria e do hectorSLAM for
           superior a 0.5m*/
42
43       tf::StampedTransform transform;
44       try {
45         _listener.lookupTransform("/map", "/scanmatcher_frame", ros::Time
              (0),transform);
46       } catch (tf::TransformException ex) {
```

```
47          ROS_ERROR("%s", ex.what());
48          ros::Duration(1.0).sleep();
49        }
50
51        /* Iguala a posicao de hodometria a posicao do hectorSLAM se
              getDistance()==1*/
52        _position.x = transform.getOrigin().x();
53        _position.y = transform.getOrigin().y();
54        _position.th = tf::getYaw(transform.getRotation());
55      }
56    }
57
58    /* uma vez que a hodometria tem 6graus de liberdade e necessario criar um
          quaternion atraves do yaw */
59    geometry_msgs::Quaternion odom_quat =
60        tf::createQuaternionMsgFromYaw(_position.th);
61
62    // publicacao da transformada pelo tf
63    geometry_msgs::TransformStamped odom_trans;
64    odom_trans.header.stamp = current_time;
65    odom_trans.header.frame_id = "odom_wheel";
66    odom_trans.child_frame_id = "odom_frame";
67
68    odom_trans.transform.translation.x = _position.x;
69    odom_trans.transform.translation.y = _position.y;
70    odom_trans.transform.translation.z = 0.0;
71    odom_trans.transform.rotation = odom_quat;
72
73    // envio da transformada
74    _odom_broadcaster.sendTransform(odom_trans);
75
76    // publicacao da hodometria por uma mensagem ROS
77    nav_msgs::Odometry odom;
78    odom.header.stamp = current_time;
79    odom.header.frame_id = "odom_wheel";
80
81    // definicao da posicao
82    odom.pose.pose.position.x = _position.x;
83    odom.pose.pose.position.y = _position.y;
84    odom.pose.pose.position.z = 0.0;
85    odom.pose.pose.orientation = odom_quat;
86
87    // definicao da velocidade
88    odom.child_frame_id = "odom_frame";
89    odom.twist.twist.linear.x = vx;
90    odom.twist.twist.linear.y = vy;
91    odom.twist.twist.angular.z = vth;
92
93    // publicacao da mensagem
94    _odom_pub.publish(odom);
95  }
```

3.1.4 Nodo ROS

Foi criado um novo pacote ROS para realizar o cálculo da hodometria. Este nodo subscreve ao tópico `/pid_data` que contém a informação da leitura dos *encoders*. É então calculado o

deslocamento e a velocidade da plataforma. Como é possível observar na figura 3.4, o nodo publica a transformada que contém a informação do deslocamento da plataforma e um tópico `/odom_wheel`, do tipo `nav_msgs/Odometry`, que contém informação da posição e da velocidade da plataforma. O nodo subscreve a transformada fornecida pelo mapeamento. Isto permite que, quando a transformada da hodometria se distancia da transformada do mapeamento devido a erros acumulados, a posição seja corrigida.

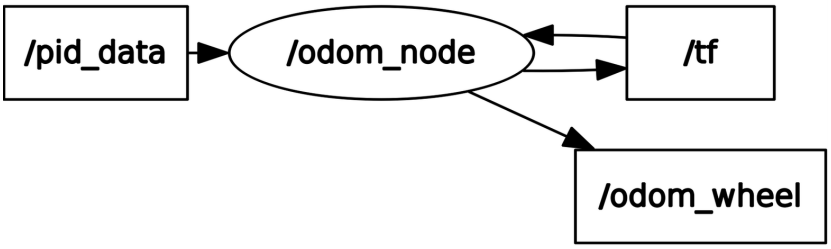


Figura 3.4: Tópicos subscritos pelo nodo `odom_node`.

3.2 Mapeamento

A correspondência de modelos é uma técnica em que o robô utiliza sensores para criar um mapa do seu meio envolvente. Este mapa é então comparado com um mapa previamente guardado em memória, de modo a que se conheça a localização do robô.

Tendo em conta os objetivos desta dissertação, é interessante apenas explorar a criação de um mapa em tempo real, não sendo necessário realizar a comparação com um mapa previamente guardado. Isto porque a navegação é realizada de uma forma semi-manual e não automaticamente, cabendo apenas ao utilizador conhecer o espaço, permitindo a navegação em novos espaços.

Desta forma, o principal objetivo do mapeamento é conhecer os obstáculos do meio envolvente e fornecer informação adicional ao utilizador de modo a facilitar a navegação.

3.2.1 Sistema LIDAR

Um sistema LIDAR (*Light Detection and Ranging*)[25] é utilizado para efetuar o mapeamento. Os lasers disponíveis são o Hokuyo URG-04LX-UG01 e Hokuyo UTM-30LX, ambos lasers frequentemente utilizados em aplicações robóticas. Assim, existem pacotes ROS que realizam a leitura dos dados laser. O hokuyo_node [26] foi o pacote utilizado, que permite realizar a leitura de ambos os lasers, apenas necessitando alterar os parâmetros do ângulo de leitura. Os dados laser podem então ser representados em RViz, como se encontra ilustrado na figura 3.5

3.2.2 Leitura de Múltiplos Dados Laser

O nodo hokuyo pode ser lançado duas vezes para realizar a leitura dos dois lasers. Para tal, foi necessário alterar os parâmetros na *launch file* de modo a adaptar a porta que é

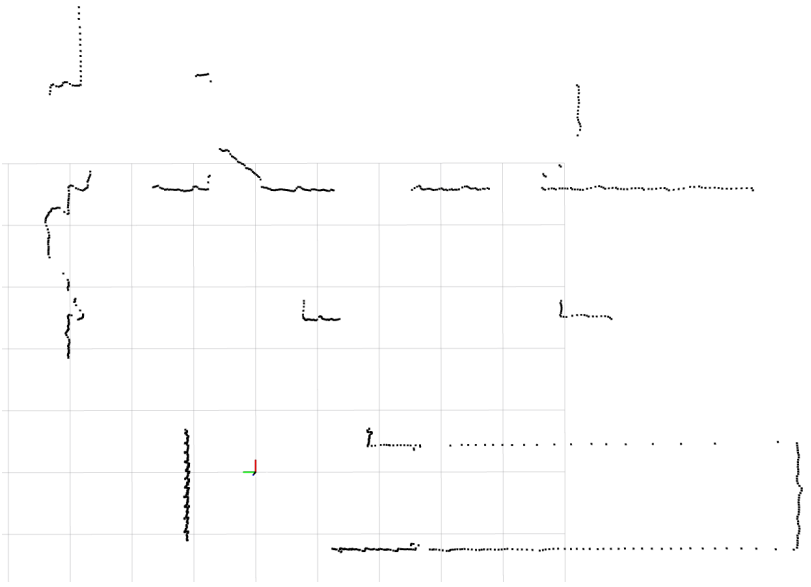


Figura 3.5: Representação gráfica dos dados laser em RViz.

utilizada e o nome do sistema de coordenadas (*frame*). Também, o nome tópico onde são publicados os dados do laser é alterado.

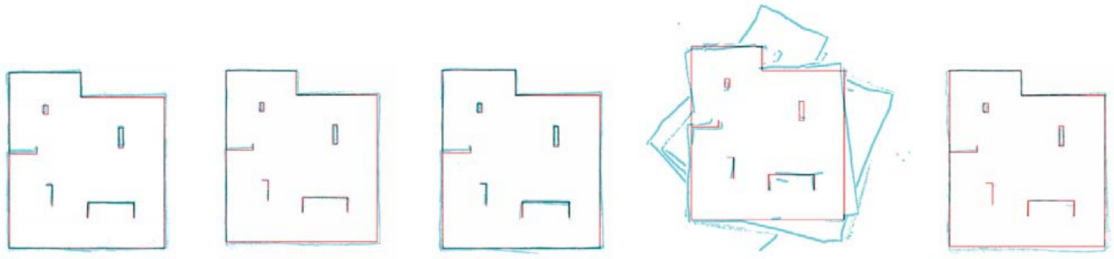
A informação dos lasers pode ser inserida no nodo de mapeamento separadamente criando dois mapas. No entanto, para o utilizador apenas há interesse em observar um mapa. A alternativa aplicada foi a utilização do pacote ira_laser_tools [27]. Este pacote permite fundir a informação de dois lasers numa só, mesmo quando estes se encontram em locais distintos. Para utilizar o pacote, indicaram-se os tópicos dos dois lasers utilizados e a transformada de um laser para o outro, sendo o sistema de coordenadas de um laser o pai e o outro o filho. O resultado final é um tópico com a informação de ambos os lasers e um sistema de coordenadas coincidente com o sistema de coordenadas pai.

O facto da união dos laser apresentar apenas um sistema de coordenadas, que coincide com a origem de um dos lasers, apresenta algumas desvantagens quando é realizado o mapeamento, pelo que esta solução foi abandonada para efetuar o mapeamento, como será discutido mais adiante. No entanto, esta solução - a fusão dos dados dos dois lasers - foi utilizada para a realização a navegação automática do robô.

3.2.3 Técnicas de Mapeamento

Estão disponíveis em ROS diferentes técnicas de *Simultaneous Localization and Mapping* (SLAM) que podem ser utilizadas para realizar o mapeamento, como HectorSLAM, Gmapping, Kar-toSLAM, CoreSLAM e LagoSLAM. A figura 3.6, obtida de [28], mostra resultados obtidos em ambientes reais com cada uma destas técnicas, sendo representado a vermelho o edifício real e a azul os resultados obtidos.

Destas técnicas de SLAM, o Gmapping e o HectorSLAM originaram mapas semelhantes ao edificio real, sendo que são as mais frequentemente utilizadas. A principal diferença entre estas duas técnicas é que, contrariamente ao HectorSLAM, o Gmapping necessita de recorrer à hodometria para realizar a localização da plataforma. Isto representa uma vantagem, principalmente em zonas como corredores, mas leva a resultados menos precisos quando as rodas



(a) HectorSLAM (b) Gmapping (c) KartoSLAM (d) CoreSLAM (e) LagoSLAM

Figura 3.6: Mapas obtidos em Ambiente real [28].

da plataforma deslizam. As diferentes técnicas de localização utilizadas pelo HectorSLAM e Gmapping originam a resultados mais ou menos positivos dependendo do equipamento utilizado para obter a nuvem de pontos [29] [30]. Para obter bons resultados com o HectoSLAM é recomendado utilizar um *Laser Rangefinder* (LRF) de alta frequência, como é o caso do Hokuyo UTM-30LX presente na plataforma. O HectorSLAM não necessita que o laser se encontre montado paralelamente ao plano do chão como no caso do Gmapping. Esta foi uma das razões que levou à escolha do HectorSLAM para realizar o mapeamento, uma vez que, numa fase inicial, o laser traseiro não se encontrava paralelo ao plano do chão. O desenvolvimento da hodometria decorreu em paralelo com a implementação do SLAM sendo que, apenas o HectorSLAM permitia a sua implementação sem a hodometria totalmente desenvolvida. Uma vez que ambas as técnicas de SLAM, HectorSLAM e Gmapping, originam resultados fidedignos, pelos motivos anteriormente referidos, foi optado a utilização do HectorSLAM.

3.2.4 HectorSLAM

Sistema de Coordenadas

Para iniciar a utilização do HectorSLAM é necessário fornecer um conjunto de transformadas, representados de uma forma bidimensional na figura 3.7.

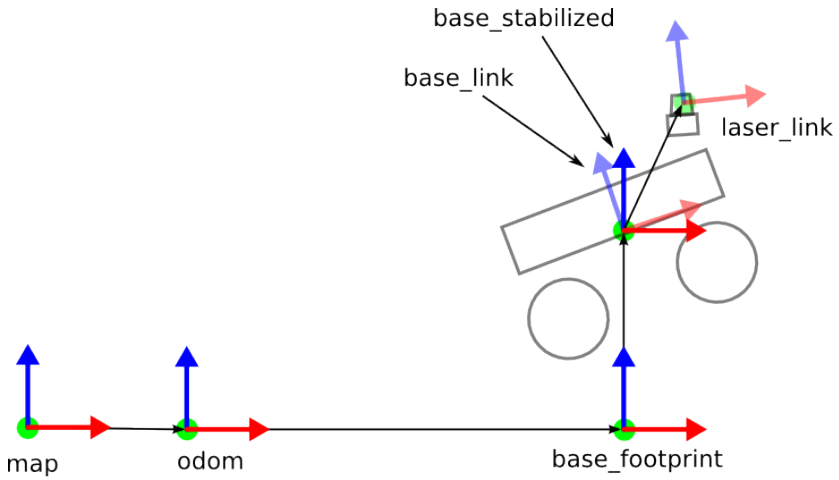


Figura 3.7: Sistemas de coordenadas necessários para o HectorSLAM.

Uma vez que o robô não apresenta partes móveis, o sistema de coordenadas `laser_link` encontra-se estacionário relativamente a `base_footprint` pelo que, de forma a simplificar as transformadas, definiu-se que estes dois referenciais se encontram sobrepostos.

Criação do Mapa

O HectorSLAM subscreve os dados laser publicados pelo nodo `hokuyo_node` de forma a criar um mapa. O espaço entre a origem do laser e um primeiro obstáculo detetado pelo laser é então definido como espaço livre. O espaço que se encontra atrás do objeto detetado pelo laser é definido como desconhecido. As zonas em que o laser não deteta nenhum objeto, ou por não se encontrarem no ângulo de ação do laser ou porque o objeto mais próximo está mais longe do que o alcance do laser, são definidas como desconhecidas. A figura 3.8 representa o mapa criado pelo HectorSLAM quando lhe são fornecidos os dados laser que se encontram representados na figura 3.5.

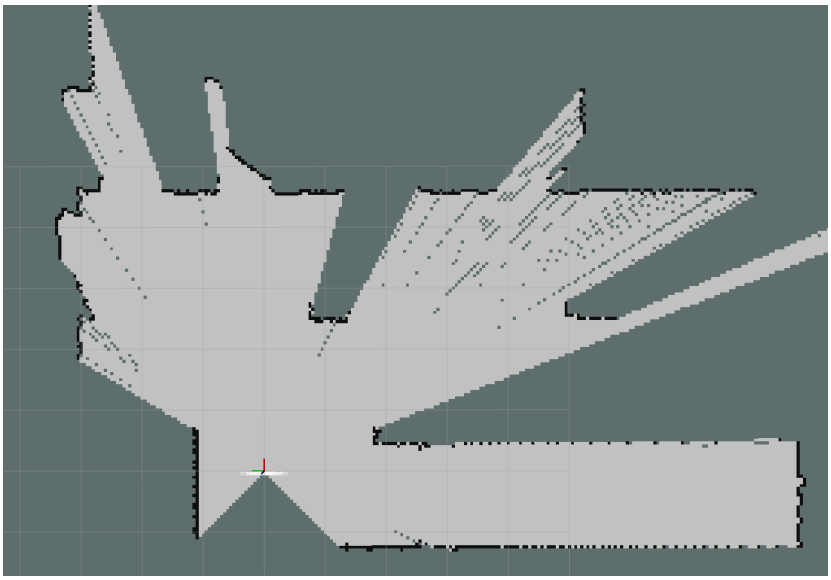


Figura 3.8: Sistemas de coordenadas necessários para o HectorSLAM.

Quando a plataforma se desloca é fornecida nova informação ao HectorSLAM, que utiliza a informação que adquiriu previamente e compara-a com os novos dados. Assim, procura encontrar semelhanças de modo a determinar a sua nova posição. Esta técnica apresenta problemas, nomeadamente quando a plataforma se desloca em corredores em que apenas vê as paredes laterais, o que leva a que os dados do laser sejam idênticos ao longo do corredor. Ocorrem erros também quando a plataforma se encontra em locais em que existem simetrias como salas quadradas ou salas vidradas em que ocorre reflexo do laser originando, assim, erros de leitura.

É possível observar a ocorrência de erros de mapeamento na figura 3.9, quando a plataforma entrou numa sala vidrada e com diversas simetrias. Quando a plataforma saiu da sala, a orientação determinada pelo HectorSLAM foi incorreta, levando à criação de um mapa semelhante ao que tinha já sido criado mas rodado cerca de 45°.

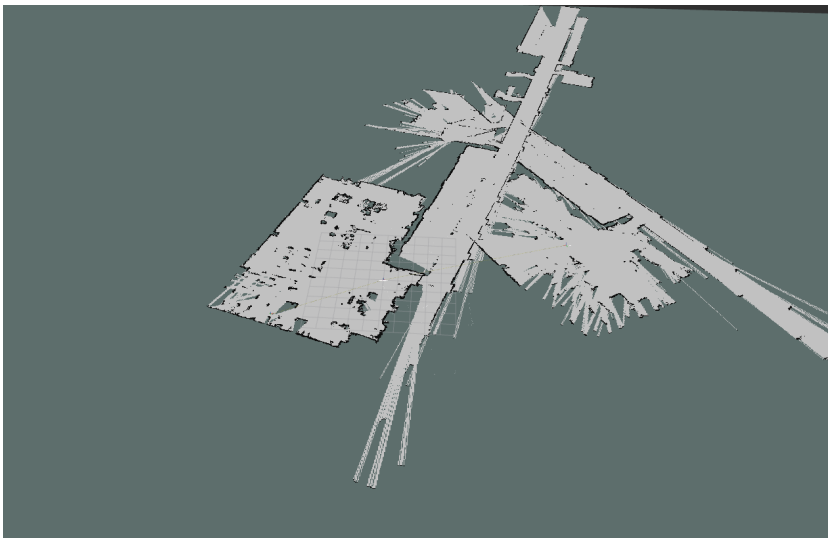


Figura 3.9: Ocorrência de erros aquando a utilização do HectorSLAM.

Mapeamento com Múltiplos Lasers

Como foi referido anteriormente, o HectorSLAM cria o mapa quando deteta um obstáculo, definindo o espaço entre a origem do laser e obstáculo como estando livre. Quando múltiplos lasers são utilizados e fundidos utilizando o pacote `ira_laser_tools`, apenas uma origem é definida apesar de os lasers se localizarem em posições distintas. Isto leva a uma interpretação, por parte do HectorSLAM, que pode ser incorreta. A figura 3.10 ilustra a informação adquirida por dois lasers quando se encontram diante de um pilar circular, numa vista aérea. O laser 1, a vermelho, encontra um obstáculo representado com tracejado vermelho podendo então, o HectorSLAM, definir a área a vermelho como espaço livre. O mesmo acontece com o laser 2, a verde, sendo a área verde espaço livre. No entanto, como o HectorSLAM considera apenas uma origem, neste caso localizado no laser 1, apenas a área vermelha será considerada espaço livre. Esta centralização de origem pode levar a erros de interpretação por parte do HectorSLAM.

Uma alternativa seria a criação de dois mapas e fundir a informação. Devido ao menor raio de ação do laser traseiro e da menor frequência de leituras, este laser encontra-se mais propício a originar erros semelhantes ao que se encontra na figura 3.9. Também o desfasamento de apenas uns centímetros entre a localização realizada pelo HectorSLAM do lazer frontal em relação ao laser traseiro leva a ruído que irá dificultar a compreensão do mapa. A criação de dois mapas em simultâneo requer uma maior exigência de recursos computacionais, que estão limitados, dada a necessidade de processamento em outras tarefas como a navegação autónoma.

O raio de ação, o alcance e a frequência de varrimento do laser frontal da plataforma é superior ao laser traseiro, o que leva a uma construção de mapa mais fiável. Por este motivo, e pelo facto de haver possibilidade de erros quando são utilizados multiplos lasers, foi optado pela utilização de apenas o laser frontal para a criação de um mapa.

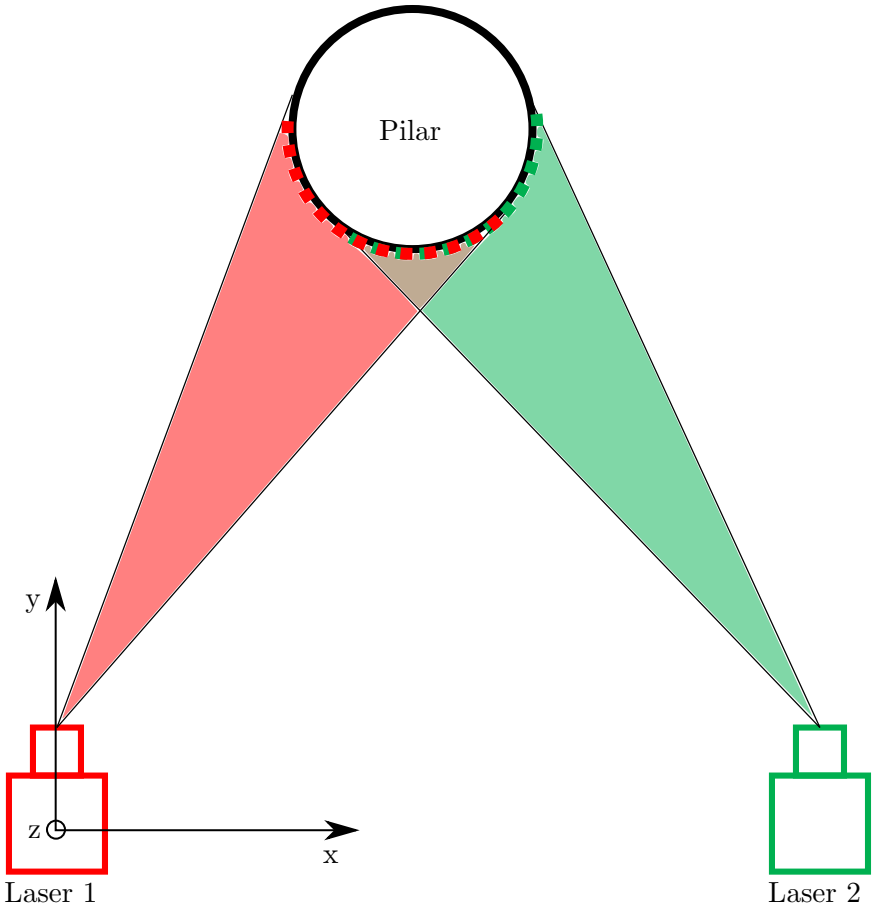


Figura 3.10: Criação do mapa com dois lasers. Laser 1 a vermelho e laser 2 a verde. Representação aérea.

Capítulo 4

Navegação

Para efetuar a navegação da plataforma, o utilizador necessita de uma interface com a mesma, assim como condições de segurança que facilitem o seu controlo em ambientes complexos e dinâmicos. São fornecidas ao utilizador ferramentas necessárias para o fazer. Para controlar a plataforma é disponibilizado um comando XBox e um ambiente gráfico para observar o estado da plataforma. A plataforma está também equipada com dispositivos de segurança que permitem prever eventuais colisões e responder de forma adequada.

4.1 Sistema de Controlo Remoto

De modo a controlar remotamente a plataforma, foi desenvolvido um sistema de controlo para observar e controlar a plataforma através de uma rede wi-fi, não sendo necessário que o utilizador se encontre no mesmo espaço físico.

O utilizador controla a plataforma com um comando XBox e o auxílio de sistemas LIDAR para navegar em segurança. Na figura 4.1 está representado o comando XBox e na tabela 4.1 encontra-se a descrição das funções de cada um dos seus botões.

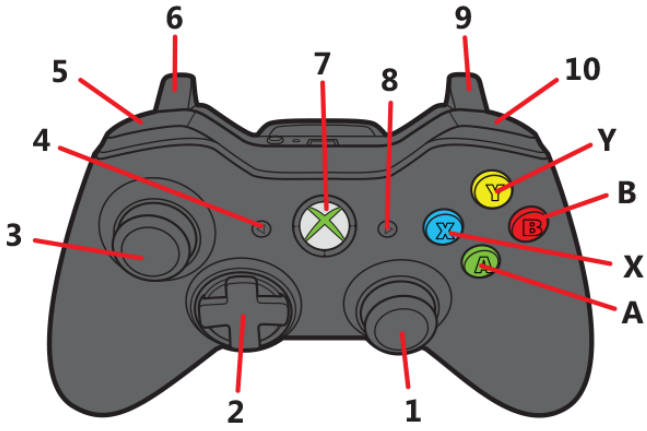


Figura 4.1: Botões do comando XBOX 360. A função de cada botão encontra-se descrita na tabela 4.1.

Tabela 4.1: Descrição da função dos botões do controlador XBox.

Botão	Função
1	Controlo da velocidade angular. Movimento para a esquerda e direita no eixo horizontal.
2	Não utilizado.
3	Controlo da velocidade linear. Movimento para cima e para baixo no eixo vertical.
4	Não utilizado.
5	Diminuição da velocidade máxima.
6	Botão de segurança.
7	Liga o comando. Pressionar durante 1s.
8	Não utilizado.
9	Botão de segurança para o controlo semi-manual.
10	Aumento da velocidade máxima.
A	Não utilizado.
B	Não utilizado.
X	Não utilizado.
Y	Não utilizado.

Existem diversos botões sem funções atribuídas. No entanto o nodo, que faz a leitura dos botões pressionados, faz a publicação dessa informação, mesmo não sendo utilizada. Deste modo, é possível adicionar nodos, ou alterar os já existentes, para atribuir funções a estes botões.

4.1.1 ROS Distribuído

Para que o utilizador comunique com a plataforma é necessário que possua um computador com ROS, para visualizar a informação proveniente da plataforma, como o mapa, e poder enviar instruções.

O objetivo é que o controlo remoto da plataforma seja possível a qualquer distância, desde que tanto o utilizador como a plataforma se encontrem ligados à mesma rede. Como a rede da universidade, a "eduroam", possuí diversas restrições, a rede utilizada é gerada pela plataforma (Robonuc ou Robonuc_5G). Esta rede encontra-se limitada em termos de alcance mas poderia ser substituída por outra, sendo utilizada para testar o sistema. Em [31] encontram-se descritos os passos necessários para configurar a rede.

O alcance da plataforma também se encontra limitado pelo alcance do recetor do comando XBox. É necessário que o recetor se encontre suficientemente próximo do utilizador. A solução encontrada passa pela utilização do ROS distribuído em múltiplas máquinas, sendo por isso necessário o utilizador possuir um computador com ROS.

O ROS foi desenvolvido para funcionar também em sistemas computacionais distribuídos [32]. Assim, é possível correr dezenas, ou mesmo centenas, de nodos distribuídos em múltiplas máquinas. Estes podem, então, comunicar entre si a qualquer momento.

Deste modo, é possível correr alguns nodos diretamente na plataforma enquanto outros correm no computador do utilizador. A plataforma então irá correr todos os nodos necessários à navegação, com a exceção do nodo `/joy`. Este nodo é responsável pelo leitura dos botões

pressionados no controlador XBox. A informação publicada pela plataforma é recebida pelo utilizador, e inclui a leitura dos sensores laser, o mapa, a localização da plataforma, entre outras, e pode ser visualizada em RViz.

O RViz é um ambiente de visualização 3D que permite observar uma grande diversidade de dados, incluindo o que o robô observa. Ou seja, no caso da plataforma ROBONUC, é possível visualizar os dados provenientes dos lasers, construir mapas, imagens provenientes de câmaras, entre outros. É interessante a utilização do RViz numa fase de desenvolvimento uma vez que permite realizar *debugging* mas também é útil para o utilizador para conhecer o estado da plataforma sem necessitar de estar necessariamente a observar a plataforma. O RViz é facilmente personalizado ajustando-se às preferências do utilizador. Uma disposição por defeito para o controlo remoto da plataforma é apresentado na figura 4.2.

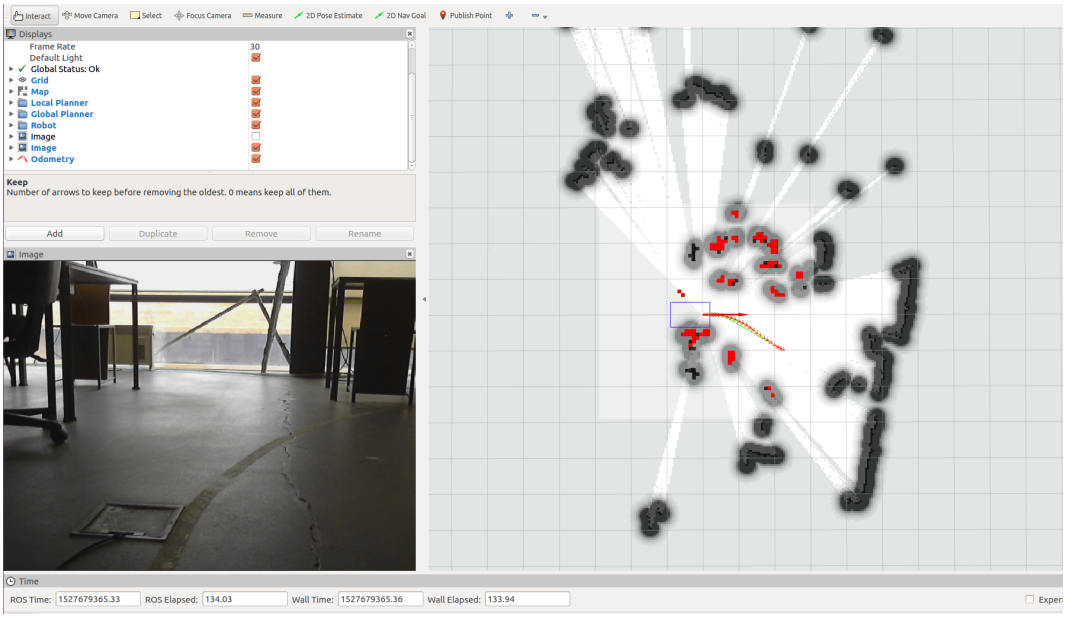


Figura 4.2: RViz apresentado por defeito para o controlo remoto.

No canto canto inferior esquerdo da figura 4.2 encontra-se a imagem obtida por uma *web-cam*; no lado direito encontra-se o mapa do espaço envolvente, a localização da plataforma indicada por um retângulo azul e a rota definida pelo modo automático para que a plataforma atinja o objetivo definido pelo utilizador. Para se definir o objetivo para a posição da plataforma o utilizador deverá selecionar o botão localizado na zona superior do ecrã que indica "2D Nav Goal"e de seguida pressionar no mapa o local de destino pretendido.

Apesar das potencialidades da representação gráfica tridimensional do RViz, optou-se pela visualização bidimensional. Como os lasers fazem o varrimento do espaço num plano horizontal não existe efetiva informação tridimensional. A informação tridimensional conhecida restringe-se à plataforma em si. Esta poderia ser representada tridimensionalmente, mas não traria informação útil para o utilizador já que o espaço envolvente é representado em apenas duas dimensões. Uma representação bidimensional da plataforma simplifica e facilita a compreensão dos dados representados em RViz.

Para facilitar o controlo remoto foi adicionada uma *webcam* na frente do robô. Esta câmara, é meramente demonstrativa do conceito, não sendo a mais adequada para este

propósito, uma vez que o seu ângulo de visão é reduzido, dando pouca informação sobre o espaço envolvente. Por este motivo, a câmara foi fixada de forma temporária na frente do robô, como é visível na figura 4.3. Recomenda-se a sua substituição por um modelo com maior ângulo, como por exemplo a Genius 120-degree Ultra Wide Angle [33]. Esta câmara foi escolhida por ser da mesma família da que está a ser atualmente utilizada na plataforma e por ter um custo reduzido. No entanto, uma outra câmara com um ângulo de visão de 120º ou superior, com interface USB também seria adequada. O software que realiza o registo das imagens, para ser visualizada em RViz, é versátil sendo compatível com a maior parte dos dispositivos com interface USB, podendo apenas ser necessário alterar os parâmetros da imagem como, por exemplo, a sua dimensão.

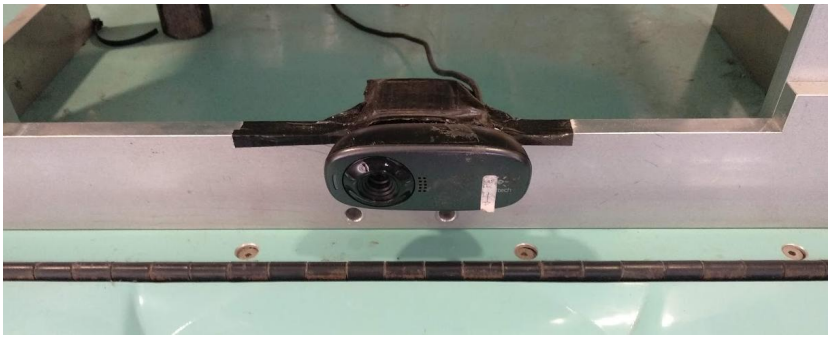


Figura 4.3: *Webcam* adicionada à plataforma.

Para controlar a plataforma remotamente, o utilizador deverá ter um computador com ROS e o pacote *joy*. Instruções para instalar o ROS encontram-se em [34] e para instalar o pacote *joy* basta inserir no terminal o seguinte comando:

```
sudo apt-get install ros-kinetic-joy
```

Reunindo as condições necessárias, o utilizador poderá conectar-se à rede da plataforma e iniciar a comunicação com a mesma. Para o fazer inserir no terminal os seguintes comandos:

```
ssh robuter@192.168.0.123
export ROS_IP=192.168.0.123
export ROS_Master_URI=http://192.168.0.123:11311
roslaunch r_platform r_hybrid.launch
```

Estes comandos iniciam o `roscore`, cujos nodos são requeridos para qualquer sistema ROS, e é definido o `ROS_Master_URI` que irá permitir que outros equipamentos encontrem comuniquem com os nodos da plataforma. São iniciados os nodos necessários à navegação da plataforma com a exceção do nodo `joy`. Será necessário realizar um processo semelhante mas no computador do utilizador. Num novo terminal deverão ser introduzidos os seguintes comandos:

```
export ROS_IP=192:168.0.xx (ip do utilizador)
export ROS_Master_URI=http://192.168.0.123:11311
roslaunch r_external hybrid.launch
```

4.2 Navegação Assistida

A plataforma pode ser controlada de dois modos. O primeiro modo consiste numa navegação completamente autónoma. Para realizar a navegação autónoma o utilizador deverá indicar no RViz o local de destino, que pode ser alterado ao longo da navegação, e pressionar o botão de segurança (botão número 6, figura 4.1). O segundo modo é o modo semi-manual. Este modo é ativado quando é pressionado o botão de segurança em simultâneo com o botão de segurança para o controlo semi-manual (botões 6 e 9, figura 4.1). Este modo permite o controlo da plataforma através do comando mas, em caso de risco de colisão, o controlo automático é ativado. Se o utilizador tiver indicado previamente o local onde pretende deslocar a plataforma, o modo automático irá determinar uma solução para contornar o obstáculo e, quando a plataforma se encontrar num local seguro, o modo semi-automático irá retornar. Caso não seja indicado o local de destino, a plataforma parará obrigando o utilizador a procurar um trajeto seguro para a plataforma.

4.2.1 Modo Automático

Para efetuar a navegação autónoma é utilizado o pacote ROS `teb_local_planner` [35]. Trata-se de um pacote complementar ao pacote `navigation` [36]. O `navigation` aceita informação de hodometria, dos lasers e uma posição objetivo e retorna uma trajetória segura para a unidade móvel. A trajetória inicial gerada pelo `navigation` é então otimizada pelo `teb_local_planner` de modo a minimizar o tempo de execução da trajetória, manter uma distância de segurança aos obstáculos e manter a conformidade com as limitações cinemáticas da plataforma, como a velocidade máxima e aceleração.

Para realizar a navegação segura foi necessário indicar as características cinemáticas e dimensionais da plataforma. O pacote `teb_local_planner` permite definir facilmente uma plataforma com tração diferencial, como é o caso da plataforma móvel ROBONUC. Para o fazer, basta indicar que o raio mínimo para realizar uma curva é nulo, contrariamente aos robôs com direções do tipo *Ackerman* que necessitam de um espaço mínimo para efetuar uma curva. É necessário definir velocidades linear e angular máximas assim como as dimensões da plataforma. As dimensões da plataforma são definidas pela sua pegada ou, por outras palavras, pelo espaço ocupado segundo os eixos xx e yy. Deste modo, o espaço ocupado pela plataforma pode ser representado por um retângulo.

Ambos os pacotes utilizados para a navegação necessitam que seja fornecido previamente um mapa do local. Como o objetivo é que a plataforma seja capaz de navegar em segurança em ambientes desconhecidos, isso impossibilita o fornecimento prévio do mapa do local. O pacote `HectorSLAM` permite criar o mapa em tempo real à medida que a plataforma se desloca. Este mapa pode então ser fornecido ao `teb_local_planner`. No entanto, esta solução apresenta desvantagens. Como é possível observar na figura 3.9, esta técnica de SLAM pode levar à ocorrência de erros; isto pode levar a que algumas paredes e obstáculos sejam inseridos no mapa incorretamente, obstruindo o acesso ao local de destino, não existindo solução de trajetória para atingir o objetivo. Na eventualidade de ocorrência de um erro desta natureza, em que o utilizador detete a obstrução de locais de passagem no mapa, a solução é eliminar manualmente a informação do tópico `/map` de modo a reiniciar o mapeamento. Pode-se eliminar o mapa inserindo no terminal o seguinte comando:

```
rostopic pub /syscommand std_msgs/String "reset"
```

Uma vez que o mapa é desconhecido no início da navegação e vai sendo construido à medida que a plataforma se desloca, optou-se por definir o espaço desconhecido como espaço livre. Desta forma, enquanto a plataforma se desloca, a trajetória vai sendo atualizada à medida que é fornecida mais informação sobre o espaço envolvente.

4.2.2 Modo semi-Automático

O modo semi-manual consiste na integração do modo de navegação autónoma quando a plataforma é controlada manualmente. O modo automático é introduzido no caso de as instruções do utilizador gerarem risco de colisão. Se o utilizador tiver previamente definido um local de destino para a plataforma, o modo automático irá contornar o obstáculo, tentando direcionar-se para o objetivo. Caso contrário, a plataforma irá imobilizar-se, evitando colisão. Para avaliar a existência de risco de colisão é analisada a informação proveniente dos sensores laser. Um nodo, que tem a função de árbitro, verifica se na direção de deslocamento da plataforma o espaço se encontra obstruído, optando pela velocidade indicada pelo nodo manual ou automático em função da informação proveniente dos lasers. É relevante salientar que as zonas de risco necessitam de ser dinâmicas. Por exemplo, se a plataforma se encontrar estacionária com a traseira encostada a uma parede não existirá risco de colisão se o utilizador conferir uma velocidade linear positiva à plataforma, mas existirá se conferir uma velocidade negativa. Na figura 4.4 é possível observar o diagrama de funcionamento do modo semi-automático.

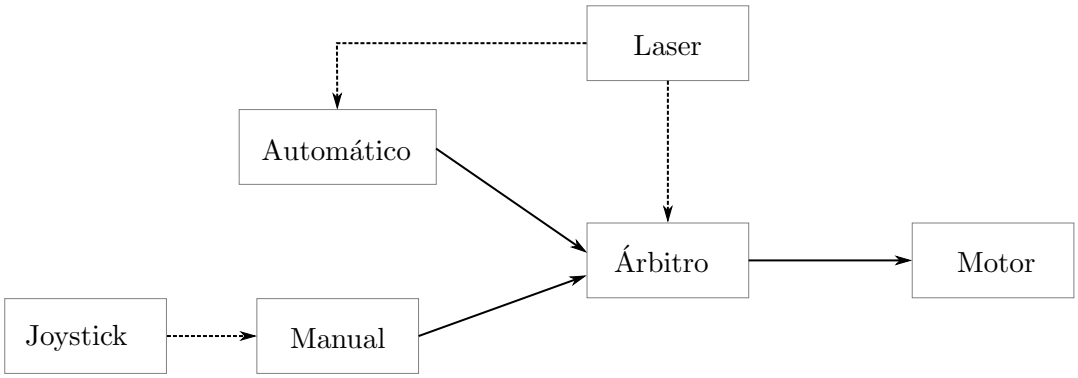


Figura 4.4: Diagrama de funcionamento do modo semi-automático.

O controlo da plataforma em modo semi-automático é semelhante ao controlo em modo manual. O utilizador indica a direção em que pretende deslocar a plataforma com o comando XBox e um nodo com a função de árbitro avalia se o movimento é seguro tendo em conta os objetos detetados pelos sensores laser. Se o árbitro considerar que a manobra é perigosa a velocidade passa a ser definida pelo nodo automático.

Devido a limitações do algoritmo de navegação autónoma, nem sempre este é capaz de determinar uma rota livre. Assim, apesar da principal função do modo semi-automático ser o auxílio do utilizador a evitar e a contornar obstáculos com a navegação autónoma, o contrário também ocorre. O utilizador pode optar por navegar em modo automático e auxiliar com comandos manuais sempre que este não encontre uma solução para o destino pretendido, com a garantia de que os sensores evitam possíveis colisões.

4.2.3 Distância de Paragem

Para definir o momento em que o modo semi-automático comuta o controlo da plataforma pelo utilizador com o *joystick* para o modo automático foram definidas zonas de risco. Se existir algum objeto nessa área é considerado que a manobra inserida pelo utilizador apresenta risco e o controlo é comutado para modo automático.

Na figura 4.5 é possível observar a trajetória da plataforma quando esta apresenta uma velocidade linear positiva e uma velocidade angular não nula, resultando no vetor velocidade $\vec{v}$. Encontra-se também representado o espaço que será ocupado pela plataforma dentro de um curto intervalo de tempo se esta mantiver a velocidade constante. O espaço entre esta posição e a posição inicial é considerado zona de risco pelo que a plataforma deverá mudar o modo de funcionamento para automático caso se encontre um obstáculo neste espaço.

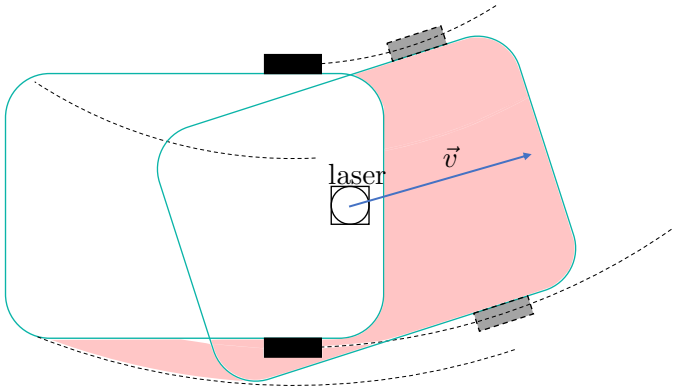


Figura 4.5: Deslocamento da plataforma e zonas de risco de colisão para uma velocidade linear positiva e velocidade angular não nula.

De forma idêntica à figura 4.5, a figura 4.6 representa o deslocamento da plataforma quando esta apresenta uma velocidade linear negativa.

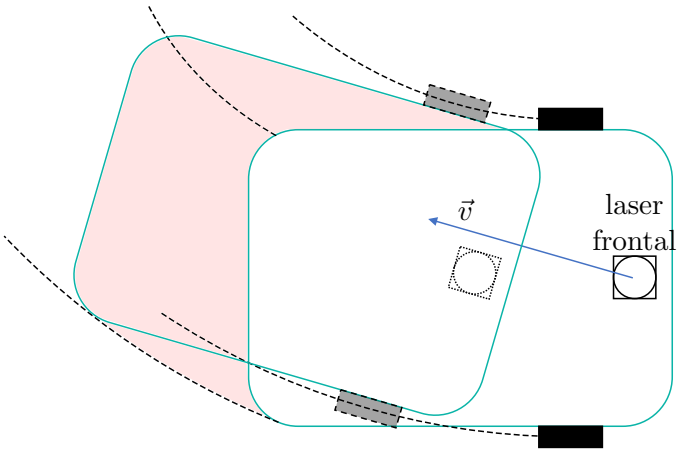


Figura 4.6: Deslocamento da plataforma e zonas de risco de colisão para uma velocidade linear negativa e velocidade angular não nula.

As zonas de risco, realçadas a vermelho, têm uma dimensão variável, sendo que a sua

dimensão será proporcional à velocidade. Se um obstáculo se encontrar nesse local será necessário imobilizar a plataforma antes da colisão. Uma vez que não é possível imobilizar a plataforma num intervalo de tempo nulo, será necessário determinar a distância percorrida pela plataforma desde o aparecimento de um obstáculo até à paragem da plataforma. A distância percorrida varia em função da máxima aceleração que é possível atingir tendo em conta o atrito entre as rodas e o pavimento e da distância d_r percorrida no tempo entre o aparecimento de um obstáculo e a atuação dos motores para imobilizar a plataforma. A distância percorrida entre o aparecimento de um obstáculo e a atuação dos motores é definida pela expressão (4.1), sendo v a velocidade linear da plataforma e t o tempo de atuação dos motores desde o aparecimento de um obstáculo. O tempo t corresponde ao tempo de leitura dos sensores laser e do seu processamento.

$$d_r = v \times t \tag{4.1}$$

Esta distância d_r pode ser nomeada como distância de reação, uma vez que se trata da distância percorrida entre o aparecimento de um obstáculo e o momento em que a velocidade é alterada de forma a evitá-lo.

As áreas terão de ser tanto maiores quanto maior for a distância de paragem d_p. A distância de paragem pode ser calculada determinando o trabalho necessário para dissipar a energia cinética da plataforma. A energia cinética, E, pode ser determinada pela expressão (4.2) e o trabalho de uma travagem, W, pela pela equação (4.3), em que m é a massa do veículo, v a sua velocidade, g a aceleração gravítica e d_p a distância de travagem.

$$E = \frac{1}{2}mv^2 \tag{4.2}$$

$$W = \mu m g d_p \tag{4.3}$$

Igualando o trabalho à energia cinética obtém-se a expressão (4.4).

$$d_p = \frac{v^2}{2\mu g} \tag{4.4}$$

Somando a distância de reação com a distância de paragem obtemos a expressão (4.5) em que a e b são constantes a definir. Apesar de b depender do coeficiente de atrito, que é variável em função do piso, será considerado constante para simplificar o problema.

$$d = av + bv^2 \tag{4.5}$$

O valor de b pode ser determinado se for conhecido o coeficiente de atrito. Assume-se um valor de 0.75 [37], valor típico para pneus de bicicletas no asfalto, que serve apenas como referência e a aceleração gravítica aproximadamente igual a $9.8m/s^2$, então $b = 0.068$. O laser frontal realiza uma leitura do espaço a cada 0.025s e o laser traseiro realiza uma leitura a cada 0.1s . O tempo de processamento dos laser é na ordem dos μs, não sendo significativo. A velocidade é então definida para cada uma das rodas sendo atualizada a cada 0.05s. Considerando a situação mais desfavorável, com o laser traseiro, o tempo de reação é de 0.15s. Substituindo os valores da equação (4.5) obtém-se a equação (4.6) que permite determinar a distância mínima teórica que a plataforma necessita para parar em segurança.

$$d = 0.15|v| + 0.068v^2 \tag{4.6}$$

Isto demonstra que, quando se desloca à velocidade máxima de 1m/s, a plataforma necessita aproximadamente de 20cm para parar após a deteção de um obstáculo.

Apesar do movimento circular gerar uma expressão de rotação de paragem distinta da equação (4.6), a expressão gerada é da mesma natureza que (4.5). Deste modo, a determinação da rotação da plataforma de paragem, derivada do movimento rotacional, será semelhante à determinação da distância de paragem do movimento linear. Assim a rotação da plataforma pode ser aproximada pela expressão 4.7.

$$\theta = (k_1|\omega| + k_2\omega^2) * \text{sinal}(\omega) \tag{4.7}$$

Sendo que k_1 e k_2 são aproximados a 0.15 e 0.068 respetivamente, e em que *sinal()* retorna 1 ou -1 se o número for positivo ou negativo respetivamente.

4.2.4 Zonas de Risco

Para definir zonas de risco, as áreas representadas nas figuras 4.5 e 4.6 foram simplificadas para linhas retas e curvas, sendo que o espaço entre as linhas e o laser é considerado zona de risco, como é possível observar nas figuras 4.7 e 4.8. O laser frontal é responsável por garantir a segurança quando a plataforma se desloca com velocidade linear positiva ou nula, sendo que o laser traseiro é responsável por garantir a segurança quando a plataforma se desloca com velocidade linear negativa ou nula. São definidas 3 áreas para cada situação. As áreas têm dimensões variáveis e são definidas de modo a abranger o máximo de área de risco.

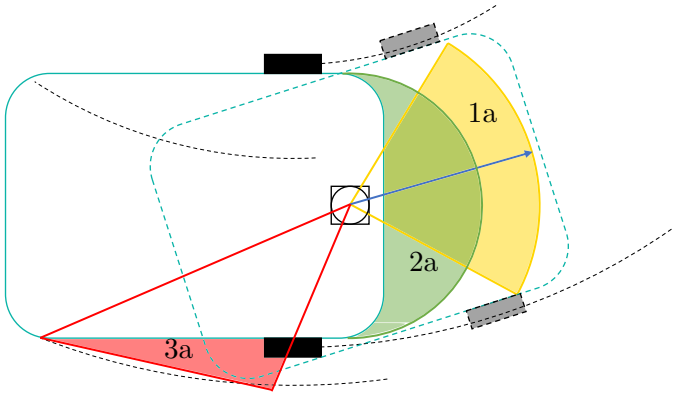


Figura 4.7: Áreas definidas como zonas de risco para velocidade linear positiva. Variável em função de $\vec{v}$. Zona colorida a amarelo - Área 1a. Zona colorida a verde - Área 2a. Zona colorida a vermelho - Área 3a.

As áreas 1, representadas a amarelo nas figuras 4.7 e 4.8, rodam em torno do laser em função da velocidade angular e o seu raio será determinado em função da velocidade linear proporcionalmente à equação (4.5). As áreas 2, representadas a verde, são áreas de risco constante, sendo a sua dimensão mínima igual à largura da plataforma e de dimensão superior para velocidades lineares não nulas. São aplicadas sempre que as velocidades angular ou linear são não nulas. As áreas 3 são tanto maiores quanto maior for a velocidade angular. É de notar que quando a velocidade linear é nula, mas não a angular, as áreas 2a, 2b, 3a e 3b representadas nas figuras 4.7 e 4.8 são aplicadas. Se a velocidade linear for positiva apenas as áreas 1a, 2a e 3a são aplicadas, sendo que se a velocidade linear for negativa as áreas apenas

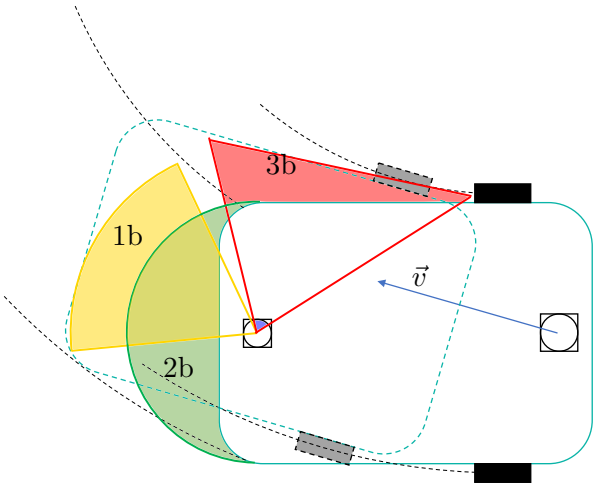


Figura 4.8: Áreas definidas como zonas de risco para velocidade linear negativa. Variável em função de $\vec{v}$. Zona colorida a amarelo - Área 1b. Zona colorida a verde - Área 2b. Zona colorida a vermelho - Área 3b.

as áreas 1b, 2b e 3b são aplicadas. Na tabela 4.2 encontram-se as condições de velocidade angular e linear necessárias para cada uma das áreas.

Tabela 4.2: Condições de velocidade linear e angular para a utilização das áreas 1a, 2a, 3a, 1b, 2b e 3b. Ver figuras 4.7 e 4.8.

		Área					
Velocidade Linear	**Velocidade Angular**	1a	2a	3a	1b	2b	3b
$v > 0$	$\omega = 0$	✓	✓	-	-	-	-
	$\omega \neq 0$	✓	✓	✓	-	-	-
$v < 0$	$\omega = 0$	-	-	-	✓	✓	-
	$\omega \neq 0$	-	-	-	✓	✓	✓
$v = 0$	$\omega = 0$	-	-	-	-	-	-
	$\omega \neq 0$	-	✓	✓	-	✓	✓

Para verificar se existem obstáculos nas zonas de risco é necessário interpretar a informação proveniente dos sensores laser. A informação é recebida como um vetor onde cada elemento contém a informação da distância entre o laser e o objeto mais próximo. O primeiro elemento do vetor corresponde à leitura no ângulo mínimo do sensor e o último ao ângulo máximo. No caso do laser frontal, valores inferiores a 0.1m e superiores a 30m devem ser descartados já que se encontram fora do alcance do laser. Para o laser traseiro são descartados valores inferiores a 0.02m e superiores a 5.6m. Para cada elemento do vetor é necessário definir qual é o valor de distância máximo e mínimo que este tem de apresentar para que seja considerado existir um obstáculo dentro de umas das zonas de risco. Na tabela 4.3 encontra-se a nomenclatura utilizada para determinar a distância.

A relação entre o índice do vetor do laser, i, e o ângulo do laser, β, é uma relação linear

Variável	Significado
ω	Velocidade angular.
v	Velocidade linear.
θ	Rotação da plataforma
β	Intervalo que varia entre $[\beta_{min}, \beta_{max}]$, em rad, da zona do laser a analisar. Está contido no intervalo $\left[-\frac{3\pi}{4}, \frac{3\pi}{4}\right]$rad para o laser frontal e $\left[-\frac{2\pi}{3}, \frac{2\pi}{3}\right]$rad para o laser traseiro.
i	Índices do vetor com a leitura do laser correspondentes aos ângulos β.
d_l	Distância obtida no vetor de leitura do laser.
i_{size}	Dimensão do vetor de leitura do laser.
α	Ângulo entre β_{min} e β_{max}

Tabela 4.3: Nomenclatura utilizada para determinar a área de risco.

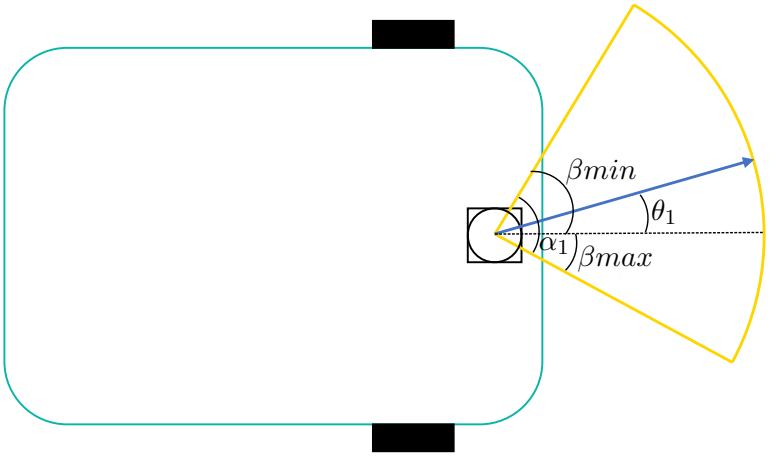


Figura 4.9: Representação gráfica da nomenclatura utilizada.

e pode ser definida pela expressão 4.8.

$$i(\beta) = \left(\frac{1}{2} + \frac{2\beta}{3\pi}\right) * i_{size} \tag{4.8}$$

Para definir as zonas 1, 2 e 3 foram determinados entre que valores varia o ângulo β e distância d.

Zona 1

A zona 1a, representada na figura 4.10, é definida por um arco de circunferência quando a plataforma se desloca com uma velocidade linear positiva. Isso significa que a distância máxima da área de risco de colisão em função do ângulo β é constante. Esta distância é definida pela equação (4.6) multiplicada por uma constante de segurança n_1, obtida em testes experimentais. É necessário adicionar uma constante uma, vez que o laser não se encontra na extremidade da plataforma. O ângulo α_1 é constante, apresentado o mesmo valor independentemente das velocidades linear e angular, e foi definido de forma experimental de modo a que, a elevadas velocidades, a área tenha uma maior largura do que a plataforma,

mas menor para velocidades reduzidas. Deste modo, quando a plataforma atravessa locais apertados, como portas, a plataforma é obrigada a circular a baixa velocidade.

A área não se encontra centrada, sofrendo uma rotação. A rotação é determinada em função da velocidade angular, sendo representado pelo ângulo θ_1 e determinado pela equação 4.7.

A equação (4.9) define a variação do ângulo β e a equação (4.10) a variação da distância d para a zona 1. A variável n_1 trata-se de um coeficiente de segurança determinado experimentalmente de modo a que a plataforma pare em segurança a diferentes velocidades, tendo o valor igual a 4.

$$\theta_1 - \frac{\alpha_1}{2} < \beta < \theta_1 + \frac{\alpha_1}{2} \tag{4.9}$$

$$0.1 < d < (0.15v + 0.068v^2) * n_1 + 0.1 \tag{4.10}$$

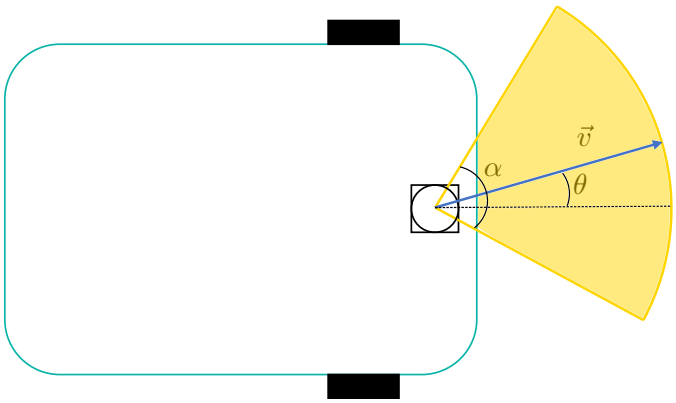


Figura 4.10: Zona de risco número 1.

A zona 1b é definida do mesmo modo que a zona 1a, mas contrariamente a zona 1a esta é definida quando a plataforma se desloca para trás. Também o coeficiente de segurança necessita de ser superior devido à menor frequência de leitura do laser, sendo que $n_1 = 5$.

Zona 2

O objetivo da zona 2 é complementar a zona 1, garantindo a proteção da zona frontal ou traseira quando a velocidade linear é nula ou reduzida e a angular é não nula, levando a que a plataforma rode sobre si mesma. Tal como a zona 1, esta área é limitada por um arco de circunferência sendo a distância d constante ao longo do ângulo β. A distância d deverá ser metade da largura da plataforma para velocidades lineares nulas e aumentar em função da velocidade linear. O ângulo β deverá variar entre $-\frac{\pi}{2}$ e $\frac{\pi}{2}$.

A equação (4.11) define a variação do ângulo β e a equação (4.12) a variação da distância d para a zona 2.

$$-\frac{\pi}{2} < \beta < \frac{\pi}{2} \tag{4.11}$$

$$0.1 < d < 0.37 + (0.15|v| + 0.068v^2) * n_2 \tag{4.12}$$

A zona 2 é definida de forma idêntica quando a plataforma se desloca para a frente (zona 2a) e quando se desloca para trás (zona 2b) sendo apenas o coeficiente de segurança n_2 distinto. Para a zona 2a o coeficiente de segurança $n_2 = 0.5$ e para a zona 2b $n_2 = 2$.

Zona 3

A zona 3 tem como objetivo garantir que não há colisão lateralmente quando a plataforma roda. Por este motivo a dimensão desta área será tanto maior quanto maior for a velocidade angular.

Como é possível observar nas imagens 4.7 e 4.8 a zona 3, quando a plataforma tem uma velocidade linear positiva, é diferente quando a plataforma tem uma velocidade linear negativa. Deste modo, é necessário definir condições distintas para as duas situações.

A zona 3 é delimitada por uma reta em que a distância ao laser não é constante. Na figura 4.11 encontra-se a representação da área 3a quando a plataforma se desloca para a frente com uma velocidade angular negativa. O ângulo γ foi definido como sendo constante, tendo um valor de 20º ou $\frac{\pi}{9}$rad. O ângulo β varia segundo a equação (4.13) quando a velocidade angular é negativa e varia segundo a equação (4.14) quando a velocidade angular é positiva. A área que foi definida inicialmente, a vermelho, é delimitada por uma reta não paralela à plataforma e com um declive variável. Devido à sua complexidade, não foi encontrada a solução algébrica para determinar a distância de um ponto genérico da reta ao laser. Por este motivo, a área foi expandida para permitir o cálculo da distância d. A nova geometria, a traço descontínuo na figura 4.11, garante as condições de segurança da geometria original. No entanto, pode originar falsos positivos por abranger uma área maior. A reta, paralela à plataforma, encontra-se a uma distância de 35mm (metade da largura da plataforma) acrescido de x, em que x é um valor variável em função da velocidade angular. A distância d pode ser determina pelas expressões (4.15) e (4.16).

$$\frac{\pi}{2} + \frac{\pi}{9} < \beta < \frac{3\pi}{4},\ se\,\theta < 0 \tag{4.13}$$

$$-\frac{3\pi}{4} < \beta < -\frac{\pi}{2} - \frac{\pi}{9},\ se\,\theta > 0 \tag{4.14}$$

$$0.1 < d < \frac{x + 35}{\cos\left(\beta + \frac{\pi}{2}\right)},\ se\,\theta < 0 \tag{4.15}$$

$$0.1 < d < \frac{x + 35}{\cos(\beta - \frac{\pi}{2})},\ se\,\theta > 0 \tag{4.16}$$

Nas expressões (4.15) e (4.16) não foi introduzido um coeficiente de segurança uma vez que este foi introduzido no valor de x de modo a que d seja sempre definido como uma reta paralela à plataforma.

Verificou-se que o laser frontal permite a proteção da zona lateral da plataforma, tendo em conta que caso o comprimento da plataforma fosse superior, poderia ser necessário complementar esta área com o laser traseiro, uma vez que o laser frontal tem um campo de visão de 270º.

No caso da velocidade linear ser negativa, a área gerada apresenta a mesma dificuldade para definir a expressão da reta que não é paralela à plataforma. A área foi então expandida de forma similar ao que foi realizado quando a plataforma tem velocidade positiva gerando

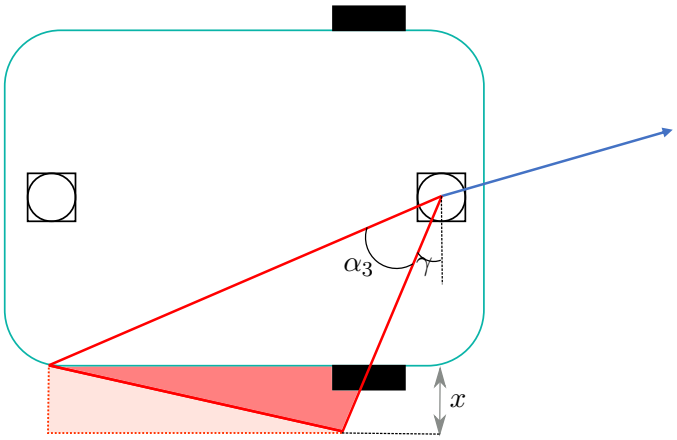


Figura 4.11: Zona de risco 3a.

uma área semelhante à área representada na figura (4.11). Assim, para o laser traseiro, a distância d pode ser determinada do mesmo modo que para o laser frontal, equações 4.15 e 4.16. O ângulo β deverá variar entre $\frac{\pi}{2}$ e a amplitude máxima do laser de acordo com a equação (4.17) para velocidades angulares negativas e de acordo com a equação (4.17) para velocidades angulares positivas.

$$\frac{\pi}{2} < \beta < \frac{2\pi}{3} \tag{4.17}$$

$$-\frac{2\pi}{3} < \beta < -\frac{\pi}{2} \tag{4.18}$$

O valor de x deverá ser diretamente proporcional à rotação da plataforma, sendo definida pela equação (4.19), em que n_3 é um coeficiente de segurança. Verificou-se que a dimensão de x deverá ser maior para a zona 3a do que para a zona 3b. Apesar disso, como o laser traseiro possui uma frequência de leitura inferior, o coeficiente de segurança n_3 determinado experimentalmente é idêntico para ambas as áreas e possui um valor de 1.5.

$$x = (0.15\theta + 0.068\theta^2)n_3 \tag{4.19}$$

4.2.5 Nodos ROS

Na figura 2.10 encontra-se o esquema dos nodos ROS responsáveis pelo controlo da plataforma sem qualquer tipo de sistema de segurança. com o intuito de introduzir o sistema automático e semi-manual, o nodo `/r_teleop` foi substituído. Este nodo tem como função receber as mensagens do nodo `/joy_node` (nodo que faz a leitura dos botões pressionados no comando) e converter as instruções do comando em velocidades lineares e angulares. Foi então criado um nodo que, para além de receber as instruções do comando, também recebe a informação do nodo `/move_base_simple` e `/hokuyo_node`. O nodo `/move_base_simple` pertence ao pacote `navigation` responsável por efetuar a navegação autónoma, e envia para o nodo `/r_hybrid` uma velocidade que terá de decidir se irá conferir à plataforma uma velocidade proveniente da navegação autónoma ou do controlo manual. A seleção é feita em função das pretensões do utilizador, sendo que se pressionar o botão 6 será utilizada a velocidade do

modo automático, e se forem pressionados os botões 6 e 9 em simultâneo será ativado o modo manual (figura 4.1 e tabela 4.1). Se for selecionado o modo manual, a informação do laser é utilizada para garantir que não há risco colisão com a plataforma. Caso seja considerada a existência de risco de colisão, o modo automático é selecionado. Na figura 4.12 encontra-se o diagrama do funcionamento do modo automático e semi-manual. De forma a facilitar a sua análise, o diagrama foi dividido em 6 partes, numeradas de 1 a 6.

Modo Autónomo

Na figura 4.13 encontra-se o diagrama do modo automático. O nodo `/move_base`, que pertence ao pacote `navigation`, recebe as coordenadas para onde o utilizador pretende navegar a plataforma através do tópico `move_base_simple/goal` e determina o trajeto adequado para atingir esse objetivo de modo a evitar obstáculos. Para tal subscreve o tópico `/map` e `/scan` que contem o mapa do espaço e a informação proveniente do laser respetivamente. Também é acedida à localização da plataforma através da sua transformada, `/tf`.

O nodo publica diversos tópicos sendo que servem maioritariamente para realizar *debug* e observar a trajetória calculada. O tópico `/cmd_vel` contém valores de velocidade linear e angular instantâneos que a plataforma deve adquirir para realizar a trajetória pretendida. É esse o tópico lido pelo nodo `/r_hybrid`.

Leitura do Laser

O nodo `/hokuyo_node` é responsável por realizar a leitura dos lasers, que publica o tópico `/scan` utilizado pelo `/hector_mapping` (nodo do pacote `hector_slam`) para criar o mapa, e pelo `/r_hybrid` para navegar de modo semi-manual em segurança. Na figura 4.14 é possível observar o diagrama de leitura dos dados laser.

Leitura do *Joystick*

A leitura do *joystick* é realizada pelo nodo `/joy_node`; publica essa informação no tópico `/joy`, que por sua vez é lido pelo nodo `/r_hybrid`. Na figura 4.15 é possível observar o diagrama de leitura do *joystick*.

Controlo da Velocidade da Plataforma

Após a obtenção das velocidades linear e angular que a plataforma deve adquirir, o controlo da plataforma é realizado de forma idêntica ao feito anteriormente com a plataforma controlada apenas com o comando XBox (figura 2.10). A origem da informação é que é distinta uma vez que passa a ser proveniente do nodo `/r_hybrid`. Com a informação do tópico `/pid_data` é realizado o cálculo da hodomertria pelo nodo `/odom_node`. Na figura 4.16 é possível observar o diagrama para o controlo da velocidade da plataforma.

Transformações

O pacote `tf` é responsável por manter o registo dos referenciais da plataforma. Os referenciais permitem estabelecer a localização da plataforma relativamente a um referencial fixo. Muitos referenciais são estabelecidos devido às exigências do HectorSLAM. Os referenciais são então utilizados pelo HectorSLAM, hodometria e o `/move_base` (navegação autónoma)

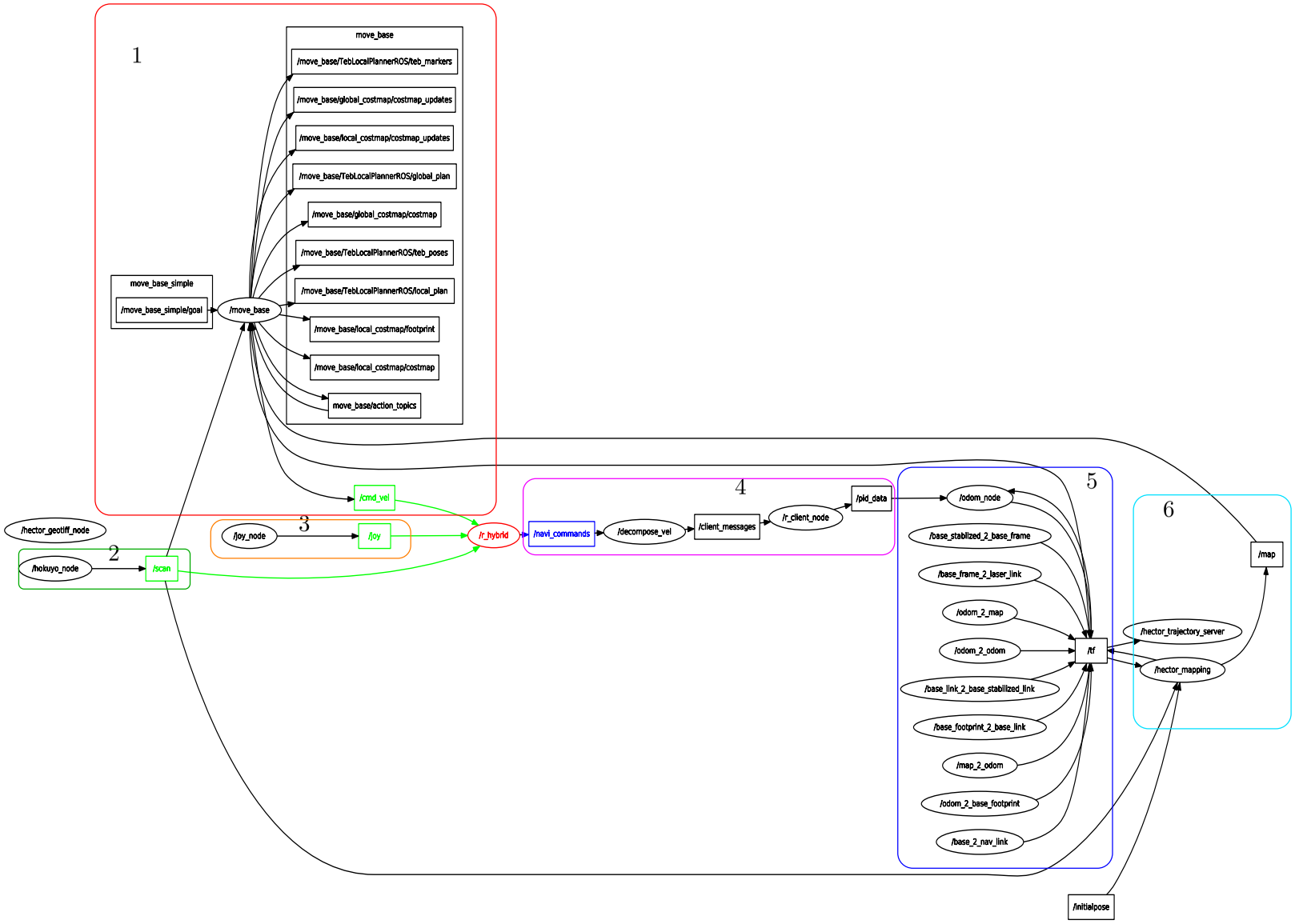


Figura 4.12: Diagrama de funcionamento do modo automático e semi-manual.

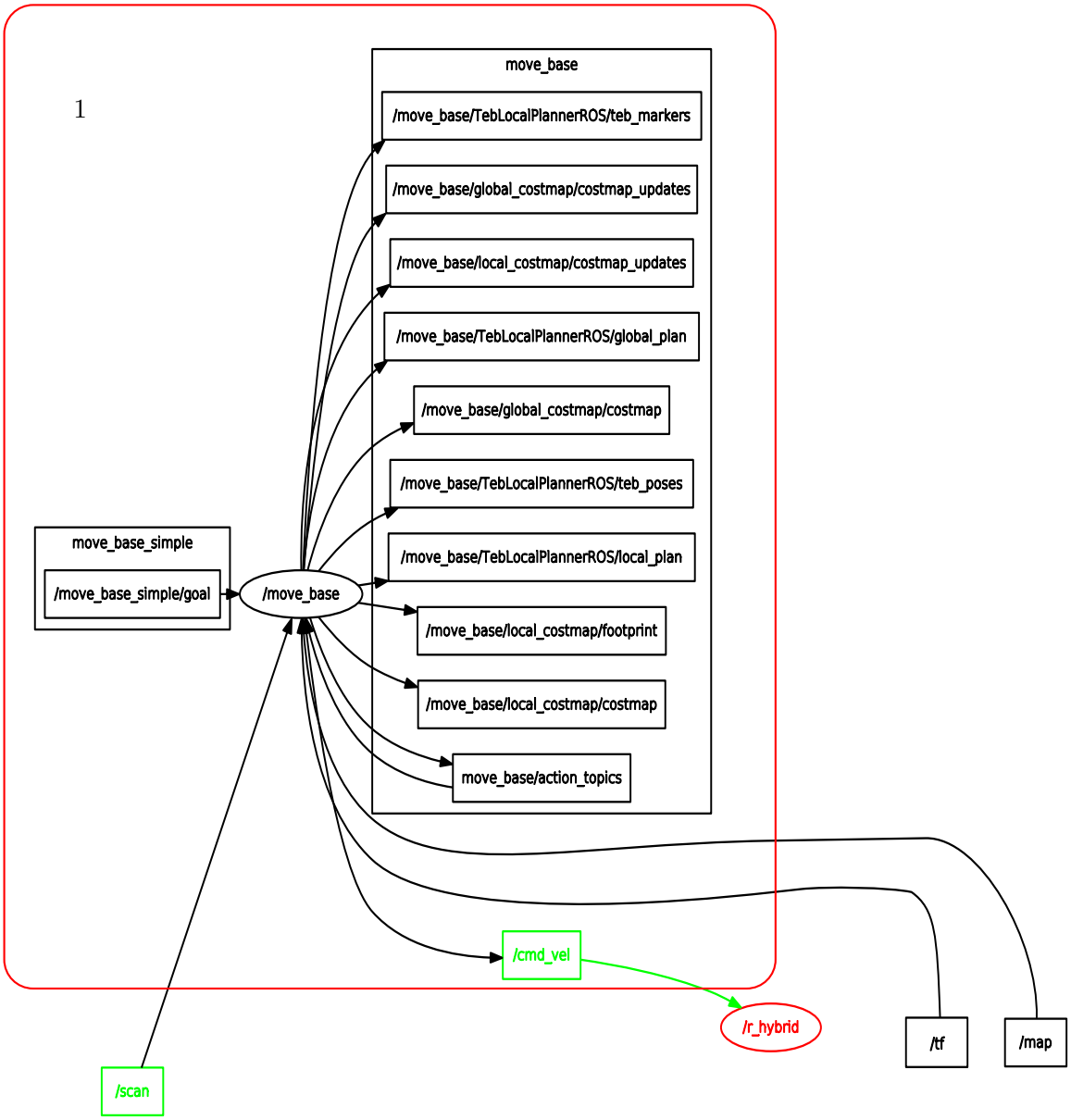


Figura 4.13: Diagrama do modo automático.

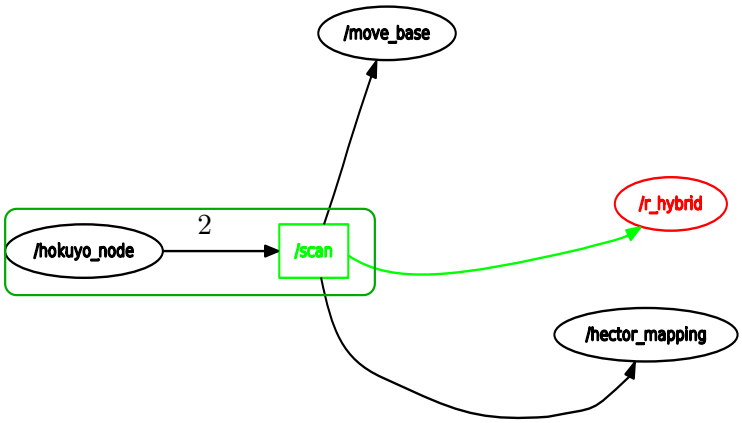


Figura 4.14: Diagrama de leitura do laser.

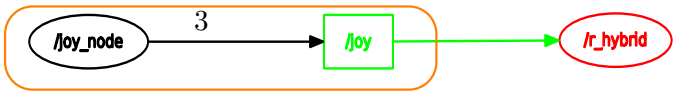


Figura 4.15: Diagrama de leitura do comando XBox.

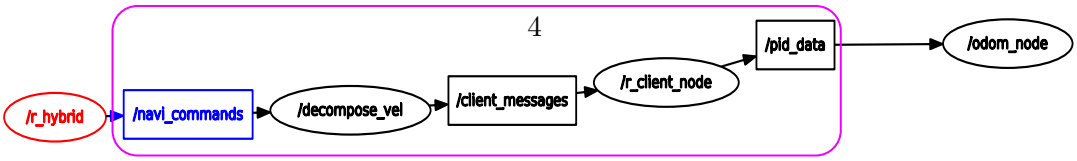


Figura 4.16: Diagrama de controlo da velocidade da plataforma.

para definir ou conhecer a localização da plataforma. Na figura 4.17 encontra-se o diagrama das transformações da plataforma.

Mapeamento

O pacote responsável pelo mapeamento é o `hector_slam`. Este tem dois nodos: `/hector_trajectory_server` e `/hector_mapping`. O primeiro é responsável por realizar o registo da trajetória realizada pela plataforma. O segundo é responsável por criar o mapa recorrendo à informação proveniente do laser. Este mapa pode então ser utilizado pelo nodo `/move_base` para realizar a navegação autónoma. Na figura 4.18 é possível observar o diagrama para o mapeamento do meio envolvente.

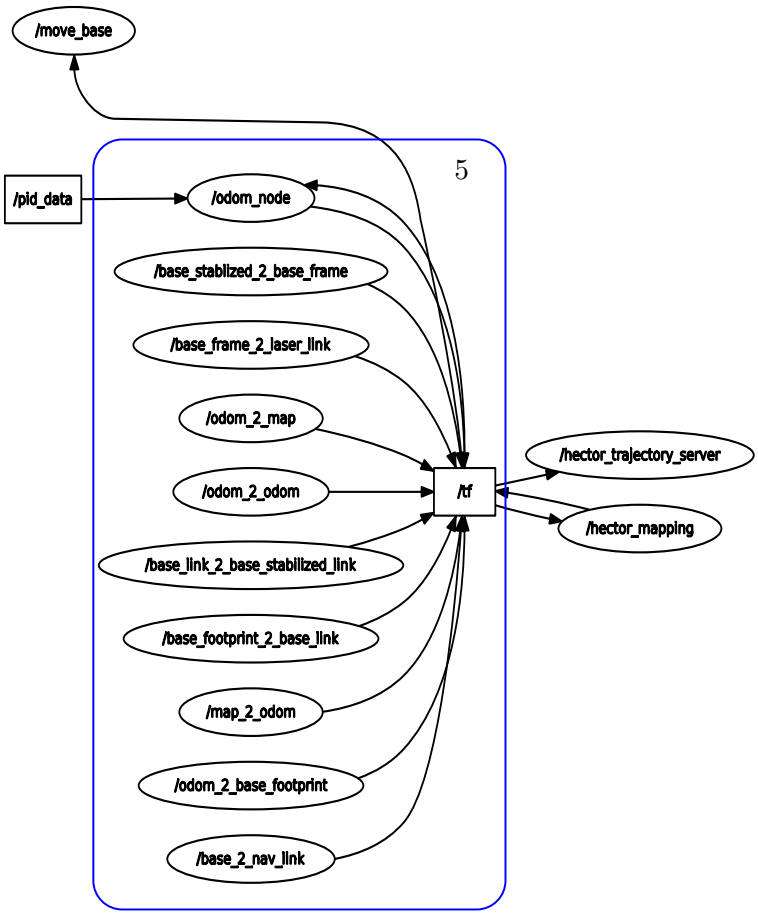


Figura 4.17: Diagrama das transformações da plataforma.

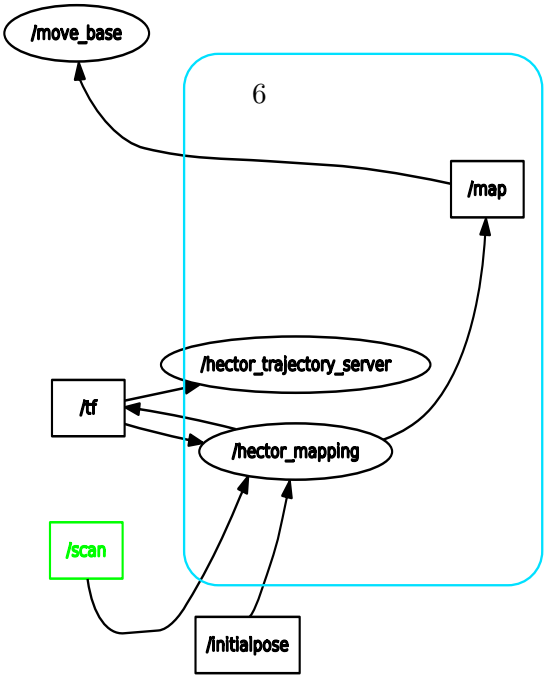


Figura 4.18: Diagrama do mapeamento.

Capítulo 5

Testes e Resultados

De forma a avaliar a precisão das unidades das técnicas de localização (hodometria e mapeamento), o nível de segurança e a integração da navegação autónoma na operação remota foram realizados uma série de testes. Neste capítulo são descritos os testes efetuados, assim como os resultados obtidos.

5.1 Hodometria

A hodometria deverá ser aplicada a curtas distâncias uma vez que apresenta erros que são cumulativos. Estes erros podem ser derivados de situações externas que não podem ser controlados pelo melhoramento do código, como deslizamentos das rodas. Podem também ser derivados de erros de calibração das medidas da plataforma. Foram sobre estes parâmetros que recaíram os testes, sendo que nas sua realização não foram aplicadas variações de velocidade acentuadas de modo a evitar deslizamentos das rodas. Como foi demonstrado pelo sistema de equações 3.2, os parâmetros que devem ser calibrados para definir a cinemática da plataforma são o raio da roda e a distância entre as duas rodas.

Foi definido um trajeto retilíneo para a plataforma percorrer. Isto permitiu verificar que a velocidade da roda direita e esquerda são determinadas com exatidão, não sendo necessariamente precisas. Se o trajeto definido pela hodometria for uma reta, então as velocidades das rodas esquerda e direita são idênticas e, por isso, as medições são exatas. Para verificar a precisão das leituras, e determinar se o raio das rodas foi definido de forma correta, foi comparada a distância percorrida pela plataforma com a distância determinada por hodometria. Na figura 5.1 é possível observar o trajeto definido pela hodometria quando a plataforma percorreu 5m em linha reta. Como se observa na figura, a hodometria definiu um segmento de reta com pequenas curvaturas associadas ao controlo humano. Foram realizados 5 testes em que a plataforma percorreu 5m e a hodometria registou 4,66m em média para os 5 testes, representando um erro de 6,6%.

Com o intuito de diminuir o erro o diâmetro da roda foi ajustado de 12mm para 12.8mm. Apesar de este valor ser ligeiramente superior, a raio da roda da plataforma origina resultados de hodometria mais semelhantes à distância percorrida pela plataforma. Na tabela 5.1 encontram-se os resultados de hodometria obtidos para as rodas de raio de 12mm e 12.8mm.

Para comprovar que a definição da distância entre as rodas se encontra correta realizou-se um trajeto em que a plataforma começa e termina no mesmo local. Se na hodometria se verificar que o local inicial é o mesmo que o local final podemos concluir que a definição da

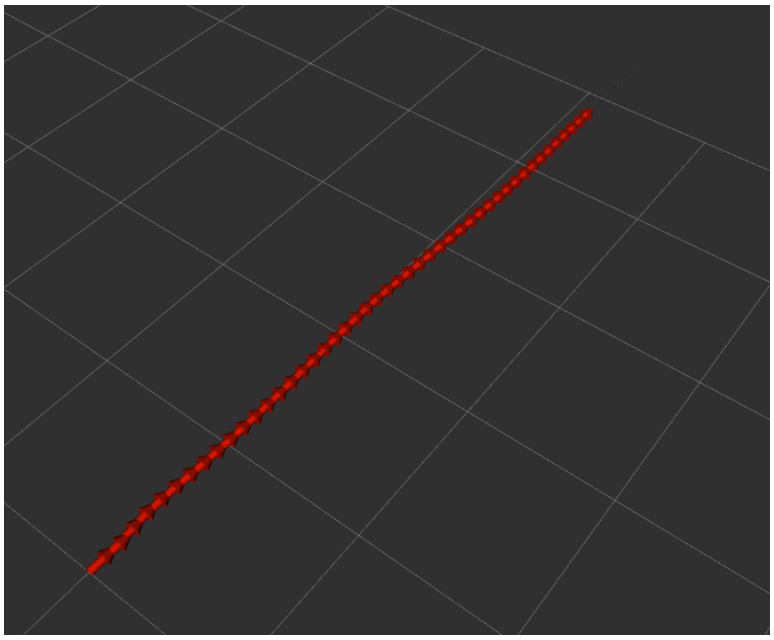


Figura 5.1: Percurso de 5m em linha reta registado com hodometria.

Tabela 5.1: Distância determinada por hodometria para raio de rodas de 12mm e 12.8mm.

Raio(mm)	Distância(m)					Média(m)	Erro(%)
12	4.670	4.678	4.667	4.658	4.662	4.667	6.657
12.8	4.983	4.967	4.968	4.972	4.963	4.971	0.587

distância entre as duas rodas é adequada. Na figura 5.2 é possível observar o trajeto definido pela hodometria num percurso oval. Na imagem encontram-se dois referenciais, um para a hodometria e outro para o HetorSLAM, que se encontram a uma distância reduzida um do outro, apontando para o reduzido erro da hodometria.

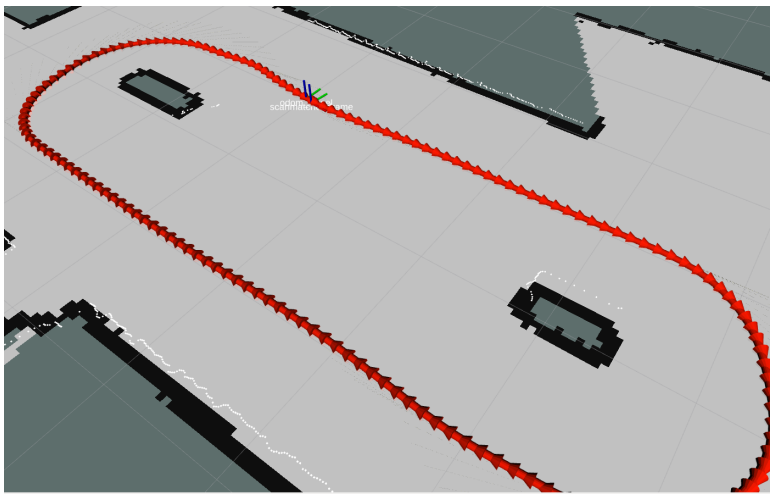


Figura 5.2: Percurso com inicio e final no mesmo local registado com hodometria.

5.2 HectorSLAM

O HectorSLAM é uma ferramenta eficiente que não requer configurações complexas e apresenta resultados semelhantes aos reais. Onde o HectorSLAM apresenta maiores erros é ao efetuar mudanças de direção a elevada velocidade, podendo resultar na sua desorientação. Pretendendo averiguar a velocidade angular máxima recomendada para realizar o mapeamento, foi efetuado um movimento circular da plataforma com velocidade linear nula. A velocidade angular foi aumentada incrementalmente até à ocorrência de erros no mapeamento. Isto permite averiguar a velocidade angular máxima recomendada para realizar o mapeamento com o laser Hokuyo presente na plataforma. Verificou-se que o fator que delimita o correto mapeamento é o tempo de processamento e não o de leitura. Deste modo, o mapeamento depende do computador que efetua o processamento e do número de processos que estão em funcionamento em simultâneo. Devido a problemas de *hardware* do CPU1 não foi possível realizar os testes nesse computador não sendo possível definir velocidades máximas recomendadas para o mapeamento. No entanto, é recomendado evitar inversões repentinas de velocidade.

5.3 Segurança

Garantir a segurança é uma prioridade quando se manobra uma plataforma com a massa e dimensões do ROBONUC. Por este motivo, foram efetuados inúmeros testes de colisão em que se aplica uma velocidade à plataforma que leve à colisão nas 5 direções representadas na figura 5.3. Apenas foi testada metade da plataforma, uma vez que as medidas de segurança são simétricas.

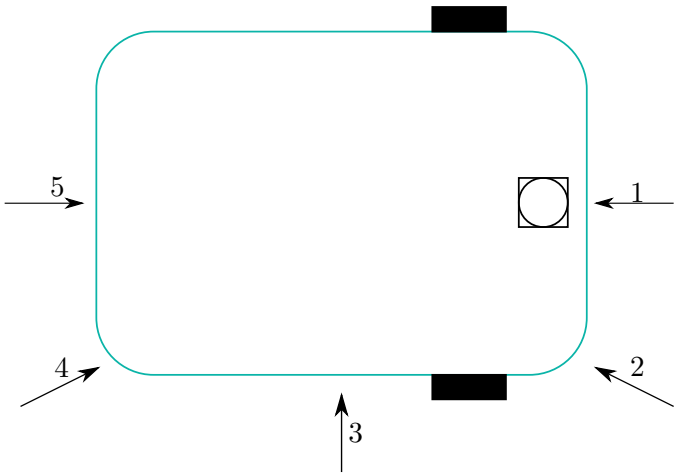


Figura 5.3: Áreas do teste de colisão.

Os testes foram conduzidos a baixas velocidades e na velocidade máxima, verificando-se a distância entre a plataforma e o obstáculo após a paragem da plataforma. Para cada direção e velocidade foram realizados 5 ensaios (5.2. A velocidade introduzida para as direções 1 e 2 é uma velocidade linear positiva, sendo que para as zonas 4 e 5 a velocidade é negativa. Para a direção 3 a velocidade aplicada é apenas angular, sendo por isso indicada em rad/s.

Tabela 5.2: Distância de paragem entre a plataforma e um obstáculo.

Direção	**1 (m/s)**		**2 (m/s)**		**3 (rad/s)**		**4 (m/s)**		**5 (m/s)**	
Velocidade	**0,15**	**0,5**	**0,15**	**0,5**	**0,15**	**0,5**	**0,15**	**0,5**	**0,15**	**0,5**
Distância Média (cm)	20,4	28,4	11,4	18	15,6	29,8	12,4	20,2	22,8	28,4
Desvio Padrão	0,55	2,88	0,65	1,87	3,36	2,28	1,14	2,28	0,84	0,89

Com estes testes foi possível verificar que a plataforma pára em segurança em todas as situações testadas. No entanto em ambiente real os movimentos da plataforma são mais complexos, assim como os obstáculos. Foram realizadas várias viagens com a plataforma, não sendo identificados situações que a mesma não evite-se colisão, desde que o obstáculo estivesse elevado à altura dos lasers.

5.4 Navegação semi-Automática

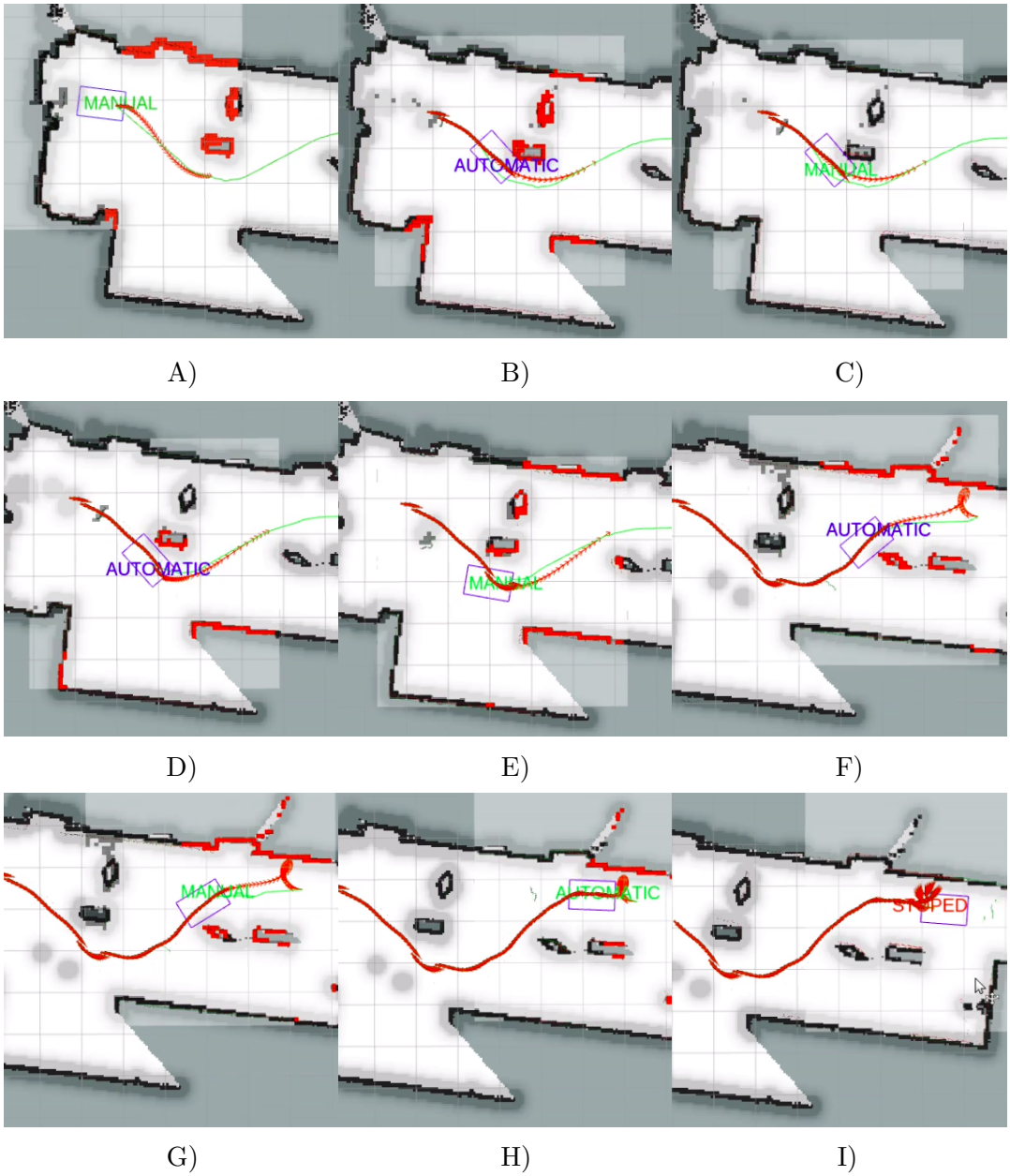
Para testar o modo semi-automático foram realizados dois percursos com característica distintas. No primeiro trajeto a plataforma contornou um conjunto de obstáculos, sendo controlado pelo utilizador. Foram então observados os locais, em que devido ao risco de colisão, o movimento foi comutado para modo automático. No final do percurso a plataforma foi orientada pelo modo automático.

Na figura 5.4 é possível observar o percurso realizado pela plataforma no primeiro trajeto. A figura encontra-se dividida em nove partes nomeadas de A) a I), sendo que cada parte indica o local onde a plataforma altera o seu modo de funcionamento. Desde o local A) a B), de C) a D), de E) a F) e de G) a H) a plataforma é controlada manualmente pelo utilizador. Nos instantes B), D) e F) o controlo semi-automático determinou que existia risco de colisão e retirou o controlo do utilizador passando a plataforma a ser controlada autonomamente. Por fim, no instante H) o utilizador pretendia inverter a plataforma num local com obstáculos próximos e optou por ceder o controlo ao sistema autónomo, terminado o percurso no instante I).

Nos locais B), D) e F) em que a velocidade introduzida pelo utilizador foi considerada ter risco de colisão, o sistema autónomo passou a controlar a plataforma, contornando o obstáculo. No entanto, se o sistema automático não encontrar um trajeto válido ou se o utilizador não tivesse introduzido um local de destino, a plataforma iria imobilizar-se sendo necessário o utilizador introduzir uma nova direção em que não houvesse risco de colisão.

No segundo trajeto foi definido um destino e a plataforma foi guiada pelo modo automático. O trajeto incluiu a passagem por uma porta com 90cm de largura, tendo a plataforma apenas 20cm de margem para efetuar a manobra. O sistema autónomo por vezes tem dificuldade em realizar a manobra sendo por isso auxiliado pelo utilizador. Na figura 5.5 é possível observar o percurso realizado pela plataforma no segundo trajeto. A figura encontra-se dividida em seis partes nomeadas de A) a F), sendo que cada parte indica o local onde a plataforma altera o seu modo de funcionamento.

No local A) a plataforma inicia o seu trajeto em modo automático até ao local B) onde se encontra a porta. Neste local o nodo de navegação autónoma não é capaz de determinar uma rota para a plataforma atravessar a porta. Assim o utilizador toma controlo da plataforma



A) B) C)

D) E) F)

G) H) I)

Figura 5.4: Áreas do teste de colisão. Nomeado de A) a I), representando cada letra o momento em que há alteração do modo de funcionamento.

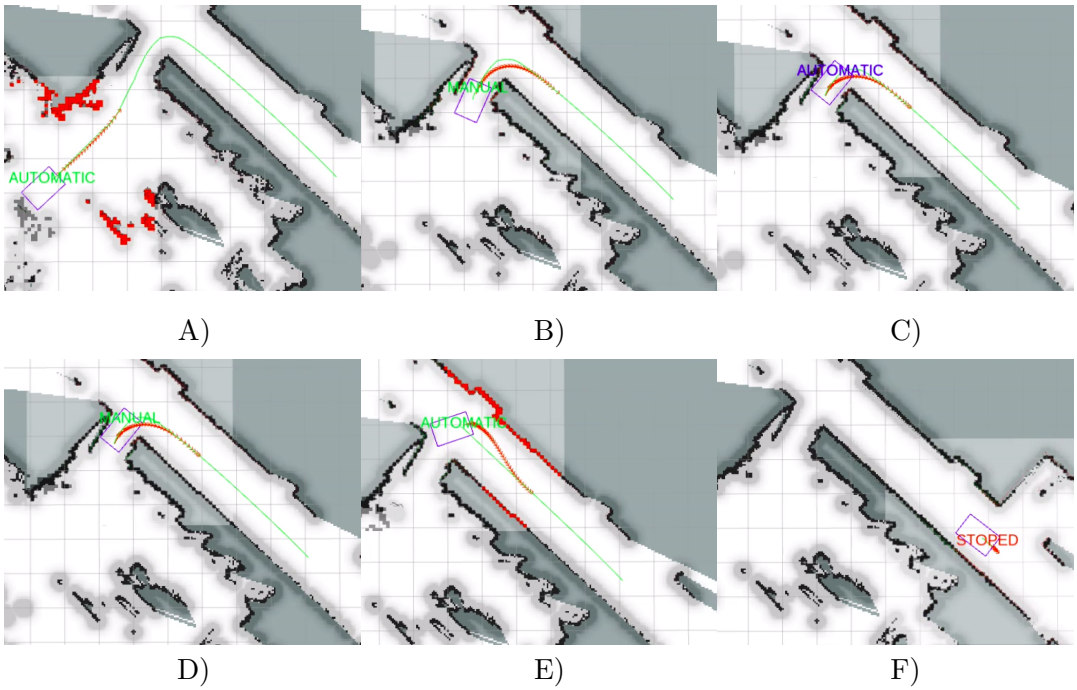


Figura 5.5: Áreas do teste de colisão. Nomeado de A) a F), representando cada letra o momento em que há alteração do modo de funcionamento.

até ao local E). Entre B) e E) o utilizador introduz uma velocidade considerada de risco no local C) sendo que o modo automático é activado. Como o nodo autónomo não foi capaz de determinar uma rota a plataforma imobiliza, continuando o seu percurso em modo manual em D). No local E) a plataforma segue o seu percurso em modo automático, chegando ao seu destino em F).

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusões

O principal objetivo deste trabalho - o controlo remoto semi-automático da plataforma ROBONUC - foi atingido. Mecanismos de segurança foram desenvolvidos para detetar potenciais colisões , permitindo atuar preventivamenteo. Após o trabalho desenvolvido na plataforma é possível navegá-la não só de forma manual, como também de modo semi-automático e automático.

O recurso ao ambiente ROS distribuído permitiu a integração de programas desenvolvidos por terceiros, sendo que foi possível implementar uma solução mais segura e completa do que seria possível caso o projeto fosse desenvolvido de raiz. A estrutura modular da plataforma permite também o constante melhoramento e adição de funções à plataforma.

A navegação autónoma é uma área que se encontra bastante desenvolvida em ambiente ROS sendo que foram aplicados pacotes ROS, como o `navigation` e o `teb_local_planner` que permitem realizar a navegação autónoma da plataforma de forma segura.

Foram introduzidas técnicas que permitem identificar a localização da plataforma. A hodometria é determinada através dos *encoders* presentes nas rodas. Esta informação é essencial para realizar a navegação autónoma para que esta conheça o deslocamento da plataforma, de modo a ajustar em tempo real a trajetória. A navegação autónoma também necessita de um mapa do local de modo a calcular a trajetória mais indicada. Para que seja possível navegar em novos ambientes o mapa é gerado em tempo real recorrendo a técnicas de SLAM. O pacote ROS HectorSLAM foi utilizado para este efeito.

Após a introdução de um modo automático, e com o sistema manual já presente na plataforma, foi criado um modo híbrido, o modo semi-automático. Este modo permite o controlo manual pelo utilizador e quando é detetado risco de colisão é ativado o modo automático que irá contornar o obstáculo. Para detetar risco de colisão foram definidas áreas de risco de dimensão variável. A área tem uma dimensão tanto maior quanto maior for a velocidade da plataforma, sendo proporcional à distancia de paragem da plataforma após a deteção de um obstáculo.

O sistema semi-automático também permite complementar limitações do sistema autónomo com a introdução de instruções do utilizador em zonas que não seja possível navegar em sistema autónomo.

O utilizador pode controlar a plataforma de forma remota, necessitado apenas de se encontrar conectado à mesma rede que a plataforma. O sistema dispõe de um controlador XBox

para operar a plataforma, a imagem de uma **web cam** instalada na frente da plataforma, um mapa do espaço onde o robô navega bem como a sua localização.

É de salientar que o presente trabalho foi desenvolvido tendo em conta o equipamento já presente na plataforma, sendo que não foi necessário investir em *hardware*, tendo sido apenas instalada uma *web cam*.

6.2 Trabalho Futuro

Os trabalhos futuros deverão seguir no sentido de integrar o manipulador na plataforma, ou seja, que estes trabalhem em conjunto e não com elementos separados. Para isso será necessário definir o *Unified Robot Description Format* (URDF) da plataforma e do manipulador. O URDF do manipulador já existe. Será necessário terminar a definição da plataforma e juntar ao do manipulador. Deste modo poderá ser evitado colisões do manipulador com a plataforma ou com os lasers, que se encontram no espaço de trabalho do manipulador.

O sistema de segurança deverá ser melhorado sempre que situações de risco sejam identificadas. O robô será utilizado em futuros projetos e poderá ser sujeito a condições que não foram previstas. Essas condições deverão ser identificadas e uma solução deverá ser aplicada para que a plataforma seja cada vez mais segura com o uso.

Como os lasers se encontram paralelos ao chão existem obstáculos que não são identificados. Objetos baixos, objetos suspensos e depressões como escadas e buracos não são identificados levando a situações de perigo. Sistemas adicionais, como sistemas de visão lasers 3D ou lasers 2D adicionais, deverão ser implementados de forma a fornecer à plataforma mais informação do espaço envolvente.

O sistema de controlo remoto deverá ser implementado com uma rede externa à plataforma. O *router* deverá encontra-se junto do utilizador, possivelmente ligado por cabo, e a plataforma conectar-se por "wi-f". Deste modo, será necessário configurar a plataforma para comunicar em simultâneo por "wi-fi"com o utilizador e por cabo com o CPU2.

A navegação está preparada para ser efetuada em ambientes novos. Deverá ser também preparada para realizar em ambientes conhecidos de modo mais eficiente. Poderá ser adicionado uma interface para adicionar mapas do local pelo utilizador, de modo a não ser necessário realizar o mapeamento em tempo real. Os nodos de navegação já estão preparados para receber mapas, mas será necessário alterar no nodo de hodometria para este se localizar no mapa sem o apoio do HectorSLAM.

Referências

[1] Oliver Brock. *Mobile Manipulation*. 2012. URL: `http://www.mobilemanipulation.org/index.php` (acedido em 17/05/2018).

[2] Robert Holmberg e Oussama Khatib. «Development and control of a holonomic mobile robot for mobile manipulation tasks». Em: *International Journal of Robotics Research* (2000).

[3] Wolfgang Merkt et al. «Robust shared autonomy for mobile manipulation with continuous scene monitoring». Em: *IEEE International Conference on Automation Science and Engineering*. IEEE, 2018.

[4] Bruno Carvalho Vieira. «Reconversão da Plataforma Robuter num AGV com Guiamento Visual». Em: *Dissertação de Mestrado, Univ. de Aveiro* (2007).

[5] Vítor Augusto Nogueira da Silva. «Integração de Manipulador FANUC na Plataforma Robuter para Manipulação Móvel». Em: *Dissertação de Mestrado, Univ. de Aveiro* (2017).

[6] Youcef Hargas et al. «Mobile manipulator path planning based on artificial potential field: Application on RobuTER/ULM». Em: *2015 4th International Conference on Electrical Engineering (ICEE)*. IEEE, 2015.

[7] A. Hentout et al. «Mobile Manipulation: A Case Study». Em: *Robot Manipulators New Achievements* (2010).

[8] A Hentout et al. «A Telerobotic Human / Robot Interface for Mobile Manipulators : A Study of Human Operator Performance». Em: (2013).

[9] B. H. Krogh. «A generalized potential field approach to obstacle avoidance control». Em: *World Conference on Robotics Research*. 1984.

[10] Oussama Khatib. «Real-Time Obstacle Avoidance for Manipulators and Mobile Robots». Em: *The International Journal of Robotics Research* 5.1 (1986).

[11] Selim Temizer. *Comparison of Two Navigation Methods*. URL: `http://people.csail.mit.edu/lpk/mars/temizer{\_}2001/Method{\_}Comparison/` (acedido em 16/05/2018).

[12] Andrey V. Savkin e Chao Wang. «A method for collision free assisted navigation of semi-autonomous vehicles in dynamic environments with moving and static obstacles». Em: *2015 10th Asian Control Conference: Emerging Control Techniques for a Sustainable World, ASCC 2015*. IEEE, 2015.

[13] Michael Hoy e Chao Wang. «Safe Robot Navigation Among Moving and Steady Obstacles». Em: (2017).

[14] MSI. *Overview for Cubi 2 — Desktop - The most versatile consumer pc — MSI Global.* URL: https://www.msi.com/Desktop/Cubi-2.html (acedido em 13/05/2018).

[15] Arduino. *Arduino Leonardo.* 2014. URL: https://store.arduino.cc/arduino-leonardo-ethhttp://arduino.cc/en/Main/arduinoBoardLeonardo (acedido em 13/05/2018).

[16] Arduino. *Arduino Micro.* 2014. URL: https://store.arduino.cc/arduino-microhttp://arduino.cc/en/Main/arduinoBoardMicro (acedido em 14/05/2018).

[17] Hokuyo. *URG-04LX-UG01 Product Details.* URL: https://www.hokuyo-aut.jp/search/single.php?serial=166 (acedido em 14/05/2018).

[18] Asus. *RT-AC51U.* URL: https://www.asus.com/pt/Networking/RTAC51U/.

[19] *Xbox 360 manuals and specifications.* URL: https://support.xbox.com/en-US/xbox-360/console/manual-specs (acedido em 23/06/2018).

[20] J Borenstein et al. «Mobile Robot Positioning & Sensors and Techniques». Em: *Journal of Robotic Systems, Special Issue on Mobile Robots* (1997).

[21] Veladri Kavati. «National Conference on Technological Advancements in Mechanical Engineering Trajectory Planning of a ...» Em: September (2016).

[22] Arduino © CC. *Arduino Playground - RotaryEncoders.* 2018. URL: https://playground.arduino.cc/Main/RotaryEncodershttp://www.playground.arduino.cc/Code/Filters (acedido em 24/04/2018).

[23] *navigation/Tutorials/RobotSetup/Odom - ROS Wiki.* URL: http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom (acedido em 07/03/2018).

[24] Arduino. *Arduino - Reference.* 2016. URL: https://www.arduino.cc/reference/en/language/variables/data-types/float/https://www.arduino.cc/en/Reference/HomePage (acedido em 07/05/2018).

[25] National Oceanic US Department of Commerce e Atmospheric Administration. *What is LIDAR.* 2014. URL: https://oceanservice.noaa.gov/facts/lidar.htmlhttp://oceanservice.noaa.gov/facts/lidar.html (acedido em 07/05/2018).

[26] *hokuyo_node - ROS Wiki.* URL: http://wiki.ros.org/hokuyo{_}node (acedido em 08/05/2018).

[27] *iralabdisco/ira_laser_tools: All laser type assemblers and manipulators.* URL: https://github.com/iralabdisco/ira{_}laser{_}tools (acedido em 08/05/2018).

[28] Joao Machado Santos, David Portugal e Rui P Rocha. «An evaluation of 2D SLAM techniques available in Robot Operating System». Em: *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics.* 2013.

[29] Kamarulzaman Kamarudin et al. «Performance analysis of the microsoft kinect sensor for 2D simultaneous localization and mapping (SLAM) techniques». Em: *Sensors (Switzerland)* (2014).

[30] *Navigation stack test: GMapping vs Hector Slam — Geduino Foundation.* URL: http://www.geduino.org/site/archives/36 (acedido em 08/05/2018).

[31] Matt Young. *ROS/NetworkSetup - ROS Wiki.* URL: http://wiki.ros.org/ROS/NetworkSetup (acedido em 21/05/2018).

[32] Jan Debiec. *ROS/Tutorials/MultipleMachines - ROS Wiki*. URL: http://wiki.ros.org/ROS/Tutorials/MultipleMachines (acedido em 21/05/2018).

[33] Genius. *Ultra wide Full HD webcam*. URL: http://us.geniusnet.com/product/widecam-f100 (acedido em 30/05/2018).

[34] *kinetic/Installation/Ubuntu*. URL: http://wiki.ros.org/kinetic/Installation/Ubuntu (acedido em 30/05/2018).

[35] Christoph Rösmann. *teb_local_planner - ROS Wiki*. 2016. URL: http://wiki.ros.org/teb{_}local{_}planner (acedido em 22/05/2018).

[36] Matthias Gruhler. *navigation*. URL: http://wiki.ros.org/navigation.

[37] Jo Yung Wong. *Theory of ground vehicles*. Wiley, 2008.