FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Fiabilidade e robustez da localização de robôs móveis

Héber Miguel Plácido Sobreira

Programa Doutoral em Engenharia Electrotécnica e de Computadores

Orientador: Prof. Dr. António Paulo Gomes Mendes Moreira

Co-orientador: Prof. Dr. Paulo José Cerqueira Gomes da Costa

24 de Janeiro de 2017

Fiabilidade e robustez da localização de robôs móveis

Héber Miguel Plácido Sobreira

Programa Doutoral em Engenharia Electrotécnica e de Computadores

24 de Janeiro de 2017

Documento escrito ao abrigo do antigo acordo ortográfico por opção do autor.

Resumo

A fiabilidade e robustez do sistema de localização de um robô móvel são fatores críticos no que toca aos níveis de desempenho e segurança com que este realiza as operações para as quais foi concebido. Um sistema de localização fiável é aquele que assegura níveis de precisão para um vasto leque de condições de funcionamento, permitindo a operação contínua do robô. Um sistema robusto é aquele que apresenta elevada tolerância a interferências externas, tendo a capacidade de detectar e reagir de forma adequada a falhas.

Com este trabalho, pretende-se desenvolver algoritmos que aumentem a fiabilidade e robustez dos sistemas de localização de forma a estender a aplicação de diferentes técnicas a outros domínios, nomeadamente sistemas industriais. Aqui, onde as soluções de navegação tendem a ser conservadoras, constitui-se um ambiente interessante para análise, no qual o sistema de localização se baseia na detecção de reflectores instalados a uma altura elevada em relação ao chão utilizando um sensor dedicado para esse efeito (laser de navegação). Desta forma, consegue-se uma boa qualidade sensorial, evitando oclusões e diminuindo a ocorrência de outliers. Foram efectuados desenvolvimentos tanto em sistemas que recorrem a reflectores como em sistemas que recorrem unicamente à geometria do meio circundante ao robô (contornos).

Em relação à localização baseada em reflectores, recorrendo a um algoritmo de fusão sensorial (filtro de kalman Estendido) e a um esquema de filtragem de *outliers*, desenvolveu-se um sistema com precisão e robustez suficientes para um aplicação industrial para reflectores instalados junto ao chão, a ser detectados por um laser de segurança, evitando-se assim o custo da utilização de um sensor instalado no topo do robô.

Em relação à localização baseada em contornos, foi feito um estudo comparativo entre diferentes algoritmos de *Map-Matching*. A partir desse estudo, foi evidenciada a relevância do algoritmo Perfect Match. Foram feitas diversas melhorias ao nível da robustez a *outliers* de forma a atingir os níveis exigidos numa aplicação industrial. Neste ponto, também foram abordadas as questões da localização global e da detecção de falhas.

Os desenvolvimentos efetuados no âmbito deste trabalho foram extensivamente testados, tanto em simulação como em robôs reais e em diferentes ambientes. Obtiveram-se resultados que evidenciaram a fiabilidade das soluções propostas, tanto ao nível da precisão como ao nível da robustez, sendo promissora a sua aplicação em ambientes exigentes, como é o caso das aplicações industriais.

Abstract

The reliability and robustness of a mobile robot's tracking system are critical to how well and safely it performs its tasks.

A reliable tracking system is one that assures accuracy levels fit for several different working conditions, allowing the robot to work continuously. A robust system is that which presents high tolerance against external interference, by detecting failures and properly reacting to them.

The goal of the present work is to develop algorithms that increase existing localization systems' reliability and robustness, so as to extend their application to other fields, namely to industrial systems. Herein, solutions tend to be conservative, and it becomes interesting to explore this environment, where the tracking system relies upon the detection of reflectors installed high above the ground, by means of a specific sensor (navigation laser).

This enables good sensory quality, avoiding occlusions and diminishing the occurrence of outliers. Both reflector based systems and systems that rely solely on the geometry of the environment surrounding the robot (contour-based) were developed.

Regarding reflector-based tracking systems, a sensory fusion algorithm (Extended kalman Filter) and an outlier filtering process were used to achieve a system with enough precision and robustness to be applied industrially, with the use of ground level reflectors, thus avoiding the cost associated to the installation of a sensor on top of the robot.

As for contour-based tracking, a comparative study between different Map Matching algorithms was performed. This study rendered the Perfect Match highly relevant. Several improvements towards the increase of tolerance to outliers were made, in order to achieve the levels required for industrial application. Global tracking and failure detection were also explored here.

All of the aforementioned developments were extensively tested, both through simulation and using real robots, in several environments. The results obtained brought to light the reliability of the proposed solutions, both in terms of precision and in terms of robustness; their application to demanding environments, such as industrial systems, is thus considered promising.

Agradecimentos

Este trabalho apesar da da sua natureza individual não teria sido possível sem as contribuições de um conjunto de pessoas, as quais tenho o gosto e honra de enunciar nas próximas linhas.

Começo pelos meus orientadores, Professor António Paulo Moreira e ao Professor Paulo Costa. Obrigado pela oportunidade de fazer um doutoramento convosco e por toda a confiança e apoio que depositaram no meu trabalho.

Aos meus colegas de trabalho, companheiros de luta e principalmente amigos, Filipe Santos, Luis Rocha, Miguel Pinto, Luis Oliveira, Carlos Costa, Miguel Oliveira, Tiago Pereira e Ivo Sousa quero que saibam que é uma honra fazer ciência convosco. Sem as nossas discussões e partilha de ideias este trabalho seria certamente muito mais pobre.

Helena Marinho nunca me esquecerei de todo o apoio e ajuda que me deste durante grande parte deste longo percurso. Este trabalho também foi feito à custa de muito sacrifício teu. Obrigado por tudo e desculpa.

Ao Ricardo Silva e ao Humberto Rocha, obrigado por me receberem em vossa casa, pelos jantares e pela amizade que tornaram muito mais agradável a escrita desta tese.

E finalmente quero deixar um agradecimento muito especial a três mulheres fantásticas, Ana Catarina Pinto, Ana Filipa Santos e Ana Sofia Barbosa. Vocês formam essenciais nesta fase final desta importante etapa da minha vida. Obrigado pelas revisões exaustivas, por não me deixarem desistir, pela ajuda incondicional, pelas palavras de apoio, pela amizade, pelas noitadas de escrita, pela escolta no dia da apresentação e por muito mais. Não sei se teria conseguido chegar ao fim sem o vosso apoio! Estou-vos eternamente grato!

Obrigado a todos!

Héber Sobreira

F1

"Se os robots tiverem uma religião politeísta, tu vais ser oficialmente parte integrante do olimpo!!"

Lady Ana Lee

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AGV	Automatic Guided Vehicle
AMCL	Augmented Monte Carlo Localization
CPU	Central Processing Unit
EKF	Extended Kalman Filter
FOV	Field Of View
HSV	hue, saturation and Value
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
LUT-ICP	Lookup Tables-Iterative Closest Point
MSL	Middle Size League
NDT	Normal Distributions Transform
PCL	Point Cloud Library
PM	Perfect Match
RPROP	Resilient Back-Propagation
SLAM	Simultaneous Localization and Mapping

Capítulo 1

Introdução

A fiabilidade e robustez do sistema de localização de um robô móvel são um factor crítico no que toca aos níveis de desempenho e segurança com que este realiza as operações para que foi concebido. O sistema de localização recorre aos dados adquiridos pelos sensores instalados no robô de forma a estimar a sua posição e orientação em relação a um referencial externo e fixo. O foco deste trabalho é optimizar os níveis de fiabilidade e robustez dos algoritmos de localização de forma a aplicar técnicas inovadoras, economicamente mais viáveis, a ambientes desafiantes. A fiabilidade é a capacidade de assegurar níveis de precisão para um vasto leque de condições de funcionamento, permitindo a operação contínua do robô. A robustez relaciona-se com a tolerância a interferências externas e com a capacidade de detectar e reagir de forma adequada a falhas.

O estado da arte na localização de robôs móveis é vasto e apresenta inúmeras soluções baseadas em diferentes tipos de algoritmos e diferentes tipos de sensores. No entanto, no que toca a ofertas comerciais, o número de soluções com sucesso é reduzido e aplicado em ambientes muitos controlados. Esta discrepância de avanço tecnológico deve-se tanto à falta de fiabilidade e robustez dos algoritmos como aos elevados custos associados a algumas tecnologias. Uma aplicação em que essa discrepância é bastante assinalável é a utilização de robôs móveis em ambiente industriais (Automatic Guided Vehicles, AGVs). Aqui, as soluções de navegação tendem a ser conservadoras no sentido de assegurar uma boa qualidade sensorial, constituindo portanto um ambiente interessante de análise. Neste sentido, é dada especial atenção à aplicação em ambientes industriais. Contudo, a aplicação noutros contextos também é analisada, nomeadamente a robôs de serviços. Parte-se do princípio que as tecnologias desenvolvidas neste âmbito sejam em certa medida transversais a várias áreas de aplicação.

Os AGVs já são utilizados na indústria há mais de 50 anos e, ao longo deste tempo, foram desenvolvidas diversas opções tecnológicas no que diz respeito a custo, robustez, precisão e flexibilidade. No caso concreto do sistema de localização, existem várias soluções baseadas em diferentes sensores, dentro das quais são mais usuais as soluções baseadas no seguimento de faixas e triangulação laser (Schulze e Wullner, 2006) (Schulze et al., 2008).

Dentro dos sistemas baseados no seguimento de faixas têm-se, por exemplo, os filo-guiados em que o AGV segue um caminho definido por um campo magnético criado por um condutor

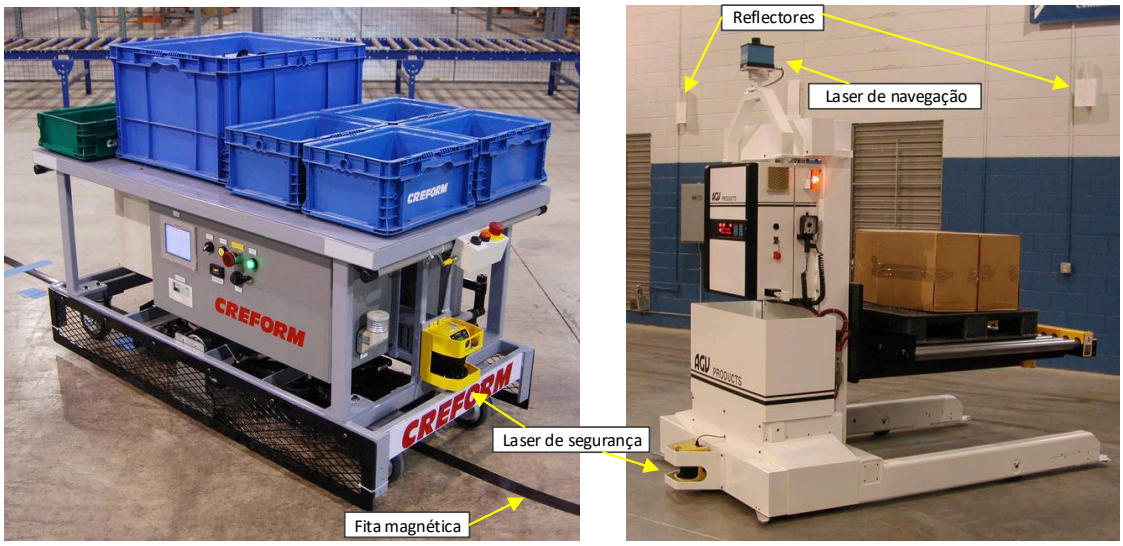


Figura 1.1: Exemplo de duas soluções de localização de AGVs. Esquerda: Sistema baseado no seguimento de fita magnética. Direita: Sistema baseado em triangulação laser.

implantado no pavimento. De forma similar, têm-se também sistemas baseados no seguimento de fita magnética e faixas coloridas desenhadas no pavimento (Figura 1.1, esquerda). Estes tipos de sistema têm sido largamente utilizados na indústria pelo seu relativo baixo custo e simplicidade. Mas, por outro lado, são pouco flexíveis, no sentido em que a alteração dos percursos efectuados pelo robô implica a alteração das faixas instaladas no pavimento. Consequentemente, são inadequados para instalações industriais em que ocorram frequentes alterações de *layouts* das linhas de produção.

Os sistemas baseados em triangulação *laser*, através de um laser rotativo (*laser* de navegação) colocado no topo do AGV e dos sinais reflectidos por um conjunto de reflectores (mínimo três) instalados no ambiente (Figura 1.1, direita), estimam com grande precisão a posição e orientação do AGV. Como tal a triangulação laser é considerada a tecnologia de localização mais precisa, robusta e flexível, sendo o tipo de solução mais frequente na localização de AGVs no meio industrial. A flexibilidade prende-se ao facto da navegação não se restringir a caminhos definidos por uma faixa instalada no chão mas sim a uma área com uma determinada granularidade de reflectores instalados. Ou seja, o problema de alteração dos caminhos percorridos pelo AGV resume-se a uma alteração das configurações do sistema de navegação. Por outro lado, esta solução possui um custo relativamente elevado face às anteriores. Tal deve-se tanto ao custo do sensor utilizado (*laser* de navegação) como ao custo da instalação dos reflectores na área de navegação (uma instalação de um sistema deste tipo implica normalmente a instalação de centenas de reflectores na área de navegação do AGV). Além disto, a tecnologia de triangulação *laser* pode apresentar problemas devido à detecção de falsos positivos ou em situações específicas como a necessidades de determinar a localização de um AGV num corredor estreito.

É também pertinente referir outra componente no contexto do presente trabalho: a existência obrigatória dos sistemas de seguranças dos AGVs. Os AGVs operam frequentemente em áreas onde existem pessoas, sendo premente assegurar e garantir a segurança das mesmas. Por isso, deve existir um sistema que detecte o perigo de eventuais colisões e que imobilize o AGV atempadamente. Inclusivamente, existem normas de segurança que obrigam ao uso de lasers de segurança a partir de determinada velocidade/peso do AGV. Na Figura 1.1 temos dois exemplos de diferentes AGVs (um utiliza fita magnética e outro triangulação laser) e ambos utilizam um laser de segurança junto ao chão de forma a detectar obstáculos.

1.1 Objectivos e motivações

O objectivo deste trabalho é a análise de diferentes técnicas de localização aplicadas a robôs móveis e o desenvolvimento de algoritmos que visam um aumento da sua precisão e robustez. As aplicações industriais são tipicamente muito exigentes em termos de precisão e robustez, sendo portanto um ambiente adequado para validar as soluções propostas. Mais concretamente, o objectivo é abordar os seguintes casos de estudo:

- Localização baseada na detecção de reflectores utilizando o *laser* de segurança;
- Localização baseada em contornos medidos por um *laser* de navegação;
- Detecção e recuperação de falhas do sistema de localização.

Com isto, pretende-se acrescentar valor e inovação em relação às soluções existentes, a nível económico, de flexibilidade e de facilidade de instalação, mantendo a robustez e precisão adequadas à aplicação em causa.

1.1.1 Localização baseada na detecção de reflectores utilizando o *laser* de segurança

Como já foi referido, o *laser* de segurança é um equipamento obrigatório na maioria das aplicações que usam AGVs. Assim sendo, o reaproveitamento deste equipamento na localização liberta-nos da necessidade da utilização de um segundo *laser* para navegação no topo no robô, o que permite uma redução de custos em equipamento.

Além disso, em algumas aplicações, não é possível utilizar um laser de navegação devido a limitações da altura máxima do AGV, como se pode observar no caso particular representado na Figura 1.2. Neste caso o AGV transporta mesas dotadas de rodas, sendo obrigatoriamente um robô baixo o suficiente para conseguir passar por baixo das referidas mesas. Desta forma não é possível a utilização de um *laser* de navegação elevado em relação ao chão como se vê na Figura 1.1 à direita.

Um possível indicador do valor de mercado para uma potencial solução baseada em localização com reflectores e *laser* de segurança é o facto de já existirem fabricantes que fornecem modelos de *lasers* de segurança com a capacidade de detectar reflectores.

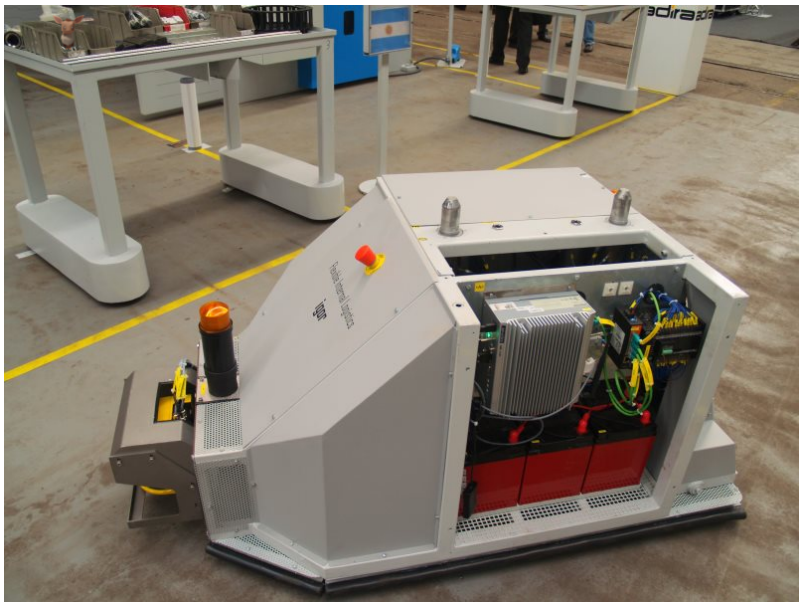


Figura 1.2: Exemplo de uma aplicação com AGVs. O veículo transporta mesas com rodas que levam material para abastecer linhas de montagem com matérias-primas.

Todavia, existem vários entraves à utilização deste tipo de solução. O primeiro entrave tem a ver com a abertura angular do laser de segurança, que é menor do que a de um laser de navegação (tipicamente a abertura angular de um laser de navegação é de 360°, enquanto, num laser de segurança é 190° ou 270°, dependendo do sítio onde é instalado no AGV). Em segundo lugar, existe também a questão da altura em relação ao chão a que está o laser de segurança. Estando numa posição mais baixa, a possibilidade de oclusão de reflectores é maior, bem como a possibilidade da detecção de falsos positivos porque, tipicamente, existe uma maior quantidade/diversidade de objectos a uma distância curta do chão. A utilização de um *laser* de segurança exige portanto algoritmos que assegurem maior robustez do que os algoritmos utilizados em sistemas com *lasers* de navegação. É portanto notória a necessidade de novos algoritmos, bastante mais robustos e eficientes/fiáveis, que suportem os anteriormente referidos falsos positivos e ao mesmo tempo uma quantidade inferior de informação, sem que haja uma degradação significativa da precisão da localização nem que haja a entrada frequente em situação de perdido, ou seja, falha completa do sistema de localização.

1.1.2 Localização baseada em contornos medidos por um *laser* de navegação

Em primeiro lugar, convém esclarecer que "contornos", neste contexto, dizem respeito ao conjunto de distâncias/ângulos a objectos detectados (paredes, portas, mobília, etc.) por um *laser range finde*r rotativo. Assim, os contornos constituem uma representação bidimensional da área livre ao nível do sensor. Actualmente, existem modelos industriais de *lasers* de navegação e de segurança que fornecem a informação dos contornos medidos. Neste caso, a localização é

conseguida através de uma correspondência entre as medidas do laser e de uma representação do espaço na memória interna do robô (ao qual nos referimos como mapa).

Com esta abordagem, tem-se o benefício de evitar o custo de instalação de reflectores (cujo número ascende tipicamente às centenas) para além de se minimizar a interferência com o ambiente, pois a instalação de quantidades elevadas de reflectores pode ser considerada como inestética e inaceitável para alguns clientes. Por outro lado, espera-se que, ao contrário da triangulação *laser*, esta abordagem não apresente problemas em localizar o AGV em corredores estreitos.

A localização baseada em contornos, também conhecida por "*environment-based navigation*" (Tomatis, 2011), não é um problema novo. Na realidade, este problema tem sido alvo de intensos estudos pela comunidade científica desde os anos 90, mas até os dias de hoje este tipo de soluções não tem vingado no mundo industrial, certamente devido à sua baixa robustez e fiabilidade. Tomatis (2011), embora não especifique, fala em centenas de pequenos problemas em cenários reais e de necessidades específicas da indústria. São estes problemas e necessidades específicas que pretendemos analisar juntamente com parceiros industriais.

Uma das principais dificuldades que este tipo de tecnologia enfrenta é a detecção pelo *laser range finder* de objectos não mapeados (pessoas em movimento, alteração da disposição de objectos), em ambientes pouco estruturados. Nestas situações, a robustez e precisão deste sistema podem ser fortemente afectadas, tornando inviável a sua utilização em AGVs industriais, que devem ter sempre grande precisão e muito raramente algum problema de falha de localização.

1.1.3 Detecção e recuperação de falhas do sistema de localização

Por motivos relacionados com o peso computacional, é habitual recorrer-se a algoritmos de pose *tracking* para resolver o problema da localização. Isto pressupõe a existência de uma estimativa aproximada da verdadeira pose do robô com um determinado erro máximo. Esta estimativa é obtida a partir da combinação da última pose estimada com os dados de deslocamento relativo, obtidos a partir de encoders, IMUs, etc. Contudo, falhas e/ou interferências nos sensores podem afectar a estimativa inicial com um erro significativo, o que leva o algoritmo de pose *tracking* a divergir da solução desejada. Uma causa típica desta divergência é a derrapagem das rodas que leva a erros grosseiros na medida do deslocamento relativo do robô medido pelos seus *encoders*. Nesta situação, o algoritmo de *tracking* fornece continuamente soluções erradas e muito dificilmente recupera sem intervenção externa. Se esta situação não for detectada pode levar o robô a desviar-se da trajectória pretendida, pondo a segurança da sua operação em risco. Neste sentido, as questões da detecção de falhas e a sua recuperação serão abordadas. Mais concretamente, será abordada a detecção da divergência do algoritmo de *tracking* da pose e a localização global, isto é, a estimação da pose absoluta na situação em que não existe nenhuma estimativa prévia da pose do robô.

1.2 Requisitos gerais do sistema de localização

O sistema de localização tem como objectivo estimar a pose absoluta do robô (X_v), isto é, a sua posição e orientação num referencial fixo e externo ao robô (W_x W_y). Em traços gerais, o processo de estimação da pose absoluta de um robô é feito a partir da procura de uma correspondência entre informação fornecida pelos sensores e uma descrição do meio envolvente onde o sistema vai operar. Neste, documento vamos passar a referir-nos à informação sensorial através da palavra "observações", e à representação do meio envolvente como "mapa". Aqui, observações correspondem a um conjunto de posições relativas a um referencial local ao robô (R_x R_y), enquanto o mapa é referenciado ao mesmo referencial de X_v. Como já foi referido, neste trabalho em concreto, analisaremos duas classes de observações: a detecção de reflectores, e contornos.

Teremos também os seguintes requisitos:

- Na perspectiva de utilização industrial é necessário ter uma precisão/repetibilidade da estimativa da pose absoluta do robô com qualidade suficientemente alta para permitir ao AGV deslocar-se em segurança, como é o exemplo da deslocação entre postos de trabalho e da realização de manobras de *docking*. A precisão do sistema de localização do AGV durante a deslocação entre postos de trabalho tem de ser suficiente para que o robô não saia acidentalmente da área destinada à sua circulação. Os requisitos ao nível da repetibilidade nos casos de manobras de *docking* são tipicamente mais exigentes. A configuração dos pontos de *docking* é feita muitas vezes por demonstração.

- É essencial quantificar a precisão da estimativa da pose do robô, X_v, de forma a verificar se é possível realizar em segurança a navegação do AGV. Esta está dependente de factores externos ao robô (como por exemplo, a disposição relativa dos reflectores ao robô, a estrutura dos contornos observados, oclusões, etc.) e deve por isso ser feita de forma contínua.

- O AGV deve ser robusto a interferências do meio envolvente. Além do ruído inerente às medidas das observações, o sistema de localização é também sujeito a observações corrompidas com erros grosseiros, como por exemplo falsos positivos (detecção de falsos reflectores e objectos não mapeados), oclusões de reflectores, etc.

- É importante o uso da redundância na detecção de falhas, de modo a aumentar as garantias de segurança da navegação do AGV. É recorrente confrontar o resultado do deslocamento relativo dado pelos *encoders* e o mesmo resultado obtido por estimativa da pose absoluta.

- É muito importante a existência de um mecanismo de detecção de falha do algoritmo de localização, ou seja, que detecte que o robô está perdido e a fornecer estimativas da pose do robô totalmente erradas. Conforme a aplicação, poderá ser de interesse, nesses casos, parar e gerar alarmes ou tentar uma manobra/algoritmo de recuperação.

Estes requisitos são transversais aos chamados robôs de serviços, mas, no caso dos AGVs industriais eles ganham ainda mais importância, pois as operações com AGVs implicam a deslocação de materiais pesados, tornando crítica a questão da segurança de bens e pessoas. Desta

forma, em aplicações industriais, os robôs tipicamente não se podem deslocar se não tiverem disponível uma estimativa precisa da sua pose. As trajetórias são determinísticas e não são toleráveis grandes desvios em relação a estas. Por outro lado, os robôs de serviços normalmente têm trajetórias dinâmicas em função dos obstáculos detectados, o que torna mais aceitável a sua deslocação mesmo quando existe uma grande incerteza em relação à pose do robô.

1.3 Contributos científicos

Deste trabalho resultaram contribuições ao nível da localização *indoor* de robôs móveis. Em traços gerais efectuaram-se melhorias a técnicas que permitiram um aumento significativo da sua precisão e robustez de forma a permitir a sua aplicação em novos e mais exigentes ambientes, como é o caso das aplicações industriais. Foi abordado o problema da localização baseada em reflectores e em contornos naturais, e desta abordagem resultaram diversas contribuições, na forma de novos protótipos, artigos científicos e novos algoritmos.

De forma mais concreta as mais relevantes contribuições deste trabalho foram:

- O desenvolvimento de um sistema de localização baseado na detecção de reflectores por um laser de segurança. Este sistema resultou da combinação de um algoritmo de fusão sensorial com um esquema de filtragem de *outliers*. Com esta combinação obteve-se um algoritmo mais robusto que permite localizar um robô com menos informação sensorial e de menor qualidade: os sistemas comerciais de triangulação *laser* necessitam de pelo menos três reflectores em linha de vista, enquanto que, com apenas dois reflectores, o sistema desenvolvido apresentou erros de 5mm em termos de posição e 0.1° em termos de orientação, demonstrando uma precisão ao nível dos sistemas comerciais demonstrando uma precisão ao nível dos sistemas comerciais e com menos reflectores. No contexto destes desenvolvimentos foi publicado um artigo de conferência com o título "Robust Mobile Robot Localization Based on Security Laser Scanner" e um artigo de revista com o título "Mobile Robot Localization Based on a Security Laser: An Industry Scene Implementation". Além disso também foi desenvolvido um protótipo de um AGV industrial. Os desenvolvimentos neste âmbito foram também os resultados mais relevantes do projecto "PRODUTECH PTI PPS4.3 - Robótica Móvel".

- Uma análise dos algoritmos de *Map Matching* mais relevantes do estado da arte. Desta análise resultou o artigo em conferência "2D Cloud Template Matching - A Comparison between Iterative Closest Point and Perfect Match" e o artigo de revista "Map-Matching Algorithms for Robot Self-Localization: A comparison between Perfect Match, Iterative Closest Point and Normal Distributions Transform" este último já aceite para publicação e em fase de revisões finais. Este estudo realçou as vantagens em termos de eficiência computacional e robustez a erros de inicialização do PM face aos algoritmos disponiveis no PCL, um projecto *open-source* de referência.

- O desenvolvimento de um sistema de localização baseado na detecção de contornos naturais. Tendo em conta os resultados do estudo referido foram introduzidos um conjunto de melhorias no PM que aumentaram a sua precisão e robustez. Também foi desenvolvido um protótipo de um AGV industrial com o qual os desenvolvimentos propostos foram extensivamente testados e obtiveram-se resultados promissores quanto à sua aplicação em ambiente industrial. Nas experiências efectuadas com dados reais obtiveram-se erros médios de posição e orientação de 13mm e de 0,19°, respectivamente enquanto que na implementação original do PM registou-se um erro médio de 49mm e 0,32° em posição e orientação, respectivamente. Destes trabalhos também resultou um artigo em conferência "Robust robot localization based on the perfect match algorithm" e foi um dos resultados do projecto "PRODUTECH PTI PPS4.3 - Robótica Móvel".

- O desenvolvimento de novas abordagens ao conceito de detecção de falha no algoritmo de localização e respetivos mecanismos de recuperação. Neste sentido foi desenvolvido um algoritmo de localização global que permite detectar a falha e reinicializar o sistema de localização. Este sistema foi testado num robô de vigilância e é usado actualmente pela equipa de futebol robótico 5DPO. Os desenvolvimentos neste contexto também deram origem aos artigos de conferência e revista com os títulos: "Global Localisation Algorithm from a Multiple Hypotheses Set" e "Self-Localisation of Indoor Mobile Robots using Multi-hypotheses and a Matching Algorithm".

1.4 Organização do documento

O presente trabalho começa por expôr, em seguida, o estado da arte na localização de robôs móveis (Capítulo 2). No capítulo seguinte, descrevem-se as abordagens propostas no âmbito deste trabalho, começando por se detalhar as situações em análise (Secção 3.1) e depois os desenvolvimentos no âmbito da localização baseada em reflectores (Secção 3.2.1) e contornos (Secção 3.2.2). O Capitulo 4 é dedicado à descrição das experiências efectuadas e resultados obtidos com as metodologias propostas. Termina-se com o Capitulo 5, referente às conclusões e propostas para trabalho futuro.

Capítulo 2

Estado da arte

Há mais de vinte anos que uma vasta comunidade científica está dedicada ao problema da localização de robôs móveis. O estado da arte nesta área é vasto e apresenta inúmeras propostas de soluções baseadas em diferentes classes de algoritmos e diferentes tipos de sensores. No entanto, quando se olha para as ofertas comerciais, o número de soluções com sucesso é reduzido e aplicado em ambientes muito controlados. Neste trabalho, analisam-se os motivos desta discrepância e, em colaboração com parceiros industriais, procura-se identificar pontos de melhoria que viabilizem a aplicação das técnicas mais relevantes do estado da arte, em ambientes com elevados requisitos de fiabilidade e robustez, como é o caso dos ambientes industriais.

Sabattini et al. (2013) lista as fraquezas dos actuais sistemas logísticos que utilizam AGVs e apresenta algumas propostas de melhoria. Entre elas, está a melhoria dos sistemas de localização dos robôs, de forma a tornar possível a localização baseada em contornos. Nesta linha de pensamento encontram-se os trabalhos (Reinke e Beinschob, 2013) e (Beinschob e Reinke, 2013), nos quais é discutida a aplicação da localização baseada em contornos e extracção de informação tridimensional em ambiente industrial. É de realçar que estes artigos são resultado do projecto europeu PAN ROBOTS.

Contudo, a precisão e robustez dos algoritmos que exploraram a detecção de contornos naturais dependem da dinâmica e da estrutura do ambiente em que o robô navega, nomeadamente de um certo nível de invariância da disposição dos objectos circundantes ao robô que não é possível garantir em todas as situações. Os elevados requisitos de precisão e robustez tipicamente exigidos às aplicações industriais levam geralmente à adopção de soluções que recorrem à detecção de balizas artificiais (reflectores). Desta forma, garante-se a existência de um conjunto de características no ambiente, utilizadas pelo sistema de localização, com uma configuração adequada ao nível de precisão pretendido.

2.1 Localização baseada em reflectores

A localização baseada em reflectores pertence à família de técnicas que recorrem a balizas artificiais, isto é, implicam a instalação de equipamento externo ao robô para fins de localização.

O estado da arte apresenta várias alternativas aos reflectores das quais se apresentam alguns exemplos. Tem-se o caso dos trabalhos (Lee e Yu, 2014), (Zhang et al., 2012) e (Loevsky e Shimshoni, 2010) que recorrem à detecção de balizas visuais a partir de algoritmos de processamento de imagem, no entanto este tipo de técnicas é susceptíveis a interferências externas originadas por variações de iluminação. Outro exemplo são as abordagens que recorrem a sinais de rádio, têm-se os trabalhos (Wang e Yang, 2011) e (Han et al., 2013), que exploram a utilização dos sinais de comunicação *wireless* e o trabalho (Zhou e Shi, 2009), que apresenta uma revisão dos trabalhos que recorrem à tecnologia RFID (*Radio-Frequency IDentification*). Contudo, a aplicação destas tecnologias tende a ter alguns problemas de precisão devido a interferências e reflexões, especialmente em meios industriais, devido à abundância de estruturas metálicas. Nos trabalhos (Pierlot e Van Droogenbroeck, 2014a) e (Pierlot e Van Droogenbroeck, 2014b) recorre-se a balizas emissoras de sinais infravermelhos e é proposto um algoritmo de triangulação de três balizas. No entanto, a utilização de balizas activas (aquelas que necessitam de energia) pode ser inviável devido a custos de instalação e manutenção, principalmente no caso em que a área de navegação do robô é elevada. A utilização de reflectores detectados por um *laser scanner* montado no topo do robô é das soluções com mais sucesso em ambiente industrial. Este tipo de soluções recorre tipicamente a algoritmos de triangulação que estimam a pose a partir dos ângulos entre os reflectores detectados e o robô. Este tipo de soluções oferece uma qualidade sensorial superior, na medida em que consegue adquirir medidas com uma precisão e frequência relativamente elevadas. É de realçar que a disposição dos reflectores na área de navegação é um factor crítico na precisão obtida por um sistema de triangulação *laser*. Gao et al. (2016) analisa esta questão e recorre a algoritmos genéticos que optimizam a distribuição dos reflectores que maximiza o ângulo observável entre estes.

Em relação a trabalhos relacionados com a localização baseada em reflectores são de particular pertinência aqueles que abordam o problema da localização com balizas indistinguíveis entre si. Neste âmbito, encontra-se o algoritmo apresentado por Thrun et al. (2005) intitulado "EKF Localization with Unknown Correspondences", que é baseado no filtro de kalman estendido. O ponto-chave desta família de algoritmos é o processo de associação, isto é, o estabelecimento de uma relação entre as balizas (reflectores) que estamos a observar, que são indistinguíveis entre si, e o mapa de balizas (lista de posições das balizas no mundo). A solução de associação descrita por Thrun et al. (2005) assenta na "*maximum likelihood data estimation*" que, em termos gerais, define que, para cada observação, o valor da função de densidade de probabilidade de ocorrer essa observação é calculado para todas as balizas do mapa. A observação é associada à baliza que maximiza esse valor. Finalmente, apresentamos o trabalho de Ronzoni et al. (2011), onde é apresentada uma solução para a localização global em sistemas que recorrem a reflectores. Mais concretamente, é proposto um algoritmo de associação baseado nas distâncias entre reflectores observados. Tendo em conta que essa grandeza é invariante em relação à pose do robô, o processo de associação não depende de nenhum conhecimento prévio da pose do robô.

2.2 Localização baseada em contornos

Em relação à localização baseada em contornos, Thrun et al. (2005) apresenta a aplicação de um filtro de partículas com algumas melhorias. Este algoritmo, com o nome de AMCL (*Augmented Monte Carlo Localization*) tem a característica de ter um número de partículas que varia em função da incerteza da pose; além disso, acrescenta uma estratégia de localização global, isto é, permite localizar o robô sem nenhuma estimativa anterior da localização. Existem também outras abordagens à problemática da localização global, como podemos verificar nos trabalhos apresentados nas seguintes referências: (Zhang et al., 2011), (Vahdat, 2007), (Thrun et al., 2000) e (Liu et al., 2007).

Uma ferramenta importante nesta área são os algoritmos de *Map Matching*. Estes algoritmos são muito utilizados na localização de robôs na qual um mapa local do ambiente é comparado com um mapa global guardado em memória. O mapa local, ou de contornos, refere-se ao conjunto de posições (relativas ao referencial do robô) detectadas pelos sensores a bordo do robô. O mapa global representa o conhecimento do meio envolvente ao robô e geralmente consiste num mapa de ocupação que descreve se determinada coordenada global está livre ou ocupada. Estabelecendo a correspondência entre os dois mapas, calcula-se a pose actual do robô em relação ao referencial do mapa global. No estado da arte existem diversas propostas deste tipo de algoritmos. A procura de correspondência consiste geralmente numa procura iterativa da transformação rígida (pose) que melhor alinha dois conjuntos de pontos. O algoritmo *Perfect Match*, apresentado por Lauer et al. (2006) é um algoritmo de *Matching* entre um conjunto de pontos e uma grelha de ocupação. Este foi adoptado por muitas das equipas que participam nos campeonatos de futebol robótico, sendo que o sucesso deste trabalho passa pelo facto de ser um algoritmo relativamente leve em termos computacionais o que, por sua vez, permite a utilização deste a uma frequência elevada. Existem outros trabalhos concorrentes a este último como, por exemplo, o algoritmo conhecido por *Iterative Closest Point* (ICP) (Besl e McKay, 1992), muito usado em *scan-matching*. Este algoritmo procura a transformação rígida que melhor alinha dois conjuntos de pontos. Lu e Milios (1997) apresentam uma aplicação do ICP com os dados obtidos de um *laser range finder*. Ainda na mesma linha do algoritmo anterior, temos o Iterative Closest Line (ICL) (Censi, 2008), em que é feito *matching* entre um conjunto de pontos e um conjunto de linhas. A grande desvantagem de todos estes algoritmos é o elevado peso computacional da procura de correspondência entre os dois conjuntos de pontos, pois essa procura tem de ser feita a cada iteração. O "Polar Scan Matching"(PSM) (Diosi e Kleeman, 2007) evita essa procura de correspondência tirando partido da natureza polar das medidas de um *laser scanner*. Por fim, refere-se o algoritmo Vasco, disponível na biblioteca *open-source* CARMEN, que consiste num conjunto de aproximações sucessivas, até que o passo de teste seja abaixo de um determinado valor.

Até aqui, os algoritmos de *matching* apresentados baseiam-se numa procura de um mínimo local partindo do princípio que a solução encontrada corresponde ao mínimo global. Contudo, este tipo de abordagens pode dar problemas se o erro de inicialização for elevado. Por isso, e contrariamente ao que foi apresentado até agora, referimos o trabalho de Olson (2009) em que é

feita uma procura exaustiva em todo espaço de soluções. O autor defende que com um conjunto de optimizações é possível utilizar este algoritmo em aplicações em tempo real, suportando isto com um conjunto de resultados comparativos com outros algoritmos no que toca a tempo de execução.

A localização baseada em contornos só é possível caso exista uma descrição do espaço ocupado passível de ser detectado pelo *laser range finder* do robô. Essa descrição pode ser obtida através de um algoritmo de SLAM (*Simultaneous Localization and Mapping*), sendo que através de demonstração o sistema gera automaticamente uma grelha de ocupação, posteriormente utilizada pelos sistemas de localização. Exemplos deste tipo de algoritmos são o Gmapping (Grisetti et al., 2007) e o Hector Slam (Kohlbrecher et al., 2011), ambos disponíveis no projecto *open-source* ROS (Quigley et al., 2009). O Gmapping recorre a uma família de filtro de partículas com o nome Rao-Blackwellized e depende de um sistema odométrico com alguma qualidade. Por outro lado, o Hector Slam é muito mais leve em termos computacionais e pode funcionar até mesmo sem odometria, mas está mais dependente da qualidade da informação sensorial fornecida pelo laser e do nível de estruturação do meio que está a ser mapeado. Por fim, referimos o trabalho publicado por Adams et al. (2014) que nos oferece uma revisão dos trabalhos recentes na área do SLAM.

O ponto de partida dos desenvolvimentos referentes à localização baseada em contornos foi o algoritmo "Perfect Match" apresentado por Lauer et al. (2006). De forma a justificar esta escolha, foi realizada uma comparação entre este último e outros dois algoritmos, o ICP (Besl e McKay, 1992) e o NDT (Biber e Strasser, 2003). Tendo isto em conta, apresenta-se com mais algum detalhe este conjunto de algoritmos.

2.2.1 Perfect Match (PM)

O Perfect Match é um algoritmo conhecido por ter um baixo peso computacional proposto por Lauer et al. (2006). Neste trabalho, apresenta-se uma solução para a localização de um robô utilizado nas competições da liga de Futebol Robótico Médio (*Middle Size League* – MSL). Aqui, a pose é estimada a partir do cálculo da correspondência entre os dados extraídos da imagem obtida por uma câmara omnidireccional instalada no robô, e o mapa das linhas colocadas no chão, dispostas de forma similar às de um campo de futebol. Em traços gerais, este algoritmo baseia-se na minimização de uma função custo que traduz o erro de correspondência entre os dados do sensor e o mapa do ambiente. Resumidamente, o algoritmo *Perfect Match* divide-se em três passos:

1. Cálculo de gradientes;
2. Rotina de optimização baseada no *Resilient Back-Propagation* (RPROP);
3. Estimação da co-variância a partir da segunda derivada.

É utilizada uma grelha de ocupação para descrever o meio envolvente do robô, ou seja, recorre-se a uma matriz em que cada célula indica se a área circundante a determinada posição absoluta

está livre ou ocupada. No caso particular do futebol robótico, cada célula corresponde a uma zona branca (referente a uma linha) ou a uma zona verde. A partir desta grelha de ocupação são pré-calculadas mais três matrizes: o mapa de distâncias e os correspondentes gradientes. No mapa de distâncias, cada célula indica a distância à célula ocupada mais próxima. Em relação aos gradientes, temos duas matrizes: uma referente à derivada do mapa de distâncias em relação ao eixo horizontal (eixo x) e outra referente ao eixo vertical (eixo y). Estes três mapas (mapa de distâncias, gradiente em relação a x e gradiente em relação a y) podem ser pré-calculados uma vez no início do programa de forma a aumentar a eficiência computacional do algoritmo em termos de tempo de CPU ocupado.

$$E = \sum_{i=1}^{PCount} E_i,\ E_i = \frac{L_c^2}{L_c^2 + d_i^2} \tag{2.1}$$

A Equação 2.1 apresenta a função de custo utilizada no PM. Esta consiste no somatório da contribuição no erro de *matching* de PCount medidas adquiridas pelos sensores do robô. Cada termo E_i é função do parâmetro L_c e da distância d_i, em que d_i é o valor da célula do mapa de distâncias correspondente à posição da medida i no referencial absoluto. De forma a aumentar a robustez a *outliers* recorre-se à função E_i em que L_c é um parâmetro que limita a contribuição de uma medida com um erro muito elevado. Para minimizar a função custo E, o algoritmo PM recorre ao método de optimização RPROP. Ou seja, a saída desde algoritmo será a pose que minimiza a função custo apresentada em 2.1. Posteriormente serão apresentados mais detalhes sobre este algoritmo.

2.2.2 Iterative Closest Point (ICP)

O algoritmo *Iterative Closest Point* (ICP) (Besl e McKay, 1992) é um algoritmo de registo que procura minimizar a distância euclidiana entre um conjunto de dados de entrada e um modelo de referência (no caso particular da localização, isto poderia corresponder aos dados de um *laser range finder* e ao mapa do ambiente).

Como já se referiu, existem diversas variantes deste algoritmo. Apresenta-se uma implementação disponível no projecto "*Point Cloud Library*"(PCL) (Rusu e Cousins, 2011), que foi utilizada neste trabalho. Do ponto de vista matemático consideram-se dois conjuntos de pontos: um conjunto A (com n elementos) referente aos dados do sensor e outro conjunto B (com m elementos) referente ao mapa de referência, ambos $\in R^2$. O objectivo é optimizar a função de transformação $u : A \rightarrow B$ que minimiza a média da distância quadrática (*MSD*) entre os conjuntos A e B (ver Equação 2.2).

$$MSD(A,B,u) = \frac{1}{n}\sum_{a\in A} \|a - u(a)\|^2 \tag{2.2}$$

Incorporando a matriz de rotação (R) e o vector translação (t) na Equação 2.2 e assumindo $u(a_i) = b_i$ onde $a_i \in A$ e $b_i \in B$, o problema de minimização pode ser reescrito da seguinte forma:

$$\min_{u:\ A \to B} \frac{1}{n} \sum_{i=1}^{n} \|Ra_i - t - b_i\|^2 \tag{2.3}$$

A partir desta formulação, o ICP minimiza iterativamente o $MSD(A,B,u)$. Cada iteração pode ser dividida em duas fases: a fase de procura de correspondências e a fase de optimização.

Durante a fase de procura de correspondências, o objectivo é minimizar o $MSD(A,B,u)$ encontrando a melhor correspondência directa entre um ponto $a_i \in A$ e um ponto $b_i \in B$. Este passo consiste na sua forma mais básica em seleccionar o ponto $b_i \in B$ mais próximo (menor distância euclidiana) do ponto $a_i \in A$ (diagrama de Voronoy).

Durante a fase de optimização, o objectivo é calcular a matriz de rotação R e o vector de translação t que minimizam a Equação 2.3 utilizando as correspondências calculadas na fase anterior. A implementação do ICP considerada recorre a um simples optimizador baseado em mínimos quadrados para encontrar a matriz de transformação linear óptima $(R|t)$ que minimiza a equação 2.3 (Besl e McKay, 1992). Neste sentido, o algoritmo começa por calcular os dois centróides de A e B e subtraí-los aos dois conjuntos de pontos como se apresenta nas Equações 2.4 e 2.5.

$$a'_i = a_i - \frac{1}{n} \sum_{i=0}^{n} a_i \tag{2.4}$$

$$b'_i = b_i - \frac{1}{n} \sum_{i=0}^{n} b_i \tag{2.5}$$

Este passo ajuda à simplificação do problema de minimização (Besl e McKay, 1992). De seguida, a matriz de "*cross-variance*" é calculada segundo a Equação 2.6 utilizando A' e B' sendo estes os conjuntos de pontos a'_i e b'_i respectivamente.

$$H = A'B'^T \tag{2.6}$$

Agora, o ângulo de rotação θ pode ser calculado a partir da equação 2.7:

$$\theta = atan2((H(0,1) - H(1,0)), (H(0,0) + H(1,1)) \tag{2.7}$$

A solução óptima para R e t que minimiza a função objectivo 2.3 é definida nas equações 2.8 e 2.9 em que $\bar{a}$ e $\bar{b}$ são os centróides calculados nas equações 2.4 e 2.5.

No fim é aplicada a transformação calculada ao conjunto A e volta-se a repetir o processo, executando uma nova iteração onde são calculadas as novas correspondências e sua respectiva transformação óptima até que um determinado critério de paragem seja atingido. Este critério pode ser por exemplo um número máximo de iterações ou a diferença de translação e rotação

entre duas iterações consecutivas.

$$R = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \tag{2.8}$$

$$t = \bar{b} - R\bar{a} \tag{2.9}$$

2.2.3 Normal Distributions Transform (NDT)

O algoritmo "*Normal Distributions Transform*"(NDT) foi inicialmente apresentado por Biber e Strasser (2003) como um método de *matching* 2D e posteriormente estendido para 3D nos trabalhos (Magnusson et al., 2007) e (Magnusson, 2009). Este algoritmo de *matching* modela o ambiente através de um conjunto de funções de densidade de probabilidade (PDFs). Esta representação é gerada a partir dos pontos de referência (mapa), na qual são feitos conjuntos tendo em conta uma grelha que divide o espaço em células de tamanho fixo (*voxel grid*). Para cada célula da *voxel grid* que tenha pelo menos seis pontos é calculada a média (q) e a matriz de covariância (Σ) recorrendo às equações 2.10:

$$q = \frac{1}{n}\sum_{i=1}^{n} k_i, \quad \Sigma = \frac{1}{n}\sum_{i=1}^{n}(k_i - q)(k_i - q)^T \tag{2.10}$$

Em que $k_{i=1..n}$ representa o conjunto de pontos contidos numa determinada célula da *voxel grid*.

Após inicializar esta representação 3D do ambiente, a probabilidade da ocorrência de uma medida na região delimitada por uma célula da *voxel grid* é dada por:

$$p(k) \sim -d_1 e^{-\frac{d_2(k-q)^T \Sigma^{-1}(k-q)}{2}} \tag{2.11}$$

Em que d_1 e d_2 são constantes para limitar o efeito de possíveis *outliers* que possam afectar as medidas dos sensores (Magnusson, 2009), enquanto q e Σ são respectivamente a média e a covariância dos pontos de referência contidos no interior da *voxel grid*.

Para utilizar a abordagem NDT para o seguimento de pose 3D são definidos os parâmetros a optimizar: $\overrightarrow{w} = \{tx, ty, tz, \phi_x, \phi_y, \phi_z\}$. A função de transformação entre duas coordenadas do referencial do robô é representada por $\{tx, ty, tz\}$ para a translação e pelos ângulos de euler z-y-x $\{\phi_x, \phi_y, \phi_z\}$ para a rotação, que pode ser representada por:

$$T(\overrightarrow{w}, \overrightarrow{k}) = R_x R_y R_z \overrightarrow{k} + \overrightarrow{t} \tag{2.12}$$

Em que t representa os parâmetros $\{tx, ty, tz\}$ de translação e $R_x R_y R_z$ são as matrizes de rotação correspondentes aos ângulos de euler $\{\phi_x, \phi_y, \phi_z\}$. O objectivo do registo 3D é estimar estes seis parâmetros a partir dos dados dos sensores e de uma representação 3D do ambiente pré-processada.

Esta estimação é baseada numa optimização que recorre ao método de Newton (More et al. (1992)) na qual é minimizada a seguinte função de custo:

$$score(p(k')) = \sum_{i=1}^{n} -d_1 e^{-\frac{d_2 (k'_i - q_i)^T \Sigma_i^{-1} (k'_i - q_i)}{2}} \tag{2.13}$$

O custo (*score*) consiste no somatório dos valores negativos das distribuições normais associadas aos pontos k'_i. O método de Newton calcula as correcções necessárias na pose do robô (Δk) para minimizar o *score* a partir do gradiente g e da matriz Hessiana H. Estas correcções são calculadas em função da expressão 2.14.

$$H\Delta k = -g \tag{2.14}$$

As aplicações industriais apresentam tipicamente requisitos muito exigentes em termos de repetibilidade e robustez para os sistemas de navegação que não formam ainda muito explorados pela comunidade científica, e as soluções comerciais concentram-se na aplicação de sistemas baseados em triangulação laser instalados a uma altura elevada em relação ao chão para diminuir o número de oclusões e *outliers*. Neste trabalho propõe-se um conjunto de melhorias em termos de fusão sensorial e filtragem de *outliers* de forma a aumentar a precisão e robustez de técnicas existentes com o objectivo de viabilizar a sua aplicação em robôs móveis industriais. Apresenta-se um novo sistema baseado na detecção de reflectores instalados junto ao chão por um laser de segurança e propõe-se um conjunto de melhorias ao algoritmo de *map-matching Perfect Match* de forma a desenvolver um sistema de localização baseado na detecção de contornos naturais para ser utilizado numa aplicação industrial.

Outro assunto ainda pouco abordado nesta área é a recuperação de falhas dos sistemas de localização, especialmente no caso dos sistemas baseados na detecção de contornos naturais. Neste sentido é proposta uma nova abordagem de detecção de falhas do sistema de localização e subsequente processo de localização global, isto é, localizar o robô sem nenhuma estimativa da sua pose.

Capítulo 3

Abordagem e soluções propostas

Este capítulo é dedicado à apresentação dos desenvolvimentos efectuados na localização baseada em reflectores e na localização baseada em contornos. Para cada uma das situações em análise o problema é formulado e as assunções tidas em conta são listadas (Secção 3.1). De seguida são apresentados os algoritmos e implementações desenvolvidas (Secção 3.2).

3.1 Descrição das situações em análise

3.1.1 Localização baseada em reflectores

Nesta secção formaliza-se o problema da localização baseada em reflectores. De forma mais genérica, estamos a definir o problema de localizar um robô móvel através da detecção de balizas artificiais. A designação de "balizas" tem a ver com o facto de estes reflectores não serem detectados pontualmente, como é o caso dos marcadores, mas sim ao longo de uma porção da trajectória. Chamam-se artificiais porque são instalados propositadamente no meio para fins de localização. Geralmente basta detectar um marcador para ser possível calcular a pose de um robô, mas essa detecção só costuma ser possível quando o marcador se encontra a uma curta distância, obrigando a que os percursos do robô passem por marcadores. Pelo contrário, as balizas são detectáveis a uma distância superior e geralmente é preciso detectar mais que uma baliza para estimar a pose de um robô. Em Esteves (2005) são apresentados os diferentes tipos de métodos para localização de robôs móveis, nomeadamente a diferenciação entre métodos que recorrem a marcos e balizas.

O problema da localização baseada em reflectores consiste em estimar a pose absoluta do robô $X_v = [x_v \; y_v \; \theta_v]^T$ num referencial fixo e externo ao robô $W_x \; W_y$ (Figura 3.1), em que $W_x \; W_y$ é o referencial do mundo e $R_x \; R_y$ é o referencial relativo ao robô.

Esta estimação é realizada a partir:

- Do mapa de reflectores $M_B = [M_{B,1} \; ... \; M_{B,numB}]$. Este consiste num conjunto de $numB$ posições fixas nas quais os reflectores foram instalados. Ou seja, $M_{B,i} = [x_{B,i} \; y_{B,i}]$ corresponde à posição do reflector i em relação ao referencial externo $W_x \; W_y$. Os reflectores têm uma

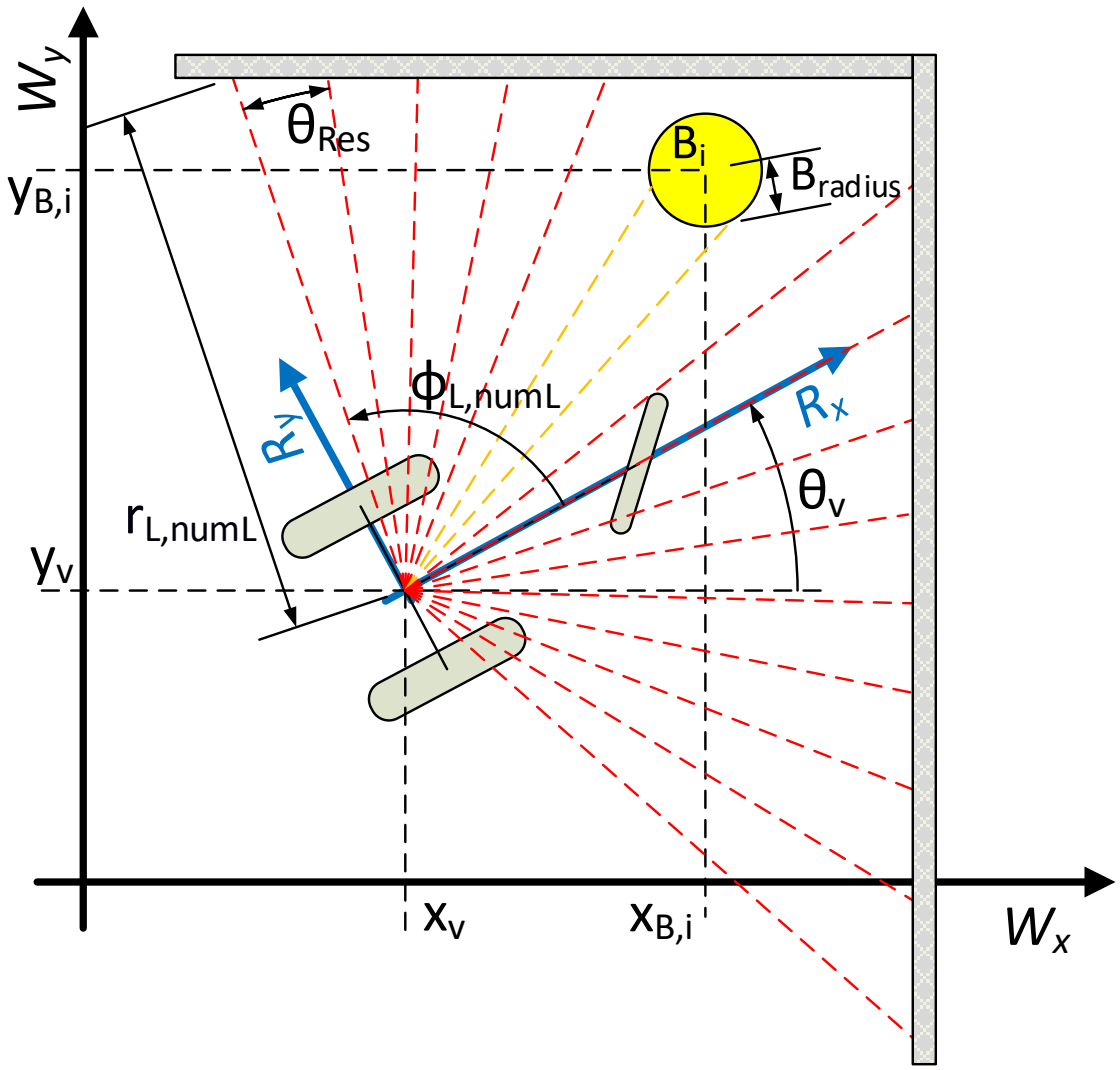


Figura 3.1: Representação do referencial relativo (R_x R_y) do robô na pose $X_v = [x_v\ y_v\ \theta_v]^T$ relativamente ao referencial do mundo W_x W_y. O círculo amarelo representa um reflector (elemento do mapa de reflectores M_B) na posição fixa $[x_{B,i}\ y_{B,i}]$. As linhas tracejadas representam um conjunto de medidas obtidas com um *laser scanner*. *numL* corresponde à ultima medida, com as coordenadas polares $Z_{L,numL} = [r_{L,numL} \quad \phi_{L,numL}]$. Medidas com alta reflectividade e baixa reflectividade são diferenciadas na imagem através das cores laranja e vermelha respectivamente.

forma cilíndrica, com um raio fixo e conhecido (B_{radius}) e são indistinguíveis entre si. Na Figura 3.1 temos a representação de um reflector B_i através de um círculo amarelo;

- Das medidas (observações) obtidas através do *laser scanner* $Z_L(k) = \{Z_{L,i}(k),\ C_{L,i}(k) : i \in 1, \ldots, numL\}$ (*numL* corresponde ao número de medidas adquiridas numa amostragem do *laser*); $Z_{L,i}$ corresponde à posição do obstáculo, detectado na medida i, no instante k, em relação ao referencial relativo do robô (R_x R_y). Esta posição é expressa em coordenadas polares (compostas pela distância $r_{L,i}$ e o ângulo $\phi_{L,i}$). $c_{L,i}$ é uma variável booleana que

indica se o objecto detectado tem ou não uma grande reflectividade. As linhas tracejadas na Figura 3.1 representam as medidas do *laser*. As linhas laranja cujos feixes intersectaram o reflector correspondem ao $c_{L,i}$ com um valor booleano igual a verdadeiro; por outro lado, as medidas a vermelho representam medidas correspondentes ao $c_{L,i}$ com um valor booleano igual a falso;

- Dos dados de deslocamento relativo fornecidos pelos *encoders* do robô $u(k) = [\Delta x_{odos}(k)\ \Delta y_{odos}(k)\ \Delta\theta_{odos}(k)$ $U(k)$ representa a variação da posição $\Delta x_{odos}(k)$ e $\Delta y_{odos}(k)$ e orientação $\Delta\theta_{odos}(k)$ do robô em relação ao referencial relativo (R_x R_y) no intervalo de tempo k considerado.

No problema aqui apresentado têm-se em conta as seguintes assunções:

- Existe uma estimação aproximada da pose do robô. Está-se, portanto, a definir um problema de *tracking*. Pretende-se no futuro desenvolver soluções que inicializem e supervisionem o resultado do sistema aqui desenvolvido.
- Os reflectores têm todos o mesmo tamanho e em posições conhecidas caracterizadas por M_B. A aquisição de M_B consiste num problema de mapeamento e não será abordado neste trabalho, mas existem diversas soluções no estado da arte, como é o caso de Blanco et al. (2012).
- O *laser scanner* distingue entre objectos de alta e baixa reflectividade. Os reflectores têm alta reflectividade, mas não são os únicos objectos com essa característica. Portanto, a possibilidade de detectar falsos positivos é elevada.
- Apenas um pequeno subconjunto dos reflectores instalados estará dentro do campo de visão (FOV) do *laser scanner*.

Como já foi referido no Capítulo 1 está-se a analisar o caso da utilização de um *laser* de segurança para localização e, sendo a aquisição de dados feita a uma altura baixa em relação ao chão a quantidade e tipo de objectos presentes são maiores. Isto leva a que a qualidade dos dados sensoriais, obtidos por um *laser* de segurança, seja inferior àquela que se obtém através dos tradicionais *lasers* de navegação. Contudo, apesar disto, pretende-se desenvolver uma solução com precisão e robustez suficiente para integrar um AGV industrial.

3.1.2 Localização baseada em contornos

Nesta secção formaliza-se o problema da localização baseada em contornos. Esta insere-se na classe de soluções que não requerem a preparação do ambiente (*enviroment base navigation*), isto é, que não implicam a instalação de equipamento específico na área de navegação do robô para fins de localização (por exemplo, reflectores).

Ao contrário da localização baseada em balizas artificiais, a localização com base em contornos evita custos com a instalação de reflectores, ou outros marcadores ou balizas artificiais, e permite uma menor interferência com a área circundante ao robô. Contudo, os desafios inerentes

a este tipo de soluções são maiores, pois no caso da localização baseada em contornos, abdica-se da medida da reflectividade dos objectos, sendo considerada apenas a totalidade das medidas de distância fornecida pelo *laser scanner* (contornos). Tal conduz a uma maior quantidade de dados e, por consequente, *outliers*.

A robustez e a precisão dos algoritmos que exploram as características naturais da área de navegação estão dependentes de geometrias na distribuição dos objectos que não se alterem significativamente ao longo do tempo, sendo suficientemente estruturadas e distinguíveis, de modo a que seja possível extrair uma pose absoluta fiável do robô.

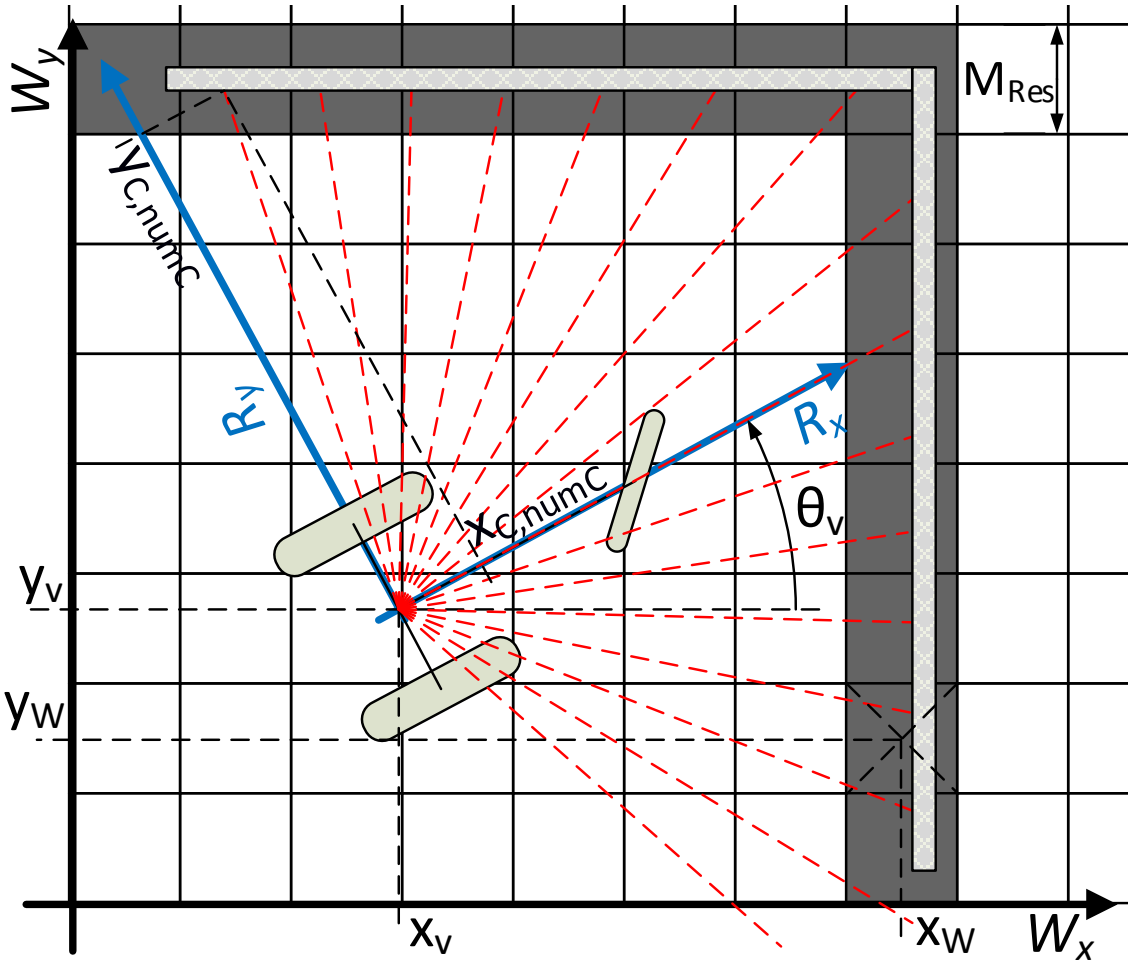


Figura 3.2: Representação do referencial relativo (R_x R_y) do robô na pose $X_v = [x_v\ y_v\ \theta_v]^T$ relativamente ao referencial do mundo (W_x W_y). As linhas tracejadas representam um conjunto de medidas obtidas com um *laser scanner*, em que *numC* corresponde à ultima medida, com as coordenadas cartesianas $Z_{C,numC} = [x_{C,numC}\ \ y_{C,numC}]$. A grelha representa o mapa de ocupação M_C em que cada célula tem uma dimensão M_{Res}. Cada célula de M_C pode ter o valor de ocupado (cinzento escuro) ou livre (branco).

O problema da localização baseada em contornos consiste em estimar a pose absoluta do robô $X_v = [x_v\ y_v\ \theta_v]^T$ num referencial fixo e externo ao robô W_xW_y (Figura 3.2), em que W_xW_y é o referencial do mundo e R_xR_y é o referencial relativo ao robô.

Esta estimativa é realizada a partir dos seguintes dados:

- Do mapa de ocupação M_C, que traduz o conhecimento que o robô tem do espaço envolvente. Este consiste numa matriz bidimensional em que cada célula indica se a posição correspondente no mundo (referencial W_x, W_y) está livre ou ocupada. Associado a M_C temos também uma resolução (M_{Res}) que indica a largura de cada célula, e um *offset* que define a pose do referencial absoluto $W_x W_y$ em relação a M_C. De forma a simplificar a apresentação do trabalho, assume-se que o referencial está localizado no canto inferior esquerdo de M_C como se pode observar na figura 3.2.

- Dos contornos extraídos do *laser scanner* $Z_C(k) = \{x_{C,i}(k), y_{C,i}(k) : i \in 1, \ldots, numC\}$. Em que $numC$ corresponde ao número de medidas adquiridas numa amostragem do sensor, e o par $x_{C,i}$ e $y_{C,i}$ diz respeito a uma posição relativa (referencial R_x, R_y) correspondente a um local no meio ocupado. Z_C é equivalente a Z_L da secção anterior, embora não considerando a reflectividade do objecto detectado ($C_{L,i}(k)$). Aqui é usada uma representação cartesiana (e não a polar, como anteriormente). Numa situação ideal (sem erros), e conhecendo a transformação entre o referencial do mundo e o referencial do robô, os pontos detectados pelos sensores do robô devem todos corresponder a células ocupadas no mapa de ocupação (Figura 3.2).

- Dos dados dos *encoders* $u(k) = [\Delta x_{odos}\ \Delta y_{odos}\ \Delta\theta_{odos}]$ já descritos na secção anterior.

Neste problema têm-se em conta as seguintes assunções:

- É conhecida a pose aproximada do robô, dado que aqui estamos a considerar um algoritmo de pose *tracking*. Contudo vamos também abordar a questão da localização global, isto é um sistema de localização que considera a não existência de um conhecimento prévio da pose do robô tendo a capacidade de se inicializar automaticamente.

- Assume-se a existência de M_C e que este não varia ao longo do tempo. O problema da aquisição de M_C não será considerado neste trabalho, sendo este um problema de mapeamento geralmente resolvido por algoritmos de SLAM (*Simultaneous Localization and Mapping*). Algumas referências na área que foram utilizadas para a obtenção dos mapas utilizados são (Grisetti et al., 2007) e (Kohlbrecher et al., 2011).

Devido à elevada precisão e robustez tipicamente exigidas nas aplicações industriais, e considerando as dificuldades associadas à utilização de contornos, é comum a utilização de lasers de navegação. Com isto, aumenta-se a qualidade sensorial da informação utilizada no sistema de localização, uma vez que este laser permite uma maior abertura angular (FOV) e encontra-se mais elevado em relação ao chão. Esta altura tem grande impacto na informação acerca da dinâmica do ambiente em que o robô se encontra. A Figura 3.3 apresenta o resultado de dois mapeamentos duma mesma zona, realizados com dois sensores diferentes: a da esquerda com um laser de navegação montado a cerca de 1.9m de altura e a da direita com um laser de segurança montado

Figura 3.3: Comparação do resultado do mapeamento utilizando um laser de navegação a 1,9 metros de altura (esquerda) e um laser de segurança a 0,1 metros de altura (direita).

a cerca de 0.1m de altura. É notória a menor estruturação e a presença de objectos que não são fixos (como por exemplo cadeiras, caixas, etc) no mapa gerado a partir do laser de segurança, o que torna mais difícil a utilização deste sensor em localização a partir de contornos.

3.2 Algoritmos e implementações desenvolvidas

3.2.1 Localização baseada em reflectores

Nesta secção apresenta-se a proposta de solução para um sistema de localização baseado na detecção de reflectores a partir de um laser de segurança. Começa-se por apresentar uma visão geral da arquitectura do sistema, seguida de uma descrição detalhada de cada componente.

A Figura 3.4 ilustra a arquitectura do sistema. Cada bloco preto representa um componente do sistema com as suas respectivas entradas e saídas. Os blocos vermelhos e azuis representam, respectivamente, os dados de entrada e saída de cada iteração do algoritmo enquanto os blocos verdes correspondem aos parâmetros. A solução proposta é baseada na aplicação de um algoritmo de fusão sensorial, conhecido como *Extended Kalman Filter* (EKF), representado na Figura 3.4 a cinzento. A aplicação do EKF é combinada com outros dois componentes: o *Reflector Detector* e o *Association/Outlier Filter*. Estes componentes pré-processam os dados usados no filtro e procuram detectar *outliers* nos dados sensoriais, de forma a rejeitar falsas detecções de reflectores e aumentar a robustez do sistema de localização. Além disto, desenvolveu-se um componente de supervisão (*Uncertainty Supervisor*) para avaliar a incerteza da localização estimada, de forma a detectar situações em que a qualidade da informação sensorial não é suficiente para assegurar uma navegação segura. Embora o EKF seja bem conhecido e já muito estudado, a sua robustez e correcto funcionamento dependem muito de uma correcta e cuidada implementação de filtros (como o *Reflector Detector* e *Association/Outlier Filter*). Com estes filtros procura-se eliminar perturbações nas medidas que não são modeladas pelo EKF.

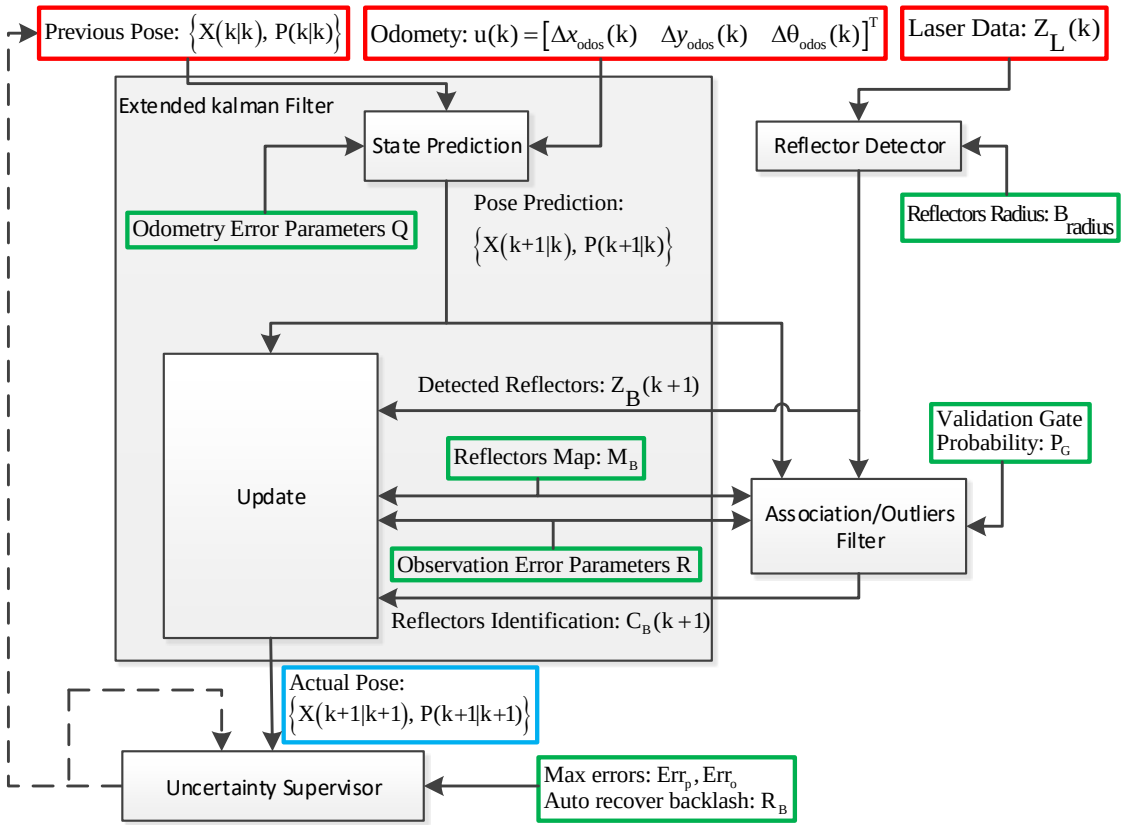


Figura 3.4: Arquitectura do sistema. Cada bloco preto representa um componente do sistema com as suas respectivas entradas e saídas. Os blocos vermelhos, azuis e verdes representam respectivamente as entradas, saídas e parâmetros do sistema. A solução proposta consiste na aplicação de um *Extended Kalman Filter* (área a cinzento) combinado com outros dois métodos: *Reflector Detector* e *Association/Outliers Filter*. Desenvolveu-se também um módulo para avaliar a incerteza da estimativa da localização designado por *Uncertainty Supervisor*.

3.2.1.1 Extended Kalman Filter

O EKF é um algoritmo de fusão sensorial muito utilizado em robótica móvel, inclusive na resolução do problema da localização. Foi considerado neste problema por causa das seguintes vantagens:

- Fusão sensorial: combinação da informação da distâncias e ângulos aos reflectores detectados com os dados da odometria;
- Fornecimento de uma descrição estatista do conjunto de variáveis que está a estimar (estado), através de uma distribuição gaussiana multivariável, parametrizada por um vector de média e respectiva matriz de covariância;
- Consideração de uma descrição estatística dos erros que afectam a informação fornecida pelos sensores. Este erro é modelado por ruído gaussiano de média nula;

- Eficiência computacional: Thrun et al. (2005) analisa esta questão e refere que a complexidade do filtro de Kalman é pelo menos $O(d^{2.4})$ em que d é dimensão do vector das observações. Neste caso em concreto, são processadas sequencialmente observações de dimensão dois, o que torna a aplicação do filtro de Kalman uma solução eficiente em termos computacionais.

Contudo, o EKF apresenta as seguintes desvantagens:

- Está limitado a distribuições gaussianas. Esta descrição pode não ser suficiente para modelar propriedades importantes dos erros que afectam as medidas dos sensores do robô, como por exemplo os erros sistemáticos. Sendo também a distribuição gaussiana unimodal, com um único máximo, este algoritmo não se adequa à resolução do problema da localização global (cálculo da pose do robô sem a necessidade de um valor aproximado da sua pose actual), onde tipicamente é necessário considerar mais de que uma solução para a pose do robô;

- Erros de linearização: como este algoritmo é baseado em linearizações de funções não lineares, os erros de linearização podem ser significativos se os erros das variáveis aleatórias em jogo forem elevados. Estes erros de linearização levam a uma degradação das estimativas, que no limite, pode fazer com que a estimativa do filtro divirja do valor real.

No caso em particular das aplicações industriais, as desvantagens relacionadas com linearizações perdem alguma relevância no sentido em que, por questões de segurança, os erros de estimativa da posição e orientação não devem ser muito elevados (baixa covariância). Além disto, aplicações industriais tipicamente exigem soluções com uma taxa de amostragem alta sendo os filtros de Kalman mais adequados do que outras abordagens, como por exemplo os filtros de partículas (Grisetti et al., 2007).

De forma a utilizar o EKF, é necessário definir o modelo de transição de estado e o modelo de observação. Relativamente ao modelo de transição de estado considera-se a proposta apresentada por Eliazar e Parr (2004).

$$X_v(k+1) = \begin{bmatrix} x_v(k+1) \\ y_v(k+1) \\ \theta_v(k+1) \end{bmatrix} = f(X_v(k), u(k)) + N(0, Q(k)) \tag{3.1}$$

A equação 3.1 modela a evolução da pose do robô. Esta é composta pela soma de dois componentes: a função $f(.)$ não linear, de transição de estado e $N(.)$, que diz respeito ao ruído de estado e corresponde a uma distribuição gaussiana de média nula e covariância $Q(.)$.

A função $f(.)$ tem como entradas a pose actual $X_v(k)$ e os dados da odometria representados pelo vector $u(k) = [\Delta x_{odos}(k)\ \Delta y_{odos}(k)\ \Delta\theta_{odos}(k)]$ em que cada componente corresponde à variação da pose em relação ao referencial relativo do robô. Utilizando-se aproximação discreta das

diferenças centradas, obtém-se a definição de $f(.)$ enunciada na Equação 3.2.

$$f(X_v(k),u(k)) = \begin{bmatrix} x_v(k) + \Delta x_{odos}(k)\cos\left(\theta_v(k) + \frac{\Delta\theta_{odos}(k)}{2}\right) - \Delta y_{odos}(k)\sin\left(\theta_v(k) + \frac{\Delta\theta_{odos}(k)}{2}\right) \\ y_v(k) + \Delta x_{odos}(k)\sin\left(\theta_v(k) + \frac{\Delta\theta_{odos}(k)}{2}\right) + \Delta y_{odos}(k)\cos\left(\theta_v(k) + \frac{\Delta\theta_{odos}(k)}{2}\right) \\ \theta_v(k) + \Delta\theta_{odos}(k) \end{bmatrix} \tag{3.2}$$

Para utilizar o EKF é necessário linearizar o modelo de transição de estado. Neste sentido, a partir das equações 3.3 e 3.4 apresenta-se a definição do jacobiano de $f(.)$ em relação a $X_v(k)$:

$$\nabla f_X(X_v(k),u(k)) = \begin{bmatrix} \frac{\partial x_v(k+1)}{\partial x_v(k)} & \frac{\partial x_v(k+1)}{\partial y_v(k)} & \frac{\partial x_v(k+1)}{\partial \theta_v(k)} \\ \frac{\partial y_v(k+1)}{\partial x_v(k)} & \frac{\partial y_v(k+1)}{\partial y_v(k)} & \frac{\partial y_v(k+1)}{\partial \theta_v(k)} \\ \frac{\partial \theta_v(k+1)}{\partial x_v(k)} & \frac{\partial \theta_v(k+1)}{\partial y_v(k)} & \frac{\partial \theta_v(k+1)}{\partial \theta_v(k)} \end{bmatrix} \tag{3.3}$$

$$\nabla f_X(X_v(k),u(k)) = \begin{bmatrix} 1 & 0 & -\Delta x_{odos}(k)\sin\left(\theta_v(k) + \frac{\Delta\theta_{odos}(k)}{2}\right) - \Delta y_{odos}(k)\cos\left(\theta_v(k) + \frac{\Delta\theta_{odos}(k)}{2}\right) \\ 0 & 1 & \Delta x_{odos}(k)\cos\left(\theta_v(k) + \frac{\Delta\theta_{odos}(k)}{2}\right) - \Delta y_{odos}(k)\sin\left(\theta_v(k) + \frac{\Delta\theta_{odos}(k)}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$Q(.)$ representa a matriz de covariância do ruído de odometria e depende da quantidade de movimento $u(k)$ e da pose actual; é definida pela equação 3.5.

$$Q(k) = \nabla f_u(X_v(k),u(k)) * \begin{bmatrix} (\sigma_{\Delta x_{odos}})^2 & 0 & 0 \\ 0 & (\sigma_{\Delta y_{odos}})^2 & 0 \\ 0 & 0 & (\sigma_{\Delta\theta_{odos}})^2 \end{bmatrix} * \nabla f_u{}^T(X_v(k),u(k)) \tag{3.5}$$

Na equação 3.5 $\nabla f_u(.)$ é o Jacobiano de $f(.)$ em relação a $u(k)$ e a matriz diagonal representa a matriz de covariância dos dados de odometria. Ou, seja assume-se que $\Delta x_{odos}(k)$, $\Delta y_{odos}(k)$ e $\Delta\theta_{odos}(k)$ são variáveis aleatórias gaussianas, independentes entre si e caracterizadas respectivamente pelas variâncias: $(\sigma_{\Delta x_{odos}})^2$, $(\sigma_{\Delta y_{odos}})^2$ e $(\sigma_{\Delta\theta_{odos}})^2$. Eliazar e Parr (2004) propõe um modelo no qual o desvio padrão do erro das medidas obtidas a partir da odometria é proporcional ao valor absoluto do movimento de translação e de rotação (Equações 3.6 e 3.7).

$$\sigma_{\Delta x_{odos}} = \sigma_{\Delta y_{odos}} = \sigma_{\min,\Delta d} + \alpha_1\sqrt{(\Delta x_{odos})^2 + (\Delta y_{odos})^2} + \alpha_2|\Delta\theta_{odos}(k)| \tag{3.6}$$

$$\sigma_{\Delta\theta_{odos}} = \sigma_{\min,\Delta\theta} + \alpha_3\sqrt{(\Delta x_{odos})^2 + (\Delta y_{odos})^2} + \alpha_4|\Delta\theta_{odos}(k)| \tag{3.7}$$

Este modelo tem um conjunto de parâmetros: $\{\sigma_{\min,\Delta d}, \sigma_{\min,\Delta\theta}, \alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ em que:

- $\sigma_{\min,\Delta d}$ e $\sigma_{\min,\Delta\theta}$: constituem uma quantidade mínima de incerteza de posição e orientação, respectivamente que é adicionada ao estado do robô a cada iteração do filtro. Este parâmetro

serve para limitar a covariância mínima da pose estimada e este parâmetro tem especial impacto quando o robô está parado.

- α_1: ganho da contribuição da translação do robô no erro da medida de translação obtida pela odometria;
- α_2: ganho da contribuição da rotação do robô no erro da medida de translação obtida pela odometria;
- α_3: ganho da contribuição da translação do robô no erro da medida de rotação obtida pela odometria;
- α_4: ganho da contribuição da rotação do robô no erro da medida de rotação obtida pela odometria;

Termina-se a apresentação do modelo de transição de estado com a definição do Jacobiano de $f(.)$ em relação a $u(k)$ utilizada na equação 3.5:

$$\nabla f_u\left(X_v(k),u(k)\right) = \begin{bmatrix} \frac{\partial x_v(k+1)}{\partial \Delta x_{odos}} & \frac{\partial x_v(k+1)}{\partial \Delta y_{odos}} & \frac{\partial x_v(k+1)}{\partial \Delta \theta_{odos}(k)} \\ \frac{\partial y_v(k+1)}{\partial \Delta x_{odos}} & \frac{\partial y_v(k+1)}{\partial \Delta y_{odos}} & \frac{\partial y_v(k+1)}{\partial \Delta \theta_{odos}(k)} \\ \frac{\partial \theta_v(k+1)}{\partial \Delta x_{odos}} & \frac{\partial \theta_v(k+1)}{\partial \Delta y_{odos}} & \frac{\partial \theta_v(k+1)}{\partial \Delta \theta_{odos}(k)} \end{bmatrix} \tag{3.8}$$

$$\nabla f_u\left(X_v(k),u(k)\right) = \begin{bmatrix} \cos\left(\theta_v(k)+\frac{\Delta\theta_{odos}(k)}{2}\right) & -\sin\left(\theta_v(k)+\frac{\Delta\theta_{odos}(k)}{2}\right) & -\frac{1}{2}\Delta x_{odos}(k)\sin\left(\theta_v(k)+\frac{\Delta\theta_{odos}(k)}{2}\right)-\frac{1}{2}\Delta y_{odos}(k)\cos\left(\theta_v(k)+\frac{\Delta\theta_{odos}(k)}{2}\right) \\ \sin\left(\theta_v(k)+\frac{\Delta\theta_{odos}(k)}{2}\right) & \cos\left(\theta_v(k)+\frac{\Delta\theta_{odos}(k)}{2}\right) & \frac{1}{2}\Delta x_{odos}(k)\cos\left(\theta_v(k)+\frac{\Delta\theta_{odos}(k)}{2}\right)-\frac{1}{2}\Delta y_{odos}(k)\sin\left(\theta_v(k)+\frac{\Delta\theta_{odos}(k)}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

Considerando agora o modelo de observação apresentado na Equação 3.10, $h(.)$ modela a posição relativa esperada (Z_B) em função da pose do robô $X_v(k)$. A partir da pose absoluta do AGV e da posição absoluta do reflector i ($X_v(k)$ e $M_{B,i}$ respectivamente, expressos no referencial W_x W_y), $h(.)$ dá-nos a posição relativa esperada do reflector i (Z_B expresso no referencial R_x R_y). O modelo de observação assume que as medidas são afectadas por um ruído aditivo gaussiano de média nula e covariância R. No presente trabalho, considera-se que R é constante e é um parâmetro do sistema. Futuramente, pretende-se refinar este modelo e tornar R função da confiança com que um reflector é detectado.

$$\begin{bmatrix} r_{B,i} \\ \phi_{B,i} \end{bmatrix} = h\left(M_{B,i},X_v(k)\right)+N(0,R) \tag{3.10}$$

Em que:

$$h\left(M_{B,i},X_v(k)\right) = \begin{bmatrix} \sqrt{\left(x_{B,i}-x_v\right)^2+\left(y_{B,i}-y_v\right)^2} \\ a\tan 2(y_{B,i}-y_v,x_{B,i}-x_v)-\theta_v(k) \end{bmatrix} \tag{3.11}$$

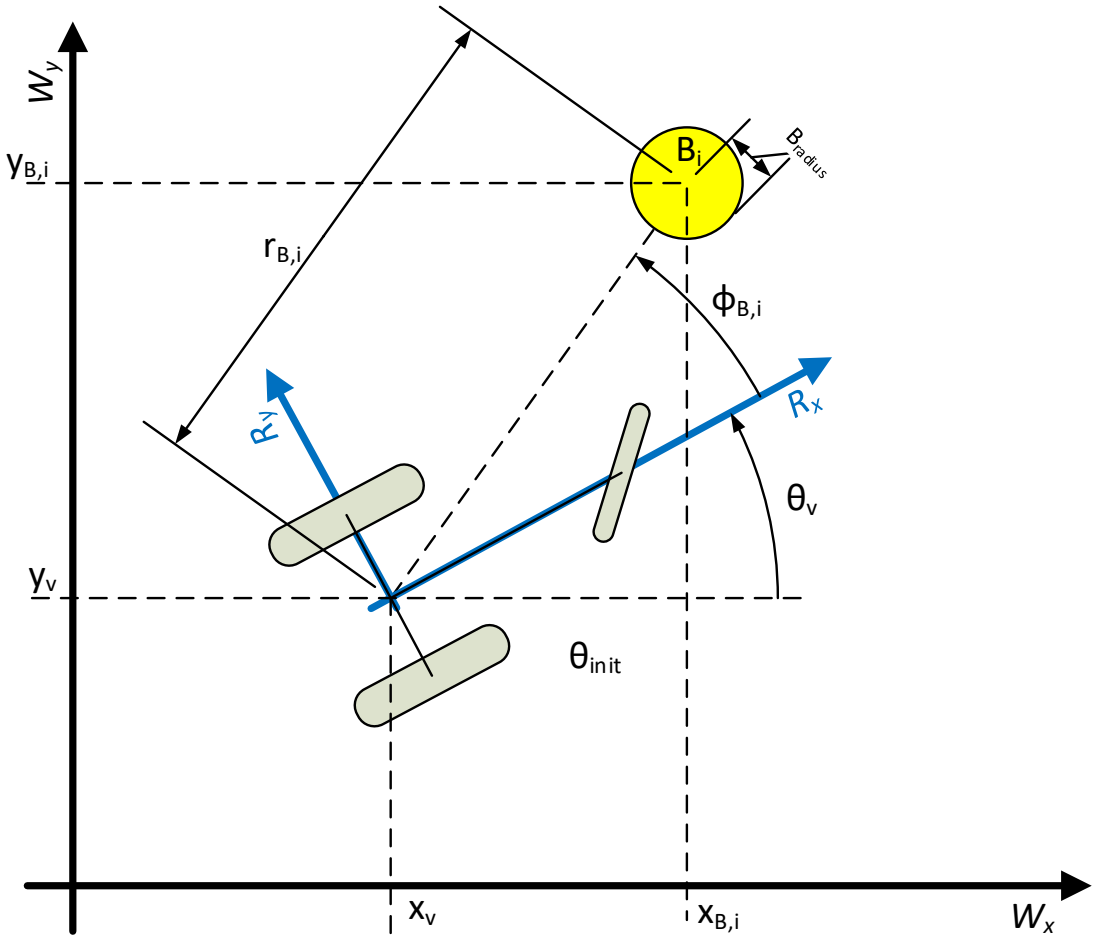


Figura 3.5: Representação das variáveis relacionadas com o modelo de observação. O modelo de observação modela a posição relativa da detecção de um refletor ($r_{B,i}$, $\phi_{B,i}$) em função da pose absoluta do robô (x_v, y_v e θ_v) e da posição absoluta do reflector detectado ($x_{B,i}$ e $y_{B,i}$).

Na equação 3.10, considera-se que a matriz de covariância R é constante definida pelos parâmetros $\sigma_{obs} = \{\sigma_r, \sigma_\phi\}$ que correspondem à variância da medida da distância e do ângulo do reflector detectado:

$$R = \begin{bmatrix} {\sigma_r}^2 & 0 \\ 0 & {\sigma_\phi}^2 \end{bmatrix} \tag{3.12}$$

Também é necessário linearizar o modelo de observação para utilizar o EKF: as equações 3.13 e 3.14 apresentam o jocobiano de $h(.)$:

$$\nabla h\left(M_{B,i}, X_v(k)\right) = \begin{bmatrix} \frac{\partial r_i}{\partial x_v(k)} & \frac{\partial r_i}{\partial y_v(k)} & \frac{\partial r_i}{\partial \theta_v(k)} \\ \frac{\partial \phi}{\partial x_v(k)} & \frac{\partial \phi}{\partial y_v(k)} & \frac{\partial \phi}{\partial \theta_v(k)} \end{bmatrix} \tag{3.13}$$

$$\nabla h\left(M_{B,i},X_v(k)\right)=\begin{bmatrix} -\frac{x_{B,i}-x_v(k)}{\sqrt{(x_{B,i}-x_v(k))^2+(y_{B,i}-y_v(k))^2}} & -\frac{y_{B,i}-y_v(k)}{\sqrt{(x_{B,i}-x_v(k))^2+(y_{B,i}-y_v(k))^2}} & 0 \\ \frac{y_{B,i}-y_v(k)}{(x_{B,i}-x_v(k))^2+(y_{B,i}-y_v(k))^2} & -\frac{x_{B,i}-x_v(k)}{(x_{B,i}-x_v(k))^2+(y_{B,i}-y_v(k))^2} & -1 \end{bmatrix} \tag{3.14}$$

Tendo definido o modelo de transição de estado, o modelo da observação e respectivos jacobianos, estamos em condições de apresentar a aplicação do EKF ao problema de localização em causa. Esta consistiu numa adaptação do algoritmo "*EKF Localization with Unknown Correspondences*" apresentado em Thrun et al. (2005).

Input : $\widehat{X}(k|k),\ P(k|k),\ u(k),\ Z_B(k)$
Output: $\widehat{X}(k+1|k+1), P(k+1|k+1)$

begin
Previsão Pose:
$\widehat{\mathrm{X}}(\mathrm{k}+1|\mathrm{k}) = f\left(\widehat{\mathrm{X}}(\mathrm{k}|\mathrm{k}),u(k)\right)$
$P(\mathrm{k}+1|\mathrm{k}) = \nabla f_X\left(\widehat{\mathrm{X}}(\mathrm{k}|\mathrm{k}),u(k)\right) P(k|k) \nabla f_X{}^T\left(\widehat{\mathrm{X}}(\mathrm{k}|\mathrm{k}),u(k)\right) + Q(k)$
$C_B(k+1) = \text{Association\_Outiliers\_Filter}\left(M_B,\widehat{\mathrm{X}}(\mathrm{k}+1|\mathrm{k}),P(\mathrm{k}+1|\mathrm{k}),Z_B(k+1)\right)$
Actualização do estado estimado:
$\widehat{X}(k+1|k+1) = \widehat{X}(k+1|k)$
$P(k+1|k+1) = P(k+1|k)$
for *all detected beacons* $Z_{B,i}$ **do**
  if $C_{B,i}$ *!= INVALID_BEACON_ID* **then**
    $\widehat{Z}_{B,i}(k+1) = h\left(M_{B,j=C_{B,i}(k+1)},\hat{\mathrm{X}}(\mathrm{k}+1|\mathrm{k}+1)\right)$
    $\nabla h = \nabla h\left(M_{B,j=C_{B,i}(k+1)},\hat{\mathrm{X}}(\mathrm{k}+1|\mathrm{k}+1)\right)$
    $S_i(k+1) = \nabla h * P(\mathrm{k}+1|\mathrm{k}+1) * \nabla h^T + R$
    $K_i(k+1) = P(\mathrm{k}+1|\mathrm{k}+1) * \nabla h^T * [S_i(k+1)]^{-1}$
    $\widehat{X}(k+1|k+1) = \widehat{X}(k+1|k+1) + K_i(k+1) * \left[Z_{B,i}(k+1) - \widehat{Z}_{B,i}(k+1)\right]$
    $P(k+1|k+1) = [I - K_i(k+1) * \nabla h] * P(k+1|k+1)$
  end
end
end

Algorithm 1: Extended Kalman Filter.

No algoritmo 1 temos detalhados os cálculos efectuados no filtro de Kalman. Tem como entradas do algoritmo: a estimativa anterior da pose e sua covariância ($X(k|k)\ P(k|k)$); posição relativa dos reflectores detectados ($Z_B(k)$); e os dados de odometria $u(k)$. $Z_B(k)$ é o resultado do pré-processamento dos dados do laser $Z_L(k)$ realizado pelo módulo *Reflector Detector* que será descrito posteriormente na Secção 3.2.1.2. Como saída do filtro temos a nova pose estimada e a sua respectiva matriz de covariância ($X(k+1|k+1)\ P(k+1|k+1)$).

De seguida apresenta-se uma breve descrição do Algoritmo 1; podem ser encontrados mais detalhes em Thrun et al. (2005);

- Linha 3: Aplicação da função de transição de estado. Com isto obtemos a previsão da próxima pose em função dos dados de odometria;
- Linha 4: Cálculo da respectiva covariância da nova estimativa da pose. Esta é composta por dois componentes, uma proveniente da propagação da covariância da pose no modelo de transição de estado, e outra proveniente da matriz $Q(.)$ definida na equação 3.5.
- Linha 5: Identificação dos reflectores detectados: Correspondência/associação entre os reflectores detectados ($Z_B(k)$) e o mapa de reflectores (M_B) (apresentado mais tarde, na Secção 3.2.1.3). A saída C_B é um vector de inteiros com o mesmo tamanho de Z_B. Cada elemento de C_B, $C_{B,i}$, corresponde ao índice do mapa de reflectores ao qual a detecção $Z_{B,i}(k)$ foi associada;
- Linha 7 e Linha 8: Inicialização dos *outputs* do algoritmo. A pose é estimada pelo filtro após processar as observações $X(k+1|k+1)$ e sua respectiva covariância $P(k+1|k+1)$. Se não existirem observações, a próxima estimativa será a calculada apenas com base na função de transição de estado.
- Linha 9 e Linha 10: A informação de todas as medidas não marcadas como *outliers* é integrada no estado estimado pelo filtro. É nesta fase do EKF que é feita a correcção do estado estimado.
- Linha 11 e linha 12: Cálculo da previsão das observações com a aplicação do modelo de observação em função da pose prevista;
- Linha 13: Cálculo da covariância das observações previstas;
- Linha 14: Cálculo o ganho de Kalman;
- Linha 15: Nova pose estimada a partir dos novos dados observados;
- Linha 16: Nova covariância da pose resultante do cálculo anterior.

Feita a descrição da aplicação do EKF, apresenta-se os restantes módulos que compõem o sistema. Estes módulos são mais específicos do problema em questão e têm como objectivo aumentar a robustez global da solução apresentada.

3.2.1.2 Detector-reflectores

Este módulo é o responsável por processar os dados provenientes do laser Z_L de forma a detectar reflectores e medir as suas posições relativamente ao referencial do robô (Z_B em relação ao referencial R_x R_y, Figura 3.1). Z_B define um conjunto de posições polares denominadas por $[r_i, \phi_i]^T$. Como já foi referido anteriormente, Z_L é composto por um conjunto de medidas, estando associado um valor $C_{L,i}$ a cada uma delas, que indica se o objecto detectado tem ou não uma elevada reflectividade. Os reflectores utilizados para a localização têm naturalmente uma grande

reflectividade mas não são os únicos objectos detectados pelo laser com essa característica, o que faz com que, por vezes, medidas com um valor de $C_{L,i}$ igual a verdadeiro não correspondam na realidade a reflectores (falsos positivos).

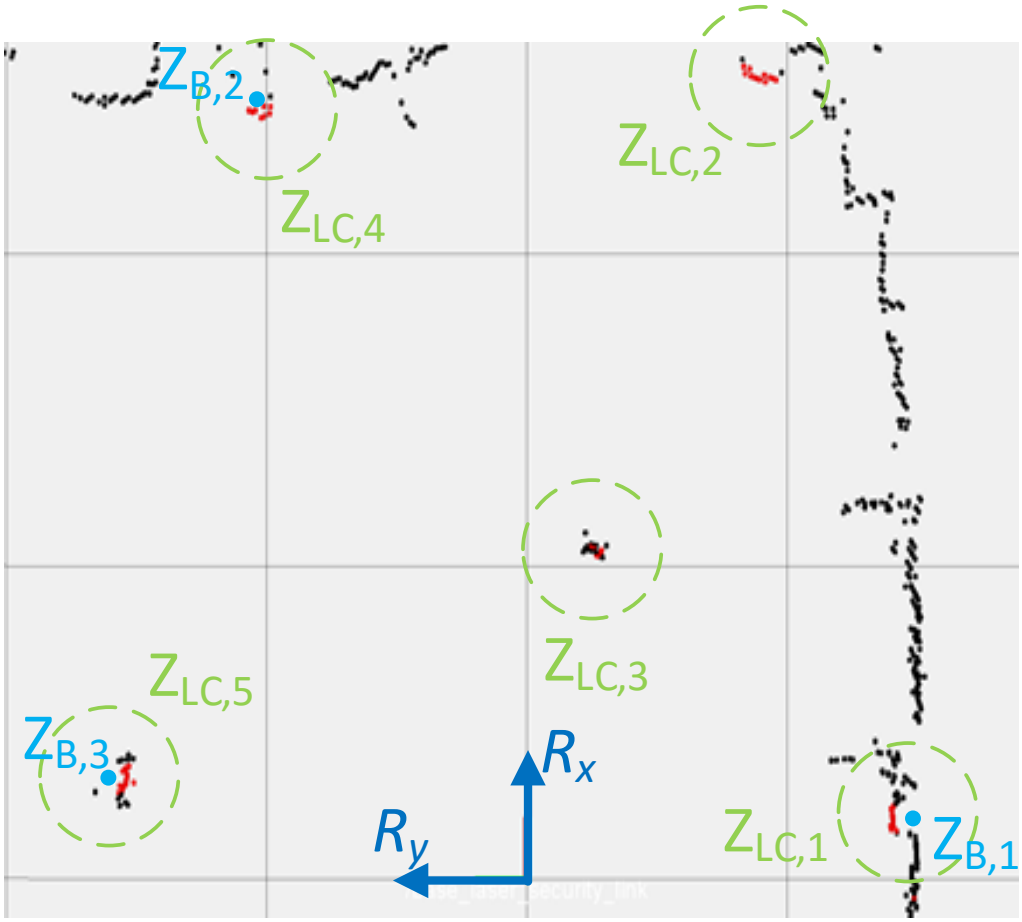


Figura 3.6: Exemplo dos dados adquiridos por um *laser scanner* (Z_L). Os pontos azuis e vermelhos são posições relativas (no referencial R_x R_y) correspondentes aos objectos detectados, sendo os pontos vermelhos medidas com elevada reflectividade. A verde e a azul representam-se os dados relativos ao módulo "*refector detector*". Neste exemplo, estão assinalados cinco conjuntos indicados de $Z_{LC,1}$ a $Z_{LC,5}$. Cada conjunto corresponde a uma sequência de medidas com elevada reflectividade. Destes conjuntos, apenas três correspondem a reflectores. Na Figura é indicada a posição relativa dos reflectores detectados ($Z_{B,1}$ a $Z_{B,3}$), e os conjuntos ($Z_{LC,2}$ e $Z_{LC,3}$) são identificados como *outliers* e ignorados pelo sistema.

Na figura 3.6, apresenta-se um exemplo de um *scan* de um *laser* onde se podem observar cinco conjuntos de pontos sinalizados a vermelho. Estes conjuntos correspondem a detecções com alta reflectividade, mas apenas três são medidas provenientes de reflectores. De forma a tornar a detecção mais robusta, integrou-se na detecção um filtro ("*Detector Filter*") que tem em conta o tamanho dos reflectores instalados (B_{radius}).

Na Figura 3.7 encontra-se uma representação da relação geométrica utilizada no "*Detector Filter*": sabendo a distância entre o laser scanner e o reflector é possível calcular o número de

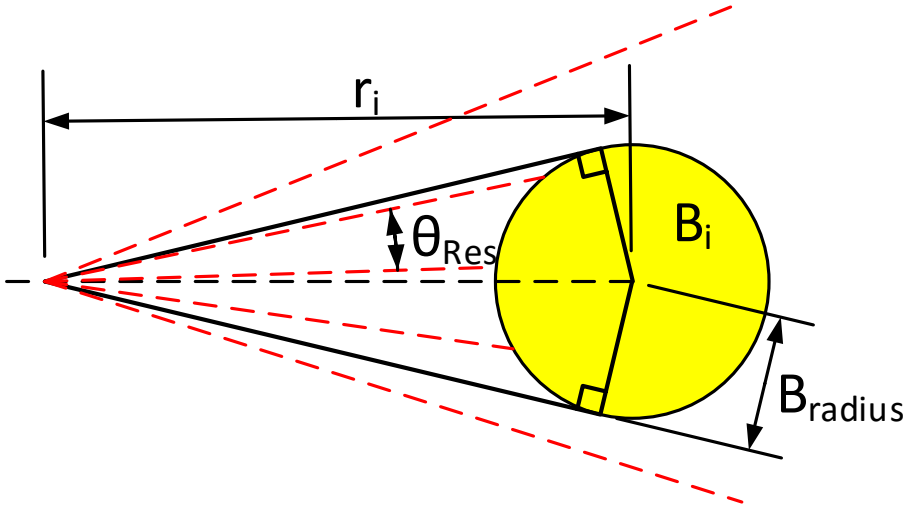


Figura 3.7: Relação geométrica entre a distância (r_i) a um reflector (B_i) e o número de feixes do *laser scanner*.

feixes (número de elementos num conjunto) que interceptam o reflector. Com isto, sabendo o raio dos reflectores instalados (B_{radius}) e a resolução do *laser scanner* (θ_{Res}), definimos o modelo presente na equação 3.15:

$$M_{num}(r) = floor\left(\frac{2\arcsin\left(\frac{B_{radius}}{r}\right)}{\theta_{\text{Res}}}\right) \tag{3.15}$$

Utilizando este filtro é possível distinguir reflectores de outros objectos, tendo em conta as suas dimensões. No Algoritmo 2 encontram-se os cálculos do módulo "*Reflector Detector*"cuja descrição se apresenta em seguida.

Input : $Z_L(k)$
Output: $Z_B(k)$
begin
  $Z_{LC}(k) = \{Z_{LC,i}(k) : i \in 1, \ldots, numOfClusters\}$
  $Z_{LC}(k) = ProcessClusters(Z_L)$
  for $allclustersZ_{LC,i}(k)$ **do**
    $\begin{bmatrix} r_{B,new} \\ \phi_{B,new} \end{bmatrix} = ComputeReflectorCenter(Z_{LC,i}(k))$
    $modelError = size((Z_{LC,i}(k)) - M_{num}(r_{B,new})$
    if $(modelError \leq MaxModelError)$ *and* $(modelError \geq MinModelError)$ **then**
      $add \begin{bmatrix} r_{B,new} \\ \phi_{B,new} \end{bmatrix} to\ Z_B(k)$
    end
  end
end

Algorithm 2: Reflector Detector.

- Linha 2 e 3: Os dados do laser Z_L são divididos em conjuntos em função da sua reflectividade ($C_{L,i}$). Um cluster, $Z_{LC,i}(k)$, é uma sequência de posições relativas com alta reflectividade. No exemplo da Figura 3.6, são apresentados cinco conjuntos, indicados por $Z_{LC,1}(k)$ a $Z_{LC,5}(k)$.

- Linha 5: Para cada *cluster* são calculados, em coordenadas polares, a sua distância e ângulo ($r_{B,new}$ e $\phi_{B,new}$) em relação ao referencial do robô (R_x R_y). Por enquanto, "*ComputeReflectorCenter*"consiste simplesmente em considerar o ângulo e distância (somado pelo raio dos reflectores B_{radius}) do elemento central do *cluster*. Consequentemente, também se considera na equação 3.10 o ruído de observação R constante. Pretende-se refinar este modelo no futuro.

- Linha 6: É calculado o erro entre o valor teórico dado pela equação 3.15 e o número de medidas associadas a um *cluster*;

- Linha 7 e Linha 9: A aplicação "*Detector Filter*" identifica e elimina possíveis *outliers*. "*MaxModelError*"e "*MinModelError*"são, respectivamente, a tolerância máxima e mínima de erro entre o modelo e o número de pontos observados.

De forma a verificar a qualidade do modelo definido na Equação 3.15 foi realizada uma experiência, tendo-se registado o número de feixes incidentes num reflector para várias distâncias conhecidas. Os resultados estão apresentados no gráfico da Figura 3.8. Isto tornou possível a validação da função distância-número de feixes (Equação 3.15) e a escolha de valores adequados para a tolerância do "*detector filter*". Foram considerados os valores dois e zero para "*MaxModelError*"e "*MinModelError*", respectivamente.

3.2.1.3 Associação/Filtro outliers

Este módulo tem duas funções: identificar os reflectores detectados e filtrar *outliers* que não tenham sido detectados pelo filtro anterior. Esta filtragem tem em conta a estimativa actual da pose do robô $X(.)$ a covariância $P(.)$ e o mapa de reflectores M_B rejeitando medidas de acordo com a sua probabilidade da sua ocorrência.

Este algoritmo é apresentado em 3, e tem como entradas a pose do robô e respectiva covariância após o processamento dos dados de odometria ($\widehat{Z}_B(k)$ e $S(k)$) e o conjunto dos reflectores detectados Z_B. Como resultado deste algoritmo temos C_B, que consiste num vector de inteiros (com o mesmo número de elementos de Z_B). Cada elemento $C_{B,i}$ indica qual o índice do mapa de reflectores M_B ao qual observação (detecção) $Z_{B,i}$ foi associada; caso a observação deva ser rejeitada, o elemento $C_{B,i}$ devolve um valor indicando isso mesmo: *INVALID_BEACON_ID*.

De seguida apresenta-se uma descrição do Algoritmo 3:

- Linha 3 à Linha 5: Para todos os elementos do mapa M_B são calculadas a previsão da observação e a sua respectiva covariância.

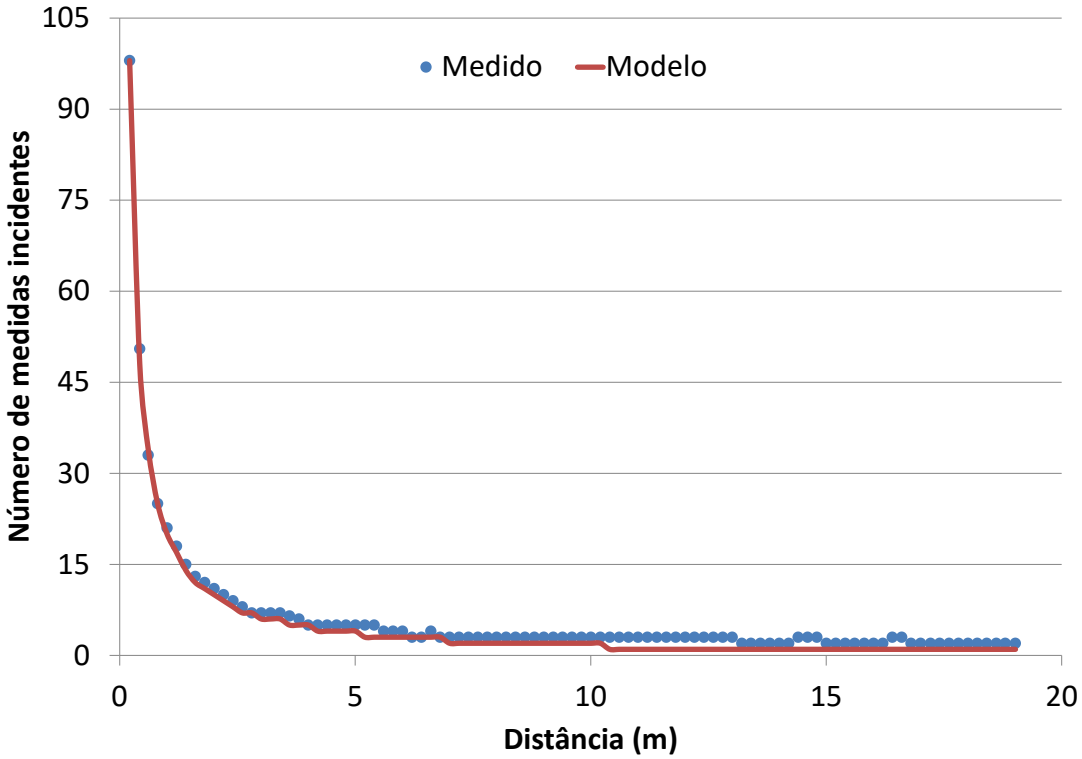


Figura 3.8: Validação experimental do modelo apresentado na Equação 3.15. Os resultados experimentais referentes ao número de feixes incidentes num reflector a uma distância conhecida estão representados pela linha azul. A correspondente resposta do modelo é representada pela linha vermelha.

- Linha 9: Aqui a observação $Z_{B,i}$ é associada ao elemento $M_{B,i}$ que maximiza a probabilidade da sua ocorrência. Para cada elemento j do mapa M_B é calculado o valor da função de distribuição de probabilidade da ocorrência da observação $Z_{B,i}$ (*likelihood*). A observação i é associada ao elemento j do mapa que apresentar um valor superior (*maximum likelihood*). Podem ser encontrados mais detalhes acerca desta técnica de associação em (Thrun et al., 2005).

- Linha 11: χ^2_{df} é um valor constante, função de P_G (figura 3.4, probabilidade mínima de validação). Define no espaço das observações uma área na qual a probabilidade da ocorrência da observação é superior a P_G. χ^2_{df} corresponde ao inverso da função de densidade de uma distribuição chi-quadrado com dois graus de liberdade. Neste ponto, acrescenta-se uma descrição estatística mais adequada à descrição de um *outlier* do que em (Thrun et al., 2005) visto que estes autores sugerem uma filtragem apenas com base no valor de *likelihood* das observações. Esta questão será abordada com mais detalhe na Secção 3.2.1.5.

Na Figura 4.16 têm-se uma representação da área de validação no espaço cartesiano. A elipse azul corresponde à área onde, tendo em conta a pose estimada do AGV, a probabilidade de detectar

Input : $Z_B(k)$, $\widehat{Z}_B(k)$, $S(k)$
Output: $C_B(k)$

begin
for *all beacons* $M_{B,i}$ **do**
  $\widehat{Z}_{B,i}(k) = h\left(M_{B,i}, \hat{\mathrm{X}}\left(\mathrm{k}+1|\mathrm{k}\right)\right)$
  $\nabla h = \nabla h\left(M_{B,i}, \hat{\mathrm{X}}\left(\mathrm{k}+1|\mathrm{k}\right)\right)$
  $S_i(k) = \nabla h * P\left(\mathrm{k}+1|\mathrm{k}\right) * \nabla h^T + R$
end
for *all detected beacons* $Z_{B,i}$ **do**
  Association:
  $C_{B,i}(k) = \underset{j}{\operatorname{argmax}} \det\left(2\pi S_j(k)\right) e^{-\frac{1}{2}\left(Z_{B,i}(k)-\widehat{Z}_{B,j}(k)\right)^T \left[S_j(k)\right]^{-1}\left(Z_{B,i}(k)-\widehat{Z}_{B,j}(k)\right)}$
  Association filter:
  if $\left(Z_{B,i}(k) - \widehat{Z}_{j=C_{B,i}(k)}(k)\right)^T \left(\mathrm{S}_{j=C_{B,i}(k)}(\mathrm{k})\right)^{-1} \left(Z_{B,i}(k) - \widehat{Z}_{j=C_{B,i}(k)}(k)\right) > \chi^2_{df=2}$ **then**
    $C_{B,i}$ = INVALID_BEACON_ID
  end
end
end

Algorithm 3: Association Outliers Filter.

um reflector é P_G, que neste caso em particular é 95%.

3.2.1.4 Supervisor de incerteza

Este módulo é responsável por supervisionar o estado do sistema de localização. Esta supervisão é baseada na quantidade de incerteza estimada pelo filtro de Kalman estendido.O seu objectivo é detectar falhas no sistema de localização, nas quais a estimação da pose do AGV não é confiável o suficiente para assegurar uma navegação segura.

Existem diversas situações que podem levar a um aumento significativo da incerteza da estimativa da pose do robô e que podem levar à situação de falha do sistema de localização. Uma situação comum é a quantidade de reflectores detectados ser inferior a dois por longos períodos de tempo. Este défice de informação sensorial pode ser provocado pela oclusão de reflectores e/ou por uma má distribuição de reflectores na área de navegação. Outra situação mais problemática ocorre quando o filtro de kalman falha o *tracking* da pose e converge para a solução errada. Isto pode ser causado por erros grosseiros nos sensores do robô, como por exemplo a derrapagem das rodas do robô, o que leva a medidas erradas do deslocamento, ou o processamento de um falso reflector que não tenha sido filtrado pelo sistema. Além de questões relacionadas com a segurança da navegação do AGV, existe uma motivação extra para não permitir incertezas elevadas. É preciso notar que se recorreu a um filtro de kalman estendido para *tracking* da pose e, em geral, técnicas de linearização gaussianas funcionam melhor se a quantidade de incerteza for baixa. Existem portanto várias razões para se desejar uma incerteza baixa, por exemplo, o risco de identificações erradas de reflectores diminui quando a incerteza da estimação do filtro é baixa. Isto é importante porque basta apenas uma falsa identificação para fazer divergir o filtro; a partir desse momento, todo o

processo de identificação de reflectores fica comprometido. Provocando uma situação de falha que obriga a reinicializar o filtro. Além disto, geralmente o erro de linearização é baixo apenas na proximidade do ponto de linearização. (Thrun et al., 2005) aborda estas questões e refere que na prática se o desvio padrão para a orientação for maior que $\pm 20^{\circ}$, os erros de linearização poderão provavelmente levar a que o Filtro de kalman estendido falhe. Pretende-se que, quando o sistema detecte uma falha ("*fail mode*"), o AGV pare. No entanto, quando tal se deve à situação em que os reflectores estão temporariamente obstruídos (pessoas ou outras máquinas em movimento) temos interesse em que o sistema volte a ficar operacional automaticamente sem intervenção externa. Neste sentido, mesmo em "*fail mode*", o sistema de localização continua a processar os dados dos sensores e a verificar se estes diminuem a incerteza da localização.

Input : $\widehat{X}(k+1|k+1), P(k+1|k+1), \widehat{X}(k|k), P(k|k)$
Output: $\widehat{X}_v(k+1|k+1), P_v(k+1|k+1)$

begin
Err_p := Parâmetro, erro max de posição
Err_o := Parâmetro, erro max de orientação
R_B := Parâmetro, folga para recuperação automática
$P(k+1|k+1) = \begin{bmatrix} \text{var}(x_v) & \text{cov}(x_v, y_v) & \text{cov}(x_v, \theta_v) \\ \text{cov}(x_v, y_v) & \text{var}(y_v) & \text{cov}(y_v, \theta_v) \\ \text{cov}(x_v, \theta_v) & \text{cov}(y_v, \theta_v) & \text{var}(\theta_v) \end{bmatrix}$
$P_{Norm}(k+1|k+1) = \begin{bmatrix} \frac{\text{var}(x_v)}{Err_p{}^2} & \frac{\text{cov}(x_v,y_v)}{Err_p{}^2} & \frac{\text{cov}(x_v,\theta_v)}{Err_pErr_o} \\ \frac{\text{cov}(x_v,y_v)}{Err_p{}^2} & \frac{\text{var}(y_v)}{Err_p{}^2} & \frac{\text{cov}(y_v,\theta_v)}{Err_pErr_o} \\ \frac{\text{cov}(x_v,\theta_v)}{Err_pErr_o} & \frac{\text{cov}(y_v,\theta_v)}{Err_pErr_o} & \frac{\text{var}(\theta_v)}{Err_o{}^2} \end{bmatrix}$
$MaxVar$:= ComputeLargestEigenvalue $(P_{Norm}(k+1|k+1))$
$FaultConditionState : S_{Fault}$
if $S_{Fault} = false$ **then**
  if $\left(2\sqrt{MaxVar} > 1\right)$ **then**
    S_{Fault}:= true
  end
else
  if $\left(2\sqrt{MaxVar} < R_B\right)$ **then**
    S_{Fault}:= false
  end
end
if $S_{Fault} = false$ **then**
  $\widehat{X}_v(k+1|k+1)$:=$\widehat{X}(k+1|k+1)$
  $P_v(k+1|k+1)$:=$P(k+1|k+1)$
else
  $\widehat{X}_v(k+1|k+1)$:=$\widehat{X}(k|k)$
  $P_v(k+1|k+1)$:=$P(k|k)$
end
end

Algorithm 4: Uncertainty supervisor.

O Algoritmo 4 descreve o módulo "*Uncertainty supervisor*". Este tem como entradas o estado

anterior do EKF ($\widehat{X}(k|k)$, $P(k|k)$) e o novo estado após processar a nova informação sensorial ($\widehat{X}(k+1|k+1)$, $P(k+1|k+1)$). Uma destas duas entradas será a saída deste módulo ($\widehat{X}_v(k+1|k+1), P_v(k+1|k+1)$). O módulo tem como parâmetros o erro máximo de posição (Err_p) e erro máximo de orientação (Err_o) e o parâmetro R_B ("*auto recover backlash*") serve para evitar transições intermitentes do sistema entre o modo de falha e o modo operacional; R_B está definido entre zero e um. Como existe um parâmetro referente ao erro de posição e outro para a orientação, torna-se necessário normalizar a matriz de covariância da pose estimada $P(k+1,k+1)$ (Linha 6). A incerteza máxima (*MaxVar*) é extraída de $P(k+1,k+1)$ na Linha 7, a partir do cálculo do seu maior valor próprio. O valor de *MaxVar* é então utilizado na Linha 10 como factor de decisão para o sistema entrar em modo de falha. Se esta condição se verificar, o sistema entra em modo de falha e o novo resultado do EKF ($\widehat{X}(k+1|k+1)$, $P(k+1|k+1)$ é descartado. A partir deste ponto passa, a ser processada a Linha 14. Aqui, a condição para o sistema sair do estado de falha é testada, o que corresponde aos desvios padrões abaixo de Err_p e Err_o depois da nova informação sensorial ser processada.

Como trabalho futuro, pretende-se aprofundar mais a questão da divergência do filtro bem como até que ponto a quantidade de incerteza estimada permite detectar com confiança esta perigosa situação.

3.2.1.5 Inovações introduzidas no algoritmo base

O algoritmo descrito é baseado no Algoritmo "*EKF_localization*" apresentado no livro "*Probabilistic Robotics*" (Thrun et al., 2005). Foram efectuadas algumas alterações, nomeadamente ao nível da filtragem de *outliers* e na sequência de operações de associação/*udpate*. Nas próximas secções descrevem as alterações efectuadas e o motivo pelo qual estas foram efectuadas.

Filtragem de outliers

Para filtragem de *outliers*, Thrun et al. (2005) sugere usar o valor da função de probabilidade (linha 9 do Algoritmo 3), isto é, o valor da "*observation likehood*", o qual se procura maximizar durante o processo de associação. Ou seja, observações com um valor de *likehood* baixo devem ser ignoradas.

Na abordagem aqui proposta, recorre-se à descrição estatística da ocorrência das medidas, composta pelo valor médio e respectiva covariância, obtidos no cálculo da previsão das observações. Com isto define-se uma probabilidade mínima para que um reflector detectado seja considerado. Considera-se esta abordagem estatisticamente mais precisa, com um parâmetro de filtragem com um significado numérico bem definido. Por exemplo, na actual implementação, observações com uma probabilidade inferior a 5% são ignoradas.

Para dar uma ideia visual da diferença entre usar o valor de *likehood* ou a probabilidade de uma medida, apresenta-se a Figura 3.9, na qual se pode observar duas distribuições gaussianas com diferentes variâncias. A linha verde representa um valor hipotético de *likehood* que definiria se uma medida seria rejeitada ou não. Neste exemplo, esse valor é 0.134. Pode-se observar que, na distribuição a vermelho, existem valores com uma *likehood* superior a 0.134, cuja probabilidade

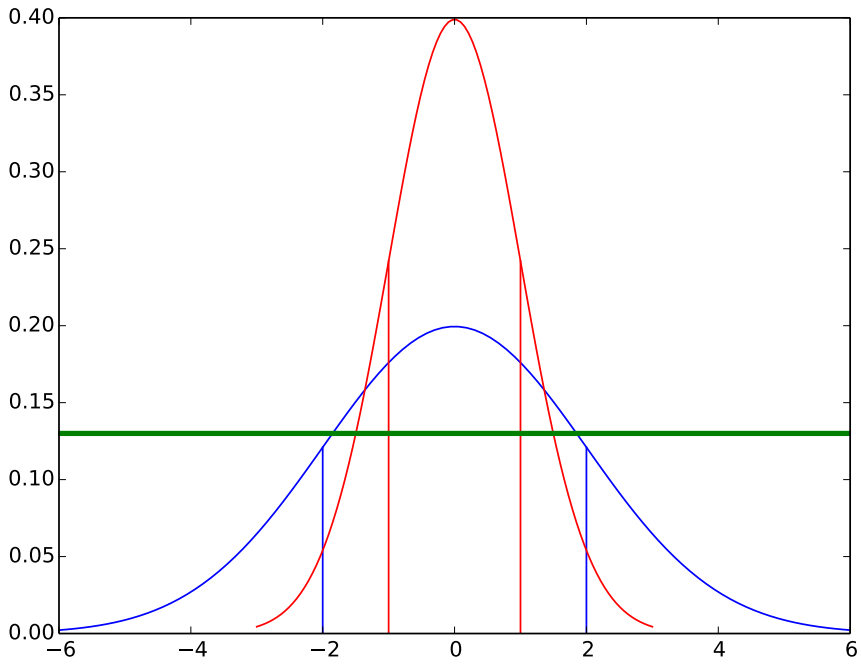


Figura 3.9: Exemplo de duas distribuições normais com diferentes variâncias. As linhas verticais (azuis e vermelhas) delimitam o intervalo de confiança correspondente a um desvio padrão. A verde temos um exemplo de um valor de *likehood*.

de ocorrência é inferior a 63%. Por outro lado, na distribuição a azul, existem valores com uma *likehood* inferior a 0.134, cuja probabilidade de ocorrência é superior a 63%. O valor de desvio padrão (63%) está representado na Figura 3.9 através de linhas verticais, para cada distribuição.

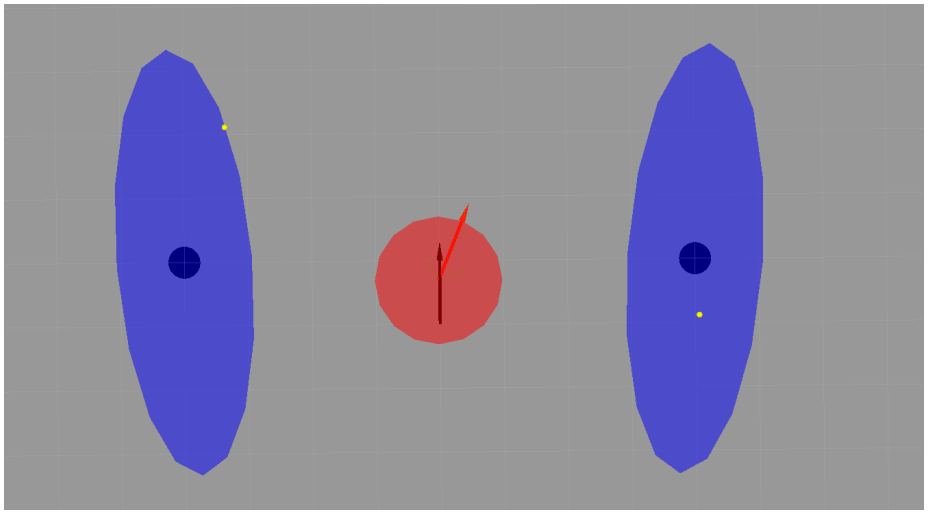
Sequência de operações

Considerando-se que durante um *scan* são detectados *n* reflectores, para Thrun et al. (2005) essas *n* medidas são tratadas de forma independente. Ou seja, é equivalente detectar *n* reflectores num único *scan* ou detectá-los num conjunto de *n scans* sucessivos, sendo detectado apenas um reflector por *scan*. Em traços gerais, cada observação é associada ao mapa de reflectores e depois integrada no filtro de forma sequencial. No trabalho aqui descrito, associam-se primeiro todos os reflectores detectados e só depois é que estes são fundidos com o estado do robô. Uma das motivações que levou a esta alteração prende-se com facto de, no algoritmo original, a ordem com que os reflectores detectados são processados num *scan* poder influenciar o resultado final da localização.

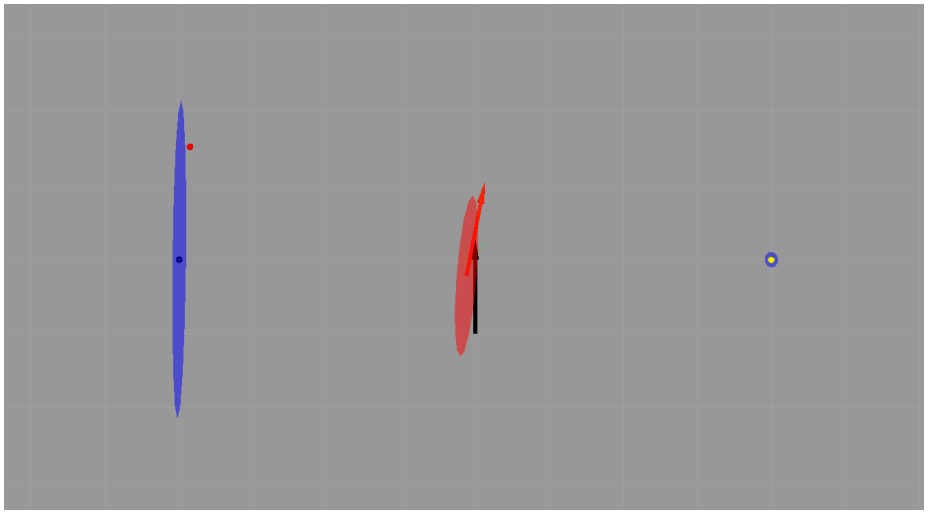
A Figura 3.10a ilustra uma situação em que a ordem de processamento das observações tem impacto no resultado final. São representadas a pose e respectiva covariância (seta e elipse vermelha respectivamente) antes de serem processadas as detecções (aqui representadas pelos pontos

amarelos); a seta a preto indica a verdadeira pose do robô. Neste caso, devido a erros de linearização, se se processar primeiro a medida referente ao reflector da direita, o reflector da esquerda vai ser rejeitado na iteração seguinte, por estar fora da zona de validação, tal como se pode observar em 3.10b. Por outro lado, se se processar primeiro o reflector da esquerda e depois o da direita, esta situação já não acontece.

Na abordagem proposta neste trabalho, a ordem pela qual os reflectores detectados são processados durante o processo de associação é irrelevante, já que o processo de associação/filtragem depende apenas da previsão da pose do robô após a integração dos dados da odometria.



(a) Antes do processamento das observações.



(b) Após do processamento das observações.

Figura 3.10: Exemplo de uma situação em que a ordem de processamento das observações tem impacto importante no resultado final da solução apresentada por Thrun et al. (2005). A seta preta e as circunferências a azul escuro indicam, respectivamente, a verdadeira pose do robô e a posição de dois reflectores. A seta e a elipse vermelha representam a pose estimada e respectiva covariância. As elipses azuis representam as covariâncias da previsão das observações.

Por fim, resta referir que esta alteração na sequência de operações no filtro torna o algoritmo

mais leve computacionalmente. Mediu-se o peso computacional (utilizando um CPU Intel Core i7 870) das duas abordagens para um mapa de reflectores com 30 elementos, em duas situações com diferentes quantidades de detecções e obtiveram-se os tempos de processamento apresentados na Tabela 3.1.

Tabela 3.1: Influência das inovações introduzidas no algoritmo base no peso computacional.

Reflectores observados	"*Probabilistic Robotics*"	Abordagem aqui apresentada
3	2.5ms	1.9ms
30	13ms	8.0ms

Como se pode observar na Tabela 3.1, a alteração efectuada apresenta uma melhoria da desempenho computacional do algoritmo de localização. Esta redução de peso computacional deve-se, em traços gerais, ao facto que dos cálculos referentes à previsão das observações serem efectuados menos vezes.

Em suma, utiliza-se um filtro de kalman estendido para combinar a informação dos reflectores detectados com a odometria. A robustez do sistema assenta na filtragem de *outliers* efectuada por dois filtros em série: "*detector filter*" e o "*association filter*". O "*detector filter*" tem em conta o tamanho dos reflectores detectados, enquanto, o "*association filter*", baseado na pose estimada e respectiva covariância, rejeita medidas improváveis de reflectores. Como em certas aplicações (nomeadamente as industriais) é indispensável ter garantias de segurança, analisou-se a importância de ter um supervisor para detectar falhas no sistema de localização.

3.2.2 Localização baseada em contornos

Nesta secção, apresenta-se o trabalho desenvolvido relativamente à utilização de contornos para a localização de um robô móvel. Neste âmbito, foi desenvolvido um algoritmo de localização baseado no trabalho apresentado por Lauer et al. (2006). Este algoritmo, apelidado de "*Perfect Match*"(PM), é um algoritmo que calcula a transformação entre um conjunto de pontos e uma grelha de ocupação, pertencendo à classe de algoritmos de *Map Matching*. Este algoritmo tem grandes vantagens em termos de peso computacional, tendo sido aplicado com sucesso nos sistemas de localização de robôs utilizados no campeonato de futebol robótico médio. Os desenvolvimentos efectuados têm como objectivo aumentar a precisão e robustez do Perfect Match, bem como estender a sua aplicação a outros domínios, como por exemplo os AGVs industriais e outros robôs móveis. Para tal, foi dada especial atenção à precisão, uma vez que é um requisito fundamental em ambientes industriais. Foram estudadas formas de assegurar valores adequados de precisão nestes ambientes dinâmicos onde os sistemas de localização são afectados por *outliers* correspondentes a objectos que não fazem parte do mapa que o robô utiliza. Neste sentido, foram adicionadas as seguintes funcionalidades:

- Melhoramento da fusão sensorial entre os dados dos sensores de distâncias a obstáculos (geralmente *lasers*) e sensores associados à odometria (geralmente encoders), recorrendo a um Filtro de Kalman;

- Verificação de convergência da solução do algoritmo de localização. Foi definido um critério de convergência, permitindo um número de iterações dinâmico, o que não acontecia na solução original;

- Testes à alteração da função de custo, de forma a ter em conta a distância a que os objectos são detectados pelo sensor;

- Análise da influência dos *outliers* na precisão do sistema de localização e desenvolvimento de algoritmos de identificação dos mesmos no conjunto das medidas obtidas pelo laser. Os *outliers* correspondem a objectos que não estão mapeados e que devem ser rejeitadas.

De forma a aumentar a robustez do sistema de localização, também é abordada a questão da localização global. Sendo o "*Perfect Match*"um algoritmo de *tracking* ele necessita de uma estimativa inicial da pose do robô com um determinado erro máximo em relação à sua verdadeira pose. Essa estimativa é obtida recorrendo à pose anteriormente calculada e ao sistema odométrico do robô. Isto permite ultrapassar as situações de falha do *tracking* da pose, nas quais o sistema de localização devido por exemplo a um erro grosseiro na odometria, está continuamente a fornecer estimativas erróneas da pose do robô, situação em que normalmente é difícil recuperar sem intervenção humana. Estas falhas são apelidadas de "situação de perdido". Desta forma, foi adicionado um conjunto de melhorias com o objectivo de aumentar a robustez do "*Perfect Match*". Estas melhorias consistiram na:

- Implementação de um algoritmo de detecção de perdido, isto é, da falha completa da estimativa da localização. Esta implementação explora a redundância entre o valor de orientação global obtida pelo sistema de localização e, por uma bússola magnética instalada no robô.

- Implementação de um algoritmo de localização global, *i.e*, de um algoritmo que procura a pose do robô não recorrendo a estimativas anteriores. No momento em que é detectada a situação de perdido, o sistema de localização activa o modo de localização global, testando diversas hipóteses ao longo da área de navegação, procurando uma solução global que maximize a correspondência entre os pontos do sistema global e o mapa do mundo.

A Figura 3.11 apresenta a arquitectura geral do sistema desenvolvido. Sendo um algoritmo de *tracking*, este tem como entradas o estado anterior do filtro $X(k|k)$ e $P(k|k)$, além dos dados dos sensores compostos pela odometria ($u(k)$) e os contornos ($Z_C(k)$). As entradas na Figura 3.11 são assinaladas pelos rectângulos vermelhos. A saída deste sistema é o novo estado do sistema de localização, composto pela nova pose estimada e pela respectiva covariância (rectângulo azul). Os rectângulos negros representam os diferentes componentes da solução proposta.

Em traços gerais, a solução proposta consiste na combinação de um algoritmo de fusão sensorial (EKF) com um algoritmo de *map matching*. No lado esquerdo da Figura 3.11 encontram-se os componentes do filtro de Kalman, enquanto, no lado direito estão presentes os componentes relativos ao algoritmo de *map matching*.

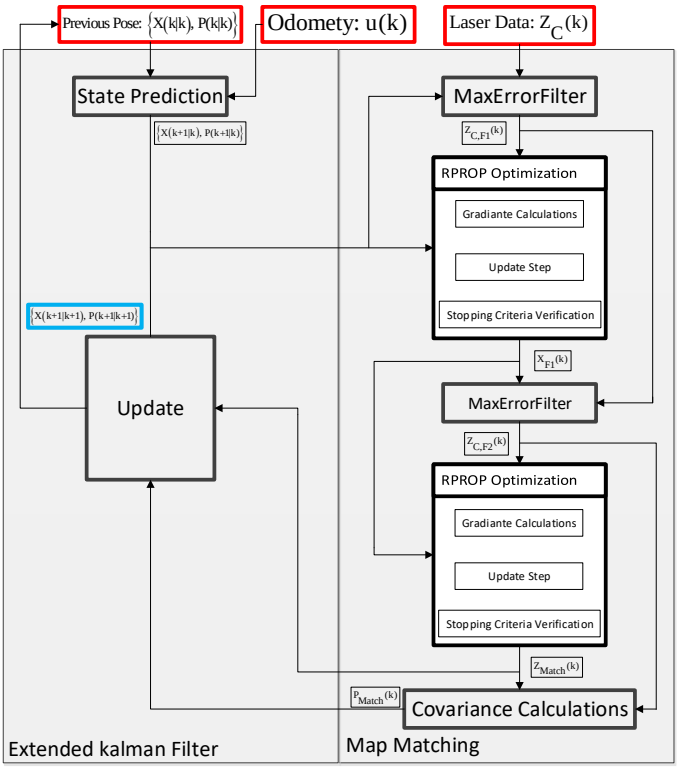


Figura 3.11: Arquitectura do sistema de localização baseado em contornos. Cada bloco representa um componente do sistema com as suas respectivas entradas e saídas. As entradas e saídas do sistema estão representados pelos blocos vermelhos e azuis, respectivamente. A solução proposta consiste na combinação de um algoritmo de fusão sensorial (filtro de Kalman) com um algoritmo de *map matching*.

3.2.2.1 Extended Kalman Filter

À semelhança da solução baseada em reflectores, também aqui se recorreu a um Filtro de Kalman Estendido para fundir a informação sensorial dos *encoders*, $u(k)$, com os dados dos sensores, Z_{Match}, de forma a obter uma estimativa da pose X_v, caracterizada por uma matriz de covariância P. Como referido na Secção 3.2.1.1, para possibilitar a utilização do EKF é necessário definir o modelo de transição de estado e o modelo de observação. No que concerne ao modelo de transição de estado, foi utilizado o apresentado na Secção 3.2.1.1. Em relação ao modelo de observação, foi considerado que o filtro observa directamente o seu estado através de Z_{Match}, sendo afectado por ruído gaussiano. Este último é caracterizado pela matriz de covariância P_{Match} dado pela Equação 3.16.

$$Z_{Match} = X_V + N(0, P_{Match}) \tag{3.16}$$

Em que X_V representa a pose do robô, enquanto Z_{Match} e P_{Match} correspondem, respectivamente, à medida e respectiva matriz de covariância da pose do robô obtidas a partir do algoritmo de *matching*.

Input : $\widehat{X}(k|k)$, $P(k|k)$, $u(k)$, $Z_{Match}(k)$, $P_{Match}(k)$
Output: $\widehat{X}(k+1|k+1), P(k+1|k+1)$
1 **begin**
2 Previsão Pose:
3 $\widehat{X}(k+1|k) = f\left(\widehat{X}(k|k), u(k)\right)$
4 $P(k+1|k) = \nabla f_X\left(\widehat{X}(k|k), u(k)\right) * P(k|k) * \nabla f_X^T\left(\widehat{X}(k|k), u(k)\right) + Q(k)$
5 Actualização do estado estimado:
6 $K(k) = P(k+1|k)\left[P(k+1|k) + P_{Match}(k)\right]^{-1}$
7 $\widehat{X}(k+1|k+1) = \widehat{X}(k+1|k) + K(k)\left(Z_{Match}(k) - \widehat{X}(k+1|k)\right)$
8 $P(k+1|k+1) = [I - K(k)]P(k+1|k)$
9 **end**

Algorithm 5: Extended Kalman Filter.

No Algoritmo 5 são especificados os cálculos efectuados na fusão sensorial no sistema de localização baseado em contornos. As Linhas 3 e 4 são similares ao Algoritmo 1, e correspondem à aplicação do modelo de transição de estado. No modelo de observação definido a previsão das observações corresponde à pose do robô após integração com os dados da odometria. O jacobiano do modelo de observações é uma matriz identidade. Tal simplifica o cálculo do ganho de Kalman (Linha 6), bem como o do novo estado e respectiva matriz de covariância, nas Linhas 7 e 8.

3.2.2.2 Algoritmo de Map Matching

O algoritmo *Map Matching* procura a melhor correspondência entre o mapa M_C e a informação sensorial Z_C através da minimização do valor de uma função custo que representa uma modelação de um erro de correspondência. Essa minimização é realizada com recurso a um método numérico conhecido por RPROP (Riedmiller e Braun, 1993). Também é derivado um conjunto de valores que permite caraterizar a estimativa do erro de localização e orientação do robô obtido através da minimização anterior. Este valor é usado no processo de fusão sensorial apresentado anteriormente, permitindo aumentar a robustez geral do sistema.

Neste contexto, a pose absoluta do robô $X_V = [x_v\ y_v\ \theta_v]^T$ é obtida através da minimização do erro de correspondência $ErrMatch$, variando X_V. O robô obtém dos seus sensores $numC$ posições em obstáculos expressas no referencial do robô, $Z_C(k) = \{x_{C,i}(k), y_{C,i}(k) : i \in 1, \ldots, numC\}$, correspondentes aos contornos detectados. A partir destas variáveis é definido o problema de minimização (Equação 3.17).

$$\underset{X_V}{minimize}\ ErrMatch(X_V, Z_C) = \sum_{i}^{numL} \left| \begin{bmatrix} x_{C,i} \\ y_{C,i} \end{bmatrix} \right| err_i \left(D_i \left(\begin{bmatrix} x_{w,i}(X_V, Z_{C,i}) \\ y_{w,i}(X_V, Z_{C,i}) \end{bmatrix} \right) \right) \tag{3.17}$$

$$\begin{bmatrix} x_{w,i}(X_V, Z_{C,i}) \\ y_{w,i}(X_V, Z_{C,i}) \end{bmatrix} = \begin{bmatrix} x_V + \cos(\theta_V)x_{C,i} - \sin(\theta_V)y_{C,i} \\ y_V + \sin(\theta_V)x_{C,i} + \cos(\theta_V)y_{C,i} \end{bmatrix} \tag{3.18}$$

As funções $[x_{w,i}(X_V, Z_{C,i})\ y_{w,i}(X_V, Z_{C,i})]^T$ transformam uma posição relativa de uma medida ($Z_{C,i}$) numa posição absoluta em função da pose do robô (X_V). Esta transformação é definida na Equação 3.18. D_i é a função que calcula a distância entre uma posição da medida no referencial global e a célula do mapa de ocupação ocupada mais próxima. A função $err_i(.)$ penaliza desvios entre o observado pelos sensores e o mapa de ocupação. Tradicionalmente, em problemas de optimização utiliza-se o erro quadrático $err_i = \frac{1}{2}D_i^2$. É este o caso do algoritmo de *Map Matching* conhecido por ICP (Besl e McKay, 1992). Já Lauer et al. (2006) recorre a uma função custo mais robusta, uma vez que considera a presença de *outliers*. Esta é apresentada na Equação 3.19.

$$err_i(D_i) := 1 - \frac{L_c^2}{L_c^2 + D_i^2} \tag{3.19}$$

A função acima é parametrizada por L_c e limita o peso de *outliers* no processo de optimização. Na Figura 3.12, retirada de (Lauer et al., 2006), é feita a comparação entre a função de erro quadrática (a tracejado) e a Equação 3.19 (linha a cheio), usando um L_c de 250. Observa-se que estas equações são muito similares para erros (D_i) inferiores a L_c, mas para erros superiores a variação de $erri(.)$ vai diminuindo, estando limitada para erros elevados. Desta forma, a influência de *outliers* na optimização é limitada, tornando o processo de procura de correspondências mais robusto.

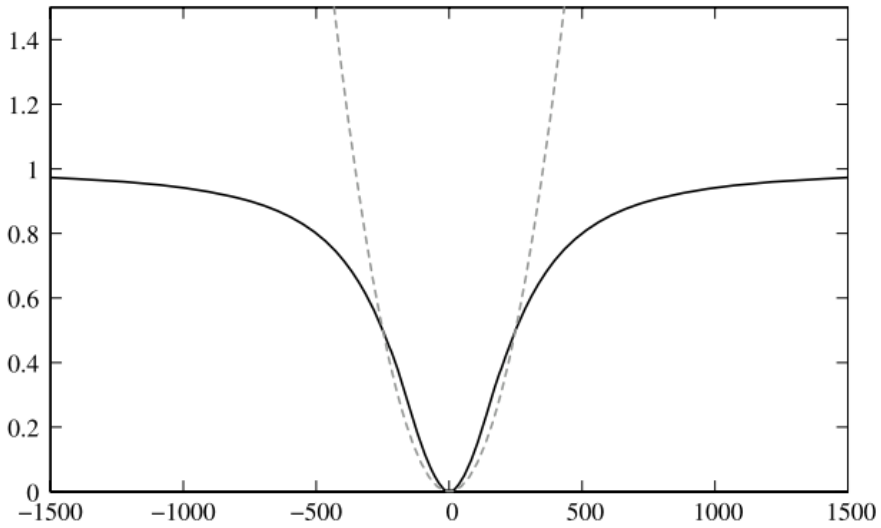


Figura 3.12: Comparação da função custo quadrática (tracejado) com a função custo utilizada por (Lauer et al., 2006) no algoritmo Perfect Match.

Na Equação 3.17 foi introduzida uma alteração face à função de custo original apresentada por Lauer et al. (2006). Aqui é adicionado um factor multiplicativo, que escala a contribuição de cada medida na função custo. Este factor corresponde à distância de uma medida $\left|[x_{C,i}\ y_{C,i}]^T\right|$. Com isto, será atribuído um peso maior a medidas mais distantes. Assim, é possível contrariar a característica radial dos sensores normalmente utilizados para adquirir contornos (como por exemplo um *laser range finder*). Num sensor radial, os objectos mais próximos têm um peso

maior que objectos com a mesma dimensão a uma distância maior do robô. Além disto, a precisão da medida da distância dada por um *laser range finder* não é significativamente afectada pela distância a que o objecto detectado se encontra.

Input : $\widehat{X}(k+1|k)$, $Z_C(k)$
Output: $Z_{Match}(k)$, $P_{Match}(k)$
1 **begin**
2 $X_{Init}(k) = \left[\frac{1}{M_{Res}} \ \frac{1}{M_{Res}} \ 1\right] * \widehat{X}(k+1|k)$
3 **for** *all measures* $Z_{C,i}(k)$ *in* $Z_C(k)$ **do**
4 $\quad Z_{Pix,i}(k) = \left[\frac{1}{M_{Res}} \ \frac{1}{M_{Res}}\right] * Z_{C,i}(k)$
5 **end**
6 $MaxError1 > MaxError2$
7 $Z_{C,F1}(k) = MaxErrorFilter(Z_{Pix}(k), X_{Init}(k), MaxError1)$
8 $X_{F1}(k) = OptimizeMatch(Z_{C,F1}(k), X_{Init}(k))$
9 $Z_{C,F2}(k) = MaxErrorFilter(Z_{C,F1}(k), X_{F1}(k), MaxError2)$
10 $X_{F2}(k) = OptimizeMatch(Z_{C,F2}(k), X_{F1}(k))$
11 $P_{Match}(k) = CovarianceCalculations(Z_{C,F2}(k), X_{F2}(k))$
12 $Z_{Match}(k) = [M_{Res} \ M_{Res} \ 1] * X_{F2}(k)$
13 **end**

Algorithm 6: Map Matching.

O Algoritmo 6 apresenta os traços gerais da solução de *Map Matching* proposta. Tem como entradas a pose do robô após serem processados os dados da odometria ($X(k+1|k)$), bem como um conjunto de posições relativas referentes aos objetos detectados pelos sensores ($Z_C(k)$).

De forma a optimizar o peso computacional, a pose do robô e as medidas dos sensores são escaladas em função da resolução (M_{Res}) do Mapa de Ocupação (M_C). Desta forma, uma posição no referencial global corresponde aos índices da sua respectiva célula do mapa. Neste sentido, o algoritmo começa por escalar a pose do robô (Linha 2) e das medidas (Linha 4), finalizando com a conversão do resultado da optimização para o referencial absoluto (Linha 12).

Entre a Linha 6 e a Linha 10, é apresentada a arquitectura geral da solução. Esta consiste numa pré-filtragem dos *outliers* dos dados sensoriais (Linha 7), seguida da optimização da Função 3.17 (Linha 8). Esta sequência é posteriormente repetida usando uma tolerância mais apertada (Linhas 9 e 10). A inovação em relação ao algoritmo original consiste na aplicação de dois filtros intercalados pelo algoritmo de *matching* (*Perfect Match*), em que a tolerância da filtragem inicial é superior à filtragem final.

A filtragem acima descrita, abordada posteriormente com mais detalhe, é baseada na rejeição dos pontos com erro associado acima de um dado *threshold*, indicando a presença de objetos/pessoas externas ao mapa. Numa primeira fase, rejeitam-se medidas com um valor de distância máxima menos exigente (por exemplo $MaxError1 = 1m$), de forma a acomodar o erro de inicialização provocado pela odometria. Posteriormente, é executado o algoritmo de otimização RPROP. Neste ponto, o erro de inicialização é diminuído, o que leva a que o erro de distância observado nas medidas seja maioritariamente influenciado pela presença de *outliers*. Nesta segunda corrida

utiliza-se um valor de distância mais exigente (por exemplo $MaxError2 = 0.01m$). O resultado final é obtido após nova execução do algoritmo de optimização RPROP e tem, uma precisão bastante superior. Note-se que se fosse efetuada a optimização apenas uma vez e utilizando o valor de $MaxError2$ o resultado final não seria o mesmos pois seriam rejeitadas muitas medidas que na realidade não correspondiam a *outliers*. Assim a primeira iteração permite-nos ter uma estimativa mais próxima do valor real, rejeitando apenas *outliers* evidentes, e depois, no segundo passo, sermos mais exigentes na rejeição dos *outliers*.

Na Linha 12, é calculada a matriz de covariância $P_{Match}(k)$, que procura quantificar a incerteza associada ao cálculo da pose do robô. A quantificação desta incerteza baseia-se numa heurística que utiliza o valor da segunda derivada da função custo 3.17 para se caracterizar estatisticamente a estimativa obtida (Lauer et al., 2006), resultando numa matriz de covariância $P_{Match}(k)$ que em função da pose final do robô (X_{F2}) e dos dados do laser ($Z_{C,F2}$).

De seguida serão descritos os Algoritmos "MaxErrorFilter", "OptimizeMatch"e "ProcessSecondDerivative"utilizados no Algoritmo 6 e o cálculo das derivadas utilizado no Algoritmo "OptimizeMatch".

Optimização baseada no RPROP

O Algoritmo 7 é o responsável por resolver o problema de optimização definido pela função custo 3.17. Tem como entradas a pose inicial ($X(k)$) e um conjunto de posições relativas aos sensores do robô ($Z(k)$) e como saída a pose $X_{Match}(k)$, que minimiza a função custo 3.17. Esta optimização recorre ao RPROP apresentado em (Riedmiller e Braun, 1993).

O RPROP é um algoritmo de minimização baseado apenas no sinal das primeiras derivadas da função de custo e foi concebido para problemas de optimização não lineares. Este calcula iterativamente a variação de X_{Match}, ΔX, minimizando $ErrMatch$ em função do sinal das suas derivadas parciais (∇).

O ciclo principal do algoritmo começa por recorrer à função "UpdateGrads"para calcular as derivadas parciais $\nabla = \left[\frac{\partial ErrMatch(.)}{\partial x_V} \; \frac{\partial ErrMatch(.)}{\partial y_V} \; \frac{\partial ErrMatch(.)}{\partial \theta_V}\right]^T$, na Linha 8. De seguida são processados sequencialmente os três elementos do vector $X_{Match} = [x_{Match}, \; y_{Match}, \; \theta_{Match}]$ (Linha 9). Para cada j, o passo $\Delta X[j]$ é calculado em função do sinal de $\nabla[j]$ e $\nabla_{last}[j]$. Se o sinal destes for igual (Linha 11), o valor absoluto do passo $\Delta X[j]$ sobe sendo multiplicado por η_+ (Linha 12). Se os respectivos sinais forem diferentes, isso significa que se passou pelo mínimo local e $\Delta X[j]$ desce sendo multiplicado por η_- (Linha 15). Os valores de η_+ e η_- são constantes, sendo utilizados como exemplo os recomendados por Riedmiller e Braun (1993). Por fim, o valor de $\Delta X[j]$ é somado ou subtraído a $X_{Match}(k)[j]$ em função do sinal de $\nabla[j]$ (Linha 18 até Linha 21).

Por último, na Linha 28, é analisado o critério de paragem do algoritmo. No trabalho apresentado por Lauer et al. (2006) é definido um número fixo de iterações, mais concretamente 10 iterações. Utilizando um número fixo de 10 iterações, verificava-se uma pequena oscilação na resposta do algoritmo. Foi também verificado que um número superior de iterações diminuía esta oscilação. Neste sentido, foi adicionado outro critério de paragem, avaliando a diferença de translação (Linha 26) e rotação (Linha 27) entre duas iterações consecutivas. Se o valor absoluto desta

Input : $Z(k), X(k)$
Output: $X_{Match}(k)$

begin
  $\Delta X = \begin{bmatrix} \Delta x_{init} & \Delta y_{init} & \Delta\theta_{init} \end{bmatrix}^T$
  $\eta_+ = 1.2$
  $\eta_- = 0.5$
  $\nabla_{last} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$
  $X_{Match}(k) = X(k)$
  for *i=0 to MaxIterationNum* **do**
    $\nabla = \text{UpdateGrads}(Z(k), X_{Match})$;
    for *j=1 to 3* **do**
      if $\nabla[j] \neq 0$ **then**
        if $\nabla_{last}[j] * \nabla[j] > 0$ **then**
          $\Delta X[j] = \Delta X[j] * \eta_+$
        else
          if $\nabla_{last}[j] * \nabla[j] < 0$ **then**
            $\Delta X[j] = \Delta X[j] * \eta_-$
          end
        end
        if $\nabla[j] > 0$ **then**
          $X_{Match}(k)[j] = X_{Match}(k)[j] - \Delta X[j]$
        else
          $X_{Match}(k)[j] = X_{Match}(k)[j] + \Delta X[j]$
        end
      end
    end
    $\nabla_{last} = \nabla$
    $\Delta Trans = \sqrt{\Delta X[1]^2 + \Delta X[2]^2}$
    $\Delta Rot = |\Delta X[3]|$
    if $((\Delta Trans < \Delta Trans_{min})$ *and* $(\Delta Rot < \Delta Rot_{min}))$ *or* $((\nabla[1], \nabla[2], \nabla[3] = 0)$ **then**
      stop
    end
  end
end

Algorithm 7: RPROP Optimization.

diferença for inferior ao *threshold* definido, o processo de optimização é parado. Note-se que foi utilizado um valor de 0.1 para inicializar o passo de optimização (Δx_{init}, Δy_{init} e $\Delta\theta_{init}$). Contudo, utilizando o critério de paragem anterior, o valor destes parâmetros deixa de ser tão crítico, interferindo no número de iterações executadas mas não na qualidade do resultado final.

O Algoritmo 8 calcula as derivadas parciais da Função 3.17 (∇) em ordem a $X_V = [x_V\ y_V\ \theta_V]^T$ e tem como entradas os dados dos sensores $Z_C(k)$ e a pose X_V. Estas derivadas são recalculadas a

Input : $Z_C = \left\{ [x_{C,i}\ y_{C,i}]^T : i \in 1,\ \ldots,\ numL \right\},\ X_V = [x_V\ y_V\ \theta_V]^T$
Output: $\nabla = \left[\frac{\partial ErrMatch(.)}{\partial x_V}\ \frac{\partial ErrMatch(.)}{\partial y_V}\ \frac{\partial ErrMatch(.)}{\partial \theta_V} \right]^T$

1 **begin**
2 $\nabla = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$
3 **for** *all relative meassures* $Z_{C,i}$ *in* $Z_C(k)$ **do**
4 $\begin{bmatrix} x_{w,i} \\ y_{w,i} \end{bmatrix} = \begin{bmatrix} x_V + \cos(\theta_V)x_{C,i} - \sin(\theta_V)y_{C,i} \\ y_V + \sin(\theta_V)x_{C,i} + \cos(\theta_V)y_{C,i} \end{bmatrix}$
5 $D_i = M_{Dist}[round(x_{w,i})\ round(y_{w,i})]$
6 $\frac{\partial D_i}{\partial x_w} = M_{GradX}[round(x_{w,i})\ round(y_{w,i})]$
7 $\frac{\partial D_i}{\partial y_w} = M_{GradY}[round(x_{w,i})\ round(y_{w,i})]$
8 $\frac{\partial err_i}{\partial D_i} = \frac{L_c^2 \cdot D_i}{\left(L_c^2 + D_i^2\right)^2}$
9 $\nabla_i = \frac{\partial err_i}{\partial D_i} \cdot \begin{bmatrix} \frac{\partial D_i}{\partial x_w} \\ \frac{\partial D_i}{\partial y_w} \\ \frac{\partial D_i}{\partial x_w} \cdot (-\sin(\theta_V)x_{C,i} - \cos(\theta_V)y_{C,i}) + \frac{\partial D_i}{\partial y_w} \cdot (\cos(\theta_V)x_{C,i} - \sin(\theta_V)y_{C,i}) \end{bmatrix}$
10 $\nabla = \nabla + \sqrt{x_{C,i}^2 + y_{C,i}^2} \cdot \nabla_i$
11 **end**
12 **end**

Algorithm 8: UpdateGrads.

cada iteração do algoritmo de optimização e são definidas pelas seguintes equações:

$$\frac{\partial ErrMatch(.)}{\partial X_V} = \sum_i^{numL} \left\| \begin{bmatrix} x_{C,i} \\ y_{C,i} \end{bmatrix} \right\| \frac{\partial err_i(.)}{\partial X_V} \tag{3.20}$$

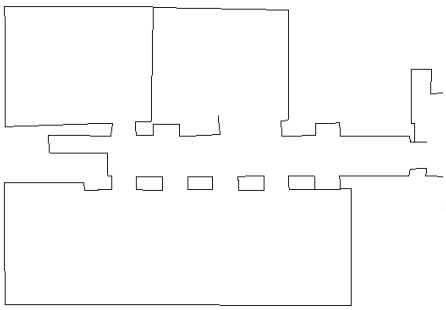
$$\frac{\partial err_i(.)}{\partial x_V} = \frac{\partial err_i(.)}{\partial D_i(.)} \left(\frac{\partial D_i(.)}{\partial x_w(.)} \frac{\partial x_w(.)}{\partial x_V} + \frac{\partial D_i(.)}{\partial y_w(.)} \frac{\partial y_w(.)}{\partial x_V} \right) = \frac{L_c^2 \cdot D_i}{\left(L_c^2 + D_i^2\right)^2} \cdot \frac{\partial D_i(.)}{\partial x_w(.)} \tag{3.21}$$

$$\frac{\partial err_i(.)}{\partial y_V} = \frac{\partial err_i(.)}{\partial D_i(.)} \left(\frac{\partial D_i(.)}{\partial x_w(.)} \frac{\partial x_w(.)}{\partial y_V} + \frac{\partial D_i(.)}{\partial y_w(.)} \frac{\partial y_w(.)}{\partial y_V} \right) = \frac{L_c^2 \cdot D_i}{\left(L_c^2 + D_i^2\right)^2} \cdot \frac{\partial D_i(.)}{\partial y_w(.)} \tag{3.22}$$

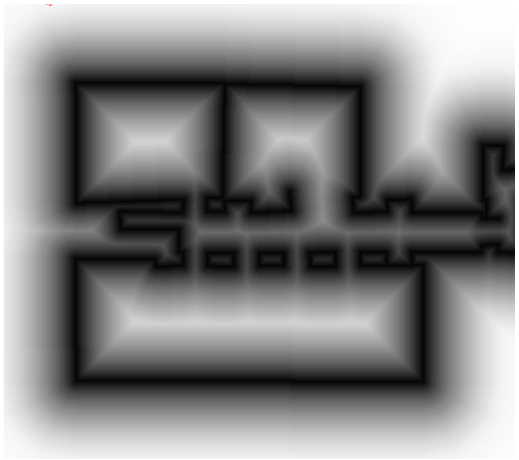
$$\begin{aligned} \frac{\partial err_i(.)}{\partial \theta_V} &= \frac{\partial err_i(.)}{\partial D_i(.)} \left(\frac{\partial D_i(.)}{\partial x_w(.)} \frac{\partial x_w(.)}{\partial \theta_V} + \frac{\partial D_i(.)}{\partial y_w(.)} \frac{\partial y_w(.)}{\partial \theta_V} \right) = \\ &= \frac{L_c^2 \cdot D_i}{\left(L_c^2 + D_i^2\right)^2} \left(\frac{\partial D_i}{\partial x_w}(-\sin(\theta_V)x_{C,i} - \cos(\theta_V)y_{C,i}) + \frac{\partial D_i}{\partial y_w}(\cos(\theta_V)x_{C,i} - \sin(\theta_V)y_{C,i}) \right) \end{aligned} \tag{3.23}$$

Para cada medida $Z_{C,i}$, são calculadas as respectivas coordenadas no referencial global já com as unidades convertidas para índices das células da matriz de representação do mapa (Linha 4). Na linha 5, é calculada a distância (D_i) à célula do mapa de ocupação (M_C) ocupada mais próxima através da matriz M_{Dist}, e nas Linhas 6 e 7 são calculadas as derivadas de D_i em relação a x_w e

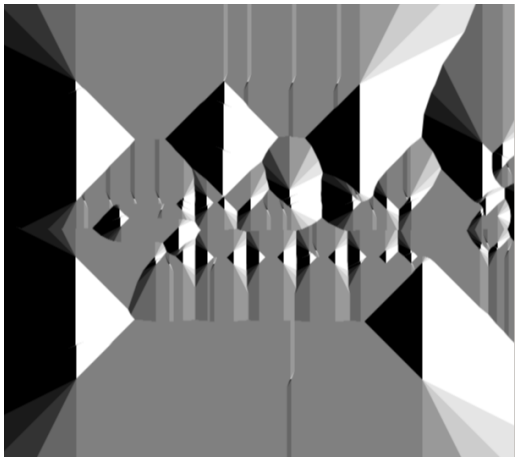
y_w, respectivamente. Na Linha 9 é calculada a contribuição ∇_i da medida $Z_{C,i}$ para o cálculo do gradiente, aplicando as Equações 3.21, 3.22 e 3.23. Por fim, na Linha 10 é, aplicado o somatório apresentado na Equação 3.20.



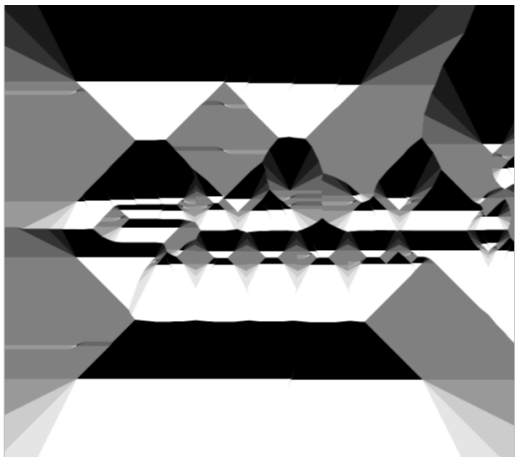
(a) Mapa de ocupação (M_C).



(b) Mapa de Distância (M_{Dist}).



(c) Derivada do Mapa de Distância ao longo do eixo horizontal (M_{GradX}).



(d) Derivada do Mapa de Distância ao longo do eixo vertical (M_{GradY}).

Figura 3.13: Exemplo das *lookup tables* utilizadas pelo algoritmo Perfect Match.

Como referido anteriormente, a função $D_i(.)$ corresponde à distância entre uma posição no mundo e a célula do mapa de ocupação (M_C) ocupada mais próxima. Esta função pode ser pré-calculada a partir de M_C e guardada numa *lookup table* (M_{Dist}), designada de mapa de distâncias. O que faz com que em termos computacionais $D_i(.)$ consista numa leitura de um elemento de uma matriz. Como a optimização é baseada em gradientes, há mais duas matrizes pré-processadas a partir de M_{Dist}, uma correspondente ao gradiente $D_i(.)$ ao longo do eixo x (M_{GradX}) e outra correspondente ao gradiente ao longo do eixo y (M_{Grady}). Para gerar estas matrizes, recorreu-se às funções de um projecto *open-source* conhecido por openCV. M_{Dist} obtém-se aplicando a transformada de distância na matriz binária M_C (o algoritmo utilizado está descrito em (Borgefors e Gunilla, 1986)). M_{GradX} e M_{GradY} obtêm-se aplicando um algoritmo conhecido por "Sovel",

muito utilizado para calcular derivadas em imagens. M_{GradX} e M_{GradY} são gerados a partir do cálculo das derivadas horizontal e vertical de M_{Dist}, respectivamente.

Na Figura 3.13a, temos um exemplo de um mapa de ocupação (M_C) e, na Figura 3.13b, podemos ver o correspondente mapa de distâncias (M_{Dist}). Na Figura 3.13b, a intensidade de cada pixel é proporcional ao valor de distância ao pixel em M_C ocupado mais próximo. Nas Figuras 3.13c e 3.13d, podemos ver M_{GradX} e M_{GradY} geradas a partir do M_{Dist} da Figura 3.13b.

Filtragem de outliers

Input : $Z_I = \left\{[x_{C,i}\ y_{C,i}]^T : i \in 1,\ \ldots,\ numI\right\}$, $X_V = [x_V\ y_V\ \theta_V]^T$, $MaxErrordist$
Output: $Z_O = \left\{[x_{O,i}\ y_{O,i}]^T : i \in 1,\ \ldots,\ numO\right\}$

```
1  begin
2  for all relative meassures Z_{I,i} in Z_I(k) do
3  |   [x_{w,i}; y_{w,i}] = [x_V + cos(θ_V)x_{C,i} − sin(θ_V)y_{C,i}; y_V + sin(θ_V)x_{C,i} + cos(θ_V)y_{C,i}]
4  |   D_i = M_Dist[round(x_{w,i}) round(y_{w,i})]
5  |   if D_i < MaxErrordist then
6  |   |   add Z_{I,i} to Z_O
7  |   |   inc numO
8  |   end
9  end
10 end
```

Algorithm 9: MaxErrorFilter.

O Algoritmo 9 pré-processa os dados dos sensores de forma a remover as medidas a objectos não mapeados (*outliers*). Tem como entradas um conjunto de pontos adquiridos pelo *laser scanner* (Z_I), uma pose referente à estimativa da pose absoluta do robô (X_V) e $MaxErrordist$ que corresponde à distância máxima entre a posição de uma medida e a posição no mapa de ocupação M_C da célula ocupada mais próxima. Tem como saída Z_O que corresponde a um conjunto de medidas filtradas cujo número $numO$ é obviamente inferior ou igual ao número total de medidas $numI$.

De seguida apresenta-se o Algoritmo 9. Para cada medida $Z_{I,i}$ é calculada a sua respectiva posição absoluta em função de X_V aplicando a Equação 3.18 (Linha 3); De seguida recorrendo à matriz M_{Dist} é obtida a distância D_i à célula ocupada mais próxima de M_C (Linha 4); Se a distância D_i for inferior a $MaxErrordist$ (Linha 5) $Z_{I,i}$ é adicionado Z_O (Linha 6 e 7).

A Figura 3.14 dá uma ideia da visual do efeito deste filtro. Associando este filtro ao filtro integrado na função custo do PM (Figura 3.14) aumenta-se a precisão e robustez da estimativa da pose removendo medidas correspondentes a objectos não mapeados que apenas degradam a resposta do algoritmo.

Cálculo das covariâncias

Nem sempre é possível obter uma estimativa fiável da pose do robô a partir da correspondência entre as medidas e o mapa do ambiente. Entre possíveis causas deste fenómeno apresentam-se a falta de estrutura do meio envolvente, por exemplo, o robô navegar ao longo de um extenso

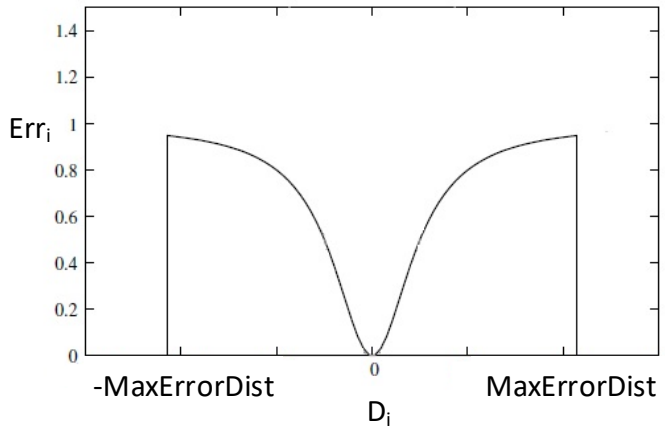


Figura 3.14: Função custo 2.1 com *threshold MaxErrorDist*.

corredor, ou a existência de muitos *outliers* que levam a um número reduzido de medidas válidas. Em casos como estes, o mínimo local de $ErrMatch$ (Função 3.17) não será bem definido numa das dimensões de X_V. Assim, o Algoritmo 10 quantifica a incerteza da pose X_V em função dos dados sensoriais Z_C (entradas) a partir do cálculo da matriz de covariância P. O Filtro de Kalman apresentado no Algoritmo 5 funde a estimativa X_V e respetiva covariânvia P com os dados da odometria, de forma a obter uma estimativa da pose absoluta do robô mais robusta.

Lauer et al. (2006) propõe recorrer à análise da segunda derivada da função custo $ErrMatch$ para modelar o grau de incerteza:

$$\frac{\partial^2 ErrMatch(.)}{\partial^2 X_V} = \sum_{i}^{numL} \frac{\partial^2 err_i(.)}{\partial^2 X_V} \tag{3.24}$$

No modelo proposto por Lauer et al. (2006) são feitas algumas simplificações, nomeadamente:

- Considerar a função de erro quadrática $\frac{1}{2}\left(\frac{D_i}{L_c}\right)^2$ em vez da Função 3.19 e ignorar medidas em que o seu correspondente D_i seja superior a $\frac{L_c}{\sqrt{3}}$. A função 3.19 não é completamente positiva definida, apenas no intervalo $\left(\frac{L_c}{\sqrt{3}}, -\frac{L_c}{\sqrt{3}}\right)$, e a análise da curvatura pode falhar. Isto acontece em pontos na Função 3.19 em que a segunda derivada tem um valor elevado e estes não correspondem a um mínimo local.

- Assumir que as segundas derivadas parciais $\frac{\partial^2 D_i}{\partial^2 x_w}$ e $\frac{\partial^2 D_i}{\partial^2 y_w}$ são nulas. Esta aproximação é válida para as situações em que o Mapa de ocupação (M_C) é maioritariamente composto por linhas rectas, como é o caso típico de o interior de um edifício.

O Algoritmo 10 apresenta os detalhes do cálculo da matriz de covariância P_{Match}. Na Linha 2 é apresentado o cálculo da segunda derivada de $err_i(.)$ em ordem a D_i. De seguida é calculada, para cada valor de $Z_{I,i}$, a contribuição das segundas derivadas $\frac{\partial^2 err_i(.)}{\partial^2 x_V}$, $\frac{\partial^2 err_i(.)}{\partial^2 y_V}$ e $\frac{\partial^2 err_i(.)}{\partial^2 \theta_V}$ nas Linhas 9, 11 e 19, respectivamente. Estes cálculos têm em consideração as simplificações acima enunciadas. Da Linha 20 à Linha 22 demonstra-se a implementação do somatório definido na Equação

3.24. Da Linha 25 à Linha 27 é definida a matriz P_{Match}. Esta é uma matriz diagonal em que as respectivas variâncias são modeladas como o inverso da segunda derivada de $ErrMatch$ em ordem ao respectivo estado multiplicado por uma constante. O valor destas constante deve ser ajustado devidamente conforme a qualidade por exemplo dos sensores e da odometria. Este processo é uma heurística que nos permite fundir resultados obtidos por um processo não estatístico (o algoritmo PM) com dados obtidos por um processo estatístico (o modelo de odometria).

Esta proposta de quantificação de incerteza apresenta algumas limitações e pretende-se refinar este modelo como trabalho futuro. A qualidade da estimação da incerteza é um factor crítico na robustez nos algoritmos de fusão sensorial. O modelo proposto tem uma qualidade aceitável se o mapa de ocupação utilizado for maioritariamente composto por linhas horizontais e verticais, não se dando valor, até ao momento, à correlação entre os diferentes elementos de X_V para o cálculo matriz de covariância P_{Match}.

Neste sentido, como trabalho futuro pretende-se refinar a definição de P_{Match} de forma a incluir as covariâncias entre os diferentes elementos de X_V, removendo as simplificações acima mencionadas. É de realçar um caso específico em que a simplificação de considerar nulas $\frac{\partial^2 D_i}{\partial^2 x_w}$ e $\frac{\partial^2 D_i}{\partial^2 y_w}$ tem um impacto significativo: no Futebol Robótico quando um robô está no centro do campo grande parte dos seus dados sensoriais correspondem à detecção do círculo existente nesse local. Nessa situação, apesar de a orientação do robô não estar bem definida, a variância da orientação obtida pelo Algoritmo 10 vai gerar um valor baixo devido à simplificação usada, o que pode levar a falhas no sistema de localização. Também se pretende analisar alternativas à segunda derivada na modelação da incerteza.

3.2.2.3 Localização Global

Até este ponto os Algoritmos de localização apresentados fazem parte uma classe de soluções chamada de Algoritmos de *Pose Tracking*. Nesta secção será abordada a questão da Localização Global, *i. e.*,a estimativa da pose do robô sem nenhum conhecimento *a priori*. Apesar da grande vantagem que é o baixo peso computacional dos algoritmos de *Pose Tracking*, estes dependem da existência de uma estimativa anterior da pose do robô. Assim, em momentos em que a pose não está disponível devido a: o robô ser deslocado com o sistema de localização desligado (situação de rapto), ocorrência de uma falha no sistema sensorial gerada por um número elevado de *outliers*, derrapagem das rodas, entre outros, o algoritmo não consegue convergir para o valor correcto, dada a discrepância entre a estimativa do valor inicial da pose e o valor real da mesma.

A Figura 3.15 apresenta em traços gerais a arquitectura da proposta para a Localização Global. O sistema apresenta dois modos de funcionamento: *Pose Tracking* e *Global Localization*. O sistema opera normalmente no modo *Pose Tracking*. Em simultâneo, o módulo *LostDetector* analisa a existência de falhas no sistema de localização. Quando uma falha é detectada, o sistema passa para o modo *Global Localization*, dando-se início a um processo de estimação da pose absoluta sem recorrer a conhecimento prévio da pose do robô. Terminado o processo de localização global, o sistema de *Pose Tracking* é reinicializado com a pose absoluta estimada.

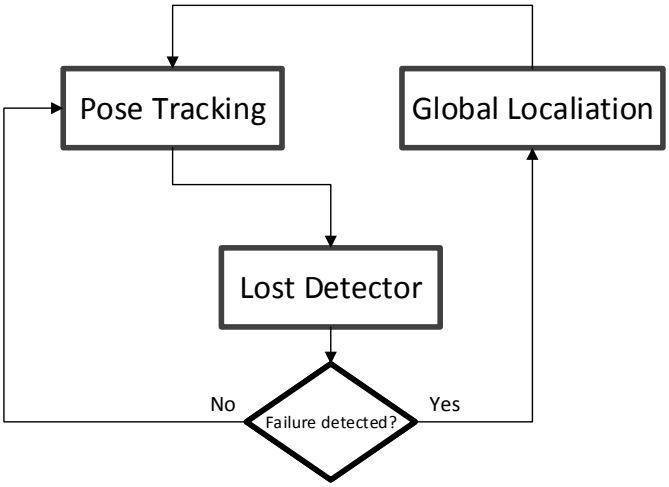


Figura 3.15: Arquitectura do sistema de localização. Propõe-se um sistema hibrido em que o sistema recorre um algoritmo de Pose Tracking durante o modo normal de operação. Se for detectada uma falha o sistema passa a utilizar o módulo Global Localization, (computacionalmente mais pesado) de forma a recuperar a pose a absoluta e reinicializar o módulo de Pose Tracking.

O módulo *Pose Tracking* comporta os algoritmos apresentados na Secção 3.2.2.2. O módulo *LostDetector* explora redundâncias entre a orientação absoluta obtida através do *Map Matching* e uma bússola magnética. Já o módulo *Global Localization* procura a pose absoluta executando diversas instâncias do módulo *Pose Tracking* com diferentes poses iniciais.

Módulo LostDetector

O módulo *LostDetector* explora a redundância entre a orientação absoluta medida por uma bussola magnética instalada no robô e a orientação retornada pelo algoritmo de *Pose Tracking*. A diferença entre estas duas grandezas é filtrada por um filtro passa-baixo ($\Delta\theta_{MagF}$) e, caso $\Delta\theta_{MagF}$ seja superior a α_{Lost} durante um intervalo de tempo superior a t_{Lost}, é sinalizada uma falha no sistema de localização.

Os valores de α_{Lost} e t_{Lost} são parâmetros do sistema e são escolhidos em função da qualidade da medida da orientação absoluta obtida a partir da bússola e da estrutura do mapa de ocupação. Para mapas em que a disposição dos objectos é tendencialmente ortogonal, como por exemplo no caso do futebol robótico, em que o mapa é maioritariamente constituído por linhas horizontais e verticais, utiliza-se um valor de 45° para α_{Lost} e um valor de 4s para t_{Lost}.

As bússolas magnéticas são sensores susceptíveis de serem afectados por interferências externas devido à possível existência de campos magnéticos provocados, por exemplo, pela presença de objetos metálicos, não sendo possível a utilização deste sensor em alguns ambientes. Como trabalho futuro pretende-se analisar a possibilidade de recorrer à redundância entre a medida da velocidade do robô obtida através da odometria (*encoders*, por exemplo) e pelo sistema de localização de forma a aumentar a robustez deste módulo.

Módulo Global Localization

Este módulo procura o mínimo local da Função 3.17, que corresponde à verdadeira pose do robô, de forma a reinicializar corretamente o sistema de *Pose Tracking*. O peso computacional deste módulo depende do tamanho do mapa de referência, pois é efectuada uma procura quase exaustiva ao longo de todo o mapa. Nesse sentido, esta operação só é iniciada quando é detectada uma falha no sistema de localização.

São consideradas a seguintes suposições:

- Não existe informação relativamente à posição actual do robô. Considera-se um conjunto de posições possíveis distribuídas na área de navegação do robô numa das quais o robô se encontra relativamente próximo;
- A orientação absoluta é estimada através da medida fornecida pela bússola magnética;
- Devido ao ruído presente nas medidas adquiridas pelos sensores, o mínimo global da função 3.17 não corresponde necessariamente à verdadeira pose do robô;
- Um dos mínimos locais da função 3.17 corresponde à verdadeira pose do robô.

O processo de localização global arranca com um determinado conjunto inicial de hipóteses. Para cada uma é executada uma instância do algoritmo de *Pose Tracking*. Com a chegada de nova informação sensorial, o número de hipóteses vai diminuindo através de um esquema de eliminação e fusão, até que sobre uma hipótese válida passível de ser assumida como a verdadeira pose do robô. O critério para eliminar uma hipótese é o mesmo que despoleta o processo da localização global, ou seja, para cada hipótese existe uma instância do módulo *LostDetector* que analisa a divergência entre a orientação dada pela bússola e pelos sensores utilizados no sistema de localização. O critério de fusão entre duas hipóteses está relacionado com a sua distância euclideana e diferença de ângulos.

De seguida descreve-se o o processo de localização global:

1. Ao longo do mapa são distribuídas múltiplas hipóteses igualmente espaçadas entre si. Para o valor de orientação é escolhido o valor fornecido pela bussola magnética;
2. Cada vez que novos dados sensoriais estão disponíveis, as múltiplas hipóteses são continuamente iteradas da seguinte forma:
   (a) O valor de cada hipótese é atualizado com a informação dos *encoders* aplicando o modelo cinemático definido pela Equação 3.2, e com isto obtêm-se $X^{Hyp}_{Odo,i}$.
   (b) Para cada $X^{Hyp}_{Odo,i}$, juntamente com os dados dos sensores, é executado o algoritmo de *Map Matching* apresentado como Algoritmo 7. Com isto obtém-se o novo valor $X^{Hyp}_{Match,i}$ de cada hipótese;
   (c) Após o processamento dos dados sensoriais as hipóteses $X^{Hyp}_{Match,i}$ classificadas como inválidas pela sua respectiva instância do módulo *LostDetector* são eliminadas;

(d) As hipóteses repetidas são fundidas. Considera-se hipóteses repetidas aquelas cuja a distância euclediana é menor que d_{Merge} e a diferença angular é menor α_{Merge};

3. Considera-se que o algoritmo estabiliza quando o número de hipóteses eliminadas e fundidas é inferior a Δh durante um intervalo de tempo t_{Conv}. É considerado que após o algoritmo estabilizar, cada hipótese "sobrevivente"se encontra num minimo local;

4. No fim do processo estabilizar, inicia-se um processo de atribuição de pontuação às hipóteses restantes. A pontuação de uma hipótese é incrementada quando o seu custo (Equação 3.17) for pelo menos duas vezes menor do que o das outras hipóteses.

5. O processo termina quando uma das hipóteses atingir a pontuação h_{score} e o valor dessa hipótese será utilizado para reinicializar o módulo *Pose Tracking*.

O peso computacional do módulo *Global Localization* é cerca de n vezes superior ao módulo *Pose Tracking*, em que n corresponde ao número de hipóteses "vivas". Esse número vai diminuindo com o decorrer do algoritmo e, consequentemente, o peso computacional. As instâncias do *Pose Tracking* utilizadas em cada hipótese são configuradas de forma a limitar o seu peso computacional, sacrificando um pouco a precisão do resultado do *matching*. Assim, o número de iterações máximo executado por instância do *Pose Tracking* é configurado com um valor relativamente baixo - 10 iterações. O valor dos parâmetros utilizados nesta abordagem foram obtidos experimentalmente e são transversais a dois casos de testes distintos (que serão apresentados na Secção 5.2.3). A distância (d_{Merge}) e erro de ângulo máximo (α_{Merge}) para fundir duas hipóteses são $0.1m$ e 10°. Em relação ao parâmetros relativos à convergência do algoritmo são usados os valores de 4s para t_{Conv}, 10 para Δh e 9 para h_{score}.

Como não existe garantia de que a informação sensorial que está ao alcance do robô numa determinada pose seja distinta o suficiente para que seja possível determinar a sua posição absoluta, o robô deve mover-se de forma a obter mais informação acerca do meio envolvente. No entanto, nesta situação, este movimento é realizado sem acesso a dados de localização. Este tipo de navegação nem sempre é admissível, nomeadamente no caso típico de aplicações industriais, nas quais geralmente os robôs são de grandes dimensões e exige-se que estes tenham um comportamento determinístico, isto é, que se desloquem apenas ao longo de trajetórias pré-definidas. No entanto, em aplicações não industriais em que a sensorização assegura por si só a segurança do deslocação do robô, este pode mover-se de forma a determinar a sua posição absoluta. Por exemplo: no caso do futebol robótico, estes podem deslocar-se de forma aleatória, evitando transpor uma linha branca ou, no caso de robôs de serviços, estes podem deslocar-se no sentido da área que estiver livre.

Input : $Z_C = \left\{ [x_{C,i}\ y_{C,i}]^T : i \in 1,\ \ldots,\ numI \right\},\ X_V = [x_V\ y_V\ \theta_V]^T$
Output: P_{Match}

begin
$\frac{\partial^2 err_i}{\partial^2 D_i} = \frac{1}{L_c^2}$
$\frac{\partial^2 ErrMatch(.)}{\partial^2 x_V} = 0,\ \frac{\partial^2 ErrMatch(.)}{\partial^2 y_V} = 0,\ \frac{\partial^2 ErrMatch(.)}{\partial^2 \theta_V} = 0$
for *all relative meassures* $Z_{I,i}$ *in* $Z_I(k)$ **do**
  $\begin{bmatrix} x_{w,i} \\ y_{w,i} \end{bmatrix} = \begin{bmatrix} x_V + \cos(\theta_V)x_{C,i} - \sin(\theta_V)y_{C,i} \\ y_V + \sin(\theta_V)x_{C,i} + \cos(\theta_V)y_{C,i} \end{bmatrix}$
  $D_i = M_{Dist}[round(x_{w,i})\ round(y_{w,i})]$
  if $D_i < \frac{L_c}{\sqrt{3}}$ **then**
    $\frac{\partial D_i}{\partial x_w} = M_{GradX}[round(x_{w,i})\ round(y_{w,i})]$
    $\frac{\partial^2 err_i(.)}{\partial^2 x_V} = \frac{\partial^2 err_i(.)}{\partial^2 D_i}\left(\frac{\partial D_i}{\partial x_w}\right)^2$
    $\frac{\partial D_i}{\partial y_w} = M_{GradY}[round(x_{w,i})\ round(y_{w,i})]$
    $\frac{\partial^2 err_i}{\partial^2 y_V} = \frac{\partial^2 err_i(.)}{\partial^2 D_i}\left(\frac{\partial D_i}{\partial y_w}\right)^2$
    $\frac{\partial err_i}{\partial D_i} = \frac{D_i}{L_c^2}$
    $\frac{\partial D_i}{\partial x_w} = M_{GradX}[round(x_{w,i})\ round(y_{w,i})]$
    $\frac{\partial D_i}{\partial y_w} = M_{GradY}[round(x_{w,i})\ round(y_{w,i})]$
    $\frac{\partial x_w}{\partial \theta_V} = -\sin(\theta_V)x_{C,i} - \cos(\theta_V)y_{C,i}$
    $\frac{\partial y_w}{\partial \theta_V} = \cos(\theta_V)x_{C,i} - \sin(\theta_V)y_{C,i}$
    $\frac{\partial^2 x_w}{\partial^2 \theta_V} = -\cos(\theta_V)x_{C,i} + \sin(\theta_V)y_{C,i}$
    $\frac{\partial^2 y_w}{\partial^2 \theta_V} = -\sin(\theta_V)x_{C,i} - \cos(\theta_V)y_{C,i}$
    $\frac{\partial^2 err_i}{\partial^2 \theta_V} = \frac{\partial^2 err_i}{\partial^2 D_i}\left(\frac{\partial D_i}{\partial x_w}\frac{\partial x_w}{\partial \theta_V} + \frac{\partial D_i}{\partial y_w}\frac{\partial y_w}{\partial \theta_V}\right)^2 + \frac{\partial err_i}{\partial D_i}\left(\frac{\partial D_i}{\partial x_w}\frac{\partial^2 x_w}{\partial^2 \theta_V} + \frac{\partial D_i}{\partial y_w}\frac{\partial^2 y_w}{\partial^2 \theta_V}\right)$
    $\frac{\partial^2 ErrMatch}{\partial^2 x_V} = \frac{\partial^2 ErrMatch}{\partial^2 x_V} + \frac{\partial^2 err_i}{\partial^2 x_V}$
    $\frac{\partial^2 ErrMatch}{\partial^2 y_V} = \frac{\partial^2 ErrMatch}{\partial^2 y_V} + \frac{\partial^2 err_i}{\partial^2 y_V}$
    $\frac{\partial^2 ErrMatch}{\partial^2 \theta_V} = \frac{\partial^2 ErrMatch}{\partial^2 \theta_V} + \frac{\partial^2 err_i}{\partial^2 \theta_V}$
  end
end
$var(x_V) = \left(\left|\frac{\partial^2 ErrMatch}{\partial^2 x_V}\right|\right)^{-1} \cdot SensorsKErrorPos$
$var(y_V) = \left(\left|\frac{\partial^2 ErrMatch}{\partial^2 y_V}\right|\right)^{-1} \cdot SensorsKErrorPos$
$var(\theta_V) = \left(\left|\frac{\partial^2 ErrMatch}{\partial^2 \theta_V}\right|\right)^{-1} \cdot SensorsKErrorOri$
$P_{Match}(k) = \begin{bmatrix} var(x_V) & 0 & 0 \\ 0 & var(y_V) & 0 \\ 0 & 0 & var(\theta_V) \end{bmatrix}$
end

Algorithm 10: Covariance Calculations.

Capítulo 4

Resultados

Neste capítulo apresentam-se os resultados obtidos no âmbito deste trabalho. Começa-se por descrever os ambientes de teste e, de seguida, apresentam-se os resultados relativos à localização baseada em reflectores e os resultados relativos à localização baseada em contornos.

4.1 Descrição dos ambientes de teste

Foram realizadas diversas experiências em diferentes ambientes de teste, tanto em simulação como com robôs reais, nomeadamente testes em espaços industriais e até campeonatos de futebol robótico. Nas próximas secções apresentam-se os diferentes ambientes onde foram recolhidos os resultados apresentados neste trabalho.

4.1.1 Simulador Stage

O "Stage"é um simulador de robôs disponível na *framework* ROS. O objectivo deste simulador é oferecer um bom compromisso entre o realismo da simulação, a sua simplicidade e o seu peso computacional. Nele é possível simular robôs móveis (com odometria e com diferentes configurações cinemáticas, nomeadamente diferenciais, triciclo e omnidirecionais) bem como diversos sensores (por exemplo, *laser scanners*). Na Figura 4.1 pode-se observar a interface do ambiente de simulação utilizado neste trabalho. A simulação possui os seguintes elementos:

- Um mapa de ocupação que descreve a planta do ambiente de simulação (disposição das paredes);
- Um robô móvel de tracção do tipo triciclo com odometria;
- Um *laser scanner* com uma abertura 360° com 288 ou 1440 medidas por *scan*;
- Um conjunto de objectos detectáveis pelo *laser scanner* que simulam a ocorrência de *outliers*.

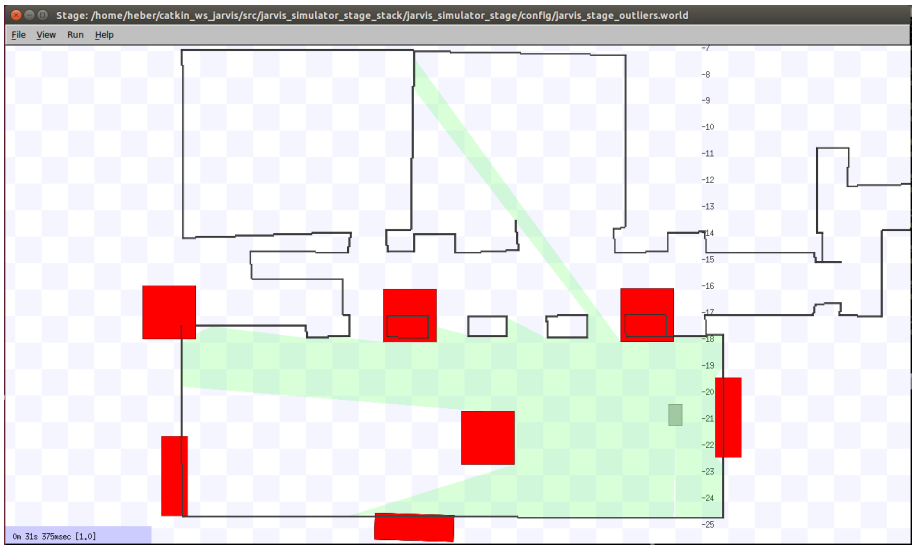


Figura 4.1: Interface do simulador "Stage"em que se pode observar a posição do robô, simbolizada pelo retângulo cinzento, os dados do *laser scanner* a verde, o mapa do ambiente utilizado a preto e a disposição dos *outliers*, representados pelos retângulos vermelhos.

Este ambiente de simulação foi utilizado nas experiências relacionadas com os desenvolvimentos na localização baseada em contornos. Utilizou-se um modelo do laser sem ruído e o mapa de configuração do espaço de simulação é o mesmo do sistema de localização. O objectivo era dar preferência a um ambiente de simulação simples para analisar pontos em concreto nos sistemas desenvolvidos em deterimento de uma simulação altamente realista, pois têm-se acesso a plataformas reais.

4.1.2 Robô Jarvis

O Robô Jarvis é um protótipo de um AGV industrial utilizado para testes e validação de algoritmos de navegação. Apresenta-se uma breve descrição deste protótipo e serão listadas as características relevantes dos sensores utilizados neste trabalho.

A descrição e caracterização do Jarvis é relevante por dois motivos: por um lado, porque ele é uma das principais ferramentas de testes e validação dos algoritmos desenvolvidos; por outro lado, porque ele se encontra equipado com um sistema de localização industrial comercial, cuja caracterização pode ser utilizada como termo de comparação com sistemas deste tipo.

A plataforma desenvolvida está apresentada na Figura 4.2. Esta consiste num protótipo de um AGV equipado com diferentes sensores, desde *lasers range finders* a câmaras, de modo a permitir testes de diferentes famílias de algoritmos utilizados em robôs móveis. Este robô possui uma tracção do tipo triciclo, ou seja, possui uma roda simultaneamente motriz e direccional e duas rodas livres. No que concerne aos sensores relevantes para este trabalho, a plataforma comporta um laser de navegação e um laser de segurança, que serão descritos nas próximas secções.

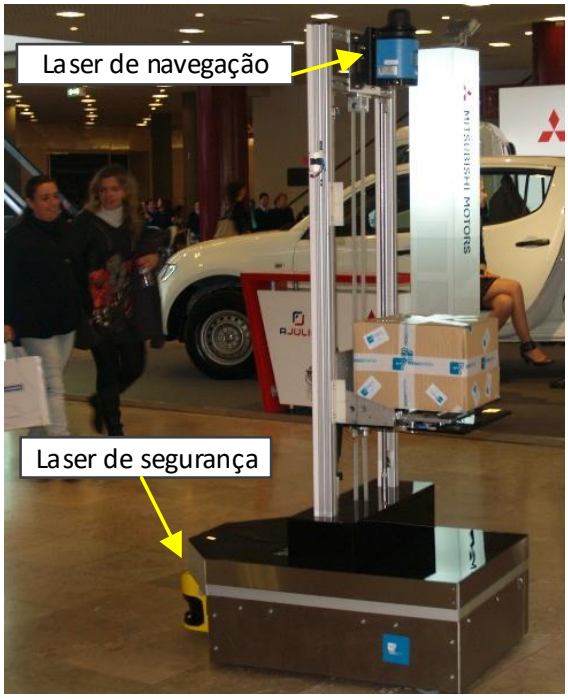


Figura 4.2: Robô Jarvis.

4.1.2.1 Laser de Segurança

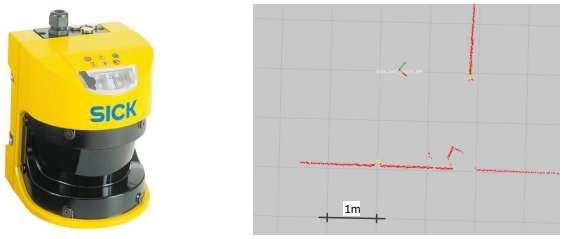


Figura 4.3: Esquerda: Laser de segurança Sick S3000 expert; Direita: exemplo de um scan, os pontos amarelos correspondem a medidas incidentes em reflectores e os pontos vermelhos a outras medidas, com os dois segmentos vermelho e verde temos a representação do referencial do laser.

A plataforma está equipada com um laser de segurança do modelo *sick S3000 expert*. Além de desempenhar a função de sensor de obstáculos no sistema de segurança, este equipamento fornece dados dos reflectores e contornos detectados. Mais concretamente, cada *scan* é composto por um conjunto de medidas que correspondem a valores de distância/ângulo face ao objecto detectado, indicando também se esse objecto é ou não um reflector. Na Figura 4.3, à direita, vemos uma representação da informação que podemos obter de um *scan* do laser de segurança. A Tabela 4.1 sumaria algumas características do laser em análise.

Este laser de segurança pode funcionar numa das duas resoluções angulares representadas na Tabela 4.1: cada uma delas corresponde a um diferente período de amostragem. A reemissão do

Tabela 4.1: Características do laser de segurança.

Abertura angular	190°
Resolução angular	0.5° ou 0.25°
Período de amostragem	30ms ou 60ms
Alcance (máximo)	49m
Alcance (90% reemissão)	40m
Alcance (20% reemissão)	17m
Alcance contornos (10% reemissão)	12m
Erro sistemático da distância	±5mm
Erros aleatório da distância	±19mm

sinal do laser por um objecto influencia a distância máxima a que este pode ser detectado, daí serem apresentados diferentes valores de alcance para diferentes valores de reemissão. Por fim, os valores de erro sistemáticos/aleatórios dizem respeito à detecção de um objecto até uma distância de 5.5m para medidas com uma reemissão no limiar de detecção do sensor. O valor de reemissão mínimo de um material para ser detectado pelo sensor é 1.8%. Um exemplo de um material com esta característica é o couro preto.

A documentação técnica deste laser de segurança tem uma série de recomendações em relação à detecção dos reflectores:

- Podem ser utilizados reflectores de forma plana ou cilíndrica, sendo que os reflectores planos são mais fáceis de instalar em superfícies planas como, por exemplo, paredes, enquanto que os reflectores cilíndricos são mais fáceis de detectar de diferentes ângulos;
- Os reflectores devem ter uma altura mínima de 500mm;
- Objectos altamente reflectores (por exemplo, superfícies de aço inoxidável) podem ser detectados como reflectores (falsos positivos);
- A distância mínima de detecção de um reflector é 40cm.
- O tamanho recomendado para um reflector é dado em função da distância máxima de detecção pretendida (Figura 4.4).

4.1.2.2 Laser de navegação

O laser de navegação visível nas Figuras 4.2 e 4.5 é do modelo *Sick Nav350*. Este, além de ser um dispositivo de localização baseado em triangulação laser, permite medir contornos e detectar/medir posições de reflectores, sendo que podemos obter as seguintes informações:

- Medidas dos contornos – Distâncias, ângulos e reemissão dos objectos circundantes ao laser. Ao contrário do laser de segurança apresentado anteriormente, este dispositivo fornece o valor real de reemissão e não um valor booleano que indique se está a ser ou não detectado um reflector;

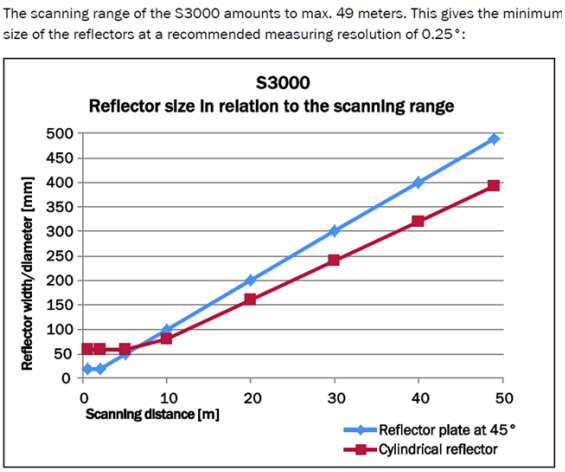


Figura 4.4: Relação entre tamanho recomendado e a distância máxima aos reflectores.

- Medidas dos reflectores – Distância e ângulos aos reflectores detectados. Para obter esta informação, o dispositivo tem de ser configurado com a informação da forma (plana ou cilíndrica) e tamanho dos reflectores instalados no meio;
- Medida da pose do robô – Posição e orientação absoluta do AGV obtidas a partir da detecção de reflectores instalados na área de navegação. Para obter esta informação, o dispositivo tem de ser configurado com a posição absoluta dos reflectores. Alternativamente, esta configuração pode ser feita por demonstração; no entanto, este procedimento é desaconselhado pelos fabricantes, que afirmam que o sistema é mais preciso se a posição dos reflectores for medida por pessoal qualificado. Também é possível obter do dispositivo um indicador de qualidade da medida da pose do AGV. Este indicador corresponde ao desvio padrão do erro entre as posições dos reflectores detectados num *scan* e as posições absolutas dos reflectores guardadas no laser de navegação.

Tabela 4.2: Características do laser de navegação referentes à medida dos contornos.

Abertura angular	360°
Resolução angular	0.25°
Período de amostragem	125ms
Alcance (máximo)	0.5m a 250m
Alcance (90% reemissão)	0.5m a 100m
Alcance (20% reemissão)	0.5m a 50m
Alcance contornos (10% reemissão)	0.5m a 35m
Erro sistemático da distância	±15mm
Erros aleatório da distância	±15mm

A Tabela 4.2 lista algumas características presentes na documentação técnica do equipamento referentes às medidas dos contornos. Os valores de erro sistemático e aleatório da medida de distância apresentados correspondem a valores de reemissão entre os 20% e 90% a uma temperatura

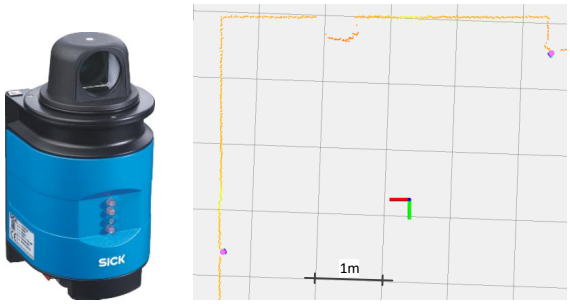


Figura 4.5: Esquerda: Laser de navegação Sick Nav350; Direita: exemplo de um *scan*, os contornos estão representados por um conjunto de pontos coloridos com uma cor proporcional ao valor de reemissão; a detecção dos reflectores está representada pelos círculos a violeta; a representação do referencial do laser encontra-se a vermelho e verde.

de 25°.

Tabela 4.3: Características do laser de navegação referentes à medida dos reflectores.

Alcance (máximo)	0.5m a 70m
Erro sistemático distância	±10mm
Erros aleatório distância	±10mm
Erro sistemático ângulo	±0.1° com o reflector a uma distância até 10m ≥ ±0.25° com o reflector a uma distância superior a 30m
Erros aleatório ângulo	±0.05°

A Tabela 4.3 apresenta as características de medida de posição dos reflectores pelo laser de navegação. Os valores apresentados correspondem à utilização de reflectores com o diâmetro mínimo suportado (80mm) e com o AGV parado. Também é recomendada uma altura mínima para os reflectores de 500mm para uma distância até 30m, 750mm para uma distância de 46m e 1000mm para uma distância até 70m. Da mesma forma, no laser de segurança também é recomendado o uso da folha reflectora "Diamond Grade".

Tabela 4.4: Características do laser de navegação referentes à medida da pose.

Erro de posição	±4mm com reflectores a uma distância até 10m ±10mm com reflectores a uma distância até 20m ±15mm com reflectores a uma distância até 30m ≥ ±25mm com reflectores a uma distância até 50m
Erro de orientação	±0.1° com o reflector a uma distância até 10m ≥ ±0.25° com o reflector a uma distância superior a 20m

A informação técnica na Tabela 4.4 apresenta os valores típicos da medida da pose do laser de navegação através de reflectores. Estes valores correspondem à utilização deste equipamento em ambiente industrial com reflectores de 80mm de diâmetro. Outras condições para obter estes valores de precisão são: a visualização de, pelo menos, seis reflectores por medição; a visualização

de igual número de reflectores de ambos os lados da trajectória do AGV; valor máximo de 120° para o ângulo visível entre reflectores.

No que toca à localização baseada na detecção de reflectores, um dos factores determinantes na qualidade dos resultados é a disposição dos reflectores na área de navegação do AGV. A informação técnica do equipamento que estamos a descrever fornece uma série de recomendações acerca das posições a que os reflectores devem ser instalados, que acha-se pertinente sumariar:

- No mínimo três reflectores têm de ser visíveis ao longo da trajectória do AGV, mas é recomendado que sejam visíveis cinco;
- Devem estar visíveis o mesmo número de reflectores em ambos os lados do AGV;
- O ângulo entre dois reflectores sucessivos não deve ser superior a 120°;
- Os reflectores devem ser instalados de forma assimétrica e as distâncias entre reflectores devem variar pelo menos 500mm. Isto permite ao sistema determinar a pose inicial do robô. Este processo é designado muitas vezes por localização global e consiste em determinar a pose do AGV sem nenhuma estimativa desta. O processo de localização global é executado apenas no arranque do sistema e pode demorar vários segundos dependendo do número de reflectores vistos e mapeados;
- Os reflectores têm de ser instalados de forma a ficarem a uma distância superior a 300mm de objectos que possam ser incorrectamente identificados como reflectores, como por exemplo, janelas, peças em aço inoxidável, tubos de metal, etc;
- Os reflectores devem ser instalados próximos de zonas da trajectória do AGV em que os requisitos de precisão são mais elevados. Mesmo os reflectores mais distantes devem ser filtrados; para isso, existem dois métodos: usar um número máximo de reflectores mais próximos; usar apenas os reflectores que estejam a uma distância menor que um determinado valor.

Por fim, deve ser fornecido ao equipamento as medidas de velocidade do AGV provenientes dos *encoders*. Esta informação permite corrigir a posição das balizas medidas aumentando a precisão/robustez do sistema e extrapolar a medida da pose do AGV para instantes de tempo diferentes do tempo de aquisição das medidas.

4.1.3 Robô Igor

O robô Igor, presente na Figura 4.6 é um protótipo de um AGV industrial desenvolvido no âmbito do projecto PRODUTECH PSI PPS3. Este robô possui uma tracção do tipo triciclo e está equipado com o laser de segurança descrito na Secção 4.1.2.1.

O Igor foi desenvolvido com o intuito de transportar mesas com material utilizado numa linha de produção industrial. Não possui *laser* de navegação de forma a baixar a sua altura e permitir a sua passagem por baixo das mesas que transporta.

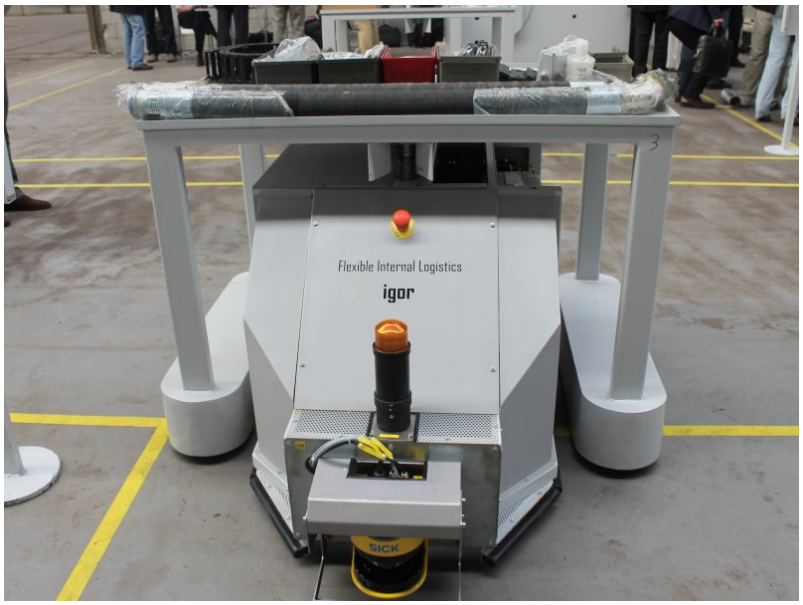


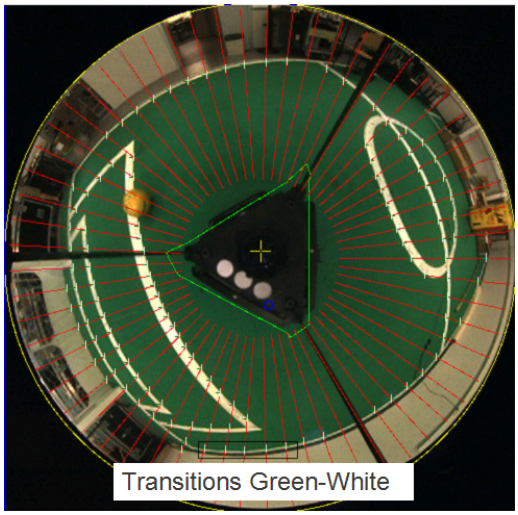
Figura 4.6: Robô Igor.

O processo de auto-localização é feito través do laser de segurança que detecta reflectores instalados junto ao chão. Foi o Igor que permitiu o teste do sistema de localização baseado em reflectores em ambiente industrial.

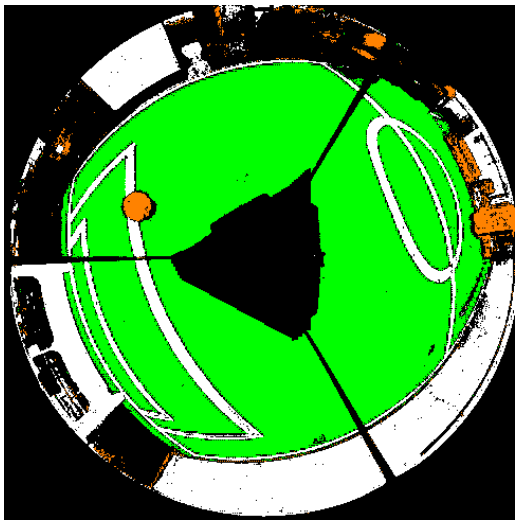
4.1.4 Futebol Robótico

Figura 4.7: Elemento da equipa de futebol robótico 5dpo.

Na Figura 4.7 observa-se um elemento da equipa 5DPO, que compete na liga de Futebol Robótico Médio (*Medium Size League*, MSL). Estes robôs possuem uma tração omnidirecional, e uma câmara omnidirecional, que permite captar uma imagem com uma abertura angular de 360° (Figura 4.8a) do ambiente envolvente. É efectuada uma calibração inicial manual na imagem da câmara, de forma a gerar uma *look up table* que faz corresponder determinada cor (no espaço HSV) a um objecto de interesse. Com esta *look up table* as cores da imagem são segmentadas, utilizando um algoritmo de processamento de imagem, de forma a detectar o verde do campo, o branco das linhas, a cor da bola e dos outros robôs (Figura 4.8b), permitindo a sua distinção.



(a) Detecção das transições de verde para branco.



(b) Segmentação das cores.

Figura 4.8: Exemplo da imagem captada pelo sistema de visão artificial de um robô do futebol robótico. À esquerda temos a detecção das transições de verde para branco ao longo das linhas radiais vermelhas. Este informação é utilizada pelo sistema de localização do robô. À direita o resultado da segmentação das cores, são segmentadas as cores verde e branco do campo e a cor da bola.

A partir da imagem segmentada é possível detectar as transições de verde para branco, ao longo de um conjunto de semi-rectas radiais (Figura 4.8a). A partir dessas detecções é calculado um conjunto de posições relativas. O algoritmo de *Map Matching* apresentado na Secção 3.2.2 é aplicado a este conjunto de posições, bem como ao mapa das linhas do campo, permitindo localizar o robô.

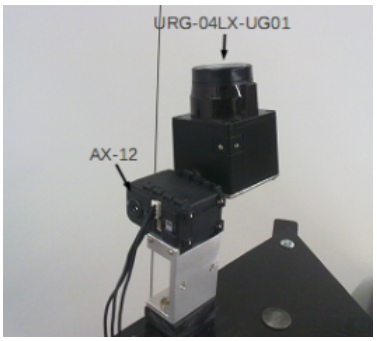
Na liga MSL, as dimensões do campo são conhecidas à priori, o que torna possível a geração das três matrizes fundamentais utilizadas pelo algoritmo de Lauer et al. (2006): a matriz de distâncias, e as respetivas derivadas ao longo do eixo horizontal e vertical. Estas matrizes são guardadas no formato de *look up tables* e utilizadas no algoritmo de *Map Matching*. A utilização destas *look up tables* torna o algoritmo extremamente rápido, o que permite a sua utilização *online* a frequências elevadas.

Estes robôs também estão equipados com uma bússola magnética que permite medir a sua orientação em relação ao norte magnético da Terra. Estes robôs utilizam o sistema de localização global apresentado na Secção 3.2.2.3.

4.1.5 Robô vigilante

O Robô Vigilante é um robô de tracção diferencial que navega em ambientes *indoor* recorrendo a informação tridimensional adquirida por um *laser scanner* rotativo. Na Figura 4.9, têm-se uma imagem do Robô Vigilante e, na Figura, 4.9b pode-se observar o *laser scanner* (Hokuyo URG-04LX-UG01) e o servo-motor (AX-12) montado no topo do robô. De forma a obter informação

(a) Robô vigilante em operação durante uma demonstração pública.



(b) *Laser scanner* acoplado a um servo-motor de forma a adquirir dados 3D.

Figura 4.9: Plataforma Robô Vigilante.

tridimensional do meio ambiente, desenvolveu-se uma plataforma rotativa baseada num servo-motor na qual o *laser-scanner* está montado. Conhecendo a transformação entre o referencial do *laser* e o do robô (incluindo o ângulo e velocidade do eixo do servo-motor adquiridos a cada ciclo) é possível aplicar um conjunto de transformações homogéneas (rotações e translações) que permitem transformar os pontos 2D do referencial do laser em pontos 3D no referencial do robô.

A parte superior do interior de um edifício, nomeadamente as paredes e o tecto acima da altura habitual de uma pessoa (considera-se uma altura de 1.8m) pode ser considerada um cenário predominantemente estático, sem obstáculos móveis, que se mantêm inalterada por longos períodos de tempo. A informação 3D medida pelo *laser* rotativo desse cenário estático pode ser utilizada para localizar o robô (no espaço 2D) mesmo em ambientes dinâmicos com a presença de um grande número de pessoas movendo-se na área de navegação.

Se um mapa tridimensional do cenário estático for previamente criado, o algoritmo de *Map Matching* apresentado por Lauer et al. (2006) pode ser estendido de forma a utilizar as medidas 3D adquiridos pelo *laser* rotativo.

Como o mapa da área de navegação correspondente a esta situação não é conhecido à partida (contrariamente ao caso do Futebol Robótico) desenvolveu-se um filtro de kalman estendido que resolve o problema de SLAM (*Simultaneous Localisation and Mapping*) *offline*. Obtêm-se assim o mapa de ocupação tridimensional do espaço e, a partir deste, calculam-se a matriz de distâncias e as respectivas matrizes de gradientes ao longo da direcção x e y utilizadas pelo algoritmo de *Map Matching*.

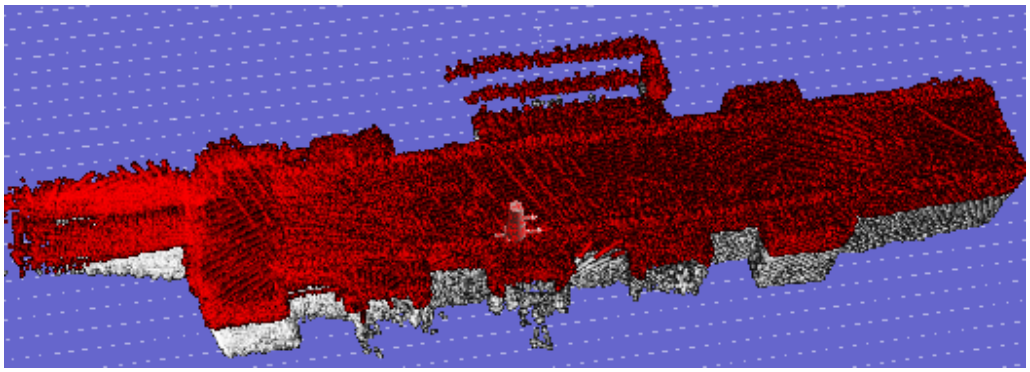


Figura 4.10: Mapa de ocupação tridimensional utilizado pelo sistema de localização do Robô Vigilante. A zona marcada a vermelho é tipicamente mais estática por estar a uma altura elevada em relação do chão, oferecendo ao sistema de localização uma qualidade sensorial superior.

Figura 4.11: Fatia do mapa de distâncias tridimensional correspondente ao mapa de ocupação da Figura 4.10.

Na Figura 4.10 temos o exemplo de um mapa tridimensional gerado pelo algoritmo referido anteriormente a parte superior do mapa (a vermelho) corresponde a uma altura acima dos 1.8m. É esta zona que é utilizada pelo sistema de localização. Na Figura 4.11 temos a representação de uma fatia da correspondente matriz de distâncias, isto é o conjunto de posições para uma altura igual a 1.8 em que a intensidade da cor é proporcional à distância.

Como já foi referido, estas matrizes são guardadas em memória e utilizadas pelo sistema de localização do robô. A partir da informação 3D do *laser* rotativo e de um mapa de ocupação 3D do espaço (anteriormente gerado aquando da configuração do sistema) o sistema de localização estima a pose 2D do robô.

A pertinência da breve descrição que se fez deste sistema prende-se com o facto de terem sido testados os desenvolvimentos relativos à localização global. Este robô também está equipado com uma bússola magnética. A extensão do algoritmo de *Map Matching* referida nesta secção e o processo de aquisição do mapa tridimensional não fazem parte do presente trabalho, no entanto, eles podem ser encontrados em detalhe nos trabalhos Pinto et al. (2012) e Pinto et al. (2013).

4.2 Resultados referentes à localização baseada em reflectores

Nesta secção apresentam-se resultados experimentais que procuram validar a abordagem em relação aos reflectores a dois níveis: precisão e robustez. Para testar a precisão, foram realizadas experiências laboratoriais em que a resposta do sistema desenvolvido foi comparada com a resposta de uma solução comercial de localização. Para testar a robustez a outliers, analisou-se a estabilidade da resposta do sistema com diferentes configurações. Para tal recorreu-se a um *dataset* adquirido em ambiente industrial.

4.2.1 Análise de precisão

De forma a obter resultados experimentais com *ground truth* recorreu-se ao AGV descrito na Secção 4.1.2.2. Este está equipado com um laser de navegação. Assim, a resposta do algoritmo pode ser comparada com a resposta de um sistema de localização industrial comercial baseado em triangulação laser, assumido como *ground truth*. A experiência que se descreve em seguida recorreu a um laser de navegação com a designação "SICK NAV350". Este além de fornecer a pose do robô, fornece também a posição relativa dos reflectores detectados (o equivalente ao Z_B da Figura 3.4).

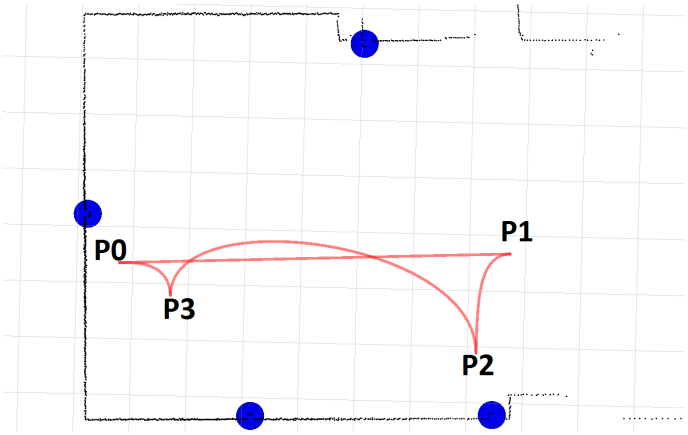


Figura 4.12: Disposição dos reflectores (círculos azuis) e trajectória utilizada (linha vermelha) durante os testes de precisão.

Na Figura 4.12 está assinalada a posição dos quatro reflectores (círculos azuis) e a trajectória (linha vermelha) utilizada na experiência que se está a apresentar. Na trajectória considerada o robô desloca-se em linha recta de $P0$ para $P1$, invertendo seguidamente o sentido até $P2$, seguindo em frente até $P3$ e finalizando o trajecto em $P0$. Aqui inverte mais uma vez o sentido de movimento. Ao longo deste trajecto foi comparada a pose fornecida pelo sistema desenvolvido com a pose fornecida pelo sistema comercial. Ressalva-se que nesta experiencia está a ser utilizado um laser de navegação e não um laser de segurança e, portanto, os quarto reflectores estão instalados nas paredes a uma certa altura na qual estão sempre visíveis ao longo de toda a trajectória, uma vez o laser de navegação tem um ângulo de visão (FOV) de 360°. O *output* do sistema comercial de

triangulação laser foi usada pelo controlador de trajectórias. Este último consiste num controlador PID para curvas paramétricas baseado em *splines*.

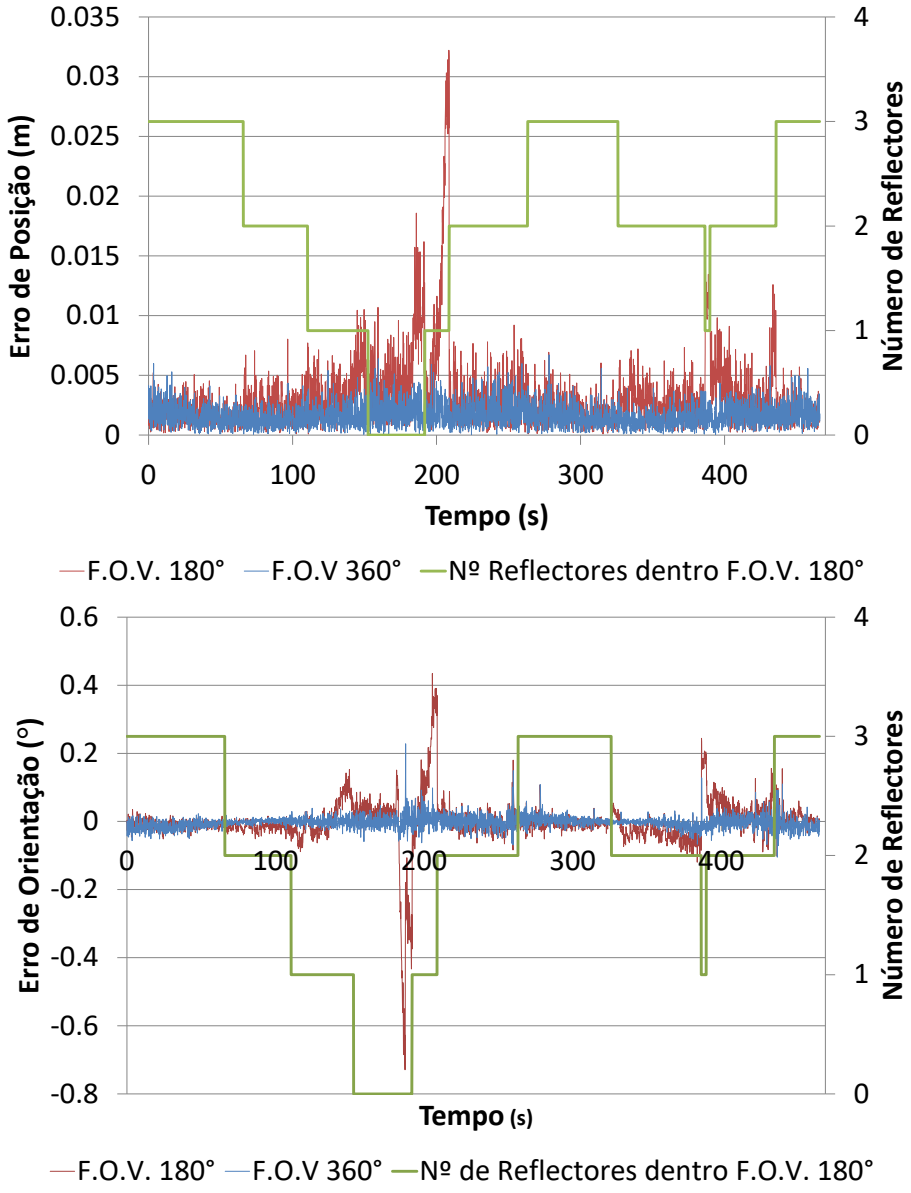


Figura 4.13: Comparação dos erros da solução proposta para dois FOV diferentes (linha azul: FOV de 360°, linha vermelha: FOV de 180°). O gráfico de cima apresenta os erros de posição, e o gráfico de baixo apresenta os erros de orientação.

Como seria de esperar obtiveram-se erros baixos de posição e orientação, o que pode ser confirmado pelas linhas azuis na Figura 4.13, na qual o algoritmo desenvolvido funcionou com um FOV de 360°. Utilizando-se esta mesma experiência e respectivos dados gerados, pode ser analisada a forma como o sistema reage ao ser exposto a menor informação sensorial. Para tal, o campo de visão do sistema aqui desenvolvido foi reduzido artificialmente por software para 180° (isto é, ignorando medidas superiores a $\pm 90°$). Esta situação é análoga à utilização de lasers

de segurança na qual o campo de visão é tipicamente mais reduzido. Impondo esta restrição, foram obtidos os resultados apresentados na Figura 4.13 pelas linhas vermelhas. As linhas verdes indicam o número de reflectores visíveis dentro do reduzido FOV de 180°.

Observa-se que a precisão do sistema em análise não é significativamente afectada pela redução do FOV, excepto em situações onde estão visíveis menos de dois reflectores ou em casos em que apenas a odometria é usada por longos períodos de tempo. Estes resultados evidenciam a vantagem da utilização do filtro de Kalman. O *ground truth* utilizado é baseado em triangulação laser, pelo que são necessários pelo menos três reflectores visíveis simultaneamente (sendo que o valor recomendado pelo fabricante é de, no mínimo, cinco reflectores visíveis). Por outro lado, o filtro de Kalman, ao fundir medidas de ângulos e distâncias a reflectores com a odometria, consegue estimar a pose do robô mesmo tendo menos do que três reflectores em linha de vista. Isto é uma grande vantagem, uma vez que reduz o número de reflectores a instalar e, portanto, custos inerentes ao *setup* de AGVs.

4.2.2 Análise de robustez: Rejeição de outliers

Enquanto na secção anterior foi analisada a precisão do sistema recorrendo a um laser de navegação, nesta analisa-se a robustez utilizando um laser de segurança quando exposto a adversidades (*outliers*) em ambiente fabril real. Para efeitos de teste, foi utilizada a plataforma apresentada na Secção 4.1.2.1.

Figura 4.14: Ambiente industrial onde foram efetuados os testes de robustez a *outliers*.

No âmbito do projecto PRODUTECH PSI PPS3, foi realizada uma demostração pública na qual foi testado em ambiente fabril o sistema aqui apresentado. O último pode ser visto na Figura

4.14. Além da recolha de dados, foi também possível validar-se o sistema, uma vez que o robô realizou a sua missão com sucesso durante um período consecutivo de 6 horas, sem se verificar falhas. A missão do robô consistia em transportar mesas entre postos de trabalho. Para efectuar esta tarefa é necessária uma precisão de cerca de 1cm (posição) e 3 graus (orientação). O AGV navegou numa área de 24m x 13m a uma velocidade de 0.5m/s. Este teste foi especialmente exigente para o sistema de localização pelo facto de estarem cerca de meia centena de pessoas a circular na área de navegação do AGV e o chão ser muito irregular. Em certos pontos na trajectória, o alcance do laser ficava reduzido a 4m, uma vez que o laser apontava para o chão. Estas adversidades aumentaram o número de oclusões de reflectores e aumentaram a quantidade de outliers a que o sistema foi exposto.

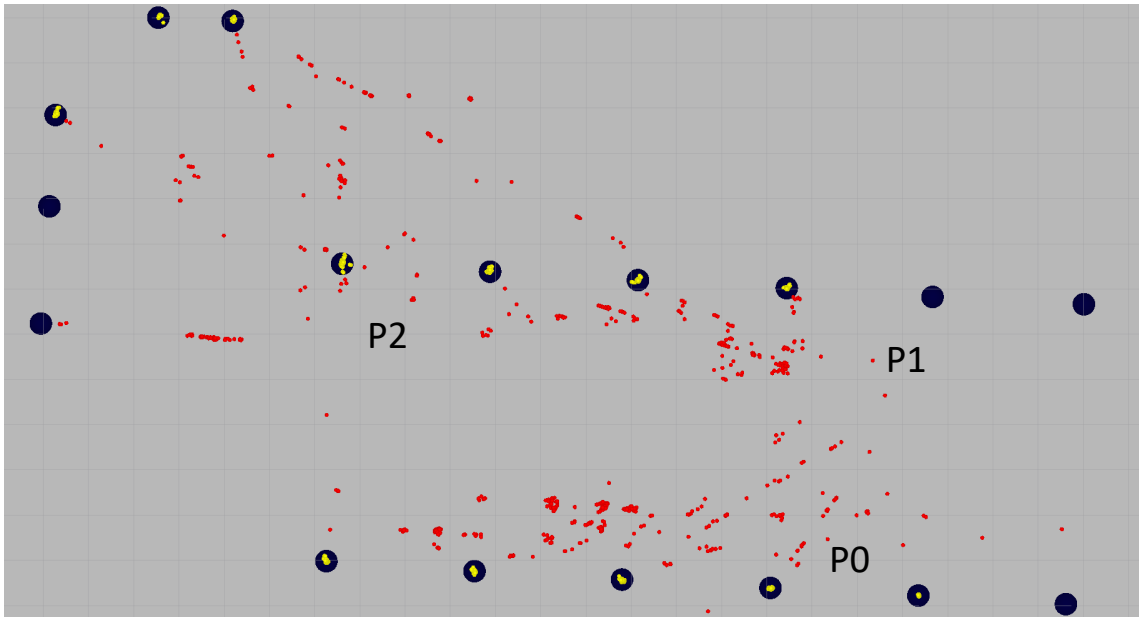


Figura 4.15: Ocorrência de *outliers* em ambiente industrial. As circunferências azuis escuras indicam a posição dos reflectores instalados em ambiente industrial. Os pontos vermelhos e amarelos representam as posições dos reflectores detectados pelo *laser* de segurança durante um percurso de P0 para P1 e depois para P2. As detecções marcadas a vermelho correspondem aos *outliers* rejeitados pelo "*Reflector Detector*"ou pelo "*Association/Outlier Filter*", e os pontos amarelos correspondem a *inliers*. Tamanho da grelha: 1m.

De forma a dar uma ideia da quantidade de *outliers* a que sistema foi exposto nesta experiencia apresenta-se a Figura 4.15. Nesta imagem estão assinaladas as posições dos reflectores usando circunferências azuis escuras e os pontos de passagem. O AGV deslocou-se de marcha a trás de $P0$ para $P1$ e depois seguiu em frente até $P2$. Durante este percurso, a posição dos reflectores detectados pelo laser de segurança está representada pelos pontos amarelos e vermelhos. Os pontos amarelos correspondem a *inliers* e os pontos vermelhos a *outliers* rejeitados pelo "*Reflector Detector*"ou pelo "*Association/Outlier Filter*".

Na Figura 4.16, retirada de um *screenshot* da interface do sistema de navegação, durante o

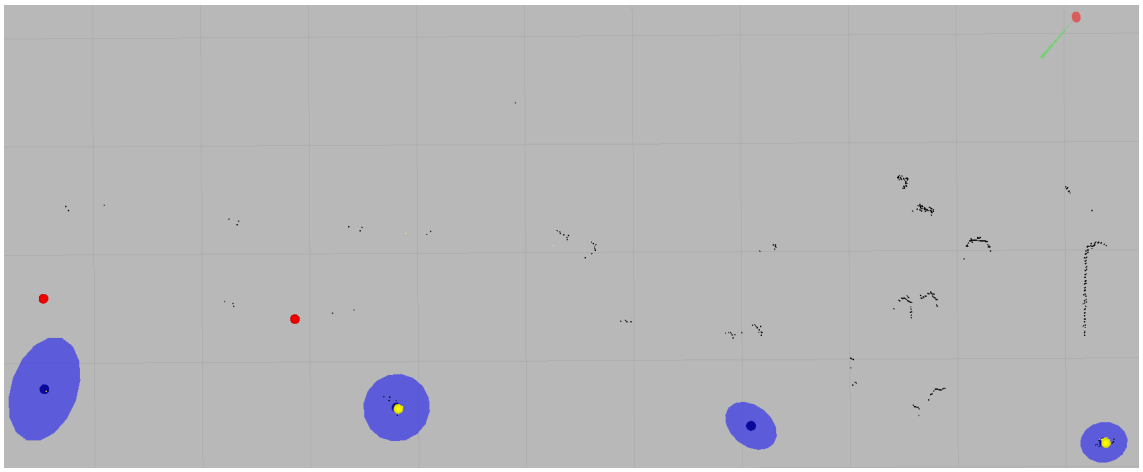


Figura 4.16: *Screenshot* da interface do sistema de localização durante os testes de robustez. Os pontos a vermelho são os reflectores rejeitados, por outro lado os pontos a amarelo representam os reflectores aceites. As elipses azuis representam a área fora da qual a ocorrência de um detecção é considerada um outlier, e a elipse vermelha (canto superior direito) representa a covariância da posição estimada. Os restantes pontos pretos representam os dados adquiridos pelo laser de segurança. Tamanho da grelha: 1m.

percurso descrito anteriormente, podemos observar o mapa de reflectores M_i (pontos a azul escuro), a covariância da posição estimada do AGV através da elipse vermelha e as correspondentes covariâncias das previsões das observações representadas pelas elipses azuis. A zona em torno da posição dos reflectores no mundo corresponde a uma zona de validação, na qual se baseia o filtro de outliers (Algoritmo 3, Linha 11). Observações para além desta zona têm probabilidade de ocorrência menor que 5%, sendo assinaladas como *outliers* e ignoradas pelo sistema. Como se vê na imagem, os reflectores detectados (observações) considerados *inliers* estão representados com pontos amarelos e a vermelho estão representadas as medidas rejeitadas (*outliers*). Os pontos pretos representam medidas de distâncias fornecidas pelo laser de segurança (Z_L).

Neste ponto torna-se evidente a existência de um grande número de falsos positivos na detecção de reflectores através de um laser de segurança. A correcta filtragem de outliers neste tipo de aplicações toma então um papel fulcral. Na abordagem aqui apresentada esta filtragem é feita em feita em dois pontos do sistema, ao nível do detector de reflectores ("*Detector Filter*") e após o processo de associação ("*Association Filter*"). De forma a evidenciar a importância do processo de filtragem seleccionou-se uma porção dos dados recolhidos e realizaram-se três experiências com três configurações diferentes. A Figura 4.17 apresenta o resultado do registo da posição e orientação obtida nas três configurações. Na primeira experiência não se filtrou *outliers* (linha azul), na segunda foi activado o "*Detector Filter*" (linha verde), e por fim ambos filtros foram activados "*Detector Filter*" e "*Association Filter*" (linha vermelha). Como se pode perceber facilmente, o sistema rapidamente diverge da solução verdadeira aquando da inexistência de um processo de filtragem, entrando num modo em que o erro em relação à pose verdadeira é demasiado grande para que o processo de associação seja feito correctamente. Nesta situação a estimação da pose

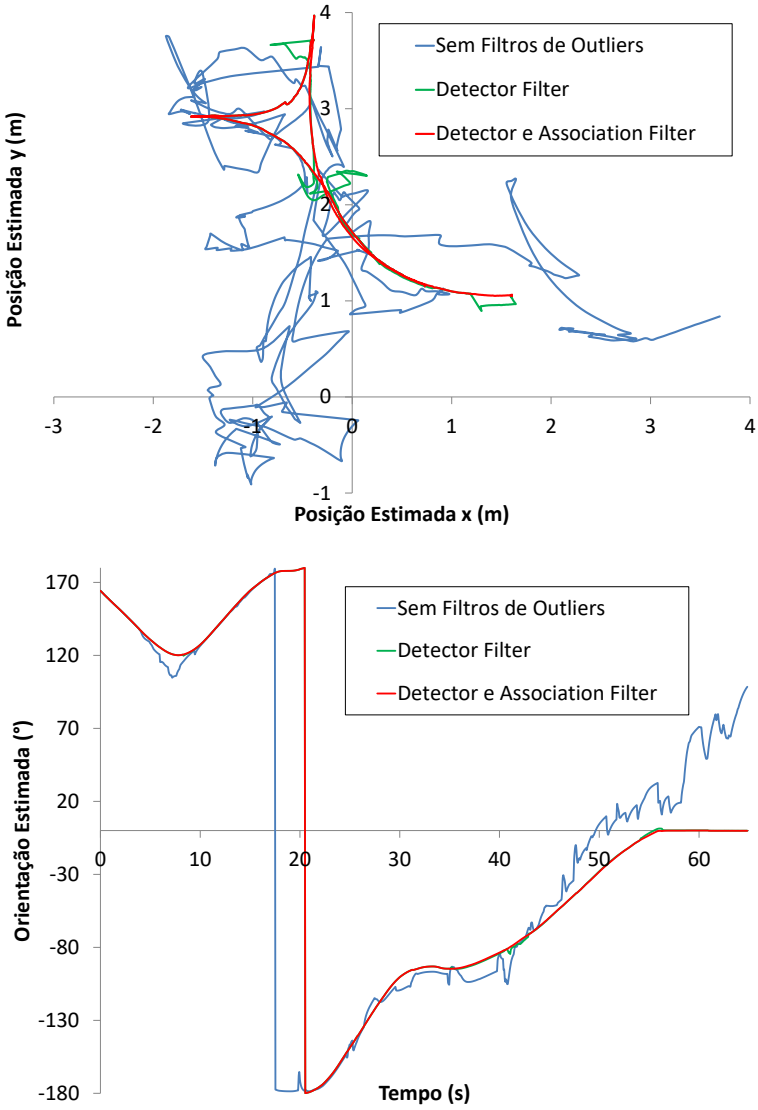


Figura 4.17: Resposta do sistema em posição (gráfico de cima) e orientação (gráfico de baixo) a um *dataset* recolhido em ambiente industrial. Estão representados os resultados para três configurações diferentes. Sem filtros (azul), apenas com o "*Detector Filter*"(verde) e com ambos os filtros (vermelho).

do AGV falha completamente. Activando o *"Detector Filter"* há uma melhoria substancial, já que o filtro não perde "*tracking* à pose", embora em alguns pontos da trajectória se verifique variações abrutas na estimação da posição. Por fim, activando ambos os filtros, todos os *outliers* são correctamente identificados e não são verificadas variações abrutas de posição e orientação.

Nos seguintes links podem ser vistos vídeos da interface do sistema de localização correspondentes às três experiências acima descritas com informação similar à apresentada na Figura 4.16:

- Sem filtros de outliers: `http://youtu.be/wCv9qVTSICg`;

- "*Detector Filter*": http://youtu.be/iTCb5UR6CRE;
- "*Detector Filter*"e "*Association Filter*": http://youtu.be/4_Io52ORvOE.

Termina-se esta secção com dois links onde se pode observar o AGV a realizar a sua missão de transporte de mesas: uma em ambiente laboratorial e outra no ambiente fabril referido anteriormente:

- Laboratório: http://youtu.be/6SQ3llbTSFk;
- Ambiente industrial: https://youtu.be/cyokKOBxcY0.

4.3 Resultados referentes à localização baseada em contornos

Nesta secção é feita uma exposição dos resultados referentes à localização baseada em contornos. Como já foi referido o algoritmo *Perfect Match* foi a base de desenvolvimento da solução proposta, neste sentido é apresentado um estudo comparativo de vários algoritmos de *map matching* de forma a suportar esta escolha (Secção 4.3.1). De seguida na Secção 4.3.2 é feita uma análise com base no *setup* experimental utilizado na análise comparativa de forma a evidenciar o impacto das alterações propostas. Por fim a Secção 4.3.3 é apresenta-se os resultados referentes aos desenvolvimentos na localização global.

4.3.1 Análise comparativa de algoritmos de matching

Os desenvolvimentos aqui apresentados em relação à localização por contornos baseiam-se no algoritmo *Perfect Match* (PM) apresentado por Lauer et al. (2006). Este algoritmo ganhou popularidade principalmente devido ao sucesso da sua aplicação nas competições de futebol robótico médio (*Middle Size League*). Com esta técnica foi possível obter uma localização robusta para os robôs de futebol que requerem frequências elevadas de controlo na tomada de decisões. Contudo, existem outros algoritmos de correspondência de mapas com bastante relevância no estado da arte. Por exemplo, no projecto *open-source* "*Point Cloud Library* "(PCL) estão disponíveis diversas implementações dos algoritmos mais utilizados para procurar a correspondência entre nuvens de pontos 2D e 3D. Nomeadamente o "*Iterative Closest Point*"(ICP) e o "*Normal Distributions Transform*"(NDT) descritos anteriormente nas secções 2.2.2 e 2.2.3. Neste sentido, foi realizada uma análise comparativa entre o PM, o ICP e o NDT, de forma a reforçar a relevância do PM no estado na arte e justificar a sua escolha como base de desenvolvimento. A avaliação e comparação dos algoritmos em estudo foi feita considerando diferentes métricas:

- Peso computacional;
- Velocidade de convergência;
- Erro máximo de inicialização (erro máximo na estimativa inicial da pose do robô que o algoritmo pode tolerar);

- Robustez a *outliers*.

4.3.1.1 Setup experimental

De modo a evidenciar as vantagens de cada algoritmo, foi realizada uma análise apenas ao core de cada, minimizando quanto possível a influência da parametrização. As experiências foram feitas em ambiente ROS (Quigley et al., 2009), recorrendo-se ao trabalho de Costa et al. (2016), que oferece uma interface ente ROS e PCL. Este último foi desenvolvido para resolver o problema da localização de robôs e é completamente parametrizável. Foram realizadas experiências recorrendo ao ambiente de simulação descrito na secção 4.1.1 e também a uma plataforma robótica real descrita na Secção 4.1.2, Figura 4.2. Esta está equipada com um sistema de localização comercial utilizado aqui como *groundtruth*.

De seguida são listadas as condições em que as experiências foram realizadas e os parâmetros utilizados:

- Todos os algoritmos processam os mesmos dados sensoriais;
- Todos os algoritmos usam o mesmo mapa de referência;
- Todos os algoritmos usam os mesmos critérios de paragem;
- Nenhum dos algoritmos tem acesso aos dados de odometria, neste caso o erro de inicialização é causado pelo movimento do robô, que se desloca a uma velocidade de 0.5m/s. Ou seja o erro de inicialização corresponde à quantidade de deslocamento efectuado pelo robô desde a última vez que a rotina de map-matching foi executada.
- A versão do ICP disponível no PCL utilizada nos testes foi a $iterative\_closest\_point\_2d$ disponível em (Costa et al., 2016), sem RANSAC ($max\_number\_of\_ransaciterations$: 0). Para assegurar o processamento de todos os dados, a distância máxima de procura foi configurada com um valor elevado ($max\_correspondence\_distance$: 9999.0); Além disto, a opção de utilizar correspondências recíprocas através do parâmetro $use\_reciprocal\_correspondences$: $false$ foi desactivada;
- Utilizou-se a versão 3D da implementação do NDT disponível no PCL;
- O computador utilizado possui um processador com um Intel Core i5 450M @ 2.40 GHz.

O PCL possui, também, uma versão 2D do algoritmo NDT, que foi inicialmente considerada para esta análise comparativa de algoritimos de *matching*. Contudo, não foi possível obter uma resposta estável da implementação 2D disponível no PCL. Recorreu-se, portanto, à versão 3D do NDT, assumindo que a implementação 2D tem erros de implementação que impedem o seu correcto funcionamento. Apesar deste problema, as versões 2D e 3D foram comparadas em termos de peso computacional, observando-se que a 3D é cerca de quatro vezes mais lenta que a 2D. Esta diferença pode ser explicada pelo facto do NDT recorrer a várias grelhas sobrepostas de forma

a diminuir o efeito de discretização. Tal conduz a que o cálculo do score (Equação 2.13) seja repetido o mesmo número de vezes que o número de grelhas utilizadas: a versão 2D utiliza quatro grelhas bidimensionais enquanto a versão 3D recorre a oito grelhas tridimensionais (*voxel grids*).

4.3.1.2 Resultados relativos ao peso computacional

Num primeiro teste, o objectivo principal foi avaliar o peso computacional das fases fundamentais dos algoritmos em análise, mais concretamente, o peso computacional duma iteração. No caso do ICP, avaliou-se o tempo gasto na procura de correspondências (na implementação do PCL utilizam-se Kd-tree (Muja e Lowe, 2009) e uma minimização baseada em mínimos quadrados.No caso do PM, o tempo gasto no cálculo de gradientes através de *look-up tables* e na optimização baseada no RPROP (Riedmiller e Braun, 1993). E finalmente no caso do NDT, o cálculo do *score* e optimização baseada no método de Newton.

De forma a não penalizar nenhum dos algoritmos em análise, o número de iterações de cada um foi fixada em 50. Também não se usaram filtros ou outros artefactos que possam influenciar a avaliação do peso computacional. Com isto, é pretendido verificar de que forma a quantidade de informação sensorial (neste caso, o número de medidas do *laser scanner*), o tamanho do mapa de referência (resolução) e a presença de ruído nos dados sensoriais influenciam o peso computacional. Na Figura 4.1 apresentamos a interface do simulador utilizado neste conjunto de experiências, onde a posição dos objectos não mapeados (*outliers*) é indicada pelos retângulos vermelhos.

A Tabela 4.5 apresenta o tempo consumido nas diferentes condições enunciadas dos algoritmos em análise. Como se pode observar, a implementação do PM utilizada é a computacionalmente mais leve em todas as situações analisadas. Observa-se também que o peso computacional do PM e do NTD é principalmente afectado pelo tamanho dos dados dos sensores, enquanto que o ICP é afectado por todas as situações analisadas (tamanho dos dados sensoriais, resolução do mapa de referência e a presença de *outliers*). Este elevado peso computacional observado no ICP pode ser explicado pela utilização de Kd-trees para representar o mapa de referência (na implementação do ICP no PCL) o que torna o acesso à informação do mapa de referência mais lento e menos determinístico quando comparado à utilização de "*lookup tables*" utilizadas no PM. As *Kd-trees* foram desenvolvidas com o intuito de optimizar a quantidade de memória ocupada pelo mapa de ocupação, sacrificando o tempo de acesso aos dados. Por outro lado também foi verificado que o PM ocupa três vezes mais memória que os outros algoritmos (ICP e NDT), mas para a aplicação em análise (localização 2D de um robô), a frequência de operação é um factor fulcral e as exigências em termos de memória são relativamente pequenas. Além disso, as exigências em termos de quantidade de memória podem ser atenuadas através de esquemas de gestão de mapas mais complexos, por exemplo, em vez de carregar todo o mapa de referência pode-se utilizar apenas a parte do mapa de referência que correspondente à zona onde o robô se encontra.

Analisou-se também a distribuição de tempo utilizada no ICP entre procura de correspondências e optimização. Dos resultados pode-se concluir que 90% do tempo ocupado por uma iteração é gasto na procura de correspondências no mapa de referência através da *Kd-tree*. Tendo isto em conta, foi proposto um método alternativo para a procura de correspondências no ICP. A

Tabela 4.5: Tempo computacional (em ms) gasto por cada algoritmo na execução de 50 iterações.

		Resolução do Mapa: 5cm				Resolução do Mapa: 1cm			
		Pontos: 288		Pontos: 1440		Pontos: 288		Pontos: 1440	
		Sem Out	Outliers	Sem Out	Outliers	Sem Out	Outliers	Sem Out	Outliers
PM	Tempo Médio	1	1	5	5	1	2	6	6
	Tempo Max	5	3	13	12	6	5	15	15
	Tempo Min	<1	<1	2	2	<1	<1	3	3
ICP	Tempo Médio	29	32	114	125	38	49	126	181
	Tempo Max	43	47	145	175	64	76	157	267
	Tempo Min	22	23	103	109	29	37	115	145
LUT-ICP	Tempo Médio	6	6	19	19	6	6	19	20
	Tempo Max	14	14	31	40	14	16	32	33
	Tempo Min	3	3	13	13	3	3	13	14
NDT	Tempo Médio	72	52	335	309	60	69	339	310
	Tempo Max	94	85	386	355	77	93	471	376
	Tempo Min	60	43	299	231	52	46	301	228

implementação disponível no PCL foi alterada de forma a utilizar uma *lookup table* em vez de *Kd-tree*. Esta *lookup table* guarda em cada célula as coordenadas da célula mais próxima ocupada no mapa. Na Tabela 4.5 é visível o aumento de desempenho em termos de peso computacional conseguido com esta proposta de alteração (*Lookup Table Iterative Closest Point*, LUT-ICP), a qual se apresenta cinco a seis vezes mais rápida do que a implementação *standard* disponível no PCL (baseada em *Kd-trees*). No LUT-ICP a etapa de procura de correspondências é simplificada, consistindo num acesso a uma matriz por cada ponto dos dados dos sensores, apresentando resultados muito mais próximos ao PM. Contudo, a geração da *lookup table* utilizada no LUT-ICP é um processo computacionalmente pesado e pode levar alguns segundos (e até alguns minutos, no caso de um mapa com 1cm de resolução e dimensões 40x50m) a ser calculado. Porém, este tempo de processamento extra não é crítico, uma vez que apenas é necessário construir a *lookup table* uma vez por mapa de referência. A última afirmação só é válida se o mapa de referência for estático. No caso de aplicações em que o mapa de referência é actualizado ao longo do tempo (como é no caso do SLAM, *Simultaneous Localization and Mapping*), tal obrigaria a recalcular as *lookup tables* utilizadas. Sugere-se como trabalho futuro analisar a possibilidade de otpimizar a geração das *lookup tables* para o caso em que mapas de referência são dinâmicos, tanto para o caso do PM e do ICP, comparando o peso computacional entre actualizar parte das *lookup tables* em função da zona do mapa de referência que é alterado ou actualizar a *Kd-tree*.

Analisando a Tabela 4.5, conclui-se que o NDT não é afectado pelo resolução do mapa de referência, já que este algoritmo discretiza o mapa do ambiente internamente num conjunto de distribuições normais com uma resolução fixa. Observa-se, também, que este foi o que obteve piores resultados em termos de peso computacional. Contudo, não se considera este resultado inteiramente conclusivo por estarmos a utilizar uma versão 3D do NDT.

4.3.1.3 Resultados relativos à velocidade de convergência

Neste ponto pode-se concluir que uma iteração do PM pode ser 72 vezes mais rápida que uma iteração do NDT e 32 vezes mais rápida que uma iteração do ICP (considerando as implementações disponíveis no PCL). Mas isto levanta outra questão, qual o número de iterações necessárias para a solução convergir, ou seja, a velocidade de convergência. Uma iteração do PM pode ter um custo computacional mais baixo, mas pode requerer maior número de iterações para convergir para uma correcta solução.

De forma a analisar a velocidade de convergência adicionou-se outro critério de paragem aos algoritmos de correspondências de mapas em análise. Este consiste em avaliar a diferença de translação e rotação entre duas iterações consecutivas. Se valor absoluto dessa diferença for abaixo de um determinado valor, o processo de optimização pára. Os valores utilizados neste critério de paragem foram de 0.01m na translação e 0.8 graus na rotação para todos os algoritmos. O cenário escolhido para este teste corresponde à situação da primeira coluna da Tabela 4.5, isto é, menos densidade de dados sensoriais, mapa de referência com menor resolução e ausência de *outliers*.

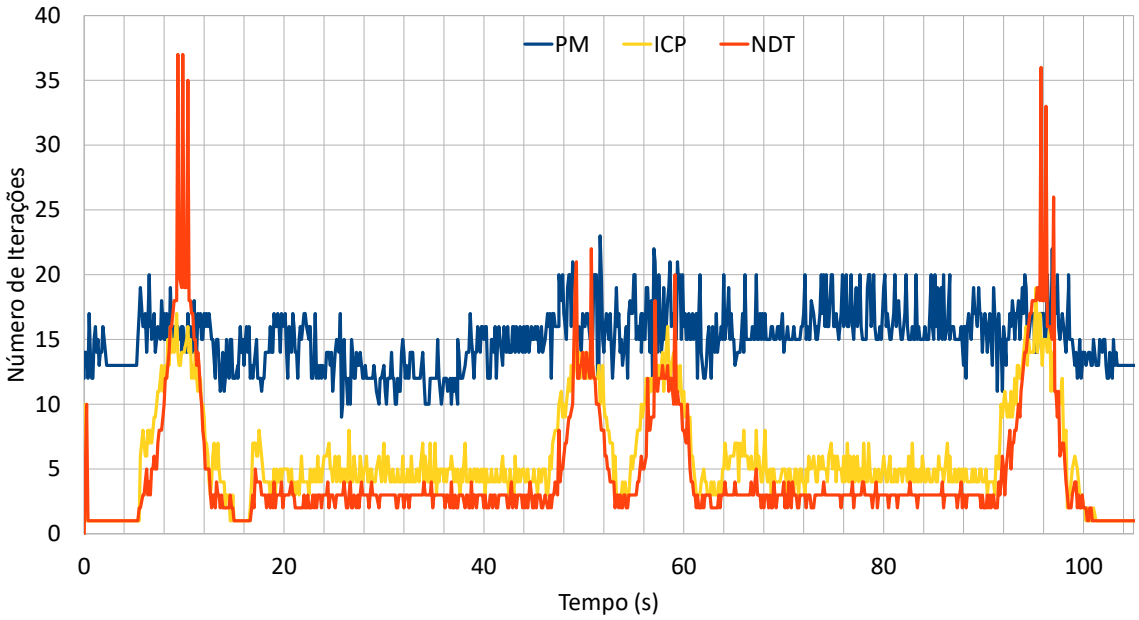


Figura 4.18: Número de iterações efectuadas pelo PM, ICP and NDT

Nas Figuras 4.18 e 4.19 apresentam-se os resultados experimentais obtidos para o PM (azul), ICP (amarelo) e NDT (laranja), enquanto o robô executa a trajectória apresentada na Figura 4.21. O gráfico da Figura 4.18 apresenta o número de iterações efectuadas pelos algoritmos ao longo tempo, enquanto a Figura 4.19 apresenta o tempo de execução gasto. Como pode ser visto na Figura 4.18, o PM precisa de mais iterações para convergir, mas mesmo assim continua a ser mais leve computacionalmente que o ICP e o NDT (Figura 4.19). Também se observa que o NDT é o algoritmo que maioritariamente converge num menor número de iterações. Tal pode dever-se ao facto do NDT se basear no método de Newton para resolver o problema de minimização, que recorre à primeira e segunda derivada da função a minimizar (*-score*), enquanto que o PM e o ICP

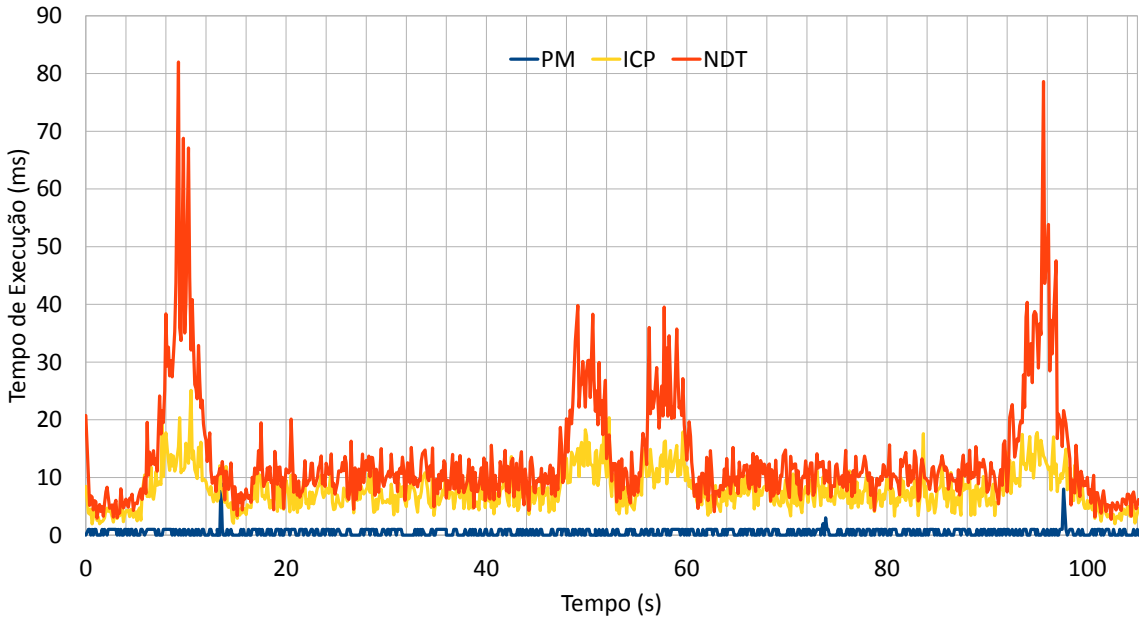


Figura 4.19: Tempo de execução do PM, ICP and NDT

recorrem apenas à primeira derivada. Além disso, durante a minimização é utilizado um tamanho de passo adaptativo de forma a garantir o decréscimo do valor do *score*. É também possível identificar na Figura 4.18 três situações: uma zona em que ICP e o NDT utilizam apenas uma iteração, e que corresponde à situação em que o robô está parado, outra zona em que o ICP e o NDT utilizam entre duas a oito iterações, e que corresponde ao caso em que o robô se está a deslocar em linha recta, e outra zona em que observa um aumento significativo do número de iterações, chegando mesmo a ultrapassar o número de iterações utilizadas pelo PM. Esta última situação corresponde à realização de curvas pelo robô. Adicionando outliers aos dados dos sensores não foram verificadas alterações significativas aos resultados apresentados.

Conclui-se, então, que o desempenho do ICP e do NDT em termos de velocidade de convergência é significativamente degradada pelo erro de rotação, ao contrário do PM. Por exemplo, aplicando um erro de rotação na pose do robô de 11.4° o ICP necessita de 46 iterações para convergir enquanto que o PM apenas necessita de 29 iterações.

Como seria de esperar, a alteração no ICP proposta na secção anterior, o ICP baseado em *lookup tables* (LUT-ICP), não tem implicações na velocidade de convergência em termos de número de iterações, mas com esta nova proposta conseguimos tempos de processamento muito mais baixos e próximos aos conseguidos pelo PM, como se pode confirmar na Figura 4.20.

4.3.1.4 Resultados relativos ao erro máximo de inicialização

Até agora os testes efectuados demonstraram que o PM é computacionalmente mais o leve que o ICP e o NDT. Nesta secção pretende-se analisar a tolerância dos algoritmos ao erro máximo de inicialização, isto é o erro máximo de pose inicial a partir do qual a solução do algoritmo de matching diverge. Com isto testa-se a robustez a mínimos locais. Nestas experiências foram

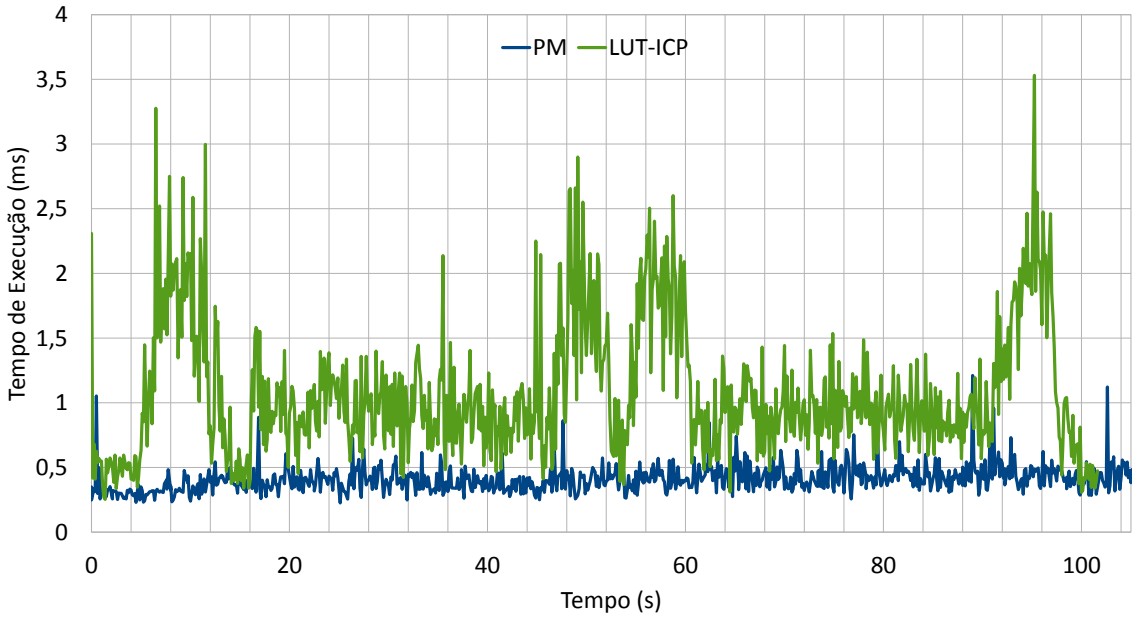


Figura 4.20: Tempo de execução do PM e o LUT-ICP

usados os mesmos parâmetros da Secção 4.3.1.2, isto é, um número fixo de iterações, a distância máxima de procura do ICP com um valor elevado e todos os filtros do NDT desligados. De forma a observar a tolerância dos algoritmos ao erro de orientação, parou-se o robô em alguns pontos da trajectória e foram reinicializados os algoritmos com poses cujo erro de orientação era gradualmente maior (1 grau de resolução), até se observar a divergência da solução obtida. Na Figura 4.21 pode-se observar o conjunto de poses onde os testes foram efectuados (setas pretas). Para cada uma dessas poses existem outros três pares de setas (vermelhas, violetas e verdes). Estas correspondem aos limites de erro de orientação nos quais o PM, o ICP e o NDT, respectivamente, ainda convergem para a solução correcta.

Analisando os resultados da Figura 4.21, é claro que o PM suporta, na maioria dos casos, um erro inicialização na rotação superior ao ICP e ao NDT. Pelo contrário, o NDT apresenta uma tolerância ao erro de ângulo consideravelmente inferior aos outros algoritmos em estudo. Também foram feitos alguns testes preliminares em relação ao erro de posição nos quais não se verificaram diferenças significativas em termos de resultados entre os algoritmos.

Um aspecto importante a referir é o forte impacto que o parâmetro *use_reciprocal_correspondences* tem no erro máximo de inicialização suportado pelo ICP. Colocando este parâmetro a verdadeiro, o desempenho é acentuadamente degradado. Foram verificadas variações no erro máximo tolerado pelo ICP superiores $\pm 40^{\circ}$ depois de alterado este parâmetro. Estas conclusões mantêm-se para o LUT-ICP no qual, como seria de esperar, as únicas diferenças observadas em relação ao ICP original é em termos de peso computacional.

Também foram feitas experiências limitando a distância máxima de correspondência do ICP (*max_correspondence_distance*), também sem melhorias. Este limite funciona como um filtro para aumentar a robustez do ICP a *outliers*. O PM tem embutido na sua função de optimização um

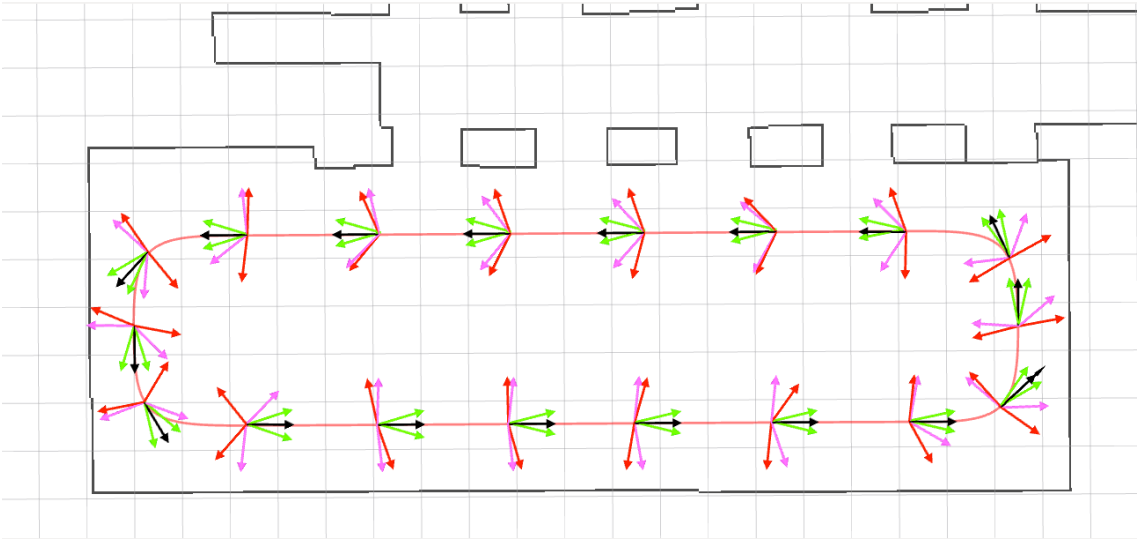


Figura 4.21: Intervalo de erro máximo de inicialização em relação à rotação suportada pelo PM (vermelho), ICP (violeta) e NDT (verde) para várias poses no mapa (preto). Também está indicado a vermelho a trajectória utilizada nos testes efectuados em simulação.

filtro de *outliers*, configurado pelo parâmetro L_c (Equação 3.19). Durante esta análise foi observado que estes parâmetros têm um impacto significativo no erro máximo de inicialização do PM e do ICP. Nesse sentido, para uma determinada pose do robô, repetiu-se o teste para vários valores de L_c e *max_correspondence_distance*. Como se pode observar na Figura 4.22, baixando o valor destes parâmetros (com isto diminuindo a sensibilidade dos algoritmos a *outliers*), o desempenho do PM e do ICP em termos de erro de inicialização é sacrificada. Contudo, este teste continua a evidenciar um desempenho superior do PM em relação ao ICP.

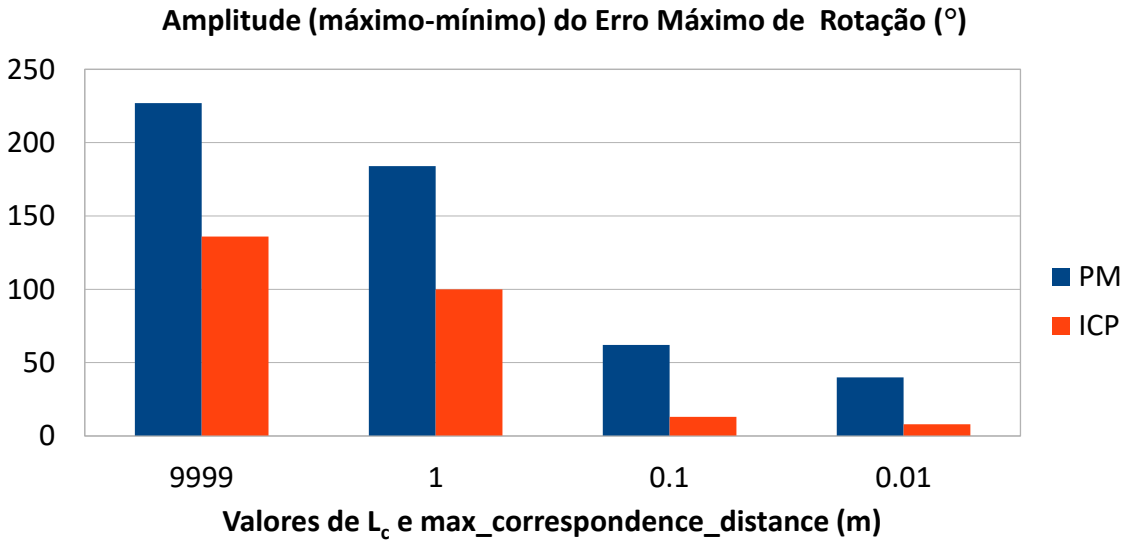


Figura 4.22: Intervalo de erro máximo de inicialização de rotação suportado pelo PM e o ICP para diferentes parâmetros de L_c e *max_correspondence_distance*.

4.3.1.5 Resultados relativos à robustez a outliers

Até esta secção é clara a vantagem do PM em relação ao peso computacional. Contudo, isto leva-nos à questão da precisão e da robustez a *outliers*. Existe um grande número de estudos nos quais se desenvolveram técnicas para aumentar a robustez dos algoritmos de matching, em particular o uso do RANSAC (Fischler e Bolles, 1981), para identificar a presença de *outliers* nos dados dos sensores. Muitos destes métodos são transversais aos algoritmos de *matching* e podem também ser usados no PM. Assim, será interessante analisar os algoritmos de *matching* sem filtros e outros artifícios e examinar se a eficiência computacional demonstrada pelo PM não é conseguida sacrificando precisão e robustez. O PM tem embutido na sua função de optimização um filtro de outliers, configurado pelo parâmetro Lc. Nestas experiências utilizou-se um Lc = 0.1 metros. No sentido de tornar esta comparação mais justa foi alterado o parâmetro correspondente à máxima distância de correspondência (*max_correspondence_distance*) para 0.1 metros. Em relação ao NDT, no parâmetro relacionado com a percentagem de *outliers* esperada nos dados sensoriais, foi utilizado o valor de 0.55.

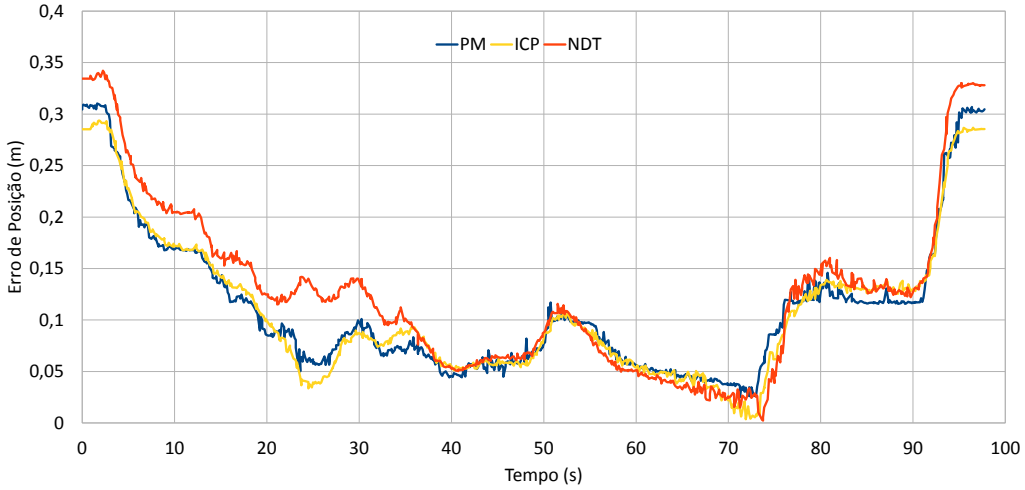


Figura 4.23: Erro de posição ao longo da trajectória com outliers (simulação).

Nas Figuras 4.23 e 4.24 encontram-se os resultados da evolução do erro de posição e orientação ao longo da trajectória apresentada na Figura 4.21. Estes resultados foram obtidos em simulação. Na Figura 4.1 pode-se observar a disposição dos *outliers* a vermelho. Apesar da forma como os algoritmos lidam com os *outliers* não ser equivalente, conclui-se que estes apresentam uma robustez similar entre si, tendo o PM ter demonstrado um desempenho superior em termos de peso computacional e tolerância a erro de inicialização nos testes anteriores. Outro facto a ter em conta é que a forma como o ICP lida com os *outliers* também é aplicável no PM. Outra observação foi que aumentando a sensibilidade dos algoritmos aos *outliers* (diminuindo o valor de L_c e *max_correspondence_distance* para o PM e o ICP, respectivamente, e subindo a percentagem de *outliers* esperados no NDT) obtêm-se, como seria de esperar, melhores resultados em termos de precisão e orientação. No entanto, como foi visto na Secção 4.3.1.4, a tolerância ao erro de

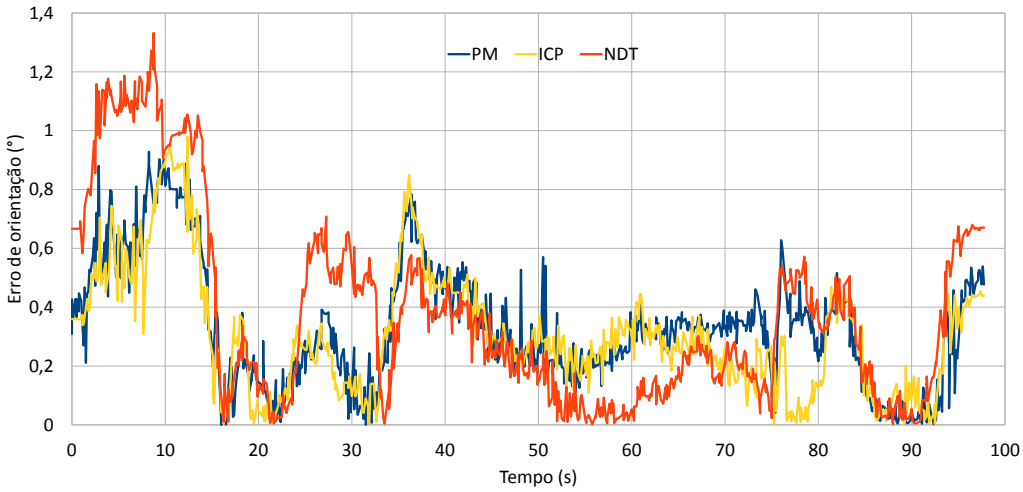


Figura 4.24: Erro de orientação ao longo da trajectória com outliers (simulação).

inicialização é diminuída. Isto evidencia a existência de um *trade-off* entre a tolerância a erros de inicialização e a robustez a *outliers*.

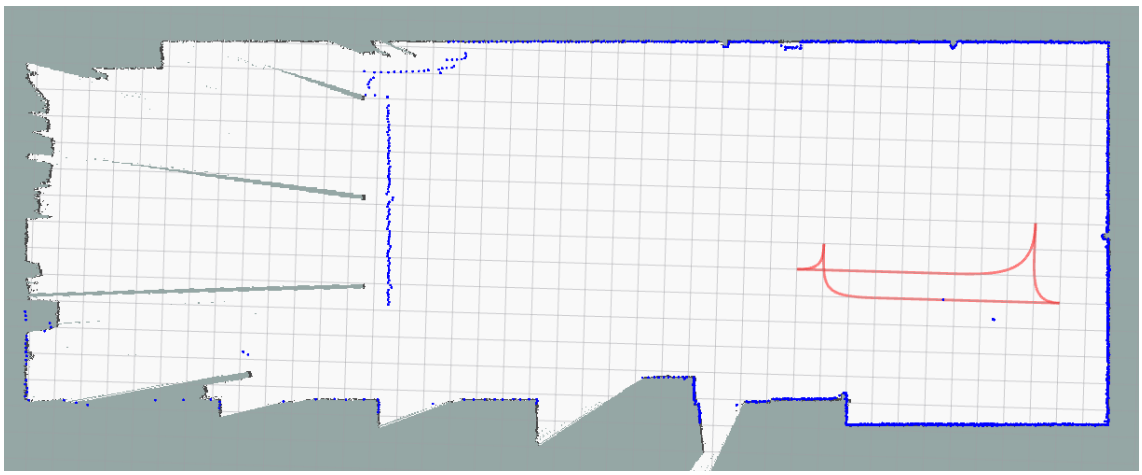


Figura 4.25: Trajectória executada pelo robô em ambiente real representada pela linha vermelha. Têm-se também os dados do *laser* a azul e o mapa de referência utilizado com uma resolução de 0.02m.

Até este ponto todas as experiências foram realizadas em simulação. Muitas vezes os simuladores não modelam detalhes importantes que podem ter um impacto significativo no desempenho dos algoritmos. Neste sentido, repetiram-se as experiências acima com o robô real descrito na secção 4.1.2. Este está equipado com um sistema comercial de triangulação laser que utiliza reflectores instalados na parede para localizar o robô (ver Secção 4.1.2.2). Este sistema é utilizado nestas experiências como *ground-truth*. Nesta experiência o robô deslocou-se a uma velocidade de 0.05m/s com um mapa com as dimensões de 20x8m, executando a trajectória apresentada na Figura 4.25. Foi utilizada uma velocidade reduzida para diminuir a distorção das medidas adquiridas pelo laser de navegação. Esta distorção é mais acentuada durante os movimentos de rotação

do robô. Contudo, é facilmente compensada por software, recorrendo a sensores de movimento com uma frequência de aquisição mais rápida que a do laser de navegação (8 Hz), como é o caso de *encoders*. De forma a simplificar a experiência, foi preferida a utilização de velocidades mais baixas em detrimento de módulos extra de software. Na Figura 4.25 mostra-se um exemplo de uma amostragem do *laser scanner* a azul em que é visível a presença de *outliers*.

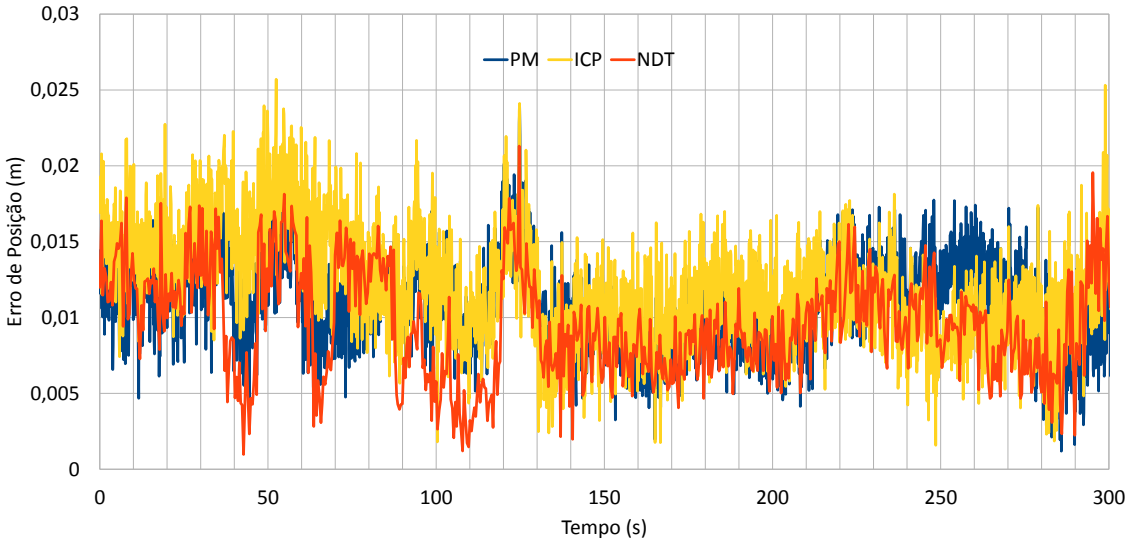


Figura 4.26: Erro de posição ao longo da trajectória - robô real

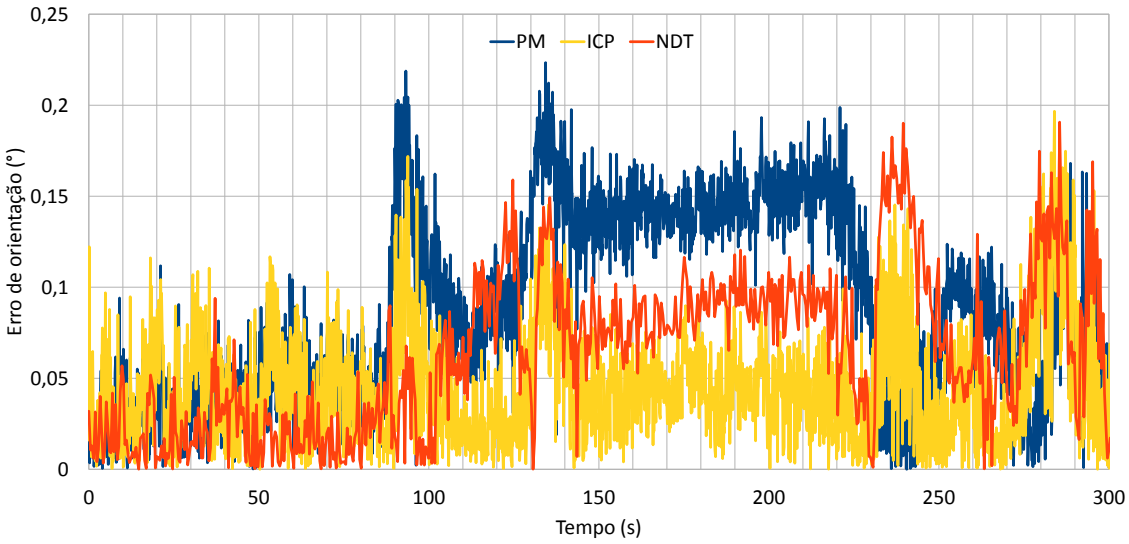


Figura 4.27: Erro de orientação ao longo da trajectória - robô real

As Figuras 4.26 e 4.27 apresentam os resultados referentes à análise de precisão em ambiente real com a natural presença de *outlers*. Confirma-se que os algoritmos em análise apresentam uma robustez similar face ao ruído presente nas medidas adquiridos pelos sensor do robô. Foram também repetidos os testes anteriores em simulação e as observações e conclusões confirmaram-se no robô real.

A Tabela 4.6 resume em termos de média e desvio padrão os testes de precisão realizados em simulação e no robô real. Observa-se que todos os algoritmos obtiveram resultados semelhantes em termos de precisão de posição e orientação. Neste ponto é possível concluir que o PM é equivalente ao ICP e NDT em termos de precisão, mas oferece maior robustez face a erros de inicialização com um custo computacional menor, deixando tempo disponível para filtros mais avançados que aumentem a eficiência dos algoritmos de localização que usem o PM.

Tabela 4.6: Média e desvio padrão dos erros de posição e orientação obtidos em simulação e em ambiente real com a presença de *outliers*.

	Simulação				Robô Real			
	Erro de Posição (m)		Erro de Orientação (°)		Erro de Posição (m)		Erro de Orientação (°)	
	Média	Des. Padrão	Média	Des. Padrão	Média	Des. Padrão	Média	Des. Padrão
PM	0.112	0.0715	0.333	0.207	0.0110	0.00310	0.0889	0.0521
ICP	0.110	0.0704	0.305	0.209	0.0120	0.00384	0.0448	0.0311
NDT	0.127	0.0819	0.378	0.325	0.00951	0.00357	0.0640	0.0428

No decorrer destas experiências observou-se que, em condições ideais, isto é, em simulação (sem ruído nas medidas e sem *outliers*), a precisão do NDT é inferior à do PM e ICP. Utilizando o mapa da Figura 4.21, o PM e o ICP apresentam valores de erro de posição inferiores a 1cm, enquanto que o NDT apresenta erros de 1.8cm. Esta diferença no erro deve-se, provavelmente, ao erro de discretização inerente ao NDT, no qual o mapa de referência é aproximado a um conjunto de distribuições normais. Além disso, o mapa da Figura 4.21 favorece o NDT, pois este é composto maioritariamente por linhas rectas.

Repetiu-se a experiência para o caso em que o ambiente é irregular (sem paredes rectas) e observou-se que enquanto no PM e no ICP o erro de posicionamento não é afectado, no NDT chegou aos 16cm. Ou seja, a forma do mapa referência influencia a precisão do NDT.

4.3.2 Análise de desempenho da solução de Pose Tracking desenvolvida

Na secção anterior efetuaram-se diversas experiências que evidenciaram o bom desempenho do *Perfect Match* (PM) em termos computacionais, robustez a erros de inicialização e robustez a *outliers*. O *Perfect Match* foi utilizado com sucesso por diversas equipas de futebol robótico e, com as alterações propostas, pretende-se aumentar a sua precisão de forma a estender a sua aplicação a outros domínios, como por exemplo AGVs industriais.

Nesta secção apresentam-se os resultados obtidos com os desenvolvimentos ao nível da solução de *Pose Tracking* descritos na Secção 3.2.2. Foi feita uma análise que procurou evidenciar o impacto no desempenho em relação ao PM original. Esta análise recorreu ao *setup* experimental da secção anterior e foi analisado o impacto das alterações efetuadas ao *Perfect Match* nomeadamente no peso computacional, na tolerância ao erro de inicialização e na robustez a *outliers*. Também se apresentam alguns casos em concreto e descrevem-se demonstrações públicas nas quais foram utilizados os desenvolvimentos apresentados neste trabalho.

Em relação ao peso computacional, o impacto dos desenvolvimentos efectuados aumentaram, como seria de esperar, o peso computacional para o dobro. Esta duplicação deve-se à execução do módulo "RPROP Optimization"duas vezes, como se pode pode observar na Figura 3.11. Como este módulo corresponde à aplicação do PM, o número de iterações é duplicado. No entanto, graças à eficiência do PM, este aumento de peso computacional não tem um impacto relevante.

Como trabalho futuro, pretende-se analisar a influencia do valor do passo inicial na optimização (Δx_{init}, Δy_{init} e $\Delta\theta_{init}$, Linha 2 do Algoritmo 7) no número de iterações necessárias.

Utilizando o *setup* experimental da Secção 4.3.1.4 e 4.3.1.5, analisou-se o impacto das alterações efectuadas ao PM em termos de tolerância ao erro máximo de inicialização e robustez a *outliers*. Utilizaram-se os valores de 1.0m, 0.1m para $MaxError1$ e $MaxError2$ respectivamente para os parâmetros do Algoritmo 6 e um L_c igual a 1m (Equação 3.19).

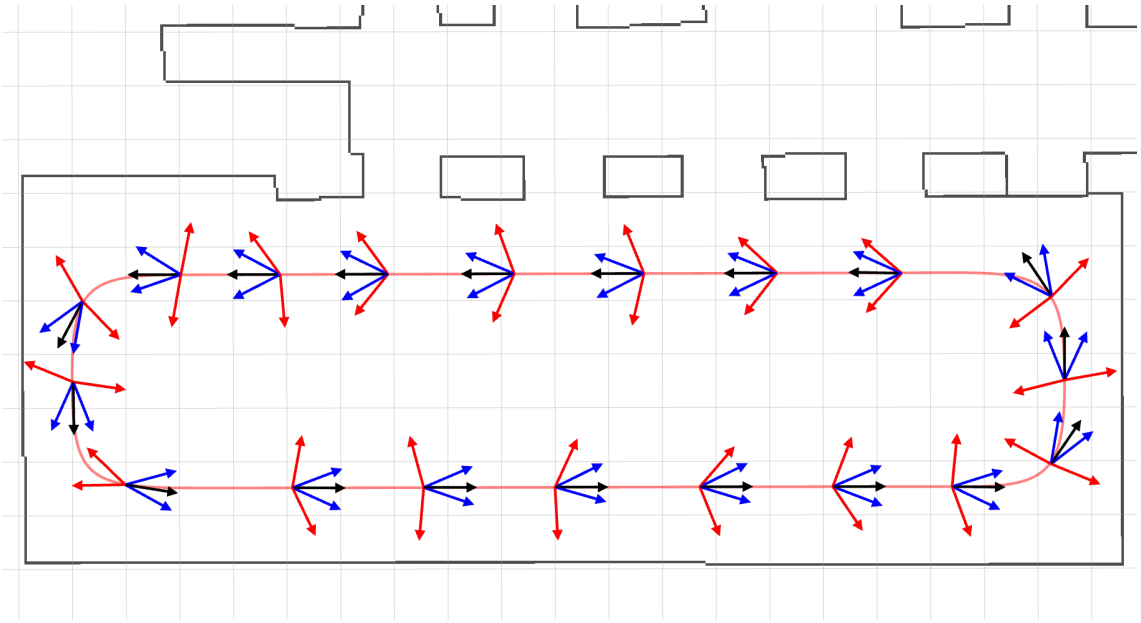


Figura 4.28: Intervalo de erro máximo de inicialização em relação à rotação suportada pelo Perfect Match original (vermelho) e pela nova abordagem proposta (azul) para várias poses no mapa (preto). Também está indicado a vermelho a trajectória utilizada nos testes efectuados em simulação.

A Figura 4.28 evidencia o impacto das alteração ao PM em termos de tolerância ao erro de inicialização angular. Observa-se uma diminuição significativa da tolerância ao erro de inicialização: nesta experiência, obteve-se uma amplitude do erro máximo de rotação de cerca de 40° no PM alterado ao longo das várias poses testadas, enquanto que o PM original apresenta valores superiores a 90°. Contudo, o valor de 40° é suficiente para a maioria das aplicações, especialmente nas industriais, nas quais não se esperam erros de inicialização elevados. As causas principais dos erros de inicialização são a inerente acumulação de erro da odometria entre ciclos do sistema de localização, ou a derrapagem das rodas do robô. Nos AGVs industriais, os sistemas de localização trabalham a frequências elevadas e a derrapagem das rodas é uma situação atípica. Como já

foi referido, existe um compromisso entre tolerância a erros de inicialização e robustez a *outliers*, e tipicamente os requisitos em termos de precisão para aplicações industriais são elevados. Relembra-se que a abordagem proposta está configurada para considerar, *outliers* erros superiores a 0.1m ($MaxError2$); ora observando os resultados apresentados na Figura 4.22, na situação mais equivalente (L_c e $max\_corresponde\_distance$ iguais a 0.1m), o PM original e o ICP obtiveram tolerâncias de 62° e 13°, respectivamente.

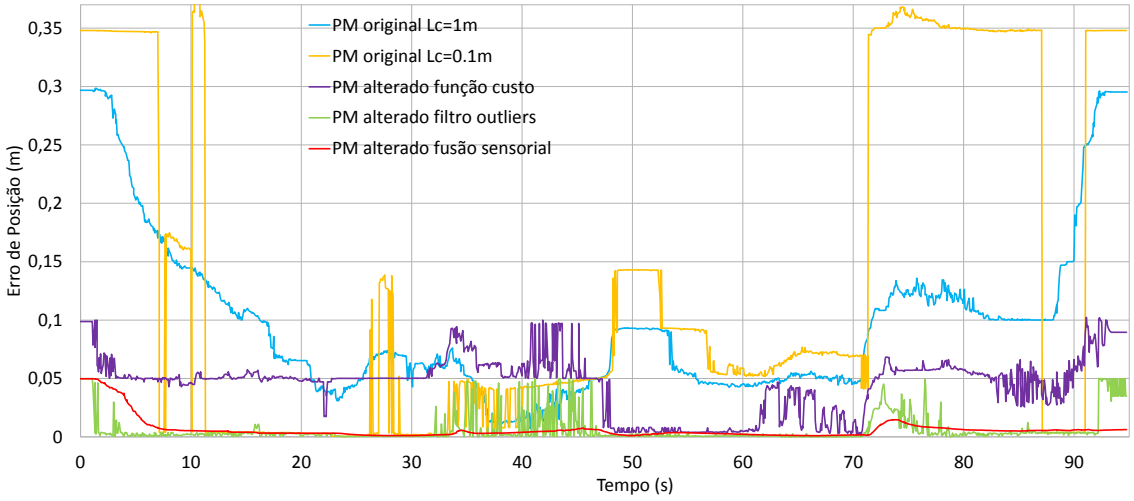


Figura 4.29: Erro de posição ao longo da trajectória com *outliers* (simulação). Aqui analisa-se o impacto na robustez a *outliers* das alterações propostas ao PM.

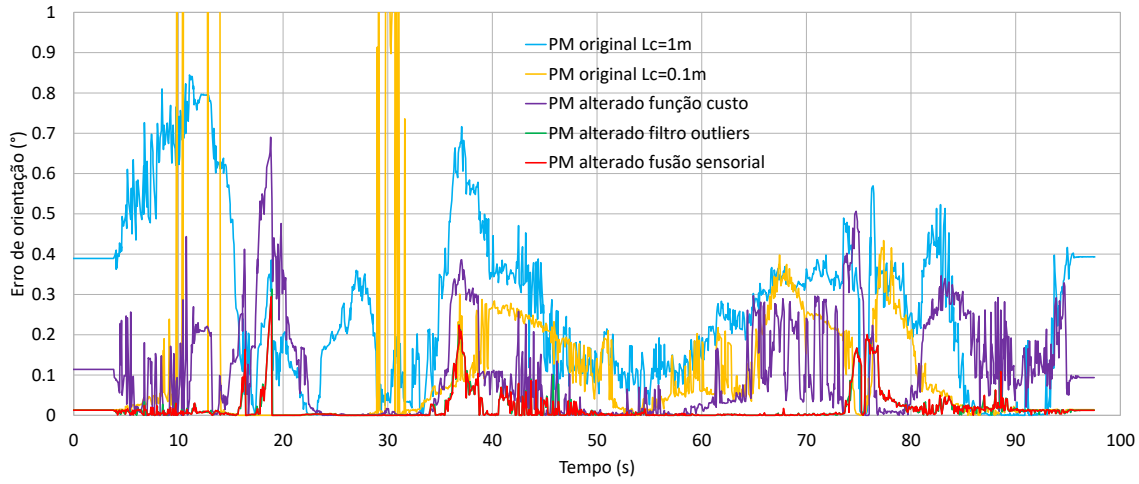


Figura 4.30: Erro de orientação ao longo da trajectória com *outliers* (simulação). Aqui analisa-se o impacto na robustez a *outliers* das alterações propostas ao PM.

Os resultados apresentados nas Figuras 4.29 e 4.30 evidenciam as melhorias de precisão conseguidas em relação ao PM original. Apresentam-se várias configurações tanto para o PM original como para o proposto neste trabalho de forma a evidenciar o impacto das melhoria propostas. Analisou-se a resposta do PM original para dois valores de L_c de 1m e 0.1m respectivamente. E em

relação às melhorias propostas, estas vão sendo activadas progressivamente, e de uma forma acumulativa. Ou seja, em relação ao algoritmo original, quando se introduz uma alteração mantém-se as introduzidas anteriormente. Testa-se uma configuração só com a alteração da função custo, outra em que é adicionado os filtros de *outliers* e por sim adiciona-se a fusão com a odometria através de um filtro de Kalman. Os resultados referentes às propostas de alteração do PM utilizam um valor de L_c de 1m. A Figura 4.29 apresenta o erro de posição e a Figura 4.30 o erro de orientação obtido utilizando o *setup* utilizado na secção 4.3.1.5. São resultados obtidos recorrendo ao simulador descrito na secção 4.1.1, na Figura 4.1 têm-se a distribuição dos *outliers* e a trajectória efectuada está representada na Figura 4.28.

Nas Figuras 4.29 e 4.30 a linha azul e laranja apresentam o erro de posição e orientação do PM original com um L_c de 1m e 0.1m respectivamente. Observa-se que o filtro integrado na função custo do PM não consegue por si só filtrar eficientemente os *outliers* em todas as situações e existem casos em que o PM é mais afectado pelos *outliers* com um valor de L_c inferior. Um dos casos mais desfavorável ao PM (e também aos outros algoritmos de *Map Matching* analisados) é o representado na Figura 4.1, aqui existe um *outlier* próximo do robô e este encontra-se próximo de uma zona ocupada no mapa de ocupação (está próximo de uma parede). Estando o *outlier* próximo do robô este têm mais peso na optimização por ter mais medidas associadas e estando próximo de uma zona ocupada no mapa torna mais complicado a sua distinção dos *inliers*.

A linha violeta das Figuras 4.29 e 4.30 corresponde ao erro de posição e orientação obtido apenas com a alteração da função custo 3.17 descrita na secção 3.2.2.2. Esta alteração consiste em tornar o peso de cada medida proporcional à sua distância do sensor. Com esta alteração observa-se uma melhoria da precisão do PM na grande maioria das situações.

A verde temos o erro de posição (Figura 4.29) e orientação (Figura 4.30) obtido activando os filtro de *outliers* observa-se uma clara melhoria do desempenho do PM em termos de precisão. Realça-se que a disposição escolhida para os *outliers* (Figura 4.1) resulta de várias experiências em que foram testadas diferentes disposições em que procurava uma situação desafiadora para os sistemas de localização em termos de presença de *outliers*.

Por fim, a vermelho, temos o erro de posição (Figura 4.29) e orientação (Figura 4.30) obtidos adicionando a fusão sensorial com a odometria (Algoritmo 5). Nesta configuração, estão activas todas as alterações propostas ao PM em simultâneo. Em relação à configuração anterior, é notório que o filtro de kalman remove a instabilidade da resposta do algoritmo de Matching que se observava nalguns pontos da trajectória. Na tabela 4.7 apresentam-se os erros médios, desvios-padrão e erro máximo obtidos nesta experiência:

Até este ponto foi demonstrado em simulação o impacto das alterações efectuadas ao PM. Foi sacrificada alguma tolerância aos erro de inicialização, mas conseguiu um aumento significativo da precisão. Apenas com a alteração da função custo, em que o peso de cada medida na optimização é proporcional à sua distância, obtêm-se melhorias significativas. Isto, em conjunto com o esquema de filtragem de *outliers*, leva a um bom desempenho em termos de precisão.

No entanto, existem características significativas do ambiente que não são modeladas em simulação, nomeadamente a influência da distância dos objectos no erro de medida dos contornos.

Tabela 4.7: Caracterização dos erros de posição e orientação obtidos em simulação com a presença de *outliers*. Foram analisadas diferentes configurações. É apresentado o erro médio, desvio padrão e erro máximo.

	Erro de Posição (m)			Erro de Orientação (°)		
	Média	Des. Padrão	Máximo	Média	Des. Padrão	Máximo
PM original Lc=1m	0.110	0.064	0.299	0.279	0.159	0.845
PM original Lc=0.1m	0.150	0.137	0.484	0.164	0.189	4.196
PM alterado função custo	0.049	0.019	0.102	0.109	0.093	0.690
PM alterado filtro outliers	0.009	0.012	0.050	0.017	0.020	0.313
PM alterado fusão sensorial	0.009	0.009	0.050	0.016	0.019	0.294

Desníveis acentuados do chão aliados à presença de objectos verticalmente irregulares podem também afectar significativamente a qualidade sensorial das medidas dos contornos.

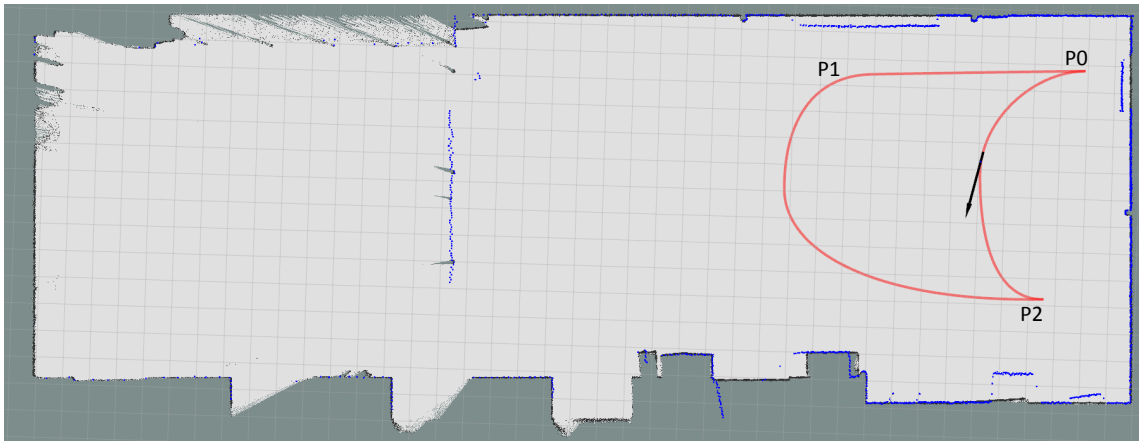


Figura 4.31: Trajectória (a vermelho) e mapa de ocupação (a preto) durante os testes com dados reais. Os pontos azuis representam as medidas adquiridas pelo *laser scanner* quando o robô se encontra na pose indicada pela seta preta. Cada célula da grelha tem 1 metro de lado.

Neste sentido, à semelhança da secção 4.3.1.5, repetiram-se os testes de análise de precisão com dados reais. Recorrendo à plataforma robótica descrita na Secção 4.1.2 a resposta do sistema de triangulação laser é comparada as diferentes configurações do PM anteriormente analisadas em simulação. Nesta experiência foram acrescentados objectos de forma a aumentar o número de *outliers* e evidenciar o impacto das melhorias propostas neste trabalho. Na Figura 4.31 têm-se indicado a vermelho a trajectória efectuada pelo robô durante esta experiência. O robô arranca de P0 passando por P1 até P2; aqui inverte o sentido de marcha e volta até P0. Além disto, a azul, têm-se representado uma aquisição do *laser scanner* quando o robô se encontra na pose indicada pela seta preta. Através da análise da sobreposição dos dados do *laser scanner* com o mapa de ocupação (a preto) é possível inferir acerca da disposição dos *outliers*. A resolução do mapa de ocupação utilizado é de 1cm.

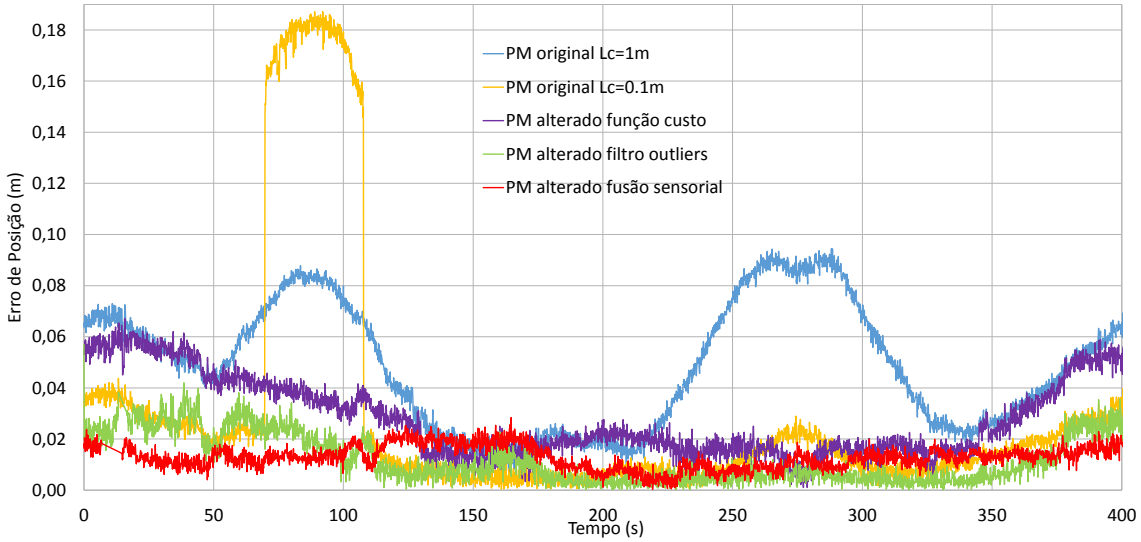


Figura 4.32: Erro de posição ao longo da trajectória com *outliers* (robô real). Aqui analisa-se o impacto na robustez a *outliers* das alterações propostas ao PM.

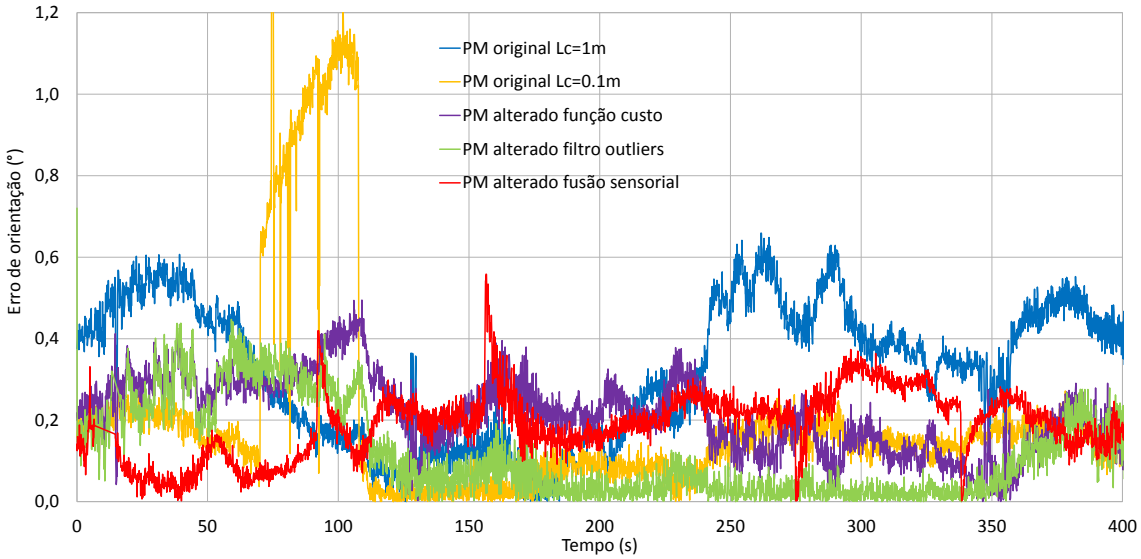


Figura 4.33: Erro de orientação ao longo da trajectória com *outliers* (robô real). Aqui analisa-se o impacto na robustez a *outliers* das alterações propostas ao PM.

As Figuras 4.32 e 4.33 apresentam o erro de posição e orientação em função do tempo enquanto o robô executa a trajectória indicada na Figura 4.31, das várias configurações do PM em análise. Como se pode observar, os resultados estão de acordo com as conclusões obtidas nos testes anteriores, realizados em ambiente de simulação. A linha a azul, correspondente à implementação original do PM com $L_c = 1m$, verifica-se uma degradação significativa da precisão devido à presença de *outliers*.

A alteração do parâmetro L_c para 0.1m (linha a amarelo) aumenta, de uma forma generalizada, a precisão do PM; no entanto, em situações pontuais, a precisão é gravemente afectada, atingindo

erros superiores a 18cm e erros de orientação superiores a 1 grau. Utilizando apenas a nova função custo (Equação 3.17) proposta neste trabalho (linha violeta) observa-se um aumento da robustez para a grande maioria das situações em relação à implementação original com o mesmo $L_c = 1m$ (linha azul). O impacto na precisão da filtragem de *outliers* é evidente por comparação entre a linha verde e a linha violeta em que se verifica uma clara diminuição do erro de posição e orientação, em particular nas situações pontuais anteriormente referidas.

A linha a vermelho corresponde ao erro obtido activando a fusão sensorial entre os dados do *laser scanner* e a odometria. Neste último, observa-se em determinados momentos um pequeno aumento do erro de posição e de orientação. Esta situação corresponde aos momentos em que a velocidade angular do robô é superior. Acredita-se que o motivo desta pequena deterioração esteja relacionado com erros no modelo de odometria. Contudo, a utilização de fusão sensorial permite suavizar a resposta do sistema de localização, e aumenta a robustez global do sistema, especialmente no caso de falhas grosseiras dos sensores. Na Tabela 4.8 apresenta-se a caracterização estatística dos dados apresentados nas Figuras 4.32 e 4.33 constituída pelo erro médio, desvio padrão e erro máximo observados tanto para a posição como para a orientação. Como se pode concluir, estes dados são consistentes com os dados presentes na Tabela 4.7 com a excepção da pequena subida do erro médio na experiência correspondente à utilização da fusão sensorial, pelos motivos já referidos. No entanto apesar do erro médio subir ligeiramente o desvio padrão desce e o erro máximo também desce, quer em termos de posição quer em termos de orientação.

Tabela 4.8: Caracterização dos erros de posição e orientação obtidos com dados reais com a presença de *outliers*. Foram analisadas diferentes configurações. É apresentado o erro médio, desvio padrão e erro máximo.

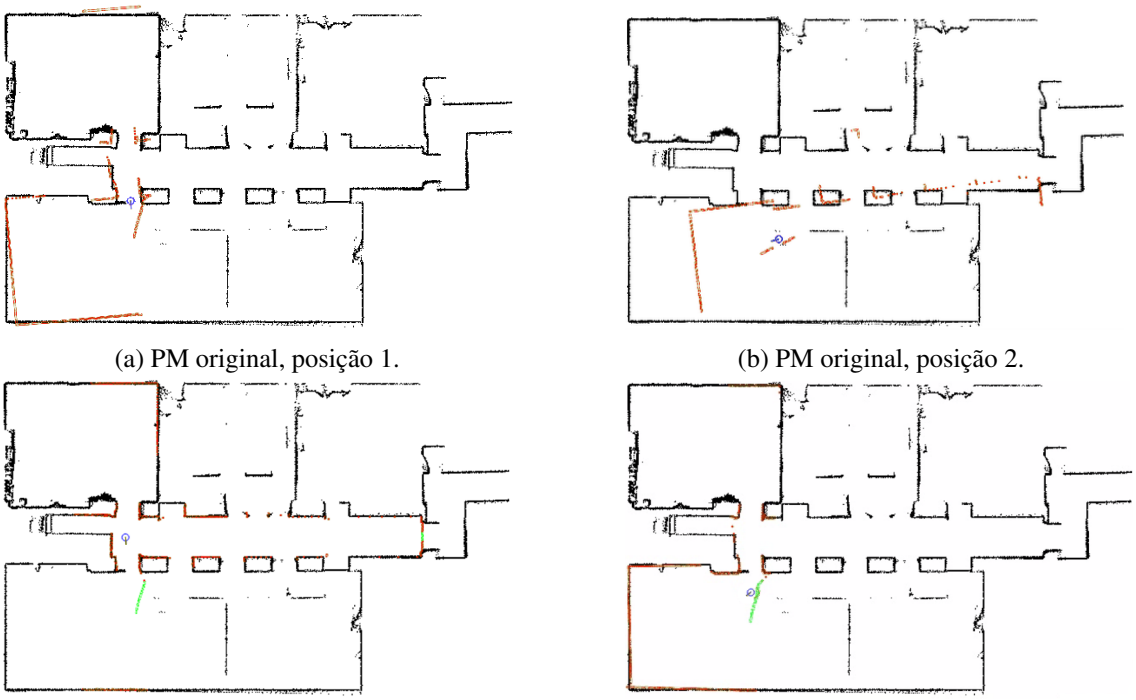
	Erro de Posição (m)			Erro de Orientação (°)		
	Média	Des. Padrão	Máximo	Média	Des. Padrão	Máximo
PM original Lc=1m	0,0486	0,0209	0,0945	0,3216	0,1456	0,6592
PM original Lc=0.1m	0,0300	0,0280	0,1872	0,2021	0,1431	1,3910
PM alterado função custo	0,0281	0,0134	0,0671	0,2156	0,0750	0,4949
PM alterado filtro outliers	0,0120	0,0082	0,0570	0,1195	0,0955	0,7202
PM alterado fusão sensorial	0,0129	0,0036	0,0284	0,1878	0,0642	0,5583

Com a plataforma robótica descrita na Secção 4.1.2 foi também implementado um cenário de demonstração com o objectivo de facilitar testes ao sistema em diferentes ambientes e melhorar a divulgação externa dos desenvolvimentos realizados no âmbito deste trabalho. O cenário definido exige ao robô que consiga carregar/descarregar uma caixa numa posição pré-definida. Realçamos que o robô não tem sensores que detectem a posição da caixa. Deste modo, o sistema de navegação tem de ter precisão/repetibilidade suficiente (na ordem dos milímetros) para parar na posição/orientação certa. Foram feitos testes em diferentes ambientes, cada um destes com características próprias que permitiram testar a robustez do sistema desenvolvido a várias adversidades. As características em cada demonstração foram as seguintes:

- Laboratório de robótica: Paredes próximas, pouco dinâmico;

- Exposição EMAF na Exponor: Ambiente amplo (136x98 metros);
- Mostra da Universidade do Porto 2013: Ambiente amplo (170x98 metros) com piso irregular, obstáculos à altura do laser de navegação;
- Festival Nacional Robótica 2013: Paredes em vidro não detectáveis pelo laser de navegação.

Em todos estes cenários o sistema provou ter a robustez necessária para realizar com sucesso a demonstração acima descrita. Realça-se que, durante estes eventos, foi recolhida uma grande quantidade de dados e foram testadas diferentes configurações do sistema, que demonstraram a importância das melhorias propostas, visto que desactivando algumas delas (alteração da função custo, filtro de *outliers* ou o filtro de Kalman) a repetibilidade do sistema de navegação degrada-se, não permitindo as operações de carga/descarga.



(a) PM original, posição 1.

(b) PM original, posição 2.

(c) PM alterado, posição 1.

(d) PM alterado, posição 2.

Figura 4.34: Resposta do sistema de *Pose Tracking* a um *dataset* com *outliers*. Nas Figuras 4.34a e 4.34b recorre-se ao PM original e o sistema falha. Nas Figuras 4.34c e 4.34d, utilizando filtros, os *outliers* são ignorados, prevenindo a falha do sistema. A vermelho estão representadas as medidas utilizadas, a verde as medidas ignoradas e a azul a pose obtida.

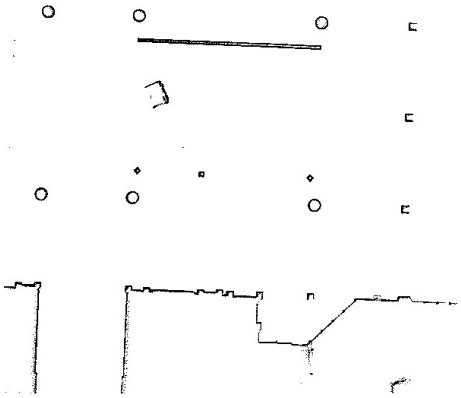
Em relação ao testes no Laboratório de robótica, apresenta-se um caso especifico em que as melhorias propostas neste trabalho previnem a falha do sistema de localização. Nas imagens da Figuras 4.34, apresenta-se a resposta do sistema de localização a um *dataset* com a presença de um *outlier* próximo do robô. Na Figura 4.34a, utiliza-se o PM original, observando-se uma má correspondência entre os dados do laser (a vermelho) e o mapa. A Figura 4.34b corresponde à mesma situação, alguns segundos depois, e devido à presença do *outlier*, o sistema *pose tracking*

falha, divergindo da solução correcta. As Figuras 4.34c e 4.34d demonstram a resposta PM com as melhorias propostas neste trabalho. As medidas a verde correspondentes ao *outlier* são ignoradas, prevenindo a falha do sistema de *Pose Tracking*.

Nos próximos links podem visualizar-se os testes realizados Laboratório de Robótica, e a caso retratado na Figura 4.34.

- Operação do robô sem interferências externas: `https://youtu.be/SaMYuX3-9hE`;
- Operação do robô com a interferência externa: `https://youtu.be/-yY6Q_zbFII`;
- Interface do sistema de localização sem melhorias: `https://youtu.be/VKE_RW5vDEU`;
- Interface do sistema de localização com melhorias: `https://youtu.be/_gHw1y_oiXE`.

(a) Ambiente de operação.



(b) Mapa de ocupação adquirido. Dimensão: 136 x 98m.

Figura 4.35: Demostração na EMAF.

A Figura 4.35a foi obtida durante os testes efectuados na feira internacional EMAF, e a Figura 4.35b apresenta o mapa de ocupação utilizado pelo sistema de localização. A especifidade deste ambiente em relação ao Laboratório é o facto de ser um ambiente amplo. Durante este evento, o robô navegou sem falhas durante vários dias.

A Mostra da Universidade do Porto (Figura 4.36) foi o ambiente mais desafiante pois, além de ser muito amplo, o chão era bastante irregular. Além disso, este evento realizou-se dentro de um pavilhão desportivo no qual existem bancadas ao nível do laser de navegação fazendo com que os contornos de objectos distantes fossem significativamente afectados. Aqui, as melhorias efectuadas no PM tiveram um papel crítico no sentido em que a desactivação de apenas uma delas resultava na falha do sistema em realizar a operação de carga/descarga da caixa. No seguinte link pode ser visualizada a demostração efectuada no ambiente acima descrito:

- Operação do robô na Mostra Universidade do Porto: `https://youtu.be/KLeploB4p60`.

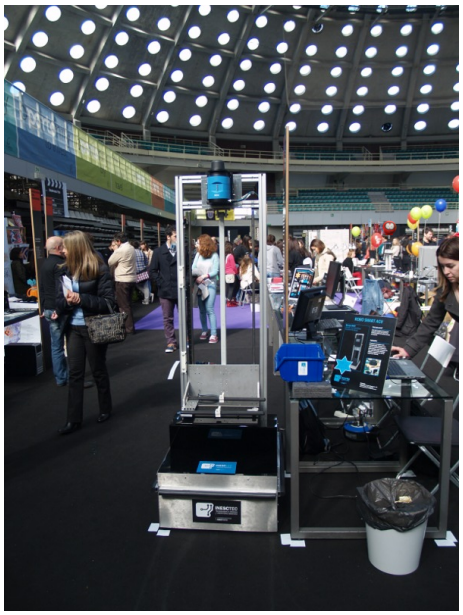


Figura 4.36: Demonstração Mostra da Universidade do Porto 2013.

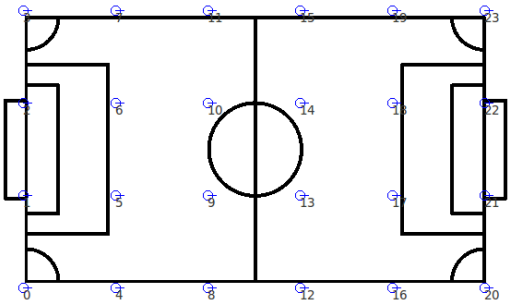
Os resultados obtidos evidenciam um aumento significativo da robustez a *outliers* de um algoritmo de *Pose Tracking* baseado em contornos naturais, o que torna promissora a sua aplicação a robôs industriais, nos quais os requisitos de precisão são geralmente muito exigentes.

4.3.3 Análise de desempenho da solução Localização Global

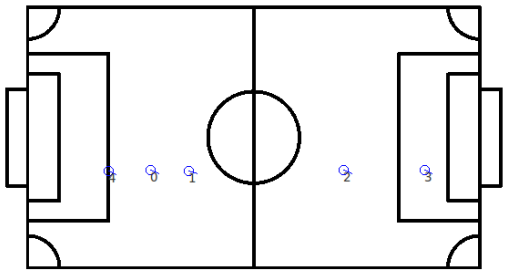
Nesta secção, apresentam-se os testes realizados com o algoritmo de Localização Global descrita na Secção 3.2.2.3. O método de Localização Global é testado em dois ambientes distintos, nas competições de Futebol Robótico (ver Secção 4.1.4) e no robô de segurança chamado robô vigilante (ver Secção 4.1.5). Em ambos os casos os robôs estão equipados com bússolas magnéticas. Contudo estes dois ambientes têm especifidades próprias. Os robôs de futebol robótico utilizam visão artificial para detectar as linhas de um campo de futebol enquanto que o robô vigilante utiliza um *laser scanner* para detectar os contornos dos objectos circundantes.

4.3.3.1 Futebol Robótico

As experiências nesta secção foram efectuadas tele-operando um robô da equipa de futebol 5dpo. Contudo este esquema de localização global é utilizado durante os jogos oficiais, e têm-se revelado uma mais valia no sentido em que a falha dos sistemas de localização é relativamente frequente. A causa destas falhas está relacionada com colisões frequentes e com as elevadas velocidades com que estes robôs se deslocam, o que aumenta a probabilidade de derrapagem das rodas e consequente ocorrência de erros grosseiros na odometria. Além disto, devido a variações bruscas da iluminação, também relativamente frequente, ocorrerem falhas no sistema de visão responsável por detectar os contornos do campo.

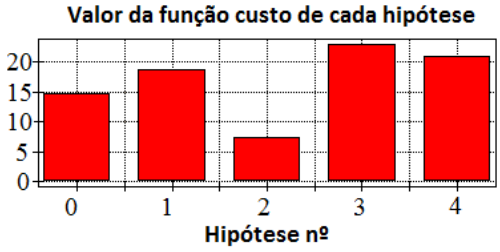


(a) Estado inicial.



(b) Estado final antes do processo de localização global terminar.

Figura 4.37: Distribuição das hipóteses durante a localização global no caso do futebol robótico.



(a) Valor da função custo.

(b) Pontuação.

Figura 4.38: Custo e pontuação das hipóteses no momento em que processo de localização global termina. Ambiente de teste: Futebol Robótico.

A Figura 4.37a mostra o início do processo de Localização Global, no qual múltiplas hipóteses são distribuídas uniformemente ao longo do mapa a uma distância fixa de 2m. A Figura 4.37b apresenta o conjunto de hipóteses válidas imediatamente antes do processo de Localização Global terminar e a posição absoluta do robô ser determinada. O Algoritmo começou com 24 hipóteses mas através do processo de eliminação de hipóteses inválidas e fusão de hipóteses repetidas apresentado na Secção 3.2.2.3 ele estabiliza com apenas 5 hipóteses. Nas Figuras 4.38a e 4.38b, encontramos os gráficos correspondentes ao estado do algoritmo no momento correspondente à situação apresentada na Figura 4.37b. A gráfico da Figura 4.38a apresenta o valor da função custo de cada hipótese, e na Figura 4.38b têm-se a pontuação atribuída a cada hipótese. O custo da hipótese número 2 obedeceu à Condição 4, Secção 3.2.2.3, h_{score} vezes, o que fez com que esta atingisse a Condição 5 da Secção 3.2.2.3. Com isto, a condição de paragem do Algoritmo é atingida e a hipótese número 2 é a escolhida para inicializar o sistema de *Pose Tracking*. Neste caso em particular, foi a hipótese número 2 que cumpriu sempre a Condição 4, mas como se verá mais à frente, isto nem sempre acontece. A linha azul da Figura 4.39 (à esquerda) apresenta-se a trajectória efectuada pelo robô durante o processo de Localização Global.

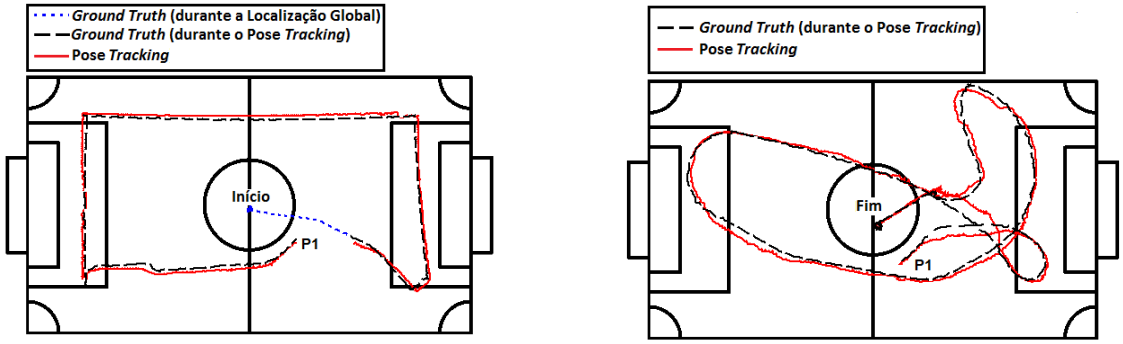


Figura 4.39: Trajectória efectuada nos testes à Localização Global no robô de futebol robótico. Esta está dividida nas duas imagens acima de forma a facilitar a visualização. O Trajecto começa na posição "Início", passa por "P1"e termina em "Fim". Nas imagens está assinalada a posição obtida pelo módulo de *Pose Tracking* (linha tracejada vermelha) e *Ground Truth* (linha tracejada preta).

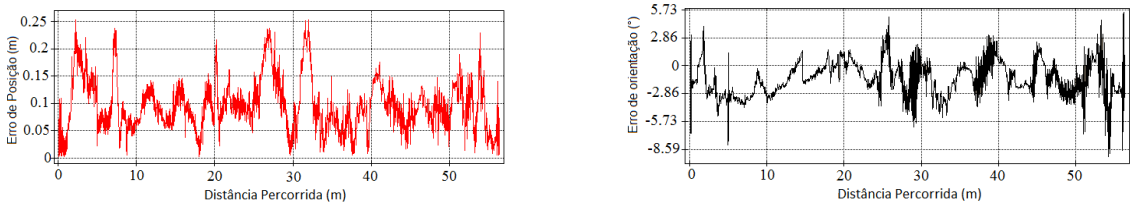


Figura 4.40: Erro de posição (esquerda) e orientação (direita) do sistema de *Pose Tracking* em função da distância percorrida durante os testes efectuados no robô de futebol robótico.

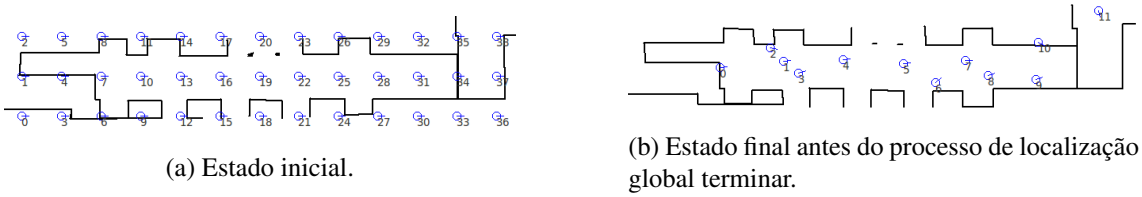
Nesta experiência também foi analisada a precisão conseguida pelo módulo *Pose Tracking* dos robôs de futebol. A pose obtida a partir da detecção das linhas do campo de futebol foi comparada com a pose obtida pelo sistema de triangulação laser descrito na Secção 4.1.2.2. A trajectória efectuada durante esta experiência está dividida pelas imagens da Figura 4.39. O robô inicia o movimento na posição assinalada por "Início", utilizando o módulo de Localização Global durante o percurso representado pela linha azul. Depois de encontrada a sua pose, passa a utilizar o módulo *Pose Tracking* cuja posição medida está sinalizada pela linha vermelha e desloca-se até à posição sinalizada por P1 e finalmente desloca-se para a posição sinalizada por "Fim". A linha preta representa as posições medidas pelo sistema de triangulação laser (*Ground Truth*). Os gráficos da Figura 4.40 apresentam o erro de posição e ângulo do sistema de localização do robô no modo de *Pose Tracking* em função da distância percorrida ao longo da trajectória. Obteve-se um erro médio de posição de $0.077m$ com um desvio padrão de $0.089m$, e um erro médio de orientação de $1.82°$ com um desvio padrão de $1.81°$. Estes valores de precisão são suficientes para que o robô navegue e realize a missão para que foi concebido, ou seja, jogar na liga de Futebol robótico de forma autónoma.

Como já foi referido, um dos pontos fortes da metodologia de localização adotada é o seu baixo peso computacional. Na plataforma utilizada na equipa de futebol robotico 5DPO, o tempo médio de execução do módulo de Pose Tracking é de 2ms para processar 50 pontos do sistema de visão.

O CPU utilizado é um Intel dual Core 2.0 GHz. Este tempo de processamento é relativamente baixo quando comparado com o peso computacional do sistema de visão artificial, que consiste num simples algoritmo de segmentação de cor numa imagem com uma resolução de 656x494 que demora em média 20ms a ser executado.

4.3.3.2 Robô Vigilante

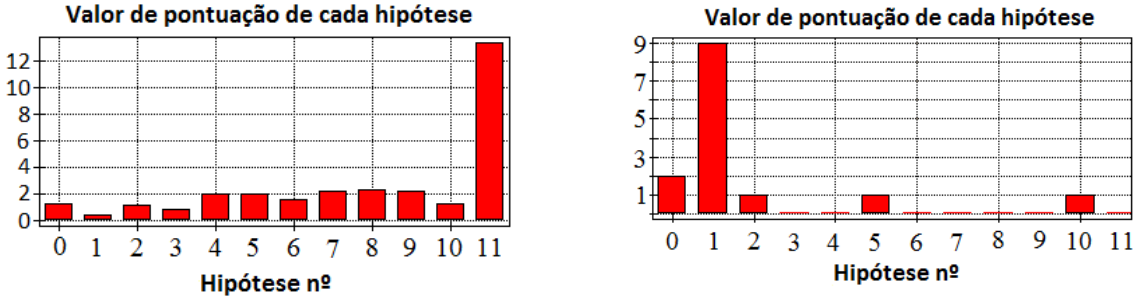
Repetiram-se os testes da secção anterior na plataforma RobotVigil apresentada na secção 4.1.5. Aqui, o robô também foi tele-operado, contudo, este robô já foi testado em diversos outros ambientes nos quais navegou com sucesso de forma autónoma.



(a) Estado inicial.

(b) Estado final antes do processo de localização global terminar.

Figura 4.41: Distribuição das hipóteses durante a localização global no caso do Robô Vigilante.



(a) Valor da função custo.

(b) Pontuação.

Figura 4.42: Custo e pontuação das hipóteses no momento em que processo de localização global termina. Ambiente de teste: Robô Vigilante.

De forma análoga à secção anterior na Figura 4.41a temos o conjunto inicial de hipóteses e as Figuras 4.41b, 4.42a e 4.42b apresentam o último estado interno do algoritmo antes deste terminar. Mais concretamente na Figura 4.41b têm-se o último estado das hipóteses e nas Figuras 4.42a e 4.42b apresentam-se, respectivamente, o custo e o *score* correspondentes a essa hipótese. Aqui o processo de localização global começa com 38 hipóteses e termina com 11 e aplicando a Condição 5 é escolhida a hipótese correcta (a número 1 da Figura 4.41b). Na Figura 4.42b observa-se que outras hipóteses, além da hipótese escolhida, cumpriram com a Condição 4, isto é, tiveram um custo de correspondência duas vezes inferior às outras hipóteses válidas. Devido ao ruido presente nas medidas adquiridas pelo robô, nem sempre o mínimo global da função custo de correspondência (Equação 3.17) corresponde à pose absoluta correcta do robô. Daí a importância do esquema de pontuação (*score*) proposto nesta abordagem.

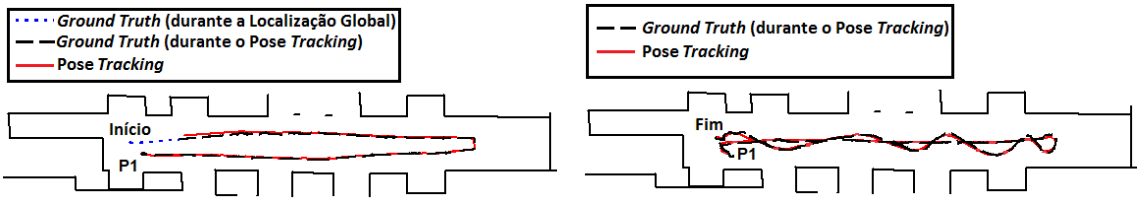


Figura 4.43: Trajectória efectuada nos testes à Localização Global no Robô Vigilante. Esta está dividida nas duas imagens acima de forma a facilitar a visualização. O Trajecto começa na posição "Start", passa por "P1"e termina em "End". Nas imagens está assinalada a posição obtida pelo módulo de *Pose Tracking* (linha tracejada vermelha) e *Ground Truth* (linha tracejada preta).

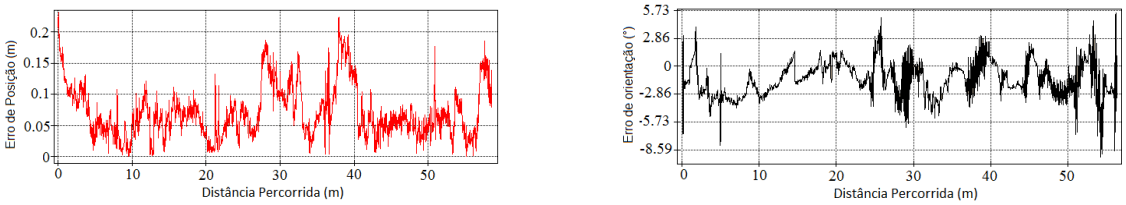


Figura 4.44: Erro de posição (esquerda) e orientação (direita) do sistema de *Pose Tracking* em função da distância percorrida durante os testes efectuados no Robô Vigilante.

Aqui também se recorreu ao sistema de triangulação laser apresentado na Secção 4.1.2.2 para analisar a precisão do sistema de localização do Robô Vigilante. Nas imagens da Figura 4.43, temos a trajectória efetuada durante esta experiência. O robô começa na posição "Início", utilizando o sistema de localização global durante o percurso sinalizado pela linha azul. Quando termina, o módulo de *Pose Tracking* é inicializado com a posição absoluta estimada e o robô continua a deslocar-se até P1 e depois para a posição "Fim". A linha preta corresponde à posição estimada pelo módulo de *Pose Tracking* e a linha vermelha corresponde à posição estimada pelo *Ground Truth*.

Os gráficos da Figura 4.44 dizem respeito ao erro de posição e orientação do sistema de *Pose Tracking* em função da distância percorrida. Obteve-se um erro médio de posição de 0.075m com um desvio padrão de 0.084m e um erro médio de orientação de 1.03° com um desvio padrão de 1.15°. Estes valores de precisão são suficientes para que o robô consiga efectuar operações de inspeção deslocando-se autonomamente em ambientes *indoor* mesmo em cenários dinâmicos.

Em termos de peso computacional, o robotvigil está equipado com uma computador de baixo puder computacional (Mini ITX, EPIA M10000G, 1.0 GHz) e processa a cada 100ms 765 pontos provenientes de um *laser-scanner* de baixo custo (Hokuyo URG-04LX-UG01) em 12ms.

Termina-se esta secção com links para vídeos correspondente aos testes realizados no Futebol Robótico e Robô Vigilante do Sistema de Localização Global e *Pose Tracking*:

- https://youtu.be/4FQCk4OGPps
- https://youtu.be/JKDoqTAEaxM?t=454

Capítulo 5

Conclusões e propostas de trabalhos futuros

5.1 Localização baseada em reflectores

Neste trabalho descreve-se o desenvolvimento de um sistema de localização para um AGV industrial. Este sistema reaproveita o laser utilizado pelo sistema de segurança para localizar o robô através da detecção de reflectores instalados junto ao chão. A abordagem proposta difere das implementações tradicionais no sentido em que estas necessitam de um equipamento dedicado instalado no topo do robô (laser de navegação), exclusivo do sistema de localização. Os resultados obtidos indicam que, apesar da qualidade sensorial oferecida pelo laser de segurança ser pior que o laser de navegação, é possível atingir os requisitos de precisão e robustez exigidos numa aplicação industrial. Tal é possível graças à aplicação de algoritmos que consigam extrair mais informação dos dados sensoriais e que, ao mesmo tempo, detectem outliers.

A proposta apresentada neste trabalho consiste na combinação de um Filtro de Kalman Extendido com outros dois filtros em série. O primeiro, "Detector Filter", detecta outliers tendo em conta a dimensão conhecida dos reflectores instalados, enquanto o segundo, "Association Filter", utiliza o valor médio e a covariância da pose estimada, rejeitando medidas pouco prováveis.

Tendo em conta os requisitos de segurança inerentes a ambientes industriais, a questão da detecção de falhas foi abordada. Apresenta-se, então, o módulo "Uncertanty Supervisor", cujo objectivo é a detecção de falhas no sistema de localização que podem ter origem na oclusão de todos os reflectores por obstáculos ou na derrapagem das rodas do AGV.

Resumidamente, em termos de análise dos resultados obtidos, foi feita uma avaliação de precisão ao sistema desenvolvido e observou-se ser possível obter uma precisão superior a 10mm com baixo peso computacional, essencial para aplicações industriais. O sistema foi também intensivamente testado em condições adversas, não laboratoriais e similares às de utilização real. Nestas foi demonstrada a importância dos módulos de filtragem desenvolvidos, uma vez que, quando estes são desactivados, se observa de imediato uma forte degradação da estabilidade da resposta do Filtro de Kalman.

Como trabalho futuro pretende-se analisar a utilização de outros algoritmos de fusão sensorial. A motivação surge com o resultado apresentado na Secção 4.2.1 na Figura 4.13 aos 200 segundos, durante os testes de precisão. Aqui verificava-se um aumento significativo do erro quando o sistema detectava apenas um reflector e utilizava, por algum tempo, apenas a odometria. Acredita-se que a origem deste erro possa estar relacionada com os erros de linearização do EKF. O plano futuro consiste em testar outros filtros não lineares, como por exemplo o *Unscented Kalman Filter* (Wan e Van Der Merwe, 2000) e o *Cubature Filter* (Arasaratnam e Haykin, 2009), de forma a melhorar este resultado. Importa referir que, durante os testes de precisão, o módulo "Uncertainty Supervisor" foi desactivado pois, caso contrário, o sistema entraria em estado de falha devido à acumulação de erro provocada pela falta de informação sensorial.

Outra questão que será abordada é a da localização global. Pretende-se combinar a solução de tracking desenvolvida com uma solução de localização global, como a apresentada no trabalho (Ronzoni et al., 2011). Com isto, seria possível inicializar o sistema automaticamente, além de aumentar a robustez com a redundância adicionada que pode ser explorada para a detecção de falhas no sistema de localização.

Neste trabalho, recorre-se a um módulo que analisa a quantidade de incerteza extraída da matriz de covariância da pose estimada de forma a detectar falhas. Como trabalho futuro, pretende-se aprofundar esta abordagem e analisar até que ponto a quantidade de incerteza estimada permite detectar com confiança a divergência do EKF.

Sumariando, o sistema aqui proposto permite localizar um AGV industrial através de um laser de segurança com a precisão e robustez necessária para operar em meio industrial. O processo de filtragem sugerido permite contornar muitos dos problemas inerentes à utilização de um laser na base do robô, tais como oclusões e detecção de *outliers*. O sistema foi testado intensivamente em condições adversas em meio industrial e demonstrou ter uma performance comparável aos sistemas industriais comerciais, com evidentes vantagens em termos de custos de equipamento e de custos de instalação.

5.2 Localização baseada em contornos

5.2.1 Comparação de algortimos de Matching

Neste trabalho apresentou-se uma comparação entre alguns dos mais relevantes algoritmos de *matching*: Perfect Match (PM), Iterative Closest Point (ICP) e Normal Distribution Transform (NDT). Esta comparação foi feita ao nível do peso computacional, velocidade de convergência, erro máximo de inicialização e robustez a *outliers*.

O PM é computacionalmente menos pesado que o ICP e que o NDT. Tal deve-se às estruturas de procura utilizadas pelo PM (*lookup tables*), que são mais eficientes em termos de tempo de acesso, quando comparadas com as *kd-trees* utilizadas no ICP implementado no PCL. Neste sentido, é proposta uma implementação alternativa do ICP na qual é utilizada uma *lookup table*

na fase de procura de correspondências em vez das *kd-trees*. Com isto, consegue-se uma diminuição significativa do peso computacional, mantendo a performance do ICP no que concerne à velocidade de convergência, erro máximo de inicialização e robustez a *outliers*. Com esta nova implementação, LUT-ICP, conseguem-se resultados em termos de peso computacional muitos próximos dos obtidos com o PM, tornando esta solução mais viável para aplicações com requisitos temporais mais exigentes. Contudo, o tempo necessário para gerar a *lookup table* utilizada no ICP é significativamente maior do que o tempo utilizado no PM para gerar as suas *lookup tables*. Este é essencialmente um problema de implementação e é pretendido, no futuro, optimizar o LUT-ICP de forma a diminuir o tempo necessário para gerar a sua *lookup table*. Esta nova implementação foi tornada pública e é uma importante contribuição no PCL. Apesar disto, o PM apresenta-se cerca de seis vezes mais rápido que o LUT-ICP, sendo provavelmente responsável por esta discrepância a diferença de eficiência computacional da optimização baseada no SVD utilizada pelo ICP e o RPROP utilizado no PM.

Em termos de velocidade de convergência, o ICP e o NDT requerem menos iterações para convergir quando o robô está parado ou a movimentar-se em linha recta. Os seus desempenhos são significativamente degradados durante movimentações que tenham componente rotacional, chegando mesmo ultrapassar o número de iterações do PM. Contudo, mesmo nas situações que favorecem o ICP e o NDT em termos de número de iterações para convergir, o PM continua a ser computacionalmente menos pesado.

A análise referente ao erro máximo de inicialização foi dos resultados mais interessantes no conjunto de testes efectuados: o PM apresenta maior tolerância a erros de inicialização na orientação do robô. Como trabalho futuro é pretendido refinar este resultado de forma a incluir o erro de posição. Contudo, testes preliminares demonstram que ambos os algoritmos exibem resposta similar em termos de erro de translação.

Em relação à precisão e robustez a *outliers*, os três algoritmos em análise demonstraram resultados similares. No entanto, a forma como o PM, o ICP e o NDT lidam com *outliers* é fundamentalmente diferente. O ICP utiliza uma função custo quadrática e aplica um filtro no qual é definida uma distância máxima entre os dados do sensor e os seus respectivos pares no mapa de referência. Aqui, pontos com um erro acima de determinado *theshold* são simplesmente descartados. Este tipo de filtragem é perfeitamente aplicável ao PM. Por outro lado, o PM utiliza uma função de custo não linear que acomoda a noção de *outlier*, limitando o peso (na optimização) de uma medida com um erro elevado. Devido à natureza não linear deste tipo de filtro no PM, tal não é directamente aplicável ao ICP.

O NDT limita a influência de *outliers*, calculando o *score* como uma mistura de distribuições normais e uma distribuição uniforme, na qual se especifica a percentagem de *outliers* esperada. Uma das desvantagens do NDT é a discretização do mapa do ambiente em células com uma resolução menor que a do mapa original. Esta abordagem introduz um erro numérico que tem um impacto direto na qualidade do resultado. Este erro numérico torna-se mais acentuado no caso em que o mapa é pouco linear (com muitas curvas e muito irregular), chegando-se a verificar erros de 18cm.

Existem diversas propostas de solução no estado da arte para a resolução do problema de correspondência com mapas. Este estudo não pretende ser uma avaliação exaustiva dos diversos algoritmos disponíveis, mas pretende sim validar a relevância do PM utilizado como ponto de partida nos desenvolvimentos aqui apresentados e futuros. Como trabalho futuro, pretendemos alargar este estudo a um conjunto maior de algoritmos, condições de teste e ambientes.

5.2.2 Algoritmo de *Pose Tracking*

Como foi referido, foi feito um estudo que analisou diferentes critérios de desempenho de alguns dos mais relevantes algoritmos de *Map Matching*. O *Perfect Match* (PM) apresentou diversas vantagens em termos de peso computacional, tolerância a erros de inicialização e robustez a *outliers*. Além disto, o PM tem dado provas da sua relevância no contexto das competições de Futebol Robótico Médio, sendo aplicado com sucesso aos algoritmos de localização dos robôs. Estes geralmente deslocam-se a grandes velocidades, necessitando de um sistema de localização que opere a frequências elevadas e que seja robusto às adversidades inerentes ao seu regime de operação, como por exemplo a derrapagem das rodas e consequente falha da odometria, falhas no sistema de visão devido a variações da iluminação ambiente, etc.

Tendo isto em conta, o PM foi escolhido como algoritmo base de desenvolvimento de um sistema de localização baseado em contornos naturais para ser aplicado a um robô industrial. Contudo, as aplicações industriais têm algumas especifidades em relação ao Futebol Robótico: a qualidade sensorial das medidas obtidas por um *laser* de navegação e pelos *encoders* de um robô industrial é geralmente melhor que a disponível nos robôs utilizados nas competições de futebol robótico. No entanto os requisitos de precisão e robustez a falhas são tipicamente muito mais exigentes. Normalmente os robôs móveis industriais realizam operações de carga e descarga que exigem uma repetibilidade no sistema de navegação inferior a 1cm em termos de posição e 1° em termos de orientação. Além disso, em muitos casos, os robôs são de grandes dimensões e um desvio de poucos centímetros da sua trajectoria pode constituir um problema de segurança.

Tendo em conta estas características particulares, foi efectuado um conjunto de alterações ao PM de forma a viabilizar a sua aplicação a um sistema de localização industrial baseado em contornos, nomeadamente:

- Adição de um critério de paragem que garante a convergência do algoritmo;
- Introdução de um esquema adicional de filtragem de *outliers*;
- Alterações e melhoramentos na função custo;
- Melhoria da fusão sensorial através da aplicação de um filtro de kalman.

O impacto das alterações propostas foi analisado a diferentes níveis, tanto em simulação como em diferentes cenários reais e obtiveram-se resultados promissores que suportam a viabilidade da aplicação do trabalho desenvolvido num sistema industrial. Foi feita uma análise em que foram medidos os mesmos critérios de desempenho utilizados na comparação dos algoritmos de *Map*

Matching de forma a evidenciar o impacto das alterações propostas ao PM. Destaca-se a análise do desempenho em termos de tolerância a erros de inicialização e robustez a *outliers*: com a abordagem proposta conseguiu-se um bom *trade-off* entre estes dois critérios tendo-se obtido um aumento significativo da precisão aumentando a robustez a *outliers* sacrificando a tolerância a erros de inicialização. Contudo este último mantém níveis de desempenho aceitáveis para a grande maioria das aplicações. A tolerância a erros de inicialização é especialmente importante para contrapor o efeito de erros grosseiros na odometria, sendo que em aplicações industriais a derrapagem das rodas não é um fenómeno frequente e os sistemas de navegação operam a uma frequência elevada. A robustez a outliers e a precisão são sem dúvida factores mais críticos em aplicações industriais. Também foi implementada uma demonstração similar a uma aplicação industrial em que um protótipo de um AGV realiza operações de carga/descarga de material numa posição fixa no espaço. Para isto, o sistema de navegação têm que conseguir parar o robô numa pose pré-configurada com uma repetibilidade muito elevada (menos de 1cm em termos de posição e menos que 1° em orientação). Esta demonstração permitiu testar a solução desenvolvida em diferentes cenários com diferentes tipos de adversidades. Estes cenários evidenciaram a importância dos desenvolvimentos efectuados no âmbito deste trabalho no sentido em que o robô provou ter a robustez necessária para realizar com sucesso a demonstração acima descrita durante várias horas consecutivas. Para além disso, desabilitando-se as alterações propostas, verificava-se uma degradação da repetibilidade do sistema navegação que levava o robô a falhar as operações de carga/descarga.

Realça-se que durante os testes e demonstrações realizadas no âmbito deste trabalho foi recolhida uma grande quantidade de dados e identificadas possíveis melhorias ao sistema.

Uma das possíveis melhorias é a optimização da eficiência computacional da arquitectara proposta. Verificou-se uma relação entre o número de iterações necessárias para o PM convergir e a amplitude do passo inicial utilizado na optimização com o RPROP. Na arquitectura proposta, são executados sequencialmente dois módulos de *Map Matching*. Usando um passo inicial mais baixo na segunda optimização, espera-se que o número de iterações necessárias diminua, pois o erro de inicialização da segunda optimização é mais baixo. Também se sugere utilizar o erro de correspondência na escolha da amplitude do passo inicial de forma a optimizar o número de iterações necessárias à convergência do PM.

Os resultados experimentais obtidos tanto em simulação como com um robô real, demonstraram as vantagens da alteração da função custo. Esta alteração consistiu em tornar o peso de uma medida proporcional à sua distância ao sensor. Dando mais peso às medidas mais distantes, procura-se contrariar o efeito radial que faz com que objectos mais próximos tenham um peso mais significativo na optimização. Futuramente, pretende-se refinar este modelo e torná-lo mais preciso incorporando uma relação mais precisa entre o peso de um objecto no *Matching* e a sua distância. No contexto da localização baseada em reflectores, foi deduzido um modelo que dá o número de medidas associadas a um objecto em função da sua distância. Sugere-se a inclusão deste modelo neste contexto a análise do seu impacto em relação aos resultados já obtidos.

A função custo utilizada originalmente no PM tem em conta a existência de *outliers* limitando

a influência de uma medida com um erro de correspondência elevado. Foi verificado que esta abordagem só por si não era suficiente para atingir a precisão pretendida, e que é necessário identificar, no conjunto de medidas adquiridas pelos sensores, quais as que correspondem a *outliers* de forma a sejam ignoradas pelo algoritmo de *Map Matching*. Optou-se por uma filtragem baseada na aplicação de dois *thresholds* de erro de correspondência: inicialmente é feita uma pré-filtragem dos dados do laser, na qual são removidas as medidas com um erro de correspondência elevado; de seguida é executado o processo de optimização; depois é aplicada uma nova filtragem com tolerância mais baixa; finalmente, é executado mais uma vez o processo de optimização. O impacto desta abordagem foi analisado experimentalmente e verificou-se um aumento da precisão do sistema. Contudo, pretende-se recorrer a mais informação de forma a aumentar o desempenho da filtragem de *outliers*: ao contrário do que acontece na presente proposta, na qual cada medida é avaliada individualmente e são utilizados *thresholds* fixos, no futuro, pretende-se incluir a noção de *cluster* segundo a qual, se uma medida é assumida como *outlier*, é muito provável que as medidas adjacentes também o sejam. Também se pretende tornar os *thresholds* adaptativos de forma a terem em conta a estimativa do erro de localização do sistema.

A estimativa do erro da pose obtida pelo processo de *Map Matching* é modelado por uma matriz de covariância. Essa pose e a sua respectiva matriz de covariância são fundidas com dados da odometria de forma a aumentar a robustez do sistema. O cálculo dessa matriz de covariância é baseado na segunda derivada da função custo do PM e neste modelo são efectuadas diversas simplificações. Nos casos de estudo analisados neste trabalho, este modelo revelou-se suficiente sendo que a utilização de um laser de navegação numa posição elevada permite a obtenção de dados sensoriais com boa qualidade. Pretende-se continuar estes desenvolvimentos na perspectiva de aplicar este trabalho a uma solução que utilize os dados adquiridos por um laser de segurança. Neste caso, a qualidade sensorial disponível é pior, e o ambiente é mais dinâmico, o que leva a um aumento da relevância da qualidade dos modelos de erro e algoritmos de fusão sensorial. Tendo isto em conta, como trabalho futuro pretende-se melhorar a precisão dos modelos de erro associados ao processo de *Matching* e recorrer a outros tipos de algoritmos de fusão sensorial. Numa primeira fase, pretende-se analisar as melhorias conseguidas eliminando as simplificações introduzidas. Depois, será relevante a introdução de outros filtros de fusão sensorial como por exemplo *Unscented Kalman Filter* (Wan e Van Der Merwe, 2000). Este último não recorre a linearizações dos modelos de observação e pode ser uma mais valia neste contexto. Por fim, também se pretende introduzir algoritmos da família dos Filtros de Partículas que apesar de serem computacionalmente pesados conseguem acomodar qualquer modelo de erro, nomeadamente modelos de erro não lineares. Os *Rao-Blackwellized Particle Filters* recorrem a algoritmos de Matching de forma a diminuir o número de partículas necessárias, combinar a aplicação de um filtro de partículas com o PM é uma abordagem que se considera promissora e que se pretende analisar.

Por fim, como trabalho futuro pretende-se combinar a utilização simultânea de contornos com reflectores. Em ambientes muito dinâmicos e pouco estruturados pode não ser possível localizar o robô com a precisão necessária, em todo o espaço de navegação, apenas com contornos, em especial no caso de utilizar um laser de segurança. Por isso, e devido ao facto de as exigências

de precisão poderem ser diferentes ao longo do percurso de um robô, existe o interesse em fundir localização baseada em contornos e reflectores num só sistema. Por exemplo, utilizar contornos em situações em que as exigências ao nível da precisão são mais baixas, como pode ser o caso da navegação entre postos de trabalho, e utilizar reflectores quando é necessário grande precisão, como em pontos de atracagem para a carga/descarga de material.

5.2.3 Algoritmo de localização global

Neste trabalho foi abordada a questão da Localização Global, isto é a determinação da pose do robô sem ter disponível uma estimativa aproximada da pose real do robô, ao contrario do que acontece nas soluções de *Pose Tracking*.

Neste contexto as soluções de *Tracking* consistem em determinar um mínimo local de uma função de custo, enquanto que a localização global implica procurar o mínimo que corresponde à verdadeira pose do robô ao longo de todo o domínio da função de custo (ou seja, ao longo de toda a área de navegação do robô). A localização global torna-se, assim, computacionalmente dispendiosa pois implica uma procura exaustiva ao longo do mapa utilizado pelo sistema de localização. Neste sentido propõe-se um esquema híbrido composto por um sistema de *Pose Tracking*, um sistema de detecção de falhas e um sistema de localização global. Em condições de funcionamento normal, é utilizado o sistema de *tracking* (computacionalmente mais leve) e paralelamente existe um módulo de supervisão que detecta a falha do *tracking* da localização. Se for detectada uma falha, recorre-se ao módulo de localização global de forma a recuperar a pose do robô e reenicializar com este o sistema de *tracking*.

O sistema de detecção de falhas procura explorar a redundância nos sensores do robô de forma a detectar a divergência do *tracking* da pose. No trabalho aqui apresentado compara-se a orientação absoluta estimada por uma bússola com orientação absoluta obtida pela correspondência de mapas. Como trabalho futuro pretende-se estender esta abordagem e explorar outro tipo de redundâncias como a odometria e sensores inerciais, de forma a tornar a detecção de falhas mais robusta e rápida.

O sistema de localização global aqui proposto consiste num esquema de multi-hipóteses em que várias instancias do sistema de *Pose Tracking* são inicializadas em vários pontos do mapa. Seguidamente, utilizando-se um esquema de eliminação/fusão, o número de hipóteses vai diminuindo até sobrar uma hipótese que ofereça confiança suficiente. Esta arquitectura de localização foi testada com sucesso no robô de serviços chamado Robô Vigilante, e é utilizada nos robôs de futebol robótico. No caso específico do futebol robótico, as condições são particularmente adversas devido às elevadas velocidades e numerosas colisões a que os robôs são sujeitos, o que leva a falhas relativamente frequentes do sistema de *tracking*. O módulo de localização global torna-se assim uma mais-valia. Em relação ao robô vigilante interessa realçar que, apesar do poder computacional do computador utilizado no robô ser relativamente baixo, consegui-se utilizar esta abordagem, o que evidencia que os requisitos computacionais do algoritmo de localização global não são muito exigentes.

Em relação ao trabalho futuro pretende-se optimizar a geração do conjunto inicial de hipóteses e analisar a sua aplicabilidade em sistemas industriais.

Referências

Martin Adams, Ba-Ngu Vo, Ronald Mahler e John Mullane. SLAM Gets a PHD: New Concepts in Map Estimation. *IEEE Robotics & Automation Magazine*, 21(2):26–37, jun 2014. ISSN 1070-9932. doi: 10.1109/MRA.2014.2304111. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6814323.

I. Arasaratnam e S. Haykin. Cubature Kalman Filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, jun 2009. ISSN 0018-9286. doi: 10.1109/TAC.2009.2019800. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4982682.

Patric Beinschob e Christoph Reinke. Strategies for 3D data acquisition and mapping in large-scale modern warehouses. In *Proceedings - 2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing, ICCP 2013*, páginas 229–234, 2013. ISBN 9781479914937. doi: 10.1109/ICCP.2013.6646113. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84891081857{&}partnerID=tZOtx3y1.

P.J. Besl e H.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. ISSN 01628828. doi: 10.1109/34.121791. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0026821209{&}partnerID=tZOtx3y1.

P. Biber e W. Strasser. The normal distributions transform: a new approach to laser scan matching. In *EEE International Conference on Intelligent Robots and System*, volume 3, páginas 2743–2748. IEEE, 2003. ISBN 0-7803-7860-1. doi: 10.1109/IROS.2003.1249285. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1249285.

Jose Blanco, Javier Jimenez e Juan Madrigal. An Alternative to the Mahalanobis Distance for Determining Optimal Correspondences in Data Association. *IEEE Transactions on Robotics*, 28(4):980–986, aug 2012. ISSN 1552-3098. doi: 10.1109/TRO.2012.2193706.

Gunilla Borgefors e Gunilla. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, jun 1986. ISSN 0734189X. doi: 10.1016/S0734-189X(86)80047-0. URL http://linkinghub.elsevier.com/retrieve/pii/S0734189X86800470.

Andrea Censi. An ICP variant using a point-to-line metric. In *2008 IEEE International Conference on Robotics and Automation*, páginas 19–25. IEEE, may 2008. ISBN 978-1-4244-1646-2. doi: 10.1109/ROBOT.2008.4543181. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4543181.

Carlos M. Costa, Héber M. Sobreira, Armando J. Sousa e Germano M. Veiga. Robust 3/6 DoF self-localization system with selective map update for mobile robot platforms. *Robotics and Autonomous Systems*, 76:113–140, 2016. ISSN 09218890. doi: 10.1016/j.robot.2015.09.030.

A. Diosi e L. Kleeman. Fast Laser Scan Matching using Polar Coordinates. *The International Journal of Robotics Research*, 26(10):1125–1153, oct 2007. ISSN 0278-3649. doi: 10.1177/0278364907082042. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-34748829174{&}partnerID=tZOtx3y1.

A. I. Eliazar e R. Parr. Learning probabilistic motion models for mobile robots. *Proceedings of International Conference on Machine Learning*, páginas 32–38, 2004.

João Sena Esteves. *Metodologia de Autolocalização Absoluta em Ambientes Quase-Estruturados*. Ph.d thesis, School of Engineering of the University of Minho, 2005.

M.A. Fischler e R.C. Bolles. Random sample consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24 (6), 1981. doi: 10.1145/358669.358692.

Xiaobin Gao, Jingchuan Wang e Weidong Chen. Land-mark placement for reliable localization of automatic guided vehicle in warehouse environment. In *2015 IEEE International Conference on Robotics and Biomimetics, IEEE-ROBIO 2015*, páginas 1900–1905, 2016. ISBN 9781467396745. doi: 10.1109/ROBIO.2015.7419050.

Giorgio Grisetti, Cyrill Stachniss e Wolfram Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, feb 2007. ISSN 1552-3098. doi: 10.1109/TRO.2006.889486. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4084563.

Guangjie Han, Huihui Xu, Trung Q. Duong, Jinfang Jiang e Takahiro Hara. Localization algorithms of Wireless Sensor Networks: a survey. *Telecommunication Systems*, 52(4):2419–2436, apr 2013. ISSN 1018-4864. doi: 10.1007/s11235-011-9564-7. URL http://link.springer.com/10.1007/s11235-011-9564-7.

Stefan Kohlbrecher, Oskar von Stryk, Johannes Meyer e Uwe Klingauf. A flexible and scalable SLAM system with full 3D motion estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, páginas 155–160. IEEE, nov 2011. ISBN 978-1-61284-769-6. doi: 10.1109/SSRR.2011.6106777. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6106777.

Martin Lauer, Sascha Lange e Martin Riedmiller. Calculating the perfect match: an efficient and accurate approach for robot self-localization. *Robocup 2005: Robot soccer world cup*, (c):142–153, 2006. URL http://www.springerlink.com/index/V59M05J034N40228.pdf.

Jae Yeong Lee e Wonpil Yu. Robust self-localization of ground vehicles using artificial landmark. In *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2014*, páginas 303–307. IEEE, nov 2014. ISBN 9781479953325. doi: 10.1109/URAI.2014.7057439. URL http://ieeexplore.ieee.org/document/7057439/.

Zhibin Liu, Zongying Shi, Mingguo Zhao e Wenli Xu. Mobile robots global localization using adaptive dynamic clustered particle filters. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, páginas 1059–1064. IEEE, oct 2007. ISBN 978-1-4244-0911-2. doi: 10.1109/IROS.2007.4399050. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-51349154746{&}partnerID=tZOtx3y1.

I. Loevsky e I. Shimshoni. Reliable and efficient landmark-based localization for mobile robots. *Robotics and Autonomous Systems*, 58(5):520–528, may 2010. ISSN 09218890. doi: 10.1016/j.robot.2010.01.006. URL http://linkinghub.elsevier.com/retrieve/pii/S0921889010000072.

Feng Lu e Evangelos Milios. Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 18 (3):249–275, 1997. ISSN 09210296. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0031100775{&}partnerID=tZOtx3y1.

Martin Magnusson. *The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection*. tese de doutoramento, Örebro universitet, 2009.

Martin Magnusson, Tom Duckett e Achim J. Lilienthal. Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10):803–827, Oct 24 2007. ISSN 1556-4959 (Print), 1556-4967 (Online).

Jorge J. More, David J. Thuente, Preprint Mcs-p, Jorge J. Moré e David J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw.*, 20(3):286–307, 1992. ISSN 0098-3500. doi: 10.1145/192115.192132. URL http://doi.acm.org/10.1145/192115.192132$\delimiter"026E30F$nhttp://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=DA15A309E3603FE39E9F72D5FD565395?doi=10.1.1.30.660{&}rep=rep1{&}type=pdf$\delimiter"026E30F$nhttp://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.30.660.

Marius Muja e David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application - VISSAPP'09)*, páginas 331–340. INSTICC Press, 2009.

E.B. Olson. Real-time correlative scan matching. In *IEEE International Conference on Robotics and Automation*, páginas 4387–4393. IEEE, may 2009. ISBN 978-1-4244-2788-8. doi: 10.1109/ROBOT.2009.5152375. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5152375.

Vincent Pierlot e Marc Van Droogenbroeck. BeAMS: A beacon-based angle measurement sensor for mobile robot positioning. *IEEE Transactions on Robotics*, 30(3):533–549, 2014a. ISSN 15523098. doi: 10.1109/TRO.2013.2293834.

Vincent Pierlot e Marc Van Droogenbroeck. A New Three Object Triangulation Algorithm for Mobile Robot Positioning. *IEEE Transactions on Robotics*, 30(3):566–577, jun 2014b. ISSN 1552-3098. doi: 10.1109/TRO.2013.2294061. URL http://ieeexplore.ieee.org/document/6693716/.

Miguel Pinto, A. Paulo Moreira, Aníbal Matos e Héber Sobreira. NOVEL 3D MATCHING SELF-LOCALISATION ALGORITHM. *International Journal of Advances in Engineering & Technology*, 5(1):1–12, 2012. ISSN 2231-1963.

Miguel Pinto, a. Paulo Moreira, Aníbal Matos, Héber Sobreira e Filipe Santos. Fast 3D Map Matching Localisation Algorithm. *Journal of Automation and Control Engineering*, 1(2):110–114, 2013. ISSN 23013702. doi: 10.12720/joace.1.2.110-114. URL http://www.joace.org/index.php?m=content{&}c=index{&}a=show{&}catid=33{&}id=56.

M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler e A. Y. Ng. ROS: an open-source Robot Operating System. In *Open-Source Software workshop of the International Conference on Robotics and Automation*, Kobe, Japan, 2009. URL http://pub1.willowgarage.com/{~}konolige/cs225B/docs/quigley-icra2009-ros.pdf.

C. Reinke e P. Beinschob. Strategies for contour-based self-localization in large-scale modern warehouses. *IEEE 9th International Conference on Intelligent Computer Communication and Processing, ICCP 2013*, páginas 223–227, 2013. doi: 10.1109/ICCP.2013.6646112. URL http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6646112.

M. Riedmiller e H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks - Conference Proceedings*, volume 1993-Janua, páginas 586–591. IEEE, 1993. ISBN 0780309995. doi: 10.1109/ICNN.1993.298623. URL http://ieeexplore.ieee.org/document/298623/.

Davide Ronzoni, Roberto Olmi, Cristian Secchi e Cesare Fantuzzi. AGV global localization using indistinguishable artificial landmarks. In *IEEE International Conference on Robotics and Automation*, páginas 287–292, Shanghai, may 2011. IEEE. ISBN 978-1-61284-386-5. doi: 10.1109/ICRA.2011.5979759. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5979759.

Radu Bogdan Rusu e Steve Cousins. 3D is here: Point Cloud Library (PCL). In *2011 IEEE International Conference on Robotics and Automation*, páginas 1–4. IEEE, may 2011. ISBN 978-1-61284-386-5. doi: 10.1109/ICRA.2011.5980567. URL http://ieeexplore.ieee.org/document/5980567/.

L. Sabattini, V. Digani, C. Secchi, G. Cotena, D. Ronzoni, M. Foppoli, F. Oleari, Christoph Reinke e Patric Beinschob. Technological roadmap to boost the introduction of AGVs in industrial applications. In *Proceedings - 2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing, ICCP 2013*, páginas 223–227, 2013. ISBN 9781479914937. doi: 10.1109/ICCP.2013.6646112. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84891125890{&}partnerID=tZOtx3y1.

L Schulze, S Behling e S Buhrs. Automated Guided Vehicle Systems: a driver for increased business performance. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, páginas 19–21, Hong Kong, 2008. URL http://www.doaj.org/doaj?func=fulltext{&}aId=551360.

Lothar Schulze e Alexander Wullner. The Approach of Automated Guided Vehicle Systems. In *2006 IEEE International Conference on Service Operations and Logistics, and Informatics*, páginas 522–527. IEEE, jun 2006. ISBN 1-4244-0318-9. doi: 10.1109/SOLI.2006.328941. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4125635.

S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte e D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot Minerva. *International Journal of Robotics Research*, 19(11):972–999, 2000. ISSN 02783649. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0034315905{&}partnerID=tZOtx3y1.

S. Thrun, W. Burgard e D. Fox. *Probabilistic robotics*. Probabilistic Robotics, 2005. ISBN 0-262-20162-3. URL http://portal.acm.org/citation.cfm?id=504754.

Nicola Tomatis. BlueBotics: Navigation for the Clever Robot [Entrepreneur]. *IEEE Robotics & Automation Magazine*, 18(2):14–16, jun 2011. ISSN 1070-9932. doi: 10.1109/MRA.2011.941629. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5876200.

Ali R. Vahdat. Mobile robot global localization using differential evolution and particle swarm optimization. In *2007 IEEE Congress on Evolutionary Computation*, páginas 1527–1534. IEEE, sep 2007. ISBN 978-1-4244-1339-3. doi: 10.1109/CEC.2007.4424654. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-71249142954{&}partnerID=tZOtx3y1.

E.A. Wan e R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, páginas 153–158. IEEE, 2000. ISBN 0-7803-5800-7. doi: 10.1109/ASSPCC.2000.882463. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=882463.

Gang Wang e Kehu Yang. A New Approach to Sensor Node Localization Using RSS Measurements in Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, 10 (5):1389–1395, may 2011. ISSN 1536-1276. doi: 10.1109/TWC.2011.031611.101585. URL http://ieeexplore.ieee.org/document/5739089/.

Hui Zhang, Lingtao Zhang e Jing Dai. Landmark-based localization for indoor mobile robots with stereo vision. In *Proceedings - 2012 International Conference on Intelligent Systems Design and Engineering Applications, ISDEA 2012*, páginas 700–702. IEEE, jan 2012. ISBN 9780769546087. doi: 10.1109/ISdea.2012.640. URL http://ieeexplore.ieee.org/document/6173302/.

Lei Zhang, René Zapata e Pascal Lépinay. Self-adaptive Monte Carlo localization for mobile robots using range finders. *Robotica*, 30(02):229–244, jun 2011. ISSN 0263-5747. doi: 10.1017/S0263574711000567. URL http://journals.cambridge.org/abstract{_}S0263574711000567.

Junyi Zhou e Jing Shi. RFID localization algorithms and applications—a review. *Journal of Intelligent Manufacturing*, 20(6):695–707, dec 2009. ISSN 0956-5515. doi: 10.1007/s10845-008-0158-5. URL http://link.springer.com/10.1007/s10845-008-0158-5.