



Localização de um AGV em Ambiente Industrial Através de um Laser de Segurança e Marcadores Artificiais

André dos Santos Granjo Oliveira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: António Paulo Moreira (Prof. Associado do DEEC-FEUP)

Co-Orientador: Heber Sobreira (Investigador do INESC TEC)

29 de Outubro de 2013

Resumo

Nos últimos anos a competitividade entre empresas da área da indústria tem aumentado a um nível de tal forma elevado, que em todos os elementos envolvidos na produção, são procuradas o máximo de vantagens competitivas.

Assim, com o desenvolvimento do conhecimento dos pilares da robótica móvel e com a evolução da tecnologia, os AGV's cada vez mais se tornam uma solução mais atrativa. Cada vez é maior o número de empresas que olham para os AGV's como uma vantagem competitiva que reduz os custos de mão de obra desnecessária, em tarefas monótonas e repetitivas que não acrescentam valor ao produto final. Desta forma, este tipo de soluções deve ser capaz de calcular a sua localização de forma a serem verdadeiramente autónomos.

As soluções mais comuns são de filoguiado (fio enterrado no piso) e seguimento de faixa (faixa disposta no piso), tendo estas a desvantagem de seguirem uma trajetória fixa, que possui custos acrescentados sempre que seja necessária a sua alteração. Torna-se então necessário o aumento da flexibilidade das trajetórias tornando-as em caminhos dinâmicos possíveis de alterar a qualquer momento, com o menor custo e a maior rapidez possíveis.

Em alternativa às anteriores, existem soluções de localização absoluta por triangulação e trilateração, baseadas num laser de 360° de deteção de balizas e o obrigatório laser de segurança. Estas são capazes de seguir caminhos dinâmicos, tendo como desvantagem a obrigatoriedade de existirem um mínimo de uma baliza em três dos quatro quadrantes do referencial e o preço relativamente elevado da tecnologia, uma vez que implica o uso de dois dispendiosos laser.

Este trabalho tem como objetivo desenvolver uma solução industrial, mais económica, robusta de fácil configuração e reconfiguração, alternativa às soluções de localização absoluta por triangulação e trilateração.

É proposto desenvolver um sistema que permita a utilização de caminhos dinâmicos unicamente com a informação proveniente do laser de segurança e do deslocamento das rodas. Uma vez que o laser de segurança é um elemento obrigatório por lei, com a presente solução é proposta a eliminação de um segundo laser de localização da plataforma robótica reduzindo significativamente o custo final.

A solução desenvolvida baseia-se na fusão entre a informação do posicionamento das balizas, proveniente do laser de segurança SICK 3000S, e da odometria, proveniente do encoder da roda motriz. Foi também desenvolvido o pré-processamento da informação proveniente do laser por forma a evitar a deteção de falsos positivos. Por fim foi desenvolvido um algoritmo, que permitisse fazer o mapeamento do espaço envolvente ao AGV, de nome EKF-SLAM.

Abstract

Over the last years the competition between industrial companies has increased to the point where all the little details has to be taken in consideration in order to find the best advantage possible.

The increase of mobile robotics knowledge and the evolution of technologies make AGV a more attractive solution. More and more companies consider the AGV as a competitive and advantaged alternative that reduces the unnecessary costs in monotonous and repetitive labour actions that do not add value to the final product. So in order to use this solution, AGVs have to be fully capable of autonomously calculate their own location.

Most common solutions are wire-guided (wire under the ground) and follow a track (track over the ground); They have the disadvantage of being limited to fixed trajectories and, when changes in path are needed, it results in increased costs; for these reasons it becomes compulsory an improving flexibility in the trajectory definition, making them dynamic and capable of being changed at any moment, with small cost and quick responses.

The goal of this project is to develop an industrial solution that is cheaper, robust, easy configurable and reconfigurable, that represents the alternative to absolute localization by triangulation and trilateration that's based on 360° lasers and passive indistinguishable beacons.

The aim is to develop an absolute localization system that allows the usage of dynamic paths solely with the information derived from the security laser and the offset of the wheels. As the security laser is an obligatory element by law, with the present solution it is proposed to eliminate the 360° laser using solely the security laser capability of detecting beacons to orientate the AGV. For these reasons it is expected a reduction both in the number of lasers as in the final cost.

The developed solution it is based on the fusion of information, using the position of beacons derived from the security laser scan SICK 3000S, and the odometry derived from the encoder positioned in the traction wheel. It was also developed the pre-processing of the data, derived from the security laser scan, in order to exclude false positive beacons. Finally it was developed an algorithm that allowed the mapping of the beacons surrounding the AGV, with the name of EKF-SLAM.

Agradecimentos

Agradeço com especial carinho aos meus Pais, por tudo o que fizeram por mim e por todos os valores que me inculcaram e me servirão de pilares para toda a vida, e à minha irmã por toda a paciência que sempre demonstrou para comigo.

Ao aluno de doutoramento Heber Sobreira, ao Prof. Dr. António Paulo Gomes Mendes Moreira e a todos os colaboradores do Grupo Robis, agradeço toda a ajuda prestada na concretização deste projeto.

E por fim aos grandes amigos que me acompanharam ao longo de toda a minha vida de estudante, agradeço por todos os momentos inesquecíveis que me proporcionaram.

André Granjo Oliveira

*“Live as if you were to die tomorrow. Learn as
if you were to live forever.”*

Mahatma Gandhi

Conteúdo

1	Introdução	1
1.1	Enquadramento e Motivação	1
1.2	Objectivos	2
1.3	Estrutura do Documento	2
2	Estado da Arte	5
2.1	Sistemas de Localização	5
2.1.1	Filoguiado	5
2.1.2	Seguimento de faixas	6
2.1.3	Marcadores	7
2.1.4	Trilateração e Triangulação com Balizas	8
2.1.5	Odometria	9
2.2	Algoritmos de Localização	10
2.2.1	Filtro de Kalman	10
2.2.2	Filtro de Partículas	14
3	Plataforma experimental e Arquitetura do Software	15
3.1	Plataforma Experimental	15
3.1.1	Plataforma do veículo	16
3.1.2	Laser Range Finder	17
3.2	Arquitetura do Software	19
3.2.1	Camada ROS	19
3.2.2	Camadas desenvolvidas	21
4	Algoritmo de localização baseado no <i>Extended Kalman Filter</i>	25
4.1	Introdução ao Algoritmo de localização baseado em EKF	25
4.1.1	Previsão	27
4.1.2	Observação	29
4.1.3	Previsão da Medição	30
4.1.4	Associação	31
4.1.5	Atualização	31
4.2	Parâmetros do Filtro	32
5	Localização e Mapeamento Simultâneo	35
5.1	Introdução	35
5.2	Nomenclatura do Algoritmo de SLAM-EKF	35
5.3	Ciclo do Algoritmo de SLAM-EKF	38
5.3.1	Previsão	40

5.3.2	Associação	41
5.3.3	Atualização	42
5.3.4	Passagem de EKF-SLAM para EKF	46
5.4	Parâmetros do Filtro	46
6	Resultados Práticos	49
6.1	Introdução	49
6.2	Resultados Práticos do Algoritmo de EKF	50
6.3	Resultados Práticos do Algoritmo de EKF	56
7	Conclusão e Trabalhos Futuros	57
7.1	Conclusão	57
7.2	Trabalhos Futuros	57
	Referências	59

Lista de Figuras

2.1	Sistema Filoguiado	6
2.2	Sistema de seguimento de faixas	6
2.3	Sistema de localização com marcadores [2]	7
2.4	Sistema de localização por Triangulação	8
2.5	Modelo de Movimento da Odometria	9
2.6	Diagramas do variantes do SLAM, X representa a pose, u representa o sinal de controlo, z representa a medida e por fim m representa o mapa [3]	13
3.1	À esquerda pode-se ver a simulação do veículo e à direita o veículo real.	15
3.2	(a)À esquerda encontra-se o encoder Drehgeber Serie 85; (b)No centro encontra-se a estrutura que suporta a roda e o seu movimento Schabmueller S2; (c)À direita encontra-se o encoder LIKA I58H	16
3.3	A vermelho esta representado o caminho a percorrer, os blocos azuis representarão as balizas mapeadas, e os círculos azuis são elementos dinâmicos que permitem a alteração do caminho	17
3.4	(a) À esquerda representa-se a deteção de balizas; (b)No centro representa-se as áreas de segurança a deteção de contornos e de balizas; (c) À direita encontram-se representadas as áreas de segurança e os atuadores aplicados nas rodas.	17
3.5	(a) À esquerda encontra-se o laser SICK 3000S; (b) À direita encontra-se o exemplo de uma baliza.	18
3.6	A vermelho está representada a baliza processada pela drive do laser. A preto e amarelo esta a informação bruta proveniente do laser que correspondem a elementos não refletos e refletos, respetivamente	18
3.7	Exemplo arquitetura ROS tipo publicador subscritor	19
3.8	Esta figura exemplifica o tipo de mensagens que podem aparecer.	20
3.9	Representação da informação proveniente de um laser 3d no rviz	21
3.10	Diagrama de camadas do Software	21
3.11	Ambiente gráfico de Debug	22
3.12	Evolução do EKF ao longo do tempo	23
3.13	Interface gráfico	23
4.1	Referencial do mundo e referencial do veículo	25
4.2	Diagrama do Algoritmo EKF	27
5.1	Referencial do mundo e referencial do veículo	36
6.1	Planta do local de Testes	49
6.2	Mapeamento da sala I -108 e -109. A vermelho está representado o percurso definido, a amarelo estão representados as Balizas mapeadas	50

6.3	Representação da assimetria no posicionamento das balizas.	51
6.4	Posicionamento das balizas	51
6.5	Coordenadas das balizas	52
6.6	Marca de Pose inicial	52
6.7	Plot dos erro entre o algoritmo EKF desenvolvido e o algoritmo de Perfect match .	53
6.8	Marcas de posição inicial.	53
6.9	Marcas de análise, presentes na zona 3.	54
6.10	Desenho do percurso percorrido pelo AGV com os diferentes algoritmos. A amarelo é descrito o algoritmo de EKF , a vermelho o algoritmo de <i>Perfect Match</i> . As bolas vermelhas representam o mapeamento das balizas e as bolas azuis são as balizas visualizadas no momento	54
6.11	Erro da trajetória na zona 2	55
6.12	Erro da posição na marca de Pose inicial	55
6.13	Marcas de análise, presentes na zona 3.	56
6.14	Covariância da pose. O retângulo a azul representa o robô e a elipse contida nele representa a covariância	56

Lista de Tabelas

4.1	Valores de erro do Laser	32
4.2	Valores de erro do Laser	32
5.1	Valores de erro do Laser	47
5.2	Valores de erro do Laser	48

Abreviaturas e Símbolos

EKF	Extended Kalman Filter
AGV	Autonomous Guided Vehicle
SLAM	Simultaneous Localization and Mapping
MGS	Magnetic Sensor Guidance
DEEC	Departamento de Engenharia Eletrotécnica e Computadores
FEUP	Faculdade de Engenharia da Universidade do Porto
ROS	Robot Operating Sistem
GUI	Grafical User Interface

Capítulo 1

Introdução

1.1 Enquadramento e Motivação

A indústria é uma área em constante desenvolvimento e consequente mudança, onde estão envolvidos diversos processos de produção que originam a todo o momento novos problemas de engenharia de elevada complexidade.

Hoje em dia a elevada concorrência no mercado passa um pouco por todas as áreas, e a área da indústria é uma área que também não escapa à regra, surgindo assim, a necessidade de encontrar soluções competitivas e económicas para estes mesmos problemas de elevada complexidade. Esta realidade impõe uma enorme pressão do lado da indústria exigindo assim, de forma a manter a competitividade, uma elevada rapidez na resposta e no cumprimento dos desafios propostos a um custo de produção reduzido, com velocidade e flexibilidade elevada, ou seja, a capacidade de mudança sem perda de capacidade produtiva. Uma das vias para atingir tal feito na área da indústria passa claramente pela automação inteligente dos processos sendo uma das suas vertentes mais importantes a robótica.

Grande parte das organizações industriais que hoje se encontram em posições confortáveis do mercado aperceberam-se destes fatores e é por isso que os veículos autonomamente guiados, mais conhecidos por AGVs (Automated Guided Vehicles), são uns dos robôs mais comuns em ambientes industriais.

Os AGVs têm o objetivo de reduzir a intervenção humana em sistemas flexíveis de fabrico, permitindo não só, a possibilidade de dispensa de trabalhadores de tarefas monótonas e repetitivas que não acrescentam valor ao produto final, mas também uma componente que nos dá a possibilidade de aumentar o grau de automação e flexibilidade, facilitando a integração total e o controlo otimizado do sistema. Desta forma conseguimos atingir um fluxo otimizado e contínuo de materiais através do escalonamento de tarefas, contribuindo para uma melhor produtividade global do sistema.

Contudo, para além das reconhecidas vantagens que o aparecimento de robots móveis trouxe consigo, também houve o aparecimento de inúmeros desafios tecnológicos como é exemplo:

- A conceção de veículos autónomos, alimentados a partir de baterias de tração, recarregáveis com a maior autonomia possível.
- Desenvolvimento de sistemas de navegação autónomos, que permitem o seu deslocamento autónomo e eficiente.
- Controlo de frotas constituídas por vários robôs móveis.

Passa por um dos objetivos deste trabalho o desenvolvimento de um AGV que seja capaz de seguir trajetórias dinâmicas com um algoritmo de localização que utilize o Laser de segurança. Com este objetivo visa-se criar grande facilidade na modificação do layout do AGV com rapidez e custos reduzidos.

Nesta dissertação a aplicação deste objetivo passará pela instalação de balizas passivas indistinguíveis que ajudarão o AGV na sua localização sendo assim possível o seguimento da sua trajetória. A aplicação deste objetivo tendencialmente reduzirá o preço do AGV quando comparado com soluções similares como é o caso de soluções baseadas em triangulação e/ou trilateração, uma vez que existe uma redução do número de lasers.

Para concluir passará também pelos objetivos desta dissertação o aumento da robustez na deteção das balizas refletoras, aumentando assim a robustez de todo o algoritmo de localização possibilitando a sua aplicação em todo o tipo de ambientes industriais.

1.2 Objectivos

Principal Esta dissertação tem como objetivo desenvolver e testar um sistema de localização de AGV's baseado no seu laser de segurança e em balizas artificiais.

Complementares

1. Implementação da aplicação informática que comunica com o sensor laser.
2. Localização ótima das balizas.
3. Mapeamento das balizas.

Extra Desenvolvimento de um Algoritmo de localização e mapeamento simultâneo.

1.3 Estrutura do Documento

Este documento, desenvolvido no âmbito da unidade curricular de Dissertação do 5º ano do Mestrado Integrado em Engenharia Electrotécnica e de Computadores, encontra-se dividido em 7 capítulos e pretende relatar todo o trabalho desenvolvido neste projecto.

No capítulo 1 é feita a introdução ao tema e ao documento. É descrita a caracterização detalhada de todo o problema a tratar e são explicados os objetivos propostos para esta dissertação.

Todo o estudo desenvolvido com o objetivo de consolidar conhecimentos e perceber as tecnologias já existentes está presente no capítulo 2. Neste capítulo são apresentadas as soluções já existentes e mais utilizadas no que toca a sistemas de localização. São também apresentados alguns algoritmos e conceitos base relativos a sistemas de localização.

O capítulo 3 é preenchido pelos principais componentes integrantes do robot. Nele é descrita tanto a parte de hardware como de software.

O capítulo 4 é dedicado ao algoritmo de localização de nome *Extended Kalman Filter* e nele são explicados todos os princípios envolventes ao processo e é feita uma descrição de todos os blocos envolventes ao algoritmo.

O capítulo 5 tem a mesma estrutura do capítulo 4 no entanto, descreve o algoritmo de localização e mapeamento de nome *Extended Kalman Filter-Simultaneous Localization and Mapping*.

No capítulo 6 são apresentados todos os resultados práticos.

Finalmente, no capítulo 7, são apresentadas as conclusões finais e sugestões de trabalho futuro.

É também importante referir a independência entre o capítulo 4 e 5. Ou seja, se o objetivo do estudo for unicamente o algoritmo de SLAM (capítulo 5) não será necessária a leitura do capítulo 4 para um bom entendimento deste.

Capítulo 2

Estado da Arte

Neste capítulo pretende-se fazer uma revisão bibliográfica focando nas características de AGVs já existentes. São referidos e explicados alguns dos sistemas de localização mais utilizados. E, por fim é feita uma abordagem sobre alguns algoritmos de localização.

2.1 Sistemas de Localização

A localização de robôs moveis é um tema já extensamente explorado e com uma base de conhecimento já alargada. Desta forma é normal a existência de inúmeras soluções que diferem principalmente na área da tecnologia e da aplicabilidade.

A escolha da aplicação a instalar deve ter em conta a precisão da localização, as condições do local da instalação, os custos finais e a flexibilidade requerida pelo problema.

Existem diversos métodos de localização que permitem seguir um caminho fixo ou dinâmico. A determinação na escolha de caminhos fixos ou dinâmicos recai nos custos de instalação, nos requisitos de flexibilidade e na necessidade ou não de futura expansão do sistema.

Os caminhos fixos, solução mais comum no mercado [1], implicam menores custos. No entanto apresentam-se como uma solução com menor flexibilidade impossibilitando assim reação rápida quando existe a necessidade de mudanças no layout.

Os caminhos dinâmicos, permitem maior flexibilidade, contudo são sistemas mais pesados e com custos de implementação mais elevados.

Existem diferentes soluções com diferentes custos associados. Na presente secção serão abordados alguns dos sistemas mais usados no mercado.

2.1.1 Filoguiado

O sistema de filoguiado, baseia-se, como representado na figura 2.1a [2], no seguimento por parte do robô de um cabo enterrado no pavimento da área de trabalho.

Este fio é um condutor percorrido por uma corrente elétrica alternada criando assim um campo elétrico em torno do mesmo, como representado na figura 2.1b [2].

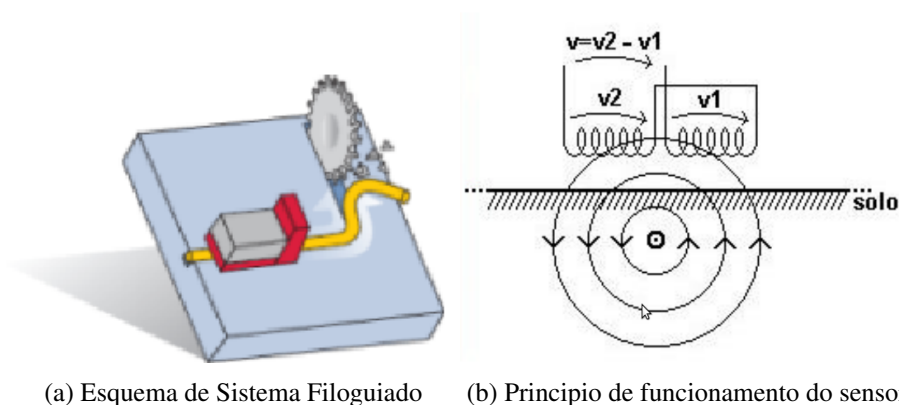


Figura 2.1: Sistema Filoguiado

O sensor, que é aplicado num ponto estratégico do robô, quando está sobre a linha, apresenta na sua saída o valor zero. Este valor vai variando mediante o deslocamento e a orientação, sendo o objetivo do sistema corrigir constantemente a posição de forma a manter-se com valor de referência zero.

Assim, pode-se concluir que é um sistema robusto, simples e barato para soluções onde se pretenda um caminho fixo, visto que, desde que esteja sobre o fio estará sempre orientado. Contudo necessita de velocidades baixas de forma a nunca se perder do fio. A grande desvantagem de tal sistema passa pelos elevados custos para enterrar os condutores no solo e consequentemente custos na mudança do layout.

2.1.2 Seguimento de faixas

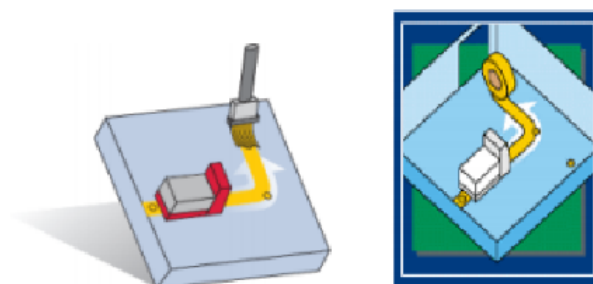


Figura 2.2: Sistema de seguimento de faixas

O seguimento de faixas é também uma solução com foco em trajetórias fixas, e em parte bastante idêntico ao filoguiado. O princípio de funcionamento desta técnica assenta, tal como ilustrado na figura 2.2 [2], no seguimento, por parte do robô, de uma linha. Essa linha pode ser pintada diretamente no solo ou criada através da colocação de fita magnética. Com o sensor apropriado este método funciona de forma muito idêntica ao anterior. Contudo, a grande vantagem deste tipo de conceito em relação ao filoguiado é que não existe a necessidade de efetuar um setup

tão moroso e complexo. Logo o custo e tempo a despendar numa alteração de percurso é bastante inferior. É dessa forma uma solução mais flexível que a solução de filoguiado.

O grande problema desta solução passa pela danificação da fita. Em grandes indústrias com máquinas de grande dimensão existe sempre a possibilidade de danificação da fita ou obstrução à sua correta visualização o que pode originar a desorientação do robô, uma vez que perde a única informação que lhe permite seguir a fita.

Contudo existem indústrias onde tal solução é possível de implementar aliando assim a robustez e simplicidade do filoguiado à facilidade de implementação e reconfiguração de um caminho, tornando-se assim um método de localização interessante.

2.1.3 Marcadores

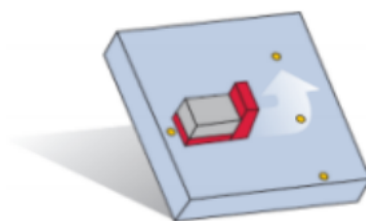


Figura 2.3: Sistema de localização com marcadores [2]

O princípio deste método de localização assenta no seguimento, por parte do robô, de uma trajetória em que a informação não se encontra sempre disponível.

Ou seja ao longo de um percurso quando o robô encontra um marcador retira informação do mesmo, sendo esta informação relativa à sua posição atual (corrigindo assim a sua localização) e por vezes traça nova trajetória para o próximo marcador, deslocando-se de seguida ao longo de uma distância sem mais nenhuma informação, até que novo marcador seja encontrado.

Os marcadores podem ser etiquetas magnéticas, refletos, RF passivas, formas geométricas ou até mesmo códigos de barras.

Nesta técnica caminhos são definidos com pequenos marcadores encontrados/colocados ao longo do percurso do robô ligando entre si uma trajetória imaginária.

Devido ao facto de os marcadores se encontrarem distanciados entre si, existem momentos onde o robô se encontra a deslocar sem nenhuma informação do ambiente em seu redor. Assim surge a necessidade de a esta técnica de localização absoluta juntar outras técnicas, conhecidas como técnicas de localização relativa, de forma a arranjar um pouco mais de informação ajudando o robô a encontrar o próximo marcador. A fusão da localização absoluta e relativa é possível através de métodos probabilísticos como é exemplo do Filtro de Kalman e do Filtro de Partículas.

É então perceptível que esta solução é versátil, flexível, simples, rápida e consequentemente de menor custo que as anteriormente referidas. Contudo está dependente da informação proveniente dos marcadores, os quais não estão sempre disponíveis ao longo do tempo. Tornando assim a

localização intermitente. Tal facto torna esta solução pouco robusta sendo passível do robô se perder dados os possíveis erros odometricos provenientes de deslizos das rodas.

2.1.4 Trilateração e Triangulação com Balizas

Trilateração e Triangulação com Balizas (Figura 2.4 [2]) é uma técnica com foco em trajetórias dinâmicas. É a solução mais flexível de todas as apresentadas até agora, sendo por isso muito utilizada na indústria quando esse aspeto é crucial. O princípio de funcionamento deste tipo de localização assenta na premissa de que existe um conjunto de balizas, ativas ou passivas, como emissores ou refletos ultra-sons, laser, IR, código de cores, etc, dispostas ao longo de toda a zona de movimentação do robô, com a ajuda das quais o robô se localiza.

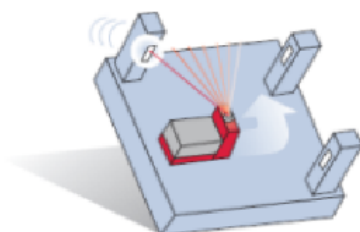


Figura 2.4: Sistema de localização por Triangulação

Para que o robô se localize é necessário que estejam visíveis, a cada momento, um número mínimo de três balizas divididas por três dos quatro quadrantes do referencial do laser, sendo aconselhável cinco ou mais. Assim, de forma a existir uma visão o mais desobstruída possível das balizas, estas são normalmente fixadas estrategicamente a uma distância do solo, em colunas, pilares, paredes, etc.

É necessário que o robô esteja equipado com um sensor capaz de identificar as balizas, e que este efetue varrimentos rotativos sucessivos. Assim, sabendo o posicionamento das balizas à priori, torna-se fácil, através de métodos de triangulação e trilateração obter o posicionamento global do robô.

Desta forma percebe-se que, atendendo à condição acima citada, em qualquer momento o robô se consiga localizar sem necessidade de auxílio, e que nunca se perca ao contrário de métodos de localização baseada em marcadores, onde é necessário em certos momentos, estimar a sua posição com recurso a métodos auxiliares.

As principais vantagens deste método são a sua grande precisão, flexibilidade, robustez e velocidade de movimentação do veículo.

A desvantagem deste método associa-se à necessidade inicial de preparação do espaço de movimentação do robô para a aplicação desta técnica (a menos de que o espaço possua já algo passível de ser identificado como baliza como é exemplo de cantos e paredes), O que implica um custo inicial mais elevado e a necessidade de um bom planeamento da disposição das balizas.

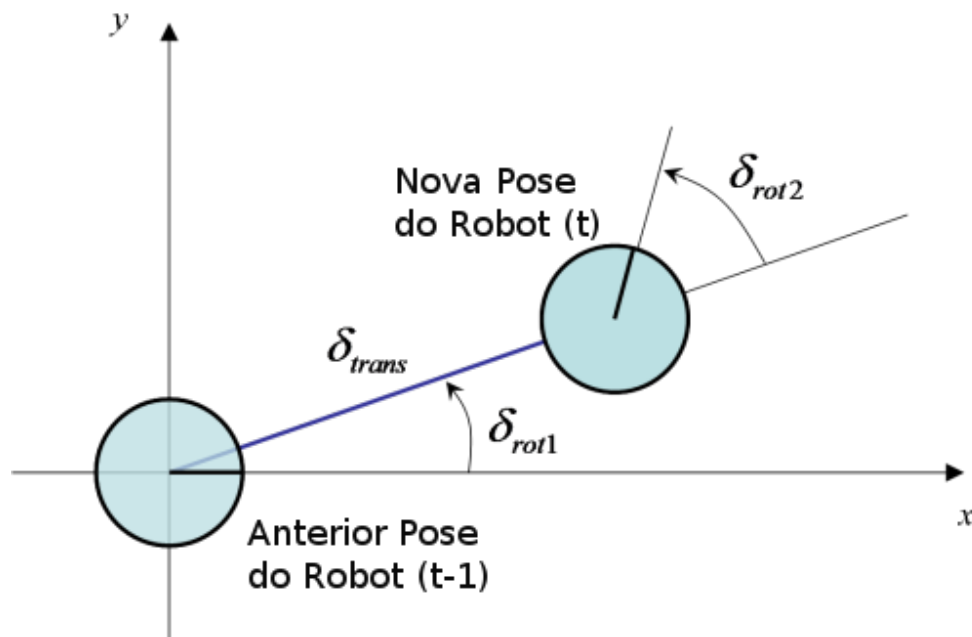


Figura 2.5: Modelo de Movimento da Odometria

2.1.5 Odometria

Esta é uma técnica de localização relativa acompanhada por um erro não limitado, sendo usual usar esta técnica em conjunto com outras referidas anteriormente, de forma a minimizar a incerteza. A odometria é uma maneira de obter a pose de um robô no momento $t + 1$ a partir da sua pose anterior t através da integração de comandos de velocidade num espaço definido de tempo ou a partir da informação proveniente dos encoders das rodas.

Estes dois modelos diferem principalmente entre a grandeza do seu erro. Embora a obtenção da odometria através de comandos de velocidade permite-nos saber facilmente a próxima posição, é acompanhada por erros associados ao modelo de velocidade e a erros de deslizamento. A odometria baseada em encoder tem provado ser a mais fiável e acompanhada de menor erro. Pois uma vez os encoders utilizados atualmente serem de alta precisão este modelo fica unicamente afetado por erros de deslizamento, e pelo pequeno erro do encoder.

Hoje em dia a maior parte das arquiteturas fornecidas pelo mercado disponibiliza o deslocamento em x, y e θ periodicamente (em intervalos de 10ms). [3]

2.1.5.1 Modelo de Movimento Utilizando a Odometria

Na odometria baseada em encoders a posição só se encontra disponível depois do movimento.

Para tal foi desenvolvido um modelo para o processamento da próxima pose, tendo este o nome de *Odometry Motion Model* [3]. Este modelo consiste em dividir o movimento em duas rotações e uma translação como exemplificado na figura 2.5 [3]. É feita uma rotação inicial entre a pose anterior($t-1$) e a linha de menor caminho entre as coordenadas anteriores e as coordenadas

atuais(δ_{rot1}). De seguida faz-se a translação entre os pontos referidos anteriormente e, por fim uma rotação para o ângulo final.

As próximas equações demonstram como é feito o cálculo das duas rotações e a translação das leituras da odometria entre $\hat{x}_{t-1} = (\hat{x} \ \hat{y} \ \hat{\theta})$ e $\hat{x}_t = (\hat{x}' \ \hat{y}' \ \hat{\theta}')$:

$$\delta_{rot1} = atan2(\hat{y}' - \hat{y}, \hat{x}' - \hat{x}) - \hat{\theta} \quad (2.1)$$

$$\delta_{trans} = \sqrt{(\hat{x} - \hat{x}')^2 + (\hat{y} - \hat{y}')^2} \quad (2.2)$$

$$\delta_{rot2} = \hat{\theta}' - \hat{\theta} - \delta_{rot1} \quad (2.3)$$

Assim sendo podemos calcular o deslocamento em x, y e θ através do seguinte modelo

$$f_v = \begin{bmatrix} \delta_{trans} \cos(\theta + \delta_{rot1}) \\ \delta_{trans} \sin(\theta + \delta_{rot1}) \\ \delta_{rot1} + \delta_{rot2} \end{bmatrix} \quad (2.4)$$

Tem-se que a posição atual, x_t , é obtida através de x_{t-1} , ou seja obtém-se a posição atual somando o deslocamento obtido pelo modelo do sistema à posição anterior:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \delta_{trans} \cos(\theta + \delta_{rot1}) \\ \delta_{trans} \sin(\theta + \delta_{rot1}) \\ \delta_{rot1} + \delta_{rot2} \end{bmatrix} \quad (2.5)$$

Cada vez que o veículo se desloca este deslocamento é acompanhado de erros provenientes de diversos fatores como é exemplo o erro do encoders ou deslizamento das rodas. É então necessário ao modelo previamente apresentado adicionar um erro gaussiano de média zero, tendo então:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \delta_{trans} \cos(\theta + \delta_{rot1}) \\ \delta_{trans} \sin(\theta + \delta_{rot1}) \\ \delta_{rot1} + \delta_{rot2} \end{bmatrix} + q_v \quad (2.6)$$

onde q_v se decompõe da seguinte forma:

$$q_v = \begin{bmatrix} \varepsilon_{xv} \\ \varepsilon_{yv} \\ \varepsilon_{\theta} \end{bmatrix} \quad (2.7)$$

2.2 Algoritmos de Localização

2.2.1 Filtro de Kalman

Normalmente os robôs possuem um leque alargado de sensores no seu sistema, contribuindo estes com diversos tipos de informação com diferente precisão. O objetivo do filtro de Kalman

consiste em fundir toda esta informação de forma, a obter a melhor estimativa possível da pose do robô. De facto, este mecanismo é mais eficiente computacionalmente que o filtro de partículas [4]. Assim ao contrário do filtro de partículas, o filtro de Kalman é menos robusto devido à suas simplificações a quando da representação da densidade das suas probabilidades no determinar da confiança de uma medida. É um filtro ideal para ser usado em modo on-line.

O filtro de Kalman calcula a estimativa do estado atual a partir de uma função multidimensional gaussiana. No momento t , a estimativa do estado é representada pela média, μ , que é o valor que melhor a define e pela covariância, Σ , que é a confiança naquela estimativa. Para que o filtro de Kalman possa ser aplicado têm de se garantir as seguintes condições:

- A função de transferência é linear e que o ruído gaussiano é de media nula. Como expresso na seguinte equação:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

- O modelo de entradas é linear e com ruído gaussiano de media nula

As condições de que as observações são dadas por funções linear do estado e que o próximo estado é dada pela função linear do estado anterior são condições necessárias para o bom funcionamento do filtro de Kalman. Infelizmente, na prática as transições de estado e as medições são raramente lineares. Por exemplo um robô que se move com velocidade translacional e velocidade rotacional constantes tipicamente mover-se-á numa trajetória circular, a qual não pode ser definida por um função linear.

2.2.1.1 Filtro Extendido de Kalman

O filtro estendido de Kalman, que ao longo deste trabalho será denominada pelo seu nome em inglês *Extendend Kalman Filter (EKF)*, relaxa na premissa das linearidades. Aqui a premissa é que a estimação do estado e a estimação das medidas são dadas por funções não linear.

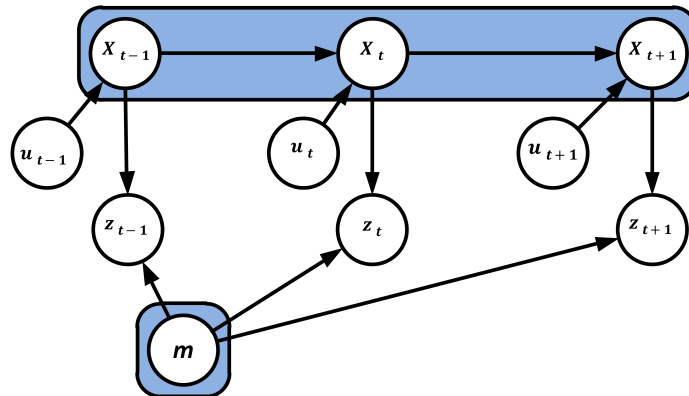
2.2.1.2 Simultaneous Localization and Mapping (SLAM)

Os problemas de SLAM aparecem quando o veículo não tem acesso ao mapeamento da área, nem á sua própria posição [5]. Sem nenhum conhecimento do espaço circundante, o SLAM permite ao veiculo indicar a sua posição global no referencial e ao mesmo tempo mapear o espaço envolvente.

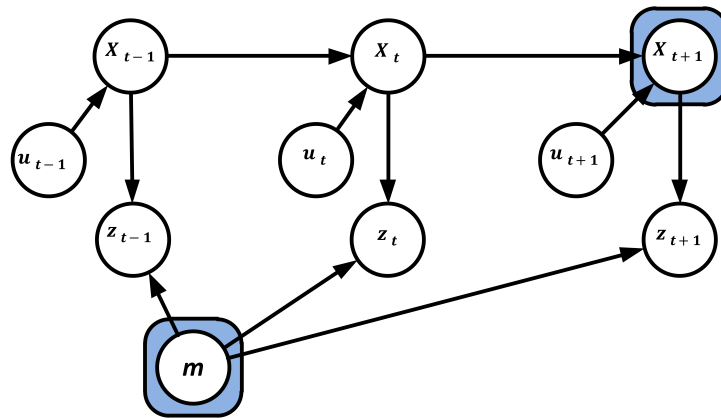
O mapa formado é um mapa com a informação da posição global, em x e em y , dos marcadores presentes na área circundante do veiculo e do desvio padrão de cada. Num segundo passo a ideia é utilizar a informação proveniente do SLAM e utilizar o algoritmo de EKF descrito no capítulo 4 de forma a localizar o veículo com precisão.

De um ponto de vista probabilístico existem dois tipos de problemas baseados no SLAM, sendo os dois de igual importância. Um deles é conhecido como *Online SLAM* e envolve estimar a pose posterior, considerando a pose atual, em simultâneo com o mapeamento dos marcadores

que rodeiam o veículo. O segundo SLAM é denominado de full SLAM e procura calcular a posição posterior, considerando todo o percurso percorrido, em simultâneo com o mapeamento dos marcadores.



(a) Full SLAM



(b) Online SLAM

Figura 2.6: Diagramas dos variantes do SLAM, X representa a pose, u representa o sinal de controle, z representa a medida e por fim m representa o mapa [3]

O Algoritmo de Online SLAM envolve estimar a posição no momento t , x_t , e em conjunto com o mapeamento das balizas. Este algoritmo é incremental e como tal descarta medidas depois de terem sido processadas. O modelo gráfico deste algoritmo é descrito em na figura 2.6b.

O algoritmo de Full SLAM procura o cálculo da estimativa da próxima pose considerando todo o percurso percorrido, $x_{1:t}$, tal é executado em conjunto com o mapeamento. O modelo gráfico deste algoritmo é descrito em na figura 2.6a.

Esta pequena diferença entre o Online SLAM e o Full SLAM cria ramificações ao nível do algoritmo, de tal ordem grandes, que são considerados algoritmos completamente diferentes. Ou seja, no Full SLAM sempre que é feita uma nova atualização do filtro são atualizados todos os dados adquiridos até ao momento, ao contrário do Online SLAM onde só é atualizado o estado do filtro.

2.2.2 Filtro de Partículas

O filtro de partículas representa a crença posterior por um conjunto de amostras aleatórias do estado do sistema. Essa representação é aproximada, porém como ela é não paramétrica, ela apresenta a possibilidade de representar um número muito maior de distribuições além da Gaussiana.

Como as partículas representam a distribuição da crença posterior, próximo ao valor esperado dessa distribuição teremos uma região com alta concentração de partículas, ao passo que ao nos afastarmos desse valor, essa concentração irá diminuir.

O filtro de partículas recebe como entrada um conjunto de partículas representando a crença sobre o estado do sistema no momento $t - 1$, juntamente com os controles u no momento t e as medições z também no momento t .

Então, criamos um novo conjunto de partículas $\hat{X}_t$, utilizando as partículas do conjunto X_{t-1} aplicando o modelo que simula o efeito da ação u_t e adicionando um ruído aleatório para representar a incerteza do resultado da ação.

Na sequência, realizamos as medições e atribuem-se pesos a cada partícula de acordo com a probabilidade da medição z_t sobre a partícula x_t .

Assim dado este conceito tão genérico de localização, torna-se necessário um mecanismo poderoso que calcule a estimativa da próxima posição em função da nova informação (proveniente de encoders e outros sensores) incorporada com a estimativa da pose anterior.

Capítulo 3

Plataforma experimental e Arquitetura do Software

Neste capítulo vão ser apresentadas todas as características tanto da parte do hardware como do software. Inicialmente serão apresentadas as características de todos os elementos constituintes da plataforma usada.

De seguida é feita uma breve descrição da arquitetura desenvolvida na parte do Software e das diferentes camadas criadas e utilizadas. Por fim, são feitas algumas considerações relativamente ao modo de comunicação entre a camada de software e a de hardware utilizando o ROS.

3.1 Plataforma Experimental

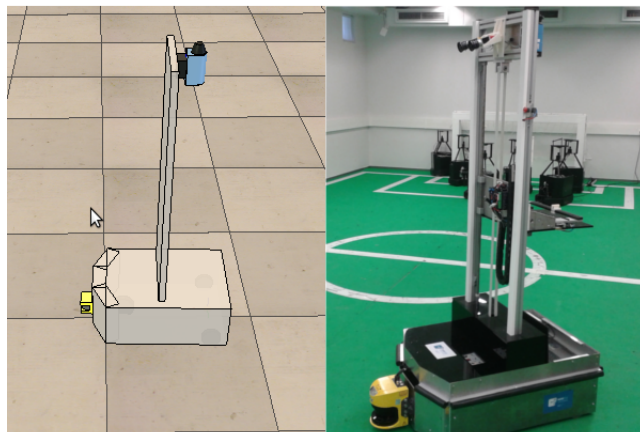


Figura 3.1: À esquerda pode-se ver a simulação do veículo e à direita o veículo real.

A plataforma utilizada no decorrer da dissertação consiste num veículo com roda dianteira de tração e configuração em triciclo, como mostrado na figura 3.1. Na presente secção vão ser explicadas todas os principais componentes constituintes deste veículo de nome Jarvis.

3.1.1 Plataforma do veículo

O principal objetivo deste veículo consiste no transporte de material percorrendo um layout prédefinido do ponto A ao ponto B com a maior precisão possível. O veículo tem um configuração em triciclo como representado na figura 3.1 onde a roda da frente é a roda motriz.



Figura 3.2: (a)À esquerda encontra-se o encoder Drehgeber Serie 85; (b)No centro encontra-se a estrutura que suporta a roda e o seu movimento Schabmueller S2; (c)À direita encontra-se o encoder LIKA I58H

A roda motriz é uma Schabmueller S2(Fig. 3.2 (b)), que controla tanto a direção como a translação do veículo. A execução da translação e rotação é feita por dois motores DC que têm acoplados dois encoders. O encoder da translação é um Krementale Drehgeber Serie85 (Fig 3.2 (a)), o encoder da rotação é um LIKA I58H(Fig 3.2 (c)), estes encoders realimentam respetivamente as drives de direção e translação. A drive de tração, DZRALTE-020L080, e rotação, DZRALTE-012L080, têm como entradas a informação relativa à direção, sentido e velocidade da roda, sendo estas provenientes do sistema de controlo, e a informação proveniente dos encoders da roda.

De forma a existir uma melhor localização do veículo, visto o erro da odometria ser ilimitado, torna-se necessário adquirir mais informação do exterior de forma a poder corrigir a posição prevista pela odometria. Para tal utiliza-se o laser de segurança (obrigatório), modelo SICK S3000 (Fig. 3.5 (a)) que deteta com precisão elevada a distância às balizas que se encontram no seu campo de alcance (mais sobre este laser será explicado na próxima secção).

O veículo é alimentado por 2 baterias de chumbo de 12V cada. O programa de localização corre em ROS (Robot Operating Sistem) e é executado num pc Toshiba Portégé com um processador i5, 2 CPU físicos, 4 threads no total, cada a 1.6Ghz. Também existe um ambiente visual (Figura 3.3) que permite ao utilizador alterar o percurso que o veículo deve percorrer, pontos de paragem, sentido do movimento e posição inicial do veículo (Ambiente desenvolvido por Heber Sobreira).

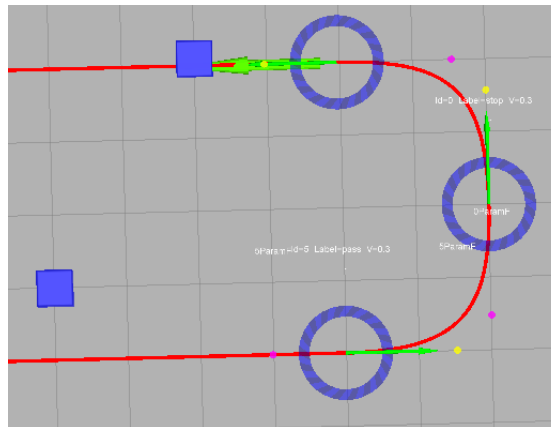


Figura 3.3: A vermelho esta representado o caminho a percorrer, os blocos azuis representarão as balizas mapeadas, e os círculos azuis são elementos dinâmicos que permitem a alteração do caminho

3.1.2 Laser Range Finder

O SICK S3000 Professional CMS (Fig. 3.5 (a)) é um dos lasers mais recentes do mercado, que para além de ter a função de fazer de sensor de segurança de forma a evitar colisões com possíveis obstáculos inesperados, tem ainda a capacidade de detetar elementos refletivos e contornos.

No que concerne à funcionalidade de segurança, é possível definir até 4 áreas distintas de segurança e associar-lhe fatores de velocidade a cada uma. Isto é, se algum objeto entrar na área de segurança ou é diminuída a velocidade ou a energia é cortada diretamente aos motores despoletando o sistema de travagem, como exemplificado na figura (b) e (c)

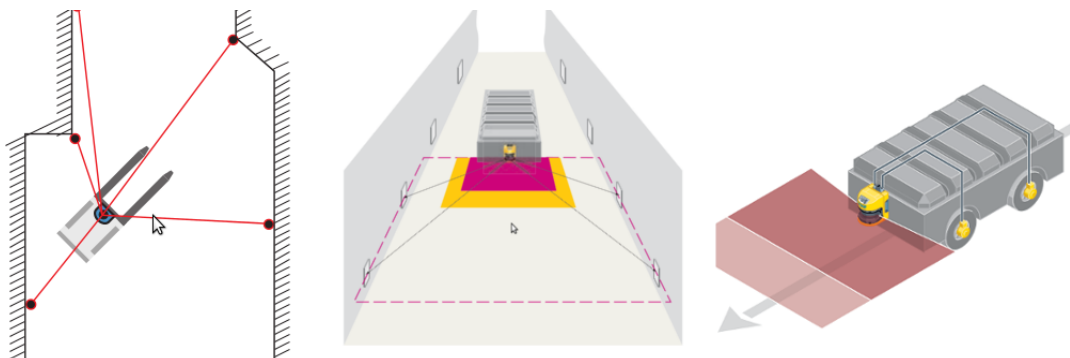


Figura 3.4: (a) À esquerda representa-se a deteção de balizas; (b) No centro representa-se as áreas de segurança a deteção de contornos e de balizas; (c) À direita encontram-se representadas as áreas de segurança e os atuadores aplicados nas rodas.

Relativamente a deteção de elementos refletivos e contornos, este laser tem um alcance máximo de 49 metros com um erro absoluto médio de 5mm por metro. A informação retornada é de dois tipos: posição da baliza e contornos (Fig. 3.4 (a) e (b) respetivamente). Ambos os tipos recebem como parâmetros a posição, x e y .

É possível detectar uma baliza a uma distância mínima de 4mm e máxima de 49m. As balizas utilizadas são tubos de pvc, com altura de 50 cm e diâmetro de 5 cm, revestidos de fita refletora da marca Diamond Grand 3000X(Fig. 3.5 (b)).



Figura 3.5: (a) À esquerda encontra-se o laser SICK 3000S; (b) À direita encontra-se o exemplo de uma baliza.

Consequente do uso de balizas cilíndricas e do elevado número de elementos refletores, encontrados em ambiente industrial, que podem ser adquiridos como falsas balizas, existe a necessidade de fazer um pré-processamento dos dados de forma a excluir falsos positivos. São feitos dois tipos de processamentos diferentes:

- Visto a baliza ser circular é necessário calcular o centro do cilindro. Assim, dos pontos visualizados para uma baliza é escolhido o ponto do meio e adicionado vetorialmente o raio do cilindro;
- Para todas as balizas até 10 metros de distância são identificados um mínimo de 4 pontos (figura 3.6), assim, de forma a evitar falsos positivos é definido um número mínimo de pontos para o qual se considera a existência da baliza;

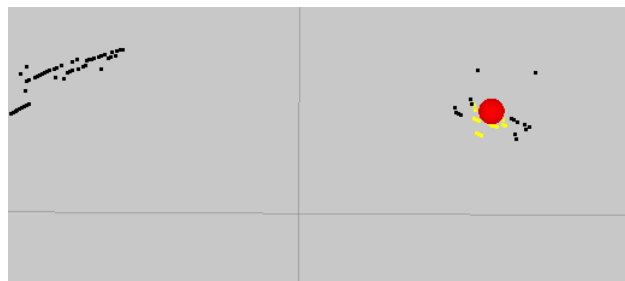


Figura 3.6: A vermelho está representada a baliza processada pela drive do laser. A preto e amarelo esta a informação bruta proveniente do laser que correspondem a elementos não refletores e refletores, respetivamente

3.2 Arquitetura do Software

3.2.1 Camada ROS

O ROS (Robot Operating System) disponibiliza bibliotecas e ferramentas que ajudam os programadores a criarem novas aplicações robóticas. Na verdade o ROS não é exatamente um sistema operativo no verdadeiro sentido da palavra, é um *framework* que é executado sobre um sistema operativo. Entre muitas outras vantagens fornece drivers, software de visualização, *message-passing*, gestão de pacotes e uma das suas maiores vantagens recai na abstração do hardware, proporcionando assim grande flexibilidade ao código criado.

Os principais conceitos do ROS passam pelos seus serviços, nodos, mensagens e tópicos [6].

Os nodos executam os programas e é onde todo o processamento computacional é executado. Um sistema é usualmente composto por múltiplos nodos, sendo cada um deles responsável por uma tarefa diferente. A comunicação entre nodos é feita pela passagem de mensagens. Uma mensagem é um tipo de dado, por exemplo int, double, etc...Os nodos enviam as mensagens publicando-as num tópico que é posteriormente subscrito. Por outras palavras os nodos seguem um serviço de publicador/subscritor.

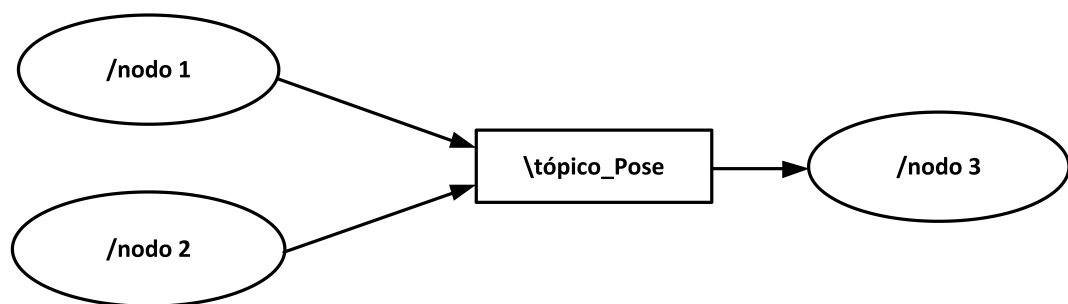


Figura 3.7: Exemplo arquitetura ROS tipo publicador subscritor

Um tópico é simplesmente o nome dado para identificar o conteúdo da mensagem. Todos os nodos interessados em determinada mensagem subscrevem o tópico referente à mesma. Podem existir diversos publicadores e subscritores para o mesmo tópico, todos ignorando a existência uns dos outros. Este paradigma é utilizado para separar os conceitos de produção e consumo.

Um serviço é outra forma de os nodos comunicarem entre si. Os serviços permitem aos nodos enviarem um pedido e receberem uma resposta. Ou seja, seguem um serviço de pedido/resposta.

Como se pode ver na figura 3.7 o nodo 1 e o nodo 2 são dois nodos que produzem mensagem do mesmo tipo. O tipo de mensagem transmitido pelo nodo 1 e 2 é denominado como "/topicoPose" e este tópico é subscrito pelo nodo 3.

Para lá dos conceitos referidos anteriormente o ROS é dotado de vários programas, que ajudam na interpretação tanto dos resultados adquiridos pelos sensores como dos possíveis erros de

conexão entre outros. São exemplo de tais programas o *rxgraph*, o *rxconsole* e o *rviz*. O *rxgraph* permite-nos ver os nodos existentes, na execução do programa, e a interação entre os mesmos (é exemplo da figura 3.7).

O *rxconsole* é um visualizador, da biblioteca *rxtools*, e apresenta todas as mensagens publicadas no "rosout". Recolhe mensagens ao longo do tempo, e permite analisar com mais detalhe filtrando as mensagens de acordo com a prioridade e tipo de mensagem.

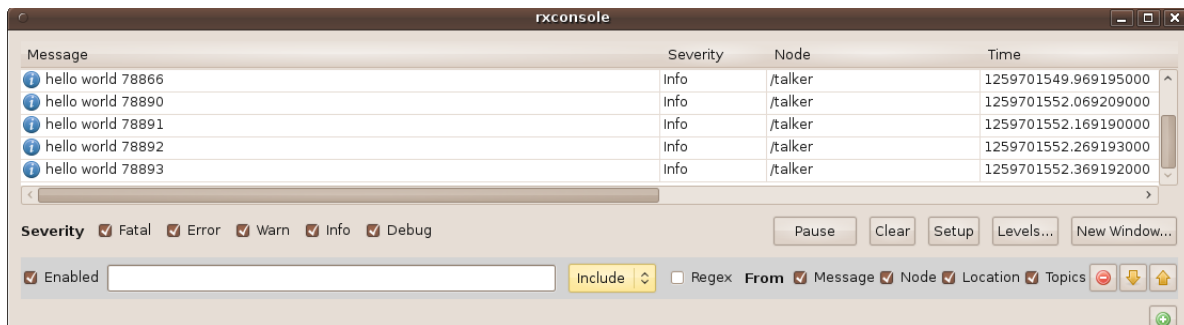


Figura 3.8: Esta figura exemplifica o tipo de mensagens que podem aparecer.

O *rviz* é um visualizador 3D de ambiente e permite saber o que o robot está a fazer sentir e ver. É ideal para utilizar durante o debug do código de forma a interpretar os comandos provenientes do algoritmo.

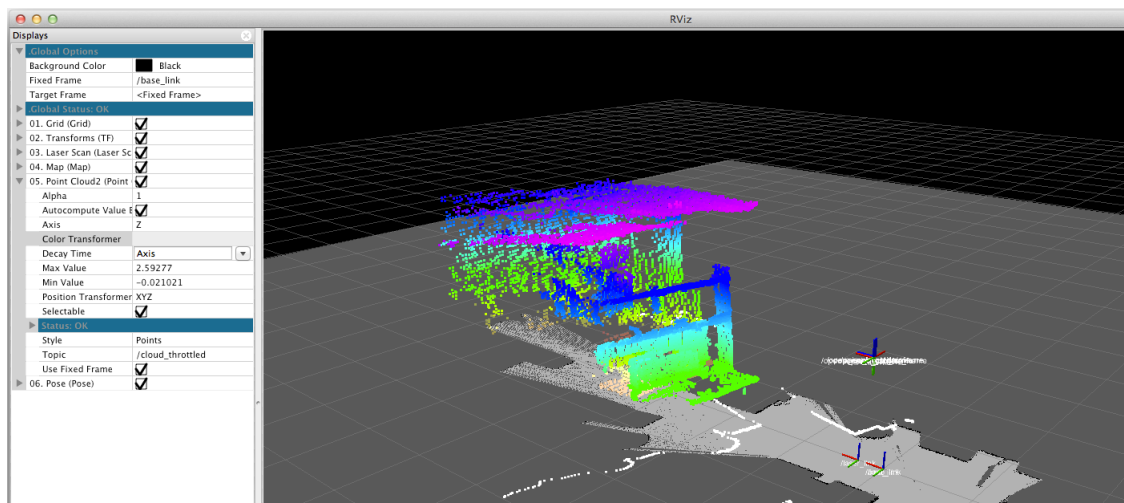


Figura 3.9: Representação da informação proveniente de um laser 3d no rviz

3.2.2 Camadas desenvolvidas

Nesta subsecção é descrito o desenvolvimento do esqueleto do programa utilizando todas as ferramentas a que ROS dá acesso.

Atendendo à capacidade de abstração do hardware que o ROS permite, é possível tornar o código muito mais flexível e reutilizável. Ou seja, em vez da configuração do hardware ser feita no início do algoritmo e o pré-processamento da informação proveniente do mesmo ser feito no decorrer do algoritmo, é tudo processado no bloco de hardware. No final, a informação proveniente do loco do hardware é enviada por mensagens de diferentes tipos, mediante o tipo de informação para o bloco de *interface*. Tal característica permite separar completamente os códigos dos diferentes blocos sendo assim possível usar um diferente hardware sem ser necessário mudar o algoritmo e também usar um diferente algoritmo sem ser necessário reconfigurar a maneira como a informação é passada pelo bloco de hardware.

Assim como representado na figura 3.10 o programa desenvolvido é constituído por 4 blocos diferentes: 1) Algoritmo; 2) Ambiente Gráfico; 3) Interface; 4) Hardware;

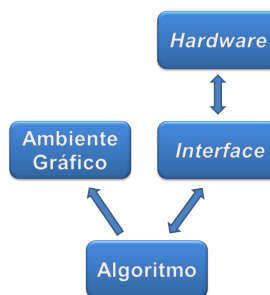


Figura 3.10: Diagrama de camadas do Software

No bloco Algoritmo é onde é feito todo o processamento do algoritmo, e onde são guardadas todas as variáveis. O ciclo principal é dividido em 3 fases:

- `getInputData();`
- `processMainCycle();`
- `setOutputData();`

Na fase de *getInputData* são atualizadas todas as variáveis com os novos dados provenientes do bloco de Hardware. No *processMainCycle* são processados todos os dados mediante o objetivo do algoritmo. E por fim novos dados já atualizados pelo algoritmo são passados na fase de *setOutputData* para fora do bloco.

É no bloco de hardware que é feito o pré-processamento da informação proveniente do hardware envolvente. Neste bloco são inicializadas as configurações de cada hardware e são feitos todos os cálculos necessários de forma a ter os dados de saída prontos a serem usados pelo algoritmo.

O Bloco de Interface é onde são feitas todas as ligações entre o bloco de Hardware e de Software. Ou seja, é onde se organiza os dados provenientes dos sensores (bloco de hardware) de forma a ficarem acessíveis ao algoritmo.

Por fim, o bloco de Ambiente Gráfico foi desenvolvido com o intuito de fazer debug na fase de testes. Este bloco vai buscar os dados provenientes do bloco do Algoritmo e interpreta-os de forma visual com o objetivo de melhor compreender os resultados do algoritmo como é possível visualizar na figura 3.11.

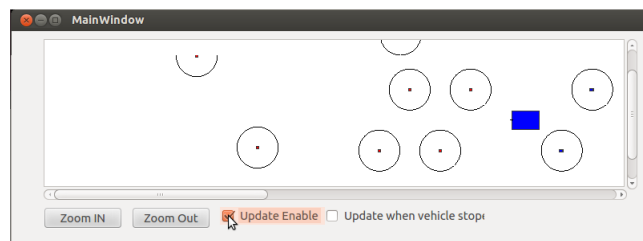


Figura 3.11: Ambiente gráfico de Debug

Esta consola de debug é constituída por dois botões de Zoom-In e Zoom-Out, uma *checkbox* de nome *Update Enable* que permite ativar o modo de atualização do Algoritmo de EKF e EKF-SLAM. Para experiências foi também desenvolvido uma *checkbox* de nome *Update when vehicle stopped* com o objetivo de atualização da pose com o veículo parado de forma a perceber tais repercussões. Na figura 3.12 é possível ver um exemplo da funcionalidade deste módulo, onde se analisa a evolução do filtro ao longo do tempo.

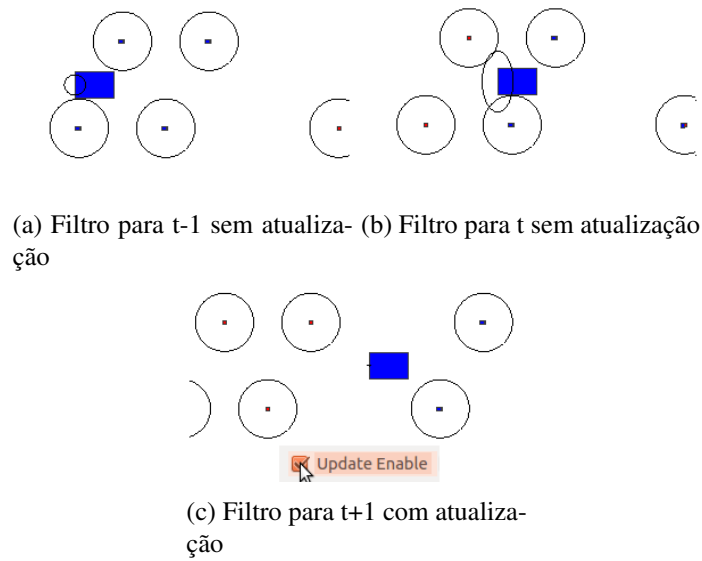


Figura 3.12: Evolução do EKF ao longo do tempo

O retângulo a azul representa a posição do veículo no referencial do mundo. Os pequenos quadrados a azul e vermelho, representam respectivamente as posições globais das balizas mapeadas e das balizas observadas. O círculo em volta das balizas mapeadas (a vermelho) representa o *threshold* para o qual uma baliza observada é associada a uma baliza mapeada.

Por fim, foi também desenvolvido um pequeno GUI (Grafical User Interface) (fig. 3.13) de forma a ser possível mudar os parâmetros do filtro para uma melhor afinação.

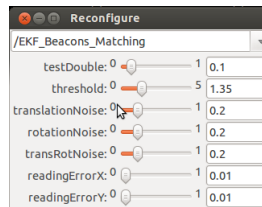


Figura 3.13: Interface gráfico

No interface da figura 3.13 estão representados os seguintes parâmetros:

- *threshold*, é a máxima diferença para a qual a posição de uma baliza pode ser associada;
- *translationNoise* é o erro associado à odometria durante a translação;
- *rotationNoise* é o erro associado à odometria durante a rotação;
- *readingErrorX* é o erro associado à leitura do laser em x ;
- *readingErrorY* é o erro associado à leitura do laser em y ;

Capítulo 4

Algoritmo de localização baseado no *Extended Kalman Filter*

Neste capítulo vai ser apresentado detalhadamente o algoritmo de localização baseado no *Extended Kalman Filter* (EKF). Inicialmente serão descritas todas as variáveis envolvidas no algoritmo, seguindo-se uma breve descrição de todo o mesmo e a respetiva interação entre os blocos que o constituem. O pseudo-código, seguido de uma explicação detalhada, vai ser utilizado de forma a melhor exemplificar o funcionamento do algoritmo e as suas diferentes fases.

Por fim são feitas pequenas considerações sobre os métodos a utilizar, os valores obtidos para os coeficientes de erro e os consequentes resultados.

4.1 Introdução ao Algoritmo de localização baseado em EKF

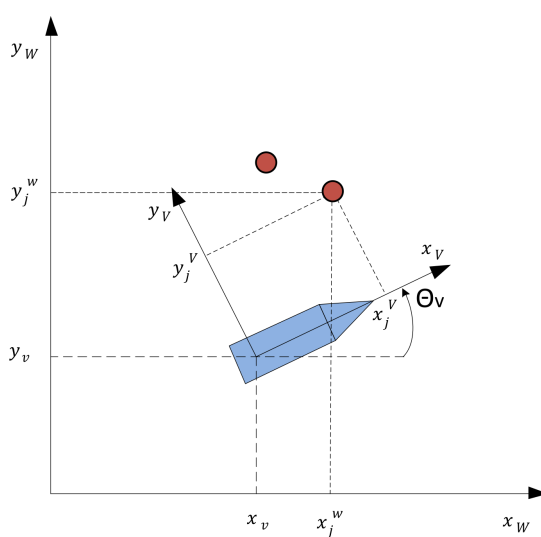


Figura 4.1: Referencial do mundo e referencial do veículo

A nomenclatura utilizada pelo EKF vai ser explicada e ilustrada na presente sub-secção.

A Fig. 5.1 mostra o referencial do mundo, com eixos (x_W, y_W) e o referencial do veículo (x_V, y_V) . A pose do veículo é representada pelo vetor (x_v, y_v, θ_v) . O referencial do veículo encontra-se no meio das duas rodas do eixo traseiro. Assim sendo, o referencial do veículo tem coordenadas relativamente ao referencial do mundo, (x_v, y_v) , e uma rotação também relativa ao referencial do mundo, θ_v .

A baliza j é representada por um ponto no referencial do mundo. É definida por X_{Pntj}^W como representado na equação 4.1, com os parâmetros x_{Pntj}^W e y_{Pntj}^W , no referencial do mundo como é representado na figura 5.1

$$X_{Pntj}^W = \begin{bmatrix} x_{Pntj}^W \\ y_{Pntj}^W \end{bmatrix} \quad (4.1)$$

A transformada das coordenadas da baliza entre o referencial do mundo e o referencial do veículo, é expressa pela seguinte equação:

$$X_{Pntj}^V = \begin{bmatrix} (x_{Pntj}^W - x_v) \cos(\theta_v) + (y_{Pntj}^W - y_v) \sin(\theta_v) \\ -(x_{Pntj}^W - x_v) \sin(\theta_v) + (y_{Pntj}^W - y_v) \cos(\theta_v) \end{bmatrix} \quad (4.2)$$

Pelo contrário a representação das coordenadas da baliza no referencial do mundo a partir das suas coordenadas no referencial do veículo é expressa por:

$$X_{Pntj}^W = \begin{bmatrix} x_{Pntj}^V \cos(\theta_v) - y_{Pntj}^V \sin(\theta_v) + x_v \\ x_{Pntj}^V \sin(\theta_v) + y_{Pntj}^V \cos(\theta_v) + y_v \end{bmatrix} \quad (4.3)$$

O ciclo de EKF é constituído por 3 passos cruciais (como é possível visualizar na figura 4.2):

- Previsão;
- Associação;
- Atualização;

No primeiro bloco de **Previsão** é usado o algoritmo de movimento previamente enunciado, no capítulo 2, e a odometria do veículo para estimar o estado actual do mesmo $\hat{X}(k+1|k)$ e a respetiva covariância $P(k+1|k)$.

Antes de poder executar o passo da associação têm de ser percorridos dos blocos auxiliares como é o exemplo do bloco de **Observação** e o de **Previsão da Medição**.

Na **Observação** dá-se a recolha das medições adquiridas através do sensor $Z(k+1)$ sendo estas a posição das balizas no seu raio de visão, relativamente ao referencial do veículo. De seguida, no passo de **Previsão da Medição** é utilizada a previsão da pose do veículo, $X(k+1|k)$, e a informação proveniente do mapa, M_{Pnt} , de forma a obter-se a distância prevista entre o veículo e as balizas presentes no alcance do seu laser, $\hat{h}_i(k+1)$.

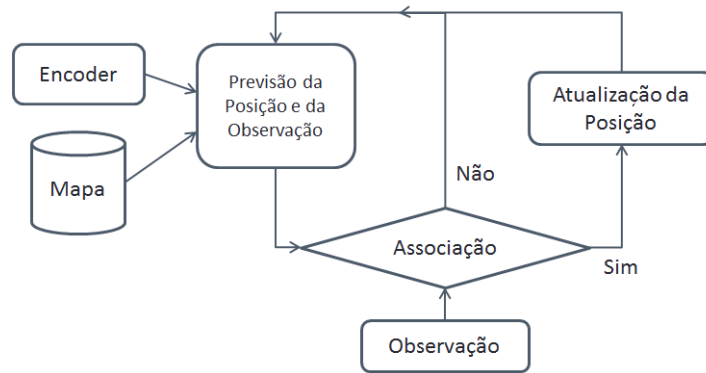


Figura 4.2: Diagrama do Algoritmo EKF

Assim no passo de **Associação** antes de mais, é necessário passar as coordenadas das balizas, que se encontram guardadas no referencial do mundo, para as coordenadas no referencial do veículo, sendo assim possível fazer uma comparação entre as medições. Com a informação retirada do passo anterior é então possível saber a diferença de distâncias entre a posição estimada das balizas, e a observada, chamando-se esta diferença de Inovação ($V = Z_j(k+1) - \hat{h}_i(k+1)$).

Por fim, resta calcular a melhor estimativa da posição do veículo $\hat{X}(k+1|k+1)$ e a sua covariância $P(k+1|k+1)$, tendo em consideração a **Previsão da medição** e a **Observação** feita pelo laser, tendo tal passo o nome de **Atualização**.

No algoritmo 1 é possível analisar o pseudocódigo correspondente ao algoritmo de EKF explicado anteriormente.

4.1.1 Previsão

No passo de previsão, o algoritmo EKF usa o modelo de movimento para prever a posição atual do veículo, através da posição anterior. Tem-se então que, no momento temporal k , o veículo encontra-se exatamente na posição $X(k) = [x \ y \ \theta]^T$, relativamente à origem, e a sua melhor estimativa é $\hat{X}(k|k) = [\hat{x} \ \hat{y} \ \hat{\theta}]^T$. Revendo novamente o algoritmo de movimento apresentado no capítulo 3 tem-se que:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \delta_{trans} \cos(\theta + \delta_{rot1}) \\ \delta_{trans} \sin(\theta + \delta_{rot1}) \\ \delta_{rot1} + \delta_{rot2} \end{bmatrix} \quad (4.4)$$

onde $\hat{X}(k|k) = (x \ y \ \theta)^T$ e $\hat{X}(k+1|k) = (x' \ y' \ \theta')$ são as poses estimadas em k e $k+1$, respetivamente. O verdadeiro movimento é descrito por uma translação e duas rotações que são sempre

Algorithm 1 Algoritmo de localização EKF**Rquire:** $\hat{X}(k|k), P(k|k), \text{Odometria}$ **EKF Previsão**

$$\hat{X}(k+1|k) \leftarrow X(k|k) + f_v$$

$$\nabla f_p \leftarrow \begin{bmatrix} 1 & 0 & -\delta_{trans} \sin(\theta + \delta_{rot1}) \\ 0 & 1 & \delta_{trans} \cos(\theta + \delta_{rot1}) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\nabla f_u \leftarrow \begin{bmatrix} \cos(\theta + \alpha_{rot1} + \alpha_{rot1}) & -\frac{1}{2} \sin(\theta + \alpha_{rot1} + \alpha_{rot1}) \\ \sin(\theta + \alpha_{rot1} + \alpha_{rot1}) & \frac{1}{2} \cos(\theta + \alpha_{rot1} + \alpha_{rot1}) \\ 0 & 1 \end{bmatrix}$$

$$P(k+1|k) \leftarrow \nabla f_p \cdot P(k|k) \cdot \nabla f_p^T + \nabla f_u \cdot Q_v(k) \cdot \nabla f_u^T$$

Associação

$$\hat{h} \leftarrow \begin{bmatrix} l_x \cdot \cos(\hat{\theta}_v) + l_y \sin(\hat{\theta}_v) \\ l_y \cdot \cos(\hat{\theta}_v) - l_x \sin(\hat{\theta}_v) \end{bmatrix}$$

$$\nabla h_{Pntj}^X \leftarrow \begin{bmatrix} -\cos(\hat{\theta}_v) & -\sin(\hat{\theta}_v) & -l_x \sin(\hat{\theta}_v) + l_y \cos(\hat{\theta}_v) \\ \sin(\hat{\theta}_v) & -\cos(\hat{\theta}_v) & -l_x \cos(\hat{\theta}_v) - l_y \sin(\hat{\theta}_v) \end{bmatrix} (k+1|k)$$

$$V_j \leftarrow Z_{Pntj} - \hat{h}_{Pntj}$$

Atualização

$$\hat{X}(k+1|k+1) \leftarrow \hat{X}(k+1|k) + W(k+1)V(k+1)$$

$$\hat{X}(k+1|k+1) \leftarrow \hat{X}(k+1|k) + W(k+1)V(k+1)$$

$$W(k+1) \leftarrow P(k+1|k) \nabla h_{Pntj}^{X^T} [\nabla h_{Pntj}^X P(k+1|k) \nabla h_{Pntj}^{X^T} + R]^{-1}$$

acompanhadas por um erro gaussiano:

$$\underbrace{\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix}}_{\hat{X}(k+1|k)} = \underbrace{\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}}_{f_p(u_k, \hat{X}(k|k))} + \underbrace{\begin{bmatrix} \delta_{trans} \cos(\theta + \delta_{rot1}) \\ \delta_{trans} \sin(\theta + \delta_{rot1}) \\ \delta_{rot1} + \delta_{rot2} \end{bmatrix}}_{q_v} + q_v \quad (4.5)$$

onde $f_p(u_k, \hat{X}(k|k))$ é o modelo que representa a resposta do sistema mediante o comando u_t , ou a informação proveniente da odometria, e a posição anterior $\hat{X}(k|k)$. O modelo cinemático do erro q_v foi modelado considerando o ruído branco com uma distribuição gaussiana de valor esperado igual a zero ($\hat{q}_v = 0$) e uma covariância Q_v . O ruído branco q_v é constituído pelo erro associado ao estado do veículo:

$$q_v = [\varepsilon_{xv} \ \varepsilon_{yv} \ \varepsilon_{\theta v}]^T \quad (4.6)$$

Sabendo os modelos do sistema e o modelo do erro, é então possível calcular a covariância associada à previsão, $P(k+1|k)$ [3]:

$$P(k+1|k) = \nabla f_p \cdot P(k|k) \cdot \nabla f_p^T + \nabla f_v \cdot Q_v(k) \cdot \nabla f_v^T \quad (4.7)$$

tal permite-nos prever a posição do veículo e a incerteza associada à mesma depois de um movimento especificado por u_k .

A jacobiana ∇f_p é a derivada da função f_p com respeito a $X(k)$. Assim sendo através da função que representa o modelo de movimento podemos retirar a seguinte matriz:

$$\nabla f_p = \begin{bmatrix} 1 & 0 & -\delta_{trans} \sin(\theta + \delta_{rot1}) \\ 0 & 1 & \delta_{trans} \cos(\theta + \delta_{rot1}) \\ 0 & 0 & 1 \end{bmatrix} \quad (4.8)$$

A matriz Q_v define o ruído existente no controlo. Q_v aumenta com a translação e rotação estimada do veículo e é definido por:

$$Q_v = \begin{bmatrix} [\delta_{trans} \cdot \cos(\theta + \delta_{rot1}) \cdot \sigma_{\delta_{trans}}]^2 & 0 & 0 \\ 0 & [\delta_{trans} \cdot \sin(\theta + \delta_{rot1}) \cdot \sigma_{\delta_{trans}}]^2 & 0 \\ 0 & 0 & (A)^2 + (B)^2 \end{bmatrix} \quad (4.9)$$

onde $A = \delta_{trans} \cdot \sigma_{d\theta}$, $B = (\delta_{rot1} + \delta_{rot2}) \cdot \sigma_{\theta}$ e $\sigma_{\delta_{trans}}$, $\sigma_{\delta_{rot1}}$, $\sigma_{\delta_{rot2}}$, $\sigma_{\delta_{d\theta}}$ são erros de desvio os quais serão explicados no final do capítulo. Ou seja quando o veículo anda um metro em frente o erro de translação e rotação estimado será $\sigma_{\delta_{trans}}$ e $\sigma_{\delta_{d\theta}}$ respetivamente. Quando o veículo faz um movimento de rotação de um radiano o erro estimado da rotação proveniente da odometria é igual a σ_{θ} .

O modelo de movimento referido na equação 4.5 necessita que o ruído do movimento seja mapeado no estado. Assim as transformada do espaço do controlo para o espaço do estado é definida por $\nabla f_v = \frac{\delta f(u_k, X_{k-1})}{\delta u_k}$ é o jacobiana da função de movimento em ordem ao aos parâmetros do movimento

$$\nabla f_u = \begin{bmatrix} \cos(\theta + \alpha_{rot1}) & -\frac{1}{2} \sin(\theta + \alpha_{rot1}) \\ \sin(\theta + \alpha_{rot1}) & \frac{1}{2} \cos(\theta + \alpha_{rot1}) \\ 0 & 1 \end{bmatrix} \quad (4.10)$$

A covariância prevista depende da dinâmica do sistema e da covariância anterior, $(\nabla f_p \cdot P(k|k) \cdot \nabla f_p^T)$. O membro $(\nabla f_u \cdot Q_v(k) \cdot \nabla f_u^T)$ da equação 4.1.1 aumenta a covariância e a incerteza da posição até ser feita uma nova observação.

4.1.2 Observação

O segundo passo consiste em obter as medidas provenientes do laser, $Z_{Obs}(k+1)$, quando o veículo se encontra na posição temporal $k+1$. Neste caso as medidas são pré-processadas e recebidas com os parâmetros x_{Pntj}^V e y_{Pntj}^V relativamente ao referencial do veículo, para cada baliza j como representado na seguinte equação:

$$Z_{Pntj} = X_{ObsPntj}^V = \begin{bmatrix} x_{ObsPntj}^V \\ y_{ObsPntj}^V \end{bmatrix} \quad (4.11)$$

As balizas são adquiridas no referencial do laser, ou seja no referencial do veículo, no entanto para fazer o matching entre as balizas observadas e as balizas previstas é necessário representá-los no mesmo referencial. Na presente abordagem vão ser transformadas as medições das balizas previstas, do referencial do mundo para o referencial do veículo como será explicado no próximo passo.

4.1.3 Previsão da Medição

A previsão da medição consiste em determinar, mediante a posição do veículo, a que distância as balizas se deveriam encontrar do mesmo. Para tal é utilizada a pose proveniente do passo da previsão $X(k+1|k)$ e a informação proveniente do mapa $M(k)$ para determinar a posição estimada da baliza, Z_{est} . Visto as coordenadas de $X(k+1|k)$ e $M(k)$ se encontrem no referencial do mundo é necessário existir uma mudança de coordenada. Assim, para a previsão de cada baliza, z_i , temos uma mudança de coordenadas h_i :

$$\begin{aligned}\hat{z}_i(k+1) &= h_i(z_i, \hat{X}(k+1|k)) \Leftrightarrow \\ \hat{z}_i(k+1) &= \hat{X}_{Pntj}^W(k+1|k)\end{aligned}\quad (4.12)$$

A atualização da posição correspondente à baliza j observada é representada por:

$$\hat{h}_{Pntj} = \begin{bmatrix} x_{Pntj}^V \\ y_{Pntj}^V \end{bmatrix} (k+1|k) \quad (4.13)$$

Usando a equação 4.2, a atualização da observação para cada ponto j, pode ser calculada de seguinte maneira:

$$l_x = x_m - \bar{x}_v, l_y = y_m - \bar{y}_v \quad (4.14)$$

$$Z_{est} = \hat{h} = \begin{bmatrix} l_x \cdot \cos(\hat{\theta}_v) + l_y \sin(\hat{\theta}_v) \\ l_y \cdot \cos(\hat{\theta}_v) - l_x \sin(\hat{\theta}_v) \end{bmatrix} \quad (4.15)$$

A posição da baliza proveniente da observação é acompanhada por um erro e pode ser expressa pela seguinte equação:

$$\begin{aligned}Z_{Pntj} &= h_{Pntj} + r_{Pntj} \Leftrightarrow \\ Z_{Pntj} &= \begin{bmatrix} x_{Pntj}^V + \varepsilon_{xj} \\ y_{Pntj}^V + \varepsilon_{yj} \end{bmatrix}\end{aligned}\quad (4.16)$$

onde o erro r_{Pntj} e o erro associado com a observação da baliza j, modelada com ruído gaussiano de média zero e covariância R_{Pntj} . A matriz de covariância associada ao modelo de erros das

observações, é dada pela seguinte expressão:

$$R_{Pntj} = \begin{bmatrix} \sigma_x & \sigma_{xy} \\ \sigma_{yx} & \sigma_y \end{bmatrix} \quad (4.17)$$

onde σ_x e σ_y são valores retirados do datasheet do laser, e σ_{xy} é zero visto não existir nenhuma relação entre os parâmetros x e y .

O gradiente do modelo de observação h_{Pntj} em ordem ao ruído da observação das balizas observadas r_{Pntj} é uma matriz identidade enquanto que em ordem ao vetor de estado no ponto j , é igual a ∇h_{Pntj}^X , e é obtido através de:

$$\nabla h_{Pntj}^X = \begin{bmatrix} -\cos(\hat{\theta}_v) & -\sin(\hat{\theta}_v) & -lx \sin(\hat{\theta}_v) + ly \cos(\hat{\theta}_v) \\ \sin(\hat{\theta}_v) & -\cos(\hat{\theta}_v) & -lx \cos(\hat{\theta}_v) - ly \sin(\hat{\theta}_v) \end{bmatrix} (k+1|k) \quad (4.18)$$

4.1.4 Associação

Neste ponto do algoritmo possuímos um conjunto de observações de balizas provenientes do laser, e também um conjunto de balizas previstas, ambas posicionadas no referencial do veículo. O processo de Associação consiste em identificar todas as observações que correspondem às balizas previstas. Por outras palavras, tenta-se encontrar uma ligação intuitiva que diga "esta observação, feita pelo laser, corresponde a determinada baliza prevista pela informação proveniente do mapa".

Para que uma associação seja feita entre uma baliza do vetor de observações, $Z_{ObsPnti}^W$, e uma baliza do vetor previsões, $\hat{Z}_{Pntj}^W$ é necessário que as seguintes condições se verifiquem:

1. A distância euclidiana entre $Z_{ObsPnti}^W$ e $\hat{Z}_{Pntj}^W$ seja menor que um parâmetro ajustável δ_{dBeac} .
2. Dos pares que passem à condição anterior só se utilizará única e exclusivamente a menor distancia ente os pares.

Feita a correspondência entre a baliza observada e a prevista é-nos então possível calcular o quanto a previsão de uma certa baliza j está afastada da real posição da mesma. Este calculo é denominado de Inovação e é calculado da seguinte maneira:

$$V_j = Z_{Pntj} - \hat{h}_{Pntj} \quad (4.19)$$

4.1.5 Atualização

A atualização, $X(k+1|k+1)$, não é nada mais do que uma correção da previsão tendo em conta as observações e respetiva associação com a informação proveniente do mapa. Ou seja a nova pose corrigida utilizando o algoritmo de EKF na fase do update é calculada da seguinte maneira [3]:

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + W(k+1)V(k+1) \quad (4.20)$$

onde $W(k+1)$ representa o ganho de Kalman. O ganho de Kalman corresponde à correção que irá ser feita de forma ao veículo se aproximar da pose real. O ganho de Kalman é igual a:

$$W(k+1) = P(k+1|k) \nabla h_{Pntj}^X{}^T [\nabla h_{Pntj}^X P(k+1|k) \nabla h_{Pntj}^X{}^T + R]^{-1} \quad (4.21)$$

finalmente a nova covariância correspondente a $\hat{X}(k+1|k+1)$ é calculada da seguinte maneira:

$$P(k+1|k+1) = P(k+1|k) - W(k+1) \nabla h_{Pntj}^X P(k+1|k) \quad (4.22)$$

4.2 Parâmetros do Filtro

Na presente secção será apresentada a justificação para os valores escolhidos dos parâmetros descritos ao longo da secção anterior.

Os valores utilizados relativamente aos parâmetros do laser, σ_x e σ_y , foram retirados do manual de instruções do instrução, e representam a estimativa da covariância associada às medições do laser. Estes valores podem ser observados na tabela 4.1.

Tabela 4.1: Valores de erro do Laser

δ_x (m/m)	0,001
δ_y (m/m)	0,001

Foram também conduzidas repetidas experiências de forma a retirar os possíveis erros da odometria. Todas as experiências para os valores apresentados foram realizadas com o veículo a uma velocidade de 0.3 m/s. Assim com um ciclo de 100 milissegundos, o veículo mover-se-a, entre ciclos, a uma distância de 0,03 metros. Assumindo então que para o pior caso o ciclo demora 3 vezes a velocidade norma de todo o processamento, o veículo move-se 0,09 metros entre ciclos. Assim no corredor I - 175 (figura 6.1) foram feitos diversos ensaios com o robô a deslocar-se em linha reta e foram retirados os erros de translação, $\sigma_{\delta_{trans}}$, e rotação $\sigma_{d\theta}$ por metro. Também foram feitos ensaios onde o veículo fez rotações de 360° e de seguida medido o erro de rotação por radiano σ_θ . Assim foram obtidos os seguintes valores da tabela 4.2

Tabela 4.2: Valores de erro do Laser

$\sigma_{\delta_{trans}}$ (m/m)	0,015
$\sigma_{d\theta}$ (rad/m)	0,07
σ_θ (rad/rad)	0,03

Por fim falta falar do parâmetro δ_{dBeac} , que define o valor máximo para o qual uma associação é feita. Uma vez que as balizas terão de estar afastadas um mínimo de 0,5 metros definiu-se o valor deste parâmetro como $\delta_{dBeac} = 0,25$ metros.

Capítulo 5

Localização e Mapeamento Simultâneo

Neste capítulo vai ser apresentado detalhadamente todo o algoritmo de *Simultaneous Localization and Mapping*. Inicialmente é feita uma introdução sobre o SLAM onde é definido o seu objetivo e são descritas todas as variáveis envolvidas no algoritmo. De seguida, é feita uma breve descrição de todo o algoritmo e como os diferentes blocos interagem entre si. O pseudocódigo vai ser utilizado de forma a melhor exemplificar o funcionamento de todo o algoritmo e as suas diferentes fases.

Também será explicado detalhadamente todo o funcionamento de cada bloco que constitui o algoritmo.

Por fim, são feitas pequenas considerações sobre os métodos a utilizar, o valor obtido para os coeficientes de erro e os resultados obtidos.

5.1 Introdução

O presente capítulo aborda um dos mais fundamentais problemas na área da robótica, o problema de localização e mapeamento simultâneo. Este problema é comumente tratado na literatura por *Simultaneous Localization and Mapping* (SLAM). Problemas de SLAM aparecem quando o veículo não tem acesso ao mapeamento da área nem à sua própria posição [5].

Como referido anteriormente (capítulo 2) existem dois tipos de SLAM, o Online SLAM e o Full SLAM. Para lá da complexidade do Full SLAM, referida no capítulo 2, é também importante considerar o elevado nível de processamento exigido por este último. Assim, considerando todos os fatores envolvidos e os requisitos de baixo processamento adjacentes ao presente projeto optou-se pelo desenvolvimento do Algoritmo de Online SLAM, algoritmo este que será pormenorizadamente descrito nos próximos capítulos.

5.2 Nomenclatura do Algoritmo de SLAM-EKF

Na presente secção vão ser introduzidas todas as variáveis de estado envolvidas ao Algoritmo de SLAM-EKF.

A figura 5.1 apresenta o referencial do mundo com eixos (x_W, y_W) e o referencial do veículo (x_V, y_V) . A pose do veículo é representada pelo vetor (x_v, y_v, θ_v) . O referencial do veículo encontra-se centrado no meio das duas rodas traseiras do veículo. Assim sendo o referencial do

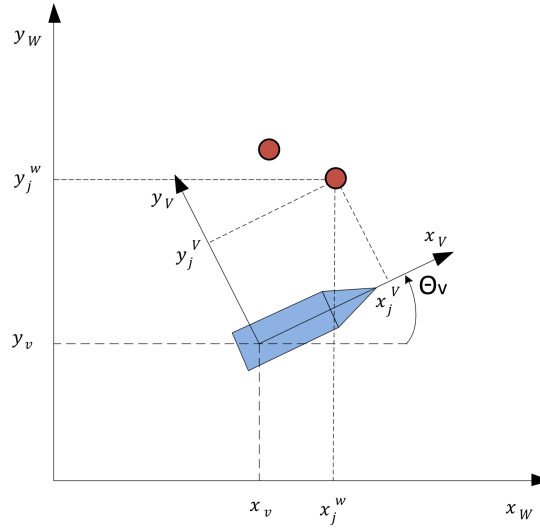


Figura 5.1: Referencial do mundo e referencial do veículo

veículo tem coordenadas relativas ao referencial do mundo iguais às do veículo (x_v, y_v) e uma rotação, também referente ao referencial do mundo, de θ_v . No algoritmo de SLAM ao contrário do algoritmo de EKF, o vetor de estado deixa de ser fixo. Neste caso o vetor de estado é composto pelo estado do veículo $X_v = [x_v, y_v, \theta_v]^T$ e pelo vetor de estado do mapa X_m^W , ambos representados no referencial do mundo. Assim sendo o vetor de estado do presente filtro é representado pela seguinte equação:

$$X = \begin{bmatrix} X_v \\ X_m^W \end{bmatrix} \quad (5.1)$$

O vetor de estado do mapa é representado por X_m^W e é incrementado cada vez que uma nova baliza é considerada válida. Desta forma podemos representar este vetor pela seguinte equação:

$$X_m^W = [X_{Pnt}^W] \quad (5.2)$$

onde X_{Pnt}^W é o vetor de estado correspondente ao conjunto de pontos invariantes que representam as balizas processadas. Assim sendo o vetor de estado do mapa é definido por:

$$X_m^W = [X_{Pnt1}^W X_{Pnt2}^W \dots X_{PntN}^W] \quad (5.3)$$

onde N é o número de pontos invariantes.

Como pode ser visto na figura 5.1 uma baliza i é representada no referencial do veículo pelas coordenadas cartesianas (x_{Pnti}^V, y_{Pnti}^V) , assim o estado observado é definido por:

$$X_{Pnti}^V = \begin{bmatrix} x_{Pnti}^V \\ y_{Pnti}^V \end{bmatrix} \quad (5.4)$$

A baliza i representada como um ponto do referencial do mundo, e convertida através da equação 4.3, é expressa da seguinte forma:

$$X_{Pnti}^W = \begin{bmatrix} x_{Pnti}^W \\ y_{Pnti}^W \end{bmatrix} \quad (5.5)$$

Extrapolando a equação 5.1 podemos obter a seguinte equação:

$$X = \left[\begin{bmatrix} x_v \\ y_v \\ \theta_v \end{bmatrix}^T \quad \begin{bmatrix} x_{Pnt1}^W \\ y_{Pnt1}^W \end{bmatrix}^T \quad \dots \quad \begin{bmatrix} x_{PntN}^W \\ y_{PntN}^W \end{bmatrix}^T \right]^T \quad (5.6)$$

A estimação do estado do filtro é representada por $\hat{X}$ sendo este composto pelos seguintes elementos:

$$\hat{X} = \begin{bmatrix} \hat{X}_v \\ \hat{X}_m^W \end{bmatrix} \quad (5.7)$$

onde $\hat{X}_v$ representa a estimação do estado do veículo, sendo este constituído pelas variáveis $(\hat{x}_v, \hat{y}_v, \hat{\theta}_v)$, e $\hat{X}_m^W$ representa a estimativa do vetor de estado do mapa, o qual é representado pela seguinte equação:

$$\hat{X}_m^W = \left[\hat{X}_{Pnt1}^W \dots \hat{X}_{PntN}^W \right]^T \quad (5.8)$$

A estimação de um ponto j no referencial do veículo e do mundo é expressa pelos seguintes vetores:

$$\hat{X}_{Pntj}^V = \begin{bmatrix} \hat{x}_{Pntj}^V \\ \hat{y}_{Pntj}^V \end{bmatrix}, \quad \hat{X}_{Pntj}^W = \begin{bmatrix} \hat{x}_{Pntj}^W \\ \hat{y}_{Pntj}^W \end{bmatrix} \quad (5.9)$$

Quando uma nova baliza é detetada pelo laser esta não é de imediato adicionada à lista de balizas. Inicialmente as coordenadas da baliza são gravadas numa lista de balizas não mapeadas (unmapped point), $X_{UnMpPntj}$, até que tal baliza tenha sido observa um número mínimo de vezes considerável, para que esta seja considerada não falsa positiva, e consequentemente adicionada ao vetor de estado ($\hat{X}_m^W$). Tal restrição é imposta de forma a evitar que falsas balizas, como material de possível reflexão, espalhado pela área de trabalho, seja adicionado à lista de balizas. O número mínimo de vezes que uma mesma baliza terá de ser vista está dependente de um parâmetro que

é explicado numa seguinte secção. Esta lista é representada no referencial do mundo da seguinte forma:

$$X_{UnMpPnt} = \begin{bmatrix} X_{UnMpPnt1} \dots X_{UnMpPntJ} \end{bmatrix}^T \quad (5.10)$$

onde J representa o número de balizas invariáveis representadas na lista de balizas não mapeadas, e o vetor $X_{UnMpPntj}$ é representado por:

$$X_{UnMpPntj} = \begin{bmatrix} x_{UnMpPntj} \\ y_{UnMpPntj} \\ Views \end{bmatrix} \quad (5.11)$$

Finalmente cada vez que uma baliza é vista um número de vezes suficiente é passada para o vetor de estado do mapa $\hat{X}_m^W$ sendo cada elemento N desta lista caracterizado da seguinte forma:

$$\hat{X}_{PntN}^W = \hat{X}_{ObsPnti}^W \Leftrightarrow \quad (5.12)$$

$$\hat{X}_{ObsPnti}^W = \begin{bmatrix} \hat{x}_{ObsPnti}^W \\ \hat{y}_{ObsPnti}^W \end{bmatrix} \quad (5.13)$$

5.3 Ciclo do Algoritmo de SLAM-EKF

Na presente sub-secção vai-se descrever mais detalhadamente todo o processo de EKF-SLAM. Embora este algoritmo partilhe de muitos dos passos referidos no capítulo de EKF, serão descritos novamente todos os passos de forma a criar independência entre os dois capítulos.

O algoritmo de EKF-SLAM é composto por três passos cruciais: 1) Previsão; 2) Associação; 3) Atualização;

O primeiro passo, o de **Previsão**, é exatamente igual ao passo referido no algoritmo de localização baseado em EKF (capítulo 4). Tem como objetivo o uso do algoritmo de movimento previamente enunciado, no capítulo 2, e a odometria do veículo de forma a estimar o estado atual do veículo $X(k+1|k)$ e a respetiva covariância $P(k+1|k)$.

Durante a observação das balizas, dá-se a recolha das posições através do sensor de laser, $Z(k+1)$, relativamente ao referencial do veículo. De seguida no passo de **Associação** é utilizada a previsão da pose do veículo, $X(k+1|k)$, e a informação proveniente do mapa, X_m^W , de forma a ser feita uma associação entre a posição prevista das balizas e a posição recolhida através do laser, ambas relativas ao referencial do mundo, $h_i(k+1)$. As observações sem associação no vetor de estado $\hat{X}(k+1|k)$ têm um índice -1 no vetor de associações $h_i(k+1)$. Para estas observações é feita a comparação com o vetor de balizas não mapeadas $X_{UnMpPnt}$. Esta correspondência é retornada nos vetores de associação de balizas não mapeadas, UnH_{Pnt} . Como anteriormente, as balizas que não têm correspondência nesta lista são identificadas com o índice -1.

No passo de **Atualização** o estado do filtro e a covariância, $X(k+1|k)$ $P(k+1|k)$, propagadas pela odometria são processados e atualizadas dando origem ao novo estado e a uma nova covariância $\hat{X}(k+1|k+1)$ e $P(k+1|k+1)$. Para tal são processadas as observações e as estimações das

balizas com correspondência no vetor $h_i(k+1)$. Neste passo são também atualizadas as listas de balizas não mapeadas, como se explica de seguida.

Todo o conjunto de balizas não mapeadas são indexadas com o número de vezes que estas foram visualizadas pelo laser e associadas. Assim sendo sempre que uma baliza encontra associação na no veto UnH_{Pnt} o seu índice é incrementado. Por fim, ainda no passo de atualização todos as balizas na lista da balizas não mapeadas, $X_{UnMpPnt}$, com um índice superior a um parâmetro, previamente definido, são adicionadas ao vetor de estado do mapa $\hat{X}_m^W$.

No seguinte quadro é possível analisar o pseudo-código correspondente ao algoritmo de EKF explicado anteriormente.

Algorithm 2 Algoritmo de localização EKF-SLAM

Rquere: $\hat{X}(k|k), P(k|k), Odometria(Odo), BalizasObservadas(BObs)$

Bloco de Previsão $(\hat{X}(k+1|k), P(k+1|k) \leftarrow Previso(\hat{X}(k|k), Odo, P(k|k))$

$\hat{X}(k+1|k) \leftarrow X(k|k) + f_v$

$$\nabla f_p \leftarrow \begin{bmatrix} 1 & 0 & -\delta_{trans} \sin(\theta + \delta_{rot1}) \\ 0 & 1 & \delta_{trans} \cos(\theta + \delta_{rot1}) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\nabla f_v \leftarrow \begin{bmatrix} \cos(\theta + \alpha_{rot1} + \alpha_{rot1}) & -\frac{1}{2} \sin(\theta + \alpha_{rot1} + \alpha_{rot1}) \\ \sin(\theta + \alpha_{rot1} + \alpha_{rot1}) & \frac{1}{2} \cos(\theta + \alpha_{rot1} + \alpha_{rot1}) \\ 0 & 1 \end{bmatrix}$$

$$P(k+1|k) \leftarrow \nabla f_p \cdot P(k|k) \cdot \nabla f_p^T + \nabla f_v \cdot Q_v(k) \cdot \nabla f_v^T$$

Associação

$$\hat{h} \leftarrow \begin{bmatrix} l_x \cdot \cos(\hat{\theta}_v) + l_y \sin(\hat{\theta}_v) \\ l_y \cdot \cos(\hat{\theta}_v) - l_x \sin(\hat{\theta}_v) \end{bmatrix}$$

$$\nabla h_{Pntj}^X \leftarrow \begin{bmatrix} -\cos(\hat{\theta}_v) & -\sin(\hat{\theta}_v) & -l_x \sin(\hat{\theta}_v) + l_y \cos(\hat{\theta}_v) \\ \sin(\hat{\theta}_v) & -\cos(\hat{\theta}_v) & -l_x \cos(\hat{\theta}_v) - l_y \sin(\hat{\theta}_v) \end{bmatrix} (k+1|k)$$

$$V_j \leftarrow Z_{Pntj} - \hat{h}_{Pntj}$$

Atualização com balizas já existentes no filtro

$$\hat{X}(k+1|k+1) \leftarrow \hat{X}(k+1|k) + W(k+1)V(k+1)$$

$$\hat{X}(k+1|k+1) \leftarrow \hat{X}(k+1|k) + W(k+1)V(k+1)$$

$$W(k+1) \leftarrow P(k+1|k) \nabla h_{Pntj}^X{}^T [\nabla h_{Pntj}^X P(k+1|k) \nabla h_{Pntj}^X{}^T + R]^{-1}$$

Atualização das balizas existentes

$$\hat{X}(k+1|k+1) \leftarrow \hat{X}(k+1|k) + W(k+1)V(k+1)$$

$$W(k+1) \leftarrow P(k+1|k) \nabla h_{Pntj}^X{}^T [\nabla h_{Pntj}^X P(k+1|k) \nabla h_{Pntj}^X{}^T + R]^{-1}$$

$$W(k+1) \leftarrow P(k+1|k) \nabla h_{Pntj}^X{}^T [\nabla h_{Pntj}^X P(k+1|k) \nabla h_{Pntj}^X{}^T + R]^{-1}$$

Ao longo do presente capítulo a seguinte nomenclatura vai ser usada:

- $0^{(a) \times (b)}$ representa uma matriz de zeros com dimensões $(a) \times (b)$.
- $I^{(a) \times (b)}$ representa a matriz identidade com dimensões $(a) \times (b)$.
- $X(a) \times (b)$ representa uma matriz genérica X com dimensões $(a) \times (b)$.

5.3.1 Previsão

Este processo é igual ao já referido na secção 4.1.1. Ou seja ciclicamente através da odometria (Odo), do estado estimado anteriormente, $\hat{X}(k|k)$, e da covariância anterior, $P(k|k)$, é calculado um novo vetor de estado, $\hat{X}(k+1|k)$, e a nova covariância, $P(k+1|k)$.

Neste passo uma vez que as balizas são elementos estáticos no referencial do mundo, unicamente o estado do veículo, (x_v, y_v, θ_v) , será alterado com a informação proveniente da odometria. Considerando o modelo de movimento do veículo e atendendo ao vetor de estado, $X(k|k)$, temos :

$$f(X(k|k), Odo, q) = \begin{bmatrix} f_v(X_v(k|k), Odo, q_v) \\ X_{Pnt1} \\ \dots \\ X_{PntN} \end{bmatrix} \quad (5.14)$$

onde N representa o número de balizas mapeadas até ao momento, e o modelo cinemático $f_v(X_v(k|k), Odo, q_v)$ é previamente referido na equação 4.5.

O modelo cinemático do erro q_v foi modelado considerando o ruído branco com uma distribuição gaussiana de valor esperado igual a zero ($\hat{q}_v = 0$) e uma covariância Q_v . O ruído branco q_v é constituído pelo erro associado ao estado do veículo:

$$q_v = [\varepsilon_{xv} \varepsilon_{yv} \varepsilon_{\theta v}]^T \quad (5.15)$$

A estimação do filtro depois do passo de previsão é obtida através da estimação do estado anterior $\hat{X}(k|k)$ e da informação proveniente da odometria (Odo) como demonstrado na seguinte equação:

$$\hat{X}(k+1|k) = f(X(k|k), Odo, q_v) \Leftrightarrow \begin{bmatrix} \hat{x}_v(k|k) + \delta_{trans} \cos(\hat{\theta}_v(k|k) + \delta_{rot1}) \\ \hat{y}_v(k|k) + \delta_{trans} \sin(\hat{\theta}_v(k|k) + \delta_{rot1}) \\ \hat{\theta}_v(k|k) + \delta_{rot1} + \delta_{rot2} \\ \hat{X}_{Pnt1}(k|k) \\ \dots \\ \hat{X}_{PntN}(k|k) \end{bmatrix} \quad (5.16)$$

Sabendo os modelos do sistema e o modelo do erro, é então possível calcular a covariância associada à previsão, $P(k+1|k)$:

$$P(k+1|k) = \nabla f_p \cdot P(k|k) \cdot \nabla f_p^T + \nabla f_u \cdot Q_v(k) \cdot \nabla f_u^T \quad (5.17)$$

A jacobina da transição do modelo em ordem ao estado do filtro é definida por:

$$\nabla f_{qv} = \frac{\nabla f}{\delta X} \Leftrightarrow \begin{bmatrix} \nabla f_{X_v}(\hat{X}(k|k), Odo) & \mathbf{0}^{(3) \times (2N)} \\ \mathbf{0}^{(2N) \times (3)} & I^{(2N) \times (2N)} \end{bmatrix} \quad (5.18)$$

onde ∇f_{X_v} é a jacobiana do modelo do movimento f_v com respeito ao estado do veículo $X_v(k)$, e é obtida da seguinte forma:

$$\nabla f_p = \begin{bmatrix} 1 & 0 & -\delta_{trans} \sin(\theta + \delta_{rot1}) \\ 0 & 1 & \delta_{trans} \cos(\theta + \delta_{rot1}) \\ 0 & 0 & 1 \end{bmatrix} \quad (5.19)$$

A Jacobina do modelo de transição em ordem ao erro da odometria q_v , é descrito pela seguinte equação:

$$\nabla f_{qv} = \frac{\delta f}{\delta q_v} \Leftrightarrow \begin{bmatrix} I^{(3) \times (3)} & \mathbf{0}^{(3) \times (N)} \\ \mathbf{0}^{(N) \times (3)} & \mathbf{0}^{(N) \times (N)} \end{bmatrix} \quad (5.20)$$

A covariância Q_v aumenta com a translação e rotação estimada do veículo:

$$Q_v = \begin{bmatrix} [\delta_{trans} \cdot \cos(\theta + \delta_{rot1}) \cdot \sigma_{\delta_{trans}}]^2 & 0 & 0 \\ 0 & [\delta_{trans} \cdot \sin(\theta + \delta_{rot1}) \cdot \sigma_{\delta_{trans}}]^2 & 0 \\ 0 & 0 & (A)^2 + (B)^2 \end{bmatrix} \quad (5.21)$$

onde $A = \delta_{trans} \cdot \sigma_{d\theta}$, $B = (\delta_{rot1} + \delta_{rot2}) \cdot \sigma_\theta$ e $\sigma_{\delta_{trans}}$, $\sigma_{\delta_{rot1}}$, $\sigma_{\delta_{rot2}}$, $\sigma_{\delta_{d\theta}}$ são erros de desvio os quais serão explicados no final do capítulo. Ou seja, quando o veículo anda um metro em frente, o erro de translação e rotação estimado é $\sigma_{\delta_{trans}}$, e $\sigma_{\delta_{d\theta}}$ respetivamente. Quando o veículo faz um movimento de translação de um radiano o erro estimado da rotação proveniente da odometria é igual a $\sigma_{\delta_{rot1}} + \sigma_{\delta_{rot2}}$.

5.3.2 Associação

No modulo de deteção de balizas obtém-se um conjunto de pontos com coordenadas referentes ao referencial do veículo. A lista de pontos referentes às posições das balizas observadas é definida pela seguinte equação:

$$PList^V = [X_{ObsPnt1}^V \dots X_{ObsPntN}^V] \quad (5.22)$$

onde X_{ObsPnt}^V é igual à equação 5.12, e representa as coordenadas de uma baliza. N representa o número de balizas observadas presentes na lista referida.

Estes pontos são passados para o referencial do mundo através da equação 4.3, e a lista que os contém é definida pela seguinte equação:

$$PList^W = [X_{ObsPnt1}^W \dots X_{ObsPntN}^W] \quad (5.23)$$

Tendo então a lista de balizas observadas, $PList^W$, e o estado do mapa, $\hat{X}_m^W$ é então possível fazer a associação entre as balizas observadas e as mapeadas. No caso de não existir nenhuma associação entre as observações e o estado do filtro ou entre as observações e a lista de balizas não mapeadas, $X_{UnMpPnt}$, é então adicionada esta nova baliza à lista de balizas não mapeadas como se irá explicar de seguida.

Durante o bloco de associação, são comparadas as balizas observadas e as balizas mapeadas de onde se retorna um vetor H_{Pnt} com valor -1 se uma baliza não for associada e valor 1 se for associada. No caso das balizas observadas que não foram associadas com a lista de balizas mapeadas é corrido o mesmo bloco mas desta vez faz-se a comparação entre as balizas observadas não associadas anteriormente e a lista de balizas não mapeadas, retornando neste caso o vetor UnH_{Pnt} .

Para que uma associação seja feita entre uma baliza da lista de observações $PList^W$, e uma baliza do estado do filtro $\hat{X}_m^W$, é necessário que as seguintes condições verifiquem:

1. A distancia euclidiana entre $X_{ObsPntN}^W$ e $\hat{X}_{Pntj}^W$ seja menor que o parâmetro ajustável $\delta_{dBaliza}$.
2. Dos pares que passem a condição anterior só se utilizará única e exclusivamente a menor distancia entre os pares.

No caso da comparação entre balizas observadas e balizas não mapeadas, embora os parâmetros de comparação sejam iguais aos referidos anteriormente, em caso de associação é incrementada a variável *views*, referente ao número de vezes que a respetiva baliza foi associada. Se no final deste bloco ainda existirem balizas observadas que não tenham sido associadas, estas são adicionadas à lista de balizas não mapeadas, com a variável *view* a zero, de forma a mais tarde, se vistas um número de vezes suficientes, serem inseridas no vetor de estado do mapa.

5.3.3 Atualização

Nesta sub-secção é considerado que o vetor do estado estimado é constituído por N posições invariantes de balizas. Assim o vetor de estado do filtro estimado, tendo em conta que este é constituído pelo vetor estado do veículo e o vetor de balizas mapeadas, é representada por $(3+2N)$. Ou seja o vetor de estado do veículo tem três dimensões associadas e cada baliza duas.

Tendo então a lista de balizas observadas, $PList^W$, o estado do veículo, $\hat{X}_v(k+1|k)$, e o estado do mapa, $\hat{X}_m^W(k+1|k)$, o conjunto de balizas associadas, H_{Pnt} , é então possível fazer a atualização do estado filtro de forma a fundir todas as informações, e obter uma estimativa da pose do veículo o mais precisa possível, $\hat{X}_v(k+1|k+1)$, e um posicionamento das balizas corrigido $\hat{X}_m^W(k+1|k+1)$.

Tal atualização vai ser criteriosamente explicada ao longo da seguinte sub-seção onde é explicada como é feita a atualização de todo o filtro e como são adicionadas novas balizas ao estado do filtro.

Inicialmente é feita a seleção dos pares possíveis de associar, e é retornado um vetor H_{Pnt} com esta informação, como explicado na sub-seção anterior. Através deste vetor é então possível obter a estimação da observação $\hat{h}_i$. Ou seja $\hat{h}_i$ é a conversão das da posição das balizas mapeadas, associadas, do referencial do mundo para o referencial do veículo considerando a posição estimada atual do veículo .

No presente caso de observação das balizas considera-se que o modelo de observação da baliza i é representado pela seguinte equação:

$$h_i = \begin{bmatrix} x_{Pnti}^V \\ y_{Pnti}^V \end{bmatrix} \quad (5.24)$$

A estimação deste modelo é representada por

$$\hat{h}_i = \begin{bmatrix} \hat{x}_{Pnti}^V \\ \hat{y}_{Pnti}^V \end{bmatrix} (k+1|k) \Leftrightarrow \quad (5.25)$$

$$\hat{h}_i = \hat{X}_{Pnti}^V(k+1|k) \quad (5.26)$$

Uma vez que as balizes presentes no vetor do estado do filtro se encontram posicionadas relativamente ao referencial do mundo é necessário fazer a conversão para o referencial do veículo tendo em consideração a pose estimada do mesmo, $\hat{X}_v$. Usando a equação 4.2, a estimação da observação para cada ponto i , pode ser calculada de seguinte maneira:

$$l_x = x_m - \bar{x}_v, l_y = y_m - \bar{y}_v \quad (5.27)$$

$$\hat{h} = \begin{bmatrix} l_x \cdot \cos(\hat{\theta}_v) + l_y \sin(\hat{\theta}_v) \\ l_y \cdot \cos(\hat{\theta}_v) - l_x \sin(\hat{\theta}_v) \end{bmatrix} \quad (5.28)$$

A observação de uma baliza pode ser representada pela seguinte equação:

$$\begin{aligned} Z_{Pnti} &= h_{Pnti} + r_{Pnti} \Leftrightarrow \\ Z_{Pnti} &= \begin{bmatrix} x_{Pnti}^V + \epsilon_{xi} \\ y_{Pnti}^V + \epsilon_{yi} \end{bmatrix} \end{aligned} \quad (5.29)$$

onde o erro r_{Pnti} e o erro associado com a observação da baliza i , modelada como ruído gaussiano de média zero e covariância R_{Pnti} . O erro r_{Pnti} é representado pelo seguinte vetor:

$$r_{Pnti} = \begin{bmatrix} \epsilon_{xi} \\ \epsilon_{yi} \end{bmatrix} \quad (5.30)$$

A matriz de covariância associada ao modelo de erros das observações, e dado pela seguinte expressão:

$$R_{Pnti} = \begin{bmatrix} \sigma_x & \sigma_{xy} \\ \sigma_{yx} & \sigma_y \end{bmatrix} \quad (5.31)$$

onde σ_x e σ_y são valores retirados do datsheet do laser e σ_{xy} é zero uma vez que não existe nenhuma relação entre os parâmetros x e y .

O gradiente do modelo de observação h_{Pnti} em ordem ao ruído da observação das balizas r_{Pnti} é uma matriz identidade enquanto que em ordem ao vetor de estado na baliza i , é igual a ∇h_{Pnti}^X , e é obtido através das seguintes equações [7]

$$\nabla h_{Pnti}^X = \begin{bmatrix} \nabla h_{Pnti}^{X_v} & \nabla h_{Pnti}^{X_m^W} \end{bmatrix} \quad (5.32)$$

$$\nabla h_{Pnti}^{X_v} = \begin{bmatrix} -\cos(\hat{\theta}_v) & -\sin(\hat{\theta}_v) & -lx \sin(\hat{\theta}_v) + ly \cos(\hat{\theta}_v) \\ \sin(\hat{\theta}_v) & -\cos(\hat{\theta}_v) & -lx \cos(\hat{\theta}_v) - ly \sin(\hat{\theta}_v) \end{bmatrix} (k+1|k) \quad (5.33)$$

$$\nabla h_{Pnti}^{X_m^W} = \begin{bmatrix} 0^{(2) \times (2j-2)} & \nabla h_{Pnti}^{X_{Pnti}^W} & 0^{(2) \times (N-2i)} \end{bmatrix} \quad (5.34)$$

$$\nabla h_{Pnti}^{X_{Pnti}^W} = \begin{bmatrix} \cos(\hat{\theta}) & \sin(\hat{\theta}_v) \\ -\sin(\hat{\theta}_v) & \cos(\hat{\theta}_v) \end{bmatrix} \quad (5.35)$$

Obtidas as informações correspondentes às associações entre as balizas observadas e a posição das balizas estimadas é então possível calcular, ente as associações, o quanto as posições diferem tendo este calculo o nome de Inovação. A inovação,tem como objetivo ajudar numa estimativa de quanto o veículo esta errado na determinação da sua pose e é calculada da seguinte forma:

$$V_{Pnti} = Z_{Pnti} - \hat{h}_{Pnti} \Leftrightarrow \quad (5.36)$$

$$V_{Pnti} = X_{ObsPnti}^V - \hat{X}_{Pnti}^V \quad (5.37)$$

Assim sendo, o vetor de inovação das posições invariantes das balizas pode ser representado por:

$$V_{Pnt} = \begin{bmatrix} V_{Pnt1} \dots V_{PntN} \end{bmatrix}^T \quad (5.38)$$

O gradiente final em ordem ao estado do filtro é representado através da seguinte expressão:

$$\nabla h_{Pnt}^X = \begin{bmatrix} \nabla h_{Pnt1}^X & \dots & \nabla h_{PntN}^X \end{bmatrix} \quad (5.39)$$

Por fim o resultado da matriz de covariância associada ao modelo de erro para todas as balizas observadas é expressa pela seguinte equação:

$$R_{Pnt} = \begin{bmatrix} R_{Pnt1} & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & R_{Pnti} \end{bmatrix} \quad (5.40)$$

Finalmente com todos os paramentos calculados basta aplicar a equações de atualização do filtro. Assim sendo o novo vetor do estado estimado, $X(k+1|k+1)$, é obtido usando as equações do EKF:

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + W(k+1)V(k+1) \quad (5.41)$$

onde $W(k+1)$ representa o ganho de Kalman. O ganho de Kalman corresponde à correção que irá ser feita de forma ao veículo se aproximar da pose real. O ganho de Kalman é igual a:

$$W(k+1) = P(k+1|k) \nabla h_{Pntj}^X{}^T [\nabla h_{Pntj}^X P(k+1|k) \nabla h_{Pntj}^X{}^T + R]^{-1} \quad (5.42)$$

finalmente a nova covaiância correspondente a $\hat{X}(k+1|k+1)$ é calculada da seguinte maneira:

$$P(k+1|k+1) = P(k+1|k) - W(k+1) \nabla h_{Pntj}^X P(k+1|k) \quad (5.43)$$

Ainda antes de terminar o processo de atualização e para terminar todo o ciclo do filtro é necessário adicionar as novas posições das balizas que tenham passado o teste correspondente ao mínimo de vez que estás foram visualizadas. Assim é corrido o bloco de nome "*true features*" onde se faz a verificação do parâmetro *views* correspondente a cada elemento da lista de balizas não mapeadas, $X_{UnMpPnt}$. Assim são corridos todos os elementos da lista de balizas não mapeadas $X_{UnMpPnti}$, e no caso do parâmetro view do elemento i for maior que um parâmetro σ_{views} , considera-se que a balizas foi observada um número de vezes aceitáveis para ser considerada real. O elemento i é então eliminado da lista de balizas não mapeadas e adicionado a sua posição e a sua covariância ao vetor do mapa do filtro, $\hat{X}_m^W(k+1|k+1)$ e à matriz de covariância, $P(k+1|k+1)$, respetivamente.

De seguida vai ser explicada com é feita a introdução de uma nova baliza no filtro. Antes de mais é de notar que a observação encontra-se no referencial do veículo sendo assim é necessário utilizar a equação 4.3 de forma a converter as coordenadas para o referencial do mundo. Assim a estimação de um ponto invariante como função da observação do laser, $X_{ObsPnti}^W$, é descrita da seguinte forma:

$$\hat{X}_{Pnti}^W = X_{ObsPnti}^W \Leftrightarrow \quad (5.44)$$

$$\hat{X}_{Pnti}^W = \begin{bmatrix} \hat{x}_{ObsPnti}^V \cos(\hat{\theta}_v) - \hat{y}_{ObsPnti}^V \sin(\hat{\theta}_v) + \hat{x}_v \\ \hat{x}_{ObsPnti}^V \sin(\hat{\theta}_v) + \hat{y}_{ObsPnti}^V \cos(\hat{\theta}_v) + \hat{y}_v \end{bmatrix} \quad (5.45)$$

Quando a observação é adicionada ao estado do filtro a nova estimação e representada pela

seguinte equação:

$$\hat{X}(k+1|k+1) = \begin{bmatrix} \hat{X}_v^T & X_{Pnt1}^{W^T} & \dots & X_{PntN}^{W^T} & X_{ObsPnti}^{W^T} \end{bmatrix}^T \quad (5.46)$$

Por fim a estimação da matrix da covariância é dada pela seguinte equação:

$$P(k+1|k+1) = \nabla h_{Pntj}^X P(k+1|k+1) \nabla h_{Pntj}^{X^T} + \nabla h_{Pntj}^{r_{Pnti}} R_{Pnti} \nabla h_{Pntj}^{r_{Pnti}^T} \quad (5.47)$$

onde o gradiente do modelo de observação em ordem ao estado do filtro é igual a [7]:

$$\nabla h_{Pnt}^X = \begin{bmatrix} \nabla h_{Pnt}^{X_v} & 0^{(3) \times (2N+2)} \\ 0^{(2N+2) \times (3)} & 0^{(2N+2) \times (2N+2)} \end{bmatrix} \quad (5.48)$$

$$\nabla h_{Pnt}^X = \begin{bmatrix} 1 & 0 & -x_{ObsPnti}^V \sin(\hat{\theta}_v) - y_{Pnti}^V \cos(\hat{\theta}_v) \\ 0 & 1 & x_{ObsPnti}^V \cos(\hat{\theta}_v) - y_{Pnti}^V \sin(\hat{\theta}_v) \end{bmatrix} (k+1|k+1) \quad (5.49)$$

O gradiente do modelo da observação em ordem ao ruído r_{Pnt} , é dado pela seguinte equação:

$$\nabla h_{Pnt}^{r_{Pnti}} = \begin{bmatrix} \cos(\theta_v) & -\sin(\theta_v) \\ \sin(\theta_v) & \cos(\theta_v) \end{bmatrix} (k+1|k+1) \quad (5.50)$$

5.3.4 Passagem de EKF-SLAM para EKF

O algoritmo EKF-SLAM é um algoritmo muito poderoso mas como previamente referido, também é muito pesado sendo este normalmente corrido em modo off-line. Assim depois de correr o algoritmo EKF-SLAM em modo online, e mapeadas todas as balizas presentes no layout de trabalho, é utilizada esta informação para correr o algoritmo de EKF, referido no capítulo 4. A grande diferença entre os dois algoritmos é que no algoritmo de EKF o estado do filtro é constituído unicamente pela posição do veículo:

$$X = \begin{bmatrix} x_v & y_v & \theta_v \end{bmatrix}^T \quad (5.51)$$

Concluindo o principal objetivo do algoritmo de EKF consiste em usar as balizas mapeadas previamente no algoritmo de EKF-SLAM, de forma a estimar unicamente o estado do veículo, $\hat{X}_v$. Assim sendo contrariamente ao algoritmo de EKF-SLAM no algoritmo de EKF:

1. O mapa previamente construído não é atualizado;
2. Novas balizas não são adicionadas ao vetor de estado do filtro;

5.4 Parâmetros do Filtro

Na presente secção será apresentada a justificação para os valores escolhidos dos parâmetros descritos ao longo da secção anterior.

Os valores utilizados relativamente aos parâmetros do laser, σ_x e σ_y , foram retirados do manual de instruções do instrutor, e representam a estimativa da covariância associada às medições do laser. Estes valores podem ser observados na tabela 5.1.

Tabela 5.1: Valores de erro do Laser

δ_x (m/m)	0,001
δ_y (m/m)	0,001

Foram também conduzidas repetidas experiências de forma a retirar os possíveis erros da odometria. Todas as experiências para os valores apresentados foram realizadas com o veículo a uma velocidade de 0.3 m/s. Assim com um ciclo de 100 milissegundos, o veículo mover-se-a, entre ciclos, a uma distância de 0,03 metros. Assumindo então que para o pior caso o ciclo demora 3 vezes a velocidade normal de todo o processamento, o veículo move-se 0,09 metros entre ciclos. Assim no corredor I - 175 (figura 6.1) foram feitos diversos ensaios com o robô a deslocar-se em linha reta e foram retirados os erros de translação, $\sigma_{\delta_{trans}}$, e rotação $\sigma_{d\theta}$ por metro. Também foram feitos ensaios onde o veículo fez rotações de 360° e de seguida medido o erro de rotação por radiano σ_{θ} . Assim foram obtidos os seguintes valores da tabela 5.2

Tabela 5.2: Valores de erro do Laser

$\sigma_{\delta_{trans}}$ (m/m)	0,015
$\sigma_{d\theta}$ (rad/m)	0,07
σ_{θ} (rad/ rad)	0,03

O parâmetro δ_{dBeac} , que define o valor máximo para o qual uma associação é feita. Uma vez que as balizas terão de estar afastadas um mínimo de 0,5 metros definiu-se o valor deste parâmetro como $\delta_{dBeac} = 0,25$ metros.

Por fim falta falar do valor mínimo de repetições para o qual uma baliza é considerada válida, o parâmetros σ_{views} é utilizado no modulo de Atualização de forma a evitar balizas falsas positivas no filtro. Uma vez que no melhor dos casos um ciclo do algoritmo o AGV desloca-se 0.03 metros, e assumindo que 0.5 metros com a visualização constante de uma baliza seria a distancia segura para definir uma baliza válida, escolheu-se o valor para este parâmetro de $\sigma_{views} = 17$.

Capítulo 6

Resultados Práticos

Este capítulo é dedicado à apresentação dos resultados obtidos na plataforma experimental descrita no capítulo 3. Inicialmente é feita uma breve descrição da pré preparação do espaço para a realização dos testes. Na secção 6.2 apresentam-se os testes relacionados com a precisão do algoritmo de localização EKF. Por fim são descritos vários métodos e técnicas utilizadas no decorrer dos testes de validação dos algoritmos criados.

6.1 Introdução

Os testes realizaram-se na Faculdade de Engenharia no departamento de Engenharia Eletrotécnica e Computadores(DEEC). Como assinalado a verde na figura 6.1 foram realizados no piso -1 do DEEC na Sala I -108 e no corredor I -175.

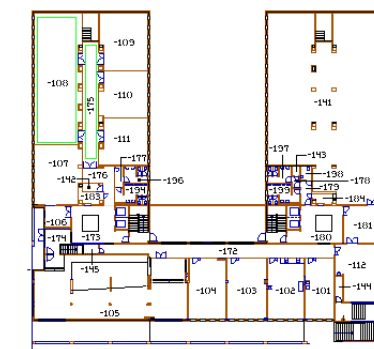


Figura 6.1: Planta do local de Testes

É também importante referir que no decorrer dos testes, um dos métodos de confirmação consistiu em usar os resultados de outro algoritmo de localização, de forma a validar os resultados do presente trabalho. Este algoritmo tem o nome de *perfect match* e foi validado por anteriores alunos de Mestrado e Doutoramento.

6.2 Resultados Práticos do Algoritmo de EKF

Inicialmente foi definido um percurso de cerca de 20 metros a ser percorrido pelo robô, unicamente na sala I -108. A representação deste percurso foi inserida no interface de Layouts, como representado na figura 6.2 a vermelho. De forma a facilitar a explicação dos resultados práticos, como é possível visualizar na figura 6.2, o mapa foi dividido em 4 zonas distintas. Assume-se também que o robô se desloca sempre no sentido crescente das zonas.

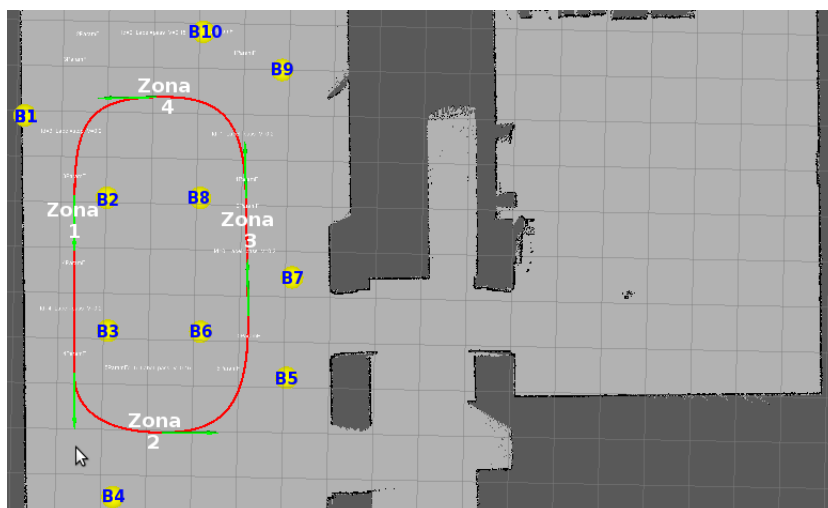


Figura 6.2: Mapeamento da sala I -108 e -109. A vermelho está representado o percurso definido, a amarelo estão representados as Balizas mapeadas

Depois do percurso definido fez-se um estudo de forma a encontrar a distribuição ideal do posicionamento das balizas. Para tal foram tidas em consideração regras de boa conduta definidas no manual do laser. Assim de maneira a que haja um optima localização é necessário que:

- Dentro do possível se garanta um igual número de refletores de ambos os lados do robô. Pois se os refletores só se encontrarem de um lado do robô, a viabilidade da localização é reduzida devido às condições geométricas.
- Garantir assimetria no padrão durante o posicionamento das balizas (como representado na figura 6.3), de forma a que o robô não se confunda em troços de caminho com aspeto idêntico.
- No posicionamento assimétrico das balizas deve-se garantir que a distância entre balizas varia um mínimo de 500 mm.

Assim, tendo em conta as considerações referidas anteriormente foram projetadas duas áreas para o posicionamento das 10 balizas de forma a se poderem conduzir variadas experiências. Existe uma área onde é sempre possível observar mais que 3 balizas(Zona 3 e 4), e outra área onde tal não acontece(Zona 1 e 2).

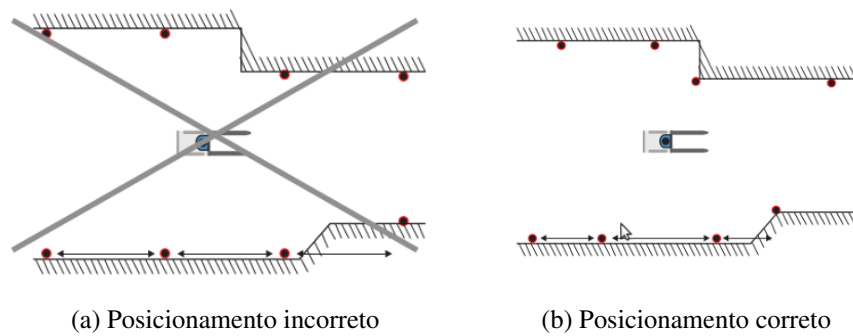


Figura 6.3: Representação da assimetria no posicionamento das balizas.

Na figura 6.2 e 6.4 pode-se observar o resultado do mapeamento e configuração real, respectivamente.

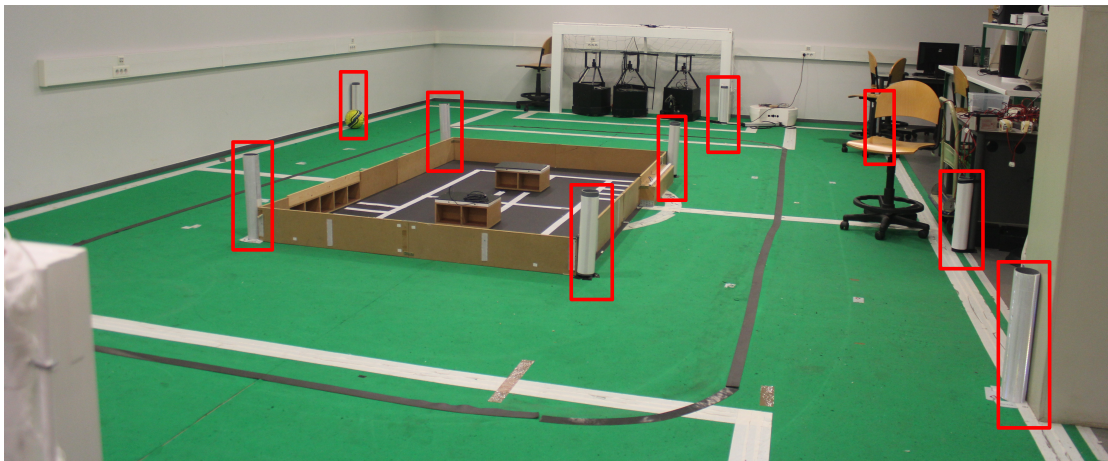
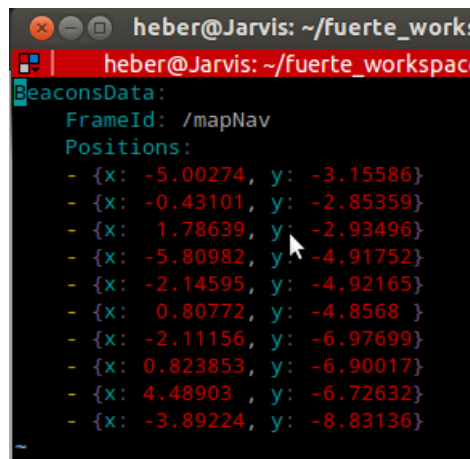


Figura 6.4: Posicionamento das balizas

Uma vez que o algoritmo de localização EKF foi desenvolvido antes do algoritmo de localização EKF-SLAM, houve a necessidade de arranjar uma forma alternativa de adquirir o posicionamento global das balizas. Para tal foi usado um algoritmo de localização previamente validado, de nome *Perfect Match* e foram feitas diversas voltas ao percurso adquirindo a posição de cada baliza. Assim utilizando este método foram adquiridas as seguintes coordenadas:

A terminal window with a black background and red and white text. The window title is 'heber@Jarvis: ~/fuerte_works'. The prompt is 'heber@Jarvis: ~/fuerte_workspace'. The output shows 'BeaconsData:' followed by 'FrameId: /mapNav' and 'Positions:' with a list of 10 coordinate pairs in curly braces.

```
heber@Jarvis: ~/fuerte_works
heber@Jarvis: ~/fuerte_workspace
BeaconsData:
FrameId: /mapNav
Positions:
- {x: -5.00274, y: -3.15586}
- {x: -0.43101, y: -2.85359}
- {x: 1.78639, y: -2.93496}
- {x: -5.80982, y: -4.91752}
- {x: -2.14595, y: -4.92165}
- {x: 0.80772, y: -4.8568 }
- {x: -2.11156, y: -6.97699}
- {x: 0.823853, y: -6.90017}
- {x: 4.48903, y: -6.72632}
- {x: -3.89224, y: -8.83136}
```

Figura 6.5: Coordenadas das balizas

Tendo todas as balizas inseridas no filtro, resta inserir a pose inicial por forma a poder iniciar o algoritmo. Para tal pôs-se o robô em cima da marca de pose inicial (figura 6.6) e foi medida a distância à origem do referencial. Uma vez que não foi desenvolvido nenhum algoritmo de identificação da pose inicial, a pose é sempre definida manualmente, tendo esta um erro inicial.

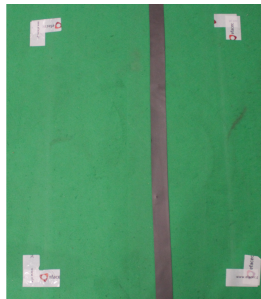


Figura 6.6: Marca de Pose inicial

Inicializados todos os parâmetros do filtro é então possível começar os testes práticos do algoritmo.

O primeiro teste consiste em comparar a saída do algoritmo de EKF com a saída do algoritmo de *Perfect Match*. Assim para este teste é considerada a premissa de que os resultados provenientes do algoritmo de *Perfect Match*, são resultados fiáveis e possuem um elevado grau de certeza.

Na figura 6.7 está representada a diferença de posição e orientação entre os algoritmos de EKF e *Perfect Match*.

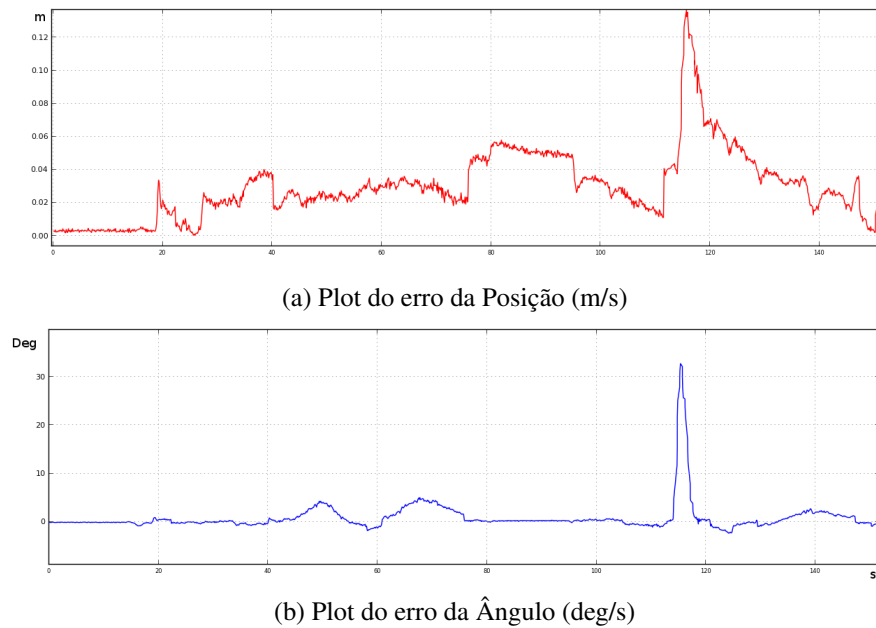


Figura 6.7: Plot dos erro entre o algoritmo EKF desenvolvido e o algoritmo de Perfect match .

No gráfico 6.7a está representada a diferença da posição calculada entre o algoritmo de EKF e o de *Perfect Match*. Neste caso o AGV começa o seu deslocamento na zona 3 visto ser uma zona abundante em balizas. Desta forma é possível afirmar que o erro entre os dois algoritmos é bastante baixo. É também possível observar que em zonas de menor quantidade de balizas, como é o exemplo do caminho entre as zonas 1 e 2 (100s a 120s), o erro tende a aumentar. Por fim, é de notar que no final do gráfico o AGV volta a entrar na zona 3 e o erro tende a diminuir o seu valor para o valor inicial, sendo assim possível concluir que o AGV volta a passar pela mesma posição inicial. Na figura 6.7b podem ser retiradas as mesmas conclusões relativamente à diferença dos ângulos que foram retiradas relativamente ao erro da posição 6.7a .

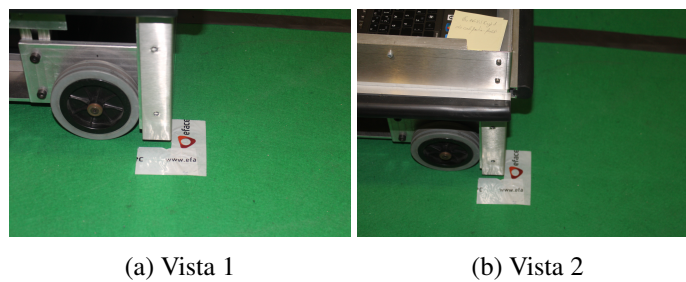
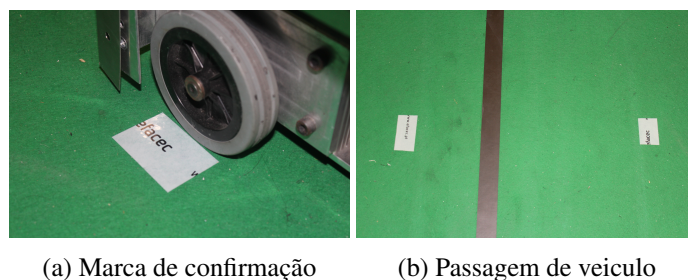


Figura 6.8: Marcas de posição inicial.

Na figura 6.10 estão representadas as trajetórias estimadas pelos dois algoritmos referidos previamente. Como já concluído anteriormente, é possível identificar as 2 áreas distintas. Na zona 3 e 4 as linhas andam sempre coincidentes. Pelo contrário, na zona 1 e 2 as linhas tomam caminhos ligeiramente desfasados.

A segunda experiência consistiu em definir marcas no chão com o objetivo de analisar se o percurso do AGV se mantinha inalterado em todas as iterações. De tal experiência foi possível



(a) Marca de confirmação

(b) Passagem de veículo

Figura 6.9: Marcas de análise, presentes na zona 3.

retirar duas observações. Na primeira como é possível observar na figura 6.9 o AGV passava a zona 3 sempre com a mesma posição global. A segunda observação pode ser retirada da figura 6.8 onde se observou que ao longo de diversas iterações o AGV pára sempre na mesma posição global.

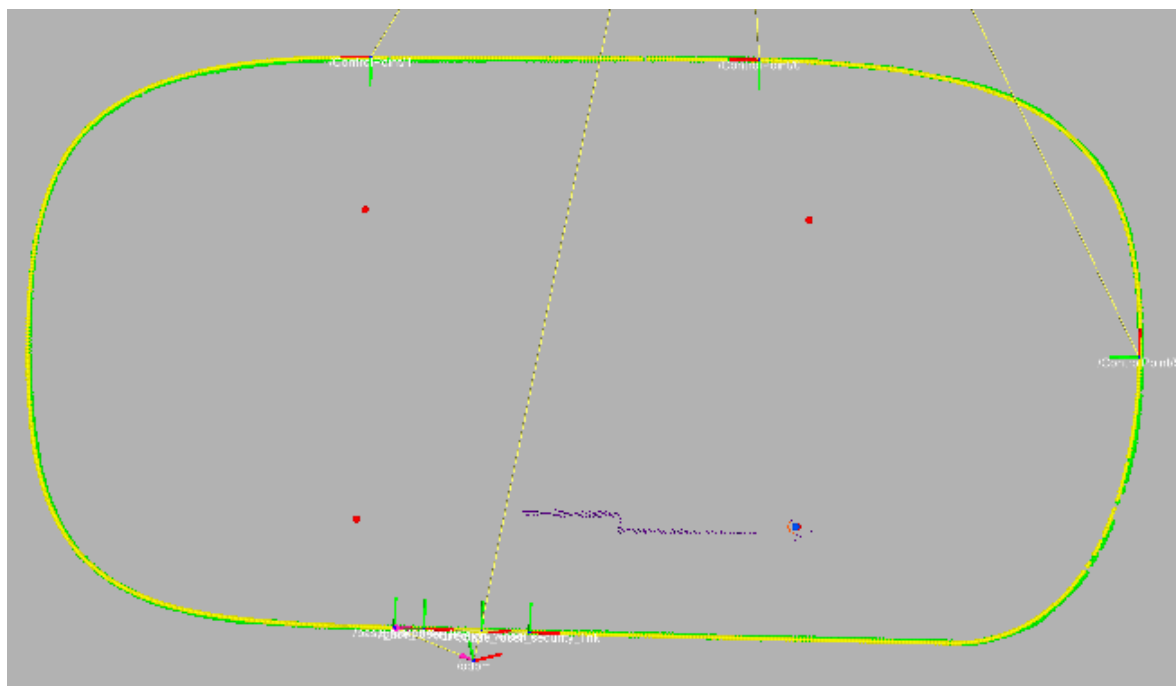


Figura 6.10: Desenho do percurso percorrido pelo AGV com os diferentes algoritmos. A amarelo é descrito o algoritmo de EKF, a vermelho o algoritmo de *Perfect Match*. As bolas vermelhas representam o mapeamento das balizas e as bolas azuis são as balizas visualizadas no momento

De forma a melhor compreender a influência das balizas no funcionamento do algoritmo, foram tapadas as balizas em parte da zona 1, na zona 2 e parte da zona 3 (nomeadamente as balizas B3, B4, B5 e B6). Assim foi feita uma volta ao circuito com as balizas destapadas e de seguida com as balizas tapadas o resultado pode-se analisar na figura 6.11. Pode-se então observar que na figura 6.11 (que representa a zona 2) quando o AGV observa a baliza B7 corrige ligeiramente a sua previsão da pose atual e à medida que começa a ver B9, B10 e B8 continua as correções

tendendo para a linha do percurso anterior. É também de referir que no final do percurso um vez não observadas as balizas tapadas B3 e B4 a sua previsão da sua pose acaba por ser errada, como é possível observar na figura 6.12. Tal não se observa na figura 6.11 visto o percurso ter parado neste ponto e não ter sido visualizada mais nenhuma balizas, impossibilitando assim ao algoritmo de EKF qualquer possível correção.

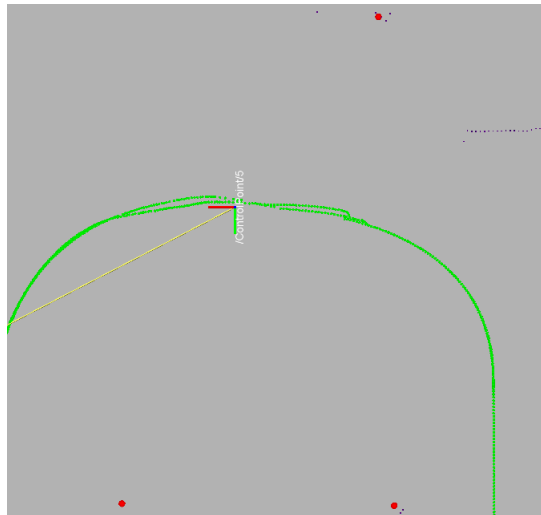


Figura 6.11: Erro da trajetória na zona 2

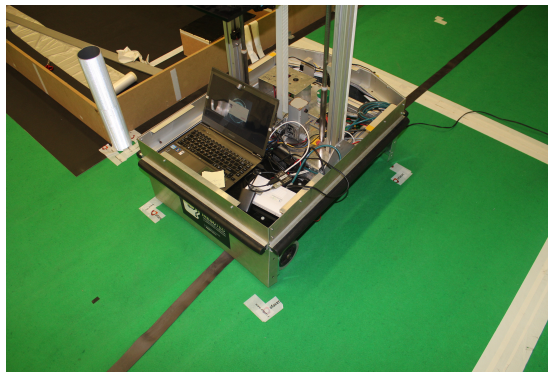


Figura 6.12: Erro da posição na marca de Pose inicial

Por fim, de forma a ter um termo de comparação entre o funcionamento do AGV com o algoritmo de EKF ou sem o algoritmo, baseando-se unicamente na odometria foram corridos vários ciclos com a função de atualização desligada. Os resultados podem ser analisados na figura 6.13 onde ciclo após ciclo o AGV passou a diferentes distâncias das marcas (figura 6.13b), e parou na marca de início com posições globais diferentes (figura 6.13a). Na figura 6.14 é possível observar a covariância calculada pelo algoritmo, representada por uma elipse em cima da representação do veículo, ou seja a área onde se prevê que o AGV se encontre.

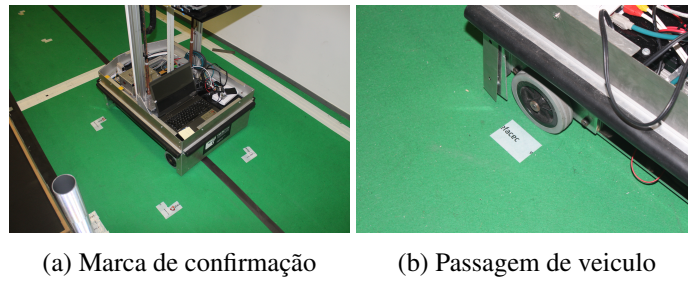


Figura 6.13: Marcas de análise, presentes na zona 3.

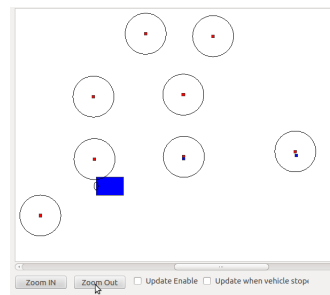


Figura 6.14: Covariância da pose. O retângulo a azul representa o robô e a elipse contida nele representa a covariância

6.3 Resultados Práticos do Algoritmo de EKF

Infelizmente o tempo de duração do presente projeto não foi o suficiente para conseguir concluir o algoritmo de SLAM. Assim nenhuns resultados serão demonstrados nesta secção.

Capítulo 7

Conclusão e Trabalhos Futuros

Neste capítulo são apresentadas as principais conclusões desta dissertação bem como descritas as perspectivas de desenvolvimentos futuros que podem surgir na sua sequência.

7.1 Conclusão

Efetivamente este trabalho permitiu aprofundar tanto conhecimentos na área de sistemas Robóticos móveis, nomeadamente no âmbito da sua orientação, como outros conhecimentos adquiridos ao longo do curso. Face ao resultado final desta dissertação, podemos concluir que os objetivos principais foram alcançados.

As soluções de localização baseadas em marcadores revelam-se muito vantajosas para aplicações com necessidades de flexibilidade e reconfiguração frequente. Conclui-se também que com a solução apresentada obtiveram-se bons resultados e provou-se que é possível passar de uma arquitetura de 2 lasers, para uma onde só é usado o laser de segurança reduzindo desta forma o custo de uma solução baseada em balizas indistinguíveis.

7.2 Trabalhos Futuros

A duração limitada desta dissertação não permitiu que fossem implementados e testados todos os sistemas que um projeto como este robô móvel industrial necessita. Na presente secção irão ser referidos alguns pontos de toda a arquitetura da plataforma utilizada, onde se pensa ainda ser possível a existência de futuras melhorias.

O ponto mais importante seria a finalização do algoritmo de EKF-SLAM, facilitando assim o mapeamento das áreas e consequente aumentando da facilidade do preenchimento dos parâmetros do filtro do algoritmo de EKF.

Embora o filtro desenvolvido no pré-processamento evite a aquisição de algumas balizas falsas positivas, por vezes ainda se verificaram aquisições erradas. Assim propõe-se que seja desenvolvido um sistema de interpolação, que com base no raio das balizas seja capaz de identificar o seu formato cilíndrico.

Foi também identificada a necessidade de desenvolver um sistema capaz de preencher a pose inicial do filtro de EKF, de forma a evitar a necessidade de existir a intervenção de um operador.

Por fim, seria importante melhorar o bloco de associação de balizas de forma a evitar qualquer associação errada com alguma baliza mais próxima, no caso de o AGV já se encontrar com um erro de posicionamento elevado.

Referências

- [1] Rui Paulo Rocha. Estado da arte da robótica móvel em portugal. Tese de mestrado, FCTUC, 2000.
- [2] David Da Silva Lima. Localização absoluta de robôs móveis em ambientes industriais. Tese de mestrado, FEUP, 2010.
- [3] Sebastian Thrun, Wolfram Burgard, Dieter Fox, et al. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2005.
- [4] Roland Siegwart e Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA, 2004.
- [5] Hugh Durrant-Whyte e Tim Bailey. Article. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, 2:2006, 2006.
- [6] ROS. Ros documentation @ONLINE.
- [7] Miguel Armando Migueis Pinto. Slam for 3d map building to be used in a matching localization algorithm. Tese de mestrado, FEUP, 2012.
- [8] Roberto José Magalhães da Silva. Localização de agvs industriais baseada em marcadores. Tese de mestrado, FEUP, 2011.
- [9] Les Kitchen Andrew Howard. Navigation using natural landmarks, article. 1, 2011.
- [10] M. CSorba. Simultaneous localization and map building, thesis. Tese de mestrado, Robotic Research Group Department of Engineering of Oxford, 1997.
- [11] S. Behling L. Schulze e S. Buhrs. Agvs in logistics systems state of the art, applications and new developments. Tese de mestrado, International Conference On Industrial Logistics (ICIL 2008), 2008.
- [12] S. Whitesides G. Dudek, K. Romanik. Localizing a robot with minimum travel, in: Proceedings of the sixth acm- slam symposium on discrete algorithms. *IEEE*, 1995.
- [13] Sascha Lange Martin Lauer e Martin Riedmiller. Calculating the perfect match: An efficient and accurate approach for robot self-localization. *University of Osnabrck, Institute of Cognitive Science and u Institute of Computer Science, 49069 Osnabr ck, Germany*, 2006.
- [14] Shung Han Cho e Sangjin Hong. Map based indoor robot navigation and localization using laser range finder. *Department of Electrical and Computer Engineering Mobile System Design Laboratory Stony Brook University - SUNY Stony Brook, NY, USA*, 2010.

- [15] Carlos Couto João Sena Esteves, Adriano Carvalho. Position and orientation errors in mobile robot absolute self-localization using an improved version of the generalized geometric triangulation algorithm. *IEEE*, 2006.