NAVEGAÇÃO DE ROBÔS MÓVEIS POR MARCAS ARTIFICIAIS EM AMBIENTES SEMI-ESTRUTURADOS

João Manuel Castanheira Teixeira

Mestrado em Engenharia Electrotécnica e de Computadores
Instituto Superior de Engenharia do Porto
2013

Tese/Dissertação, 2º ano, Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: João Manuel Castanheira Teixeira,
Nº 1030340, 1030340@isep.ipp.pt

Orientador: Eduardo Alexandre Pereira da Silva, eaps@lsa.isep.ipp.pt

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Sistemas Autónomos
Departamento de Engenharia Electrotécnica

Agradecimentos

Agradeço aos meus pais, por o seu apoio incondicional.

Agradeço aos meus amigos de longa data, bem como aos que criei neste mestrado (Pedro Gonçalves, Maria Costa e Hugo Queirós) pelos quais tenho muita estima.

Aos engenheiros Engs: Eduardo Silva, Eng. Alfredo Martins, Luís Miguel Lima, e a todos os outros que contribuíram directa e indirectamente para a realização desta dissertação.

Quero também agradecer a todo o pessoal que constitui o Laboratório de sistemas autónomos (LSA) por me proporcionar uma formação de elevada exigência e qualidade bem como ao Instituto Superior de Engenharia do Porto.

Resumo

A utilização de sistemas autónomos em ambientes industriais de pequena e média dimensão ainda se encontra bastante condicionado, essencialmente pela elevada relação custo/benefício que decorre dos elevados custos destes sistemas e dos "modestos" desempenhos face aos requisitos requeridos.

Esta dissertação aborda a navegação de sistemas autónomos em ambientes "indoor" estruturados, tendo como proposta desenvolver uma solução para a localização baseada na utilização de um sistema de visão artificial. Os sensores de visão fornecem elevada quantidade de informação face ao custo, ficando o problema do lado do processamento da informação. Pretende-se por isso neste trabalho, implementar com recurso à visão um sistema de localização que permita a navegação para "light" AGVs.

Para testar esta metodologia foi desenvolvido um prototipo para o qual foi desenvolvido um sistema de controlo de velocidade de motores baseado num controlador PID e um sistema de controlo de posição, o qual gera uma trajectória entre a localização onde o veiculo se encontra e a localização de destino.

Caracterizou-se o comportamento dos sensores que compõe o sistema, começando pela a calibração da odometria e a calibração da Camâra. Caracterizou-se e validou-se o sistema de localização implementado, através da avaliação dos erros sistemáticos da visão num percurso conhecido.

Os resultados obtidos demonstram que a solução adoptada é uma solução viável e bastante fiável, tanto para a localização do veiculo bem como para o controlo de posição deste. Foi executada uma trajectória de forma automática na qual o robô tem de passar por várias poses as quais foram previamente definidas, e verificou-se que este cumpre sistematicamente com a rota definida.

Palavras-Chave

AGV, Visão Artificial, Localização, Controlo de Posição

Abstract

The use of autonomous systems in small and medium size industrial environments is still quite restricted mainly due to their high cost/benefit relationship that stems from the high cost of these systems and average performance compared to the requirements.

This dissertation addresses the issue of the navigation of autonomous systems in structured indoor environments, and has as a goal to develop a localization solution based on the implementation of an artificial vision system. The vision sensors provide a high amount of data for their cost, leaving aside the issue of information processing. Therefore, it is intended with this project to implement a localization system, based on artificial vision, for the navigation of light AGVs.

In order to test this methodology, a prototype was created for which it was developed a speed control system for motors, based on a PID controller, as well as a position control system, which generates a path between the present location and the destination point of the vehicle.

Initially, it was analysed the behaviour of the sensors that make up the system starting with the calibration of both the camera and the odometry. Later, it was evaluated the performance of the localization system implemented through the analysis of the vision systematic errors along a predefined route.

The experimental results provide evidence that the adopted solution is viable and very reliable both for the vehicle localization, as well as, for the control of its position. The robot was sent autonomously on a track in which it had to go through several predefined checkpoints, being systematically successfully in complying with the defined route.

Keywords

AGV, Computer vision , localization, Position Control

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

AGV Autonomous Guided Vehicle

ISEP Instituto Superior de Engenharia do Porto

LSA Laboratório de Sistemas Autónomos

PC Personal Computer

CPU Central Processing Unit

BMP Bitmap

PID Proportional-Integral-Derivative (controller)

CCD Charged Coupled Device

DC Direct Current

Capítulo 1

Introdução

1.1 Âmbito da dissertação

A utilização de sistemas autónomos em ambientes industriais de pequena e média dimensão ainda se encontra bastante condicionado, essencialmente pela elevada relação custo/benefício que decorre dos elevados custos destes sistemas e dos "modestos" desempenhos face aos requisitos requeridos. É de salientar que os ambientes industriais de pequenas e médias dimensões caracterizam-se por estruturas muito flexíveis, e constantes alterações de layouts. Consequentemente, no sentido ultrapassar as dificuldades referidas é necessário prosseguir os desenvolvimentos que melhorem os desempenhos a custos controlados, Um dos pontos críticos, é o custo do sistema de navegação face ao desempenho requerido.

A navegação de veículos autónomos para ambientes industrias estruturados ainda representa um investimento elevado, o qual só pela inovação, pode ser contrariado. Nesse sentido, a navegação de sistemas autónomos é considerado um problema em aberto. Os benefícios da utilização de sistemas autónomos na indústria já se encontram comprovados, mas a necessidade de baixar a relação custo/benefício de forma a que estes sistemas atinjam mercados mais abrangentes e se tornem mais acessíveis à pequena indústria, implica a utilização de abordagens inovadoras das quais resultem fundamentalmente redução de custos em função do desempenho, consequentemente resulta na necessidade de uma optimização das arquitecturas de sistema e dos algoritmos aplicados. Esta optimização tem como objectivo aumentar a flexibilidade destes sistemas, bem como a redução dos custos associados à implementação local deste tipo de sistemas.

Os veículos autónomos que operam em ambientes "indoor" não têm acesso a sinal de GPS, por isso recorrem a outros sistemas de localização. Estes dependem de "landmarks" artificiais ou naturais presentes no ambiente, com os quais se pode obter a posição do veículo relativamente a essas "landmarks". Existem outro tipo de sistemas que acumulam erro mas

a partir destes obtém-se o deslocamento relativo do veículo em ralação a uma localização anterior. Um exemplo deste tipo de sistema são os sensores de inércia.

Esta dissertação aborda a navegação de sistemas autónomos em ambientes "indoor"estruturados, tendo como proposta desenvolver uma solução para a localização baseado na utilização de um sistema de visão artificial. Os sensores de visão fornecem elevada quantidade informação face ao custo, ficando o problema do lado do processamento da informação. Pretende-se por isso neste trabalho, implementar com recurso à visão um sistema de localização que permita a navegação para "light" AGVs. Os "light" AGVs são conhecidos pela sua pequena dimensão e baixa capacidade de carga. Vamos desenvolver uma solução recorrendo à estruturação do ambiente, nomeadamente através da colocação de "landmarks" no tecto posicionadas a uma distância pré-definida entre si, formando uma grelha, (como se pode verificar na 1.1), que funciona como referencial.

Figura 1.1: Sala de operação do robô com as marcas colocadas no tecto

1.2 Enquadramento

Esta dissertação foi realizada no ISEP-IPP (Instituto Superior de Engenharia do Porto- Instituto Politécnico do Porto), mais propriamente no LSA (Laboratório de Sistemas Autónomos), o qual, ao longo da última década tem levado a cabo projectos de desenvolvimento de veículos autónomos para diversos cenários de operação e de outros sistemas associados. Estes veículos operam em ambientes como a água, terra e ar, sendo o ROAZ II um veículo aquático de superfície, o ROV-VideoRay PRO 3E um exemplo de um subaquático, o LINCE e TIGRE veículos terrestres e o FALCOS um veículo aéreo. Uma das linhas de orientação do laboratório está direccionada na pesquisa e desenvolvimento de sistemas sensoriais que possibilitem o aumento da capacidade de percepção deste tipo de plataformas robóticas, de forma a poderem executar tarefas autonomamente.

1.3 Objectivos

O objectivo desta dissertação é contribuir para o desenvolvimento de soluções que permitam a inovação na área dos AGV e consequentemente contribuir para a existência de sistemas de baixo custo de forma a atingir novos mercados, sendo em particular relevantes os seguintes pontos:

- Caracterização do meio de operação do robô móvel, a qual consiste na semi-estruturação com "Landmarks" de forma a poder obter a posição para um robô móvel.
- Desenvolver e validar uma solução de localização de robôs móveis, para ambientes semi-estruturados, baseada em "Landmarks" reconhecidas a partir de sistema de visão, com as quais a partir de relações geométricas se obtém a posição do robô.
- Desenvolvimento do módulo de controlo de motores e de posicionamento do robô, com o qual permita a deslocação do robô entre dois pontos.
- Desenvolver e validar uma arquitectura de sistema de forma a integrar a localização e controlo de posição de forma a efectuar rotas entre diversas localizações pré-configuradas.

1.4 Estrutura do documento

Para além deste capítulo esta dissertação contem mais seis capítulos:

- No capítulo dois encontram-se descritas algumas metodologias utilizadas na indústria para a localização de robôs moveis.
- No capítulo três encontra-se descrita a proposta de localização que se pretende validar. É descrita a solução em detalhe do sistema de localização. Começando pela odometria e modelo cinemático do robô, como é realizado o processamento de imagem e o respectivo algoritmo de localização
- No capítulo quatro descreve-se a teoria do sistema de controlo de velocidade que foi implementado para os motores, bem como o controlo de posição do robô de forma a este se deslocar de um ponto A para um ponto B.
- No capítulo cinco encontra-se a descrição do software do robô desenvolvido.
- No capítulo seis estão descritos os resultados obtidos com o sistema implementado.
- Por fim são apresentadas algumas conclusões e comentários finais como as respectivas linhas de trabalho futuro, tendo em vista a continuidade do presente trabalho.

Capítulo 2

Estado da arte

2.1 Introdução

Há vários tipos de sistemas robóticos de aplicação industrial, sendo os braços robóticos os que mais se destacam. Contudo, devido à falta de mobilidade deste tipo de sistemas as suas aplicações são limitadas a um determinado número de operações.

Os robôs moveis por sua vez, em vez de estarem restritos a um espaço de trabalho fixo, são capazes de deslocar e cumprir as suas missões numa vasta área, sem intervenção humana. Desta forma é possível aplicar a robótica em diversas tarefas que vão desde a limpeza e vigilância até a assistência a deficientes, passando por transporte de cargas e limpeza de piscinas. A implementação deste tipo de robôs tem sido alvo de diversas abordagens, no entanto a sua aplicação é ainda muito limitada devido ao elevado custo do robô em si.

Para os robôs se movimentarem autonomamente é necessário que a sua localização e navegação seja calculada com elevada exactidão. O robô tem de recorrer aos seus sensores para extrair a informação necessária do meio envolvente para determinar a sua posição e decidir qual a trajectória a tomar.

A luz da tecnologia actual há uma variedade de soluções para dar resposta as três perguntas fundamentais da navegação dos sistemas moveis, sendo estas:

“ Where am I? “ , “ Where am I going? ” e How should I get there? “ [15].

A localização procura responder a pergunta ”onde estou?”, esta consiste em determinar a pose do veiculo em relação a um referencial externo, por outras palavras a posição cartesiana e a orientação do robô em relação a um referencial no mundo pré-estabelecido. É a localização que permite com que as restantes perguntas “como la chegar?” e “para onde vou? “ possam ser tecnicamente implementadas e por isso as tarefas de planeamento e a execução

de trajectórias tornam-se impossíveis sem recorrer a uma localização precisa, robusta e em tempo real.

De modo a que a utilização de robôs moveis se torne ainda mais acessível para a industria, e para as pequenas empresas, torna-se necessário ultrapassar as limitações e os custos associados aos sistemas actuais.

Tem-se verificado nos últimos anos uma crescente utilização de robôs moveis tanto na industria bem como ao nível dos serviços e por isso, pode-se encontrar no mercado um leque alargado de soluções tecnológicas que variam conforme o tipo de aplicação do robô móvel. A variação destas tecnologias depende essencialmente do tipo de navegação que se pretende obter com o robô móvel.

Para que a navegação seja possível é essencial a implementação de um sistema de localização que cumpra os requisitos necessários a tarefa a desempenhar. Sendo que esta área de investigação não é recente e ao longo dos anos, foram propostas variadíssimas técnicas de localização, cada uma das soluções acaba por ter os seus pontos fortes e suas debilidades. Esta diversidade surgiu devido ao grande leque de aplicações dadas aos robôs moveis bem como o custo/beneficio que estas tecnologias tem. Para cada uma destas aplicações, os requisitos de localização e os seus custos fazem com que tenham resultado numa variedade de tecnologias para resolver essencialmente o mesmo problema.

Por estas razões a solução técnica mais apropriada para uma dada aplicação tem de ter em conta os seguintes requisitos: a exactidão requerida, as condições do local onde vai operar o tipo de trajectória que a navegação permite executar, os custos e a necessidade ou não de flexibilidade.

De forma a restringir o leque de informação a citar e uma vez que é uma área na qual já há uma grande variedade de soluções será essencialmente focadas as soluções de localização em ambientes industriais que são mais utilizadas.

2.2 Soluções de localização de robôs moveis Industrias

2.2.1 Solução baseada em filoguiado

Nesta solução o AGV navega em relação a um fio. Isto é conseguido através da colocação de um sensor por debaixo do veiculo para detectar um sinal de radio frequência proveniente de um fio eléctrico que foi previamente embutido no chão (figura 2.1) a uma distancia entre 2 a 3 centímetros da superfície do solo [3].

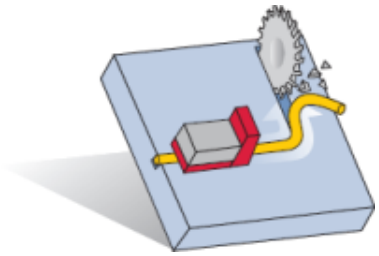


Figura 2.1: Metodo de localização baseada em filoguiado [14]

O fio condutor é atravessado por corrente alternada, que cria em torno deste um campo electromagnético. Este campo induz uma tensão aos terminais de uma bobina que se encontre na proximidade. A tensão gerada na bobina é proporcional a sua intensidade. O sensor é estrategicamente colocado por debaixo do veiculo e composto por duas bobinas cada uma colocada de cada lado do fio como se pode ver na figura 2.2.

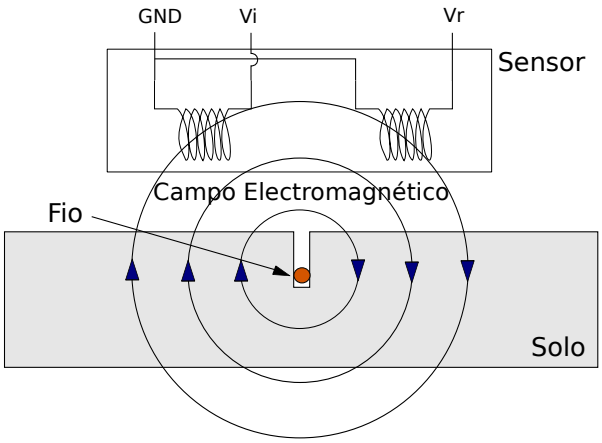


Figura 2.2: Principio de funcionamento do sistema filoguiado

Com este par de bobinas é possível obter um sinal de controlo. Este sinal é uma diferença de tensão aos terminais de cada bobina [21]. Quando o sensor está posicionado sobre o condutor a diferença de tensão entre as bobinas é nula. Assim que o veiculo se desvia da rota, a tensão aumenta numa das bobinas e diminui na outra gerando um diferença de potencial que serve de indicação para que direcção o veiculo se deve ajustar, mantendo-se desta forma dentro do caminho predefinido.

Esta solução é a técnica mais difundida na industria e é baseada em filoguiado. É a forma mas simples de navegação, esta permite executar trajectórias fixas, e foi a técnica de localização usada no primeiro AGV, construido em 1950 por Barrett Electonics .

Tendo em conta as características técnicas deste tipo de localização, verifica-se que é

uma solução com grande exactidão, uma vez que o robô segue sempre o caminho dito certo, tem no entanto a desvantagem de as deslocações se realizarem a uma velocidade reduzida, de forma a evitar que o robô deixe de detectar o fio e desta forma se perda.

A grande desvantagem deste sistema é a falta de flexibilidade, caso seja necessário mudar as trajectórias. Porque a recolocação do fio no solo implica custos de instalação, assim como a eventual paragem da produção.

Da mesma forma a preparação do ambiente onde ira operar é cara e complexa.

Um exemplo de um sistema de navegação deste tipo foi desenvolvido pela empresa Mantenimiento BAMA. Este foi concebido para o transporte de paletes o qual se encontra na figura 2.3 [9] .

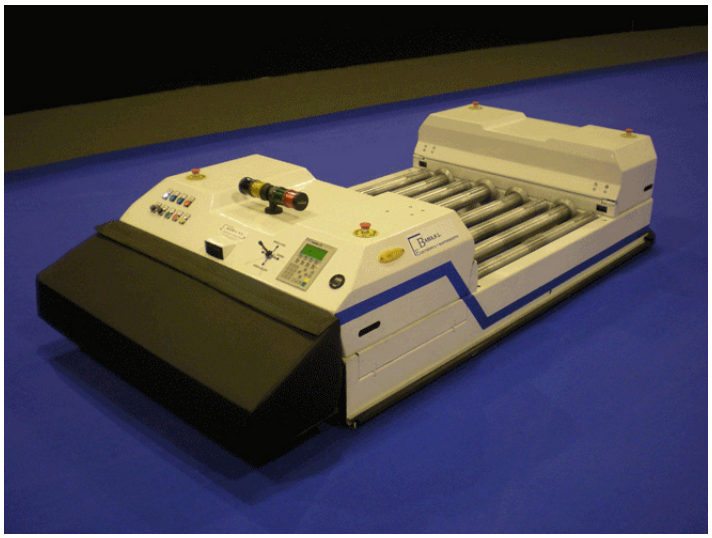


Figura 2.3: AGV produzido pela empresa Mantenimiento BAMA [9]

2.2.2 Solução baseada em faixas

Esta solução é baseada em faixas colocadas no piso, estas faixas definem as trajectórias possíveis que são executadas pelo robô. É por isso também uma solução de trajectórias fixas. Os percursos são definidos por fitas magnéticas ou coloridas colocadas no piso como se encontra representado na figura 2.4.

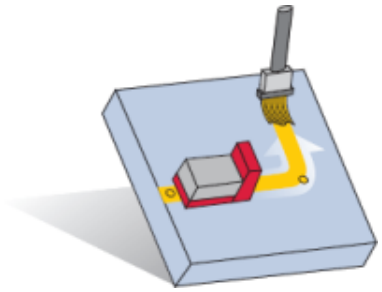


Figura 2.4: Método de localização baseada em faixas [14]

O principio de funcionamento deste tipo de robôs assenta na detecção das fitas usando os sensores apropriados ora sejam fitas coloridas ora fitas magnéticas. Este tipo de robôs funciona de forma muito idêntica à solução anterior.

A vantagem deste sistema em relação ao filoguiado é que não é necessário uma estruturação do ambiente tão complexa e demorada como a anterior. Basta apenas proceder a recolocação das faixas para alterar as trajectórias [7], o que pode ser efectuado em algumas horas.

No entanto tem a desvantagem de a localização do robô ficar dependente da perfeita identificação das faixas, pelo que em ambientes movimentados e nos quais as faixas possam ficar danificadas ou sujas a sua utilização é pouco recomendável.

Na figura 2.5 encontra-se um AGV, denominado move produzido pela empresa *trilogiq* e que utiliza o método de localização apresentado nesta secção [28].

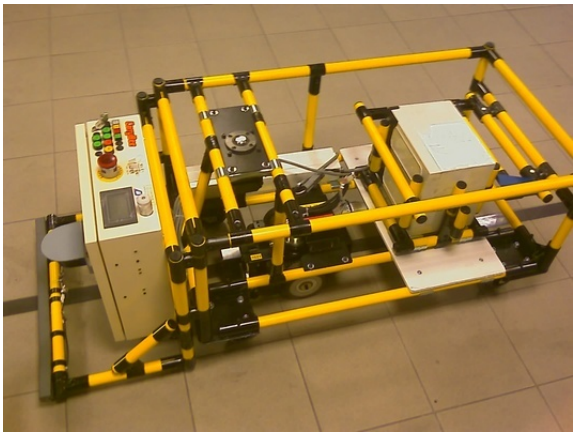


Figura 2.5: AGV move da empresa trilogiq [18]

2.2.3 Solução baseada em marcadores

Esta solução é baseada em marcadores colocados no ambiente de operação do robô. Por norma estes são colocados no pavimento e podem ser, marcadores magnéticos, marcadores visuais entre outros. Esta metodologia de navegação, permite a realização de trajectórias

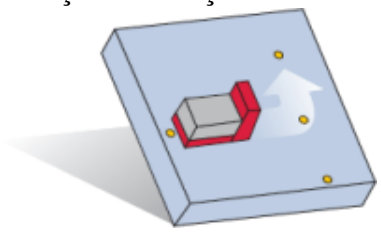
dinâmicas. Neste tipo de sistemas a informação sobre a posição absoluta nem sempre se encontra disponível. Ou seja, quando o robô detecta um marcador corrige a posição absoluta e na falta destes recorre a sistemas complementares como odometria e/ou sensores inerciais para estimar as posições intermédias.

No caso de os marcadores se encontrarem espaçados de forma inadequada o erro acumulado pelos sensores complementares, vai ultrapassar os valores admissíveis e a trajectória do robô fica comprometida. Esta situação pode levar o robô a desviar-se consideravelmente e pode mesmo falhar a detecção do próximo marcador que resulta inevitavelmente na perda da posição.

Por isso recorre-se a fusão da informação da localização absoluta e relativa tendo em conta o erro de cada sistema.

Para a implementação de alguns destes tipo de sistemas são embutidos marcadores, habitualmente ímanes ou tags RF passivas [3] no pavimento, os quais são colocados fisicamente em coordenadas conhecidas e servem como pontos de referencia para o robô corrigir os erros acumulados na percurso entre marcadores obtendo assim a posição absoluta do robô. Na figura 2.6 encontra-se uma representação de como estes marcadores são colocados.

Figura 2.6: Representação da solução baseada em marcadores [14]



Comparativamente aos sistemas filoguiados, esta solução permite uma instalação mais rápida e reconfiguração significativamente mais rápida das trajectórias, no entanto o custo do equipamento é significativamente mais elevado e normalmente a exactidão é inferior. Esta solução é também menos exactidão que os sistemas laser, mas tem um custo mais baixo que estes. O custo da colocação dos ímanes no pavimento pode mesmo ser inferior ao da colocação dos reflectores para os lasers [1].

Na figura 2.7 encontra-se um AGV produzido pela empresa Frog concebido para a industria, para tarefas como transporte de paletes, em que o método de localização depende de pequenos ímans colocados estrategicamente no solo formando uma matriz bidimensional.

Figura 2.7: AGV de transporte de paletes produzido pela Frog [6]

Na figura 2.8 apresenta-se um AGV, produzido pela Kiva concebido para o transporte de prateleiras. Neste caso em particular, os marcadores, são códigos de barras, dispostos em intervalos regulares , criando uma grelha bidimensional.

Figura 2.8: AGV de transporte de prateleiras produzido pela Kiva [8]

2.2.4 Solução baseada em Trilateração e Triangulação com Balizas

Outra solução para trajectórias flexíveis utilizada na industria, são sistemas baseados em triangulação ou trilateração por reflectores laser. Os painéis reflectores de laser são instalados nas paredes ou colunas a alguns metros do solo, de modo a evitar a obstrução entre estes e o laser scanner instalado no robô (figura 2.9). O laser realiza constantemente um varrimento rotativo [23]. Este emite feixes laser e através da reflexão destes mede a distancia e a orientação da localização dos reflectores. Como é sabida a posição absoluta dos reflectores passa a ser possível calcular a posição do robô recorrendo a técnicas de triangulação.

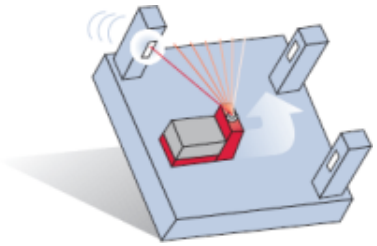


Figura 2.9: Representação da solução baseada em Trilateração e Triangulação com Balizas [14]

Este tipo de sistema tem a vantagem de ser muito exacto ter rotas flexíveis e poder operar a velocidades de movimentação mais altas comparativamente as outras soluções anteriormente apresentadas. Outro factor importante a ter em conta é que se não houver obstrução dos reflectores com este sistema em qualquer momento têm acesso a posição absoluta ao contrario dos sistemas com marcas em que era necessário, em certos momentos, estimar a sua posição, com recurso a sistemas auxiliares, e por isso a precisão da localização diminuía. Tendo em conta esse factor este método tem uma precisão superior ao sistema de marcas.

É no entanto uma solução com um custo inicial relativamente alto e que obriga a colocação de vários painéis reflectores ao longo das trajectórias. Para que se possa obter a posição absoluta do robô é necessário que em cada instante de tempo sejam visíveis ao robô pelo menos três reflectores [2] sendo aconselhável cinco ou mais. Tem por isso o inconveniente da preparação do espaço o que implica um bom planeamento da disposição dos reflectores e alem disso só pode ser utilizado em espaços sem obstruções.

Na figura 2.10 encontra-se um AGV produzido pela empresa elettric80 concebido para a industria, para o transporte de paletes.

Figura 2.10: AGV modelo *CB 08* da elettric80 com localização por laser [5]

2.2.5 Solução baseada em visão Artificial

Encontra-se em desenvolvimento pelo grupo de investigação ROBIS - Robótica e Sistemas Inteligentes do INESC Porto em colaboração com a empresa DELTAMATIC - Engenharia e Automação Industrial, uma solução baseada em visão artificial, O funcionamento desta baseia-se na colocação de marcas são colocadas no chão [16] as quais são reconhecidas pelo sistema de visão e com as quais se procede a correcção dos dados recolhidos pela odometria. Estas marcas tem o formato de triângulos e a câmara tem a sua orientação, virada para o chão. O AGV econtra-se na figura 2.11 sendo a caixa a negro onde se encontra colocada a câmara para a detecção das marcas.

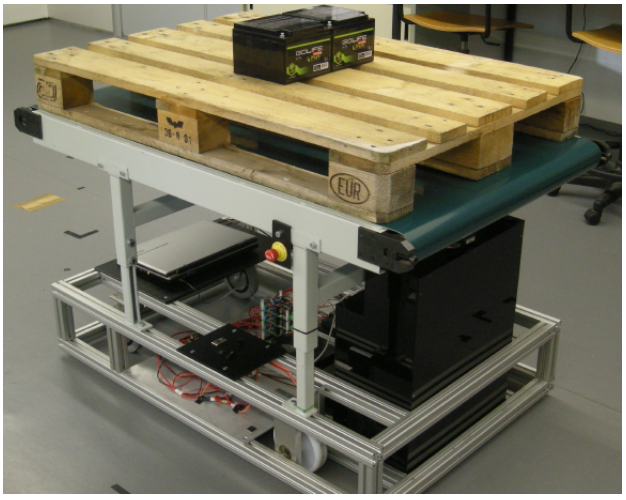


Figura 2.11: AGV desenvolvimento pelo grupo de investigação ROBIS [16]

Na figura 2.12 pode-se verificar que a área de detecção das marcas é limitada, tendo que o proprio AGV ter de passar por cima das marcas de forma a fazer a correcção adequada da localização absoluta, tendo que por isso na inexistência destas marcas confiar apenas na posição relativa obtida pela odometria.

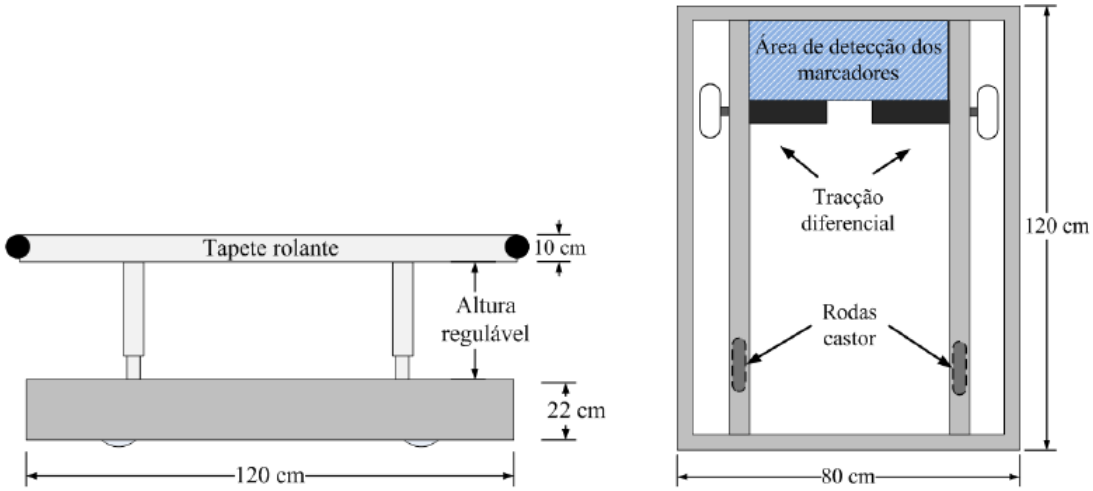


Figura 2.12: Dimensões e estrutura básica do robô móvel [16]

Este robô tem o conhecimento prévio de onde se encontram localizadas as marcas e sempre que uma destas é detectada pelo sistema de visão é corrigida a localização absoluta do robô. Existindo por isso a necessidade de serem colocadas nas zonas por onde o robô passa, de forma a que o erro acumulado pelo sistema da odometria não ultrapasse os valores aceitáveis de forma a que o robô não perda a localização.

Na figura 2.13 encontra-se uma frame onde se pode visualizar a marca que é detectada pelo sistema de visão artificial deste robô.

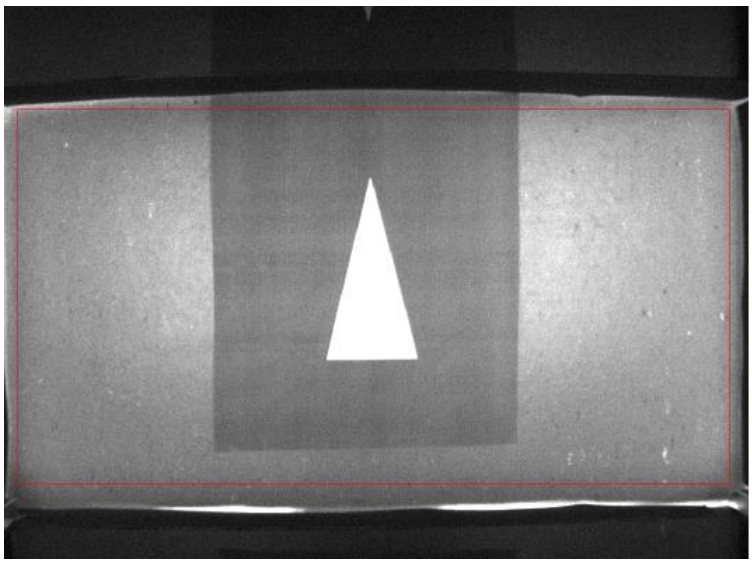


Figura 2.13: Frame onde se visualiza a marca triangular que este AGV utiliza para corrigir a posição [16]

A vantagem deste sistema é a de poder efectuar uma navegação dinâmica podendo realizar vários tipos de trajectórias, e terá uma boa localização no caso de a colocação das

marcas seja efectuada de forma a que o erro acumulado na odometria não se estenda por grandes distancias.

As desvantagens deste sistema é que pode não recuperar a localização correcta no no caso de numa determinada rota a colocação das marcas no chão não seja a adequada para serem detectadas, porque há a necessidade de que o robô passe por cima de uma marca para corrigir a sua pose e no caso de a colocação das marcas não estar ajustada a rota (ou as alterações das rotas) o robô pode percorrer grandes distancias sem detectar uma marca de correcção. Outra das desvantagens é a necessidade de as marcas terem de ser substituídas quando apresentam desgaste, uma vez que estas são colocadas no chão. Esta metodologia não é imune ao deslizamento das rodas porque quando tal ocorre a alteração da posse estimada do robô pode não levar ao encontro de uma marca de correcção. No essencial este sistema de navegação está portanto limitado ao numero de marcas que são colocadas no chão as quais a sua localização tem de ser adicionada ao sistema, exigindo que no caso de alterações significativas nas rotas estas tenham de ser acrescentadas tal como a sua localização relativamente ao sistema.

2.2.6 Observações

Todos os sistemas apresentados apresentam características de navegação bastante diferenciadas.

Os sistemas baseados em filiguiado e em fitas tem restrições relativamente ao tipo de trajectórias que permitem efectuar uma vez que a sua navegação é efectuada em torno de um referencial estático (fio ou faixas), no entanto podem ser considerados sistemas com boa exactidão.

No caso dos baseados em marcas, tem mais flexibilidade no seu tipo de navegação, contudo a sua exactidão não é constante e o seu valor mais alto ocorre quando o robô esta mais perto das marcas.

O sistema com navegação flexível e com melhor exactidão na sua localização, são os baseados em trilateração a laser, estes podem em qualquer local obter a sua localização absoluta, sendo no entanto necessário que entre os reflectores e o robô não exista qualquer obstrução.

Pode-se considerar que todas as soluções apresentadas tem um custo elevado, embora por diferentes motivos.

No caso dos sistemas baseados em fitas e filiguiado podem até ter um custo inicial mais baixo em relação as outras soluções, mas tem custos ”escondidos“, os da estruturação do ambiente e quando existe a necessidade de mudança nas trajectórias. Estas mudanças podem ocorrer porque em ambiente industrial a disposição física dos sistemas de produção pode ser

alterado devido a introdução de novas metodologias de produção.

No caso dos sistemas baseados em marcas tem desempenho limitado uma vez que a localização só é exacta por curtos espaços de tempo o que faz com que a navegação seja pouco exacta. Dito de outra forma trajectórias efectuadas tem a vantagem de ser mais dinâmicas relativamente aos anteriores mas estas tem menor exactidão em relação ao posicionamento pretendido.

No caso dos sistemas baseados em trilateração laser, o custo está associado ao tipo de sensor usado. Estes são comparativamente mais caros do que qualquer das outras soluções.

O sistema de navegação ideal seria portanto, um que tive-se uma exactidão elevada, baixo custo de estruturação, permiti-se navegação flexível e sensores de baixo custo.

Capítulo 3

Sistema de Localização

3.1 Solução de localização proposta

Este trabalho aqui apresentado explora a utilização de um sistema de visão, na percepção do ambiente externo ao robô. Em particular vamos focar o nosso trabalho no desenvolvimento de um sub-sistema de navegação recorrendo essencialmente a visão artificial.

O conceito surgiu do estudo da metodologia utilizada pelos AGVs da frog, os quais são colocados ímanes no pavimento para formar uma grelha de referencia no pavimento. Este tipo de marcadores são detectados quando o AGV passa por cima destes e não continuamente. Por isso surgiu a ideia de alterar estes tipo de marcadores por landmarks colocadoras no tecto, de forma a que com visão artificial fosse possível a detecção em continuidade destes marcadores. Estas landmarkes são passivas por isso tem um baixo custo e em termos de logística são mais fáceis de colocar.

A proposta de localização do robô apresentada nesta dissertação assenta na estruturação do ambiente com a colocação de "landmarks" passivas no tecto. Estas tem o formato apresentado na figura 3.1, onde se pode identificar um quadrado. A dimensão do lado do quadrado é de 10 centímetros. Estas "Landmarks" estão distribuídas em forma de grelha (figura 3.2). A disposição destas marcas forma uma grelha e têm uma posição relativa entre si igual. Esta característica, surge de o sistema necessitar de conhecer qual a posição relativa destas no ambiente. E poder fixar um sistema de coordenadas que será o sistema de referencia sobre o qual o robô opera.

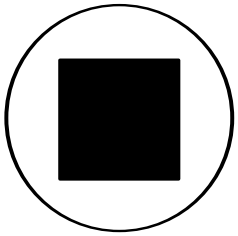


Figura 3.1: Aspecto da ”Landmarks“ criada

Recorrendo ao sistema de visão artificial pretende-se retirar do meio envolvente ao robô a informação necessária de forma a obter a posição do robô, que no caso é a localização destas ”landmarks” relativamente ao robô. Para tal, recorre-se ao modelo *pinhole camera* que descreve a relação matemática entre as coordenadas de um ponto nas três dimensões e a sua respectiva projecção no plano de imagem, e com isto obter a posição relativa de objectos que se encontrem no seu meio envolvente. Para que a posição relativa dos objectos seja correctamente calculada é necessário dotar o sistema da capacidade de identificar os objectos correctos no meio ambiente, de forma a ser possível actualizar a posição do robô.

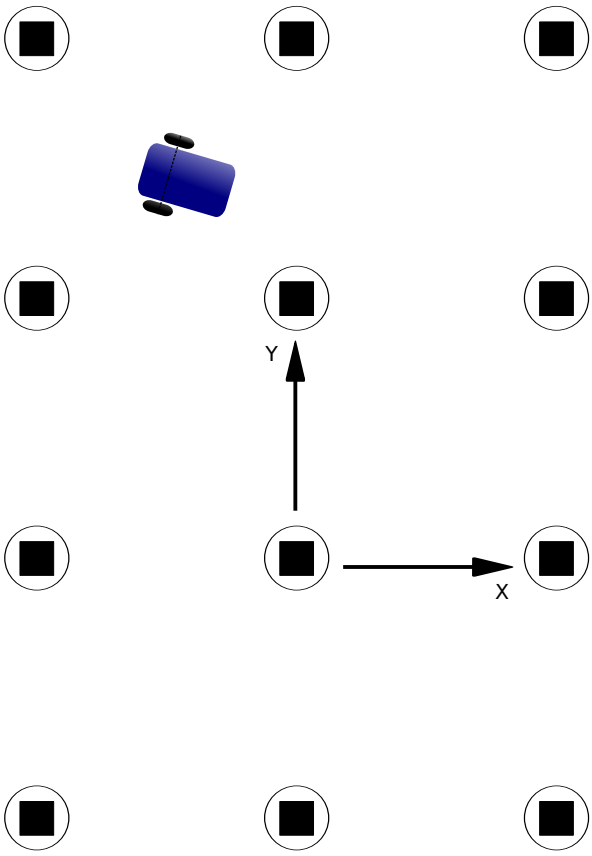


Figura 3.2: Representação da estruturação do ambiente com demonstração da colocação das marcas no tecto da sala de operação do robô

Na figura 3.2 encontra-se representado o sistema de coordenadas de operação do robô (conhecido como referencial do mundo). Este sistema de coordenadas por questões de repetibilidade necessita de ser fixo e que a cada inicialização do sistema se encontre sempre no mesmo lugar,isto é, na mesma localização física e com a mesma orientação.

Para garantir este requisito, forma implementadas condições de inicialização. O robô tem que se encontrar posicionado de forma a que a marca que define o sistema de coordenadas se encontre projectada no plano de imagem e a orientação da câmara relativamente a este tem de se encontrar dentro de um intervalo de valores. Com estas condições é possível prever como é fixado o sistema de coordenadas de referencia do robô.

O tipo de robô proposto nesta dissertação é um robô conhecido por ter tracção diferencial. O que caracteriza este tipo de robôs é que tem duas rodas fixas no mesmo eixo com tracção independente. Este modelo de robô tem uma construção simples e permite que este rode sobre o próprio eixo. Com esta configuração é possível a obter através de odometria um sistema de localização relativo, que complementa o sistema de localização proporcionado pelo sistema de visão.

A odometria permite ao sistema prever onde se localizam as marcas no plano de imagem. Porque nos dá uma estimativa da localização do robô. Esta é conseguida com base na localização anterior, mais a deslocação relativa entre essa e a que o robô tem no momento da aquisição de uma nova frame. Com este sistema complementar de localização relativa pode-se prever a posição das landmarks no plano de imagem e assim poder-se reduzir a área de pesquisa na imagem e consequentemente reduzir o processamento necessário. Torna também a localização mais robusto, porque no caso de a câmara se encontrar temporariamente obstruída ou por saturação do CCD pode na mesma prever qual a sua localização, e quando o sistema de visão recuperar o normal funcionamento encontrar onde estas marcas se devem encontrar na imagem. O esquema geral de como os dois tipos de localização interagem encontra-se na figura 3.1.

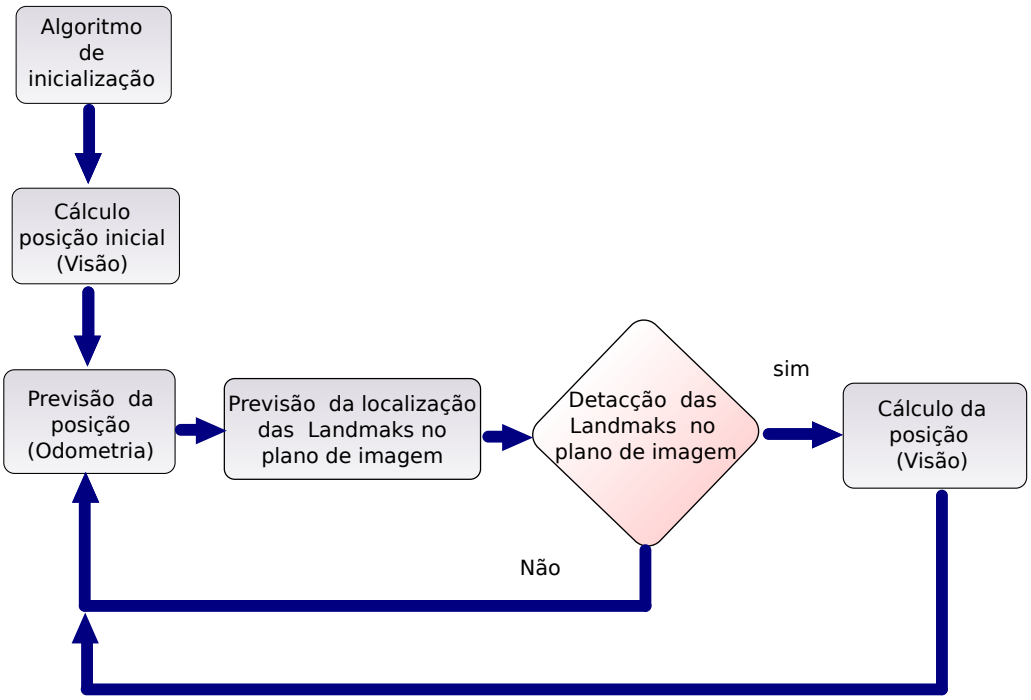


Figura 3.3: Diagrama da localização

A mobilidade do robô fica a cargo de um sistema de controlo de posição o qual é constituído por duas partes. Uma delas é um controlador de velocidade baseado no modelo do controlador PID, o qual permite o controlo de velocidade dos motores do robô. A segunda parte é um algoritmo que gera uma trajectória entre a localização onde o veiculo se encontra e uma localização de destino. Este algoritmo é também um controlador em malha fechada, em que a entrada do sistema é a pose do robô e a saída são as velocidades a aplicar ao robô para que este se desloque para uma nova localização.

3.2 Odometria

A odometria é um dos métodos mais utilizados para estimar a pose momentânea de um robô e é um método de localização relativa.

A partir de uma posição inicial conhecida a pose é obtida pela integração dos movimentos lineares de cada uma das rodas. Para integrar estes movimentos recorre-se ao modelo cinemático do robô o qual permite efectuar a partir desses dados uma estimativa da localização do robô. A vantagem da odometria está associada ao baixo custo de implementação, à elevada taxa de amostragem permitida e a uma boa precisão para curtas deslocações.

A desvantagem prende-se com a acumulação de erros, uma vez que cada medição é afectada de um erro o qual é também integrado, este facto faz com que a estimativa da pose do robô fornecida por este método, se afaste dos seus valores reais quanto maior for a deslocação. O efeito pratico é que, se um dado robô estiver limitado a uma determinada área de operação e percorrer uma grande distancia e a posição absoluta não for actualizada, o erro da estimação da pose pode ultrapassar o próprio tamanho da área de operação. Dito isto torna-se evidente que a estimação da pose não tem qualquer validade.

Os erros que afectam a odometria podem ser tanto de natureza sistemática como aleatória. No entanto a odometria proporciona uma boa precisão para curtas distancias e é utilizado na pratica conjuntamente com sistemas que periodicamente corrigem a pose medida pela odometria.

3.2.1 Modelo cinemático do robô

O modelo de robô implementado é conhecido como Robô de tracção diferencial. O que caracteriza este tipo de robôs é que tem duas rodas fixas com tracção independente, cujos veios passam pelo mesmo eixo. Este modelo necessita de um ou mais apoios extra os quais não interferem no movimento do robô. No caso implementado têm mais quatro apoios, no entanto só estão em contacto com o chão dois de cada vez, isto porque estando os quatro apoiados pode facilmente ocorrer deslizamento das rodas de tracção o que vai afectar a odometria. No entanto este modelo permite com que a sua construção seja bastante simples.

Um factor negativo a ter em conta neste modelo é de ser bastante sensível a velocidade relativa das rodas de tracção, quero com isto dizer que pequenos erros na velocidade de cada uma destas geram diferenças significativas na trajectória e por isso um bom controlo da velocidade tem de visto como um requisito essencial de forma a poder obter bons resultados nas trajectórias.

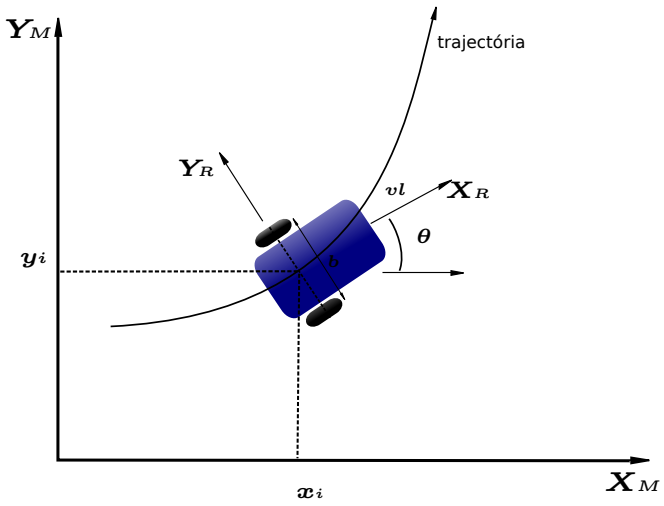


Figura 3.4: Robô diferencial

A pose de um veiculo de tracção diferencial é definida por (x,y,θ) onde x e y representam a posição no espaço do ponto médio entre as duas rodas de tracção do veiculo sendo também o ponto sobre o qual o veiculo roda e essa rotação é representada por (θ) que define a orientação do veiculo

O modelo cinemático do robô diferencial pode ser representado pelo seguinte sistema de equações [25] :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v_l(t) \times \cos(\theta(t)) \\ v_l(t) \times \sin(\theta(t)) \\ \omega \end{bmatrix} \tag{3.1}$$

As variáveis de entrada deste sistema são a velocidade linear (v_l) e a velocidade angular (ω) do veiculo que podem ser obtidas a partir da velocidade linear de cada roda de tracção.
A velocidade linear é definida por:

$$v_l(t) = \frac{v_e(t) + v_d(t)}{2} \tag{3.2}$$

Onde v_e é a velocidade linear da roda esquerda e v_d a velocidade linear da roda direita.

A velocidade angular é definida por:

$$\omega(t) = \frac{v_d(t) - v_e(t)}{b} \tag{3.3}$$

Onde (b) representa a distancia entre os pontos de contacto das rodas de tracção com o chão, Para controlar a posição do robô tem de se calcular a velocidade de rotação de cada motor a partir das velocidades lineares anteriormente descritas onde (r) representa o raio da respectiva roda.

$$\omega_d = \frac{v_d(t)}{r_d} \tag{3.4}$$

$$\omega_e = \frac{v_e(t)}{r_e} \tag{3.5}$$

É por isso fundamental ter um bom dimensionamento dos valores de (b) (r_e) (r_d).
Por questões de controlo dos motores e sendo uma "norma" que este se realize nas unidades de rotações por minuto efectua-se a seguinte conversão:

$$r.p.m = 60 \times \frac{\omega}{2 \times \pi} \tag{3.6}$$

3.2.2 Modelo da Odometria

Para obtermos o modelo da odometria é necessário proceder a discretização de todas as equações que definem o modelo o cinemático do robô [16].

Para obter a pose é necessário que a descretização seja efectuada por diferenças centradas no modelo cinemático do robô, resultando no seguinte sistema de equações 3.7.

$$\begin{bmatrix} x(i+1) \\ y(i+1) \\ \theta(i+1) \end{bmatrix} = \begin{bmatrix} x(i) + \Delta d(i) \times \cos(\theta(i) + \Delta\theta(i)) \\ y(i) + \Delta d(i) \times \sin(\theta(i) + \Delta\theta(i)) \\ \theta(i) + \Delta\theta(i) \end{bmatrix} \tag{3.7}$$

A variação da distancia linear ($\Delta d(i)$) durante um período de amostragem é dada por:

$$\Delta d(i) = \frac{\Delta d_d(i) + \Delta d_e(i)}{2} \tag{3.8}$$

A variação da orientação $\Delta\theta(i)$ durante um período de amostragem é dada por:

$$\Delta\theta(i) = \frac{\Delta d_d(i) - \Delta d_e(i)}{b} \tag{3.9}$$

Onde $\Delta d_d(i)$ é o deslocamento linear da roda direita e $\Delta d_e(i)$ é o deslocamento linear da roda esquerda. Estes deslocamentos são estimados a partir do número de impulsos, $N_{tiks}(i)$, gerados pelo encoder correspondente, isto é:

Para a roda direita:

$$\Delta d_d(i) = dist_{d_{tick}} * N_{d_{tiks}}(i) \tag{3.10}$$

Para a roda esquerda:

$$\Delta d_e(i) = dist_{e_{tick}} * N_{e_{tiks}}(i) \tag{3.11}$$

O valor de $dist_{tick}$ é a distancia linear que corresponde um impulso. Para obter essa distancia é necessário conhecer o diâmetro (D) de cada roda, a relação da caixa (N) e resolução $RE_{encoder}$ dos encoderes respectivos . O valor é obtido por:

$$dist_{tick} = \frac{\pi \times D}{N \times RE_{encoder}} \tag{3.12}$$

Os parâmetros necessários para definir o modelo de odometria são: O factor de conversão entre os impulsos dos encoders e o deslocamento ($dist_{tick}$) da respectiva roda e a distancia entre cada ponto de contacto das rodas com o chão (b) .

Para calcular a distancia por tick ($dist_{tick}$) sabemos a relação da caixa e a resolução dos encoderes, de maneira a medir o diâmetro das rodas começa-se por considerar que ambas tem o mesmo diâmetro. Percorre-se uma distancia em linha recta a baixa velocidade e durante esse percurso é guardada a quantidade total de impulsos medidos pelos encoders. Com estes dados e recorrendo as equações, 3.8, 3.10 e 3.12 obtêm-se o diâmetro das duas rodas.

3.2.3 Erros sistemáticos e calibração da Odometria

Os erros sistemáticos na odometria, podem ter as seguintes origens. [24]:

- As rodas terem diâmetros diferentes uma da outra.
- Erro na mediada do diametro médio das rodas.
- Erro na mediada da distancia entre rodas (b) .
- Rodas mal alinhadas.

Por norma este tipo de erro não se altera significativamente ao longo do tempo, mas alterações da carga que o robô transporta, podem alterar a deformação dos penus e alterar o diâmetro das rodas bem como o ponto de contacto das rodas com o chão, alterando portanto também a distancia entre rodas.

Tratando-se de erros sistemáticos é possível diminuir o seu efeito recorrendo a uma calibração. Quanto melhor calibrado tiver o sistema de odometria menor poderá ser a frequência de actualização da pose através do sistema de localização absoluta.

Para identificar e compensar os erros sistemáticos recorreu-se a técnica de calibração designada por UMBmark [10]

3.2.4 Calibração UMBmark

K. S. Chong e L. Kleeman (1997), apresentam um método sistemático de calibração do sistema de odometria de um robô de tracção diferencial. Este método permite compensar o efeito dos erros sistemáticos.

Nesta metodologia são identificados três fontes dominantes que provocam erros sistemáticos:

- O erro devido a diâmetros das rodas diferentes, que é definido por:

$$E_d = \frac{D_d}{D_e} \tag{3.13}$$

  Na robótica móvel normalmente as rodas utilizadas tem pneus de borracha de forma a melhorar a tracção. Consequentemente ocorrem pequenos desvios do valor nominal do diâmetro das rodas, os quais podem ter origem na construção do pneu e na distribuição não uniforme da carga do robô.

- O erro devido a incerteza da distancia entre os pontos de contacto das rodas com o chão, que é definido por:

$$E_b = \frac{b_{real}}{b_{actual}} \tag{3.14}$$

  Consiste na razão entre o valor real da distancia entre os pontos de contacto das rodas com o chão (b) e o valor actual usado no sistema de odometria. Esta incerteza deve-se ao facto de a superfície de contacto das rodas com o chão não ser pontual. Quanto menor a largura dos pneus menor é a incerteza.

- O erro de escala, que é definido por:

$$E_s = \frac{D_{medioreal}}{D_{medioatual}} \tag{3.15}$$

  O erro de escala é a razão entre o diâmetro médio real das duas rodas e o utilizado na odometria. O D_{medio} é fácil de obter e a forma de ser obtido encontra-se descrita na secção anterior. Considera-se que este erro já se encontra por isso correctamente compensado, e que por isso o seu efeito já não é significativo.

Com o metódo de calibração UMBmark é possível identificar separadamente o erro provocado por E_s e E_b e , desta forma compensar correctamente o efeito que cada um provoca

na odometria.

A identificação é conseguida medido a diferença entre a pose real e a pose medida pelo sistema de odometria, após o robô percorrer uma trajectória em forma de quadrado no sentido dos ponteiros do relógio (PR) e repetir o mesmo tipo no sentido contrario (CPR).

Na figura 3.5 está representado o efeito do erro E_d provocado pela diferença de diâmetro das rodas durante uma trajectória quadrangular, na qual se pode identificar uma ligeira curvatura ao longo dos seus quatro lados e que E_d apenas afecta o movimento de translação do robô. O robô acumula erro de orientação em cada lado da trajectória.

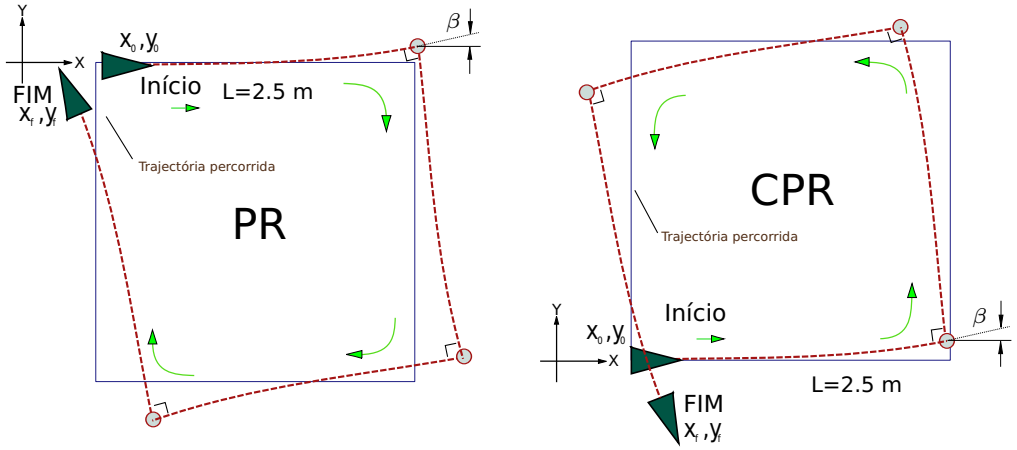


Figura 3.5: Efeito do erro E_d no sentido PR (esquerda), e no sentido CPR (direita).

Na figura 3.5 está representado o efeito do erro E_b provocado pelo erro na distancia entre os pontos de contacto das rodas com o chão. Observa-se que este erro afecta apenas os movimentos de rotação do robô o que se traduz num erro da medição da quantidade de rotação do robô (α) em cada canto do quadrado.

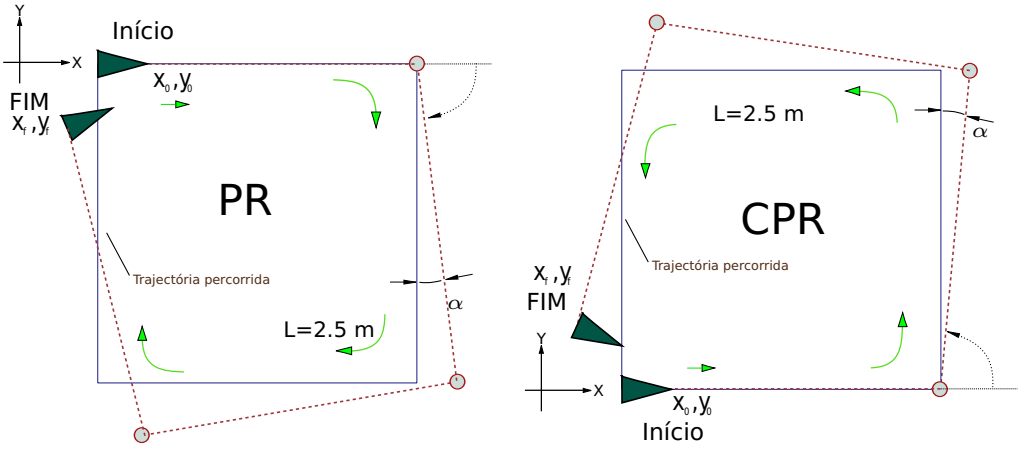


Figura 3.6: Efeito do erro E_b no sentido PR (esquerda), e no sentido CPR (direita).

Conclui-se observando a figura 3.5 que E_d reduz (ou aumenta) a quantidade de rotação efectuada durante o percurso quadrangular no sentido contrario aos ponteiros do relógio, mas aumenta (ou reduz) a quantidade total de rotação no sentido contrario dos ponteiros do relógio. Contrariamente observa-se na figura 3.6 se E_b aumenta (ou reduz) a quantidade total de rotação efectuada tanto no sentido PR como CPR. Dai, este método permite isolar os efeitos de E_d e E_b pois se na trajectória quadrangular, quando realizada num determinado sentido, estes efeitos compensam-se mutuamente (dando a ideia errada da qualidade da odometria), o mesmo não se verifica no sentido inverso, neste caso os erros somam-se de forma a aumentar o erro total da pose.

3.2.5 Calibração dos erros sistemáticos usando o método UMBmark

Para efectuar a calibração dos erros sistemáticos é necessário efectuar com o robô vários percursos em forma de quadrados, com o tamanho do lado conhecido. Os percursos tem de ser efectuados nos dois sentidos (PR e CPR). No final da trajectória é necessário verificar a pose real do robô se possível com um sistema de posicionamento absoluto. A partir dos dados obtidos pela odometria e da pose final real é possível calcular o erro em x Δx e em y Δy. As trajectórias devem ser executadas a baixa velocidade de forma a evitar a derrapagem das rodas.

A partir dos dados recolhidos para cada conjunto de trajectórias realizadas no mesmo sentido é necessário calcular o centro de massa (cm) do erro em x e em y.

$$X_{cm} = \frac{1}{n} \times \sum_{i=1}^{n} \Delta x_i \tag{3.16}$$

$$Y_{cm} = \frac{1}{n} \times \sum_{i=1}^{n} \Delta y_i \tag{3.17}$$

A partir dos quais se obtém o centro de massa para os percursos realizados no sentido dos ponteiros do relógio ($X_{cm_{PR}}$, $Y_{cm_{PR}}$) e o centro de massa para os percursos realizados no sentido contrario ao ponteiro do relógio ($X_{cm_{CPR}}$,$Y_{cm_{CPR}}$).

Com os resultados é possível calcular o ângulo α e β, uma vez que sabemos o valor de D que é a dimensão do lado do quadrado da trajectória quadrangular.

$$\alpha = media\left(\frac{X_{cm_{PR}} + X_{cm_{CPR}}}{-4D}, \frac{Y_{cm_{PR}} - Y_{cm_{CPR}}}{-4D}\right) \tag{3.18}$$

$$\beta = media\left(\frac{X_{cm_{PR}} - X_{cm_{CPR}}}{-4D}, \frac{Y_{cm_{PR}} + Y_{cm_{CPR}}}{-4D}\right) \tag{3.19}$$

A partir de α calcula-se o erro de E_b com a seguinte expressão:

$$E_b = \frac{90^o}{90^o - \alpha} \tag{3.20}$$

Sabendo o E_b pela expressão

A partir de β calcula-se E_d com as seguintes expressões :

$$R = \frac{D/2}{\sin(\beta/2)} \tag{3.21}$$

$$E_d = \frac{R + b/2}{R - b/2} \tag{3.22}$$

Calcular o novo diâmetro para cada roda com as seguintes equações:

$$D_e = \frac{2}{E_d + 1} * D_m \tag{3.23}$$

$$D_d = \frac{2}{(1/E_d) + 1} * D_m \tag{3.24}$$

3.2.6 Erros aleatórios e modelo do erro da odometria

Para alem dos erros sistemáticos, existem outros erros imprevisíveis que resultam principalmente da interacção do robô com o ambiente à sua volta [27] . Estes erros, habitualmente designados de aleatórios ou não sistemáticos, são causados, por exemplo, pelo resvalamento das rodas (associado a pisos escorregadios ou aceleração excessiva), irregularidades no pavimento ou a presença de objectos inesperados no percurso. A própria resolução finita dos encoder's, bem como a discretização, pode ser considerada um fonte de erros não sistemáticos.

Para este tipo de erros não é possível efectuar uma compensação , é no entanto possível estimar a incerteza associada a pose do robô calculada pela odometria.

A informação é decomposta em três movimentos, duas rotações e uma translação. Cada um destes movimentos é afectado de uma forma independente, por ruído de distribuição normal e media nula .

Recorreu-se então ao modelo que apresenta a seguinte forma [13] :

$$\begin{bmatrix} x(i+1) \\ y(i+1) \\ \theta(i+1) \end{bmatrix} = \begin{bmatrix} x(i) + \Delta d(i) \times \cos(\theta(i) + \Delta\theta(i)) \\ y(i) + \Delta d(i) \times \sin(\theta(i) + \Delta\theta(i)) \\ \theta(i) + \Delta\theta(i) \end{bmatrix} + W(i) \tag{3.25}$$

No qual temos as equações do modelo cinemático em que a cada período de amostragem

existe um erro $W(i)$ associado. O ruído $W(i)$ considera-se uma distribuição normal de media nula e uma matriz de covariancia $Q(i)$ definida por:

$$E[W(i)] = 0 \tag{3.26}$$

$$E[W(i) * W(i)^T] = Q(i) \tag{3.27}$$

$E[.]$ representa o valor estimado, e pressupondo que os ruídos que afectam as estimativas de x, y e θ são independentes entre si a matriz $Q(i)$ é uma matriz diagonal:

$$\begin{bmatrix} Q_{11} & 0 & 0 \\ 0 & Q_{22} & 0 \\ 0 & 0 & Q_{33} \end{bmatrix} \tag{3.28}$$

Os elementos Q_{11}, Q_{22} e Q_{33} da diagonal são aproximados da seguinte forma:

$$Q_{11} = K_{uu}|\Delta d * \cos(\Delta\theta(i))| \tag{3.29}$$

$$Q_{22} = K_{uu}|\Delta d * \sin(\Delta\theta(i))| \tag{3.30}$$

$$Q_{33} = K_{u\theta}|\Delta d| + K_{\theta\theta}|\Delta\theta(i)| \tag{3.31}$$

OndeK_{uu} é o coeficiente que relaciona o erro da odometria ao longo $d(i)$ (que corresponde a translação) em relação Δd, $K_{u\theta}$ é o coeficiente que relaciona o erro da odometria ao longo de θ (movimento de rotação) em relação a Δd e por fim $K_{\theta\theta}$ é o coeficiente que relaciona o erro da odometria ao logo de θ em relação a $\Delta\theta(i)$.

Define-se a matriz de covariancia do erro da estimativa da pose do robô, $C(i)$, definida por:

$$C(i) = \begin{bmatrix} Var(x(i)) & Covar(x(i), y(i)) & Covar(x(i), \theta(i)) \\ Covar(x(i), y(i)) & Var(y(i)) & Covar(y(i), \theta(i)) \\ Covar(x(i), \theta(i)) & Covar(y(i), \theta(i)) & Var(\theta(i)) \end{bmatrix} \tag{3.32}$$

Var(.) e Covar(.) representam respectivamente, a variância e a covariância.

A matriz de covariância da estimativa da pose fornecida pela odometria é actualizada através de:

$$C(i+1) = \Phi(i) * C(i) * \Phi^T(i) + Q(i) \tag{3.33}$$

Onde Φ corresponde a matriz de transição de estado e é dada em cada instante de

amostragem por:

$$\begin{bmatrix} 1 & 0 & -\Delta d(i) * \sin(\theta(i)) \\ 0 & 1 & -\Delta d(i) * \cos(\theta(i)) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.34}$$

De modo a estimar os valores dos parâmetros K_{uu} , $K_{u\theta}$ e $K_{\theta\theta}$ é necessário realizar ensaios experimentais. Em primeiro lugar faz-se percorrer o robô uma linha recta segundo um eixo varias vezes, e em relação a posição final e a posição dos dados da odometria calculam-se as constantes definidas como:

$$K_{uu} = \frac{Var(X)}{med(x_{total})} \tag{3.35}$$

$$K_{u\theta} = \frac{Var(\theta)}{med(x_{total})} \tag{3.36}$$

$Var(x)$ e $Var(\theta)$ representam, respectivamente, a distancia do erro da medida de x e de θ, ao passo que $med(x_{total})$ corresponde a media das distancias totais percorridas.

Na segunda experiência regista-se o erro do sistema de odometria após uma rotação do robô sobre si próprio. Este ensaio deve ser repetido varias vezes, e com este é possível estimar o valor de $K_{\theta\theta}$.

$$K_{\theta\theta} = \frac{Var(\theta)}{med(\theta_{total})} \tag{3.37}$$

É no entanto de salientar que os valores destes parâmetros do modelo de erro da odometria não dependem apenas do robô , mas também variam com o tipo de piso onde se desloca. Assim sendo no caso de o robô se deslocar em pisos mais irregulares e rugosos, é de esperar que os valores destas constantes sejam de valor superior.

3.3 Localização por visão artificial

3.3.1 Modelo Pinhole camera

O modelo *pinhole camera* descreve a relação matemática entre as coordenadas de um ponto 3D e a sua projecção no plano de imagem . A grande maioria dos algoritmos desenvolvidos para visão computacional parte do principio que o modelo da câmara segue este modelo.

Neste modelo, não são contempladas as distorções geométricas nem que os objectos se encontrem desfocados, sendo uma primeira aproximação do mapeamento da cena 3D para 2D.

Neste modelo todos os raios de luz emitidos ou reflectidos por objectos passam através do ponto "pinhole" (furo fino) e forma uma imagem invertida desse objecto no plano da imagem (ver figura 3.7). Cada ponto do objecto de três dimensões corresponde a um ponto de duas dimensões no plano de imagem. A recta que define a projecção do ponto no plano de imagem é formada pelo ponto do objecto e pelo ponto Pinhole. A este fenómeno de projecção de espaço 3D para um plano da-se o nome de projecção projectiva[22].

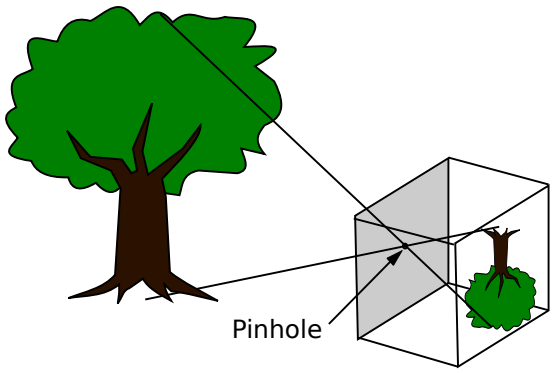


Figura 3.7: Representação típica do modelo *pinhole camera*

O modelo geométrico do modelo pinhole é por isso constituído por um plano de imagem π e o ponto C que se encontra no plano Ω o qual é chamado de plano principal . O ponto C é chamado de centro óptico ou ponto de focagem. Os planos π e Ω são paralelos e a distancia entre estes é a distancia focal a qual é representado por f (figura 3.8). A linha perpendicular ao plano principal (π) que passa pelo centro óptico C é chamado de eixo óptico, este intercepta o plano de imagem (Ω) no ponto P chamado de ponto principal [11]. A representação do modelo pinhole encontra-se na figura 3.8.

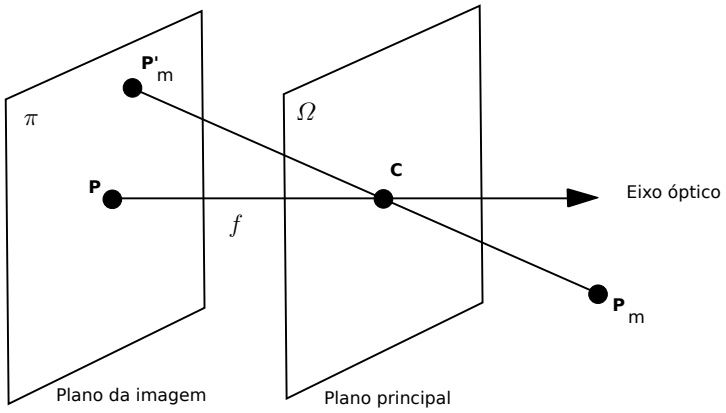


Figura 3.8: Modelo *pinhole camera*

Para a representação do sistema de coordenadas e por uma questão de simplificação o

plano de imagem é representado a distancia de f de C que no modelo original se encontra localizado a distancia de $-f$. Define-se o eixo Z na direcção frontal da câmara sendo coincidente com o eixo óptico e de origem no ponto C. O plano principal contém o referencial XYZ como se pode observar na figura 3.9.

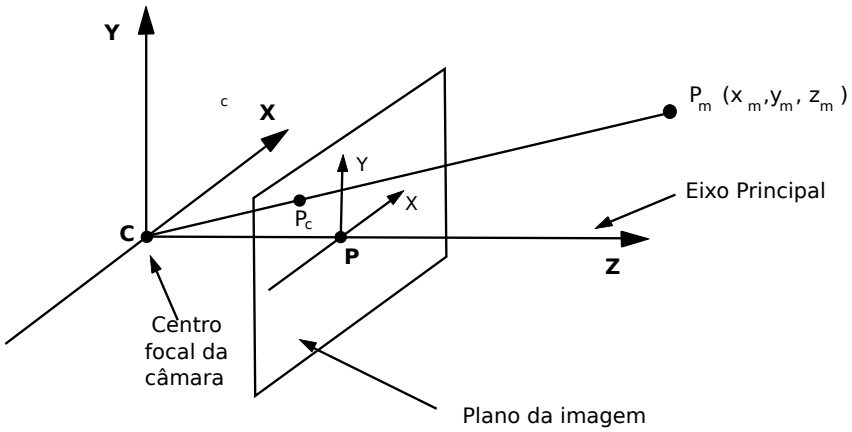


Figura 3.9: Representação dos eixos cordenados no modelo *pinhole camera*

O plano de imagem onde toda a cena 3D é projectada através do diafragma da câmara é paralelo ao plano XY do plano principal e encontra-se a distancia f da origem C.

Na figura 3.10 encontra-se projectado no plano de imagem o ponto P_m com coordenadas definidas relativamente ao sistema de coordenadas XYZ, As coordenadas deste ponto são (X_m, Y_m, Z_m). A sua projecção é o ponto P_c de duas dimensões (X_c, Y_c) onde o referencial deste sistema de coordenadas esta centrado no ponto P.

Desta forma é possível relacionar o sistema de coordenadas XYZ com o sistema de coordenadas do plano de imagem. A relação pode-se verificar na figura 3.10.

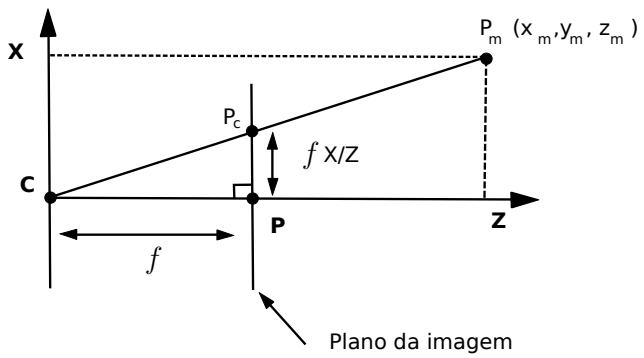


Figura 3.10: Vista de lado do modelo *pinhole camera*

Na figura esta representada a vista lateral do modelo pinhole onde por geometria se pode retirar as seguintes relações entre os dois sistemas de coordenadas:

$$x_c = f\frac{x_m}{z_m} \tag{3.38}$$

$$y_c = f\frac{y_m}{z_m} \tag{3.39}$$

Obtem-se desta forma a relação entre o que é projectado no plano de imagem e os objectos em três dimensões. É por isso possível conhecer a projecção de qualquer ponto de três dimensões no plano de imagem, sendo no entanto que o contrario já não é verdade sem haver informação adicional. Isto porque tendo um ponto no plano de imagem a informação relativa a profundidade não é possível recuperar , uma vez que para cada ponto do plano de imagem corresponde a uma recta no referencial XYZ. Por isso o valor de Z tem de ser conhecido ou obtido recorrendo a informação conhecida sobre o objecto que é observado.

O modelo *pinhole camera* assume que a origem das cordenadas na imagem do plano é feito no ponto principal P . Tipicamente em processamento de imagem o referencial da imagem encontra-se num canto da imagem e por isso é necessário ter em conta essa translação. Considerando que o canto da imagem é o canto inferior esquerdo como na figura 3.11 e as coordenadas X_0 e y_0 neste novo referencial definem a posição do ponto principal P.

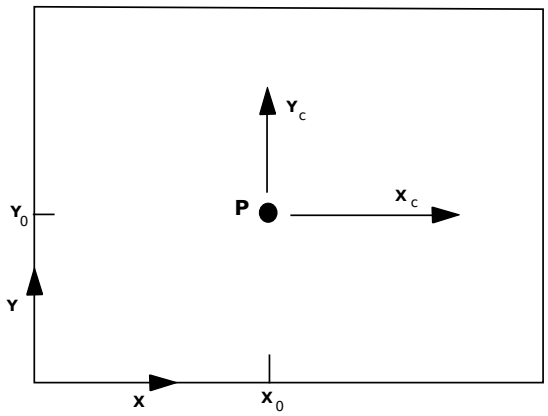


Figura 3.11: Eixo de coordenadas X_c Y_c no ponto principal e XY num dos cantos da imagem

As expressões que nos dão as coordenadas no plano de imagem com origem no canto da imagem são:

$$x_c = f\frac{x_m}{z_m} + P_x \tag{3.40}$$

$$y_c = f\frac{y_m}{z_m} + P_y \tag{3.41}$$

Onde P_x é igual a X_0 e P_y é igual a Y_0.

3.3.2 Distorção e Calibração

Distorção Óptica

A projecção da imagem na câmara esta sujeita a fenómenos de distorção ópticas sendo os dois mais significativos a distorção radial e a tangencial. Estes tipos de distorção podem ser compensadas por algoritmos de correcção.

Distorção radial

A distorção radial é de fácil detecção já que ocorre uma distorção geométricas, dos pixeis nas margens da imagem. A este fenómeno e normalmente chamado de "efeito barril" ou "olho de "peixe". Este tipo de erros diminui com a qualidade da câmara e a qualidade das lentes. A distorção aumenta desde o centro de imagem até às margens sendo nulo no centro óptico da [12].

Desta forma, o efeito Barril é mais evidente nas margens. Este tipo de erros pode ser compensado por algoritmos ou filtros. Na figura 3.12 encontra-se uma representação deste tipo de distorção.

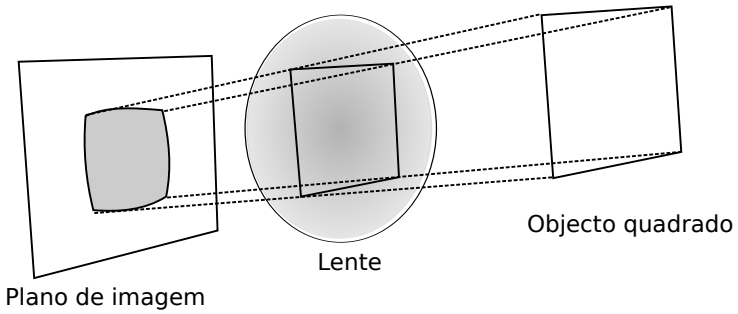


Figura 3.12: Distorção radial provocada pelas lentes

A influencia da distorção radial na imagem capturada pela câmara é bastante acentuada quando são usadas lentes com grande abertura angular. Comparando a magnitude dos dois tipos de distorção descritos neste documento, a distorção radial tem um maior impacto na imagem que as distorções tangenciais.

Distorção tangencial

A distorção tangencial, é originada por um fenómeno físico designados por efeito ”prisma” ou descentralização. Esta distorção é resultante da impossibilidade dos fabricantes, em alinhar os eixos ópticos das lentes que compõem a objectiva.

Figura 3.13: Distorção tangencial provocada pelas lentes [17]

Correcção da Distorção

Para um conjunto de câmara mais lentes é possível caracterizar-se a distorção a partir de um modelo que permite corrigir a imagem original (com distorção) de forma a obter-se uma imagem corrigida.

A correcção consiste na determinação da nova localização dos pixeis corrigidos utilizando a localização e a informação de cada um dos pixeis da imagem original.

A correcção é efectuada com o recurso as equações apresentadas de seguida , onde (x, y) corresponde à localização original do ponto com distorção e $(x_{corrigido}, y_{corrigido})$ correspondem a nova localização do ponto sem distorção. Os parâmetros $K1$, $K2$ e $K3$ correspondem aos parâmetros da distorção radial e os parâmetros $P1$ e $P2$ correspondem aos parâmetros de distorção tangencial [11].

Distorção radial:

$$x_{corrigido} = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \tag{3.42}$$

$$y_{corrigido} = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) \tag{3.43}$$

Distorção tangencial:

$$x_{corrigido} = x + [2P_1 + P_2(r^2 + 2x^2)] \tag{3.44}$$

$$y_{corrigido} = y + [P_1(r^2 + 2y^2) + 2P_2x] \tag{3.45}$$

$$r = \sqrt{(x_d - c_x)^2 + (y_d - c_y)^2} \tag{3.46}$$

Onde (x_d, y_d) correspondem às coordenadas (x, y) de um ponto (da imagem original com distorção) e (c_x, c_y) corresponde às coordenadas do centro óptico.

Calibração

A calibração de um sistema de visão é utilizada para determinar os parâmetros ou coeficientes do modelo da câmara utilizado, que relaciona pontos no espaço tridimensional em pontos no plano imagem e vice-versa. Estes parâmetros variam para cada sistema de visão e corresponde à determinação da sua geometria interna e das suas características ópticas (as quais são conhecidas por parâmetros intrínsecos) . Também é contemplada a orientação e posição 3D da câmara relativamente a um determinado sistema de coordenadas no mundo (parâmetros extrínsecos), o processo de determinação destes parâmetros é por vezes designado por estimativa da pose.

Para esta dissertação existe a necessidade de calcular os parâmetros intrínsecos, estes dizem respeito ao conjunto câmara mais lentes. Estes relacionam as coordenadas de um ponto nas três dimensões com a sua projecção no plano de imagem, tendo em conta as características físicas do conjunto. O conjunto de características que constituem os parâmetros intrínsecos são:

- Distância focal (f)
- Tamanho do pixel (s_x, s_y)
- Coeficientes de distorção
- Localização do ponto principal (c_x, c_y)

O algoritmo de Tsai [29] é um algoritmo que permite a calibração de câmaras, este é constituído por varias etapas com as quais são obtidos os parâmetros de calibração:

- Na primeira etapa é efectuada a transformação do objecto do sistema de coordenadas tridimensional da câmara para a projecção no plano de imagem através da geometria

do objecto e com recurso ao modelo *pinhole camera*. Com esta metodologia obtém-se a distancia focal (f).

- Na etapa seguinte o que se procura retirar é os parâmetros de distorção radial para isso recorre as características geométricas do objecto e faz uma estimativa destes parâmetros .

- Na etapa final, estima os factores de escala (s_x, s_y), a localização do ponto principal (c_x, c_y) e recalcula todos estes parâmetros de forma recursiva.

O algoritmo de Tsai usa uma particularidade física do sistema de visão, esta particularidade trata-se do alinhamento radial. A qual é definida em função da rotação relativa e sua translação (excepto na componente Z) entre a câmara e os pontos de calibração. Ver figura 3.14. Seguindo as seguintes características [26]:

- É assumido que a distorção radial em nada afecta a profundidade do espaço envolvente, não importa qual a magnitude da distorção. A direcção do vector O_iP_d (vector definido origem O_i e o ponto P_d que se encontra no plano de imagem) mantém-se constante e radialmente alinhado com o vector $P_{Oz}P$ (formado pela origem P_{Oz} - Dada pela coordenada Z do ponto P - e o ponto P) a coordenada Z destes dois pontos é igual .

- A distancia focal f não influencia a direcção do vector O_iP_d

- O ponto do mundo é transladado e rodado em X, Y logo O_iP_d será sempre paralelo a $P_{Oz}P$ para todos os pontos. A componente z da translação não altera a direcção de O_iP_d .

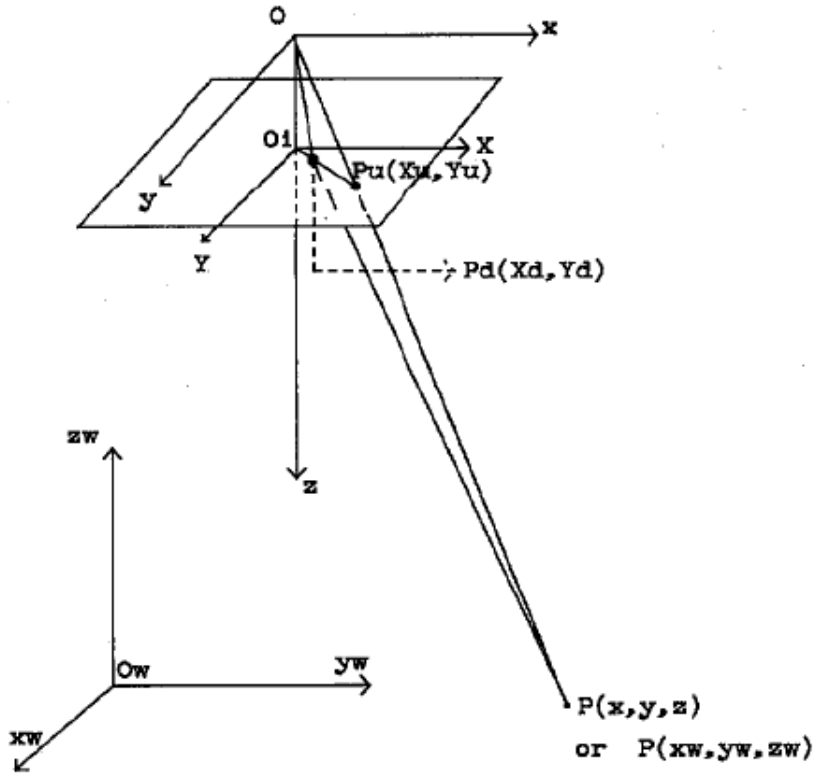


Figura 3.14: Geometria da Câmara com projecção radial e distorção de lentes.

Para obter os parâmetros intrínsecos obtidos para a câmara utilizada, recorreu-se a toolbox para matlab de nome *Camera Calibration Toolbox for Matlab* a qual se encontra disponível em [4] esta implementa um algoritmo que recorre aos princípios do algoritmo de Tsai com o qual é também possível obter os parâmetros de distorção tangencial. O objecto que com o qual o algoritmo funciona é um tabuleiro de xadrez onde vértices dos quadrados funcionam como pontos de calibração.

3.3.3 Algoritmo de detecção de quadrados

O sistema de localização deste trabalho assenta em visão artificial com a qual é necessário retirar do mundo envolvente informação relevante para o correcto funcionamento do sistema.

Como já foi referido o sistema baseia-se na detecção de marcadores com o formato de quadrados. A detecção no plano de imagem destes quadrados é que permitirá ao sistema conhecer a pose do robô no mundo uma vez que estes é que representação o sistema de coordenadas do mundo.

Nesta secção apresenta-se os algoritmos utilizados para a detecção destes marcadores. Estes permitem variar o tamanho do quadrados a detectar e foi desenvolvido de forma a

poder ser imune as variações de luminosidade que ocorrem durante o dia.

A *frame* (F) obtida pela câmara é uma matriz de pixeis, em que cada elemento $F[u, v]$ mapeia a intensidade de luz captada numa escala de cinzentos compreendida entre 0 (preto) e 255 (branco).

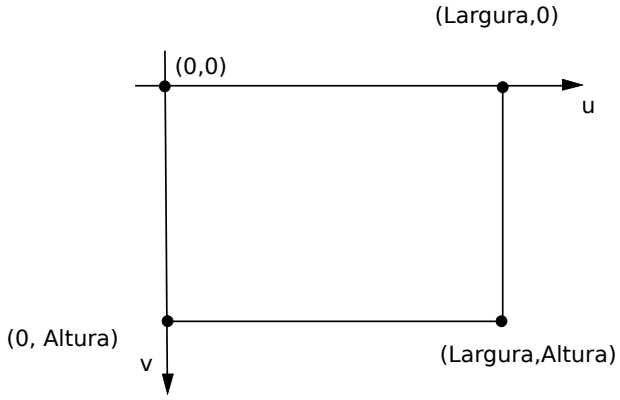


Figura 3.15: Cordenadas e dimensão de uma *frame*

Na figura 3.16 encontra-se um exemplo de uma frame captada. A imagem é processada na totalidade apenas na inicialização do sistema. Após o calculo de posição inicial esteja concluída apenas se vai processar a porções da imagem isto porque após este passo é possível prever onde os quadrados se devem encontrar sendo apenas necessário processar partes da imagem o que resulta em ganhos de tempo de processamento.

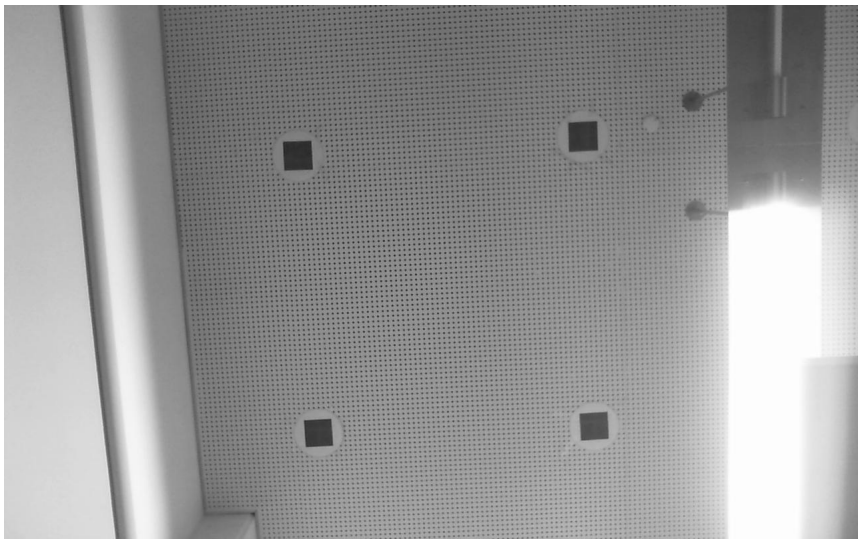


Figura 3.16: Exemplo de de uma *frame* captada

Como a detecção dos marcadores necessita de ser efectuado mesmo que o robô esteja em movimento, é fundamental garantir que todo o processamento se realize num tempo inferir ao frame rate que no caso implementado é de 10 FPS. Por isso a aplicação de software que implementa a detecção de marcadores é coordenada pela recepção dos frames da câmara. No caso de o processamento da imagem ultrapassar o intervalo de tempo até a chegada de uma nova imagem, a nova imagem será ignorada.

O na figura 3.17 encontram-se as varias etapas que o algoritmo de detecção de quadrados executa.

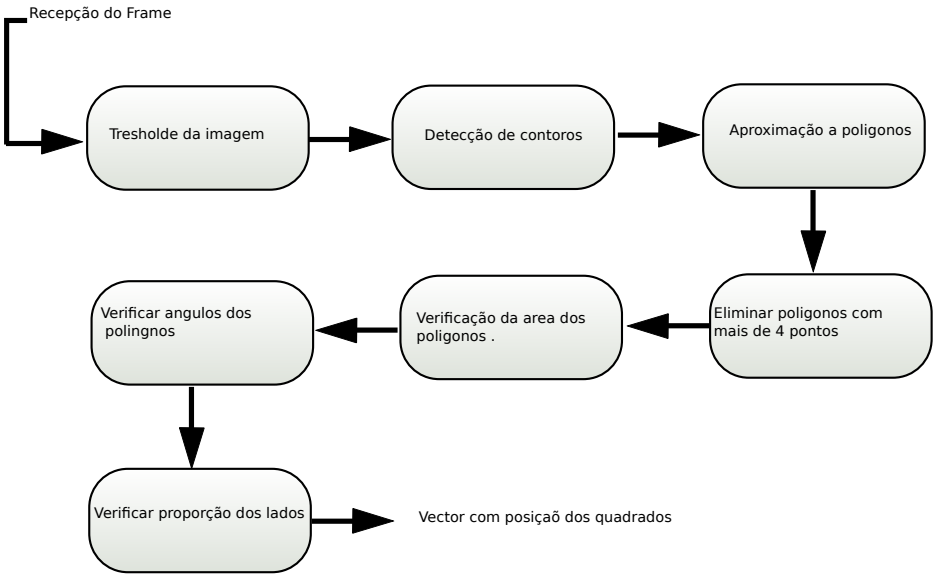


Figura 3.17: Diagrama de tratamento de imagem

Aplicação do threshold

Após a recepção da uma nova imagem, esta é processada varias vezes com diferentes níveis de threshold por isso todo o restante processamento apresentado nas outras subsecções deste sub-capitulo é repetido para os vários níveis de threshold. O threshold transforma a imagem numa imagem binária, dito de outra forma todos os pontos que se encontrem com uma intensidade de luz a baixo do valor de threshold passam a ter valor zero e se estiver a cima a valer um .

Na figura 3.18 pode-se verificar o efeito da aplicação de um threshold a uma imagem.

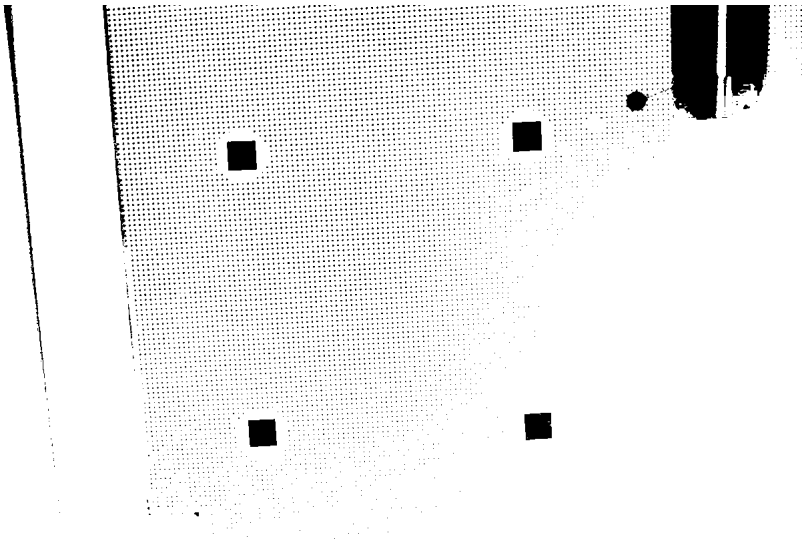


Figura 3.18: Aplicação de threshold de valor 90 a imagem 3.16

Os vários valores de threshold a serem aplicados são calculados pela seguinte formula .

$$threshold_{nivel} = l * 255/N \tag{3.47}$$

Onde N é o numero de vezes que a imagem é processada mais uma. e l qual a imagem que vai ser processada. Por outras palavras, se por exemplo a imagem for processada com seis niveis threshold o valor de N é igual a sete, quando a primeira imagem é processada valor de l é igual a um, para a segunda é igual a dois e assim sucessivamente até ao numero total de imagem que no caso é seis.

A grande vantagem que esta metodologia tem é ser independentemente da variação da claridade ambiente, isto porque faz com que exista sempre um nível de threshold aplicado que é o adequado para que o restante processamento da detecção dos quadrados não falhe a presença de nenhum deles.

Foi no entanto utilizada outra metodologia conhecida como *Canny algorithm* é um algoritmo de detecção de cantos. Este produziu bons resultados e com um gasto de tempo de processamento cerca de dois terços do utilizado, mas não era tão fiável como a solução que foi utilizada porque na ocorrência de variações de luminosidade falhava a detecção de quadrados.

Detecção de contornos

A cada imagem resultante de cada um dos thresholds aplicados é aplicado um algoritmo de detecção de contornos. O algoritmo devolve uma lista de pontos na imagens onde existem

variações rápidas dos níveis de brilho. Estes pontos definem linhas ou curvas na imagem que realçam os contornos dos objectos.

O interesse na detecção de contornos advém da redução da quantidade de informação a processar e por poderem corresponder a limites físicos dos objectos. No caso pretende-se que o algoritmo detecte os contornos com a da forma dos quadrados.

Na figura 3.19 encontra-se o resultado da aplicação deste tipo de algoritmo.

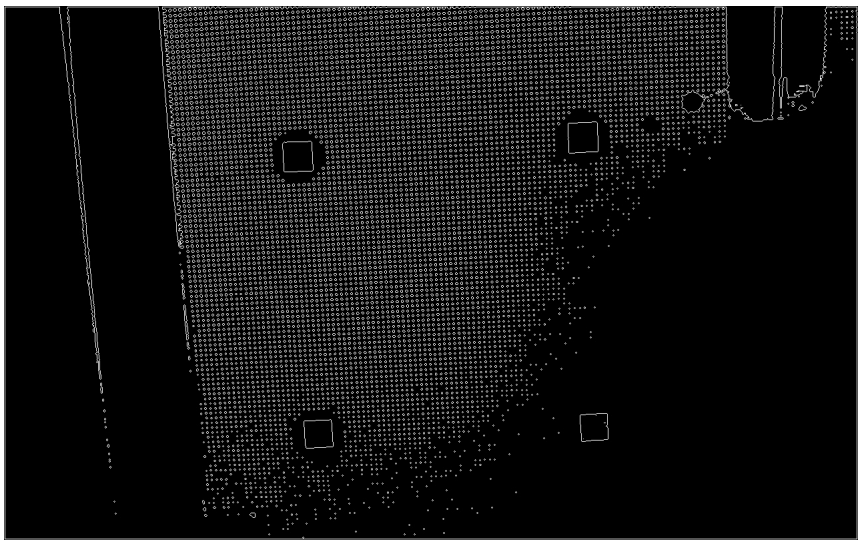


Figura 3.19: Aplicação de detecção de contornos

Aproximação a polígono

Após a detecção dos contornos, e uma vez que o objectivo do algoritmo é a detecção de quadrados os quais são polígonos, aplica-se um algoritmo conhecido como *Douglas-Peucker Algorithm* [11] , que para cada contorno detectado faz uma aproximação a um polígono, o qual descreve com menos pontos o formato do contorno. Isto é o algoritmo devolver apenas os pontos que representam os vértices do polígono que é uma aproximação do contorno detectado.

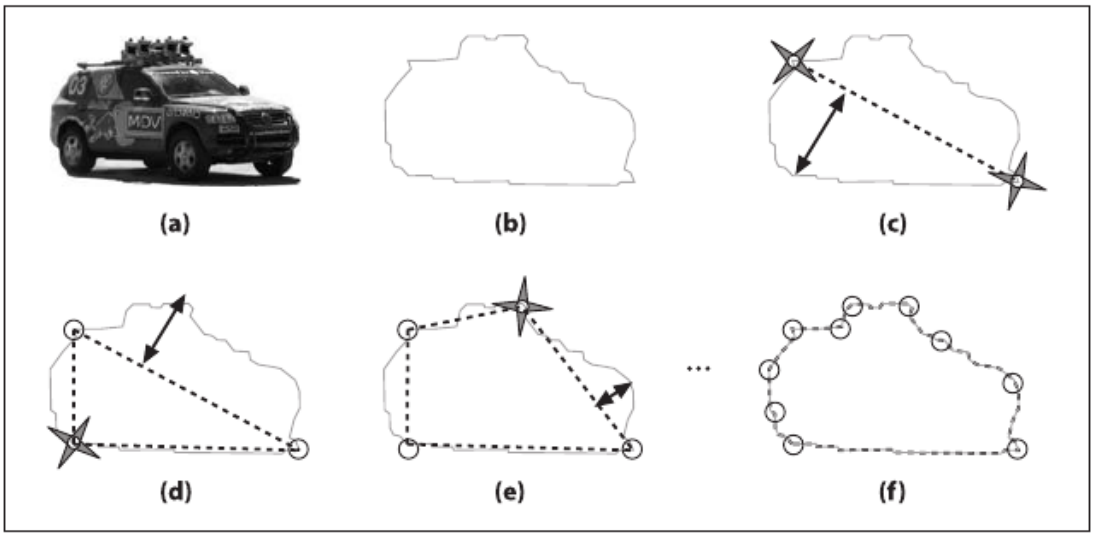


Figura 3.20: Visualização do algoritmo de Douglas Peucker [11]

Na figura 3.20 estão representados os passos executados para converter a imagem original num polígono. Onde (a) é a imagem original (b) a detecção de contorno da imagem, (c) O algoritmo começa por detectar os dois vértices que se encontram a maior distancia. os restantes vértices são adicionados de forma recursiva.

Dito de outra forma o algoritmo começa por detectar no contorno os dois pontos mais distantes e marca-os como vértices do polígono. Entre estes dois pontos o algoritmo vai procurar qual é o ponto mais distante entre o contorno e a linha formada entre os dois vértices iniciais (d). Se o ponto encontrado estiver a uma distancia da linha maior do que um valor ε que é um valor predefinido, o novo ponto é definido como um novo vértice e vai recursivamente criar uma nova linha entre o novo vértice e os outros que já estejam definidos e procurar o ponto mais distante do contorno até essa linha. Assim que não haja mais nenhum ponto no contorno que se encontre a uma distancia maior que ε o algoritmo termina.

Descartar polígonos

Os polígono podem ter variados formatos. O polígono que nos interessa detectar é o quadrado e por isso a primeira limitação a ser imposta é a de eliminar os polígonos que não tem quatro vértices, o que resulta em vários polígonos como é exemplificado na figura 3.21 e que representa uma amostra do tipo de polígonos que se podem formar com quatro vértices. É por isso necessário recorrer a outras características dos quadrados para que estes sejam correctamente identificados.

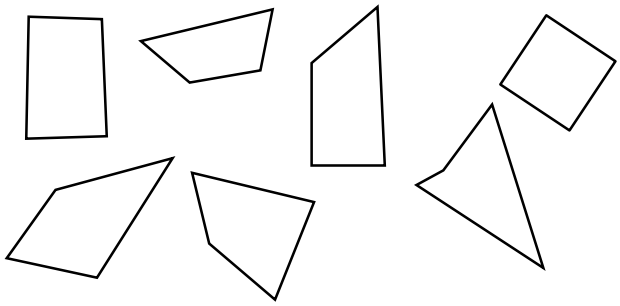


Figura 3.21: Exemplos de polígonos com quatro cantos

Para que apenas resultem polígonos quadrados é necessário efectuar as seguintes verificações

- Verificar os ângulos entre os quarto cantos
- Verificar a proporção dos lados

Para verificar o ângulo começa-se por efectuar o calculo de dois vectores, onde d_{x_1} e d_{y_1} são as componentes do primeiro vector e d_{x_2} e d_{y_2} as componentes do segundo:

$$d_{x_1} = p_{x_1} - p_{X_0} \tag{3.48}$$

$$d_{y_1} = p_{y_1} - p_{y_0} \tag{3.49}$$

$$d_{x_2} = p_{x_2} - p_{X_0} \tag{3.50}$$

$$d_{y_2} = p_{y_2} - p_{y_0} \tag{3.51}$$

A partir dos vectores calcula-se o ângulo (θ) entre os dois vectores :

$$cos\theta = \frac{(d_{x_1} * d_{x_2} + d_{y_1} * d_{y_2})}{\sqrt{(d_{x_1}^2 + d_{y_1}^2) * (d_{x_2}^2 + d_{y_2}^2)}} \tag{3.52}$$

O ângulo (θ) terá de ter um intervalo de aceitação onde o valor de referencia é $\pi/2$.

Para calcular a proporção de cada lado, a partir dos vectores obtém-se o modulo de cada um deles :

$$modulo_1 = \sqrt{d_{x_1}^2 + d_{y_1}^2} \tag{3.53}$$

$$modulo_2 = \sqrt{d_{x_2}^2 + d_{y_2}^2} \tag{3.54}$$

E calcula-se a relação entre eles:

$$rela = \frac{modulo_1}{modulo_2} \tag{3.55}$$

Se os lados tiverem a mesma proporção o valor de *rela* terá o valor de um. Desta forma podemos definir um intervalo, por exemplo de [0.9 , 1.1] em que são considerados proporcionais e por isso se trata de um quadrado.

Calculo do centro do quadrado

Os polígonos que sobram da implementação deste algoritmo são obrigatoriamente quadrados, a informação que é necessária retirar da imagem para o próximo passo é onde se encontra no plano de imagem o centro destes quadrados. Como o poligono destes quadrados é definido por quatro pontos efectua-se o seguinte calculo:

$$q_x = \frac{p_{1_x} + p_{2_x} + p_{3_x} + p_{4_x}}{4} \tag{3.56}$$

$$q_y = \frac{p_{1_y} + p_{2_y} + p_{3_y} + p_{4_y}}{4} \tag{3.57}$$

Remoção da distorção

A posição em pixeis dos quadrados de interesse tem de seguida ser recalculados de forma a remover a sua distorção. Estas novas localizações em pixeis são obtidas pelas formulas descritas na secção de "Distorção e Calibração". A partir destes valores já é possível recorrer ao modelo *pinhole camera* para obter a posição relativa dos quadrados em relação a câmara.

3.3.4 Cálculo da posição e orientação inicial

As marcas usadas são todas iguais e não permitem retirar informação sobre a orientação nem sobre a sua posição do robô por si só, por isso é necessário criar regras de inicialização para que o sistema reconheça sistematicamente o mesmo referencial do mundo e sempre relativo a mesma posição de forma a poder criar trajectórias que possam ser repetidas. A condição inicial mais relevante é que sempre que o sistema é inicializado terá de se encontrar dentro de uma determinada área. A área encontra-se por baixo dos quadrados que definem o sistema de coordenadas da zona de trabalho. A posse do robô tem de ter obedecer as seguintes condições:

- Em relação ao eixo dos X o robô terá de estar entre $[-tamanho_{rede}/2, tamanho_{rede}/2]$
- Em Y terá de estar entre $[0, tamanho_{rede}]$ tamanho da rede.
- Em relação a orientação terá de estar entre $[-\pi/4, \pi/4]$

Onde $tamanho_{rede}$ é o que define a distancia ente quadrados.

Na figura 3.22 para que o sistema de coordenadas do mundo fique definido como o apresentado, o robô terá de se localizar dentro da zona a cinza e com a orientação entre os ângulos apresentados.

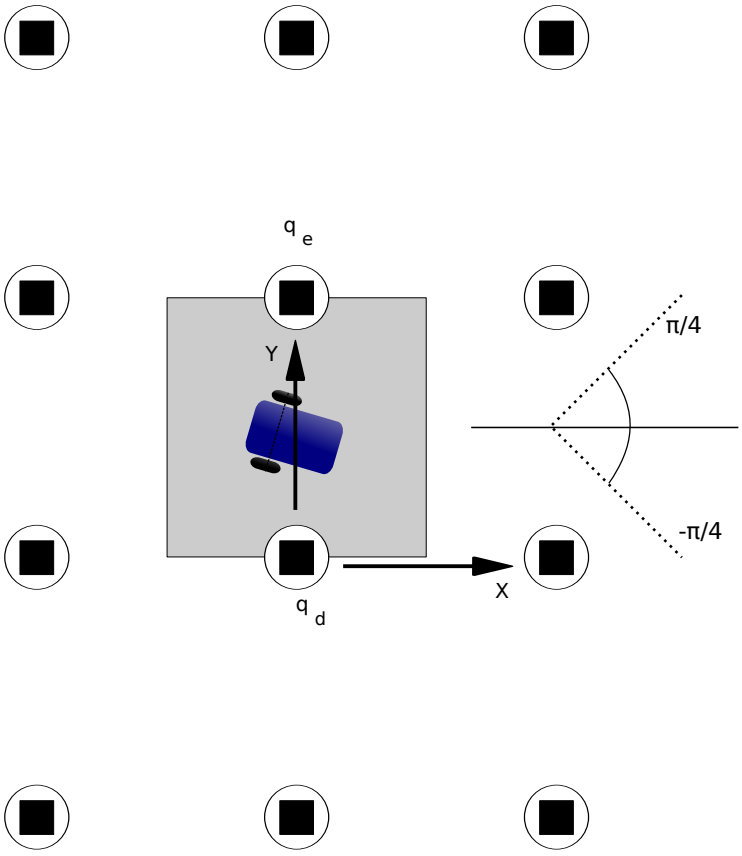


Figura 3.22: Definição do sistema de coordenadas do mundo relativamente ao robô(visto de cima) e demonstração das condições iniciais necessárias.

No caso de o robô não se encontrar dentro desta área o sistema de coordenadas vai ficar definido tendo em conta os mesmos princípios, mas simplesmente se o sistema depender de trajectórias previamente definidas este vai executar essas trajectórias relativamente a esse novo sistema de coordenadas do mundo. Por existe a necessidade que as condições de inicialização sejam cumpridas.

O algoritmo de calculo da posição inicial depois de conhecida a posição dos quadrados no plano de imagem segue o seguinte esquema:

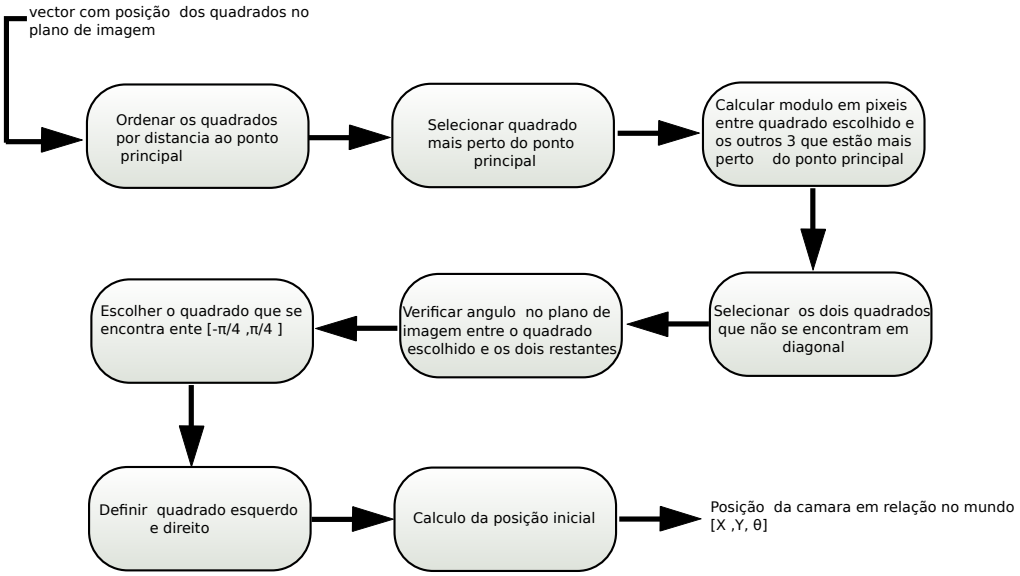


Figura 3.23: Diagrama que descreve as operações efectuadas para retirar a informação necessária dos quadrados detectados para o calculo posição inicial

Nos seguintes sub-capítulos encontram-se descritas todas as etapas de implementação deste algoritmo.

Detecção dos quadrados de interesse

Para poder calcular a pose inicial do robô é necessário detectar correctamente dois quadrados. Estes terão de ser os que se encontram referenciados na figura 3.22 como q_e e q_d. Após todos os quadrados no ambiente serem detectados existe a necessidade da implementação de um algoritmo que detecte de entre todos os dois de interesse.

O algoritmo começa com a ordenação dos quadrados que é efectuado tendo em conta a distancia ao ponto principal, na figura 3.24 encontra-se um exemplo da distancia linear entre um quadrado e o ponto principal.

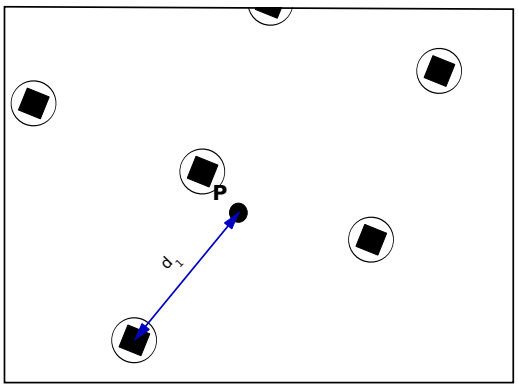


Figura 3.24: Representação da distancia linear d_1 de um quadrado ao ponto principal no plano de imagem

Para cada quadrado é efectuado o seguinte calculo:

$$d = \sqrt{(q_{n_x} - x_0)^2 (q_{n_y} - y_0)^2} \tag{3.58}$$

Os quatro quadrados de menor distancia linear são guardados. E o de menor distancia linear é fixado como um dos quadrados de interesse (q_j). Este é fixado como quadrado de interesse porque terá de ser um dos dois quadrados que corresponde no mundo a posição $x = 0$ $y = 0$ ou $x = 0$ $y = 1$. Isto resulta da condição inicial definida relativamente a área onde o robô se pode encontrar. O que faz com que um dos dois quadrados q_e ou q_d seja o que se encontra o mais próximo próximo do ponto principal.

Depois de definido o primeiro quadrado de interesse (q_j) , é necessário encontrar qual dos outros quadrados corresponde ao par que permite determinar a posição inicial. Para isso é necessário classificar os restantes quadrados de forma a poderem ser excluídos todos menos o único que realmente satisfaça todas as condições necessárias.

A primeira classificação é efectuada é a de saber quais são os quadrados que se encontram no mesmo eixo ou se encontram em diagonal. Para isso calcula-se em relação ao quadrado escolhido (q_j) a distancia linear ente este e os três restantes. Na figura 3.25 encontra-se a representação deste calculo.

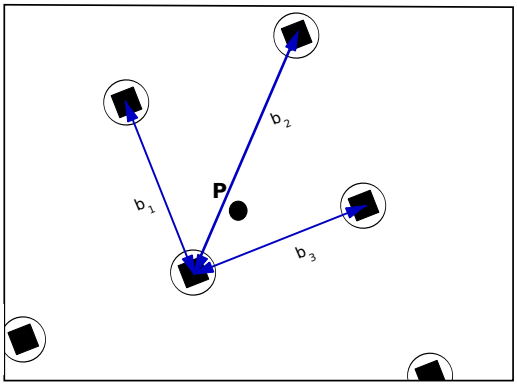


Figura 3.25: Representação das distancias lineares b_n do quadrado mais perto para os restantes quadrados validos no plano de imagem

$$b_n = \sqrt{(q_{n_x} - q_{j_x})^2 + (q_{n_y} - q_{j_y})^2} \tag{3.59}$$

A partir do calculo da distancia linear é possível verificar que a distancia de b que tiver o valor mais baixo b_{minimo} corresponde ao valor do tamanho da rede de quadrados. De forma a poder excluir os quadrados que se encontram em diagonal sem correr o risco de eliminar os quadrados que se encontram nos mesmos eixos é necessário dar uma margem de erro para definir quais são os que efectivamente são os de interesse. Para tal foi utilizado o seguinte principio:

A hipotenusa (h) de um triângulo rectângulo com dois lados iguais é igual a:

$$h = \sqrt{(lado)^2 + (lado)^2} \tag{3.60}$$

então a relação é :

$$h = \sqrt{2} * |lado| \tag{3.61}$$

e por isso :

$$h \approx 1.414213 * |lado| \tag{3.62}$$

Sabendo isto é possível definir um forma de eliminar os quadrados em diagonal sem necessariamente saber a partida o tamanho do lado em pixeis:

todos os quadrados que apresentem um valor para a distancia linear em que não cumpram a seguinte condição:

$$b_n < 1.2 * |b_{minimo}| \tag{3.63}$$

São considerados os quadrados em diagonal e por isso são excluídos. Na figura 3.25 o quadrado que se encontra em diagonal passa a ser excluído. Mas tomando como exemplo a

figura 3.24 este critério não produz efeito porque os três quadrados restantes não se encontram em diagonal em relação ao quadrado mais perto do ponto principal.

As restantes classificações a efectuar para descobrir qual dos quadrados é o certo terá de preencher as seguintes duas condições:

- A partir da posição relativamente ao eixo dos xx do quadrado de interesse (q_j) é necessário saber se este se encontra a esquerda ou a direita do ponto principal. Se este estiver a esquerda o segundo terá de estar a direita. Se estiver a direita o segundo terá de estar a esquerda.

- Por ultimo dos quadrados que se encontram do lado oposto ao primeiro de interesse é necessário perceber qual deles respeita o ângulo entre $[-\pi/4, \pi/4]$ relativamente ao eixo dos XX do plano de imagem

Para saber onde se encontra o segundo quadrado necessário para executar os cálculos da posição inicial é necessário verificar de que lado se encontra o primeiro quadrado escolhido para tal recorre-se ao seguinte calculo :

$$local = x_{quadradoinicial} - x_0 \tag{3.64}$$

Onde x_0 é o valor a posição no plano de imagem relativamente ao eixo dos xx o ponto principal . Se o valor *local* for negativo este encontra-se a esquerda se for positivo este encontra-se a direita. Na figura 3.26 é fácil de determinar qual o quadrado que se encontra mais perto do centro focal e verifica-se que neste caso está a esquerda e por isso a área de interesse a direita e está assinalada a verde.

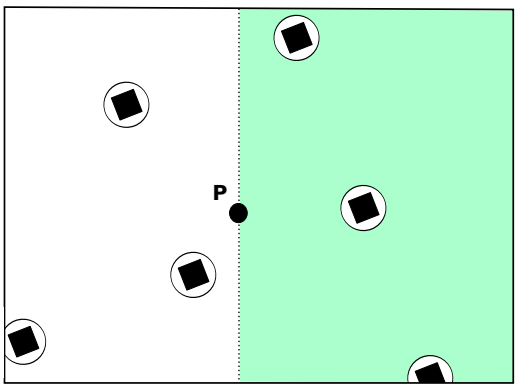


Figura 3.26: Área de procura de segundo quadrado de interesse segundo a condição de se encontrar do lado oposto ao relativamente ao primeiro em relação ao ponto principal

Na área anterior podem ainda encontrar-se vários quadrados não tenham sido eliminados pelos algoritmos descritos anteriormente e por isso é necessário entrar com uma ultima

condição. O segundo quadrado de interesse terá de formar um ângulo entre o primeiro o eixo dos XX que se encontre dentro do intervalo de ângulos definido nas condições iniciais que o qual é de $[-\pi/4, \pi/4]$.

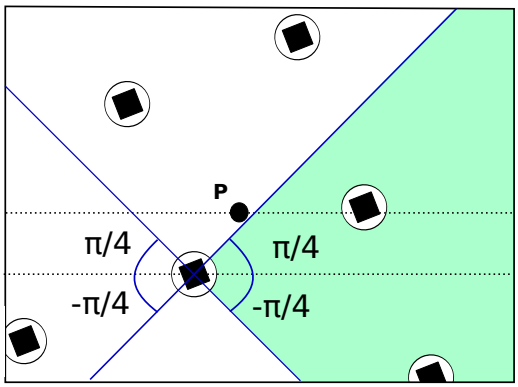


Figura 3.27: Condição do ângulo

Na figura 3.27 pode-se verificar que apenas sobra um quadrado de interesse relativamente a esta condição. Mesmo que o quadrado que se encontra no canto inferior direito estivesse inteiramente dentro do plano de imagem de forma a ser detectado já teria sido eliminado pela distancia ao ponto principal e também devido a se encontrar em diagonal relativamente ao primeiro quadrado de interesse.

Calculo da pose inicial

Já obtida a posição dos dois quadrados de interesse no plano de imagem, já temos toda a informação necessária para o calculo da pose inicial do robô.

Uma vez que o sistema funciona em ambiente estruturado sabemos a partida o tamanho da rede que foi colocada e por isso sabemos a distancia física entre os quadrados. Recorrendo ao modelo pinhole apenas sabemos que o que a posição de uma quadrado no no plano de imagem corresponde a uma recta relativamente ao referencial tridimensional da câmara. Tendo a posição de dois quadrados no plano de imagem temos duas rectas.

Neste caso temos dois quadrados e a distancia física entre estes dois, e como é assumindo que o plano de imagem é paralelo ao tecto o valor da profundidade Z é igual para ambos os quadrados. Recorrendo a modelo *pinhole camera* sabe-se que :

$$x_c = f\frac{x_m}{z_m} + P_x \tag{3.65}$$

$$y_c = f\frac{y_m}{z_m} + P_y \tag{3.66}$$

A distancia em pixeis dos dois quadrados corresponde a distancia entre quadrados no mundo. E por isso para calcular a profundidade coloca-se o sistema como só tive-se um eixo, calculando a distancia linear em pixeis de um quadrado para o outro.

$$DistanciaPixeis_x = (Q_{d_x} - Q_{e_x}); \tag{3.67}$$

$$DistanciaPixeis_y = (Q_{d_y} - Q_{e_y}); \tag{3.68}$$

$$Distancialinear_{pixeis} = \sqrt{(DistanciaPixeis_y)^2 + (DistanciaPixeis_x)^2} \tag{3.69}$$

Sabendo a distancia em pixeis pode-se deduzir a seguinte formula:

$$Z = f * tamanho_{rede}/Distancialinear_{pixeis} \tag{3.70}$$

Obtendo-se desta forma a profundidade (Z), a partir desta obtém-se a posição relativa entre a câmara e os quadrados detectados. Isto é a posição dos quadrados no referencial tridimensional da câmara.

$$X_c = (Z * (Q_{e_x} - P_x)/f) \tag{3.71}$$

$$Y_c = (Z * (Q_{e_x} - P_y)/f) \tag{3.72}$$

Agora é necessário fixar o sistema de coordenadas do mundo de forma a podermos calcular a posição da câmara relativamente a esse referencial.

Uma vez que já foi determinado qual dos dois quadrados se encontra a esquerda e a direita do ponto principal podemos definir que o quadrado esquerda corresponde a posição $(0, tamanho_{rede})$ e o da direita a posição $(0, 0)$ no referencial do mundo . O sistema de e coordenadas do muno relativamente ao que se vê no plano de imagem fica como o representado na figura 3.28.

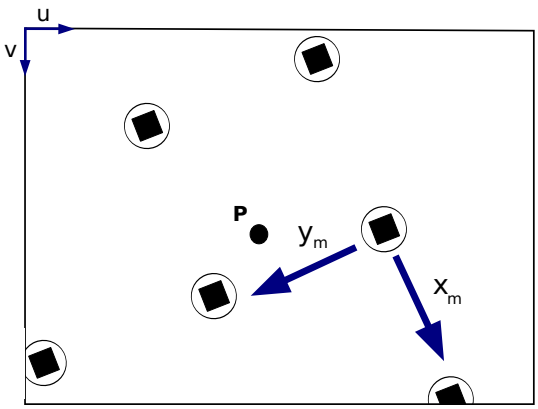


Figura 3.28: Condição do ângulo

Aparentemente este sistema de coordenadas parece invertido mas isto deve-se a que na câmara se encontra a projecção do tecto e por isso o sistema de coordenadas do mundo para uma correcta visualização também terá de ser projectado no "tecto".

Como o sistema parte do principio que o plano de imagem se encontra paralelo ao tecto a posição da câmara no Mundo é representado pelo ponto principal.

Para saber em que localização que este ponto se encontra em relação ao referencial do mundo tem de é necessário efectuar a translação e a rotação do sistema de coordenadas da câmara para o sistema de coordenadas do mundo. Por isso começa-se por calcular a rotação da Câmara:

$$d_x = q_{e_x} - q_{d_x} \tag{3.73}$$

$$d_y = q_{e_y} - q_{d_y} \tag{3.74}$$

$$\theta = -atan(\frac{d_y}{d_x}) \tag{3.75}$$

O valor de teta é negativo porque a rotação a efectuar é de alinhar os dois sistemas de coordenadas.

Para efectuar a translação é necessário somar o valor da posição do quadrados esquerdo, ao ponto principal. Dito de outra forma o Ponto principal p no sistema de coordenadas tridimensionais da câmara encontra-se no eixo dos xx e dos yy com valor zero.

$$p_t = p - q_d \tag{3.76}$$

Depois de efectuar a translação efectuas-se a rotação

$$p_r = \begin{bmatrix} cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \times p_t \tag{3.77}$$

Por fim para que os sistemas de coordenadas fiquem coerentes trocam-se os eixos e inverte-se as coordenadas.

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \times p_r \tag{3.78}$$

É obtida nesta forma a pose da câmara com (x_c, y_c , θ)

Para saber a pose do robô é necessário conhecer a distancia em x_r e em y_r e a rotação αque a câmara tem em relação ao sistema de coordenadas do robô.

$$pos_{robo} = \begin{bmatrix} cos(\theta - \alpha) & \sin(\theta - \alpha) \\ -\sin(\theta - \alpha) & \cos(\theta - \alpha) \end{bmatrix} \times \begin{bmatrix} x_r \\ y_r \end{bmatrix} + \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{3.79}$$

A orientação do robô é $(\theta - \alpha)$.

3.3.5 Previsão da localização dos quadrados no plano de imagem

A previsão da posição das marcas no plano de imagem serve para reduzir o tempo de processamento e também para que o sistema saiba qual é o seu significado no sistema de coordenadas do mundo. Isto é qual é a sua posição em relação ao referencial do mundo. A previsão da posição das marcas é efectuada a partir dos últimos dados de posicionamento obtidos pela visão artificial mais os dados recolhidos pela odometria entre a ultima pose obtida pela visão artificial e a ultima pose obtida pela odometria até a chegada da imagem. Obtendo desta forma a posição aproximada de onde o robô se deve encontrar na altura em que foi recolhido o novo *frame*.
Por isso é necessário saber a localização e orientação da câmara no mundo.

$$pos_{cmara} = pos_{robo} - \begin{bmatrix} cos(-\theta + \alpha) & \sin(\theta + \alpha) \\ -\sin(\theta + \alpha) & \cos(\theta + \alpha) \end{bmatrix} \times \begin{bmatrix} x_r \\ y_r \end{bmatrix} \tag{3.80}$$

A partir da pose prevista para a câmara no mundo, é necessário prever onde os quadrados se vão encontrar na imagem de forma a poder proceder a correcção da posição do sistema.

Esta previsão permite saber em que região do plano de imagem pode ser encontrado cada quadrado e o seu significado de cada um destes (localização física) . Esta metodologia permite ao mesmo tempo reduzir o peso do processamento da imagem uma vez que em ver de processar a imagem por completo apenas se pesquisa em áreas onde os quadrados se devem encontrar.

Como as marcas são colocadas fisicamente a uma distancia conhecida, é por com isso possível a partir da pose do câmara obter a localização aproximada destas marcas.

Assume-se que o tecto é paralelo ao plano de imagem e considera-se que em cada pose da câmara se encontram no plano de imagem projectadas a volta do ponto principal quatro marcas. São sobre estas quatro marcas que se pretende prever qual a sua posição no plano de imagem.

Calculando a divisão inteira da posição da câmara no mundo pelo tamanho da rede obtém-se o numero de quadrados que o robô se deslocou em relação ao Referencial do mundo.

$$metros_x = pos_{camara_x} / tamanho_{rede} \tag{3.81}$$

$$metros_y = pos_{camara_y} / tamanho_{rede} \tag{3.82}$$

Depois de efectuada essa divisão é necessário retirar a posição da câmara no mundo a distancia equivalente ao numero de quadrados que o sistema já avançou em relação a origem. Isto para que as marcas que estão projectadas no plano de imagem se encontrem dentro do plano de imagem disponível.

$$reste_x = pos_{camara_x} - metros_x * tamanho_{rede} \tag{3.83}$$

$$reste_y = pos_{camara_y} - metros_y * tamanho_{rede} \tag{3.84}$$

É com estes valores que se vai executar a previsão de onde se encontram os quadrados. Sabemos que no mundo os quadrados que se encontram a volta do robô, $P_0(0,0)$, $P_1(0, tamanho_{rede})$, $P_2(tamanho_{rede}, tamanho_{rede})$ e $P_3(tamanho_{rede}, 0)$ e por isso executa-se a translação destas posições de forma a saber dentro destes quadrados onde se encontra a câmara.

$$P_t = p_n - reste \tag{3.85}$$

$$^{C}P_n = (p_n - T) * R \tag{3.86}$$

De seguida executa-se a rotação que de forma a compensar a rotação da pose da câmara, obtém-se assim a posição dos quadrados no referencial tridimensional da câmara considerando apenas o plano XY.

$$P_c = \begin{bmatrix} cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \times P_t \tag{3.87}$$

Para cada p_c calcula-se a sua projecção no plano de imagem:

$$x_c = f\frac{P_{c_y}}{z_m} + p_x \tag{3.88}$$

$$y_c = f\frac{p_{c_y}}{z_m} + p_y \tag{3.89}$$

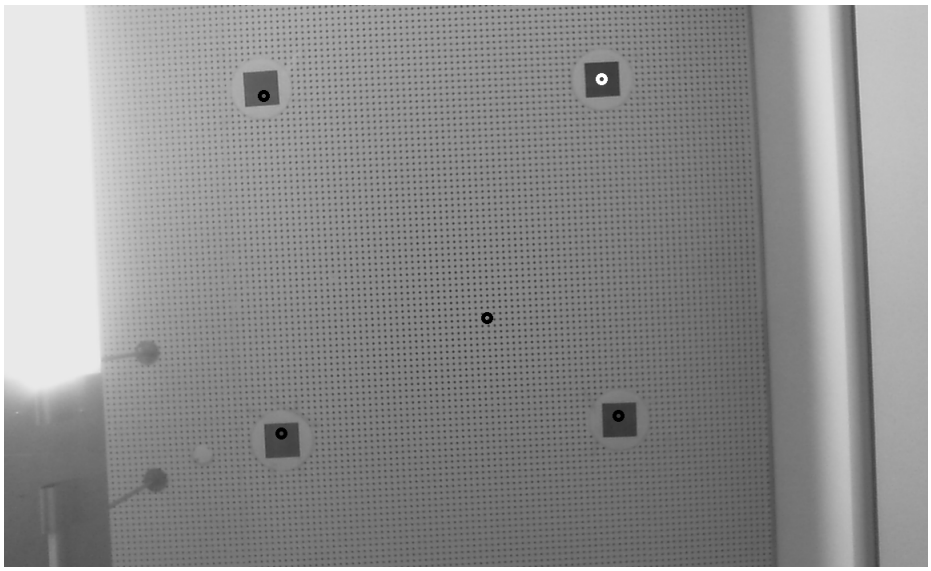


Figura 3.29: Exemplo pratico no plano de imagem da previsão de onde se encontram os quadrados. O quadrado sinalizado a branco corresponde a (0,0) e o ponto no centro da imagem ao ponto principal

3.3.6 Calculo da posição actual

A partir da previsão é possível definir uma área em pixeis para procurar os quadrados correspondentes. Na figura 3.30 encontra-se um exemplo que representa a área de procura de cada um dos quadrados para o qual foi prevista a sua posição no plano de imagem.

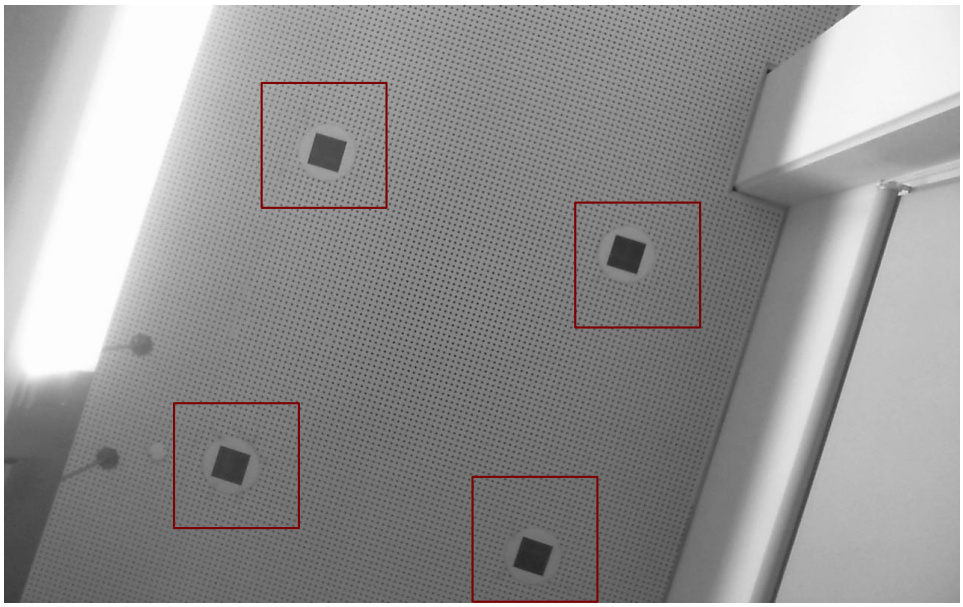


Figura 3.30: A vermelho encontra-se a área do *frame* que é processada para procurar cada um dos quadrados de forma a corrigir a posição

Depois de ser executado o algoritmo de procura de quadrados é devolvida a posição real dos quadrados com esta nova localização no plano de imagem e a partir destes terá de ser efectuada a correcção da posição do robô.

Para calcular a posição actual é necessário recorrer a informação que foi retirada na altura da previsão da posição dos quadrados no plano de imagem. Uma delas é o valor de $metros_x$ $metros_y$ e para cada um dos quadrados que foi prevista a a sua posição no plano de imagem posição correspondem no mundo.

Para se pode calcular a pose da câmara e sucessivamente a pose do robô no mundo é necessário que pelo menos um par de quadrados seja detectado. Sendo que a solução implementada usa quatro pares de quadrados, bastando apenas ser detectado um destes pares para ser possível actualizar a posição do robô. Na figura 3.31 estão representados a diferentes cores os pares de quadrados que são utilizados para calcular a posição da câmara no mundo.

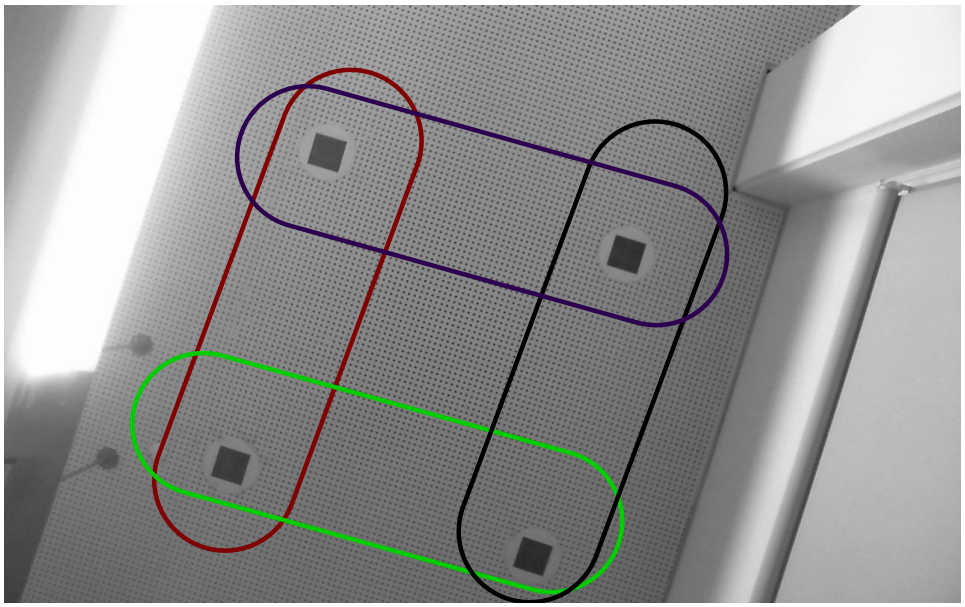


Figura 3.31: Representação de pares de quadrados que são considerados para efeitos de calculo

Na primeira fase é corrido o algoritmo de detecção de quadrados, para cada uma das áreas onde se prevê que os quadrados se encontrem. O algoritmo no caso de os quadrados serem realmente detectados devolve a posição destes no plano de imagem (d) .

Com isto podemos associar os quadrados no mundo com os quadrados no plano de imagem e formar os pares da seguinte forma :

- P_0 e P_1 os quais correspondem a d_0 e d_1
- P_2 e P_1 os quais correspondem a d_2 e d_1
- P_2 e P_3 os quais correspondem a d_2 e d_3
- P_3 e P_0 os quais correspondem a d_3 e d_0

Com cada um destes pares é possível calcular a pose da câmara no mundo. Cada um dos pares representa um calculo diferente uma vez que se encontram em posições físicas diferentes .

Primeiro par

Para o primeiro par (d_0 e d_1) a orientação da câmara no mundo obtém-se calculando o ângulo que entre as duas marcas e o eixo dos xx do plano de imagem:

$$dif_x = d_{0_x} - d_{1_x} \tag{3.90}$$

$$dif_y = d_{0_y} - d_{1_y} \tag{3.91}$$

$$\theta = -atan(\frac{dif_y}{dif_x}) \tag{3.92}$$

De seguida calcula-se a posição relativa dos quadrados em relação a câmara através modelo *pinhole camera* para a marca d_0 que corresponde a marca P_0 na previsão :

$$k_{X_0} = (Z * (d_{0_x} - P_x)/f) \tag{3.93}$$

$$k_{Y_0} = (Z * (d_{0_y} - P_y)/f) \tag{3.94}$$

Uma vez que o ponto principal se encontra na posição $x = 0$ e $y = 0$ a translação de um sistema de coordenadas par o outro é igual ao valor de k_0 que foi calculado anteriormente sobre esse valor efectua-se a rotação de forma a alinhar os dois sistemas de coordenadas.

$$p_r = \begin{bmatrix} cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \times k_0 \tag{3.95}$$

Para que os sistemas de coordenadas obtido fique coerente com o sistema de coordenadas do mundo:

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \times p_r \tag{3.96}$$

O resultado é a posição do ponto principal no sistema de coordenadas representado pelas marcas, dito de outra forma o resultado é a posição da câmara relativamente as marcas.

Por fim adicionam-se a distancia em relação ao referencial do mundo a que a marca P_0 corresponde o qual foi calculado na previsão :

$$x_a = x_c + metrosX * tamanho_{rede} \tag{3.97}$$

$$Y_a = y_c + metrosY * tamanho_{rede} \tag{3.98}$$

É obtida desta forma a pose da câmara para este par de marcas com (x_a, y_a , θ).

Segundo par

Para o segundo par (d_2 e d_1) a orientação da câmara no mundo é calculada através do ângulo que entre as duas marcas:

$$dif_x = d_{2_x} - d_{1_x} \tag{3.99}$$

$$dif_y = d_{2_y} - d_{1_y} \tag{3.100}$$

$$\alpha = -atan(\frac{dif_y}{dif_x}) \tag{3.101}$$

Ao ângulo obtido adiciona-se mais o ângulo $\pi/2$ uma vez que as marcas estão perpendiculares a orientação da câmara.

$$\theta = \alpha + \pi/2 \tag{3.102}$$

De seguida calcula-se a posição relativa dos quadrados em relação a câmara através modelo *pinhole camera* para a marca d_1:

$$k_{X_1} = (Z * (d_{1_x} - P_x)/f) \tag{3.103}$$

$$k_{Y_1} = (Z * (d_{1_y} - P_y)/f) \tag{3.104}$$

Depois efectua-se a translação e a rotação da mesma forma que para o par anterior.

$$p_r = \begin{bmatrix} cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \times k_1 \tag{3.105}$$

Para que os sistemas de coordenadas fiquem coerentes:

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \times p_r \tag{3.106}$$

Por fim adicionam-se a distancia em relação ao referencial do mundo a que a marca P_2 se encontra. Esta é igual ao valor calculado calculado na previsão para o eixo dos xx e tem mais uma deslocação de um tamanho da rede para o eixo dos yy :

$$x_a = x_c + metrosX * tamanho_{rede} \tag{3.107}$$

$$y_a = x_c + (metrosY + 1) * tamanho_{rede} \tag{3.108}$$

É obtida desta forma a pose da câmara para este par de marcas com (x_a, y_a , θ).

Terceiro par

Para o terceiro par d_2 e d_3. Calcula-se a orientação da câmara no mundo calculando o ângulo que entre as duas marcas:

$$dif_x = d_{3_x} - d_{2_x} \tag{3.109}$$

$$dif_y = d_{3_y} - d_{2_y} \tag{3.110}$$

$$\theta = -atan(\frac{dif_y}{dif_x}) \tag{3.111}$$

De seguida calcula-se a posição relativa dos quadrados em relação a câmara através modelo *pinhole camera* para a marca d_3:

$$k_{X_3} = (Z * (d_{3_x} - P_x)/f) \tag{3.112}$$

$$k_{Y_3} = (Z * (d_{3_y} - P_y)/f) \tag{3.113}$$

A translação e a rotação da câmara calcula-se da mesma forma que os anteriores mas relativamente a k_3.

$$p_r = \begin{bmatrix} cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \times k_3 \tag{3.114}$$

Para que os sistemas de coordenadas fiquem coerentes trocam-se os eixos e inverte-se as coordenadas para que este fique alinhado com o eixo de coordenadas do mundo.

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \times p_r \tag{3.115}$$

Por fim adicionam-se a distancia em relação ao referencial do mundo a que a marca P_3 se encontra que tem mais uma deslocação de um tamanho da rede em relação ao calculado na previsão e para o eixo dos xx e para o eixo dos yy é igual a previsão :

$$x_a = x_c + (metrosX + 1) * tamanho_{rede} \tag{3.116}$$

$$y_a = y_c + (metrosY) * tamanho_{rede} \tag{3.117}$$

É obtida desta forma a pose da câmara para este par de marcas com (x_a, y_a , θ).

Quarto par

Para o quarto par d_3 e d_0 calcula-se a orientação da câmara no mundo calculando o ângulo que entre as duas marcas:

$$dif_x = d_{3_x} - d_{0_x} \tag{3.118}$$

$$dif_y = d_{3_y} - d_{0_y} \tag{3.119}$$

$$\alpha = -atan(\frac{dif_y}{dif_x}) \tag{3.120}$$

Ao ângulo obtido adiciona-se mais o ângulo $\pi/2$ uma vez que as marcas estão perpendiculares a orientação da câmara.

$$\theta = \alpha + \pi/2 \tag{3.121}$$

De seguida calcula-se a posição relativa dos quadrados em relação a câmara através modelo *pinhole camera* para a marca d_0:

$$k_{X_0} = (Z * (d_{0_x} - P_x)/f) \tag{3.122}$$

$$k_{Y_0} = (Z * (d_{0_y} - P_y)/f) \tag{3.123}$$

A rotação e translação é efectuada da mesma forma que os casos anteriores mas

$$p_r = \begin{bmatrix} cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \times k_0 \tag{3.124}$$

Para que os sistemas de coordenadas fiquem coerentes trocam-se os eixos e inverte-se as coordenadas.

$$\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \times p_r \tag{3.125}$$

Por fim adicionam-se a distancia em relação ao referencial do mundo a que a marca P_0 se encontra a qual é igual ao valor calculado calculado na previsão para ambos os eixos:

$$x_c = x_a + (metrosX) * tamanho_{rede} \tag{3.126}$$

$$y_c = y_a + (metrosY) * tamanho_{rede} \tag{3.127}$$

É obtida desta forma a pose da câmara para este par de marcas com (x_a, y_a , θ).

Capítulo 4

Controlo de posição

4.1 Introdução

O controlo de posição do robô está divido em duas partes. Uma parte é o controlo de velocidade dos motores que permite com que a cada dado momento seja aplicada a velocidade correcta em cada um dos dois motores de tracção do robô. A outra parte é uma algoritmo que devolve as velocidades a aplicar em cada uma das rodas, a qual depende da localização atual do robô e a posição para onde o robô se pretende deslocar.

Neste capitulo é descrita a teoria no qual se baseia o controlo de velocidade dos motores que foi implementado, bem como o algoritmo que devolve as velocidades a aplicar para controlar a posição do robô, ambos os sistemas são sistemas de controlo em malha fechada.

4.2 Sistema de controlo de velocidade dos motores

Pelo estudo do modelo cinemático do robô sabemos que é possível através do controlo de velocidade dos motores definir para onde o robô se desloca. Foi por isso necessário implementar um sistema que faça o controlo da velocidade das rodas de tracção . Os motores que são utilizados para esta tarefa são motores DC de ímanes permanentes.

4.2.1 Modelo do motor DC de ímanes permanentes

O modelo que descreve o circuito eléctrico e mecânico de um motor DC é apresentado na Figura 4.1.

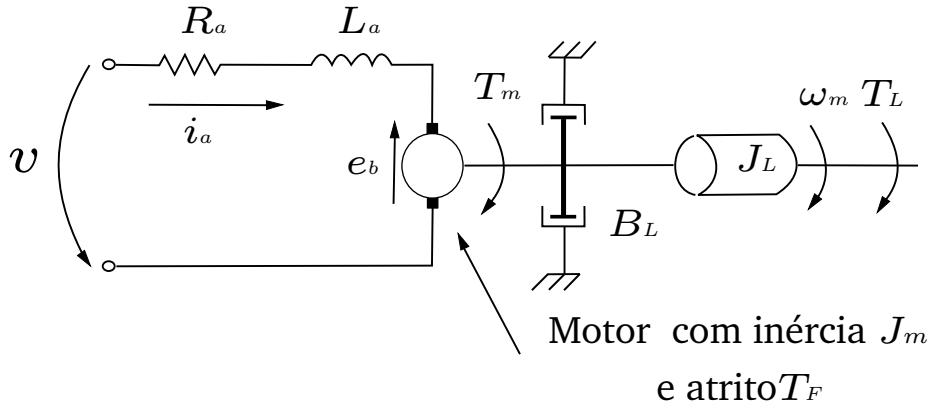


Figura 4.1: Representação do modelo de um motor DC de íman permanente

A parte eléctrica é representada pelos seguintes elementos: v que representa a tensão à entrada dos motores, R_a e L_a que são respectivamente a resistência e a indutância interna do motor, i_a representa a corrente no enrolamento da armadura e e_b a força contra electromotriz gerada pelo motor.

Para a parte mecânica: T_m representa o binário gerado pelo motor, ω_m representa a velocidade angular do motor, T_f é composto pelo binário de refrigeração e o atrito dos rolamentos do motor, J_m é a inércia do motor, J_L é a inércia da carga, B_L o atrito viscoso da carga e T_L o binário de carga.

Este modelo é descrito pelas seguintes equações:

$$\left.\begin{array}{c} v = R_a \times i_a + \frac{di_a}{dt} \times L_a + e_b \\ e_b = K_b \times \omega_m \end{array}\right\} v = R_a \times i_a + \frac{di_a}{dt} \times L_a + K_b \times \omega_m \tag{4.1}$$

$$T_m = k_t i_a \tag{4.2}$$

$$T_m - T_F - B_L \times \omega_m - T_L = (J_m + J_L)\frac{d\omega}{dt} \tag{4.3}$$

Os motores DC por questões físicas e pela forma como são construídos tem necessariamente velocidades de rotação elevadas em relação as velocidades que são aplicadas a este tipo de robôs, quero com isto dizer que a rotação nominal por norma é sempre superior as 1000 r.p.m e por isso tem de se recorrer a engrenagens de forma a baixar a velocidade e aumentar o binário que é aplicado à carga que no caso é o robô e com isto pode executar as trajectórias a velocidades baixas e consequentemente um aumento de capacidade de carga . Na figura 4.2 encontra-se a representação simplificada de uma engrenagem:

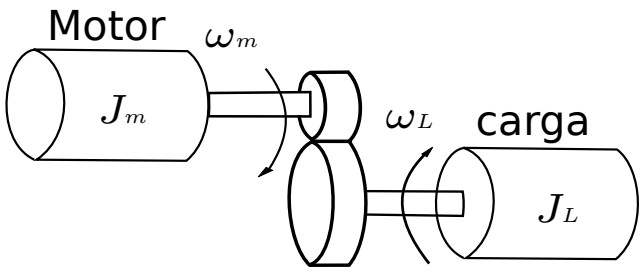


Figura 4.2: Representação de uma engrenagem

As engrenagens tem como principal característica a chamada razão de engrenagem (N), a qual se encontra definida pela seguinte equação.

$$N = \frac{\omega_m}{\omega_L} \tag{4.4}$$

Para calcular a inércia do sistema equivalente sem que houvesse engrenagem, como a energia cinética tem ser igual nos dois sistemas resulta que a Inercia J_{LS} equivalente seria:

$$\frac{1}{2} \times J_{LS} \times \omega_L^2 = \frac{1}{2} \times J_L \times \omega_L^2 \Rightarrow J_{LS} = \frac{1}{N^2} \times J_L \tag{4.5}$$

De modo semelhante conclui-se que o binário equivalente T_{LS}

$$T_{LS} = \frac{1}{N} \times T_L \tag{4.6}$$

4.2.2 Controlador PID de velocidade

Para este tipo de motores há varias metodologias possíveis para a implementação do controlo de velocidade. O que mais se destaca é o PID, no entanto existem outros modelos de controlo. A titulo de exemplo o artigo [19] descreve o comportamento do modelo de controlo do PID e do modelo de controlo fuzzy PI, para o mesmo Hardware. No qual se pode comparar o custo computacional bem como o comportamento das duas metodologias. O que se pode verificar é que o PID tem um custo computacional menor bem como resultados melhores.

Pelas razões referidas a solução implementada baseia-se no controlo com um PID. A grande vantagem deste método de controlo é o facto de não ser necessário o conhecimento do modelo matemático do sistema.

Este controlador é um controlador em malha em fechada e actua em função do erro. O erro consiste no diferencial entre o valor real de saída e o valor ideal ou desejado. Estas características, permite a utilização deste tipo de controlo numa vasta gama de aplicações

bem como uma utilização eficaz, através de um uso intuitivo dos seus parâmetros.

O controlador PID tem três acções de controlo [20] , são elas a proporcional, integrativa e derivativa, a representação matemática de todas estas componentes encontra-se na equação 4.7.

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)dt + K_d \cdot \frac{d}{dt}e(t) \tag{4.7}$$

O diagrama de blocos do controlador PID encontra-se na figura 4.3.

A entrada do sistema é a referencia (no nosso caso a velocidade angular que se pretende para o motor), o erro ($e(t)$) é calculado pela subtracção desta entrada com o valor actual que se verifica na saída, O erro é o valor de entrada para cada uma das componentes do controlador PID e $u(t)$ é a variável aplicada ao processo (no nosso caso a tensão aplicada ao motor). A acção de cada um deste é detalhada na seguintes subsecções:

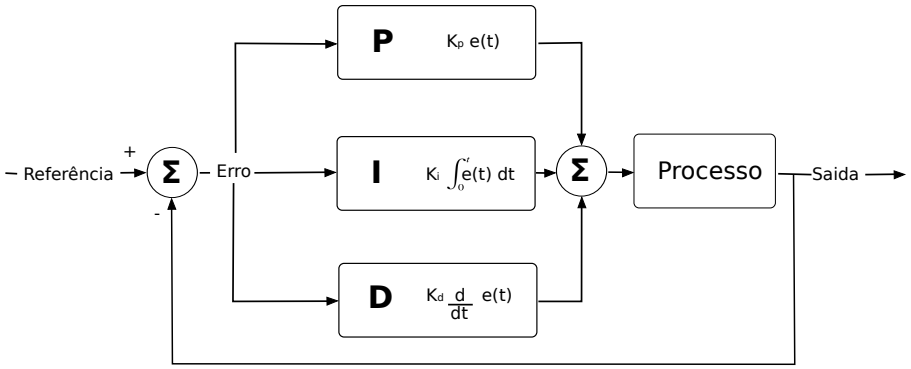


Figura 4.3: Diagrama do Controlador PID

Acção de controlo proporcional

Se o controlador apenas tiver apenas acção proporcional a acção de controlo chamado de um controlador P fica definida como:

$$u(t) = K_p \cdot e(t) \tag{4.8}$$

Onde a constante K_p, é designada por ganho proporcional. Este factor proporcional resulta do produto entre este ganho e o erro de medida. Assim, quanto maior o ganho ou o erro, maior será a saída do factor proporcional. Com isto resulta que colocando um ganho proporcional, demasiado elevado levará o controlador a exceder o valor de referência e pode colocar o sistema em oscilação. A componente proporcional mostra-se insuficiente quando o erro se torna muito pequeno e a saída do controlador se torna diminuta. Assim, quanto maior o ganho proporcional menor será o erro estacionário que não nunca deixa de existir mas

quanto e mais próximo esta do valor de referencia mais perto está de atingir a instabilidade.

Acção de controlo proporcional e integrativa

A acção de controlo de um controlador proporcional e integrativo, conhecido como controlador PI é definido por:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)dt \tag{4.9}$$

A a constante K_i, que faz parte do factor integral funciona como um acumulador de erro. Esta componente aumenta quando o erro é positivo e diminui quando o erro for negativo. A velocidade desta variação depende da constante K_i. Quando o controlador está a desempenhar bem a sua tarefa, o valor de integral deve ser praticamente nulo. Mesmo quando for demasiado pequeno para a componente proporcional, o integral está a acumulá-lo até que seja suficiente para actuar. Assim, uma das tarefas da componente integral é eliminar o erro em regime estacionário. O ponto fraco da componente de integral é que pode contribuir para a sobre-elevação, se o valor de K_i for muito elevado ao se aproximar do valor de referencia. Quanto maior o valor de K_i mais agressivo será o efeito desta componente na supressão de erro e maior será a sobre-elevação. O efeito de K_i é o de eliminar o erro no estado estacionário, mas torna a resposta transitória mais lenta.

Acção de controlo proporcional e derivativa

A acção de controlo de um controlador proporcional e derivativo conhecido por PD é definida por:

$$u(t) = K_p \cdot e(t) + K_d \cdot \frac{d}{dt}e(t) \tag{4.10}$$

A componente derivativa compara o erro actual com o erro da última verificação. Isto é, mede a variação do erro. Quanto maior o ganho derivativo (K_d) ou maior a variação do erro, maior será a componente derivativa. O efeito, da componente derivativa é prever como vai variar o erro e actuar antecipadamente de forma a diminuir o erro futuro. Um efeito pratico é a de contrapor a possível sobre-elevação provocada pela componente proporcional. Uma derivativa bem calibrada possibilita a utilização de uma componente proporcional mais agressiva. Um exemplo de onde este tipo de controlo pode ser usado é no controlo da temperatura de um forno no qual antes de atingir a temperatura de referencia a componente derivativa “desliga“ o forno antes de a temperatura ser atingida, e uma vez que a resistência ainda se encontra a uma temperatura superior ao do forno a temperatura no forno vai continuara a subir. Tendo a componente derivativa é possível aumentar os valores de K_p de forma a diminuir o erro estacionário nas não o elimina.

Acção de controlo proporcional, integrativa e derivativa

A acção de controlo com as três componentes resulta num compromisso entre as vantagens de controlo PI e PD.

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)dt + K_d \cdot \frac{d}{dt}e(t) \tag{4.11}$$

Cada uma destas componentes tem vantagens e desvantagens, as quais se complementam.

O resultado é:

- A componente proporcional terá o efeito de reduzir o tempo de subida e irá reduzir, mas nunca eliminar o erro estacionário.

- A componente integral permite eliminar o erro estacionário, mas atrasa a resposta transitória.

- A componente derivativa permite minimizar a sobre-elevação e permite aumentar o ganho proporcional e com isso permite aumentar a velocidade de resposta do sistema.

Os efeitos de cada um dos controladores de K_p, K_d e K_i são resumidos na tabela 4.1

Tabela 4.1: Comportamento das características de uma resposta face a variação (aumento) dos pararametros PID

Parâmetro	**Tempo de subida**	***Overshoot***	**Tempo de estabelecimento**	**Erro do estado estacionário**
K_p	diminui	aumenta	pequena variação	diminui
K_i	diminui	aumenta	aumenta	elimina
K_d	pequena variação	diminui	diminui	pequena variação

De notar que estas correlações não são exactas uma vez que alterando apenas um parâmetro podemos estar a alterar o efeito das outras acções por esta razão a tabela serve apenas como referencia para a determinação dos parâmetros do controlador.

A definição dos parâmetros é usualmente feita empiricamente (existindo alguns métodos que facilitam a obtenção dos parâmetros), como por exemplo o método Método Ziegler-Nichols.

4.2.3 Controlador discreto implementado

Uma vez que o sistema no qual é implementado o controlo de velocidade é um sistema discreto é necessário descretizar as componentes que constituem o controlador PID. Na forma discreta fica:

$$u(n) = K_p \cdot e(n) + K_i \cdot \sum_{n=0}^{n} e(n)dt + K_d \cdot (e(n) - e(n-1)) \tag{4.12}$$

De forma a melhorar a resposta do controlador do sistema foi adicionado um ramo de feedforward com o modelo aproximado do motor DC.

O modelo utilizado, é um modelo aproximado porque para implementar o modelo completo os custos tanto de processamento bem como do Hardware necessário aumentariam significativamente e o tipo de controlo a ser executado não necessita de tão elevada precisão.

O processo de simplificação do modelo do motor parte das seguintes equações:

$$\left.\begin{array}{c} v = R_a \times i_a + \frac{di_a}{dt} \times L_a + e_b \\ T_m = k_t i_a \end{array}\right\} \tag{4.13}$$

Uma vez que com o sistema implementado não é possível obter uma taxa de amostragem da corrente suficiente elevada é necessário considerar no modelo simplificado que o sistema actua como se não houve-se variação de corrente e por isso obtemos a primeira simplificação da expressão:

$$v = R_a \times \frac{T_m}{k_t} + K_b \times \omega_m \tag{4.14}$$

Para o calculo da tensão para uma dada velocidade, todos os parâmetros são conhecidos, com a excepção do torque. Uma vez que o torque realmente necessário em cada instante de tempo depende da carga a qual é variável o valor do torque definido passa a ser metade do torque nominal do motor.

A representação em diagrama de blocos do sistema implementado encontra-se na figura 4.4:

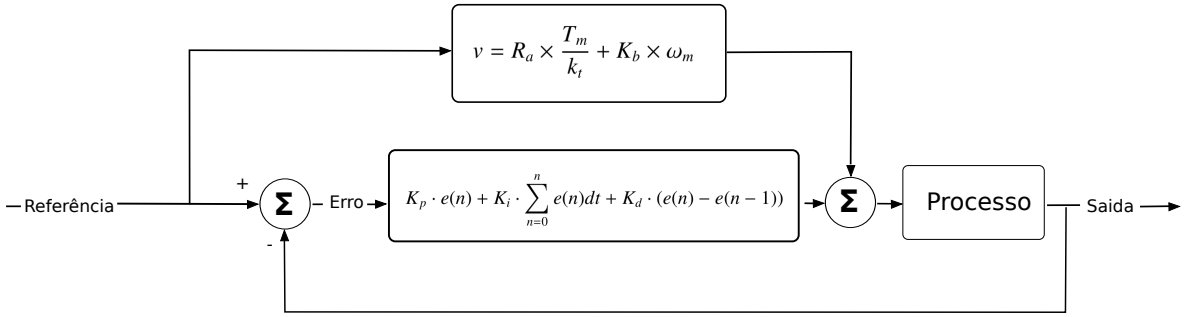


Figura 4.4: Diagrama controlador discreto

Cada motor tem um sistema de controlo independente, a actuação de cada motor faz-se através de PWM (Pulse-width modulation) respectivo e a medição da velocidade é obtida com o auxílio de encoders ópticos.

Pulse-width modulation (PWM)

A modulação por pulso, mais conhecida por PWM (Pulse-width modulation) é uma técnica usada para controlar dispositivos eléctricos. O valor médio da tensão é controlada ligando e desligando a fonte de energia da carga.

A frequência de comutação tem de ser mais rápida do que o tempo de resposta da carga, estas costumam estar a cima de 120 H_z no caso dos motores por volta de $10KH_z$.

O ciclo de funcionamento (duty cycle) é o termo que descreve a proporção do tempo em que a alimentação está "ligada"durante o período de PWM, um duty cycle. Se este for baixo corresponde à uma tensão media baixa, porque a fonte é desligada a maior parte do período. O valor do Duty cycle é definido em percentagem do período de PWM e corresponde a percentagem de tempo em que alimentação está ligada.

O valor médio da forma de onda quadrada é dado por :

$$y = \frac{1}{T}\int_0^T f(t)dt \Rightarrow y = \frac{1}{T}\left(\int_0^{DT} y_{max}dt + \int_{DT}^{T} y_{min}dt\right) \tag{4.15}$$

Onde D representa o duty cycle T o período o Y_{max} o valor ligado e Y_{min} o valor desligado. Na imagem 4.5 encontra-se representada a um exemplo deste tipo de modelação com duty cycles variáveis.

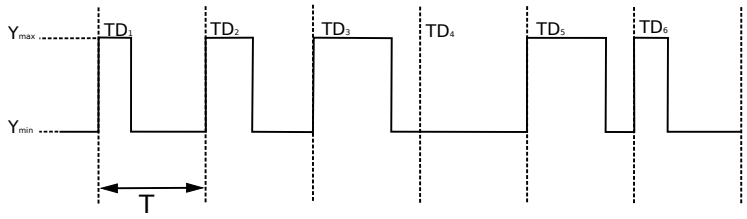


Figura 4.5: Forma de onda gerada por modelação por pulso com duty cycle variável.

Encoder óptico

Um encoder é um dispositivo que mede movimento mecânico (a rotação de um motor, p. ex.) e converte essa informação numa série de pulsos eléctricos. O encoder que é utilizado é um encoder incremental de quadratura de dois canais, em que os dois sinais de saída (canal A e canal B) se encontrão desfasados 90º como se encontra na figura 4.6.

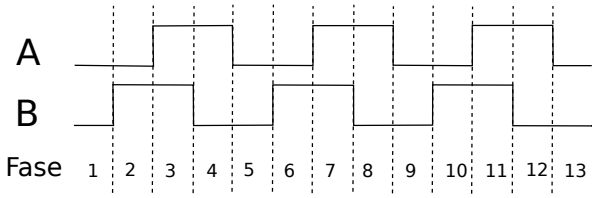


Figura 4.6: Sinais de saída de um encoder de quadratura

Este tipo de encoder permite determinar a posição e da direcção de rotação de um sistema rotacional. Tendo a posição é possível determinar a velocidade de rotação. Como se pode observar na figura 4.7 , existem quatro estados (Phase) possíveis dos canais do encoder num dado momento no tempo. Comparando o estado actual com o seguinte, é possível determinar a direcção.

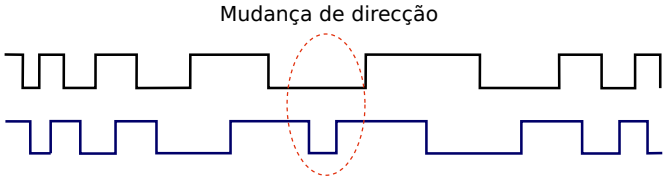


Figura 4.7: Mudança de direcção encoder

Na figura, é possível constatar que o desfasamento de 90º faz com que um dos canais "vá atrás" do outro. Por exemplo, no lado esquerdo da figura o canal B a azul está a seguir o canal A (quando A muda para "1", 90º depois o B muda também para "1" e do mesmo modo, quando A muda para "0", B muda para "0" logo após o desfasamento); No lado direito da

figura, após a mudança de direcção, passa a ser o canal A a seguir o canal B, indicando assim que o sistema está agora a trabalhar na direcção oposta.

De forma a que estes conceitos se traduzam em linguagem lógica opta-se pela estratégia de efectuar as observações na transições de um canal, isto é toma-se como referencia um dos canais e observando-se a cada transição no canal A o estado do canal B, no qual resulta a tabela de verdade 4.2 que nos define a direcção em que se dá o movimento.

Tabela 4.2: Tabela de verdade do encoder usando lógica

Estado Actual		Proximo estado	
A	B	A⊗B	Direcção
0	0	0	←
0	1	1	→
1	1	0	←
1	0	1	→

A velocidade de rotação é obtida a partir da equação 4.16 onde RE_{encode} é o resolução do encoder, por outras palavras é o numero de tiques que o encoder tem por rotação completa, ΔN_{tiks} o numero de tiques num período, N a relação da caixa e T o período.

$$r.p.m = \frac{\Delta N_{tiks}}{2 * RE_{encoder} \times N \times T} \times 60 \tag{4.16}$$

Para a determinação da Velocidade a partir da informação fornecida pelo encoder podem-se usar duas estratégias:

- Com um temporizador contar o tempo entre duas transições de um dos canais. Esta é a solução adequada para encoders de baixa resolução e para baixas velocidades. Sendo assim o valor N_{tiks} passa a ser igual a um e o que se mede é o valor do período
- Verificar periodicamente qual o numero de incrementos que ocorreram num determinado período de tempo. Está é a solução adequada para encoders com boa resolução e para altas velocidades. Onde o Valor do período T um valor conhecido e o ΔN_{tiks} é o valor a verificar.

4.3 Controlo cinemático da posição

O controlo cinemático da posição é um controlo proporcional em malha fechada. Por outras palavras e neste caso, as velocidades de controlo, são re-calculadas de cada vez que chega uma nova informação relativa a pose do robô.

O tipo de controlo implementado, assenta no calculo de num vector de erro [15], o qual se calcula tendo em conta a pose actual e a pose de destino. Este vector de erro permite o calculo das velocidades de controlo que são necessárias aplicar a cada instante de forma a que o robô chegue ao seu destino. As velocidades aplicadas fazem com que o vector de erro vá diminuindo a cada nova recepção de dados relativos a pose actual do robô levando o vector de erro para valores próximos de zero e consequentemente para a pose de destino. Quando o vector erro se aproxima de zero chega-se a posição pretendida com orientação paralela ao eixo dos xx.

Este conceito é explicado em detalhe nas subsecções seguintes bem como a forma de garantir orientações finais sem estarem alinhas com o eixo dos xx. Pois na fase de definição do vector erro e da lei de controlo a orientação final para onde o robô fica orientado é sempre paralelo ao eixo do xx .

4.3.1 Definição do vector de erro

O vector erro como já foi referido resulta da pose actual do robô $(x_{actual}, y_{actual}, \theta_{actual})$ e da posição posição para onde se pretende que o robô se dirija (x_{final}, y_{final}). Para calcular este vector recorre-se a uma transformação de coordenadas, para condenadas polares centradas no ponto de destino.

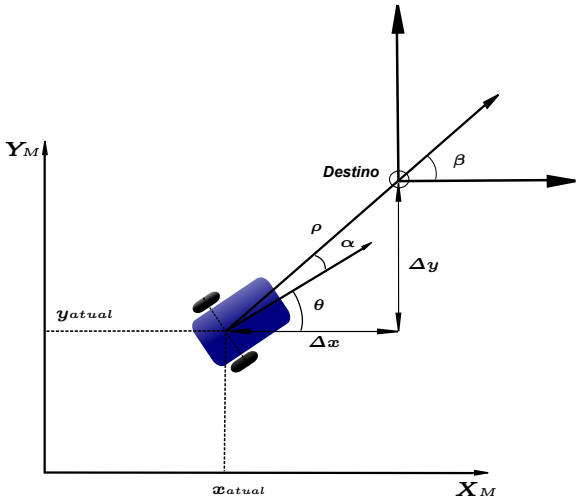


Figura 4.8: Vector erro.

Onde o vector de erro é definido pelas seguintes variáveis:

$$\begin{bmatrix} \rho \\ \alpha \\ \beta \end{bmatrix} \tag{4.17}$$

Estas variáveis podem ser facilmente descritas:

- ρ define a distancia linear ente a posição actual e a que se pretende atingir.

- α define em que direcção se encontra a posição destino em relação a orientação actual do robô.

- β representa o ângulo definido pelo segmento de recta entre a posição actual do robô e a posição de destino, relativamente ao eixo dos xx do referencial do mundo.

O calculo deste vector é obtido pelas seguintes formulas:

$$\Delta X = x_{final} - x_{actual} \tag{4.18}$$

$$\Delta Y = y_{final} - y_{actual} \tag{4.19}$$

$$\rho = \sqrt{\Delta X^2 + \Delta Y^2} \tag{4.20}$$

$$\alpha = -\theta_{actual} + \arctan(\frac{\Delta Y}{\Delta X}) \tag{4.21}$$

$$\beta = -\theta_{actual} - \alpha \tag{4.22}$$

A transformação efectuada não tem um valor definido para $x = y = 0$ por isso o sistema tem de para antes de chegar ao local de destino.

Para que o sistema não chegue a este estado a metodologia a utilizar para garantir isso não ocorre, é definir um raio de aceitação para a posição de destino como valida. Quando o robô entra dentro deste raio, isto é quando o valor de ρ tem um valor inferior ao raio definido o robô chega ao seu destino. Este deve ser definido em função da resolução do sistema de localização e o tempo de amostragem do sistema.

A descrição do sistema, no novo sistema de coordenadas é representado pelo seguinte sistema:

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ -\frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \times \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4.23}$$

4.3.2 Lei de controlo linear

A lei de controlo linear, é uma lei que actua da forma proporcional ao erro e é definida pelas seguintes equações [15] :

$$v = K_\rho \times \rho \tag{4.24}$$

$$\omega = K_\alpha \times \alpha + k_\beta \times \beta \tag{4.25}$$

Onde v e ω é respectivamente a velocidade linear e a velocidade angular a ser aplicada ao robô. As constantes de proporcionalidade actuam relativamente a uma determinada componente do vector erro:

- K_ρ - define que a velocidade linear do robô é proporcional em K_ρ da distancia a que se encontra do destino.

- K_α - define que a velocidade angular é proporcional (em K_α) ao ângulo α que é a diferença ente a orientação actual do robô e a orientação em que se encontra o local de destino.

- K_β - é obrigatoriamente menor em modulo que K_α e define à constante de proporcionalidade da velocidade angular que é proporcional em K_β do ângulo β que aponta para o ponto de destino.

O sistema realimentado fica então:

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -K_\rho \times \rho \times \cos\alpha \\ K_\rho \times \sin\alpha - K_\alpha \times \alpha - k_\beta \times \beta \\ -K_\rho \times \sin\alpha \end{bmatrix} \tag{4.26}$$

Este sistema é exponencialmente estável se forem garantidas as seguintes condições:

$$K_\rho > 0 \tag{4.27}$$

$$K_\beta < 0 \tag{4.28}$$

$$K_\alpha - K_\beta > 0 \tag{4.29}$$

Para garantir que o sistema não muda de direcção na aproximação é necessário garantir a seguinte condição:

$$K_\alpha + \frac{5}{3} \times K_\beta - \frac{2}{\pi} K_\rho > 0 \tag{4.30}$$

4.3.3 Matriz de rotação intermédia

A lei de controlo implementada faz com que os valores de ρ α e β tendam para zero o que faz com que a orientação do robô no destino seja próxima de zero. Para resolver esta limitação foi implementada uma transformação para outro sistema de coordenadas (intermédio) de forma a flexibilizar a utilização deste tipo de controlo. É portanto feita a transformação do valor da posição actual e da posição final. Esta transformação não é mais que uma matriz de rotação do sistema de eixos de condenadas original (M) para um (R) em que a orientação deste novo referencial esteja alinhada com a orientação final pretendida. Podemos desta forma garantir que o robô atinja qualquer orientação final para o robô.

A transformação tem de ser executada da seguinte forma:
- calcular a posição do ponto de chegada no novo referencial

$$\begin{bmatrix} {}^R x_{final} \\ {}^R x_{final} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{final}) & -\sin(\theta_{final}) \\ \sin(\theta_{final}) & \cos(\theta_{final}) \end{bmatrix} \times \begin{bmatrix} {}^M x_{final} \\ {}^M x_{final} \end{bmatrix} \tag{4.31}$$

-calcular a cada nova informação da pose actual a pose no referencial (R):

$$\begin{bmatrix} {}^R x_{actual} \\ {}^R x_{actual} \end{bmatrix} = \begin{bmatrix} \cos({}^M\theta_{final}) & -\sin({}^M\theta_{final}) \\ \sin({}^M\theta_{final}) & \cos({}^M\theta_{final}) \end{bmatrix} \times \begin{bmatrix} {}^M x_{actual} \\ {}^M x_{actual} \end{bmatrix} \tag{4.32}$$

Esta novas posições são as que entram no calculo do vector de erro:

$$\Delta X = {}^R x_{final} - {}^R x_{actual} \tag{4.33}$$

$$\Delta Y = {}^R y_{final} - {}^R y_{actual} \tag{4.34}$$

O valor da orientação no novo referencial é dado por:

$${}^R\theta_{actual} = {}^M\theta_{actual} - {}^M\theta_{final} \tag{4.35}$$

$$\rho = \sqrt{\Delta X^2 + \Delta Y^2} \tag{4.36}$$

$$\alpha = -\,^{R}\theta_{actual} + \arctan(\frac{\Delta Y}{\Delta X}) \tag{4.37}$$

$$\beta = -\,^{R}\theta_{actual} - \alpha \tag{4.38}$$

Tudo o resto do controlo mantêm-se inalterado:

Esta transformação intermédia pode ser facilmente percebida na seguinte figura:

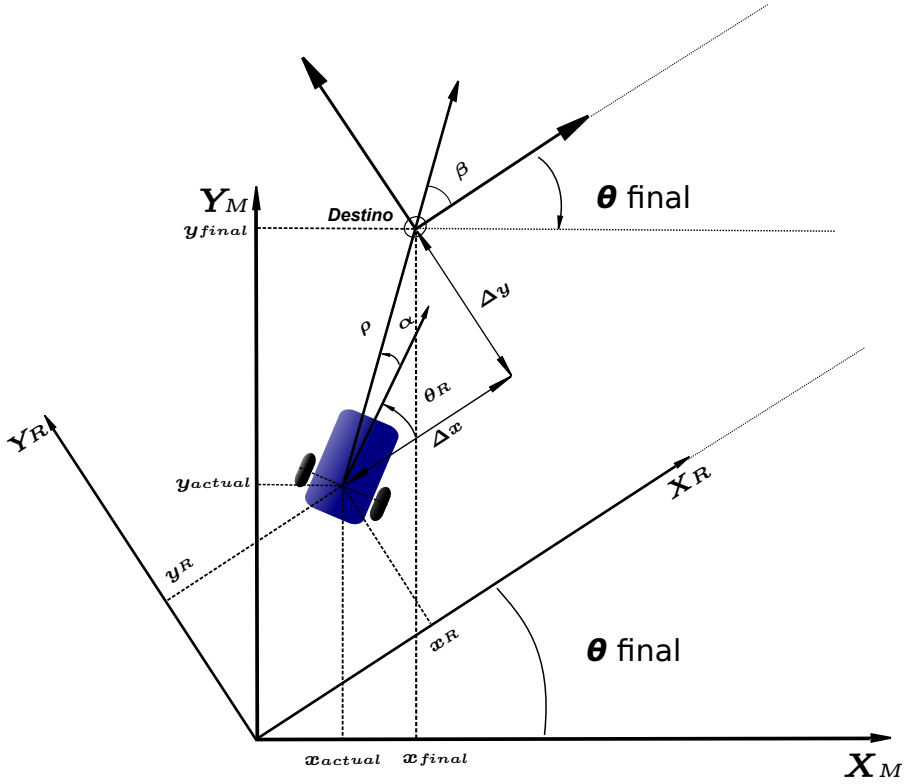


Figura 4.9: Representação da matriz intermédia

4.3.4 Comportamento do controlo cinemático

A escolha do valor das constantes de proporcionalidade geram diferentes resultados. Estes resultados são previsíveis e podem ser calibrados de forma a podermos definir diferentes de tipos de rotas.

Os resultados apresentados nesta secção resultam da simulação do controlo cinemático realizado em Matlab.

Esta simulação pode ser descrita em em três blocos essenciais:

- Modelo cinemático do robô- O modelo cinemático descrito pelo sistema de equações (3.1) recebe a velocidade linear e a velocidade angular e devolve a nova pose do robô.

- Controlo cinemático - O controlo cinemático descrito anteriormente recebe a pose de destino a pose actual e devolve a velocidade linear e angular que deve ser aplicada ao robô.

- Introdução de erro - A introdução do erro é aplicada à velocidade linear e à velocidade angular. Este é gerado pelo matlab, que devolve um valor aleatório entre -0.5 e 0.5, o qual segue uma distribuição gaussiana. O valor gerado é multiplicado por um factor de escala e é adicionado de forma proporcional às velocidades geradas. Esta metodologia de adição de erro resulta em que quanto maior for a deslocação do robô maior é o erro, mas a proporção de (erro/velocidade) permanece constante. Este erro é introduzido para simular os erros que resultam da acção dos actuadores (no caso os motores), uma vez que por melhor que seja o controlo de velocidade nunca é perfeito, bem como os que resultam das imperfeições do meio envolvente em que o robô se desloca (como por exemplo o pavimento ser irregular) e por isso estes erros têm de ser tomados em conta.

O modo como estes blocos se encontram interligados está descrito na seguinte figura:

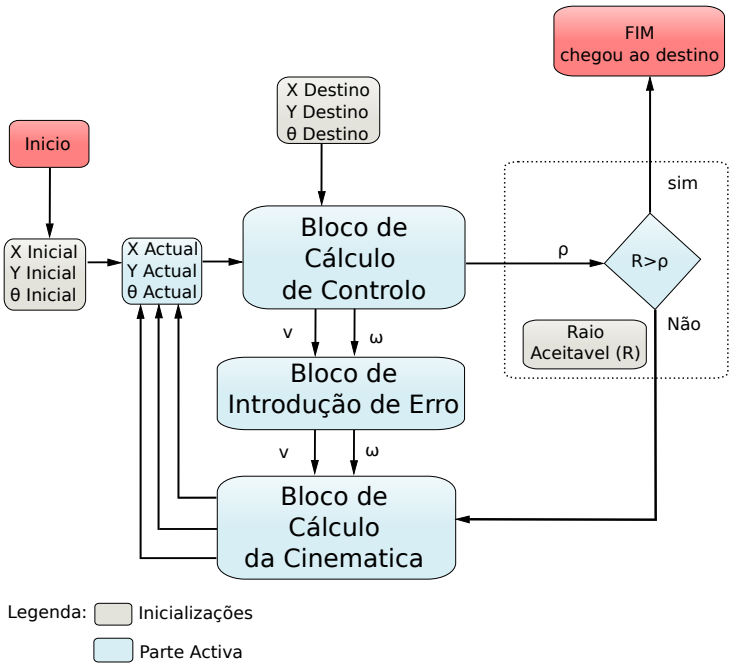


Figura 4.10: Fluxo de programa da simulação do controlo

Modo de funcionamento:

O sistema é iniciado com os seguintes dados: a pose inicial, a pose de destino e o raio (R) que define o círculo de aceitação. Este circulo de aceitação, cujo o centro é a pose de destino, tem como função assinalar que o robô atingiu a pose de destino.

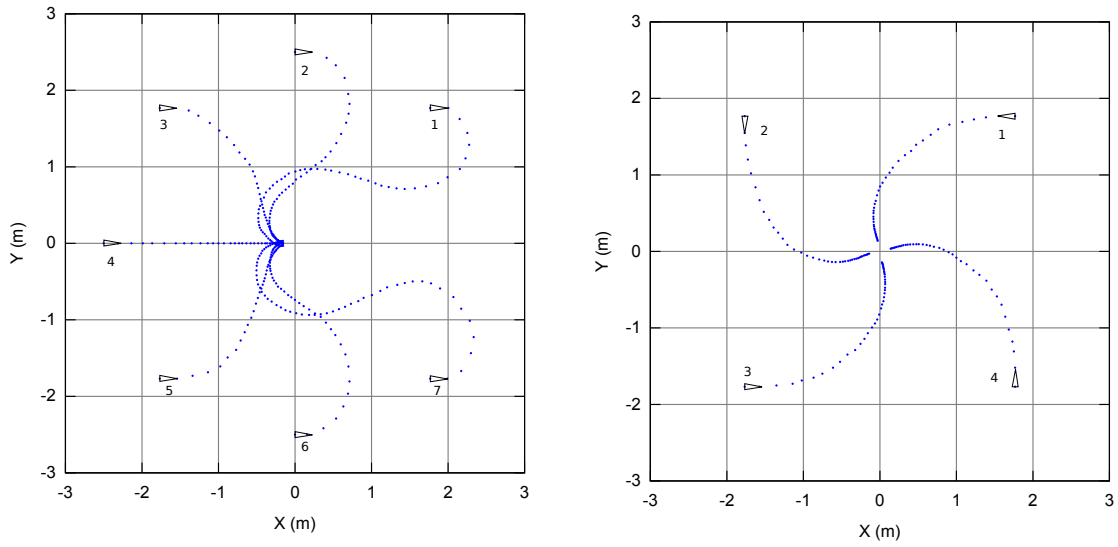
Na inicialização da simulação a pose inicial passa a ser a primeira pose actual, o bloco de controlo efectua o cálculo das velocidades de controlo e o valor de ρ a partir da pose actual e da pose de destino. O valor de ρ que define a distância linear entre a pose actual e a de destino é depois comparada com o valor de R. Caso o ρ seja maior do que R, é adicionado erro às velocidades, a partir da cinemática é calculada a nova pose actual do robô e repete-se o ciclo.

Este ciclo termina quando o ρ é menor que o valor de R, o que significa que o robô está dentro do circulo de aceitação e por isso chegou ao destino.

Segue-se em seguida um pequeno resumo das simulações realizadas com foco para as mais relevantes, em todas elas a posição inicial encontra-se num raio de dois metros e cin-

quenta centímetros e a final é para todos os casos a posição em que $X = 0$ e $Y = 0$ o que varia nas varias simulações é o orientação inicial e final:

- Exemplo 1 : $K_\rho = 0.3$ $K_\alpha = 0.8$ $K_\beta = -0.325$ $R = 0.15$;



(a) (b)

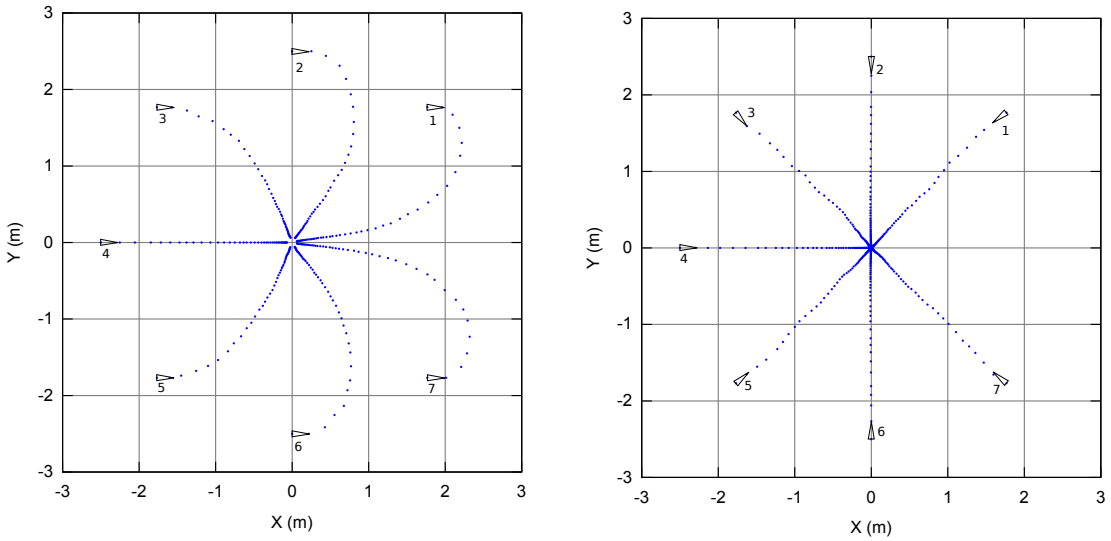
Figura 4.11: Simulação controlo Ex:1

Para a figura 4.11(a) todas as orientações iniciais e finais tem um $\theta = 0°$ o que se pretende demonstrar é que independentemente da localização o percurso é sempre possível, o destino é sempre atingido e a orientação final fica aproximado ao valor pretendido no caso 0º.

Para a figura 4.11(b) as orientações finais e iniciais aqui variam. Sendo que a orientação final se encontra sempre com mais 90º que a orientação inicial. O que se pretende demonstrar com este gráfico são varias conclusões e que dizem respeito aos dois gráficos. Em primeiro lugar e no que diz respeito ao raio de aceitação que define que a posição de destino que é considerada valida como sendo a posição pretendida esta encontra-se sempre a uma distancia linear de 15 cm do valor ideal. Valor o qual não pode ser diminuído, isto porque se se optar por diminuir este raio as orientações finais ficariam longe do valor ideal. Conclui-se portanto que neste caso que tem haver um compromisso que permita que a posição final seja aceitável numa maior área de forma a que a orientação final esteja o mais próxima do ideal. Em segundo lugar e exemplificando com a rota numero 2 da figura 4.11(b) em que a orientação final deveria ser de 0º, e é facilmente perceptível que o valor das constantes K_α e K_β não são os ideais para este tipo de rota enquanto por exemplo que em qualquer rota do gráfico

da figura a) o valor da orientação final é mais aproximado do pretendido em qualquer um dos casos. Estes resultados dão a entender que para cada tipo de rota existe um conjunto de valores das constantes as quais dariam uma rota "ideal" no que toca a orientação final .

- Exemplo 2 : Para $K_\rho = 0.3$, $K_\alpha = 1.2$, $K_\beta = 0.01$ e para a figura a) $R = 0.07$ b) $R = 0.01$



(a) (b)

Figura 4.12: Simulação controlo Ex:2

Para este exemplo o valor das constantes K_α e K_β foram modificadas de forma a que a constante K_α fosse muito mais relevante do que K_β

Para a figura 4.12(a) todas as orientações iniciais e finais tem um $\theta = 0^o$. A primeira coisa que salta a vista para estas constantes de controlo é que o requisito de orientação na posição de destino, foi completamente "negligenciado" mas em compensação o raio de aceitação foi diminuído para 7 cm o que resulta em que a posição de destino é bastante mais aproximada do valor ideal.

Para a figura 4.12(b) a orientação final e inicial coincidem e a posição inicial e a posição de destino encontram-se alinhadas. Pode-se observar que para este caso em particular a posição de destino é atingida num raio de 1 cm e a orientação final é também satisfeita. O que se pretende transmitir com este segundo gráfico é que através de uma combinação de duas rotas uma com ks de controlo presentes no primeiro exemplo e fazendo a aproximação final que esteja de acordo com o apresentado neste gráfico da figura 4.12(b) é possível atingir posições com um raio de aceitação baixo e uma orientação final bastante aproximada com a ideal.

Apesar de não estar implementado nenhum algoritmo de planeamento de rota nem de optimização de constantes de controlo, com esta implementação já é possível gerar trajectórias satisfatórias com os valores para as constantes de controlo anteriormente descritas. Desde que os conceitos anteriormente descritos sejam considerados na altura em que são definidos os pontos por onde o robô deve passar.

Capítulo 5

Software

Neste capitulo apresenta-se traços gerais o software desenvolvido, onde irá ser dado destaque à questão dos controladores de velocidade dos motores e a camada de processamento de nível superior .

5.1 Descrição geral

No que respeita a software, como já foi referido na arquitectura do robô é distribuído em duas camadas. Uma das camadas é o processamento de alto nível que é executado no portátil ou pandabord, e a outra camada é processada pelo micro controlador da placa de desenvolvimento. A comunicação entre estes dois dispositivos é efectuada através de comunicação série. Cada uma destas camadas de software é responsável por diferentes tarefas mas complementares. O resumo das tarefas encontram-se nos seguintes pontos.

O micro controlador é responsável pelas seguintes operações :

- Controlo de velocidade
- Calculo da odometria
- Verificar níveis de bateria
- Detectar uma Emergência (botão de emergência)

A camada de processamento de alto nível é responsável por:

- Processamento de imagem
- Calculo da posição
- Previsão da posição actual

- Controlo de posição

- Executar trajectórias

Estas camadas encontra-se descritas nas seguintes secções. Bem como a comunicação entre estas.

5.2 Camada de baixo nivel

Esta camada corresponde à camada implementada no micro controlador. O software está representado por três sub-camadas (figura 5.1) que são Hardware, Drivers e camada de Aplicação.

Na sub-camada de hardware estão representados os periféricos do micro controlador que foram utilizados para a implementação do sistema. Na sub-camada de software designada como drivers foi implementada uma biblioteca que fornece uma API a camada de aplicação, para configurar o hardware e a qual fornece os dados dos sensores e implementa a comunicação por porta serie. A Camada de aplicação é composta por duas sub-camadas, uma camada que funciona como biblioteca, e uma camada que é responsável pelo de fluxo programa e que recorre a interface de comunicações.

A camada de aplicação contem as funções necessárias para executar cálculos como a velocidade dos motores, os cálculos de saída do PID, os cálculos da odometria e também contem a definição do protocolo de comunicação utilizado pela porta serie.

Cada driver têm uma API que pode ser chamada pela camada de aplicação. Os drivers são dependentes em exclusividade do hardware colocado a baixo do respectivo bloco.

A camada responsável pelo fluxo de programa e de interface de comunicações tem dois componentes, a inicialização que inicializa o sistema e para isso recorre a camada de software dos drivers para inicializar o hardware necessário. O bloco de Aplicação, recebe comandos através de porta serie e executa as varias funcionalidades do sistema. As funcionalidades como são directamente dependentes do protocolo de comunicações são mais facilmente percepcionadas na descrição do protocolo de comunicações.

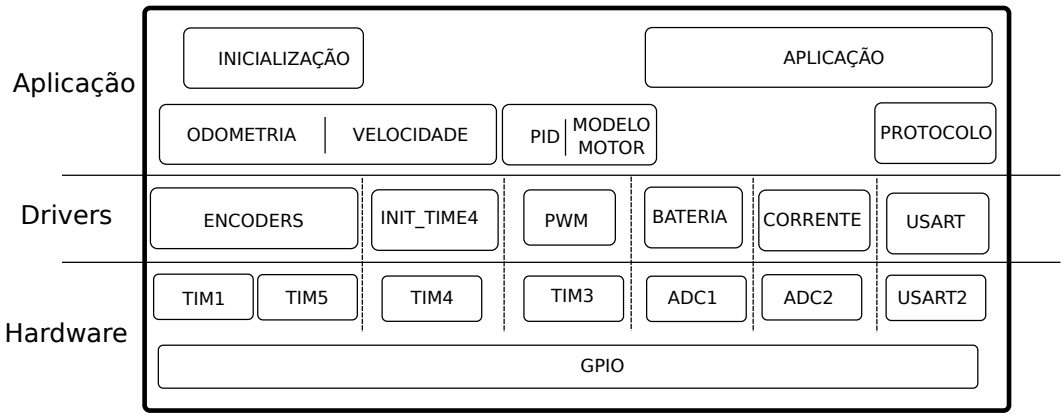


Figura 5.1: Camadas de software do micro controlador

Na figura 5.2 encontra-se representado o fluxo de informação entre os vários componentes e a forma como esta é partilhada:

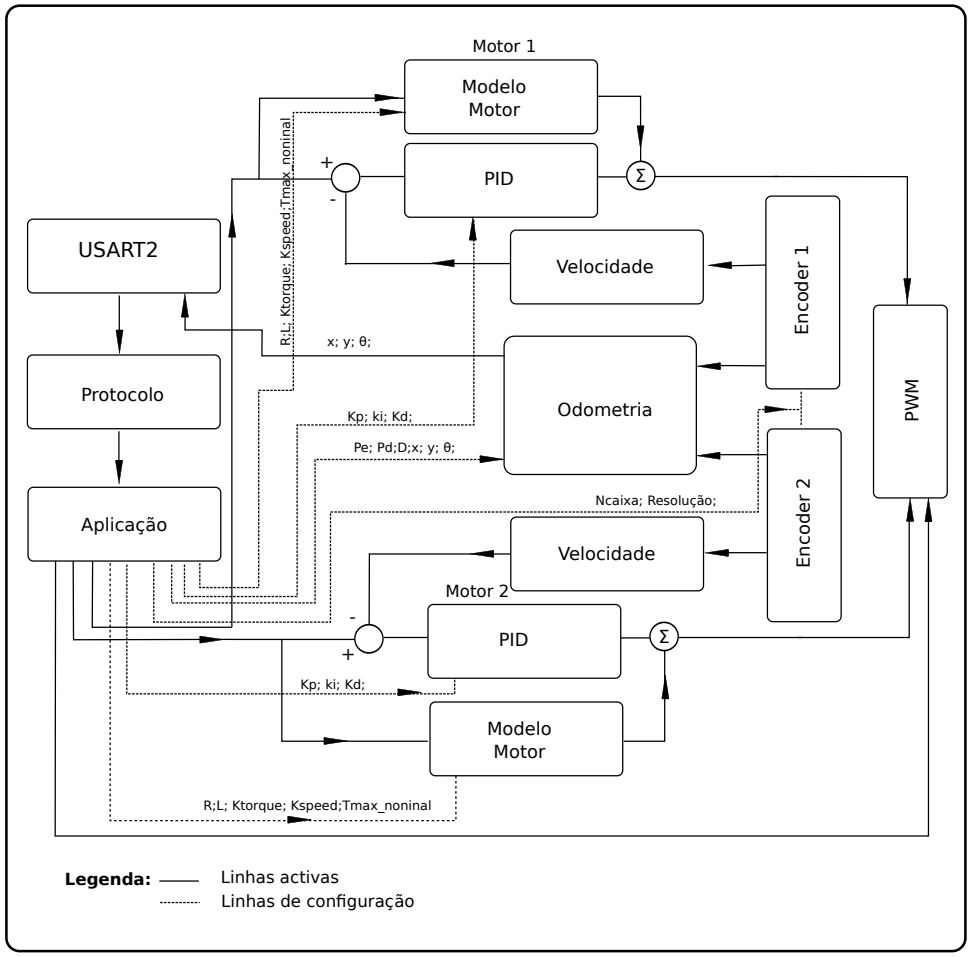


Figura 5.2: Representação do fluxo de informação

Todos os comandos são recebidos pela USART. Na camada de Aplicação há dois tipos de funcionalidade que as mensagens têm. Uma é a de configuração do hardware, represen-

tadas pelas linhas a tracejado, e a outra representada pelas linhas contínuas que representam o fluxo de informação do sistema em funcionamento. Inicialmente é necessário executar as funções de configuração para definir a calibração dos PIDs, a calibração da odometria, do modelo dos motores e a resolução dos ecoderes. Só depois se pode enviar informações como velocidade de controlo dos motores ou mandar o valor de PWM directo para os motores bem como a de receber por porta serie os valores de odometria. As funcionalidades Como o controlo por PID só estão disponíveis se todas as configurações forem realizadas correctamente. Por isso a camada de aplicação tem uma maquina de estados que evita a má utilização e configuração por parte do utilizador.

A comunicação é efectuada por RS-232 para tal foi definido um protocolo de comunicação em que a sintaxe é a seguinte:

$tipoA/B/C ... ;

Em que:

- $ - É o caractere que define inicio de comunicação.
- tipo - É a sigla que define o tipo de mensagem. Esta sigla é composta por quatro caracteres
- A, B, C - São os blocos de dados da mensagem.
- / - É o separador entre blocos de dados.
- ; - É o caractere que define o fim de uma mensagem.

Os tipos de mensagens definidos encontram-se descritos na tabela 5.1:

Tabela 5.1: Tipos de mensagens do protocolo implementado

Sigla	Função
ping	Serve para verificar se há ligação e responde com ($RECEBIDO;)
XY0_	Actualiza a posição da odometria
velo	Define a velocidade pretendida para os motores em r.p.m
powe	Activa/Desactiva o sistema. E também o Timer responsável pelo PWM dos motores
star	Activa/Desactiva o sistema de controlo de velocidade
conf	Verifica e Activa/desactiva as configurações efectuadas devolvendo informação sobre as que ainda não se encontram configuradas
pwm_	Permite controlar os motores com valores directos de PWM
pid_	Configura os valores das constantes de PID para cada motor
moto	Configura os paramentos físicos de cada um dos motores
enco	Configura as características do encoder e caixa de cada motor
bat_	Devolve a tensão das baterias em mV
odom	Configura as constantes necessárias para o calculo da odometria
odoc	Pedido de numero de tikes de cada encoder

Para que o sistema funcione correctamente é necessário proceder a configuração do sistema e só após a configuração se pode efectuar o correcto controlo e aquisição de dados fornecido pelo sistema.

De uma forma geral o sistema tem modos de funcionamento:

- Controlo de motores por PWM.
- Controlo de motores em velocidade.
- Aquisição da pose por odometria.

Na seguinte tabela encontram-se a ordem de configuração para três modos de funcionamento do sistema, sendo o ultimo modo o utilizado para o funcionamento correcto do robô :

Tabela 5.2: Ordem de configuração do hardware da camada de baixo nível

comandos	Funcionamento em modo de PWM	Funcionamento em controlo de velocidade	Funcionamento em controlo de velocidade e odometria
1	powe	pid_	pid_
2	pwm_	moto	moto
3		enco	enco
4		conf	conf
5		powe	odom
6		star	powe
7		velo	star
8			velo

5.3 Camada de Alto nivel

A camada de alto nível foi implementada na linguagem C, o tratamento de imagem foi implementado com recurso a biblioteca de visão *OpenCV* e a biblioteca *video4linux* para a aquisição de imagem (ambas as bibliotecas são opensource).

O software de alto nível é composto por vários componentes, os quais estão representados na figura 5.3

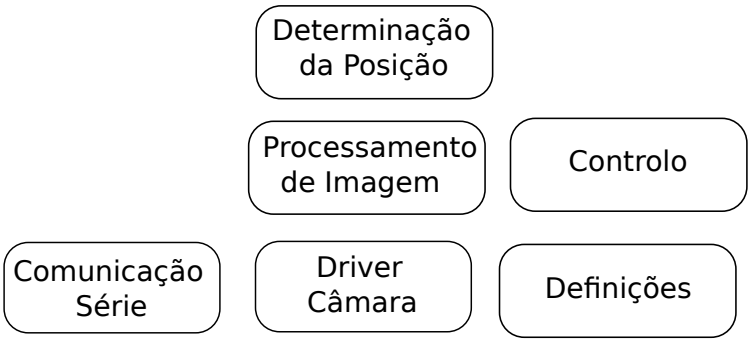


Figura 5.3: Componentes do software de alto nível

O bloco de comunicação série é responsável por a comunicação entre a camada de alto nível e a camada de baixo nível descrita anteriormente. O bloco driver da câmara que foi implementado com recurso a biblioteca de opensorce *video4linux* e é responsável pela aquisição das imagens da câmara. O bloco de processamento de imagem é responsável por devolver a posição dos quadrados detectados no plano de imagem. O bloco Determinação da posição é responsável pelo calculo da posição bem como da previsão da posição dos quadrados no plano de imagem. O bloco definições é responsável por carregar as configurações de controle dos motores, as definições do tamanho da rede de quadrados bem como os parâmetros intrínsecos da câmara utilizada. O bloco de controlo é responsável pelo controlo de posição do robô.

A inicialização do sistema segue o esquema de figura 5.4:

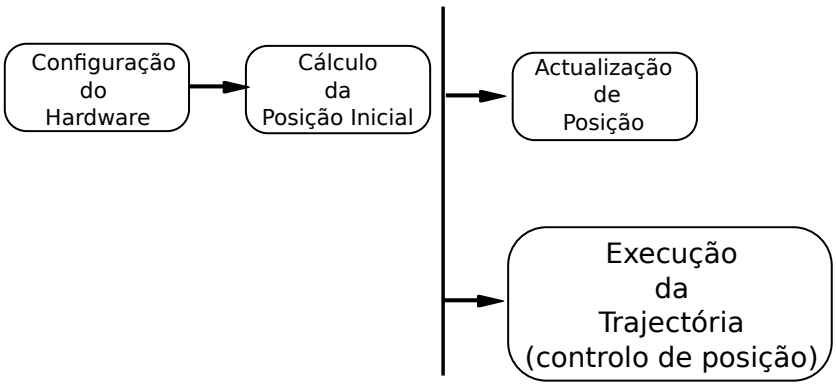


Figura 5.4: Inicialização do robô

O sistema começa por enviar as configurações do hardware da camada de baixo nível com os valores de calibração dos PIDs, parâmetros dos motores e de odometria. Configura as características que definem as dimensões da rede colocada no ambiente bem como os

parâmetros intrínsecos da câmara . Após todo o sistema se encontrar configurado passa a calcular a pose inicial do robô. De seguida passa a processar em paralelo a actualização da posição do robô e a execução das trajectórias. Os dados da posição são fornecidos ao sistema de controlo de posição e os valores de controlo são recalculados e enviados para a camada de software de baixo nível.

O calculo da posição inicial de uma forma mais detalhada segue o seguinte esquema de processamento apresentado na figura 5.5 :

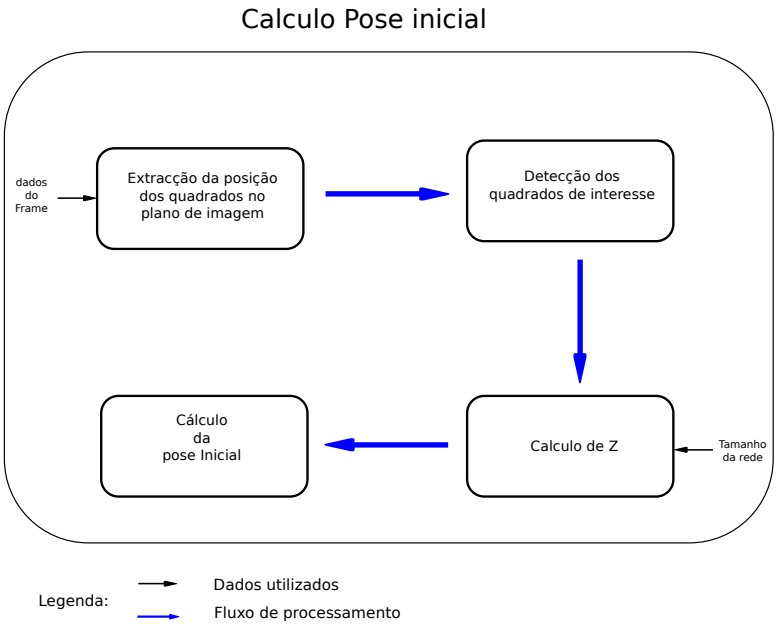


Figura 5.5: Posição inicial

A actualização da pose do robô é feita a cada recepção de um novo frame e o esquema de processamento encontra-se representado na figura 5.6 :

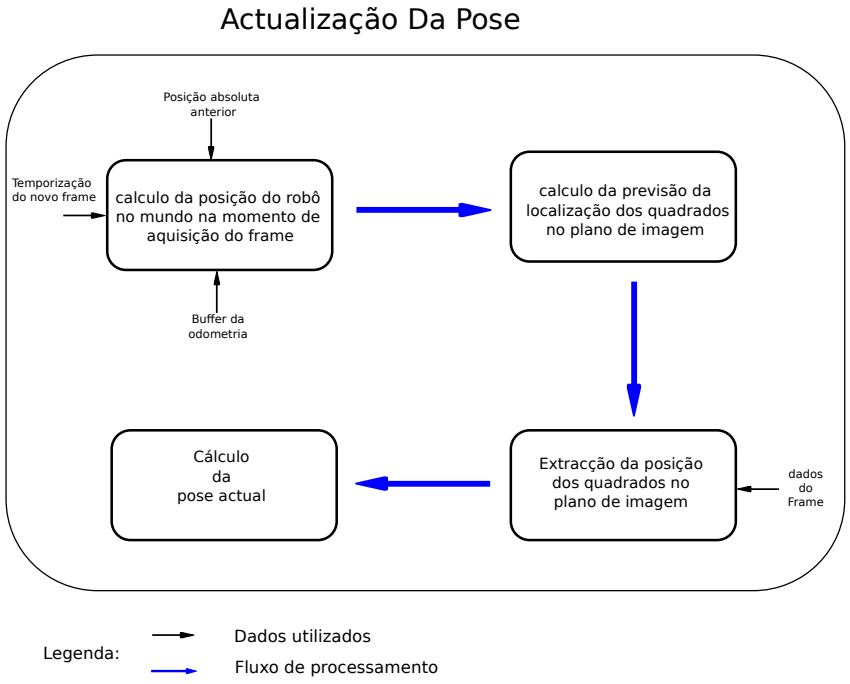


Figura 5.6: Actualização da pose do Robô

A actualização da pose do robô é feita quando chega uma nova imagem e recorre a posição anterior obtida pela visão ao qual são adicionados os dados da odometria para pre-

ver a posição onde se encontra o robô no momento em que foi recolhida a nova frame, de seguida utiliza essa informação para prever no plano de imagem onde se encontram as marcas. Estas devem encontrar-se em locais aproximados ao previsto. Após a detecção real da posição destas marcas calcula-se a pose real do robô.

Em paralelo é executada a trajectória do robô, esta encontra-se descrita mais detalhadamente na figura 5.7:

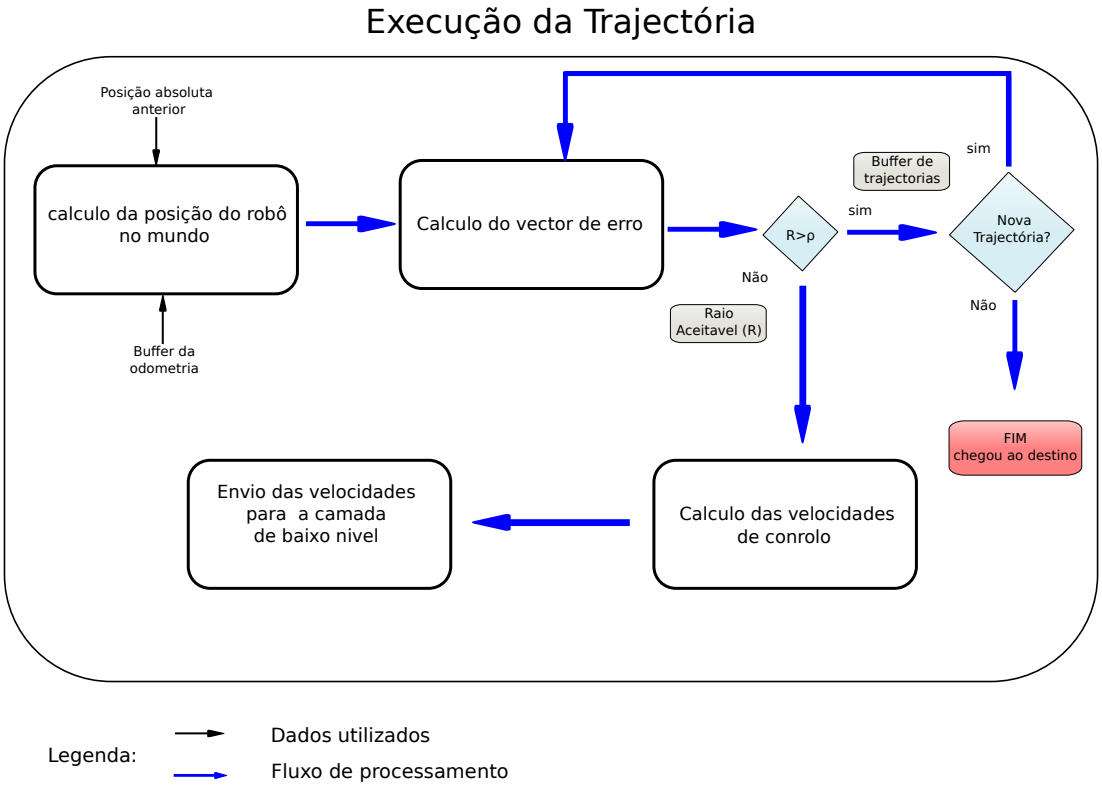


Figura 5.7: Ciclo de processamento da unidade de controlo

A execução da trajectória é efectuada periodicamente e recorre aos dados da posição obtida pela visão e de odometria para que a partir da pose do robô calcular o vector de erro relativamente a pose que se pretende atingir. Se o valor de ρ for menor que o valor do raio de aceitação de chegada ao destino passa a executar a próxima trajectória e calcula o novo vector de erro . Caso não haja uma nova trajectória o robô chegou ao destino. se o valor de ρ for maior que o valor do raio são calculadas as velocidades de controlo e estas são enviadas para a camada de baixo nível até que a condição de que o ρ seja menor que o raio se cumpra.

Capítulo 6

Resultados

Como foi mencionado anteriormente, o objectivo desta dissertação é desenvolver e testar uma solução de localização de robôs moveis de baixo custo, baseada na detecção de marcas reconhecidas a partir de visão artificial, bem como o desenvolvimento de um sistema de controlo de posição.

Para tal é necessário caracterizar como se comportam os sensores que compõe o sistema, começando pela a calibração da odometria e a calibração da Camâra. Para validar o sistema de localização implementado, e não existindo um sistema externo de validação da posição do robô, inicialmente é efectuada a caracterização dos erros sistemáticos da visão uma vez que para efeitos de calculo se considera que o plano de imagem e tecto estão perfeitamente alinhados, o que na pratica não se verifica. Para a caracterização desse erro é efectuado um percurso até um ponto conhecido e é efectuada uma rotação sobre o eixo do robô. Para caracterizar a localização são realizados percursos manualmente passando por posições no espaço conhecidas com trajectórias no sentido do ponteiro do relógio e no sentido contrario verificando a diferença entre a posição obtida pelo sistema e a posição real.

Após a caracterização da localização foi verificado o comportamento do controlo de posição. Para verificar como o sistema se comporta no seguimento de trajectórias previamente estabelecidas, foi executada um trajectórias com vários pontos por onde o robô deve passar, e comparou-se a sua prestação comparativamente a mesma trajectória com a sua simulação e com isto pretende-se verificar se o sistema de controlo de posição se comporta encontra a funcionar correctamente.

6.1 Calibração da odometria

A odometria é um sistema de posicionamento relativo o qual depende directamente de qual é o raio das rodas de tracção bem como da distancia entre estas. A calibração foi efectuada

utilizando a técnica de Calibração UMBmark.

Para calcular o diâmetro médio das duas rodas , percorreu-se em linha recta de $11,8$ ($\Delta d(i)$) metros e com o valor médio do numero de ticks obtidos para as duas rodas calculou-se o diâmetro médio.

Calcula-se a media dos ticks das duas rodas N_{tiks}:

$$N_{tiks} = \frac{N_{d_{tiks}} + N_{e_{tiks}}}{2} \tag{6.1}$$

Um exemplo do valor do numero de tiks que foi obtido para cada roda é de $N_{d_{tiks}} = 986313$ que corresponde a roda direita e de$N_{e_{tiks}} = 986024$ que corresponde a roda esquerda.

Sendo o valor médio para o total de seis rotas efectuadas de $N_{tiks} = 985650$. Com a distancia percorrida $d(i)$ e N_{tiks} calcula-se a distancia por cada tick ($dist_{d_{tick}}$)

$$\Delta d = dist_{d_{tick}} * N_{tiks} \tag{6.2}$$

Com a distancia por tick, obtém-se o diâmetro médio das rodas $D = 125.8611mm$:

$$dist_{tick} = \frac{\pi \times D}{N \times RE_{encoder}} \tag{6.3}$$

Onde N é relação da caixa e $RE_{encoder}$ a resolução do encoder.

A distancia entre os pontos de contacto entre as duas rodas foi obtido por medição directa em que $b = 435.0mm$.

Os resultados obtidos para as seis trajectórias podem ser encontrados na figura 6.1:

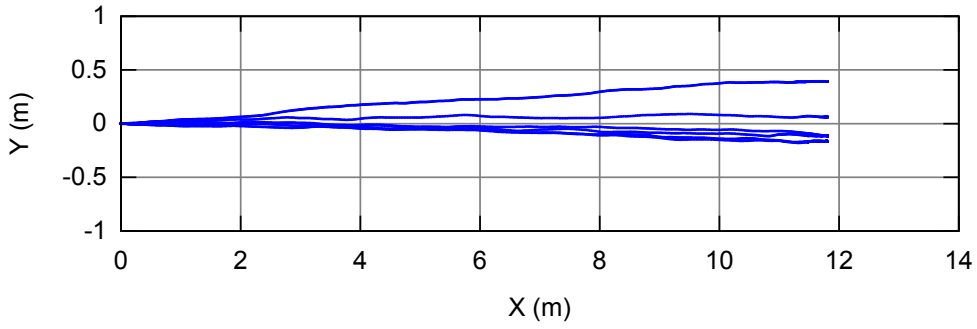


Figura 6.1: Rotas realizadas em linha recta para calcular diâmetro médio das rodas

Estes valores obtidos são a primeira aproximação aos valores reais dos parâmetros da odometria. Para corrigir os erros sistemáticos gerados gerados por esta primeira aproximação, foi utilizada a técnica de Calibração UMBmark a qual define três erros:

O erro de escala, que é definido por E_s o qual se considera corrigido pela medição do diâmetro médio das rodas fazendo com que o erro de escala seja significativamente baixo e pouco relevante comparativamente aos outros dois.

O erro devido a diâmetros das rodas diferentes, que é definido por: E_d.

O erro devido a incerteza da distancia entre os pontos de contacto das rodas com o chão, que é definido por E_b.

Para calcular estes dois erros foram executadas trajectórias com o formato de um quadrado fazendo um total de 12 trajectórias no sentido do ponteiro dos relógio e 12 no sentido contrario ao ponteiro do relógio. Na figura 6.2 pode visualizar-se os dados obtidos pela odometria numa das trajectórias efectuadas:

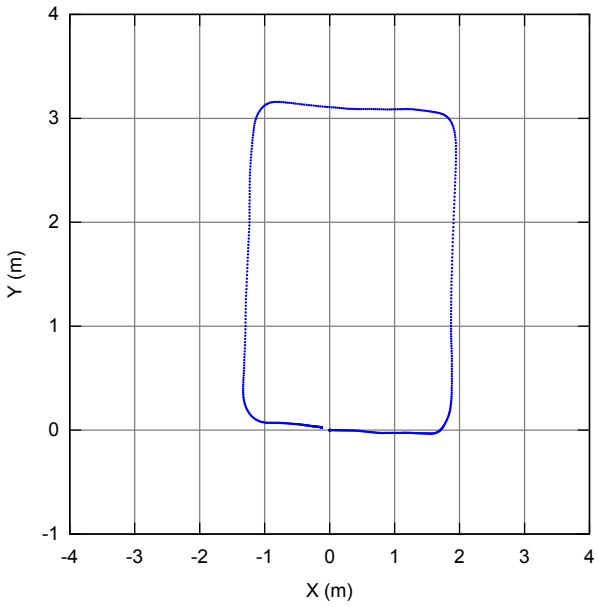


Figura 6.2: Rotas em forma de quadrado utilizada para efectuar a Calibração UMBmark

Após a realização das trajectórias calculou-se o centro de massa do erro para os dois eixos X e Y e para cada tipo de trajectória. Com as seguintes formulas:

$$X_{cm} = \frac{1}{n} \times \sum_{i=1}^{n} \Delta x_i \tag{6.4}$$

$$Y_{cm} = \frac{1}{n} \times \sum_{i=1}^{n} \Delta y_i \tag{6.5}$$

Para as trajectórias efectuadas no sentido do ponteiro do relógio o centro de massa do erro é:

$X_{cm_{PR}} = -130.1965mm$

$Y_{cm_{PR}} = 21.6120mm$

Para as trajectórias efectuadas no sentido contrario ao do ponteiro do relógio o centro de massa do erro é:

$X_{cm_{CPR}} = 93.5996mm$

$Y_{cm_{CPR}} = -5.2071mm$

O gráfico que representa os valores obtido para cada trajectória do erro em relação a posição real encontra-se na figura 6.3 :

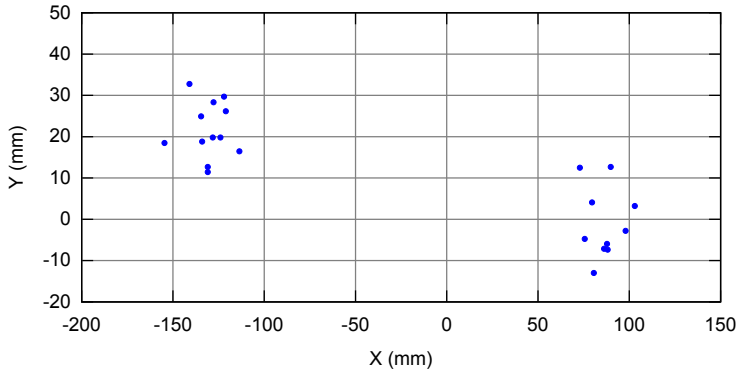


Figura 6.3: Erro em relação a posição final da cada trajectória efectuada

Uma vez que não há um sistema externo para determinar a pose inicial nem a posse final no fim da trajectória considerou-se que a posse final real é exactamente igual a pose inicial do inicio da realização da trajectória e por isso x=0 $x = 0$ $y = 0$ $\theta = 0$.

Para calcular os erros é necessário calcular o valor dos ângulos α e β

$$\alpha = media\left(\frac{X_{cm_{PR}} + X_{cm_{CPR}}}{-4D}, \frac{Y_{cm_{PR}} - Y_{cm_{CPR}}}{-4D}\right) \tag{6.6}$$

$$\beta = media\left(\frac{X_{cm_{PR}} - X_{cm_{CPR}}}{-4D}, \frac{Y_{cm_{PR}} + Y_{cm_{CPR}}}{-4D}\right) \tag{6.7}$$

Onde D é o tamanho do lado do quadrado.

Sendo obtido :

$\alpha = 0.1257°$

$\beta = -0.0769°$

Sabendo α e calcula-se E_b :

$$E_b = \frac{90^o}{90^o - \alpha} = 1.0014 \tag{6.8}$$

E consequentemente o b_{real}

$$b_{real} = E_b * b_{actual} = 435.6082cm \tag{6.9}$$

Para calcular o erro E_d:

$$R = \frac{D/2}{\sin(\beta/2)} \tag{6.10}$$

$$E_d = \frac{R + b/2}{R - b/2} = 0.999996 \tag{6.11}$$

Agora já se pode calcular o diâmetro de cada roda:

$$D_e = \frac{2}{E_d + 1} * D_m = 39.54048cm \tag{6.12}$$

$$D_d = \frac{2}{(1/E_d) + 1} * D_m = 39.54035cm \tag{6.13}$$

Refazendo o calculo da posição do robô com estes novos valores par b D_e e D_d e obetem-se o seguinte erro na posição:

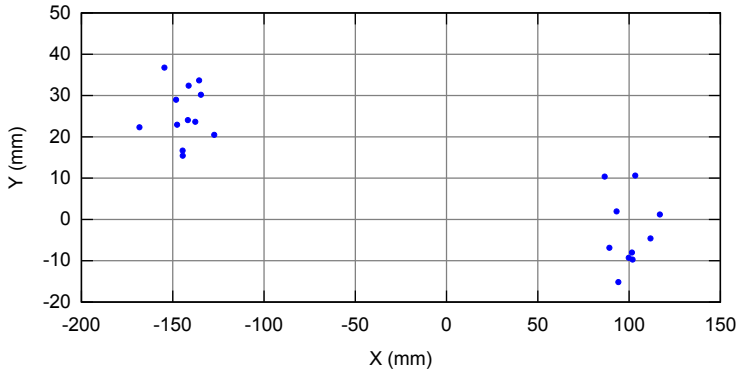


Figura 6.4: Erro em relação a posição final após a correcção

Como se pode verificar a diferença entre a calibração antes de efectuada a metodologia UMBmark e pós UMBmark não apresenta melhorias significativas. A razão para não se verificarem melhorias deve-se em primeiro lugar a que o Erro E_b e o erro E_d são bastante baixos uma vez que o valor do ângulo α é aproximadamente um décimo de grau tal como o ângulo β, e também devido a falta de um sistema de validação externo que permiti-se subtrair a diferença entre a posição inicial da trajectória e a final de forma a isolar correctamente o erro da odometria. Mas os resultados obtidos são validos, uma vez que o erro medio em xx é de 15 cm e em yy de 3 cm ao fim de um percurso de 12,228 metros.

6.2 Calibração da Camâra

A calibração da câmara utilizada foi efectuada com recurso a toolbox do Matlab de nome *Camera Calibration Toolbox*, esta permite obter os parâmetros intrínsecos da câmara a partir de aquisição de varias imagens onde um tabuleiro de xadrez se encontra com diferentes

poses.

A toolbox executa um algoritmo para a detecção dos cantos dos quadrados. A partir destes pontos de interesse e fornecido o tamanho físico do lado de um destes quadrados corre um algoritmo baseado no algoritmo Tsai e extrai os parâmetros intrínsecos da câmara.

Para a aquisição das imagens foi desenvolvido um pequeno programa recorrendo as bibliotecas open sorce *Video4Linux* e *OpenCV* de forma a poder gravar as imagens do tabuleiro sem que as imagens sofram qualquer tipo de processamento de imagem, sendo de seguida gravadas no formato BMP de forma a serem compatíveis com a toolbox do Matlab. Na figura 6.5 encontra-se uma das imagens utilizadas.

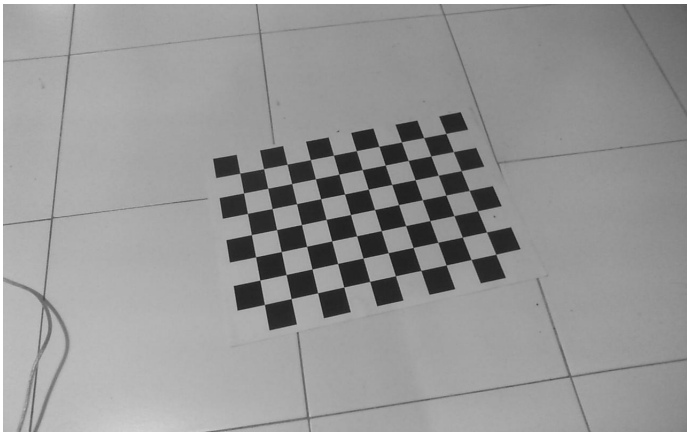


Figura 6.5: Exemplo de imagem utilizada para a calibração

Os valores obtidos para os parâmetros intrínsecos foram os seguintes:

f_x = 1132.08034 com um erro ±2.20541

f_y = 1129.94239 com um erro ±3.25904

O ponto principal localiza-se em :
S_x = 621.06768 com um erro ±2.53443

S_y = 409.93040 com um erro ±2.66033

Os parâmetros de distorção das lentes obtidos são os seguintes:
K_1 = 0.08067

$K_2 = -0.045455$

$K_3 = 0.00295$

$P_1 = 0.00014$

$P_2 = 0.0000$

Na figura 6.6 encontra-se a dispersão do erro em pixeis. Este erro representam a diferença entre a posição da detecção no plano de imagem dos cantos dos quadrados e a projecção para o plano de imagem da localização física destes cantos a partir da posição do tabuleiro :

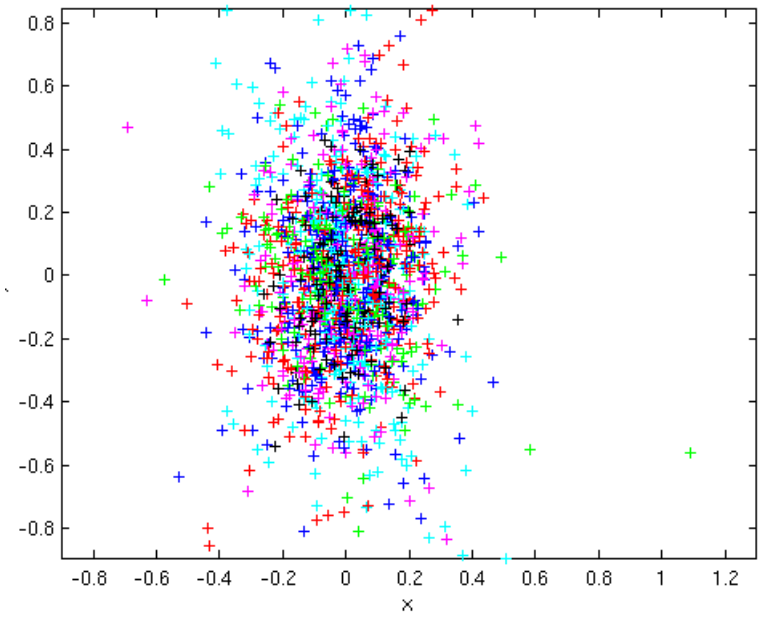


Figura 6.6: Erro em pixeis

O erro médio obtido em pixel para as imagens utilizadas é de :

0.15829 no eixo dos xx e de 0.27576 em yy

Sendo então o erro médio se encontra a baixo de um pixel, sabe-se que se obteve uma boa calibração.

6.3 Caracterização da Localização

Como já afirmado para efeitos de calculo considera-se que o plano de imagem e tecto estão perfeitamente paralelos, o que na pratica não acontece, esta característica do sistema faz com que se verifique um erro sistemático no sistema de localização o qual não foi resolvido.

Para a caracterização desse erro é efectuado um percurso até um ponto conhecido e é efectuada uma rotação sobre o eixo do robô. Esta rota encontra-se na figura 6.7.

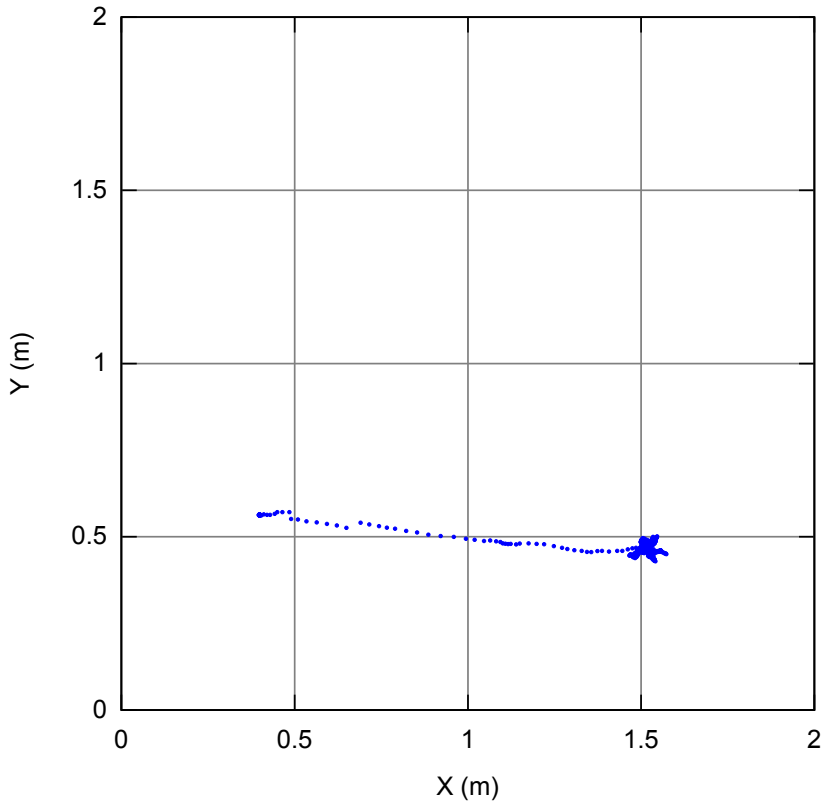


Figura 6.7: Rota efectuada com uma rotação sobre o próprio eixo

Como se pode verificar no gráfico, o que era expectável era que o ponto sobre o qual o robô rodou se mantive-se inalterado, mas os resultados demonstram a existência de um erro sistemático. O qual se deve a que o plano de imagem não esteja perfeitamente paralelo ao tecto. O tecto encontra-se a 2,86 metros do solo com um pequeno ângulo na posição da câmara de apenas 1 grau facilmente gera um erro sistemático de 5 centímetros. A dimensão deste erro não tinha sido prevista, a qual tem um enorme impacto na qualidade da localização.

A dimensão do erro sistemático encontrado tem um raio de $0.0535m$.

Outra característica do sistema e pode também ser verificado no gráfico anterior, é que sempre que o sistema de visão muda o par de marcas que utiliza como referencia para p calculo da localização, como por exemplo na posição aproximada de 0.5 em x e 0.5 em y verifica-se uma deslocação da pose do robô. Esta deslocação deve-se a colocação física das marcas, as quais não foi possível colocar milimetricamente na localização correcta. E este erro na colocação acaba por se reflectir nos resultados.

Para caracterizar o sistema de localização percorreu-se com o robô várias posições pre-determinadas com trajectória no sentido do ponteiro do relógio e sentido contrario. Na figura 6.8 está representada uma rota efectuada no sentido contrario aos ponteiros do relógio.

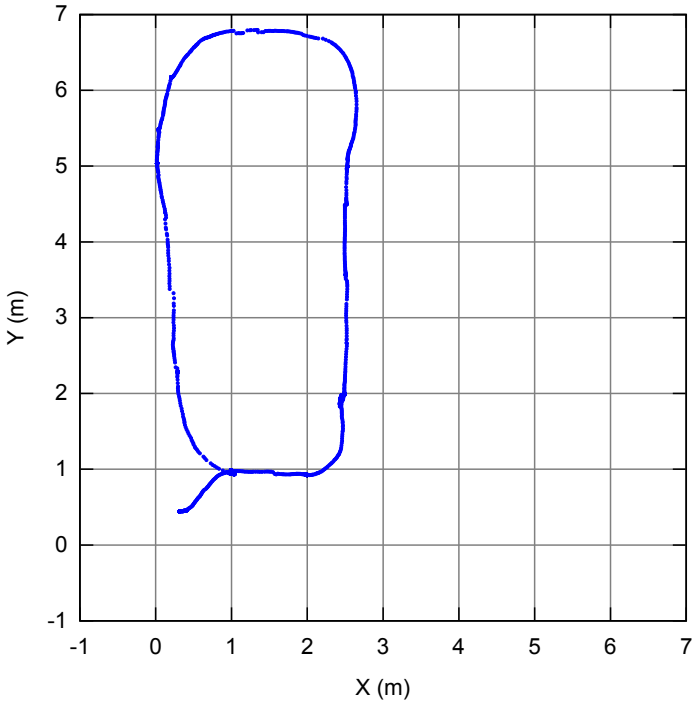


Figura 6.8: Rota efectuada no sentido contrario aos ponteiros do relógio

Em cada uma dessas trajectórias foram recolhidas as poses obtidas pelo sistema de localização, de forma a poder obter a diferença entre a posição obtida pelo sistema e a posição real. Na figura 6.9 encontra-se a localização desses pontos na rota anterior.

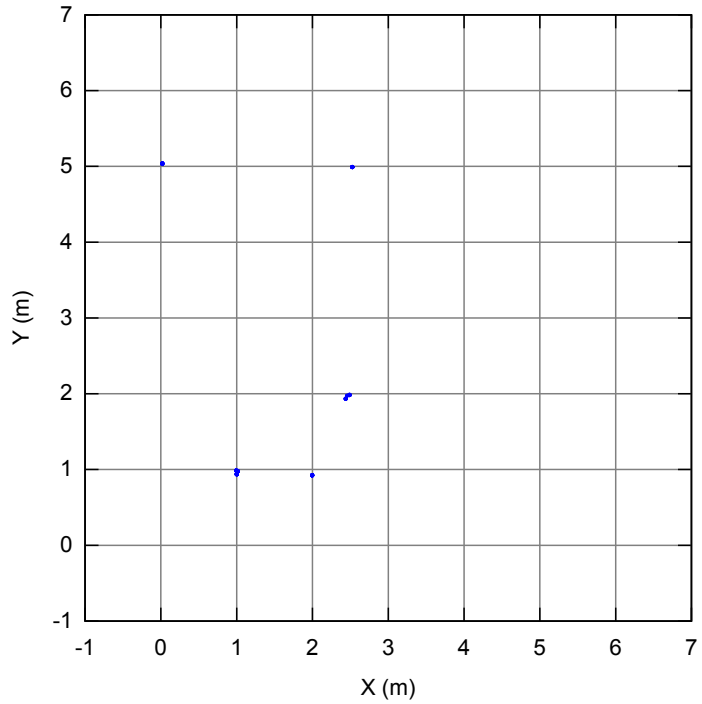


Figura 6.9: Pontos de interesse na rota no sentido contrario ao ponteiro do relógio

Os resultados obtidos encontram-se para esta rota encontra-se na tabela seguinte:

Tabela 6.1: Localizações reais versus Localizações determinadas (sentido contrario ao ponteiro do relógio)

X **(metros)**				Y **(metros)**				θ **(graus)**			
Real	**Media**	**erro médio**	**erro máximo**	**Real**	**Media**	**erro médio**	**erro máximo**	**Real**	**Media**	**erro médio**	**erro máximo**
1.000	1.005	0.008	0.0148	1.00	0.9703	0.0297	0.0626	0.0º	-1.6390º	1.6390º	2.8692º
2.00	1.9960	0.0040	0.0044	1.00	0.9251	0.0749	0.0749	0.0º	-0.0879º	0.0879º	0.4045º
2.50	2.4730	0.0270	0.0649	2.00	1.9704	0.0296	0.0662	90º	89.9751º	0.3890	1.3448
2.50	2.5267	0.0267	0.0267	5.00	4.9918	0.0082	0.0087	90º	88.8356	1.1644	1.2252
0.0	0.0174	0. 0.0174	0.0189	5.00	5.0368	0.0368	0.0384	-90º	-90.0449	0.0449	0.1348

Pelos resultados obtidos pode verificar-se um erro maximo em xx é de 6.49 cm e em yy de 7.49 cm e no ângulo de 2.8692 graus. O erro relativo a posição pelos valores apresentados parece ser consistente com os valores apresentados pelo erro sistemático anteriormente descrito. Relativamente ao erro na orientação foi obtido um erro máximo de 3 graus o qual se considera um erro baixo uma vez que metodologia utilizada para alinhar o robô fisicamente com a grelha pode facilmente dar resultado a erros na orientação nesta ordem de valor .

Na figura 6.10 encontra-se uma rota realizada no sentido dos ponteiros do relógio. Em que foram extraídos os dados de localização nos mesmos pontos de interesse que na rota anterior mas com orientações opostas.

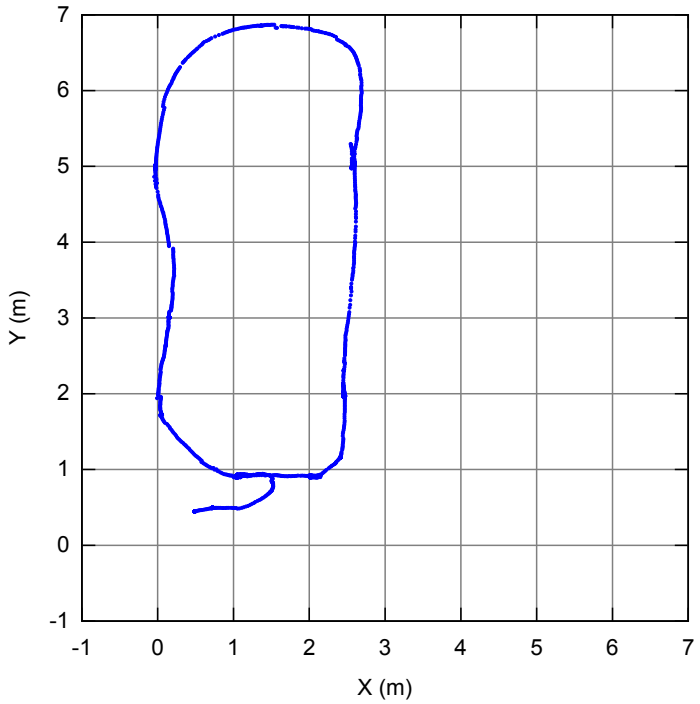


Figura 6.10: Rota efectuada no sentido dos ponteiros do relógio

Na Tabela 6.3 encontram-se os resultados obtidos comparativamente aos da localização real.

Tabela 6.2: Localizações reais versus Localizações determinadas (no sentido dos ponteiros do relógio)

X **(metros)**				Y **(metros)**				θ **(graus)**			
Real	**Media**	**erro médio**	**erro máximo**	**Real**	**Media**	**erro médio**	**erro máximo**	**Real**	**Media**	**erro médio**	**erro máximo**
1.000	1.0542	0.0542	0.0550	1.00	0.9162	0.0838	0.0846	180	176.0423	3.9577	4.2967
2.00	2.0154	0.0154	0.0154	1.00	0.9179	0.0821	0.0832	180	179.3283	0.6717	0.6725
2.50	2.4667	0.0333	0.0333	2.00	1.9569	0.0431	0.0431	-90º	-90.1398	0.1398	0.1440
2.50	2.5965	0.0965	0.0965	5.00	5.0166	0.0166	0.0174	-90º	-89.6751	0.3249	0.4044
0.0	-0.0245	0.0245	0.0298	5.00	5.0244	0.0346	0.0404	90º	89.2177	0.7823	1.9234

Na figura 6.11 encontra-se a rota efectuada comparativamente aos dados em bruto recolhidos pela odometria. A azul encontra-se a rota efectuada recorrendo aos dados da posição obtidas pela visão e a vermelho a rota que foi efectuada segundo a odometria. Pode verificar-se a acumulação de erro desta metodologia, em que a posição final obtida pela odometria é para xx de 1.418 metros e 0.915 em yy que para a localização obtida pela visão corresponde a 1.054 metros e 0.915 respectivamente.

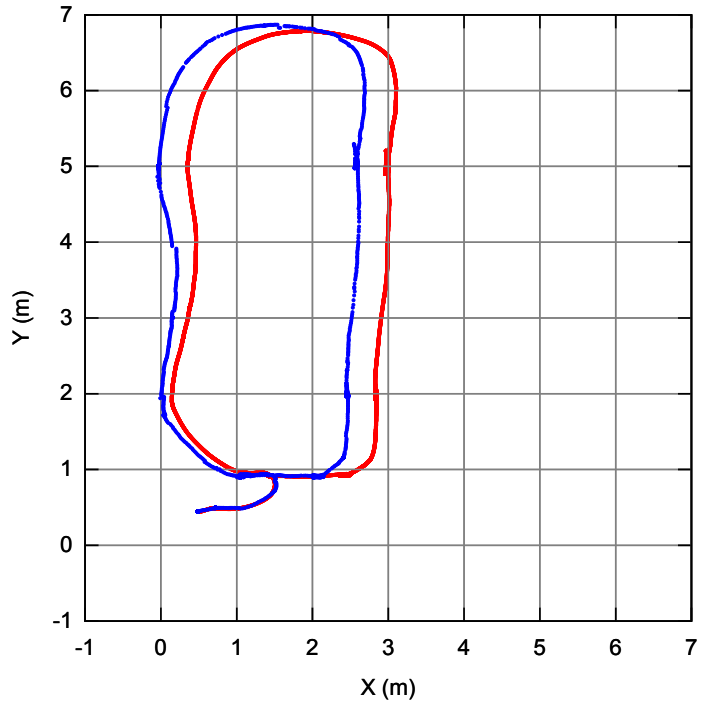


Figura 6.11: Compração da rota com os dados em bruto da odometreia

Uma das características deste sistema é que quando não é obtida a posição, por obstrução da câmara, ou por saturação do CCD da câmara a pose é obtida recorrendo a ultima pose obtida pelo sistema de visão artificial mais a localização relativa obtida pela odometria. Na figura 6.12 relativo a rota anterior a vermelho encontra-se representada a posição obtida pelo sistema de visão artificial e a azul a posição do robô prevista pela ultima localização do sistema de visão mais a posição relativa obtida pela odometria.

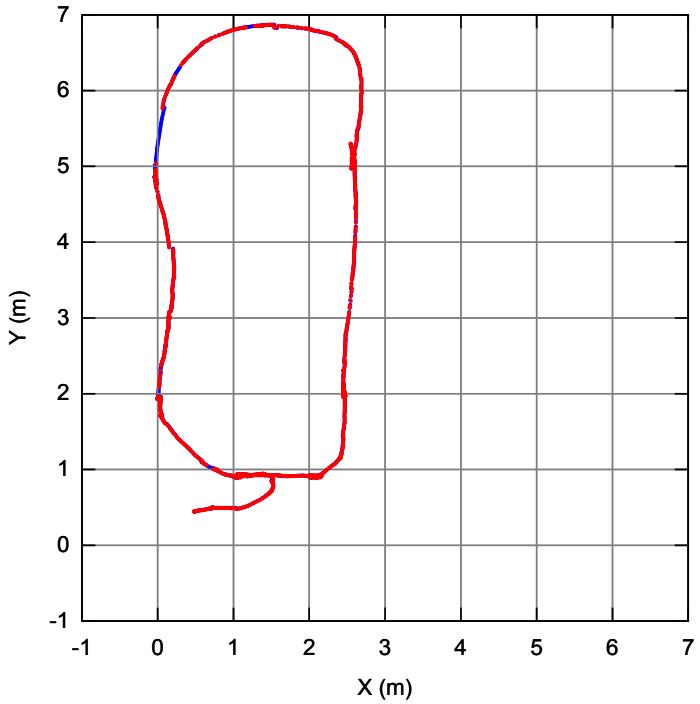


Figura 6.12: Identificação da posição relativa

Como se pode verificar a pose do robô na localização aproximada de (x=0,y=5.0) até a localização de (x=0, y=5.5) apenas existe a localização a azul que corresponde ao valor obtido pela ultima localização obtida pela visão artificial (ultimo ponto a vermelho em y=5.0) mais a posição relativa obtida pela odometria. Após a câmara se encontrar "desobstruída" o sistema volta a detectar a posição das marcas e actualiza a posição correctamente.

6.4 Trajectórias com recurso ao controlo de posição

Para validar o funcionamento do sistema de controlo de posição e de velocidade foi implementada uma rota a partir de pontos por onde o robô deve passar, de forma a verificar se o comportamento do robô é o que se esperaria em relação ao simulado. Para tal foram definidos onze localizações por onde o robô deve passar.Estes valores encontram-se na tabela 6.4 a qual contem as poses pretendidas para cada uma destas localizações bem como o raio de aceitação como posição valida já descrita na secção de controlo e também os valor das constantes de controlo definidos para cada trajectória .

Tabela 6.3: Pontos da rota, constantes de controlo e raio de aceitação (em metros)

Pose (metros)			**Constantes de controlo**				**Pose Obtida**			
X	Y	θ	k_ρ	k_α	k_β	**Raio**	X	Y	θ	**Raio**
2.5	1.50	90.0	0.250	1.10	-0.40	0.20	2.5390	1.3067	80.7251	0.1972
2.7	3.0	90.0	0.300	1.00	-0.2250	0.20	2.7560	2.8108	85.0419	0.1973
2.7	4.5	90.0	0.250	2.70	-0.010	0.20	2.7192	4.3044	94.3466	0.1965
2.7	6.0	90.0	0.250	2.70	-0.010	0.20	2.7089	5.8118	96.2826	0.1884
1.9	6.5	-180.0	0.300	1.0	-0.2250	0.20	2.0710	6.5769	160.5254	0.1875
0.8	6.6	-180.0	0.250	2.70	-0.010	0.20	0.9981	6.5934	-176.9377	0.1983
0.2	6.0	-90.0	0.300	1.00	-0.2250	0.20	0.2613	6.1667	-125.2622	0.1776
0.2	5.0	-90.0	0.250	2.70	-0.010	0.20	0.2122	5.1909	-94.7486	0.1913
0.2	3.5	-90.0	0.250	2.70	-0.010	0.20	0.1948	3.6961	-87.8816	0.1961
0.2	2.0	-90.0	0.250	2.70	-0.010	0.20	0.2006	2.1930	-90.1856	0.1930
0.5	1.0	-45.0	0.250	1.10	-0.40	0.20	0.4086	1.1652	-62.8149	0.1888
1.5	0.5	0	0.250	1.10	-0.40	0.20	1.3141	0.5298	-25.3262	0.1883
2.5	0.2	0	0.250	1.10	-0.40	0.20	2.3260	0.2340	-13.0346	0.1773

Pode verificar-se pelos valores tabelados que o sistema atinge com sucesso os pontos estipulados dentro de um raio de 20 centímetros. Na figura 6.13 encontram-se o percurso efectuado pelo robô. Onde a vermelho encontra-se a posição obtida pelo sistema de visão artificial e a azul encontra-se a localização anterior obtida pela visão mais a posição relativa obtida pela odometria.

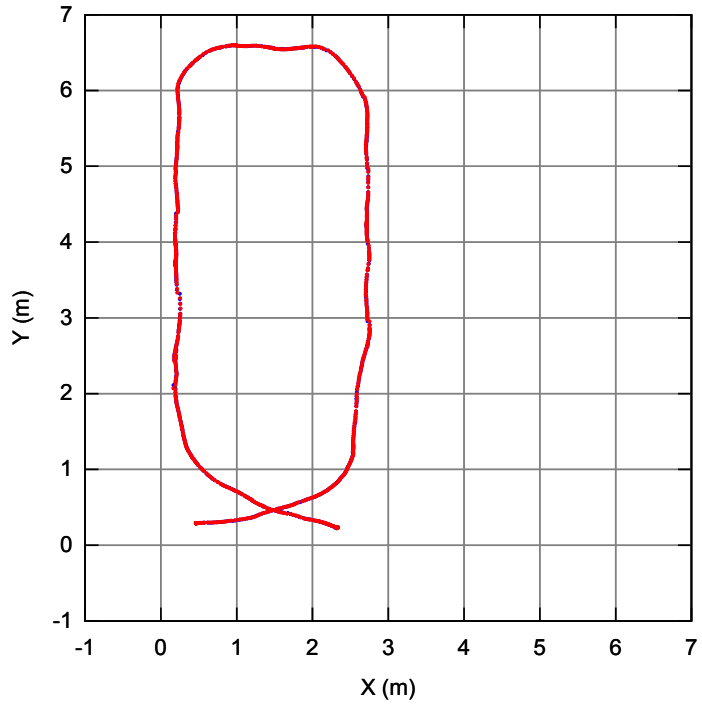


Figura 6.13: Rota efectuada pelo controlo

Na figura 6.14 encontra-se sobreposta a rota efectuada com o robô com tracejado a azul e a rota simulada com o tracejado a vermelho.

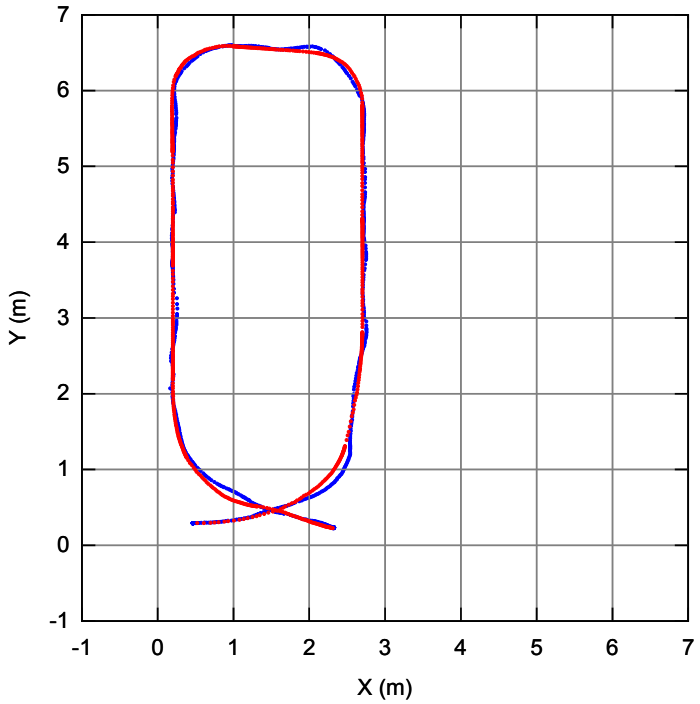


Figura 6.14: Rota efectuada pelo controlo (Azul) e simulação(Vermelho)

Como se pode verificar a trajectória resultante é bastante aproximada. Na tabela 6.4

Tabela 6.4: Pontos da rota, erros e comparção entre a rota e a simulação

Pose pretendida(metros)			**Pose de controlo**				**Pose de simulação**			
X	Y	θ	X	Y	θ	**Raio**	X	Y	θ	**Raio**
2.5	1.5	90.0	2.5390	1.3067	80.7251	0.1972	2.4745	1.3176	72.1210	0.1842
2.7	3.0	90.0	2.7560	2.8108	85.0419	0.1973	2.6996	2.8177	89.9495	0.1823
2.7	4.5	90.0	2.7192	4.3044	94.3466	0.1965	2.7003	4.3071	90.0647	0.1929
2.7	6.0	90.0	2.7089	5.8118	96.2826	0.1884	2.7000	5.8131	89.9977	0.1869
1.9	6.5	-180.0	2.0710	6.5769	160.5254	0.1875	2.0844	6.4939	171.3386	0.1845
0.8	6.6	-180.0	0.9981	6.5934	-176.9377	0.1983	0.9905	6.5862	176.5598.	0.1910
0.2	6.0	-90.0	0.2613	6.1667	-125.2622	0.1776	0.2461	6.1795	-113.9121	0.1854
0.2	5.0	-90.0	0.2122	5.1909	-94.7486	0.1913	0.1972	5.1885	-89.2851	0.1886
0.2	3.5	-90.0	0.1948	3.6961	-87.8816	0.1961	0.2000	3.6878	-90.0086	0.1878
0.2	2.0	-90.0	0.2006	2.1930	-90.1856	0.1930	0.2000	2.1914	-89.9999	0.1914
0.5	1.0	-45.0	0.4086	1.1652	-62.8149	0.1888	0.3872	1.1505	-59.0825	0.1881
1.5	0.5	0	1.3141	0.5298	-25.3262	0.1883	1.3107	0.5126	-9.3968	0.1897
2.5	0.2	0	2.3260	0.2340	-13.0346	0.1773	2.3148	0.2315	-12.8441	0.1915

Na figura 6.15 encrontra-se o gráfico da velocidade linear a que o robô se deslocou onde se pode ver claramente os picos de velocidade quando começa uma nova trajectória onde o ponto de destino se encontra a uma distancia linear elevada. A qual faz com que a velocidade linear aumente consideravelmente no inicio desse tipo de trajectória. Cada vez que se aproxima do destino a velocidade vai reduzindo. sendo possível verificar os com a excepção do inicio os 13 picos inferiores de velocidade, sendo que na ultima trajectória vai a zero uma vez que chegou ao local final.

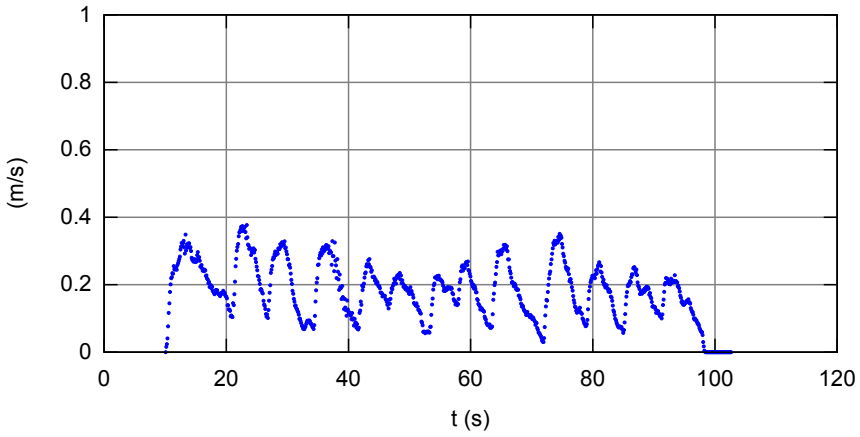


Figura 6.15: Evolução da velocidade linear durante a rota

Na figura 6.16 encontra-se a evolução da velocidade angular, esta é mais difícil de interpretar apenas de notar que a velocidade angular positiva é mais predominante que a negativa, uma vez que a rotação em torno da mesa se da no sentido contrario ao ponteiro dos relógio.

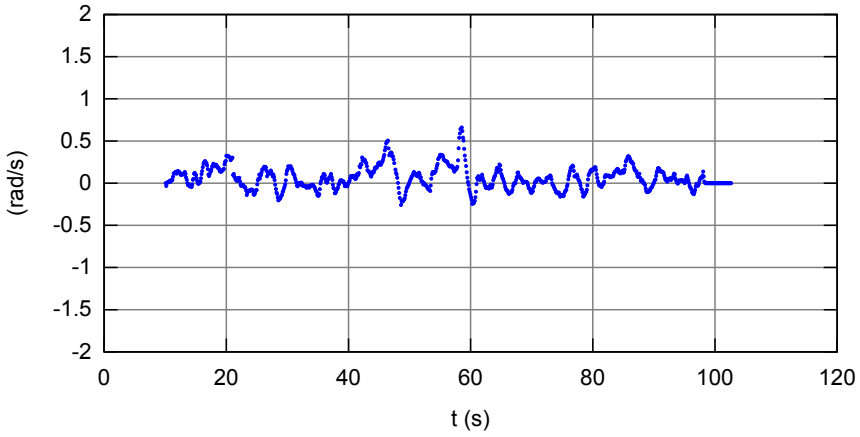


Figura 6.16: Evolução da velocidade angular durante a rota

Capítulo 7

Conclusões

O objectivo fundamental da realização desta dissertação foi o desenvolvimento de uma solução de localização/navegação para ambientes industriais semi-estruturados. A estruturação do ambiente ocorre pela colocação de “landmarks“ passivas no tecto da área de operação. Estas marcas formam a grelha que é utilizada como referência do sistema.

A detecção destas marcas é realizada por um algoritmo de visão artificial, desenvolvido de forma a detectar a posição das marcas no plano de imagem e a partir da geometria do modelo de câmara pinhole e de algoritmos de remoção da distorção das lentes obteve-se a localização do robô móvel.

A localização relativa foi obtida com recurso a odometria, com esta foi possível recuperar a posição do robô na ocorrência de obstrução da câmara e da sua saturação. Também com recurso a odometria efectuou-se a previsão da localização das ”Landmarks” no plano de imagem e com isso foi possível uma redução do tempo de processamento e principalmente conferir mais robustez ao sistema de localização. Pode-se verificar na figura 6.12 do capítulo de resultados, uma recuperação da posição a partir da odometria . Esta característica de recuperação da posição só é possível para distancias pequenas (poucos metros) uma vez que depende do erro acumulado pela odometria.

A localização, como se pode constatar pelos resultados obtidos, demonstram que a solução de localização implementada é válida. Pode-se verificar nas duas rotas efectuadas manualmente, uma delas no sentido contrário do ponteiro dos relógios (figura 6.9) e a outra no sentido dos ponteiro dos relógios (figura 6.10) que os dados recolhidos correspondem a rota efectuada que foi efectuada pelo robô. Tal pode-se comprovar pelas cinco localizações físicas que foram comparadas com as localizações obtidas pelo sistema implementado no qual foi obtido um erro máximo de 10 centímetros . O erro obtido foi alvo de um estudo mais aprofundado, com o qual se chegou a conclusão de que este é natureza sistemática. Este está evidenciado na figura 6.7, onde se encontra o exemplo de uma rotação do robô sobre o

próprio eixo , erro limitado aos 10 centímetros.

Salienta-se que o algoritmo desenvolvido necessita é sensível a exactidão do posicionamento tanto das “landmarks“, como da câmara, a qual tem de ter o seu plano de imagem o mais paralelo possível ao plano onde estão colocadas as marcas. O desalinhamento do plano de imagem e do plano das “landmarks“ é o maior contributo para o erro do sistema.

O algoritmo de controlo de posição em malha fechada que foi desenvolvido recorre aos dados relativos à posição do robô e gera as velocidades aplicadas aos motores. Como o controlo de posição depende das velocidades que são aplicadas ao robô, desenvolveu-se um módulo de controlo de velocidade para motores DC (pode-se encontrar em anexo a sua descrição). Este controlador de velocidade dos motores é baseado num controlador PID. Este sistema permite que o robô se desloque de uma localização para uma determinada pose.

De forma a validar a arquitectura de sistema que é constituída pelo conjunto localização e controlo de posição foi efectuada uma rota pré-definida entre diversas localizações, como se pode verificar na figura 6.14 do capítulo de resultados. Os resultados obtidos demonstram que este conjunto se comporta como o previsto na simulação. O sistema desenvolvido é por isso um sistema que atinge por isso o propósito para o qual foi criado.

7.1 Trabalho futuro

Apesar de os objectivos inicialmente propostos terem sido alcançados durante a realização do trabalho, foram sendo detectadas, limitações que podem ser ultrapassados em futuros desenvolvimentos.

Uma das melhorias a implementar é no controlo cinemático, para o qual é necessário implementar um algoritmo que gere as melhores trajectórias possíveis entre dois pontos. Dito de outra forma, para cada par de pose actual e pose de destino, existe um conjunto de valores para as constante de controlo que são ideais para poder obter uma pose final o mais aproximada possível do pretendido.

Outra das melhorias a implementar prende-se com a necessidade de uma filtragem estatística com recurso ao modelo cinemático do robô, de forma a que por algum motivo de má colocação das marcas e por isso a existência de saltos na localização do robô, exista uma transição mais suave entre as duas posições, o que se revela uma grande vantagem para as velocidades de controlo que não têm picos de variação, sendo o comportamento do robô mais suave.

A melhoria mais significativa e mais trabalhosa seria na mudança do algoritmo responsável pelo cálculo da posição do sistema de visão artificial. A alteração a implementar exige que a posição seja obtida com recurso a três marcas em vez de duas de forma a poder obter a relação entre o plano de imagem e o plano do tecto que é definido pelas marcas. Com esta relação é possível obter a orientação da câmara e com isso eliminar o erro sistemático encontrado. Da mesma forma teria de ser alterado o algoritmo de cálculo da posição inicial.

Bibliografia

[1] *AGV Electronics*. Disponível em http://www.agve.se/. Último acesso a Fevereiro de 2013.

[2] *Automated Guided Vehicle Brochure*. Disponível em http://www.ocme.it/website/get_download.aspx?ctrb_id=177 Último acesso a Fevereiro de 2013.

[3] *The Basics of Automatic Guided Vehicle Systems*. http://www.agvsystems.com/basics/index.htm Último acesso ao link: Fevereiro de 2013.

[4] *Camera Calibration Toolbox for Matlab*. Disponível em http://www.vision.caltech.edu/bouguetj/calib_doc/ Último acesso a Fevereiro de 2013.

[5] *elettric 80*. Disponível em http://www.elettric80.com/ Último acesso a Fevereiro de 2013.

[6] *frog AGV*. Disponível em http://www.frog.nl/ Último acesso a Fevereiro de 2013.

[7] *Guidance Options for Automatic Guided Vehicles*. Disponível em http://www.jervisbwebb.com Último acesso ao link: Fevereiro de 2013.

[8] *kiva systems*. Disponível em http://www.kivasystems.com/solutions/kiva-vs-traditional/solutionskiva-vs-traditionalkiva-vs-agvs Último acesso a Fevereiro de 2013.

[9] BAMA. *Technology & innovation in automatic systems*, 2013. http://www.bama.es/v2/ ; Último acesso ao link: Fevereiro de 2013.

[10] J. Borenstein and Liqiang Feng. Measurement and correction of systematic odometry errors in mobile robots. *Robotics and Automation, IEEE Transactions on*, 12(6):869–880, December 1996.

[11] Dr. Gary Bradski and Dr. Adrian Kaehler. *LearningOpenCV*. O'Reilly, 2008.

[12] D. C. Brown. In *Decentering Distortion of Lenses*, pages 444–462, 1966.

[13] J.L. CHENAVIER, F; CROWLEY. Position estimation for mobile robot using vision and odometry. In *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*, 1992.

[14] ek automation. *AGV's - Segurança e Navegação*, 2013. Disponível em http://www.ek-automation.com/products/automated-guided-vehicles/navigation.html Último acesso ao link: Fevereiro de 2013.

[15] e L. Feng. J. Borenstein, H. R. Everett. *Where am I?- Systems and Methods for Mobile Robot Positioning*. The University of Michigan, 1996.

[16] David Lima. Localização absoluta de robôs móveis em ambientes industriais. Master's thesis, Faculdade de Engenharia da Universidade Do Porto, 2010.

[17] Fábio Santos Lobão. Calibração com mapeamento das distorções geométricas aplicada a um sistema de visão stereo. August 2004.

[18] Mecalux. *Veicolo AGV*, 2013. http://www.logismarket.it/trilogiq-italia/veicolo-agv/1934561227-10474094-p.html ; Último acesso ao link: Fevereiro de 2013.

[19] J. J. Muñoz-César, E. A. Merchán-Cruz, L. H. Hernández-Gómez, E. Guerrero-Guadarrama, A. Jiménez-Ledesma, and I. Jaidar-Monter. Speed control of a dc brush motor with conventional pid and fuzzy pi controllers. In *Proceedings of the 2008 Electronics, Robotics and Automotive Mechanics Conference*, CERMA '08, pages 344–349, Washington, DC, USA, 2008. IEEE Computer Society.

[20] Marco Neves. Auto-tuning de controladores pid pelo método relay. Master's thesis, Instituto Superior Técnico Universidade Técnica de Lisboa, 2009.

[21] e Paulo José Costa. Pedro M. Carvalho, A. Paulo Moreira. Small and low-cost agv for distributed production lines and warehouses. *8th Portuguese Conference on Automatic Control*, page 827–832, 2008.

[22] Szenberg F Gattass M Celes W Raposo, AB. *Visão Estereoscópica, Realidade Virtual, Realidade Aumentada e Colaboração*.

[23] Rui Paulo Rocha. *Estado da Arte da Robótica Móvel em Portugal*, Março 2001. Disponivel em: http://mail.isr.uc.pt/ mrl/admin/upload/37.pdf.

[24] Roberto Silva. Localização de agv's industriais baseada em marcadores. Master's thesis, Faculdade de Engenharia da Universidade Do Porto, 2011.

[25] Héber Sobreira. *Clever Robot*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2009.

[26] Carlos Teixeira. *Ensinamento Rápido de Manipuladores Industriais* . PhD thesis, July 2009.

[27] Wolfram; FOX Dieter THRUN, Sebastian; BURGARD. *Probabilistic Robotics. Cambridge (Mass.): MIT Press, 2005. ISBN 0-262-20162-3.*

[28] Trilogiq. *Mobile operating vehicle*, 2013. http://www.trilogiq.com/en/move/assembly-line-automated-guided-vehicle.php ; Último acesso ao link: Fevereiro de 2013.

[29] Roger Y. Tsai. Radiometry. chapter A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, pages 221–244. Jones and Bartlett Publishers, Inc., USA, 1992.

Apêndice A

Hardware

Depois de definidos os requisitos do robô, procedeu-se ao desenvolvimento e montagem do hardware do robô. Neste Anexo será apresentado o hardware desenvolvido e uma apresentação geral do sistema, para de seguida ser descrita cada parte do sistema com mais pormenor.

A.1 Traços gerais

Na figura A.1, pode-se observar o robô desenvolvido no presente trabalho. Trata-se de um robô diferencial no formato de um octógono e a sua electrónica encontra-se localizada parte inferior da plataforma visível na imagem. As suas dimensões foram estudadas de forma a passar por portas de tamanho padrão, e o formato de octógono foi pensado de de forma a facilitar trajectórias curvas em zonas apertadas que no caso de a plataforma ser quadrada as suas quinas inviabilizariam muitas destas trajectórias.

O robô é uma plataforma móvel que permite que no futuro seja adaptado para diferentes tipos de aplicação. A localização da câmara tem como único requisito ser colocada de forma que o seu plano de imagem esteja paralelo ao tecto e por isso a sua colocação é bastante flexível. Isto é importante para que quando é atribuída uma funcionalidade à plataforma está possa ser recolocada bastando para isso saber a sua pose no plano XY ao referencial do robô.

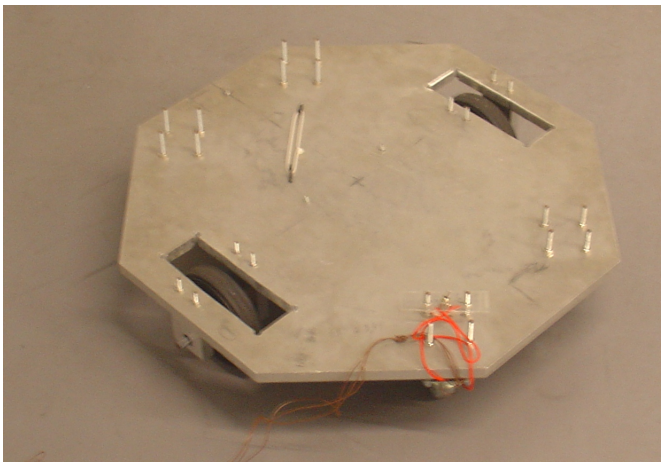


Figura A.1: Robô desenvolvido

Na figura A.2 pode-se observar o hardware que compõe o robô. Pode-se verificar a existência de duas placas de desenvolvimento. Uma delas, (pandabord) ainda não se encontra em utilização, mas pretende-se que o processamento de alto nível passe a ser executado numa plataforma deste tipo. No entanto na actualidade é feita com recurso a um portátil.

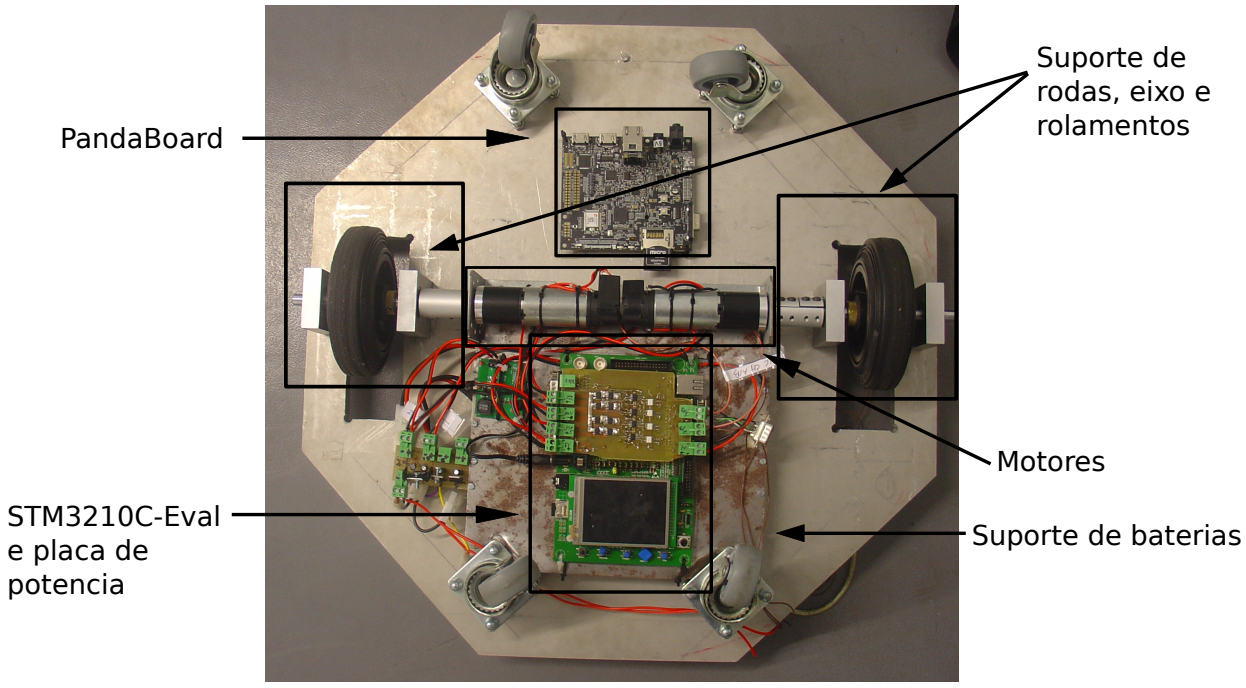


Figura A.2: Hardware do Robô

O processamento do robô divide-se em duas camadas uma camada inferior responsável, pelo controlo dos actuadores e pelo processamento de sinal do encoders e uma camada superior responsável pelo processamento de imagem e pelas decisões de alto nível, como a localização e a sistema de controlo de posição.

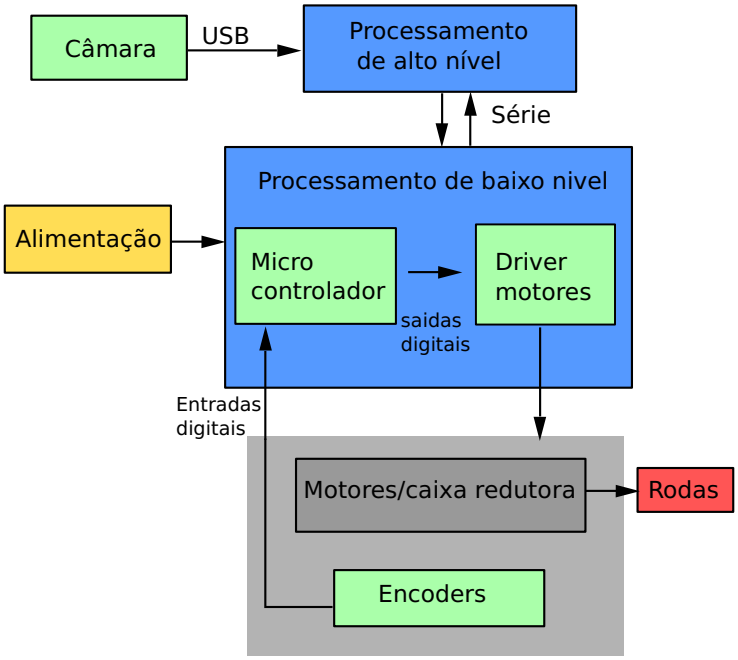


Figura A.3: Diagrama de blocos do Hardware

Podemos observar na Figura A.3 o esquema funcional do hardware desenvolvido, e nas secções seguintes cada bloco da figura será descrito com mais pormenor.

A.2 Suporte e rolamentos para o eixo das rodas

Na figura A.4 pode ver-se o pormenor da constituição dos suportes para as rodas de tracção. O conjunto é composto pela roda o eixo com um centímetro de diâmetro os rolamentos e duas peças de alumínio que seguram ambos os lados do eixo, esta montagem serve para retirar o peso do robô do eixo que liga ao motor.

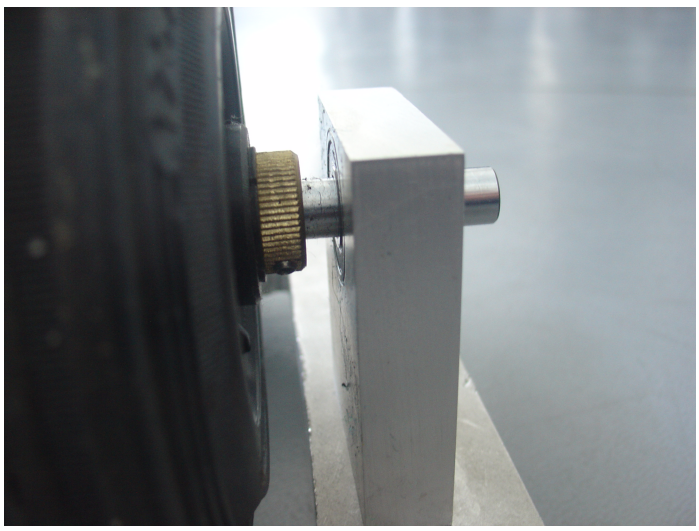


Figura A.4: Pormenor da constituição dos suportes das rodas de tracção

A.3 Rodas de tracção

Na Figura A.5 podemos ver uma da duas rodas de tracção do robô. Estas rodas permitem uma superfície de contacto com o chão não muito elevada o que ajuda a que a variação dos pontos de contacto das rodas com o chão não varie muito e com isso reduzir os erros no sistema de odometria , o que aumenta a aderência do robô ao chão, e ajuda a evitar situações de derrapagem.

Figura A.5: Rodas de tracção do robô

A.4 Rodízios

O robô diferencial necessita de um ou mais apoios alem das duas rodas de tracção estes apoios não interferem no movimento do robô. Na implementação optou-se por colocar quatro apoios, no entanto só estão em contacto com o chão dois de cada vez, isto porque estando quatro apoiados pode facilmente ocorrer deslizamento das rodas de tracção o que vai afectar a odometria. Na figura A.6 encontra-se o tipo de rodízios utilizados.

Figura A.6: Rodízios de suporte

A.5 Motor, Caixa redutora e encoderes Ópticos

Foram utilizados dois motores com uma potencia nominal de 20 Watts cada, estes estão equipados com uma caixa redutora com uma relação de 33 : 1 e um *encoder*. O encoder esta fixado ao eixo do motor e por cada rotação completa do eixo o numero de impulsos é de 500.

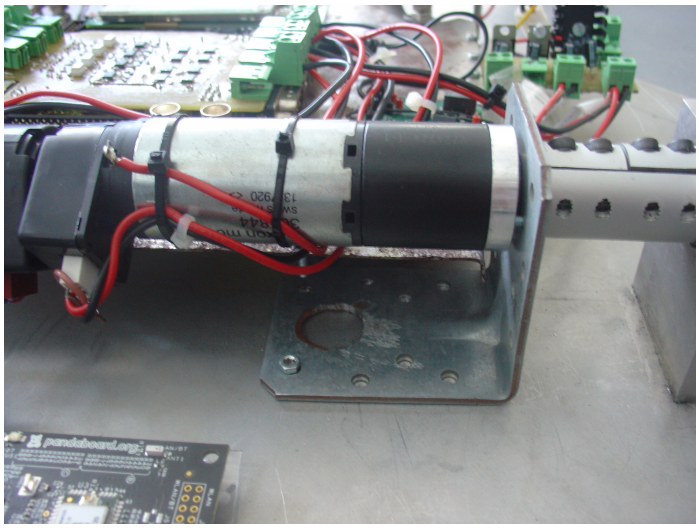


Figura A.7: Conjunto motor mais caixa e encoder

A.6 Alimentação

A alimentação do robô é fornecida por baterias de 12 V de *Nickel–metal hydride battery* (Ni-HM) , sendo que são utilizadas 2 packes em serie de forma a atingir os 24V nominais do sistema . É utilizado um regulador de tensão para 15 V os quais são utilizados pelos drivers de potencias e um um conversor DC-DC para baixar a tensão para os 5V para alimentar ambas as placas desenvolvimento. Na figura encontra-se o exemplo de um pack de baterias utilizado.

Figura A.8: Bateria de 12V de Ni-HM

A.7 Câmara

A câmara utilizada é uma *Microsoft LifeCam HD-3000* com resolução máxima de 1280 X 800 pixeis com um frame rate de 10 fps. É uma câmara de baixo custo.

Figura A.9: Câmara LifeCam HD-3000

A.8 Driver de potencia para os motores

Para este sistema foi desenvolvida uma placa de potencia que alimenta os motores, e serve para realizar todas as conexões necessárias para o correcto funcionamento dos encoderes bem como a implementação de um botão de emergência. Esta placa foi desenhada para controlar motores de tensão nominal de 24 voltes. O aspecto desta encontra-se na A.10 na qual se encontram todos identificados todos os conectores desta. Como nota fica que os conectores CN-A e CN-B são os dois conectores que encaixam na placa de desenvolvimento que se encontra descrita a seguir.

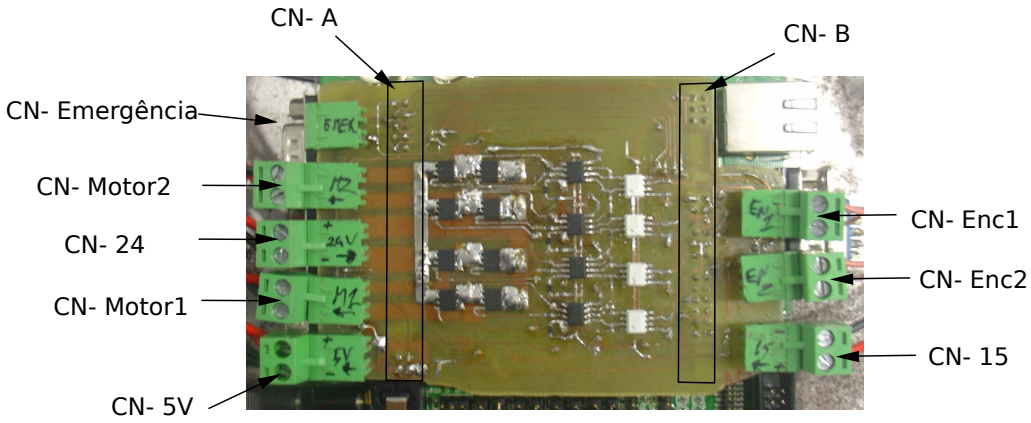


Figura A.10: Placa de potencia para os motores

A.9 Placa de desenvolvimento

O placa de desenvolvimento utilizada para a implementação do controlo de velocidade dos motores, aquisição de informação dos encoderes e calculo da odometria é o modelo STM3210C-EVAL que se encontra na figura A.11. Esta placa de desenvolvimento contem o micro controlador STM32F107VCT o qual é baseado na arquitetura ARM cortex- M3 com 64KB de

SRAM e 256 KB de memoria.

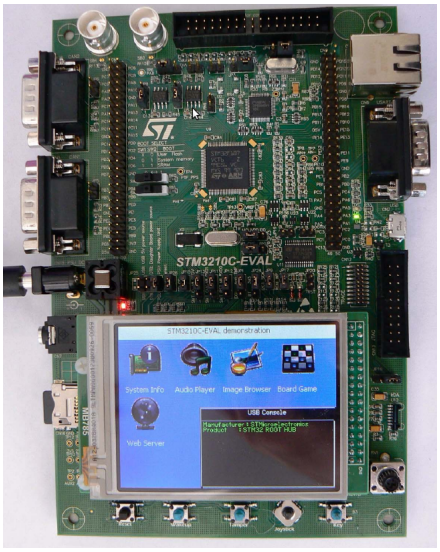


Figura A.11: Placa de desenvolvimento STM3210C-eval

A.10 Placa Panda board

Esta placa encontra-se montada no robô mas ainda não se encontra em funcionamento como trabalho futuro pretende-se portar toda o camada de software que no momento se encontra a ser executada por um PC convencional para esta placa. Esta placa tem um processador de arquitectura ARM Cortex A9 que funciona a frequência de 1.2 GHz com 1GB de memoria RAM .

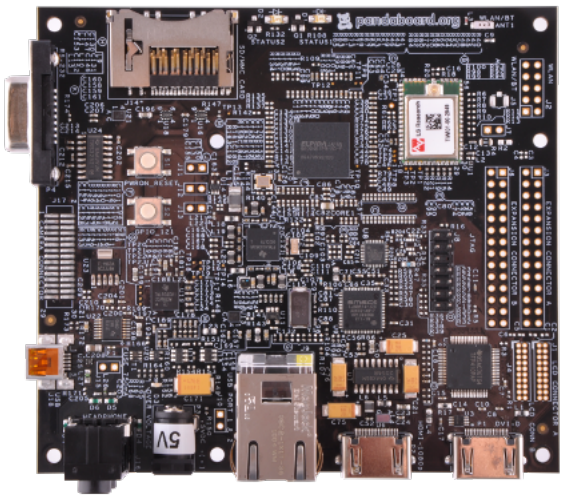


Figura A.12: Placa de desenvolvimento Panda Board