FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Navegação Topológica Aplicada a Agentes Robóticos Móveis

Pedro Guilherme Santos Carvalho Borges Ramos

Coimbra, Setembro de 2018

Navegação Topológica Aplicada a Agentes Robóticos Móveis

Orientador:
Professor Doutor Paulo Jorge Carvalho Menezes

Co-Orientador:
Doutor João Manuel Leitão Quintas

Júri:
Prof. Dr. Rui Paulo Pinto da Rocha
Prof. Dr. Jorge Nuno de Almeida e Sousa Almada Lobo
Prof. Dr. Paulo Jorge Carvalho Menezes

Dissertação de Mestrado Integrado em Engenharia Electrotécnica e de Computadores, ramo de especialização em Automação.

Coimbra, Setembro de 2018

Agradecimentos

Dedico este agradecimento não só aos que me acompanharam no presente curso, mas a todos os que me acompanharam na minha vida académica e pessoal, e ajudaram na realização deste objetivo.

Em primeiro lugar gostaria de agradecer à minha família, particularmente à pessoa dos meus pais Guilherme e Maria José e à minha irmã Diana pelo esforço, apoio e incentivo ao longo deste percurso. Obrigado por me darem sempre a mão nos momentos em que nem eu acreditava em mim.

Agradeço ao meu orientador, Professor Doutor Paulo Menezes e ao meu co-orientador Doutor João Quintas pelos conselhos, opiniões, dedicação, rigor científico e paciência os quais permitiram o desenvolvimento desta dissertação. Agradeço também aos meus colegas de laboratório, em particular ao Gonçalo Martins pelo apoio e paciência com que sempre me presenteou. Aos meus restantes amigos por todos os momentos proporcionados ao longo da minha vida académica, e pela sua amizade e apoio ao longo da vida.

Resumo

Actualmente, na sociedade em que vivemos, existe uma crescente necessidade de utilizar robôs autónomos para executar diferentes tarefas sem perturbar o ambiente onde elas são desempenhadas. Existe uma grande variedade de robôs, como robôs guia ou robôs de serviço e segurança, desenvolvidos para ajudar as pessoas a superar as dificuldades sentidas na realização de diferentes tipos de tarefas. Estes robôs estão rapidamente a ganhar um papel importante no nosso dia-a-dia, evoluindo de meros trabalhadores para companheiros. Este projecto visa criar um robô recepcionista-guia, a ser utilizado no Instituto de Sistemas e Robótica da Universidade de Coimbra. De modo a atingir este objectivo, desenvolvemos um sistema que suporta a navegação de um robô em um ambiente conhecido, baseado nos princípios da navegação topológica. Foram desenvolvidas capacidades sensoriais, de modo a assim poder evitar obstáculos que possam aparecer ao longo da trajectória. Para abordar o facto de que a estimativa da posição do robô acumula erros ao longo do tempo, um conjunto de marcadores visuais foi posicionado ao longo das trajectórias para auxiliar na localização do mesmo. Estes marcadores visuais, fornecem meios de posicionamento em relação a um sistema de coordenadas globais, podendo estes ser reconhecidos numa ampla gama de configurações de visualização. Esta é uma abordagem simples, barata, robusta e fácil de implementar para a obtenção de uma localização global. De modo a interagir com o utilizador, foi criada uma interface gráfica, onde o utilizador pode escolher entre visitas pré-planeadas ou ir diretamente a um determinado local. Foram realizadas experiências num cenário coberto, demonstrando a viabilidade e desempenho do sistema.

Palavras chave: Odometria, Navegação Topológica, Marcadores Visuais, Desvio de Obstáculos, Robô de Serviços

Abstract

In our modern society, there is a growing need to use autonomous robots to execute different types of tasks without disturbing the environment where they are introduced. There is a wide variety of robots such as guide robots or service and security robots, that were developed to help people overcome their difficulties while performing a multitude of different tasks. Robots are quickly earning an important role in our daily-life, evolving from laborers to companions. This project aims to create a recepcionist-guide robot to be used at the Institute of Systems and Robotics, University of Coimbra. To accomplish this goal, we have developed a system that supports the navigation of a robot in a known environment, based on the principles of topological navigation. In order to avoid obstacles that may appear along the trajectory, sensorial capabilities were added. To address the fact that the estimation of the position of the robot accumulates error throughout time, a set of visual markers were positioned along the trajectories to aid in localizing the robot. These landmarks provide means for positioning in reference to a global coordinare system over a wide range of viewing configurations. This is a simple, inexpensive, robust, and easy to implement aproach to global localization. To interact with the user a graphical user interface was created were the user can choose from between pre-planned visits or to go to a certain location. We have performed experiments in a indoor scenario, demonstrating the feasibility and performance of the system.

Keywords: Dead Reckoning, Topological Navigation, Visual Markers, Obstacle Avoidance, Service Robot

"But remember this... airplanes are not tools for war. They are not for making money. Airplanes are beautiful dreams. Engineers turn dreams into reality."

— Hayao Miyazaki, The Wind Rises

Conteúdo

Abreviaturas e Símbolos

DWA	Dynamic Window Approach
GPS	Global Positioning System
IMU	Inertial Measurement Unit
RFID	Radio Frequency Identification
HRI	Humam-Robot Interaction
AMCL	Adaptative Monte Carlo Localization
ROS	Robot Operating System
VFF	Virtual Force Field
VFH	Virtual Force Histogram
APF	Artifical Potential Fields
RRT	Rapidly-Exploring Random Tree
VSTP	Very Simple Trajectory Planner
QR	Quick Response
IBA	Intelligent Bug Algorithm
CAD	Computer Aided Design

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Esta dissertação descreve e discute o tema de "Navegação Topológica para Agentes Robóticos Móveis". O trabalho foi desenvolvido no Laboratório de Robótica Móvel situado no Instituto de Sistemas e Robótica, Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Existe, cada vez mais, um crescente número de robôs que trabalham em proximidade com o ser humano, interagindo e realizando tarefas em parceria com ele. O futuro passa por uma relação simbiótica entre os dois mundos. Foi neste âmbito que este projecto foi criado, tendo como objectivo o desenvolvimento de um sistema de suporte à navegação de um robô num espaço conhecido, baseando-se para tal nos princípios de navegação topológica, criando assim o suporte básico para um robô guia. Esta abordagem difere da navegação geométrica usual, no facto de que os movimentos são planeados com base na selecção dos locais a visitar e não nas primitivas de movimento geométrico que conectam os pontos inicial e final. Dito isto, a escolha do uso de mapas topológicos deve-se ao facto de a visita guiada ser feita entre locais pré-definidos (nós), e por isso adequados para a representação através de grafos. A pesquisa de um caminho que ligue a posição actual à posição destino, não é uma pesquisa geométrica, mas sim uma pesquisa num grafo de conectividade. Neste grafo, os locais de maior importância encontram-se ligados entre si, formando uma rede de caminhos possíveis. Pesquisando estes caminhos, encontra-se uma ligação entre a posição actual e a posição destino, e consequentemente uma trajectória a seguir. Durante a navegação por esta trajectória, caso o robô não tenha certeza acerca da sua localização exata, ao verificar uma característica visual sabe em que nó do grafo se encontra.

Tomemos por exemplo o cenário seguinte: se considerarmos um ambiente constituído por salas que se ligam entre si e onde cada sala está pintada com uma cor diferente. Então pode-se saber em que sala estamos sem conhecer exatamente a posição geométrica atual.

Para a realização desta dissertação foi usado o *Segway RMP 200*, figura 1.1, composto por uma plataforma circular, a qual pode efectuar o movimento de andar em frente ou para trás e rodar em torno de um eixo vertical que se situa no centro instantâneo de curvatura, o qual é definido pelo diferencial de velocidades das rodas da plataforma. A plataforma é constituída por duas rodas motrizes, colocadas num eixo que passa pelo seu centro, as quais dotam a mesma de características de locomoção diferencial. Este primeiro capítulo

apresenta o estado da arte sobre os robôs de serviços desenvolvidos ao longo dos tempos e os respectivos métodos de navegação utilizados, os objectivos e a estrutura desta dissertação.

Figura 1.1: *Segway RMP 200*

1.1 Estado da Arte

Na sociedade moderna existe cada vez mais contacto diário entre humanos e robôs. O campo de interação homem-máquina, aborda tópicos desde o design, à compreensão de sistemas robóticos, que envolvam algum tipo de comunicação entre seres humanos e robôs [24]. E portanto, o objectivo da investigação foi dar mais ênfase ao real significado destas palavras, usando sistemas robóticos para facilitar e auxiliar os humanos nas suas tarefas quotidianas.

Como tal, em 1997 foi introduzido na Alemanha, no museu "Deutsches Museum Boon"o robô guia de seu nome RHINO [13]. O objectivo primário deste robô era oferecer uma navegação segura e fiável em ambientes públicos e bastante populados. O utilizador comunicava com o robô através de botões, sendo que o robô respondia através de mensagens de voz pré-gravadas. O RHINO oferecia também uma interface web, a qual permitia uma experiência interactiva pelas diferentes exposições. Era possível teleoperar o robô pelas diferentes salas ou apenas assistir o robô a fazer uma visita guiada à distância. Para se localizar, utilizava uma versão do método proposto por Markov [51], mantendo uma confiança probabilística acerca da posição em que o robô se encontra. A navegação, híbrida, é realizada através de um mapa métrico com a ajuda das leituras sensoriais. A duração desta experiência foi de seis dias, e embora o RHINO estivesse equipado com câmaras, sensores de ultrassom, sensores de infravermelhos, *laser-range finders* e sensores de colisão, a navegação continuava a ser difícil

dadas as características do ambiente oferecido pelo museu. Isto era derivado, por exemplo, à presença de esculturas de dimensão inferior à altura dos sensores colocados mais próximos do chão. O ambiente tornava-se dinâmico devido aos visitantes terem bancos móveis para descansarem e estarem constantemente a deixá-los em sítios diferentes. Durante uma visita, e estando o visitante a ser comandado pela sua própria curiosidade, este não pode, nem quer, esperar pelo robô. Caso isto acontecesse o robô passava de um guia para um estorvo que dificultava a experiência do visitante, tornado-a menos interessante. Para tal a velocidade de locomoção chegava a ser por vezes de 0.8 m/sec de modo a conseguir acompanhar o visitante. No entanto as maiores dificuldades, foram as impostas pelos humanos presentes no museu, que desafiavam constantemente o sistema, levando-o por vezes a ir para sítios não mapeados onde existiam perigos eminentes como escadas.

Em 2002 foi introduzido no exposição nacional da Suiça o RoboX [50]. Durante cinco meses funcionou sete dias por semana, onze horas por dia, guiando perto de um milhão de pessoas. Era dotado de capacidades de comunicação com o ser humano, sendo capaz de sintetizar frases em quatro linguagens diferentes, seguir a localização da face das pessoas usando uma câmara RGB, seguimento do corpo através de *laser range finders* e até a capacidade de exibir expressões faciais através de uma matriz de leds colocada na sua cabeça. O planeamento da trajectória era baseado em mapas topológicos, sendo que utilizava as características presentes no ambiente em conjunto com um filtro de kalman extendido [4] para estimação da localização. De modo a evitar obstáculos combinava o algoritmo *dynamic window approach* [20] (DWA) e o método *elastic band* [48] para a geração de trajectórias suaves em torno dos mesmos.

O Jinny [30], foi desenvolvido no Instituto da Ciência e Tecnologia da Coreia em Setembro de 2004 com o objectivo de permanecer permanentemente nas instalações do Museu nacional do país. Para uma estadia prolongada no museu, a interação homem-máquina tinha de ser simples e intuitiva de modo a que os empregados do museu pudessem facilmente fazer alterações consoante os eventos presentes no museu. Utilizava dois *laser range finders* , dois scanners de infravermelhos e um giroscópio de fibra optica para uma localização mais precisa. Caso algo falhasse tinha como redundância, à sua volta, sensores de colisão (bumpers). A localização era feita com base no método probabilístico de Monte Carlo [19] e a planeamento através de mapas topológicos. Utilizava quatro tipos de algoritmos de navegação diferentes, de entre os quais escolhia consoante o ambiente em que se encontrava. No caso do robô estar perante uma área aberta, usava uma navegação baseada no mapa que tinha ao seu dispor. Caso estivesse em regiões confinadas baseava a sua locomoção nos sensores disponíveis, utilizando para tal um algoritmo reactivo baseado em paredes. Para cumprir a função de guia, o robô tinha de cativar os seus visitantes e captar a sua atenção. Capaz de se expressar emocionalmente através de ícones representados numa matriz de leds, de dois braços com um grau de liberdade e um pescoço com dois graus de liberdade. Reconhecia e sintetizava expressões e era capaz de manter uma conversação sobre uma gama de tópicos limitada. No final acabou por implementar com sucesso quatros tarefas, de entre elas a realização de

entregas, patrulha, robô guia e limpeza do chão.

Em [37] foi usado o PR2 numa maratona de navegação robusta em ambiente de escritório, percorrendo com sucesso $42km$ sem acidentes. A localização era feita de duas formas. Se existisse um mapa à priori a localização era probabilística e efectuada através do uso do algoritmo adaptive Monte Carlo localization (AMCL) [18]. No caso de não existir um mapa, possuía um inertial measurement unit (IMU) que permitia fazer fusão sensorial com a odometria podendo assim melhorar a estimação da posição. O planeamento era efectuado através da construção de um mapa de custo, com base na informação sensorial, onde era usado o algoritmo A* para planeamento da trajectória e o algoritmo DWA para desvio de obstáculos. O PR2 é um robô omnidireccional, construido por dois braços, cada um com um *gripper* possibilitando assim a manipulação de objectos. As tarefas realizadas eram diversas, como ir buscar cafés, itens ao frigorífico ou limpar mesas.

1.2 Objectivos

Este trabalho tem como objectivo o desenvolvimento de um sistema simples e robusto de suporte à navegação de uma plataforma, num espaço conhecido à priori, usando técnicas de navegação topológica, criando assim a base para um robô guia. Para tal foram tomadas em conta as seguintes tarefas:

1. Criação de um algoritmo de navegação
2. Desenvolvimento de um módulo de localização a partir dos já existentes marcadores visuais da biblioteca OpenAR [40]
3. Integração deste módulo com a informação odométrica
4. Integração e configuração de um planeador topológico
5. Equipar a plataforma com sensores de distância e desenvolvimento da capacidade de desvio de obstáculos
6. Criação de uma interface gráfica
7. Testar a fiabilidade do sistema

1.3 Organização do documento

Esta dissertação é constituída por 7 capítulos, estando o restante documento organizado da seguinte forma: O capitulo 2 aborda o planeamento da trajectória que leva o robô da posição onde ele se encontra para a posição destino, sendo que no capitulo 3 é explicado o algoritmo do calculo dos comandos de velocidade que permitam a realização dessa trajectória, assim como o método utilizado para o desvio de obstáculos que possam aparecer durante o

percurso. No capitulo 4 é explorada a utilização de marcadores visuais para melhoramento da estimação da posição ao longo do percurso, diminuindo assim o erro proveniente da estimação da posição através da odometria. Já no capitulo 5 é apresentada a arquitectura do sistema, implementação prática e a interface gráfica desenvolvida para uma comunicação com o utilizador do sistema. Por fim, no capitulo 6 são apresentados os resultados e no capitulo 7 as conclusões e possível trabalho futuro para esta dissertação.

Capítulo 2

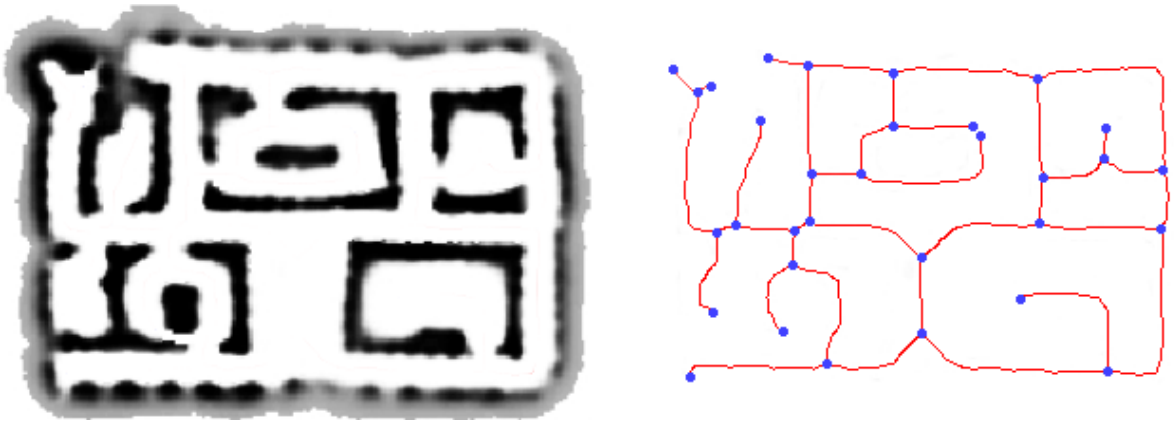
Planeamento

2.1 Planeador

Em robótica móvel, navegação é a capacidade de o robô se movimentar autonomamente e em segurança em direcção a um objectivo, tendo em conta o seu conhecimento sobre o ambiente e a sua percepção sensorial sobre o que o rodeia. Para que isto aconteça com sucesso, é necessário planear a trajectória que o robô deve seguir. A complexidade deste planeamento cresce com o aumento do número de graus de liberdade do robô. Este planeamento pode ser classificado como global ou local dependendo se a informação sobre o ambiente é conhecida pelo planeador à priori ou não. Um planeamento diz-se global quando existe um conhecimento sobre o mapa à priori. Ou seja, tendo um ponto objectivo, o planeamento de uma trajectória consiste em encontrar o caminho geométrico que consiga ligar a posição actual à posição destino. Um planeamento local não tem qualquer tipo de informação relativa ao ambiente, e portanto tende a ter um comportamento reactivo, reagindo a obstáculos na proximidade do robô. Este planeamento é muitas vezes realizado em conjunto com o planeamento global, de modo a evitar obstáculos que possam aparecer ao longo da trajectória calculada pelo planeador global.

Isto leva-nos a outra diferenciação entre ambientes estático e dinâmicos. Um ambiente é estático quando não contém qualquer tipo de objectos móveis sem ser a própria plataforma, por outro lado um ambiente dinâmico refere-se ao facto da existência de obstáculos em movimento, sendo estes até capazes de mudar o seu tamanho e forma.

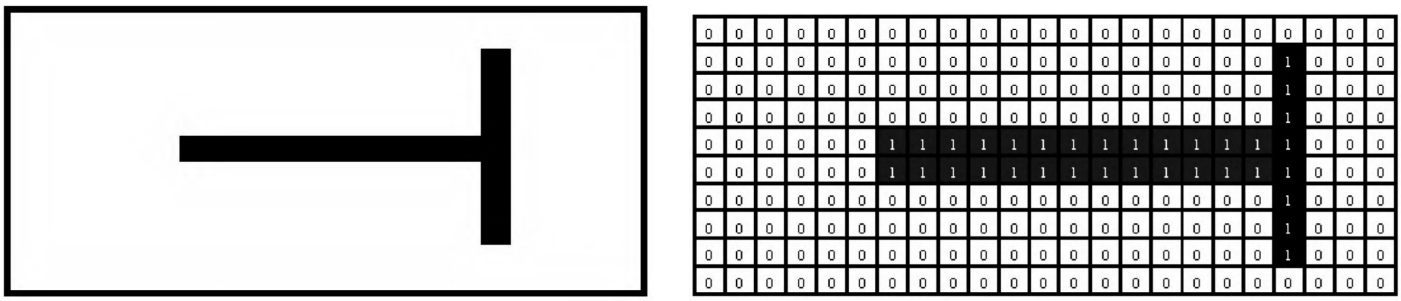
Os mapas modelam o ambiente real usando para tal as informações cartesianas do mesmo. É comum fazer-se a distinção entre dois tipos de representação do ambiente [15, 55]: mapas geométricos, topológicos.



(a) Mapa métrico (b) Mapa topológico

Figura 2.1: Comparação entre mapas métricos e topológicos. Imagem reproduzida de [38]

Os mapas métricos contam com a vantagem de serem uma representação simples e facilmente perceptível para os seres humanos, contudo devido à grande quantidade de detalhes que estes mapas possuem, o tempo necessário para a sua construção é grande. O mapeamento através de grelhas de ocupação, como o proposto por Moravec e Elfes [41, 17], é o tipo mais comum, baseando-se normalmente na discretização do espaço de navegação em regiões espaciais, dando origem a matrizes de células, onde cada célula é a representação de uma parte da área de navegação do robô. Os elementos dessa matriz contêm uma estimativa do estado dessa área no que diz respeito à presença ou não de obstáculos. Assim, para cada medida obtida pelo sensor são atualizadas as células que correspondem à posição estimada do obstáculo, atualizando o nível de confiança das células como estando ocupadas ou vazias, figura 2.2. Mas nem todos os mapas geométricos são discretos e baseados em células. Existem mapas que se baseiam em primitivas geométricas como linhas, arcos, etc, que têm a vantagem de serem escaláveis mas tipicamente mais difíceis de construção (automática).



(a) Divisão com obstáculo no centro. (b) Mapeamento da divisão a) em grelhas de ocupação

Figura 2.2: Exemplo do mapeamento em grelhas de ocupação. Imagem reproduzida de [39]

Por sua vez, os mapas topológicos representam o ambiente sobre a forma de nós e arcos, permitindo a representação do mapa sobre a forma de um grafo. Entende-se por nós, pontos de relativa importância no mundo e por arcos, a existência de um caminho direto entre esses nós. A figura 2.1 ilustra a comparação entre mapas métricos e topológicos.

Uma abordagem híbrida é usada em [25], onde o planeamento é feito através da utilização

de mapas topológicos e a localização através de princípios métricos.

O interesse da existência de um planeamento eficaz, rápido, robusto e optimizado é comum a várias áreas para além da robótica móvel. Devido a isto, ao longo dos anos foram aparecendo testes padrão para comparar a eficiência das diferentes abordagens ao problema. Entre elas, o mais conhecido é o Alpha puzzle [33], criado por Boris Yamrom e posteriormente adaptado por Nancy Amato para efeitos de referência de comparação. É constituído por dois tubos, ambos com a forma de um alpha (α), sendo que um deles é o obstáculo e o outro pode ser compreendido como um robô a realizar a sua trajectória. O objectivo é separar os dois tubos, sem colisões e através de uma passagem estreita disponível para o efeito. Uma ilustração do problema pode ser encontrado na figura 2.3. O planeamento pode ser classificado como optimo consoante a métrica de avaliação usada, podendo esta ser o comprimento da trajectória, tempo mínimo de viagem ou algo como minimizar as viragens ou as travagens.

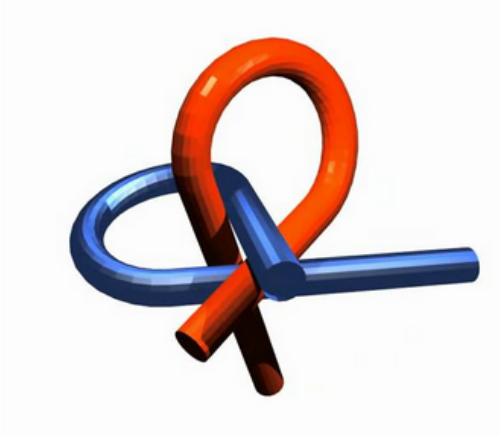
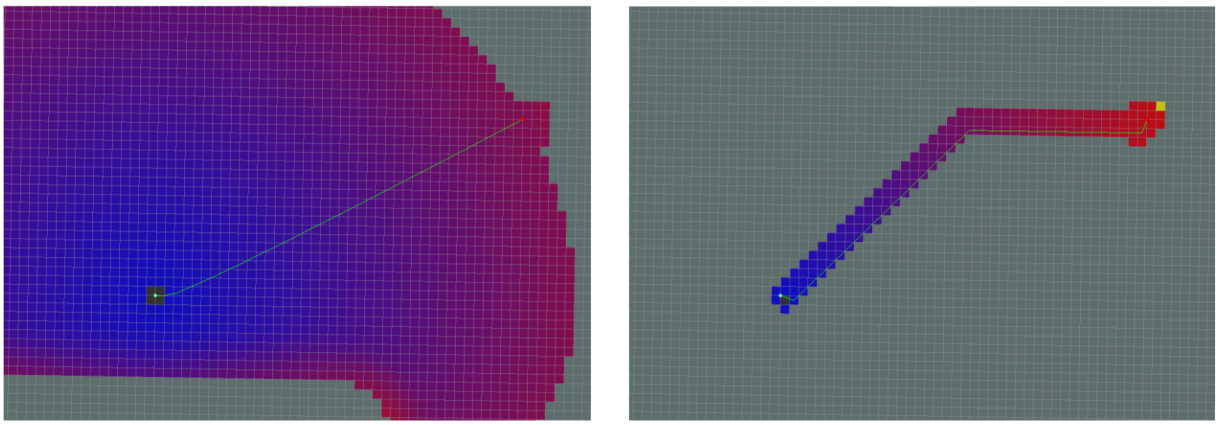


Figura 2.3: Ilustração do Alpha puzzle benchmark.

Neste trabalho é utilizado um mapa topológico para efeitos de navegação, e o planeamento baseia-se na problemática de encontrar o caminho mais curto entre dois nós através de um grafo de visibilidade. Este conceito é transversal a várias áreas. Exemplo disso é a procura da rota optima para o envio de um pacote de dados no caso das redes de computadores, ou a procura de uma trajectória que permita ir de um ponto para outro no caso da robótica móvel. Entende-se por caminho mais curto, o caminho com menos custo cumulativo, podendo este custo ser uma distância física no caso da robótica, ou um atraso temporal no caso das redes de computadores, ou uma outra métrica que tenha importância para uma aplicação especifica. O desempenho de um algoritmo de procura pode ser medido através de três métricas:

- Optimo: A solução encontrada é a melhor solução possível em termos de custo do caminho?
- Tempo necessário: Quanto tempo demora a procura até encontrar essa solução?
- Complexidade: Quanta memória e número de necessária para a realização da procura? Ou seja, de que maneira o número de operações implicadas na resolução do problema, cresce com o número de elementos do mesmo.

Um dos algoritmos de procura mais antigos, e mais simples, para este fim tem o nome de *Dijkstra's algorithm* [16]. O funcionamento consiste no seguinte: começando no nó inicial, é calculado o custo para chegar a todos os nós vizinhos desse nó. O caminho com menor custo é escolhido e o procedimento é repetido até chegar ao nó objectivo. Uma vez atingido o nó objectivo, a trajectória a seguir fica então calculada e pronta para o robô a efectuar. A função de custo é dada por $f(x) = g(x)$ onde $g(x)$ representa o custo do caminho do nó actual para o nó x. Este é um algoritmo optimo no sentido em que encontra sempre o caminho mais curto para o objectivo, no entanto, este algoritmo é computacionalmente intenso, pois para cada nó examina todos os caminhos ligados a esse mesmo nó. Assim, apareceu então uma variação do *Dijkstra's algorithm* com o nome de A^* [43]. O principio de funcionamento é similar, no entanto a função de custo é dada por $f(x) = h(x) + g(x)$ onde $h(x)$ representa uma função heurística que faz uma estimativa do custo necessário para ir do nó x para o nó objectivo. O nó que apresentar o menor valor de $f(x)$ é escolhido como o nó seguinte, combinando assim o menor caminho do nó actual para o nó seguinte, mas também do nó actual para o nó destino. A função heurística $h(x)$ não deve nunca sobrestimar este custo. Significa isto que o custo real para ir do nó actual para o nó destino, deve sempre ser maior ou igual que o custo calculado por $h(x)$. Chama-se a isto uma heurística admissível. Dito isto, enquanto o *Dijkstra's algorithm* explora todos os caminhos possíveis, o A^* tenta encontrar um caminho mais rapidamente através do uso da função heurística que dá primazia aos nós que supostamente permitam uma convergência para o objectivo mais rápida. No entanto, apenas é um algoritmo de procura optimo caso a função heurística faça uma boa estimação do custo até ao objectivo. Quanto melhor a estimação, melhores serão os resultados e mais rapidamente chegará ao nó objectivo.



(a) Dijkstra (b) A*

Figura 2.4: Comparação entre a pesquisa realizada pelo algoritmo Dijkstra e o algoritmo $A*$.

Este foi um grande avanço em relação à abordagem original, mas mesmo assim existe sempre espaço para melhoramentos. Foi o que fez Anthony Stentz em [53], apresentado o D^* como uma variação do A^* que se focou no replaneamento aquando do aparecimento de obstáculos. Ao contrário do A^*, o D^* tem a capacidade de alterar o custo do arco que liga

dois nós de maneira a conseguir replanear a trajectória em torno de um obstáculo, tentando no entanto manter a maioria da trajectória planeada anteriormente.

Existem ainda outras variações do algoritmo como o B^* e o IDA^*. Uma descrição dos mesmos pode ser encontrada em [44].

2.2 Planeador usado

Neste trabalho em especifico, foi usado um planeador baseado no algoritmo A^* de seu nome Very Simple Trajectory Planner (VSTP) [3], desenvolvido no LAAS-CNRS em Toulouse, França. O planeador funciona num ambiente de duas dimensões, e considera o robô como sendo um disco de diâmetro configurável, sendo depois calculada a trajectória mais curta entre dois pontos, tendo em conta o mapa fornecido à priori. Este mapa é construído através de segmentos de recta orientados, sendo que um dos lados deste define espaço livre e o outro espaço ocupado. O planeador recebe a posição actual e a posição destino, retornando um conjunto de pontos pelos quais o robô deve passar para chegar ao destino.

Capítulo 3

Navegação Local

Uma vez gerada a trajectória a seguir pela plataforma, é necessário começando do ponto inicial, navegar ao longo das linhas que unem nós consecutivos, até ao ponto destino. A forma escolhida para realizar a navegação no caso corrente, consiste em definir forças atractivas virtuais que iriam puxar o robô em direção a posições sucessivas da trajectória gerada até atingir o ponto destino. Essas forças virtuais, como será explicado mais adiante, servem para como formalismo de suporte à geração da sequência de comandos de velocidade necessários para guiar o robô. No entanto, em situações reais há sempre a possibilidade de obstáculos, não considerados durante a fase de planeamento, poderem aparecer pelo caminho os quais terão de ser contornados se possível. Todos os sistemas robóticos autónomos utilizam algum tipo de sistema com este objectivo pelo que este não será excepção. O comportamento em presença dos obstáculos pode ser simplesmente o de parar e solicitar o replaneamento da trajetória, à utilização de algoritmos e sistemas sensoriais que permitam comportamentos reactivos contornando desde que possível os obstáculos encontrados. A abordagem utilizada neste trabalho corresponde ao uso de um planeamento local discutido anteriormente no capitulo 2.

Nas secções seguintes serão analisados os métodos usados tradicionalmente para a navegação na presença de obstáculos, seguindo-se da apresentação do método escolhido e sua implementação. Os resultados obtidos serão posteriormente apresentados no capítulo 6.

3.1 Estado da Arte da Navegação na Presença de Obstáculos

3.1.1 *Potencial Fields*

Robustez e segurança são duas máximas de grande importância que qualquer sistema robótico deve ter em conta. A forma como se faz a navegação, o planeamento e se evita obstáculos, têm grande influência na forma como as pessoas reagem a um sistema autónomo. Andrews and Hogan em [2] e mais tarde Khatib em [29], sugeriram o conceito da existência de forças virtuais com ponto de aplicação no centro do robô, onde os obstáculos exercem uma força

repulsiva enquanto que o ponto objectivo exerce uma força atractiva. A resultante das forças, resultado do somatório da força atractiva com as repulsivas, é a nova direcção que o robô deve seguir.

Krogh em [32], melhorou este conceito tendo em conta a velocidade do robô na proximidade dos obstaculos. No entanto Brooks em [11, 12], foi o pioneiro na implementação de uma variante destes métodos numa plataforma móvel. Para tal usou um *array* de sonares onde cada leitura dos sensores contribuia com uma força repulsiva. Se a magnitude resultante da soma das forças ultrapassa-se um certo limiar, o robô parava, redirecionava-se e continuava o movimento. Apoiados nesta abordagem, Borenstein and Koren em 1989 apresentaram o método *Virtual Force Field* (VFF). Este método teve uma grande procura devido à necessidade de um baixo poder de computação, à matemática utilizada e às trajectórias suaves que era capaz de reproduzir.

Para tal, utiliza uma grelha cartesiana bidimensional (histograma bidimensional) para representar os obstáculos detectados sendo que em cada célula da grelha fica registada a confiança do algoritmo de como realmente existe um obstáculo nessa localização. Esta certeza aumenta com cada leitura do sensor que indique a presença de um obstáculo nessa célula. Esta abordagem deriva de uma técnica mais antiga chamada Grelhas de Certeza (certainty grid) [42]. Começa-se então por definir uma zona chamada região activa em torno do robô, caso se detecte a presença de um obstáculo nessa zona, essa célula contribui com uma força repulsiva sobre o robô. O valor dessa força é proporcional ao valor do histograma bidimensional e inversamente proporcional à distância entre a célula e o robô, sendo que a força atractiva para o ponto destino é constante.

No entanto o VFF não era perfeito e tinha alguns problemas como ficar encurralado em mínimos locais, não encontrar uma passagem possível entre obstáculos com um espaçamento reduzido entre eles e apresentava oscilações em passagens estreitas como corredores ou portas [56][31]. De maneira a combater estas limitações foi desenvolvido pela mesma equipa de investigadores o Vector Field Histogram (VFH). Este algoritmo elimina as limitações do VFF, tentando no entanto manter todas as suas vantagens [9].

O funcionamento e tratamento do dados pode-se resumir a duas etapas:

1. Construção de uma grelha de ocupação tal como o VFF
2. Construção de um histograma polar relativo à distribuição de obstáculos em torno do robô.

O histograma polar é obtido dividindo uma janela centrada no robô em sectores, e contabilizando o número de células ocupadas em cada sector. Obtendo o histograma polar, podemos então saber qual direcção o robô pode tomar sem haver perigo de colisão. Para tal, e devido à natureza discreta do histograma polar, este é filtrado através de um filtro passa-baixo de fase nula. O resultado desta filtragem contém picos que correspondem às regiões com maior densidade de obstáculos, sendo que os vales correspondem a regiões onde esses obstáculos são diminutos. Uma vez conhecida a posição destino e posição actual do

robô, é escolhido o vale do histograma, através de um limiar, que mais se aproxima dessa orientação objectivo.

3.1.2 *Bug algorithms*

Esta abordagem combina planeamento local com um critério de convergência global. Inicialmente, o robô move-se diretamente em direcção ao ponto destino. Quando um obstáculo é detectado, este começa a seguir o seu contorno até que uma condição se saída (que salvaguarda o critério da convergência global) for cumprida. À posição que verifica esta condição de saída, dá-se o nome de ponto de saída. Uma vantagens destes em comparação aos *Potencial Fields* é o facto de estes não caírem em mínimos locais. Isto é derivado do facto destes apenas terem em conta as percepções actuais e não as passadas.

Na abordagem Bug-1, quando detecta um obstáculo começa a seguir o contorno do mesmo, circundado-o, até chegar ao mesmo ponto em que o obstáculo foi detectado e o começou a contornar. Enquanto isto, encontra-se simultaneamente a cada instante a calcular a distância da posição actual ao ponto destino, guardando aquela que minimiza essa distância. Após um ciclo completo (contorno total do objecto encontrado) o robô passa para a segunda fase, de ir para o destino, onde o ponto de saída é o ponto previamente guardado como o ponto de menor distância ao ponto destino, figura 3.1 (b). Esta abordagem é bastante ineficiente e leva a que por vezes o robô vagueie para longe do objectivo ao tentar contornar o obstáculo [49, 1]. Outra desvantagem deste algoritmo é que aquando do inicio do seguimento do obstáculo, caso encontre um segundo pelo caminho pode colidir com ele, caso o espaçamento entre eles seja menor que o largura do robô.

O funcionamento do Bug-2, que é uma iteração do Bug-1, começa por inicialmente ser calculado o segmento de recta que liga o ponto inicial ao ponto final. O robô começa por seguir a direcção desse segmento até se deparar com um obstáculo. Aí, começa por seguir o seu contorno, calculando a cada instante se já intersectou novamente o segmento de recta calculado inicialmente. Quando esta condição se verificar, abandona o modo de contorno de obstáculo e passa para o modo de ir de encontro ao ponto destino, figura 3.1 (c). A diferença da tomada de decisão em abandonar o modo de contorno de obstáculo, torna este algoritmo mais eficiente e mais rápido a convergir para o destino. Dito isto, e mesmo sendo uma melhoria considerável em relação ao seu antecessor Bug-1, este algoritmo não faz um uso optimo dos dados recebidos pelos sensores para o calculo de uma trajectória mais curta [36].

Este foi o mote para a criação de mais uma vertente desta família que foi o Dist-Bug [28, 61]. Esta abordagem encara o problema de uma maneira um pouco diferente. Quando deparado com um obstáculo e dado inicio ao modo de desvio do mesmo, é calculada a distância ao ponto destino a partir da posição actual e da posição seguinte. O ponto de saída é quando a distância da posição seguinte ao ponto destino é maior que a distância da posição actual ao mesmo ponto ($d_{seguinte} > d_{actual}$). Mesmo com estas melhorias, o dist-bug não é um método orientado ao destino e portanto, enquanto ocupado a evitar obstáculos,

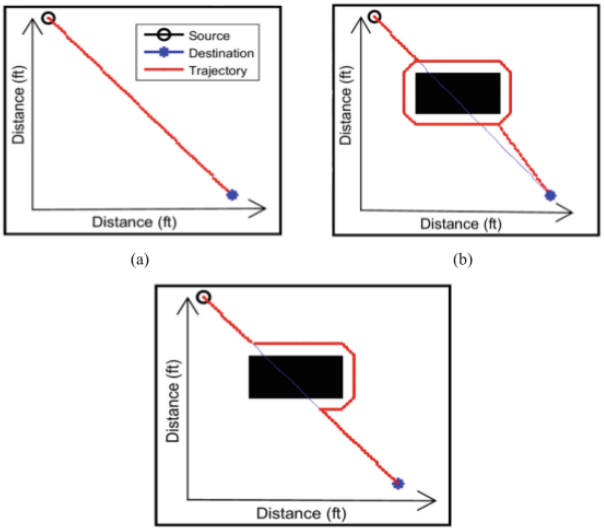


Figura 3.1: Comparação entre o Bug-1(b) e o bug-2 (c). Adaptado de [62]

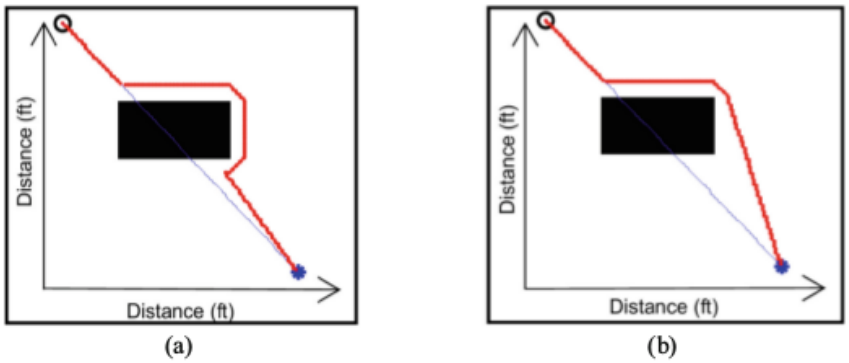


Figura 3.2: Comparação entre o Dist-bug e o IBA. Adaptado de [62]

pode levar o robô para longe do mesmo.

Este foi o motivo impulsionador para Zohaib e Pasha criarem o Inteligente Bug Algorithm (IBA) [62]. A proposta destes investigadores consiste em modificar o Dist-bug tornando-o um algoritmo orientado para o destino e tendo em conta o tempo necessário para chegar ao mesmo. Para tal, inicialmente é calculado uma trajectória de referência que o robô segue até encontrar um obstáculo. Quando isto acontece, o robô começa a seguir o seu contorno tendo em consideração a posição do ponto destino, sendo o ponto de saída do modo de desvio de obstáculos, decidido com base no caminho livre em direcção ao destino. Este caminho livre é detectado pelos sensores a bordo do robô. Uma vez tomada a decisão de ir novamente em direcção ao ponto destino, é calculada uma nova trajectória de referencia e o processo é repetido até chegar ao local pretendido.

3.1.3 *Bubble Rebound Algorithm*

O trabalho que deu mote a este algoritmo foi proposto por Quinlan em [48]. Quinlan definia a existência de uma "bolha"contendo o máximo de espaço livre à volta do robô, que podia ser navegada em qualquer direcção sem perigo de colisão. O tamanho e forma da bolha era

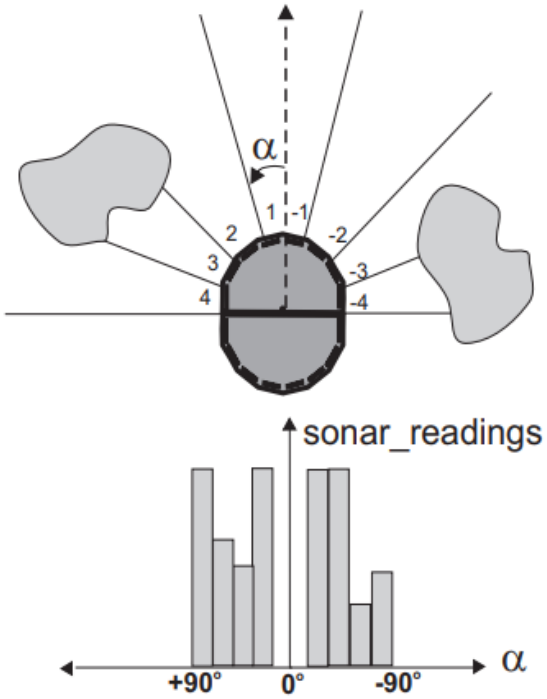


Figura 3.3: Diagrama polar realizado através das leituras dos sonares. Adaptado de [54]

determinado através de um modelo simplificado da geometria do robô, e pelas distâncias devolvidas pelos sensores. Susnea [54], pegou neste principio e criou um algoritmo simples, reactivo, capaz de evitar obstáculos em tempo real através do uso de sensores de ultrasom ou infravermelhos e com o objectivo de ser implementado num microcontrolador. Este método define uma área em torno de robô, (*bubble*), que é igualmente ajustada de acordo com a forma do mesmo. Como os sensores estão distribuídos uniformemente pelo robô, perfazendo um arco de 180^{o}, as leituras dos sonares podem ser representadas através de um diagrama polar, como pode ser verificado através da figura 3.3. Aquando da detecção de um obstáculo, o robô analisa o ambiente à sua volta, escolhendo a direcção com menor densidade de obstáculos, até o ponto objectivo ficar novamente visível ou outro obstáculo aparecer.

3.2 Locomoção

O robô utilizado foi apresentado no capítulo 1, e o controlo dos seus movimentos é realizado através da definição, a cada instante, de uma velocidade de linear, v, e uma velocidade angular, w. Para tal é definido um vector em direcção ao ponto objectivo, figura 3.4, e os comandos de velocidade são então calculados através da equação 3.1, onde α_1 e α_2 representam constantes e $\overrightarrow{A}_x$ e $\overrightarrow{A}_y$ representam a projecção desse vector atractivo no eixo de coordenadas X e Y respectivamente. Este método pode ser visto como um controlo proporcional onde o erro é a distância do robô ao ponto objectivo.

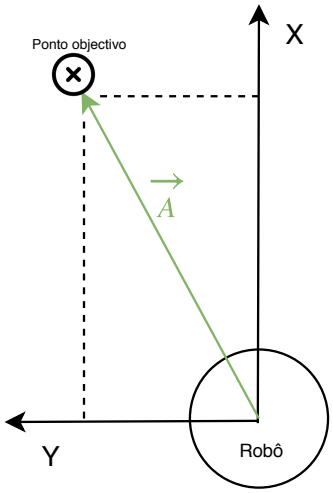


Figura 3.4: Representação da força atractiva virtual (verde) que irá puxar o robô em direcção ao ponto objectivo.

$$\begin{aligned} v &= \alpha_1 \cdot \overrightarrow{A}_x \\ w &= \alpha_2 \cdot \overrightarrow{A}_y \end{aligned} \tag{3.1}$$

Assim, caso $v \neq 0$ e $w = 0$ o robô movimenta-se em linha recta, caso $v = 0$ e $w \neq 0$ o robô descreve uma rotação sobre o seu centro, e por fim, combinações de $v \neq 0$ e $w \neq 0$ dão origem a uma trajectória com a forma de um arco.

No entanto, para esta metodologia funcionar é necessário que as coordenadas do ponto objectivo estejam no referencial do robô. O planeador devolve os pontos objectivo em coordenadas do mundo, sendo então necessário transformar o ponto objectivo localizado no referencial mundo ${}^{W}\mathrm{p}$, num ponto no referencial da plataforma móvel ${}^{R}\mathrm{p}$, figura 3.5. É de salientar que estes pontos são em 2D, mas são aqui tratados em coordenadas homogéneas.

Esta transformação é conseguida através da equação 3.2, onde $({}^{W}_{R}\mathrm{T})^{-1}$ é a inversa da matriz de transformação presente em 3.3, que por sua vez, representa a transformação entre o referencial mundo e o referencial do robô, onde θ, X e Y representam a posição actual da plataforma proveniente da localização estimada a partir da odometria.

$${}^{R}\mathrm{p} = ({}^{W}_{R}\mathrm{T})^{-1} \cdot {}^{W}\mathrm{p} \tag{3.2}$$

$${}^{W}_{R}\mathrm{T} = \begin{bmatrix} cos\,\theta & -sin\,\theta & X \\ sin\,\theta & cos\,\theta & Y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Assim obtem-se a equação 3.4, onde $X_{objectivo}$ e $Y_{objectivo}$ são pontos do referencial mundo

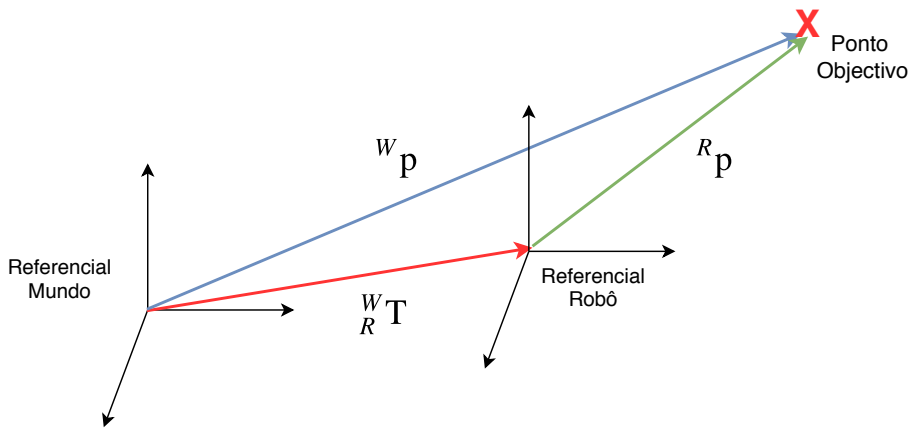


Figura 3.5: Representação da transformação entre o referencial mundo e o referencial da plataforma móvel.

e X_R e Y_R são os mesmos pontos transformados para o referencial do robô.

$$\begin{bmatrix} X_R \\ Y_R \\ 1 \end{bmatrix} = ({}^{W}_{R}\mathrm{T})^{-1} \cdot \begin{bmatrix} X_{objectivo} \\ Y_{objectivo} \\ 1 \end{bmatrix} \tag{3.4}$$

3.3 Desvio de Obstáculos

Dando seguimento à mesma filosofia usada no calculo de velocidades e à semelhança de Brooks [11, 12], nesta implementação cada sensor que retorne a informação da presença de um obstáculo, contribui com um vector repulsivo. O comprimento desse vector é inversamente proporcional à distância entre o robô e o obstáculo, sendo o vector atractivo de comprimento constante. Uma ilustração do processo pode ser encontrada na figura 3.11.

Foram utilizados sensores de ultrasom assim como sensores de infravermelhos para fazer a detecção de obstáculos durante o percurso a realizar pela plataforma. A escolha da utilização de dois tipos diferentes de sensores, foi derivada ao facto de existirem sítios onde um deles é ineficaz. Sítios estes como vidros, para os sensores de infravermelhos, e superfícies onde a onda sonora é refletida para longe do receptor, no caso dos sonares.

HC-SR04

Este sensor de ultrasom é bastante usado na comunidade da robótica devido à sua facilidade de utilização e ao custo reduzido. O principio de funcionamento baseia-se no facto da velocidade de propagação da onda sonora no ar ser conhecida. Para tal, o transdutor emissor envia uma onda sonora, enquanto que o transdutor receptor fica à espera da mesma. Quando recebida, é então produzido um pulso com a mesma duração do tempo de viagem da onda sonora.

Figura 3.6: Sonar HC-SR04

Sharp GP2Y0A02YK0F

É um sensor de distância por infravermelhos com um circuito de condicionamento de sinal integrado. Para adquirir a distância ao objecto detectado, o sensor devolve no seu terminal intermédio uma diferença de potencial, que vai corresponder à distância ao objecto. Para tal, o fabricante disponibiliza uma curva de resposta que faz correspondência entre a voltagem devolvida e a respectiva distância. Quando feitos os teste iniciais utilizando esta curva, reparou-se que os valores de distância devolvidos não eram os mais exactos.

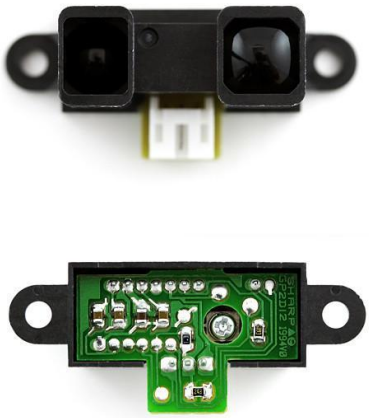


Figura 3.7: Sensor de infravermelhos Sharp GP2Y0A02YK0F

Procedeu-se então à calibração do sensor, obtendo a curva de resposta distância-tensão do mesmo. Para distâncias espaçadas de dez centímetros entre si, foram adquiridas dez amostras de tensão, feita a sua média e armazenadas numa tabela. A comparação entre a curva de calibração e a curva do fabricante podem ser vistas no gráfico da figura 3.8.

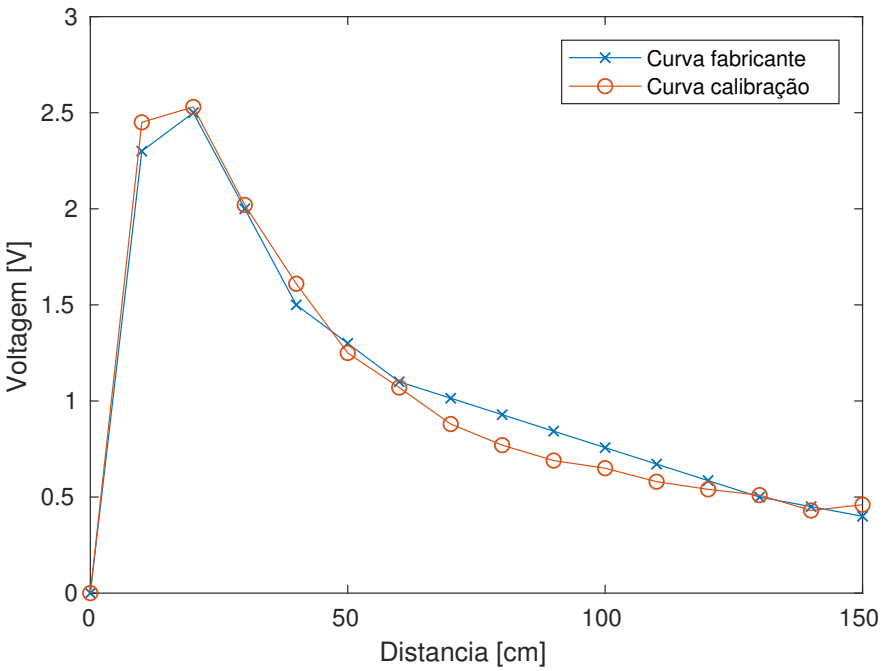


Figura 3.8: Comparação entre a curva de resposta do sensor dada pelo fabricante e da curva após a calibração.

Na figura 3.8 temos uma voltagem em função da distância. No entanto precisamos do inverso, pois o sensor devolve uma voltagem e precisamos de obter a distância correspondente. Obteve-se então o gráfico da figura 3.9, onde a função que aproxima a curva é a presente na equação 3.5.

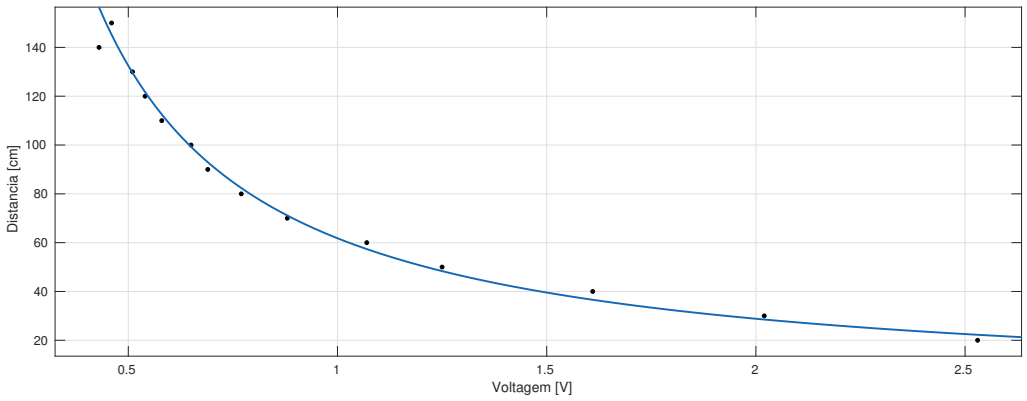


Figura 3.9: Ajuste da curva de resposta do sensor de infravermelhos.

$$d = 61.8 \cdot V_o^{-1.1} \tag{3.5}$$

Pode-se entender por d a distância do sensor ao obstáculo e por Vo a diferença de potencial devolvida pelo sensor.

A qualidade da aproximação pode ser compreendida através da tabela 3.1.

Tabela 3.1: Qualidade da aproximação da calibração do sensor.

Métrica	Valor
R-Square	0.9939
Adjustable R-square	0.9934
RMSE	3.408

O coeficiente de determinação, também chamado de *R-square*, é uma medida de ajuste de um modelo estatístico linear generalizado, como a regressão linear, em relação aos valores observados. Este pode variar entre 0 e 1, indicando, em percentagem, quanto o modelo consegue explicar os valores observados. Quanto maior, mais explicativo é o modelo e melhor ele se ajusta à amostra. Neste caso é de 0,9939, significando que 99,39 % da variável dependente consegue ser explicada pelos regressores presentes no modelo. No entanto, o problema com o R-squared é que se manterá constante ou irá aumentar com a adição de mais variáveis, mesmo que estas não ofereçam nenhuma mais valia para o ajuste. É aqui que o Adjusted R-Square entra. Este método penaliza a avaliação, caso sejam introduzidas amostras que não melhoram o modelo já existente.

Root Mean Square Error (RMSE) é o desvio padrão dos erros de predição, também conhecidos como resíduos. Estes resíduos medem quão afastado da curva de regressão linear as amostras se encontram. Esta técnica de análise utiliza a mesmo unidade de medida que a variável dependente, que neste caso é a distância. Visto que a distância varia entre 0 e 150 cm, um valor de 3.408 é bastante baixo e por consequência diz-nos que o ajuste é bastante bom.

3.3.1 Implementação

Na figura 3.10 podemos observar a disposição dos sensores no robô. Cada sensor tem um desfasamento em relação ao referencial base. Esse desfasamento, assim como o arranjo de sensores em cada posição, pode ser encontrado na tabela 3.2.

Tabela 3.2: Desfasamento e arranjo dos sensores.

Posição	Orientação	Constituintes
Grupo 1	$\pi/2$	Sonar e IR
Grupo 2	$-\pi/2$	Sonar e IR
Sensor 3	$-\pi/4$	IR
Sensor 4	$\pi/4$	IR
Sensor 5	0	Sonar

No caso dos sensores de infravermelhos, a aquisição dos dados é efectuada pelo conversor analógico-digital (ADC) de 10 bits presente no Arduino. Isto significa que irá mapear voltagens entre 0 e 5V para valores inteiros entre 0 e 1023, obtendo uma resolução de 4.9mV por

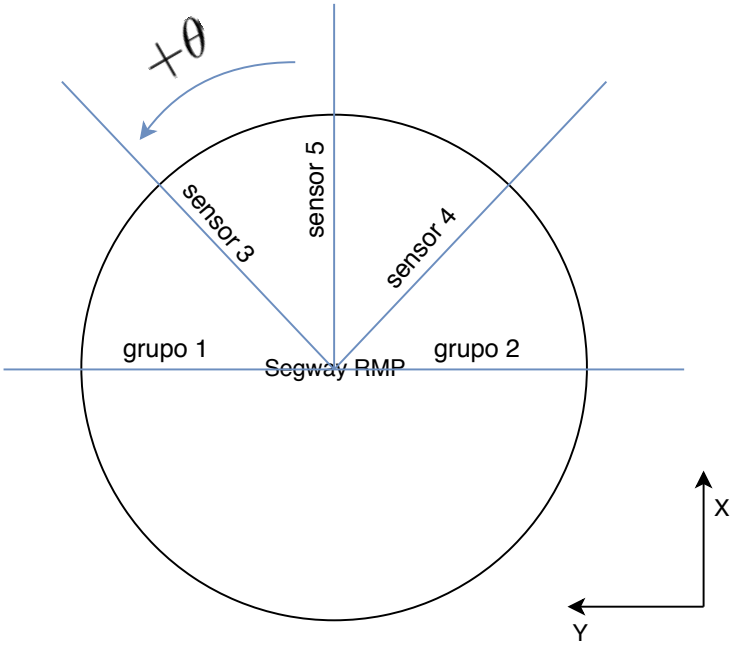


Figura 3.10: Disposição dos sensores na plataforma móvel.

unidade de escala. Uma vez que a voltagem máxima devolvida pelo sensor de infravermelhos é de 2.5V, este ADC é mais que adequado para a tarefa. Uma vez adquirido o valor da distância ao obstáculo (d), esta entra na equação 3.6 que calcula uma constante inversamente proporcional a essa distância, caso esta seja inferior a um determinado limiar.

$$\begin{cases} c = 0,\ d > limiar \\ c = \frac{k}{d},\ d < limiar \end{cases} \tag{3.6}$$

$$\overrightarrow{R} = \begin{bmatrix} c \cdot cos(\theta) \\ c \cdot sin(\theta) \end{bmatrix} \tag{3.7}$$

A variavel c, é o ganho que vai definir o comprimento do vector repulsivo, $\overrightarrow{R}$, definido pela equação 3.7, onde k é uma constante definida pelo projectista. De modo a obter um vector com sentido contrário à direcção de movimento, o θ que entra na equação 3.7, tem de estar desfasado de π do ângulo de desfasamento original da tabela 3.2.

$$\theta = \theta_{desfasamento} + \pi \tag{3.8}$$

Dependendo da localização onde a plataforma se encontra no mapa, o vector repulsivo é calculado a partir da distância devolvida pelo sensor que mais se adequa à sua posição. Isto é, o algoritmo tem zonas pré-definidas no mapa, zonas essas constituídas por vidraças, onde nesses casos apenas irá usar os valores dos sensores de ultrasom. Sendo que nas restantes zonas irá usar os valores provenientes dos sensores de infravermelhos. Dito isto, a equação 3.7 é aplicada a cada um dos cinco pontos na plataforma com sensores. No final, somam-se os cinco vectores repulsivos ($\overrightarrow{R}$), ficando com um único vector. Somando esse vector ao

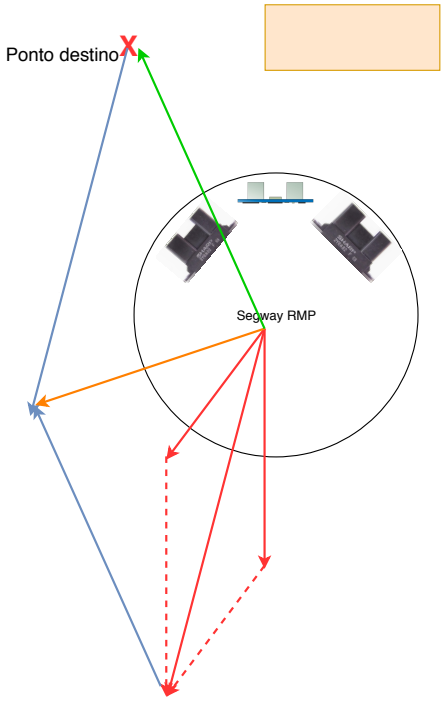


Figura 3.11: Diagrama representativo da contribuição dos vectores repulsivos e vector atractivo para o calculo de uma nova direcção de movimento. A verde é representado o vector atractivo. Sendo que nesta ilustração apenas o sensor central e o sensor direito estão a detectar o obstáculo, a vermelho são representados os dois vectores repulsivos. A vermelho tracejado pretende-se demonstrar a soma dos vectores repulsivos, sendo que a soma da resultante dos vectores repulsivos com o vector atractivo é representado pelo vector a laranja. A direcção deste é a direcção que a plataforma irá seguir naquele momento de modo a desviar do obstáculo.

vector atractivo ($\overrightarrow{A}$), obtem-se o vector da direcção a tomar ($\overrightarrow{D}$), equação 3.9.

$$\overrightarrow{D} = (\sum_{i=1}^{5} \overrightarrow{R}) + \overrightarrow{A} \tag{3.9}$$

O funcionamento do sistema está representado através do diagrama 3.12

3.4 Resumo

Neste capitulo abordou-se a técnica usada para o cálculo dos comandos de velocidade que nos permitem realizar a trajectória proveniente do planeador, assim como a utilização de vectores repulsivos para dotar o robô da capacidade de desvio de obstáculos. Dito isto, nesta fase do projecto, temos um robô capaz de navegar autonomamente e de se desviar de obstáculos. No entanto, a estimação da odometria acumula erros ao longo do tempo, os quais dão origem a uma estimação incorrecta da localização do robô. Para além disso, a localização fornecida pelo robô é obtida através da odometria num referencial não coincidente com o do mapa do ambiente, havendo assim uma diferença entre os dois que se espera que seja constante ao longo do tempo. De maneira a resolver esta questão e a corrigir a estimação da localização, foram usados marcadores visuais. O capitulo seguinte explica a metodologia e implementação deste conceito.

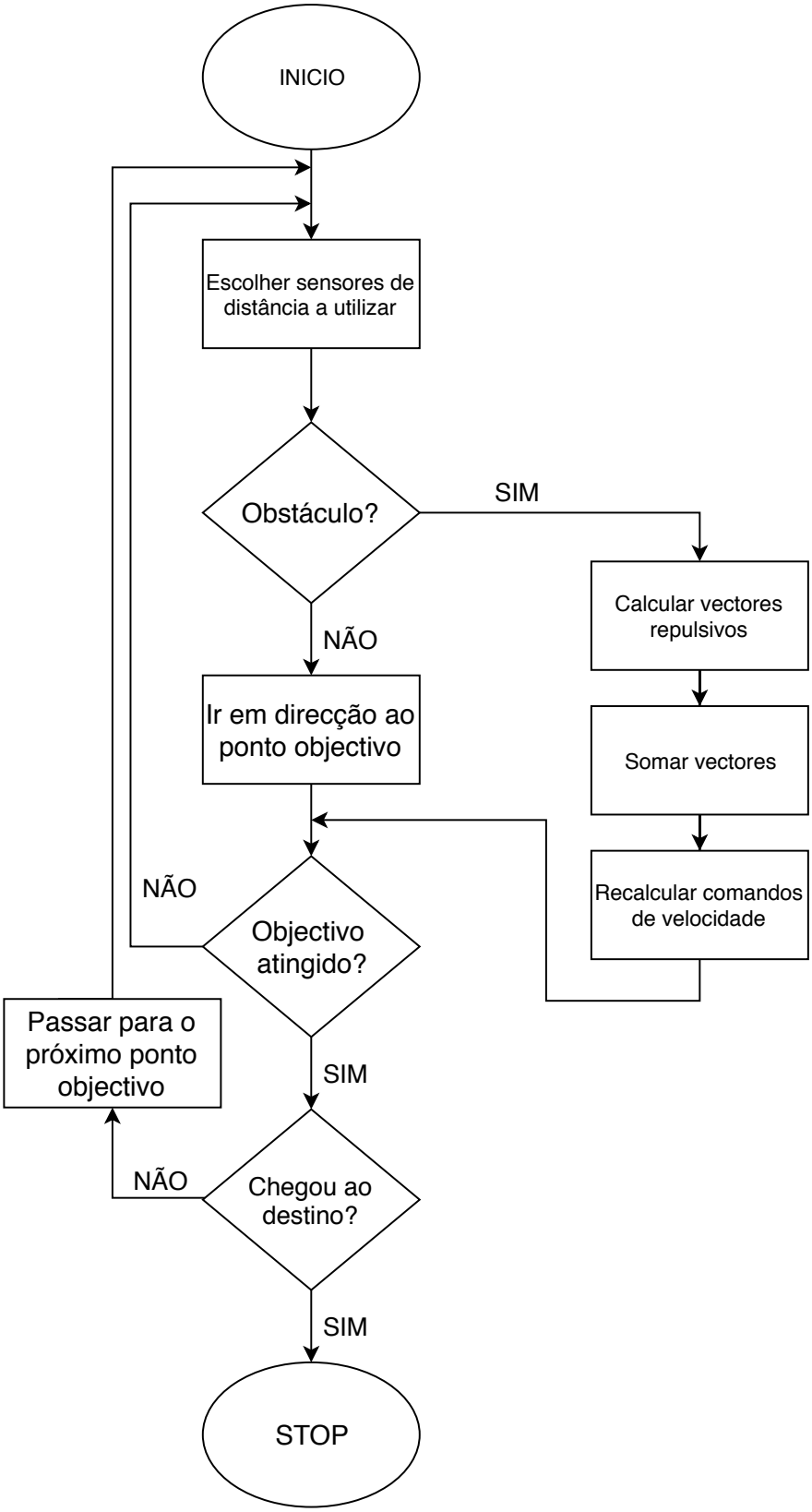


Figura 3.12: Diagrama representativo da integração do módulo de desvio de obstáculos.

Capítulo 4

Localização: Problemas e Soluções

O termo odometria é originário de duas palavras gregas, *hodos* (que significa "jornada"ou "viagem") e *metron* (que significa "medida"). Assim sendo, odometria é a medição do trajecto feito durante uma viagem. Uma plataforma móvel precisa de saber a sua localização a cada instante para a realização com sucesso das tarefas a que se propõe. Sejam elas entregar um objecto, guiar uma pessoa ou limpar o chão, este tem de ser capaz de se localizar no ambiente onde se encontra. Ou seja, para uma verdadeira autonomia ser atingida, é necessária uma correcta localização.

Dito isto, e com o objectivo de fazer uma estimação mais correcta da pose do robô, foi utilizada uma biblioteca desenvolvida ao longo dos anos no Instituto de Sistemas e Robotica, de seu nome *OpenAR* [40]. Esta biblioteca, baseada em *OpenGL*, está mais voltada para a realidade aumentada, mas tem algumas características interessantes para aplicar no mundo da robótica. Uma das quais, é a utilização de marcadores visuais para localização e posterior correcção da estimação da posição, a qual vai ser desenvolvida ao longo deste capitulo.

4.1 Estado da Arte da Localização

Uma localização exacta é essencial para uma navegação correcta. Ao longo dos anos foram desenvolvidas diferentes técnicas para estimar a localização de uma plataforma móvel. No entanto, estas pertencem a uma de duas categorias relativamente ao tipo de sensores e metodologia utilizada para tratamento dos dados [23]:

- Localização relativa: estima a posição e orientação do robô usando informação proveniente dos sensores a bordo (encoders, giroscópios, acelerometros, etc)

- Localização absoluta: obter a posição absoluta usando *beacons*, pontos de referência ou sinais de satélite

As técnicas de localização absoluta conseguem localizar correctamente o robô sem qualquer conhecimento acerca das localizações anteriores do mesmo nem da posição actual. Para tal, utilizam pontos de referência fixos no ambiente pelo qual vão navegar. Isto permite

responder ao problema do robô "raptado", onde o robô é levado para uma localização desconhecida, sem qualquer tipo de informação sobre o meio onde se encontra, com o objectivo de se localizar e navegar. No entanto, a localização absoluta é normalmente usada com o objectivo de mitigar o erro acumulado ao longo do tempo pelas técnicas de navegação relativa.

4.1.1 Localização relativa

Actualmente a odometria é um dos métodos mais amplamente utilizados para efectuar a estimação da posição. Isto deve-se ao facto de ser uma técnica barata, fornecer bons resultados a curto prazo, e permitir uma taxa de amostragem elevada. No entanto, a acumulação de erros de orientação leva a erros na estimação da posição, os quais aumentam com a distancia percorrida [8]. No caso mais especifico da robótica móvel, a odometria pode ser estimada a partir da integração ao longo do tempo do deslocamento provocado pelas velocidades lineares e angulares com que o robô se movimenta.

Esta integração do deslocamento leva à acumulação de erros que podem ser sistemáticos ou não sistemáticos [6]. Os erros sistemáticos são característicos do robô, ou dos sensores, e são uma fonte constante de erros aditivos. Exemplos disso são um diâmetro desigual das rodas e a incerteza sobre o ponto de contacto da roda com o piso. Os erros não sistemáticos são característicos da relação do robô com o ambiente, logo não podem ser contabilizados à priori pois são inesperados. Escorregamento das rodas e movimento sobre solos não uniformes são alguns exemplos destes erros [10].

Existem ainda outros sistemas de estimação de posição, como os sistema de navegação inercial, que usam giroscópios e acelerômetros para medir respectivamente a taxa de rotação e aceleração do robô. De modo a obter a posição a partir da aceleração é necessário integrar a resposta do sensor duas vezes, tornando-os sensíveis a desvios. Em alguns casos, utiliza-se fusão sensorial entre sensores odométricos e inerciais, sendo esta técnica conhecida como gyrodometry [7].

Byrne em [14], testou o uso de uma bússola eletrónica para medir a orientação do robô relativamente ao campo magnético da terra. No entanto, este sistema não é recomendado para o uso no interior de edificios, devido às distorções do campo magnético causadas pelos cabos elétricos e metal nas paredes.

4.1.2 Localização absoluta

Quando pensamos em localização absoluta o método que nos vem logo à memória é o global positioning system (GPS). Este é um método já muito estudado, bem fundamentado e aceite pela comunidade cientifica. No entanto, em ambientes cobertos, a sua utilização não é uma opção devido à dificuldade que os sinais de GPS encontram em atravessar as paredes dos edifícios [58].

Os métodos de localização através de pontos de referência estão cada vez mais presentes

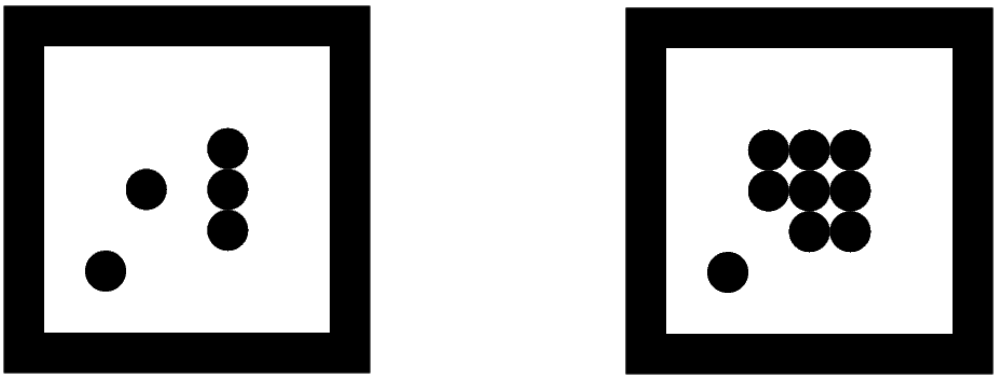
na investigação, podendo ser divididos em pontos de referência naturais ou artificiais. Ao utilizar pontos de referência naturais não existe a necessidade de modificar o ambiente, mas a escolha dos pontos de referência físicos a usar pode ser bastante complexa [35]. Por outro lado, os pontos de referência artificiais são cada vez menos intrusivos no ambiente e oferecem, no entanto, uma melhor detecção o que é necessário para uma localização fidedigna.

Yoon e Kweon em [57] e Jang em [26], usaram técnicas de processamento de cor em imagens para detectar marcadores de diferentes cores. Os códigos de barras [27] também foram utilizados como pontos de referência artificiais . Em [52], uma equipa de investigadores portugueses, desenvolveu um método onde a localização é melhorada através do uso de uma câmara para identificação de códigos de barra e consequente estimação da posição da plataforma móvel.

Mais recentemente, com o aparecimento dos códigos Quick Response (QR), e devido à facilidade de criação e detecção dos mesmos, foram desenvolvidas várias abordagens para estimar a pose de uma plataforma móvel relativamente a códigos QR colocados em sítios estratégicos. Estes podem ser visto como uma versão 2D dos códigos de barras, com a vantagem de poder conter muito mais informação. Exemplos de aplicação práticos podem ser encontrados em [22, 60], onde os códigos QR são colocados no tecto. Outas abordagens foram feitas usando técnicas como *Wi-Fi* [5], *radio frequency identification* (RFID) [45] e rádios de banda ultra larga [21]. Em [59] é medida a força do sinal wi-fi recebido e comparado com uma base de dados onde constam forças de sinal medidas em diferentes localizações do mapa. Em [34] são usadas etiquetas RFID onde cada um tem uma identificação única. Estas etiquetas, usam um campo eletromagnético para transmitir informação e podem ser activas ou passivas, dependendo se estão equipadas com uma bateria e constantemente a emitir, ou se apenas são activadas na presença do receptor. Quando o robô se aproxima destas etiquetas, o ID é detectado e obtida a posição aproximada. Neste caso, não se obtem a posição absoluta mas sim uma aproximação, pois apenas conseguimos concluir que estamos no raio de detecção possível daquele emissor em especifico.

4.2 Marcadores visuais utilizados

Os marcadores visuais utilizados são bidimensional, compostos por um quadrado/*frame* preto em fundo branco, sendo assim constituídos por oito pontos de referencia, os quatro cantos externos e outros quatro cantos internos, figura 4.7. Estes possuem um código de identificação único que os distingue entre si, código este que consiste numa distribuição binária de oito entradas no interior de cada marcador, figura 4.1. É de salientar também o ponto de orientação, figura 4.2, que é essencial para a não ocorrência de erros na localização e estimação da rotação do marcador [46].



(a) Marcador número 135 (b) Marcador número 255

Figura 4.1: Exemplo da codificação binária dos marcadores visuais. Adaptado de [46].

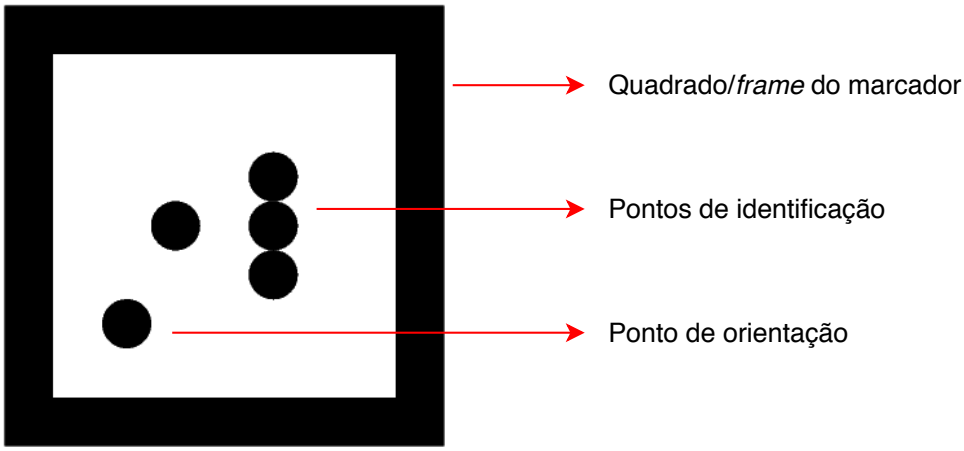


Figura 4.2: Descrição dos marcadores utilizados.

Para o correcto funcionamento do sistema a calibração da câmara é imprescindível, pois só assim é possível a extração de informação métrica a partir de imagens 2D. Para tal usou-se um método também disponível na biblioteca *OpenAR*, onde se utiliza um padrão de xadrez de modo a assim obter os parâmetros intrínsecos e os parâmetros de distorção da câmara. Estes não variam ao longo do tempo, e são específicos para cada câmara, portanto a calibração só precisa de se realizar uma vez. Por outro lado, é necessário calcular os parâmetros extrínsecos da câmara em cada imagem da sequência de detecção, pois a posição desta varia em relação aos marcadores.

A câmara utilizada funciona através do modelo *pinhole*. Este modelo é uma boa aproximação do modelo de câmara convencional e consiste numa caixa cúbica, fechada, com um pequeno orifício numa das faces, por onde os raios de projecção vão passar e formar uma imagem na face oposta à do orifício.

Este tipo de modelo baseia-se essencialmente em dois parâmetros, a distância focal (f_x, f_y) e ponto principal (c_x, c_y). Estes representam os parâmetros intrínsecos da câmara (K), equação 4.1. A distância focal é a distância que separa o plano imagem do plano focal definido pelo centro óptico. O ponto principal é o ponto de intersecção do eixo óptico com o plano imagem, sendo o eixo óptico o raio perpendicular ao plano imagem.

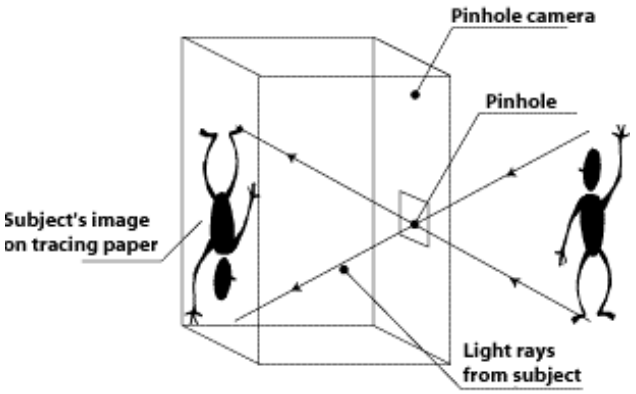


Figura 4.3: Ilustração do modelo de câmara *pinhole*.

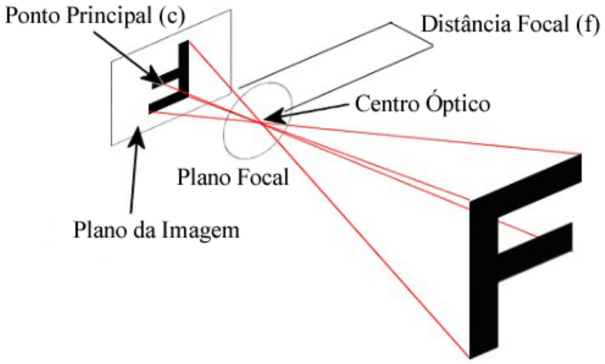


Figura 4.4: Ilustração dos diferentes parâmetros de uma câmara *pinhole*.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Ou seja, a cena que a câmara está a observar é registada através da projecção de pontos 3D do mundo real no plano imagem usando uma transformação perspectiva, equação 4.2.

$$s\, m = K\, [R|t]\, M$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.2}$$

Onde X, Y, Z são as coordenadas do ponto 3D no referencial do mundo (M), u e v são as coordenadas do ponto projectado no plano imagem (m), em pixeis, K é a matriz dos parâmetros intrínsecos e a $[R|t]$ dá-se o nome de matriz dos parâmetros extrínsecos. Esta é usada, em conjunto com a matriz dos parâmetros intrinsecos, para descrever o movimento da câmara em relação a uma cena estática ou de um objecto rígido (marcador) em relação à câmara estática. Isto é, $K[R|t]$ traduz as coordenadas de um ponto 3D no mundo para um outro sistema de coordenadas com base nos parâmetros da câmara.

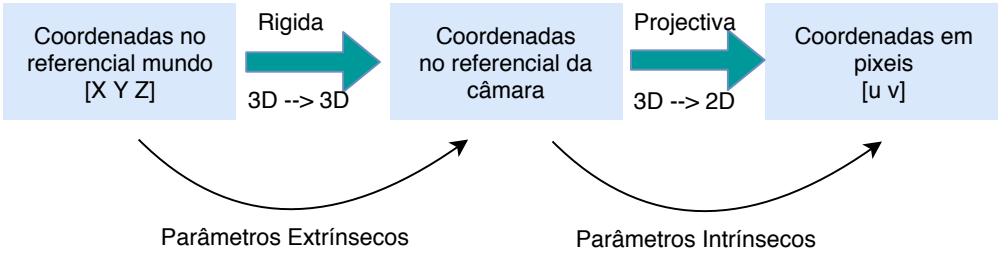


Figura 4.5: Transformações realizadas através do uso dos parâmetros extrínsecos e intrínsecos.

Assumindo que o modelo físico do marcador, que define um plano, se encontra em $Z = 0$ no referencial do mundo, os pontos do modelo físico do marcador ($\widetilde{M}$) e os pontos da imagem ($\widetilde{m}$) encontram-se relacionados por uma homografia. Esta consiste numa matriz 3×3 definida a menos de um factor de escala, equação 4.3.

$$s\widetilde{m} = H\widetilde{M} \tag{4.3}$$

Uma vez calculada a matriz de homografia, é calculada a rotação e translação que descrevem a transformação entre os pontos do marcador e os pontos imagem.

Figura 4.6: Marcador visual colocado na parede do corredor do ISR.

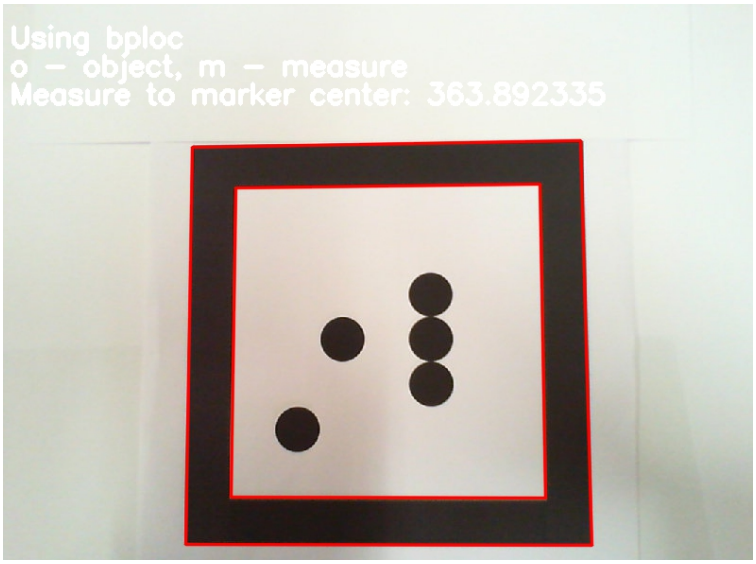


Figura 4.7: Detecção do marcador visual.

4.3 Funcionamento do sistema

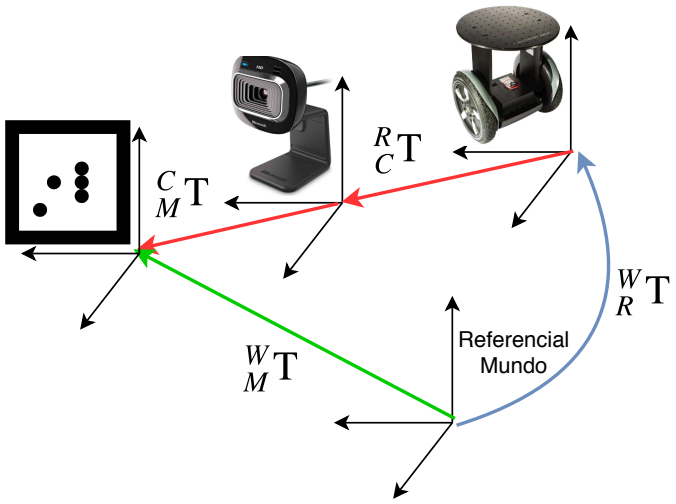


Figura 4.8: Diagrama explicativo da obtenção da posição corrigida da plataforma.

Aquando da aproximação da plataforma a um marcador e consequente detecção do mesmo, é então extraída a estimação da posição da câmara em relação ao marcador como explicado anteriormente. Esta transformação é definida por $^{C}_{M}\mathrm{T}$. De maneira a obter a pose do robô em relação ao marcador $^{R}_{M}\mathrm{T}$, equação 4.4, precisamos de mapear a câmara no referencial do robô. Para tal utiliza-se a matriz de transformação rígida definida por $^{R}_{C}\mathrm{T}$.

$$^{R}_{M}\mathrm{T} = ^{R}_{C}\mathrm{T} \cdot ^{C}_{M}\mathrm{T} \tag{4.4}$$

$$^{W}_{R}\mathrm{T} = ^{W}_{M}\mathrm{T} \cdot ^{M}_{R}\mathrm{T} \tag{4.5}$$

Para cada número identificador de um marcador existe uma matriz de transformação rígida associada. Matriz de transformação essa que define a posição e orientação do marcador em relação ao referencial mundo, definida por ${}^{W}_{M}\mathrm{T}$. Esta matriz é a verdade absoluta (*ground truth*) do sistema. Ao fecharmos o ciclo, como ilustrado na figura 4.8, e através da equação 4.5, obtemos a matriz ${}^{W}_{R}\mathrm{T}$.

4.4 Implementação

Explicado o método utilizado para a estimação da posição do robô no referencial mundo, através da detecção de um marcador visual, é agora explicado como é feita a integração desta estimação com a localização fornecida pela odometria do robô.

Quando se inicia o sistema, a posição e orientação iniciais do robô em relação ao referencial mundo são obtidas através de um marcador. Dito isto, e de modo a resolver a questão da localização fornecida pelo robô ter um referencial não coincidente com o referencial mundo, é necessário calcular a transformação entre o referencial mundo e esse referencial. Chamemos-lhe referencial da odometria. Para isso foi usada a cadeia de transformações presente na figura 4.9. Sabendo que ${}^{W}_{R}\mathrm{T}_{M}$ representa a posição e orientação do robô no mundo estimada a partir do marcador, e que ${}^{O}_{R}\mathrm{T}$ representa a posição e orientação do robô no referencial da odometria, ao fecharmos o ciclo, obtemos a transformação ${}^{W}_{O}\mathrm{T}$. Esta tranformação representa a compensação necessária para mapear o referencial da odometria no referencial do mundo, sendo obtida através da equação 4.6. Assim, a posição e orientação do robô podem ser estimadas de acordo com o referencial do mundo, através da equação 4.7.

Aquando do inicio da realização de uma trajectória e detecção de um novo marcador, é estimada uma nova matriz ${}^{W}_{R}\mathrm{T}_{M}$.

Esta nova matriz corresponde a uma actualização da estimação da posição verdadeira do robô no mundo. Assim, é calculada novamente a matriz ${}^{W}_{O}\mathrm{T}$, que tem a forma de 4.8, e a compensação entre o referencial mundo e o referencial da odometria é corrigida.

$$ {}^{W}_{O}\mathrm{T} = {}^{W}_{R}\mathrm{T}_{M} \cdot ({}^{O}_{R}\mathrm{T})^{-1} \tag{4.6} $$

$$ {}^{W}_{R}\mathrm{T} = {}^{W}_{O}\mathrm{T} \cdot {}^{O}_{R}\mathrm{T} \tag{4.7} $$

$$ {}^{W}_{O}\mathrm{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8} $$

No entanto, num sistema de visão por computador, a iluminação é um factor importante para o funcionamento correcto do mesmo. A detecção dos marcadores não é excepção, sendo influenciada pelas condições de luminosidade. Dito isto, o calculo da nova matriz ${}^{W}_{O}\mathrm{T}$ passa

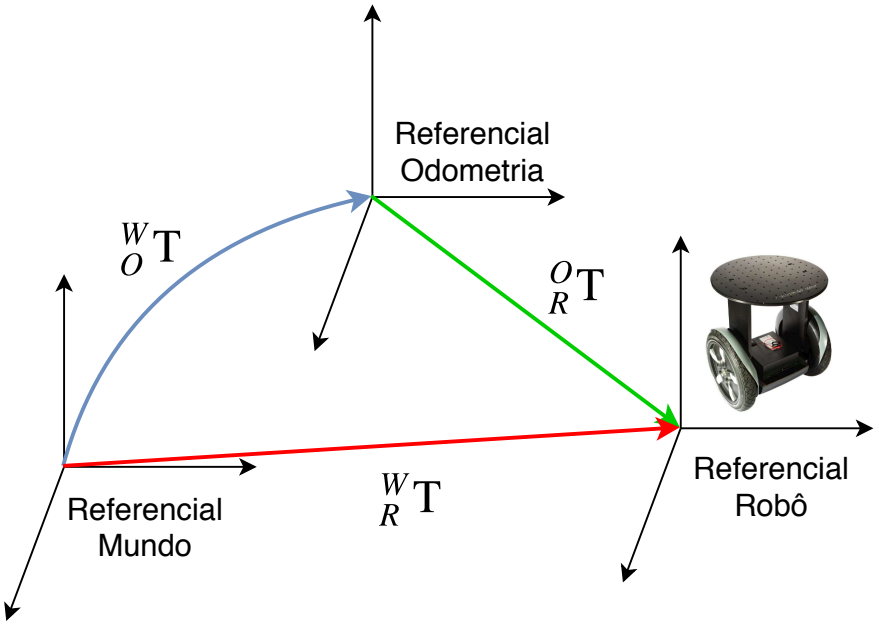


Figura 4.9: Diagrama explicativo das transformações necessárias para a correcção da estimação da odometria.

primeiro por um filtro passa-baixo. Este filtro tem em consideração que a transformação $^{W}_{O}\mathrm{T}$ deve variar muito devagar de acordo com o erro introduzido pela odometria. Ao detectar um marcador, é esperado receber um número considerável de leituras do mesmo, ou seja, várias estimativas da matriz $^{W}_{R}\mathrm{T}_{M}$. No caso de algumas destas leituras serem ruidosas, o filtro irá dar mais peso às leituras anteriores, de modo a que pequenas variações nas leituras tenham pouco efeito em $^{W}_{O}\mathrm{T}$.

O filtro, *exponentally weighted moving average*, tem um parâmetro heurístico, determinado experimentalmente, α, e formula-se do seguinte modo:

$$\begin{aligned} t_x[i+1] &= \alpha \cdot t_x[i] + (1-\alpha) \cdot t_x[i-1] \\ t_y[i+1] &= \alpha \cdot t_y[i] + (1-\alpha) \cdot t_y[i-1] \\ r_z[i+1] &= \alpha \cdot r_z[i] + (1-\alpha) \cdot r_z[i-1] \end{aligned} \tag{4.9}$$

Onde tx corresponde à translação em x, ty à translação em y e r_z à rotação aplicado sobre o eixo dos zz. Aplicado o filtro, a matriz $^{W}_{O}\mathrm{T}$ é reconstruida, e aplicada a equação 4.7 corrigindo assim a estimação da posição da plataforma no referencial do mundo.

4.5 Resumo

Neste capitulo foi abordado o uso de marcadores visuais, falando desde a sua detecção, estimação de posição e orientação, até ao seu contributo para um sistema mais robusto através da correcção da estimação da odometria e do erro acumulado pela mesma com o decorrer do tempo. O capitulo seguinte descreve a forma como todo o sistema foi implementado usando

a arquitectura ROS.

Capítulo 5

Arquitectura e Integração

5.1 Arquitectura modular baseada em *ROS*

Com o objectivo de uma implementação robusta e estandardizada com a comunidade, toda a implementação foi feita usando a arquitectura ROS. Esta arquitectura é *open source* e contém uma gama de ferramentas, livrarias e serviços construidos especificamente a pensar no mundo da robótica. Permite a utilização de diferentes linguagens de programação (C++, Python, Octave e LISP) facilitando assim a adaptação do utilizador ao sistema. Aqui será dada um breve introdução aos conceitos necessários para a compreensão deste capítulo, sendo que uma explicação mais extensiva pode ser encontrada em [47]. Dito isto, a arquitectura ROS cria uma camada de comunicação que facilita a troca de mensagens entre múltiplos nós. Entende-se por nó, um programa que utiliza as ferramentas disponibilizadas pela arquitectura ROS para comunicar e realizar as tarefas pretendidas. A comunicação entre diferentes nós é obtida através de tópicos e serviços, sendo que ambos utilizam mensagens para o efeito. Os tópicos funcionam de forma assíncrona, sendo que quando um nó publica uma mensagem num tópico todos os nós que subscreveram a esse tópico iram receber a mensagem publicada. Por outro lado, os serviços, oferecem uma forma de comunicação síncrona entre apenas dois nós, sendo que é necessário o envio de um pedido e a consequente obtenção de uma resposta, para a comunicação ser bem sucedida.

5.2 Módulos

Para uma arquitecura mais organizada, onde as partes, quando interligadas formam um todo, e devido à simplicidade de criar um sistema modular através do uso do ROS, esta foi a abordagem tomada. A figura 5.2 pretende ilustrar as comunicações entre os diferentes módulos criados.

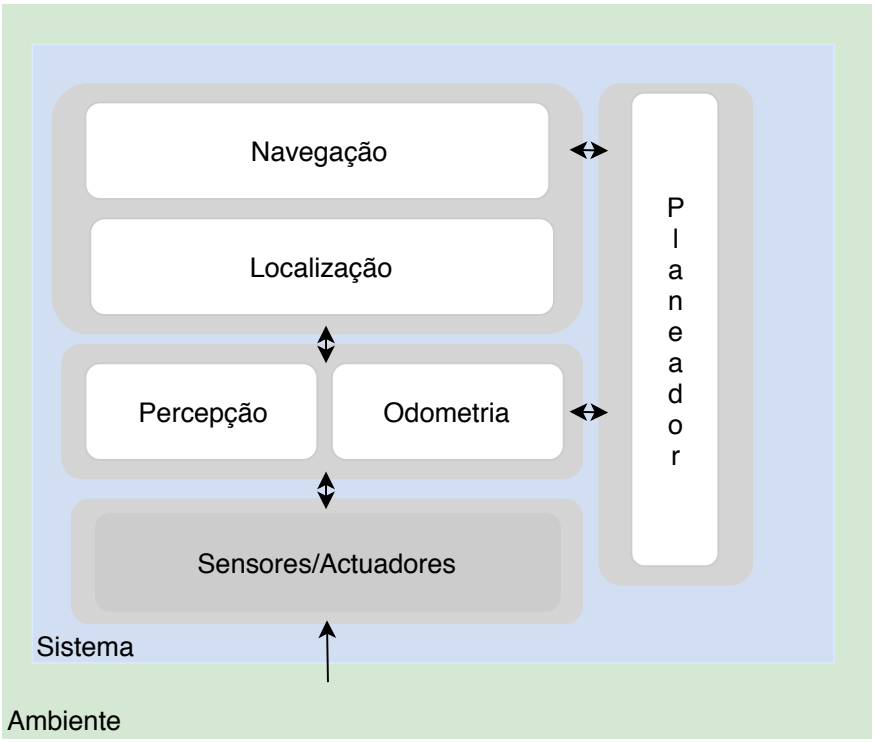


Figura 5.1: Vista geral dos módulos utilizados. Os rectângulos representam os diferentes módulos, sendo que as setas representam a existência de comunicação entre eles.

Dito isto, o sistema pode ser dividido em cinco módulos distintos Planeador, Localização, Navegação, Percepção e Odometria, como pretende ilustrar a figura 5.1.

5.2.1 Módulo de Odometria

Para estabelecer uma comunicação com o controlador presente na plataforma segway RMP 200, foi usado o pacote ROS *segway_rmp*[1]. Este pacote fornece informação relativa à odometria do robô, à percentagem de bateria disponível e possibilita o envio de comandos de velocidade para as rodas motrizes através do tópico */cmd_vel*. Para efectuar a correcção da estimação da posição do robô através da detecção dos marcadores, foi criado o nó *odomUp*. Este nó recebe a informação relativa à estimação da posição do robô no mundo, proveniente do nó *markerDetection*, e através do método descrito no capítulo 4.4, publica uma nova estimativa da localização do robô no tópico */new_odom*. No entanto, para isto acontecer é necessária a detecção dos marcadores visuais. O que nos leva a falar do módulo de localização.

5.2.2 Módulo de Localização

Os conceitos que apoiam o funcionamento deste módulo encontram-se presentes no capítulo 4. Este é maioritariamente constituído por dois serviços, *resetRMP* e *odomError*, sendo que ambos comunicam directamente com o nó responsável pela estimação da posição, *odomUp*. O

[1]Pacote disponível em http://wiki.ros.org/segway_rmp

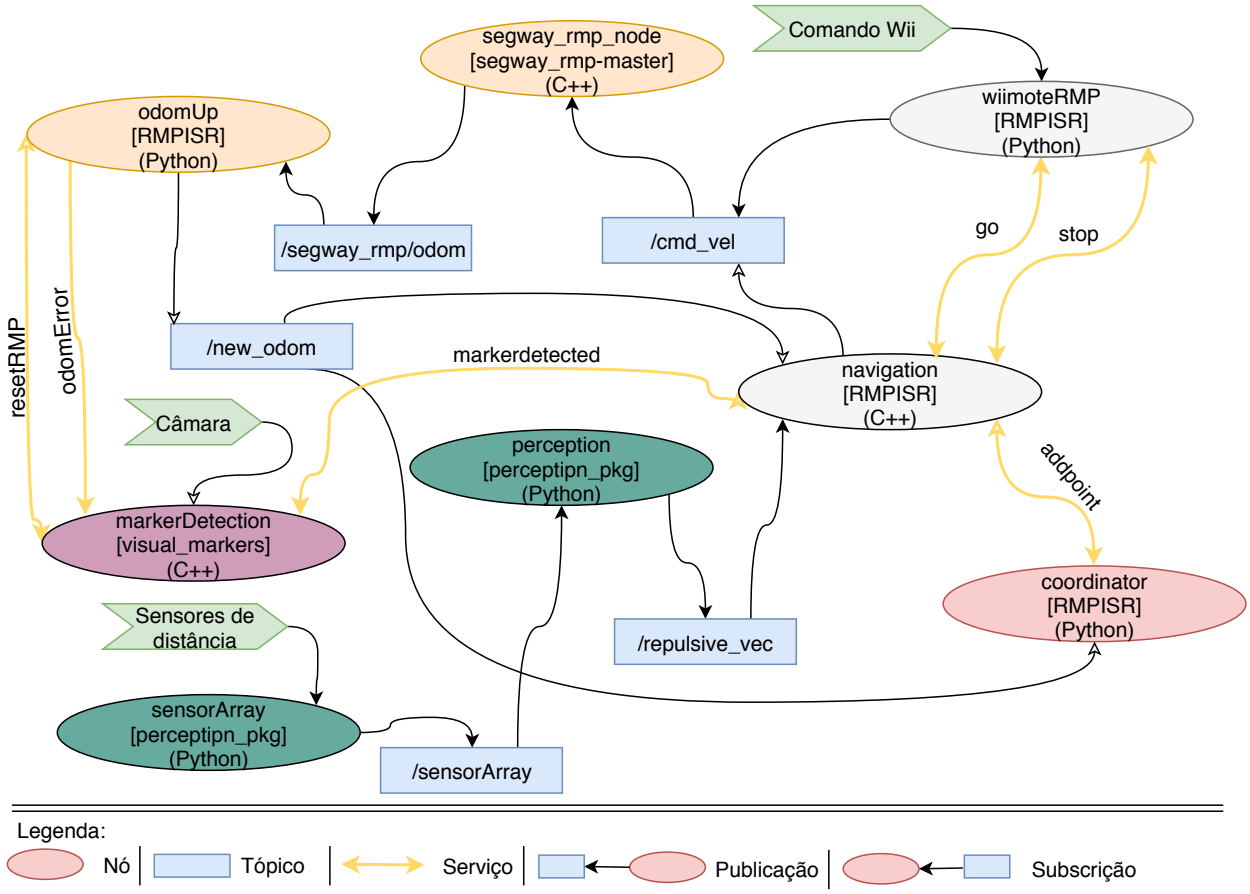


Figura 5.2: Diagrama de ligações entre os diferentes nós de ROS. A notação utilizada para os nós consiste no nome do nó, nome do pacote e linguagem de programação utilizada no mesmo. A cinza encontram-se os pacotes pertencentes à navegação, a laranja os pacotes relacionados com a odometria, a roxo o pacote da localização, a vermelho o pacote do planeamento e a verde os pacotes relacionados com a percepção.

primeiro, *resetRMP*, serve para redefinir a posição inicial, sendo que o segundo, *odomError*, envia, a cada detecção de um marcador visual, a estimação da posição verdadeira para posterior actualização da localização corrigida.

5.2.3 Módulo de Navegação

O módulo de navegação é servidor para dois serviços, *go* e *stop* e cliente para outros dois, *markerdetected* e *addpoint*. Comecemos pelo serviço *addpoint*. Este é parte integrante do módulo de planeamento, mas é no nó *navigation*, constituinte do módulo de navegação, onde a mensagem transportada pelo serviço é usada. Posto isto, este é constituído por dois campos:

- Pontos objectivo: pontos constituintes da trajectória e pelos quais o robô deve passar de modo a atingir o destino
- Variável de Limpeza: variável booleana que indica se os pontos existentes na pilha devem ser ser eliminados ou não.

Uma vez recebida a mensagem transportada pelo serviço, procede-se à criação de duas pilhas. A primeira, com todos os pontos presentes no primeiro campo do serviço, criando assim a pilha de pontos objectivo, e a segunda, com o ultimo ponto do primeiro campo do serviço, pilha de pontos destino. Por cada vez que este serviço é chamado, um novo ponto destino é adicionado, e portanto um novo ponto de paragem para o robô. Implementou-se assim a funcionalidade da criação de um itinerário, com paragem em diferentes pontos de interesse para o utilizador.

Relativamente à variável de limpeza, caso esta tenho o valor *True*, esta vai indicar ao nó *navigation* que deve apagar todos os pontos já presentes em ambas as pilhas e adicionar apenas os que acabou de receber, dando assim inicio a um novo itinerário. No caso de ter o valor *False*, deve apenas adicionar os pontos ao fim das respectivas pilhas, adicionando assim novos pontos de paragem ao itinerário já existente.

O funcionamento deste módulo pode ser representado através da maquina de estados apresentada na figura 5.3. O sistema começa num estado de inatividade, esperando o sinal de inicio por parte do utilizador. Esse sinal de início é uma chamada a um serviço *ROS* que modifica o estado do sistema para o modo *Go*. Aqui é verificado se na pilha de pontos objectivo, ainda existem pontos por alcançar ou se a pilha se encontra vazia. Caso existam, o robô dá inicio ao algoritmo de navegação, e por cada ponto que o robô alcance com sucesso, esse ponto é retirado da pilha e o próximo ponto torna-se o ponto objectivo. Este procedimento é realizado até o ponto destino ser alcançado, sendo que quando esta condição se verificar o robô cessa o movimento.

É considerado que o robô passou por um ponto objectivo, ou chegou ao ponto destino, quando a distância euclidiana a esse ponto, for inferior a um determinado limiar.

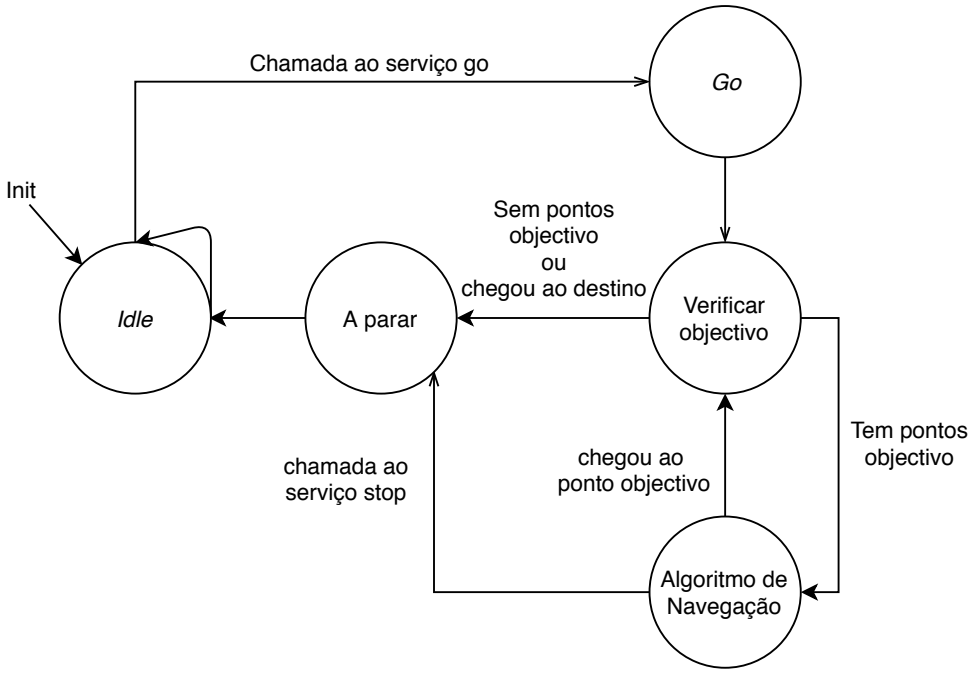


Figura 5.3: Máquina de estados representativa do funcionamento do módulo de navegação.

De modo a efectuar a chamada aos serviços modificadores do estado do sistema, usou-se

um comando wii, figura 5.4, e o pacote ROS *wiimote* [2]. Para realizar a chamada ao serviço *go* utiliza-se o botão com o símbolo (+), sendo que para realizar a chamada ao serviço *stop* utiliza-se o botão com o símbolo (-). A funcionalidade de controlar remotamente o robô também se encontra disponível através dos botões direccionais presentes no topo do comando.

Figura 5.4: Comando Wii usado para controlo remoto do robô e para a realização de chamadas a serviços ROS.

Após a implementação e primeiros testes do sistema, reparou-se que em algumas situações o robô após a detecção de um marcador, iniciava um rotação sobre o seu centro com o intuito de voltar atrás na trajectória. Isto era devido ao facto de depois da detecção do marcador e respectiva actualização da posição, a posição corrigida do robô já ter ultrapassado o ponto objectivo actual, estando este agora situado na rectaguarda do robô. De modo a resolver este problema, de cada vez que um ponto objectivo é atingido com sucesso, é calculado um versor na direcção do ponto objectivo seguinte, e de cada vez que um marcador é detectado um versor para o ponto objectico actual é igualmente calculado. Obtendo estes dois versores, é então calculado o produto interno entre eles, equação 5.1. O resultado do produto interno está entre um de três:

- -1: caso os versores tenham direcções opostas
- 0: caso os versores sejam perpendiculares
- 1: no caso dos versores terem a mesma direcção

Assim, caso o resultado seja -1 e os versores tenham direcções opostas, o navegador passa para o ponto objectivo seguinte e o robô continua a trajectória pretendida.

$$\vec{A} \cdot \vec{B} = A_x B_x + A_y B_y \tag{5.1}$$

A figura 5.5 pretende ilustar um exemplo desta problemática.

[2]Pacote disponível em http://wiki.ros.org/wiimote

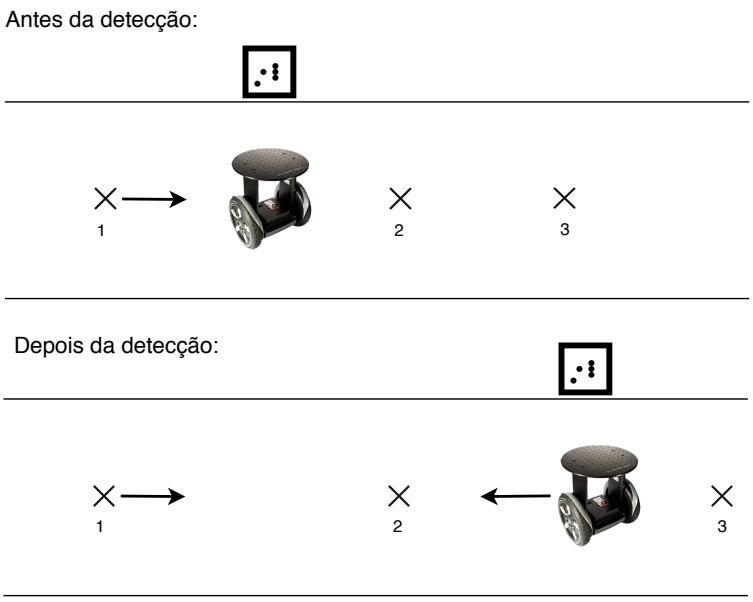


Figura 5.5: Este exemplo é constituído por 3 pontos objectivo. Considera-se que o ponto objectivo 1 foi alcançado com sucesso e portanto foi calculada a direcção do versor para o próximo ponto objectivo, neste caso o ponto 2. No entanto, entre estes dois pontos existe um marcador visual, o qual foi detectado e portanto calculada a posição corrigida do robô, tendo esta já ultrapassado o ponto objectivo 2. Aquando da detecção do marcador é calculado um novo versor de direcção e caso estes tenham direcções opostas, como representado na imagem, o ponto objectivo deixa de ser o ponto 2 e passa a ser o ponto 3.

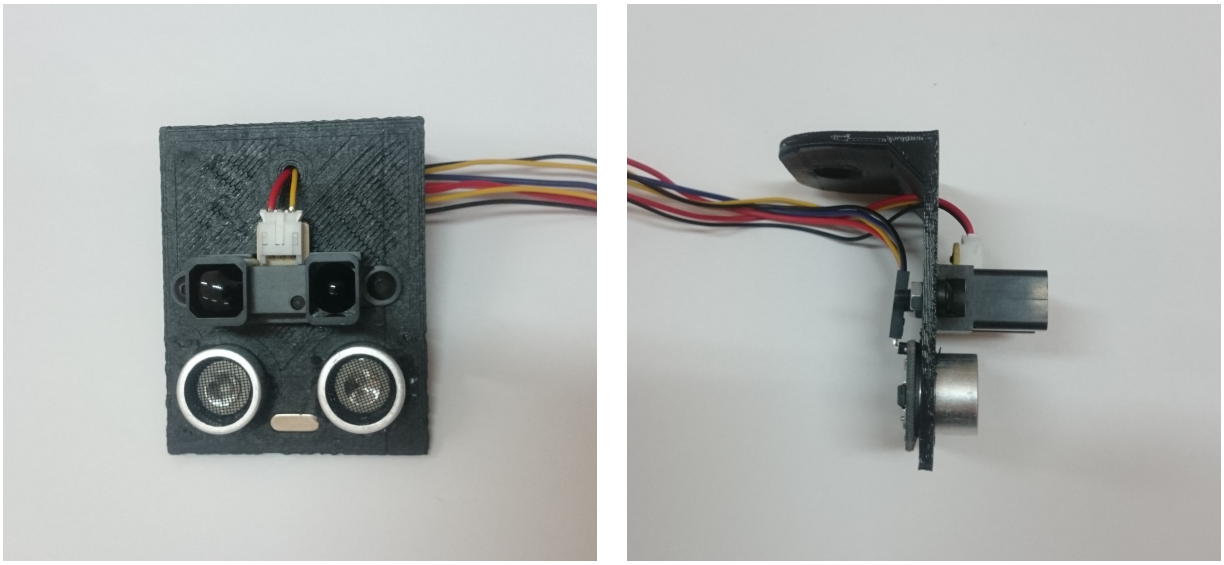
5.2.4 Módulo de Percepção

Este módulo é responsável pelo processamento e tratamento dos dados recolhidos pelos sensores de ultrasom e de infravermelhos. Para a comunicação com os sensores e subsequente recolha das leituras, foi usado um *Arduino Uno R3* que posteriormente envia os dados através de uma porta série, para o controlador principal. Este, por sua vez, procede à criação do tópico */sensorArray*, onde as leituras serão publicados para assim ficarem disponíveis no sistema. O nó *perception*, subscreve a esse tópico e procede ao tratamento dos dados, calculando os respectivos vectores repulsivos, como explicado na secção 3.3.1. Uma vez que esses vectores tenham sido calculados e somados, o vector repulsivo final é publicado no tópico */repulsive_ vec*, que por sua vez vai ser subscrito pelo nó responsável pela locomoção, *navigation*. O esquemático de ligações entre os diferentes sensores e o arduino encontra-se presente na figura 5.7.

Aplicação dos Sensores à Plataforma

Inicialmente a plataforma estava desprovida de qualquer tipo de sensores de distância. De forma a ser possível dotar o robô de capacidades de percepção, este foi equipado com três sensores de ultrasom e quatro de infravermelhos (como representado na figura 3.10), sendo necessária a criação de suportes de fixação dos sensores ao tampo do mesmo. Para tal, foram desenhadas usando ferramentas de *Computer Aided Design* (CAD) do programa *Fusion360*, dois tipos de suportes diferentes. Um para suportar um sensor de infravermelhos e um de

ultrasom, como o presente na figura 5.6, e outro para suportar apenas um sensor de ultrasom. Ambas as peças se encontram disponíveis online na plataforma *Thingiverse* [3].



(a) Vista frontal (b) Vista lateral

Figura 5.6: Suporte para os sensores de ultrasom e de infravermelhos produzido através de uma impressora 3D.

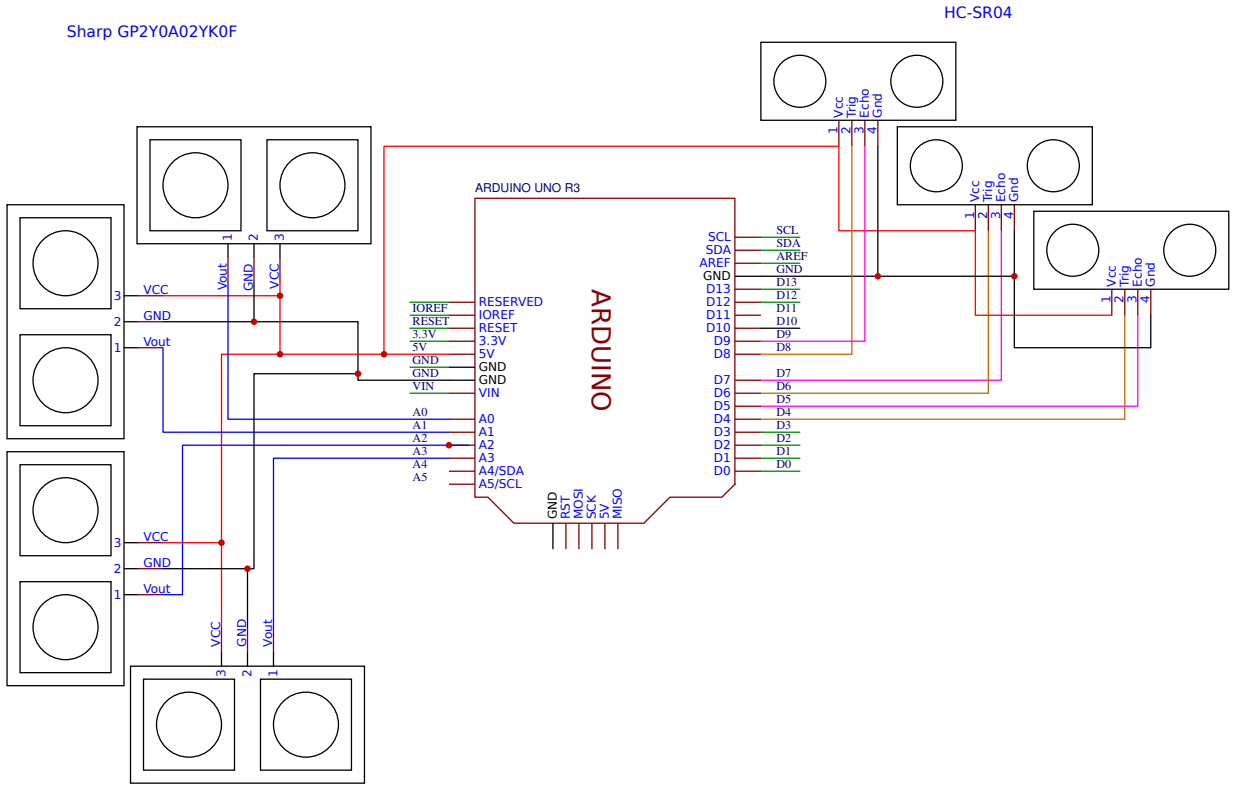


Figura 5.7: Esquema de ligação dos sensores ao Arduino.

[3]https://www.thingiverse.com/thing:2987475

5.2.5 Módulo Planeador

Este módulo é responsável pelo planeamento da trajectória e pela representação gráfica do mapa, trajectória a seguir e posição do robô.

O planeador recebe um mapa do tipo *.xml*. Este ficheiro na sua constituição tem a seguinte estrutura:

```
<map>
  <segment>
    <point><x>0.0</x><y>0.0</y></point>
    <point><x>0.0</x><y>20.4</y></point>
  </segment>
  <segment>
    <point><x>0.0</x><y>20.4</y></point>
    <point><x>-5.8</x><y>20.4</y></point>
  </segment>
  <segment>
    ...
  </segment>
  ...
</map>
```

Cada segmento de recta é constituído por dois pontos cartesianos. Estes segmentos são considerados como uma fronteira entre o espaço livre e o espaço ocupado. Portanto, estes necessitam de ser orientados. A orientação é definida da seguinte forma: indo do primeiro ponto para o segundo, o espaço que fica à direita desse segmento é considerado espaço livre.

Uma vez definido o mapa, a representação gráfica do mesmo foi feita através da biblioteca *Pygame*[4] e encontra-se representada na figura 5.8.

Para realizar o planeamento de um trajectória, o planador recebe a posição actual e a posição destino, devolvendo uma pilha de pontos intermédios que serão enviados para o nó *navigation* através do serviço addpoint descrito anteriormente.

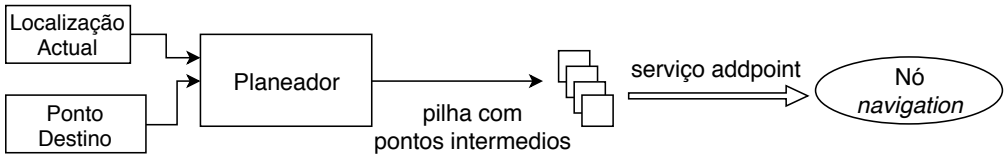


Figura 5.9: Ilustração do funcionamento do planeador e interacção com o módulo de navegação.

[4]Esta biblioteca pode ser encontrada em https://www.pygame.org/wiki/about

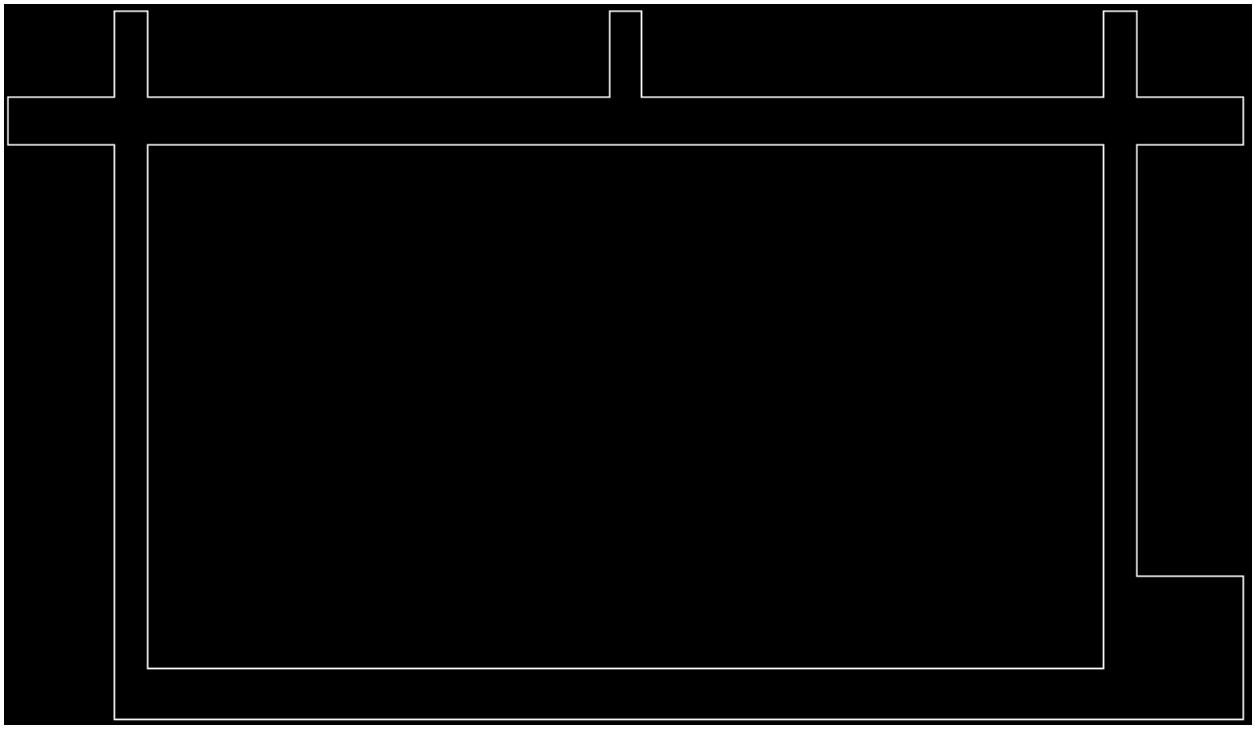


Figura 5.8: Representação gráfica do mapa usado pelo planeador.

5.3 Interface Gráfica

Foi criada uma interface gráfica com o intuito de comunicação com o utilizador final. Este utilizador pode ser um visitante ao ISR, ou um grupo de alunos que venha conhecer os vários laboratórios existentes. Com base nestes casos de uso, foi desenvolvida uma interface gráfica simples e de entendimento fácil, facilitando assim o processo de utilização da mesma.

Inicialmente o utilizador encontra a janela inicial presente na figura 5.10. Aqui, pode escolher entre ir para um sitio em especifico ou escolher entre um dos itinerários de visita pré-definidos. No primeiro caso a janela da figura 5.11 (a) é exibida e o utilizador escolhe o destino pretendido. Caso o utilizador escolha a segunda opção, é exibida a janela da figura 5.11 (b), e os itinerários disponíveis são exibidos.

Escolhido o destino, e dada a indicação ao robô para iniciar a viagem, a janela da figura 5.12 (a) é exibida. Quando chegar ao destino, o utilizador irá ser avisado através da janela da figura 5.12 (b). Aqui é questionado sobre a próxima decisão a tomar. Caso se queira dirigir a outro sitio, assim o pode fazer, caso não queira, pode enviar o robô para a sua localização de repouso, onde irá esperar por outro utilizador.

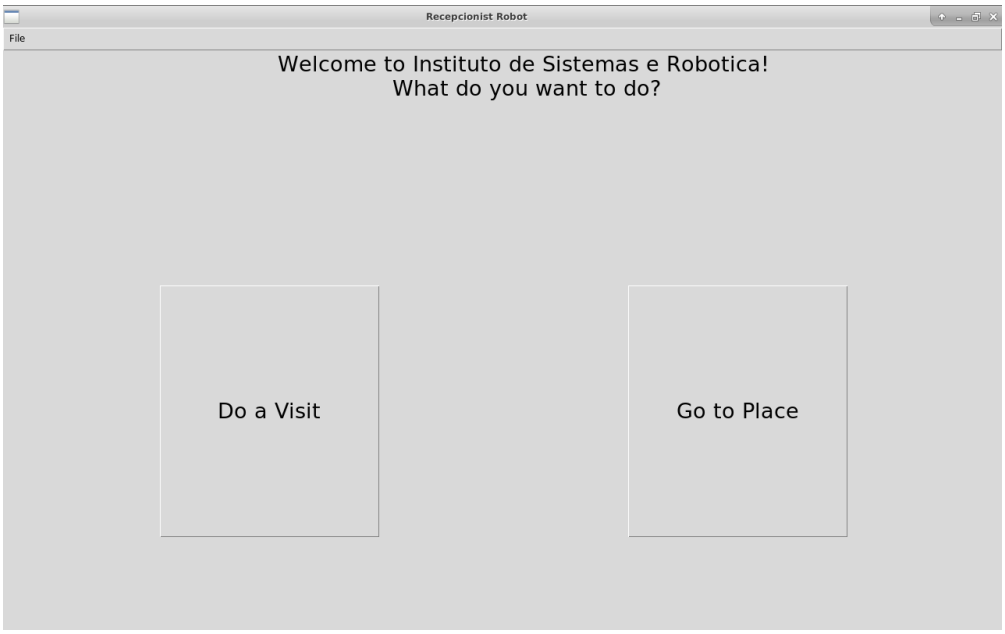
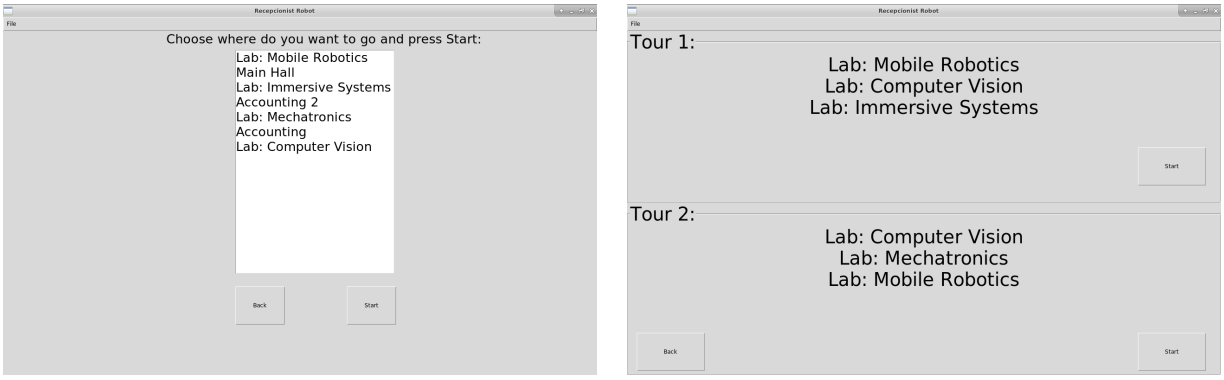


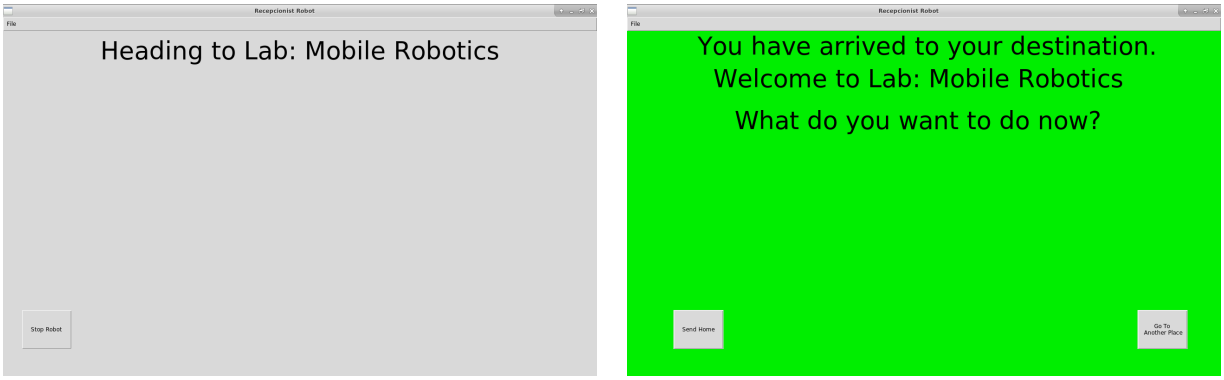
Figura 5.10: Janela principal da interface gráfica.



(a) Escolha do local a visitar

(b) Escolha do itinerário a realizar

Figura 5.11: Janelas de escolha do sitio destino na interface gráfica.



(a) Janela exibida quando o robô se encontra em movimento

(b) Janela exibida quando o robô chega ao destino

Figura 5.12: Janelas informativas do estado do robô.

Capítulo 6

Resultados e Discussão

De maneira a avaliar o desempenho do sistema, foram realizados testes experimentais em ambiente real. Tendo em conta que o sistema desenvolvido pretende criar a base para um robô recepcionista, a maneira mais apropriada para o testar seria analisar o sucesso da tarefa de navegação, tendo em consideração a chegada do robô ao destino e o erro associado. Outra análise importante seria observar a melhoria na localização resultante do uso dos marcadores visuais.

6.1 Condições de Realização da Experiência

A experiência realizada consiste na navegação através do percurso ilustrado na figura 6.1. O robô parte da posição assinalada pela bandeira axadrezada, dirigindo-se em direcção ao ponto A, seguido do ponto B, voltando ao ponto A e terminando no sitio de onde partiu. Em cada ponto mencionado anteriormente, o robô efectua uma paragem e a distância da posição real de paragem do robô em relação ao ponto pretendido para a paragem é medida. A condição de paragem num ponto, é definida através da distância euclidiana a esse ponto ser inferior a 20 centímetros.

A posição dos marcadores para correcção da estimação da posição e orientação ao longo da trajectória, encontra-se presente na tabela 6.1 e uma ilustração da posição dos mesmos em relação ao percurso efectuado na figura 6.1. O percurso foi realizado quatro vezes, com velocidades de 0.2, 0.3 e 0.6 m/s.

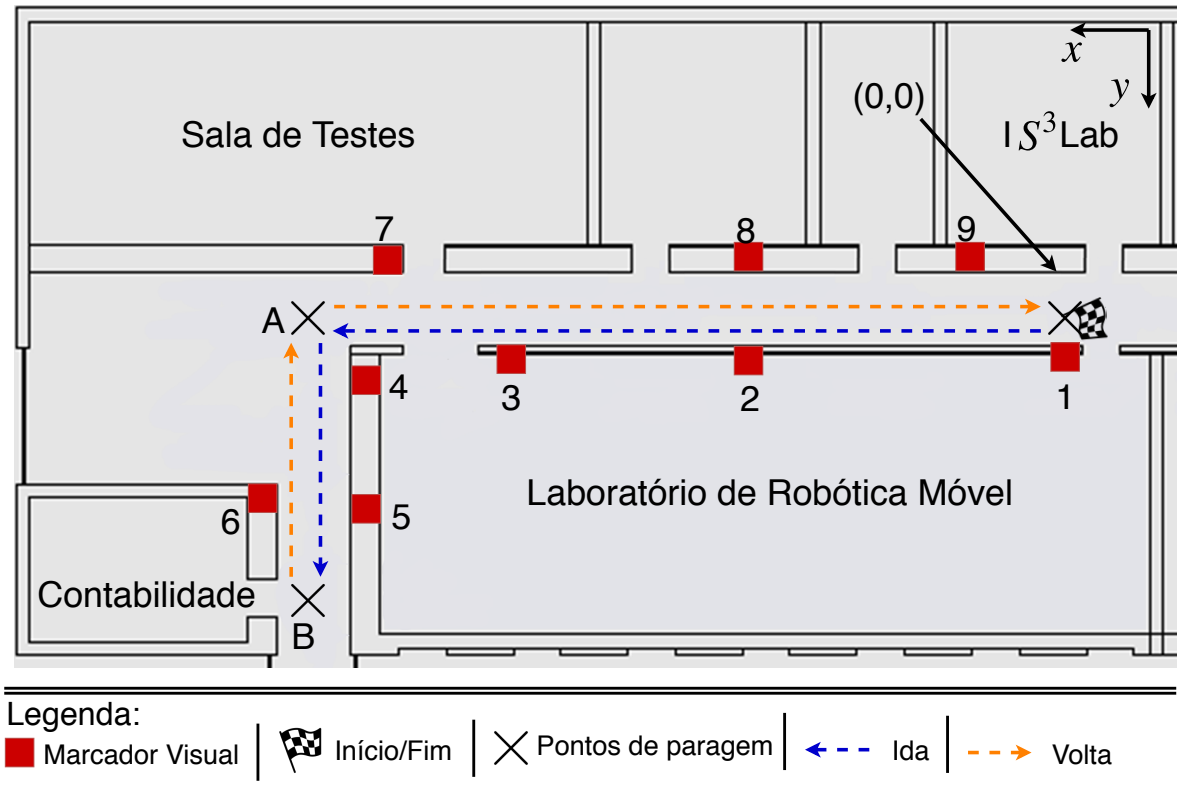


Figura 6.1: Percurso realizado durante os testes experimentais.

Tabela 6.1: Coordenadas xyz da posição dos marcadores no referencial do mundo, estando todos orientados para o interior do corredor.

	1	2	3	4	5	6	7	8	9
x [m]	0.00	7.00	13.95	17.38	17.38	19.14	16.21	7.00	0.88
y [m]	1.80	1.80	1.80	3.02	6.48	5.14	0.00	0.00	0.00
z [m]	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83

6.2 Resultados

Para a análise dos dados da experiência foram usadas como métricas a média (μ) e o desvio padrão (σ) da distância euclidiana do ponto real de paragem do robô ao ponto real de destino.

Tabela 6.2: Resultados obtidos durante a experiência.

		Distância Euclidiana [cm]			Tempo Decorrido [min]	% Sucesso
		μ	σ	desvio máximo		
$v = 0.2$ [m/s]	Ponto A	20.96	6.438	29.76	7:53:6	100%
	Ponto B	17.01	9.115	30.06		
	Ponto A	25.64	12.235	36.49		
	Fim	23.18	9.979	31.95		
$v = 0.3$ [m/s]	Ponto A	14.84	7.536	23.19	5:01:4	100%
	Ponto B	12.28	6.865	21.47		
	Ponto A	10.91	11.974	28.02		
	Fim	15.73	10.807	28.65		
$v = 0.6$ [m/s]	Ponto A	17.42	3.856	21.84	2:32:0	75%
	Ponto B	32.55	4.569	35.46		
	Ponto A	20.66	13.721	36.40		
	Fim	29.82	8.096	38.27		

Na tabela 6.2 encontram-se os resultados dos testes realizados. Nesta pode-se verificar que para as velocidades de 0.2 e 0.3 m/s a distância euclidiana média de paragem nos diferentes pontos de teste está próxima do limiar definido, que é de 20 centímetros. Já com a velocidade de 0.6 m/s o robô ficou a uma distância euclidiana média superior ao limiar, podendo esta ser justificada através do momento que o robô possui na chegada ao ponto destino, não cessando o movimento imediatamente após dada a indicação de paragem, e caso tenha um pequeno erro de orientação, devido ao número de leituras efectuada por marcador a esta velocidade ser inferior, este vai-se traduzir numa distância euclidiana ao ponto maior.

No entanto, o objectivo deste trabalho é passar o suficientemente próximo dos pontos objectivo de modo a possibilitar a realização da trajectória, e não a passagem exatamente pelo ponto objectivo. Esta escolha foi tomada, pois caso fosse necessário passar o mais próximo possível do ponto, seria necessário fazer um controlo mais fino e reduzir a velocidade de modo a obter essa exactidão. Assim, o critério adoptado foi um critério mais permissivo. Posto isto, os resultados mostraram-se bastante satisfatórios.

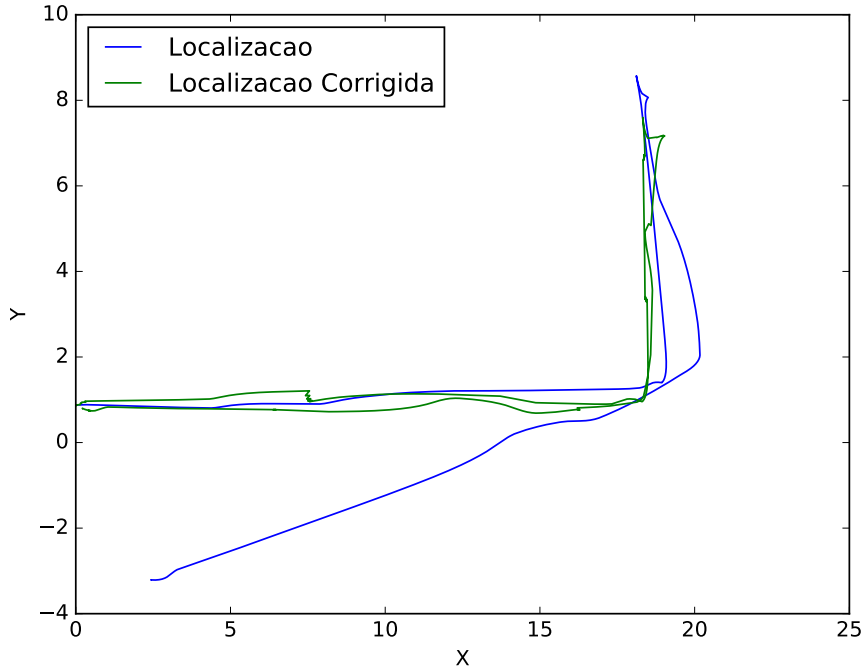
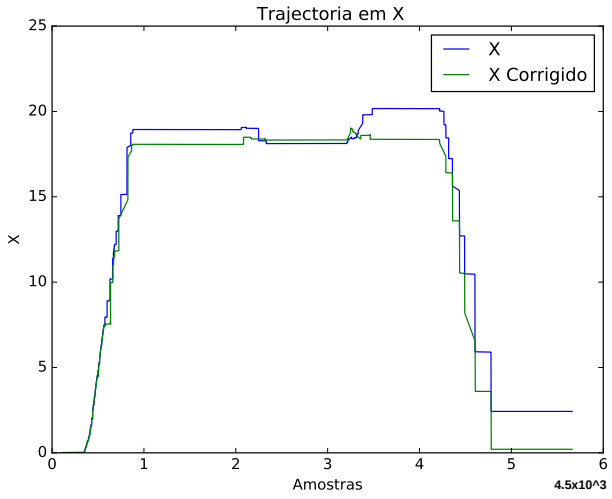
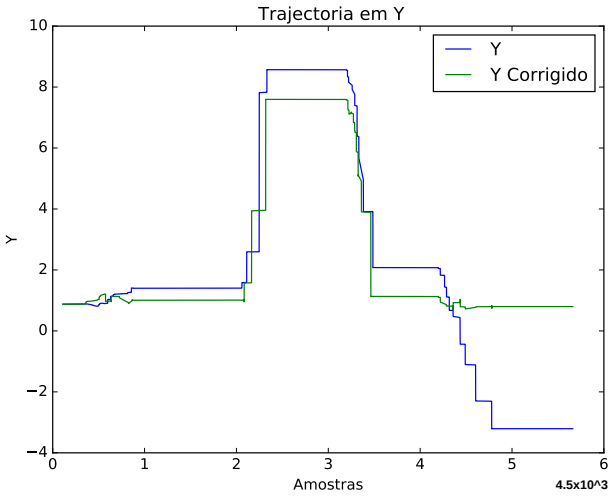


Figura 6.2: Representação de uma das trajectórias realizadas durante os testes experimentais.



(a) Evolução da posição em x



(b) Evolução da posição em y

Figura 6.3: Evolução da posição do robô no eixo de coordenadas x e y, respectivamente.

Tabela 6.3: Erro na coordenada x em relação à figura 6.3 (a), y em relação à figura 6.3 (b) e xy em relação à figura 6.2.

	$\|\mu\|$	σ	máximo
x [m]	1.555	2.002	1.740
y [m]	1.382	1.408	4.012
xy [m]	2.319	2.223	4.579

Na figura 6.2 encontra-se representada a trajectória realizada durante o percurso de testes, estando representada a localização através da odometria e a localização corrigida pelos marcadores. É possível verificar a notória melhoria proporcionada pelos marcadores visuais colocados ao longo do percurso, e como a localização estimada através da odometria diverge do mesmo, chegando a ter erros de $1.74\,m$ em x e $4.012\,m$ em y. Na figura 6.3 (a) encontra-se representada a evolução da posição na coordenada x e na figura 6.3 (b) a evolução da posição na coordenada y. A tabela 6.3 contem a média em módulo, o desvio padrão e erro máximo entre a estimação de posição através da odometria e a estimação corrigida pelos marcadores.

Capítulo 7

Conclusão e Trabalho Futuro

7.1 Conclusão

Ao longo deste trabalho foi desenvolvido um sistema com vista na criação de um robô recepcionista para o piso 0 do Instituto de Sistemas e Robótica, explorando a utilização de marcadores visuais para uma melhor estimação da localização da plataforma móvel. Os testes de desempenho efectuados permitem concluir que as soluções implementadas tem um desempenho satisfatório e que o uso dos marcadores visuais é uma abordagem viável, não intrusiva e barata. No entanto, para condições de ambiente nocturno, onde a luminosidade é baixa, a solução implementada não é recomendada. Os algoritmos utilizados não se encontram optimizados, sendo que uma implementação na linguagem C++ irá melhorar consideravelmente os tempos de processamento.

7.2 Trabalho Futuro

Para trabalho futuro existem alguns melhoramentos e diferentes abordagens que se podem realizar. Entre elas estão:

- O melhoramento do algoritmo de navegação para a realização de uma trajectória mais suaves, utilizando por exemplo o método de bandas elásticas [48]
- Estudar a utilização de outro tipo de filtragem para a estimação da localização corrigida, como o filtro de Kalman, explorando os modelos de erro dos sensores
- Melhoramento da aparência do interface gráfico

No entanto, no desenvolvimento de um robô interactivo, faz todo o sentido existirem ainda outras funcionalidades extra que se podem implementar futuramente como:

- A utilização de uma câmara a detectar o utilizador e a variar a velocidade de locomoção do robô de acordo com a velocidade da pessoa
- A comunicação com o utilizador ser feita por voz para uma interação mais natural

Bibliografia

[1] YUFKA Alpaslan and PARLAKTUNA Osman. Performance comparison of bug algorithms for mobile robots. In *5th International Advanced Technologies Symposium (IATS'09)*, volume 13, 2009.

[2] J Randolph Andrews and Neville Hogan. Impedance control as a framework for implementing obstacle avoidance in a manipulator. Master's thesis, M. I. T., Dept. of Mechanical Engineering, 1983.

[3] Aurélien Kamel Anthony Mallet. libvstp — very simple trajectory planner. Technical report, LAAS-CNRS, France, 2003.

[4] Kai O Arras, Nicola Tomatis, Björn T Jensen, and Roland Siegwart. Multisensor on-the-fly localization:: Precision and reliability for applications. *Robotics and Autonomous Systems*, 34(2-3):131–143, 2001.

[5] Joydeep Biswas and Manuela Veloso. Wifi localization and navigation for autonomous indoor mobile robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4379–4384. IEEE, 2010.

[6] Johann Borenstein. Experimental results from internal odometry error correction with the omnimate mobile robot. *IEEE Transactions on Robotics and Automation*, 14(6):963–969, 1998.

[7] Johann Borenstein and Liqiang Feng. Gyrodometry: A new method for combining data from gyros and odometry in mobile robots. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pages 423–428. IEEE, 1996.

[8] Johann Borenstein and Liqiang Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on robotics and automation*, 12(6):869–880, 1996.

[9] Johann Borenstein and Yoram Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 7(3):278–288, 1991.

[10] David M Bradley. Odometry: Calibration and error modeling. *Robotics Institute Carnegie Mellon University, Pittsburgh, Pensylvania*, 15213.

[11] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 1986.

[12] Rodney A Brooks and Jonathan H Connell. Asynchronous distributed control system for a mobile robot. In *Mobile Robots I*, volume 727, pages 77–85. International Society for Optics and Photonics, 1987.

[13] Wolfram Burgard, Armin B Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artificial intelligence*, 114(1-2):3–55, 1999.

[14] RH Byrne, PR Klarer, and JB Pletta. Techniques for autonomous navigation. 1992.

[15] Raja Chatila and Jean-Paul Laumond. Position referencing and consistent world modeling for mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 138–145. IEEE, 1985.

[16] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[17] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, (6):46–57, 1989.

[18] Dieter Fox. Kld-sampling: Adaptive particle filters. In *Advances in neural information processing systems*, pages 713–720, 2002.

[19] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999(343-349):2–2, 1999.

[20] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[21] Sinan Gezici, Zhi Tian, Georgios B Giannakis, Hisashi Kobayashi, Andreas F Molisch, H Vincent Poor, and Zafer Sahinoglu. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *IEEE signal processing magazine*, 22(4):70–84, 2005.

[22] Cimini Gionata, Ferracuti Francesco, Freddi Alessandro, Iarlori Sabrina, and Monteriù Andrea. An inertial and qr code landmarks-based navigation system for impaired wheelchair users. In *Ambient Assisted Living*, pages 205–214. Springer, 2014.

[23] P Goel, SI Roumeliotis, and GS Sukhatme. Robot localization using relative and absolute position estimates in proc ieee int. In *Conference on Robots and Systems, October, Kyongju, Korea*, 1999.

[24] Michael A Goodrich, Alan C Schultz, et al. Human–robot interaction: a survey. *Foundations and Trends® in Human–Computer Interaction*, 1(3):203–275, 2008.

[25] Huosheng Hu, Michael Brady, and Penelope Probert. Coping with uncertainty in control and planning for a mobile robot. In *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*, pages 1025–1030. IEEE, 1991.

[26] Gijeong Jang, Sungho Lee, and Inso Kweon. Color landmark based self-localization for indoor mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 1037–1042. IEEE, 2002.

[27] Klaus-Werner Jörg, TORSTEN Gattung, and JOACHIM Weber. Supporting mobile robot localization by visual bar code recognition. In *Proc. IASTED International Conference on Robotics and Applications, IASTED/Acta Press*, pages 352–357, 1999.

[28] Ishay Kamon and Ehud Rivlin. Sensory-based motion planning with global proofs. *IEEE transactions on Robotics and Automation*, 13(6):814–822, 1997.

[29] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[30] Gunhee Kim, Woojin Chung, Kyung-Rock Kim, Munsang Kim, Sangmok Han, and Richard H Shinn. The autonomous tour-guide robot jinny. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3450–3455. IEEE, 2004.

[31] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1398–1404. IEEE, 1991.

[32] Bruce Krogh. A generalized potential field approach to obstacle avoidance control. In *Proc. SME Conf. on Robotics Research: The Next Five Years and Beyond, Bethlehem, PA, 1984*, pages 11–22, 1984.

[33] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[34] N James Lavigne and Joshua A Marshall. A landmark-bounded method for large-scale underground mine mapping. *Journal of Field Robotics*, 29(6):861–879, 2012.

[35] Ren C Luo, Shih-Chi Lin, and Chun Chi Lai. Indoor autonomous mobile robot localization using natural landmark. In *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, pages 1626–1631. IEEE, 2008.

[36] Evgeni Magid and Ehud Rivlin. Cautiousbug: a competitive algorithm for sensory-based robot navigation. In *IROS*, pages 2757–2762, 2004.

[37] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 300–307. IEEE, 2010.

[38] Gonçalo Martins. A cooperative slam framework with efficient information sharing over mobile ad hoc networks. Master's thesis, Faculty of Sciences and Technology - University of Coimbra, 2014.

[39] João Martins. Localização e mapeamento simultâneo com múltiplos robôs. Master's thesis, Faculdade de Ciências e Tecnologia - Universidade de Coimbra, 2012.

[40] Paulo Menezes. Open augmented reality (openar). Available:http://orion.isr.uc.pt.

[41] Hans P Moravec. Sensor fusion in certainty grids for mobile robots. *AI magazine*, 9(2):61, 1988.

[42] Hans P Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 19–24, 1985.

[43] Masoud Nosrati, Ronak Karimi, and Hojat Allah Hasanvand. Investigation of the*(star) search algorithms: Characteristics, methods and approaches. *World Applied Programming*, 2(4):251–256, 2012.

[44] Masoud Nosrati, Ronak Karimi, and Hojat Allah Hasanvand. Investigation of the*(star) search algorithms: Characteristics, methods and approaches. *World Applied Programming*, 2(4):251–256, 2012.

[45] Brian Olszewski, Steven Fenton, Brian Tworek, Jiao Liang, and Kumar Yelamarthi. Rfid positioning robot: An indoor navigation system. In *Electro/Information Technology (EIT), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013.

[46] Bruno Patrão. Biblioteca para desenvolvimento de aplicações de realidade aumentada em marcadores binários. Master's thesis, Faculty of Sciences and Technology - University of Coimbra, 2011.

[47] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[48] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 802–807. IEEE, 1993.

[49] Volkan Sezer and Metin Gokasan. A novel obstacle avoidance algorithm:"follow the gap method". *Robotics and Autonomous Systems*, 60(9):1123–1134, 2012.

[50] Roland Siegwart, Kai O Arras, Samir Bouabdallah, Daniel Burnier, Gilles Froidevaux, Xavier Greppin, Björn Jensen, Antoine Lorotte, Laetitia Mayor, Mathieu Meisser, et al. Robox at expo. 02: A large-scale installation of personal robots. *Robotics and Autonomous Systems*, 42(3-4):203–222, 2003.

[51] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *IJCAI*, volume 95, pages 1080–1087, 1995.

[52] Armando Sousa, Catarina Santiago, Paulo Malheiros, Paulo Costa, and António Paulo Moreira. Using barcodes for robotic landmarks. In *Fourteenth Portuguese Conference on Artificial Intelligence', Aveiro (Portugal)*, 2009.

[53] Anthony Stentz. The d* algorithm for real-time planning of optimal traverses. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1994.

[54] Ioan Susnea, Adrian Filipescu, Grigore Vasiliu, George Coman, and Adrian Radaschin. The bubble rebound obstacle avoidance algorithm for mobile robots. In *Control and Automation (ICCA), 2010 8th IEEE International Conference on*, pages 540–545. IEEE, 2010.

[55] Sebastian Thrun and Arno Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 944–951, 1996.

[56] Robert B Tilove. Local obstacle avoidance for mobile robots based on the method of artificial potentials. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 566–571. IEEE, 1990.

[57] Kuk-Jin Yoon and In-So Kweon. Landmark design and real-time landmark tracking for mobile robot localization. In *Mobile Robots XVI*, volume 4573, pages 219–227. International Society for Optics and Photonics, 2002.

[58] Da Zhang, Feng Xia, Zhuo Yang, Lin Yao, and Wenhong Zhao. Localization technologies for indoor human tracking. In *Future Information Technology (FutureTech), 2010 5th International Conference on*, pages 1–6. IEEE, 2010.

[59] Da Zhang, Feng Xia, Zhuo Yang, Lin Yao, and Wenhong Zhao. Localization technologies for indoor human tracking. In *Future Information Technology (FutureTech), 2010 5th International Conference on*, pages 1–6. IEEE, 2010.

[60] Huijuan Zhang, Chengning Zhang, Wei Yang, and Chin-Yin Chen. Localization and navigation using qr code for mobile robot in indoor environment. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 2501–2506. IEEE, 2015.

[61] Muhammad Zohaib, M Pasha, RA Riaz, Nadeem Javaid, Manzoor Ilahi, and RD Khan. Control strategies for mobile robot with obstacle avoidance. *arXiv preprint arXiv:1306.1144*, 2013.

[62] Muhammad Zohaib, Syed Mustafa Pasha, Nadeem Javaid, and Jamshed Iqbal. Iba: Intelligent bug algorithm–a novel strategy to navigate mobile robots autonomously. In *International Multi Topic Conference*, pages 291–299. Springer, 2013.