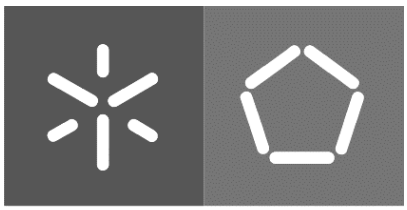
Universidade do Minho
Escola de Engenharia

André Manuel Fernandes Pereira
Nº 65152

Sistema de localização autocalibrado para robôs futebolistas

Guimarães, Junho de 2017



Universidade do Minho
Escola de Engenharia

André Manuel Fernandes Pereira
Nº 65152

Sistema de localização autocalibrado para robôs futebolistas

Dissertação de Mestrado em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do
Professor Doutor Agostinho Gil Teixeira Lopes

Guimarães, Junho de 2017

If you give up, your dreams and goals, they will be gone.

DECLARAÇÃO

André Manuel Fernandes Pereira

Endereço eletrónico: a65152@alunos.uminho.pt | andrekakapereira@gmail.com

Telefone: 918311135

Número do Bilhete de Identidade: 14182933

Título dissertação: Sistema de localização autocalibrado para robôs futebolistas

Orientador: Professor Doutor Agostinho Gil Teixeira Lopes

Ano de conclusão:2017

Mestrado integrado em Engenharia Eletrónica Industrial e Computadores

Universidade do Minho, ___ /___ /____

Assinatura: André Manuel Fernandes Pereira

AGRADECIMENTOS

O culminar de todo o trabalho desenvolvido aqui apresentado contou com importantes apoios e contribuições de algumas pessoas sem as quais este não teria sido possível e aos quais transmito desde já o mais sincero dos agradecimentos.

Ao meu orientador, Professor Doutor Gil Lopes pela sua orientação, por todo o apoio e confiança que em mim depositou, pelo saber transmitido e por todas as palavras de incentivo que me ajudaram a atingir o meu objetivo.

Ao Professor Doutor Fernando Ribeiro por toda a disponibilidade demonstrada, pela ajuda em encontrar soluções para problemas que iam surgindo, pelas opiniões e críticas, mas acima de tudo pela pessoa incansável e preocupado que demonstrou ser connosco.

Agradeço aos amigos que encontrei no **LAR,** Laboratório de Automação e Robótica, que partilharam comigo dias e noites de trabalho árduo de forma a alcançar o principal objetivo pretendido, realizar um jogo completo.

Por fim, o maior agradecimento de todos, aos meus pais Manuel Pereira e Maria Fernandes pela pessoa que me fizeram ser com a sua educação, por todo o apoio e por me possibilitarem estudar. À minha namorada por sempre ser o meu chão, ser o meu modelo de perseverança, por me presentear sempre com um sorriso, conselho ou incentivo quando necessário e por me ajudar a crescer profissionalmente e como pessoa todos os dias.

Sem vocês o trabalho aqui alcançado não seria possível.

O meu mais sincero obrigado a todos que estiveram do meu lado.

RESUMO

A obtenção de uma localização exata no ramo da robótica revela-se um dos pilares do sucesso e evolução da qualidade das tarefas desempenhadas pelos robôs, uma vez que se mostra fundamental para que um robô tenha a noção do espaço e do mundo que o rodeia. Esta importância, acentua-se no caso do futebol robótico, onde uma equipa de cinco robôs autónomos deve interagir entre si, num campo onde estão outros cinco robôs e diversos obstáculos.

Nesta dissertação pretende-se desenvolver um sistema de localização para robôs futebolistas, sendo o sistema em causa responsável por estimar a orientação, posição do robô no campo e modelação do mundo que o rodeia. É também responsável por efetuar uma avaliação da qualidade das imagens obtidas e, se necessário, alterar os parâmetros referentes à câmara de forma a melhorar a imagem obtida. Para além do descrito, pretende-se o desenvolvimento de uma ferramenta que possibilite a calibração de determinados parâmetros referentes ao robô e a todo o sistema enunciado.

A abordagem proposta para o cálculo da orientação do robô bem como da posição do mesmo no campo, assenta no processamento de imagem, uma vez que o campo oferece um sistema de referência inequívoco (em cada metade do campo), as linhas do campo. Os algoritmos desenvolvidos têm por base os pontos de linhas detetados, onde o algoritmo de cálculo da orientação constrói histogramas a partir destes pontos, sendo que o algoritmo de localização utiliza-os para comparar com um modelo do campo para obter a melhor estimativa, comparando a coordenada de cada ponto à distância à linha mais próxima desse ponto. É apresentado um método para auto calibração da câmara baseado em histogramas acerca das características que classificam uma imagem, de forma a avaliar a qualidade da mesma.

A qualidade de imagem apresentada, tanto antes de um jogo bem como durante, é sempre estável, devido ao algoritmo de auto calibração, possibilitando que o processo de localização do robô desenvolvido seja capaz de se localizar perfeitamente em qualquer campo de jogo, bem como de realizar uma modelação do mundo. O objetivo deste projeto foi atingido com sucesso, dispondo assim a equipa MINHO TEAM de diversas ferramentas que impulsionarão o sucesso da mesma.

Palavras-Chave: futebol robótico, localização, auto calibração, modelação do mundo.

ABSTRACT

Obtaining an exact location in the field of robotics is one of the pillars of the success and evolution of the quality of the tasks performed by the robots, since it is fundamental for a robot to have the notion of space and of the world that surrounds it. This importance is emphasized in the case of robotic football, in which a team of five autonomous robots must interact with each other, in a field where there are five other robots and several obstacles.

In this dissertation, we intend to develop a localization system for robot soccer players, being the system responsible for estimating the orientation, position of the robot in the field and modelling of the world that surrounds it. It is also responsible for evaluating the quality of the images obtained and, if necessary, for changing the parameters related to the camera in order to improve the image obtained. In addition to the described, it is intended the development of a tool that allows the calibration of certain parameters related to the robot and to the whole system.

The proposed approach for calculating the orientation of the robot as well as its position in the field is based on image processing, since the field provides a system of unequivocal reference (in each half of the field), the lines of the field. The algorithms developed are based on the points of lines detected, and the orientation calculation algorithm uses histograms constructed from the detected points and the localization method is based on the use of the distances of these points to the nearest field lines. A method is presented for auto-calibration of the camera based on histograms about the characteristics that classify an image, in order to evaluate the quality of the image.

The image quality presented both before and during a game is always stable due to the auto-calibration algorithm, allowing the localization process of the developed robot to be able to locate perfectly in any field of play as well as to perform a modeling of the world that surrounds it. The goal of this project was successfully achieved, thus providing the MINHO TEAM with several tools that will boost its success.

Key-Words: robotic football, localization, auto calibration, world modelling.

ÍNDICE

1. Introdução 1

1.1. RoboCup 1

1.1.1. RoboCup Middle-Size-League (MSL) 3

1.2. Problema 4

1.3. Objetivos 5

1.4. Motivação 6

1.5. Estrutura da Dissertação 7

2. Revisão Bibliográfica 8

2.1. Estado da arte 8

2.1.1. Métodos de localização 8

2.1.2. Métodos de auto calibração da câmara 12

2.2. Fundamentos teóricos 13

2.2.1. Filtro de Kalman 13

2.2.2. Robot Operating System - ROS 16

2.2.3. Processamento de Imagem 18

3. Metodologia 20

3.1. Sistema de visão 20

3.1.1. Emparelhamento câmara – espelho 21

3.1.2. Segmentação da imagem - Color Look Up Table 22

3.2. Auto calibração da câmara 24

3.3. Localização 28

3.3.1. Linhas de pesquisa na imagem – Scanlines 28

3.3.2. Deteção de linhas 30

3.3.3. Orientação 35

3.3.4. Mapeamento 40
3.3.5. Posição do robô no campo 42
3.4. World Model 51
3.4.1. Bola 51
3.4.2. Obstáculos 55
3.5. Comunicação 58
3.6. Vision Calib – Ferramenta de Calibração 59
3.6.1. LUT&MIRROR 60
3.6.2. PARAMETERS 61
3.6.3. RLE&BLOBS 62
4. Resultados 64
4.1. Auto calibração da câmara 64
4.2. Deteção das linhas 67
4.2.1. Filtragem 69
4.3. Posição do robô no campo 71
4.4. World Model 82
4.5. Vision Calib 90
4.6. Robótica 2017 90
5. Conclusão e trabalho futuro 92
5.1. Conclusão 92
5.2. Trabalho Futuro 93
6. Referências Bibliográficas 94

ÍNDICE DE FIGURAS

Figura 1 - Símbolo da RoboCup 1

Figura 2 - Campo da MSL 4

Figura 3 - Ciclos de operação do Filtro de Kalman 14

Figura 4 - Logótipo do ROS 16

Figura 5 - Arquitetura de um sistema baseado em ROS 16

Figura 6 - Exemplo de comunicações no ROS 18

Figura 7 - Sequência do Processamento Digital de Imagem 18

Figura 8 - Espelho catadióptrico 20

Figura 9 - Deformações captadas pela câmara 21

Figura 10 - Suporte da câmara 22

Figura 11 - Imagens obtidas após ajuste do novo suporte da câmara 22

Figura 12 – a) Imagem obtida; b) Imagem segmentada 23

Figura 13 - Máscara para processamento de imagem 24

Figura 14 - Fluxograma do algoritmo de auto calibração 26

Figura 15 - Fluxograma de ajuste da exposição e brilho do algoritmo de auto calibração 27

Figura 16 - Fluxograma de ajuste da saturação, brilho e white-balance do algoritmo de auto calibração 27

Figura 17 - Linhas de pesquisa radiais 29

Figura 18 - Linhas de pesquisa circulares 29

Figura 19 - Exemplo de scanline 30

Figura 20 - Exemplo do algoritmo de pesquisa 30

Figura 21 – a) Imagem obtida; b) Imagem com pontos de linha detetados 31

Figura 22 –Transições de linha detetadas 32

Figura 23 – a) Imagem com raio de luz; b) Imagem com caixa branca; c) Imagem com raios de luz e deteção desses raios como pontos de linha; d) Imagem com caixa branca e deteção da caixa como pontos de linha 33

Figura 24 - Plano e eixos coordenados de uma imagem 34

Figura 25 - Campo com sistema de eixos coordenados..... 36

Figura 26 - IMU adafrauit 10 DOF 37

Figura 27 - Fluxograma de obtenção da variação do encoder..... 38

Figura 28 - Exemplo do campo carregado para o programa 43

Figura 29 - Gráfico da equação do erro..... 45

Figura 30 – a) Imagem obtida; b) Funcionamento do algoritmo de Localização Global46

Figura 31 - Etapas do processo de localização global..... 47

Figura 32 – a) Imagem obtida; b) Funcionamento do algoritmo de Localização Local 48

Figura 33 - Etapas do processo de localização local 49

Figura 34 - Exemplo de scanline para deteção da bola 52

Figura 35 – a) Imagem obtida; b) Imagem Segmentada; c) Imagem com informação das scanlines; d) Imagem obtida com representação das transições detetadas; e) Mapeamento da bola no campo de jogo..... 53

Figura 36 - Fluxograma de validação da bola 55

Figura 37 - Exemplo de scanline para deteção de obstáculos 55

Figura 38 – a) Imagem obtida; b) Imagem Segmentada; c) Imagem com informação das scanlines; d) Imagem obtida com representação das transições detetadas; e) Mapeamento dos obstáculos no campo de jogo 57

Figura 39 - Arquitetura da rede ROS num robô..... 58

Figura 40 - Vision Calib tab LUT&MIRROR 61

Figura 41 - Vision Calib tab PARAMETERS..... 62

Figura 42 - Vision Calib tab RLE&BLOBS..... 63

Figura 43 – a) Imagem obtida antes do algoritmo de auto calibração ser aplicado; b) Imagem obtida depois do algoritmo de auto calibração ser aplicado..... 64

Figura 44 – a) Imagem obtida com scanlines radiais representadas e informação das mesmas; b) Imagem segmentada com scanlines radiais representadas e informação das mesmas; c) Mapeamento dos pontos de linha no campo de jogo..... 67

Figura 45 - a) Imagem obtida com scanlines radiais representadas e informação das mesmas; b) Imagem segmentada com scanlines radiais representadas e informação das mesmas; c) Mapeamento dos pontos de linha no campo de jogo..... 68

Figura 46 – a) Imagem obtida com pontos de linha errados devido ao feixe de luz; b) Imagem obtida com pontos de linha errados devido à presença da caixa branca no campo; c) Aplicação do filtro sobre a imagem com o feixe de luz; d) Aplicação do filtro sobre a imagem com a caixa no campo..... 70

Figura 47 - Posições escolhidas para testes..... 72

Figura 48 – a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição AM..... 73

Figura 49 – a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição BM..... 75

Figura 50 - a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição CM..... 76

Figura 51 - a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição DM com ruído propositado no campo..... 77

Figura 52 - a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição AA..... 78

Figura 53 - a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição BA..... 79

Figura 54 - a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição CA..... 80

Figura 55 – a) Imagem obtida; b) Imagem segmentada; c) Imagem com a informação contida nas scanlines; d) Mapeamento dos obstáculos a $1m$ e $2m$ em relação ao robô 83

Figura 56 - a) Imagem obtida; b) Imagem segmentada; c) Imagem com informações das scanlines comprovando a falha na deteção dos obstáculos ao perto; d) Mapeamento em relação ao robô; e) Mapeamento dos obstáculos no campo de jogo 84

Figura 57 – a) Imagem obtida; b) Imagem segmentada; c) Imagem com informações das scanlines; d) Mapeamento em relação ao robô; e) Mapeamento dos obstáculos no campo de jogo 85

Figura 58 – a) Imagem obtida; b) Imagem segmentada; c) Imagem com a informação contida nas scanlines; d) Mapeamento em relação ao robô..... 86

Figura 59 – a) Imagem obtida; b) Imagem segmentada; c) Imagem com informação contida nas scanlines; d) Mapeamento em relação ao robô..... 87

Figura 60 - a) Imagem obtida; b) Imagem Segmentada; c) Imagem com informação das scanlines; d) Imagem do mapeamento em relação ao robô; e) Mapeamento dos obstáculos e da bola no campo de jogo 89

Figura 61 - Símbolo do Robótica 2017 90

ÍNDICE DE TABELAS

Tabela 1 - Tempo de computação médio do algoritmo Perfect Match 10

Tabela 2 - Equações do Filtro de Kalman 15

Tabela 3 - Exemplo de vetor de filtragem 34

Tabela 4 - Exemplo de vetor de distância em pixéis 34

Tabela 5 - Equações do filtro de Kalman para fusão dos valores de orientação 40

Tabela 6 - Exemplo do vetor de distância real 41

Tabela 7 - Equações do filtro de Kalman para fusão dos valores de deslocamento em X 50

Tabela 8 - Tempos de ciclo do algoritmo de auto calibração com análise de diferentes quantidades de pixéis 65

Tabela 9 - Tempos de ciclo do algoritmo de auto calibração com análise de diferentes quantidades de pixéis quando não calibrado o white-balance 66

Tabela 10 - Valores de orientação de cada método na posição AM 72

Tabela 11 - Valores de orientação de cada método na posição BM 74

Tabela 12 - Valores de orientação de cada método na posição CM 76

Tabela 13 - Valores de orientação de cada método na posição DM 77

Tabela 14 - Valores de orientação de cada método nas posições AA, BA, CA 80

Tabela 15 - Tempo de processamento da localização 81

LISTA DE ACRÓNIMOS

LAR	Laboratório de Automação e Robótica
MSL	*Middle Size League*
ROS	*Robot Operation System*
IMU	*Inertial Measurement Unit*
REFBOX	*Referee Box*

1. INTRODUÇÃO

Neste capítulo é realizada uma introdução acerca do âmbito onde se insere esta dissertação, do problema que se pretende tratar, dos objetivos definidos, o que levou à realização desta dissertação e a estrutura desta dissertação de mestrado.

1.1. ROBOCUP

A ideia de robôs futebolistas foi pela primeira vez mencionada pelo Professor Alan Mackworth em 1992, num *paper* com o nome "On seeing Robots". Um ano depois, em 1993 é feito o anúncio oficial da criação do RoboCup e das suas regras. Desde então, várias foram as discussões acerca da organização, bem como de outros aspetos importantes para a criação do evento em conferências. No entanto, apenas em 1995 é anunciada a primeira Robo World Cup, RoboCup, que viria a ter lugar em 1997 [1].

Na Figura 1 é apresentado o símbolo da RoboCup.

Figura 1 - Símbolo da RoboCup

Assim, pode dizer-se que o ano de 1997, foi um ano de grande sucesso para o mundo da Robótica e da Inteligência Artificial pois foi marcado pela ocorrência de feitos importantes nestas áreas, como sejam a implantação do primeiro sistema autónomo em Marte, através da missão Pathfinder da NASA; o computador IBM Deep Blue que derrotou o campeão do Mundo de xadrez; bem como a apresentação do robô humanoide, modelo P-3, por parte da Honda.

Tal como se havia anunciado, a primeira edição da RoboCup realizou-se em 1997, em Nagoya, Japão, onde paralelamente a todos os acontecimentos supracitados, este evento começa a dar os primeiros passos no desenvolvimento de robôs futebolistas. Desde então, este encontro científico internacional é realizado anualmente sempre num país diferente. A RoboCup tem como objetivo proporcionar a educação e investigação em áreas de conhecimento como a robótica, a Inteligência Artificial e Visão por

Computadores. Porém, aquando da sua criação um objetivo bastante ambicioso foi proposto, que em 2050 uma equipa de robôs futebolistas fosse capaz de derrotar a seleção campeã do mundo.

Com o crescimento, aceitação e procura que esta competição fora alvo, a RoboCup expandiu-se para outras áreas, respondendo desta forma às necessidades demonstradas pela sociedade em tarefas pesadas ou que possuam qualquer tipo de perigo para o ser humano, sendo desta forma acrescentadas algumas ligas a esta competição. As ligas que constituem esta competição são:

→ RoboCup@Home tem como objetivo o desenvolvimento de soluções de assistência e de serviço para a área de robôs domésticos, visando a interação humano-robô.

→ RoboCup@Work e RoboCup Logistics trata-se de ligas onde o objetivo é o desenvolvimento de novas soluções para problemas encontrados em ambiente industrial.

→ RoboCup júnior visa estimular e integrar os jovens estudantes, do básico e secundário, no estudo de áreas como a robótica, incitando o desenvolvimento de valências em eletrónica, tanto software como hardware, o que poderá ajudar na escolha do ramo, a estudar ou trabalhar, no futuro.

→ RoboCup Soccer sendo esta a primeira modalidade na competição e também por ser a grande motivação na base da realização deste evento, está relacionada com os objetivos principais do evento. Esta liga tem como objetivos mais individualizados, a cooperação, coordenação, comunicação, entre outros, entre vários robôs, de forma a atingir ou concretizar uma determinada tarefa para a qual foram designados.

→ RoboCup Rescue visa fundamentalmente estimular o desenvolvimento de robôs para busca e salvamento de pessoas. Os robôs têm de ser capazes de se movimentar em cenários de desastre, mas também de cooperar, coordenar e comunicar com outros agentes de forma a tornar a concretizar o salvamento ou busca para o qual foram designados.

É de salientar que algumas destas competições estão dividas em diferentes ligas. A RoboCup possui, além destas competições, conferências cientificas, de forma a possibilitar, aos investigadores participantes, apresentar as suas investigações e resultados obtidos.

1.1.1. RoboCup Middle-Size-League (MSL)

A MSL, Middle-Size-League, é umas das ligas que constituí a RoboCup Soccer, tendo sido criada aquando do início desta competição, uma vez que constitui a base do projeto inicial. Esta tem como objetivo a simulação de um jogo de futebol, onde os jogadores são robôs, sendo que um dos requisitos é que estes, os robôs participantes no jogo, sejam completamente autónomos, ou seja, desde o apito inicial até ao apito final, estes joguem sem qualquer intervenção humana.

O futebol, como é de conhecimento geral, é o desporto mais popular e praticado em todo o mundo, apresentando vários desafios científicos, como a cooperação, coordenação e comunicação entre robôs. Para tal, é necessário o robô possuir algumas informações, como a sua própria posição no campo, o local onde se situam os adversários e colegas de equipa, entre outros. Desta forma o futebol é como um instrumento que estimula, incentiva, a investigação e o desenvolvimento nas áreas supracitadas.

Na liga em questão as equipas são constituídas por 5 elementos, 1 guarda-redes e 4 jogadores de campo, os robôs apenas podem pesar 40 quilogramas no máximo, não exceder os 80 centímetros de altura e 52 centímetros de diâmetro. O jogo em si é dividido em duas partes, de 15 minutos cada, com 5 minutos de intervalo. A par disto, existem regras logo, existem igualmente faltas, sendo as mesmas assinaladas por um humano. O campo de jogo possui 18 metros de comprimento e 12 metros de largura, possuindo ainda algumas marcas, além das linhas de campo, com o intuito de ajudar na localização no campo por parte dos robôs. O campo com as medidas correspondentes pode ser observado na Figura 2.

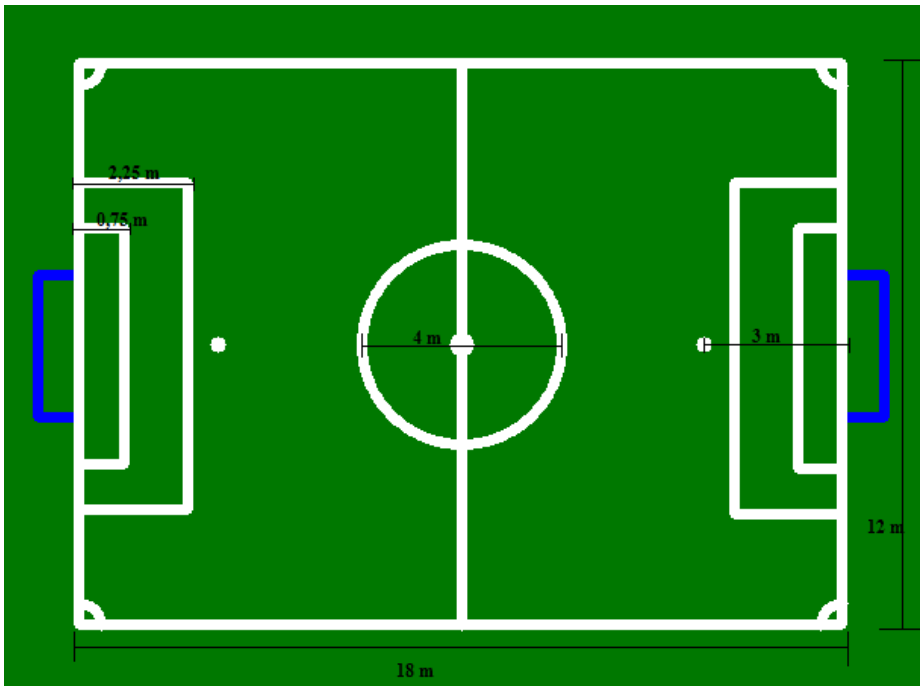


Figura 2 - Campo da MSL

1.2. PROBLEMA

Nos jogos de futebol em geral o trabalho em equipa é o principal ingrediente para atingir o sucesso, sendo que este assenta na coordenação, cooperação e comunicação entre a equipa, bem como com o treinador. O papel de treinador no futebol robótico é representado pela *BaseStation*, a qual é responsável pela definição das táticas, ou seja, é ela que define a posição de cada robô no campo consoante o estado do jogo. Para que tal seja possível, é necessário que esta possua conhecimento da posição de todos os elementos participantes num jogo, sendo neste ponto que entra a localização pois, esta recolhe e envia para a *BaseStation* diversa informação.

A localização é responsável por possibilitar o robô jogar, esta tem de ser capaz de identificar a posição e orientação do mesmo, identificar a bola e os outros robôs que se encontram no campo, calculando a posição dos mesmos relativamente a ele, recriando o ambiente que rodeia o robô, sendo esta informação posteriormente enviada para a *BaseStation*. Um robô futebolista, sem a noção da sua posição, não consegue jogar, intercetar a bola e rematar à baliza, não é capaz de cumprir qualquer tipo de ordem, ou estratégia, que a *BaseStation* tenha definido não contribuindo para a cooperação e coordenação da equipa nos processos ofensivos nem defensivos, não consegue respeitar as regras, não evitando colisões, entre outros.

Posto isto, é de notar que a localização, o processo no seu todo, é uma das, senão a, tarefa mais importante a ser realizada, sendo que uma falha leva à construção errada sobre o ambiente em que o robô se encontra o que posteriormente leva à atribuição de um movimento ou posicionamento errado por parte da *BaseStation* para a equipa ou para um robô em especifico. Porém este processo não é fácil de ser realizado pois, existem diversas variáveis que o podem afetar, como qualquer distorção ou deformação na imagem obtida, um *bug* ou erro na programação, sendo uma das variáveis mais críticas a alteração da luminosidade. Portanto, um sistema de localização autocalibrado é de enorme importância e necessidade.

1.3. OBJETIVOS

O principal objetivo desta dissertação de mestrado passou por colmatar as dificuldades apresentadas pela equipa MINHO TEAM nas últimas competições, durante as quais, a equipa não se demonstrou suficientemente competitiva.

Assim propôs-se desenvolver um método de localização para a equipa, capaz de fornecer informações acerca da posição e orientação do robô em campo o mais precisas possível e do mundo que o rodeia. Os objetivos propostos são:

- → Estudo das características que definem uma imagem com qualidade e algoritmos de auto calibração da câmara;
- → Correção das deformações e distorções obtidas pelo emparelhamento câmara-espelho;
- → Auto calibração dos parâmetros da câmara com recurso a um algoritmo;
- → Estudo dos algoritmos de localização desenvolvidos por outras equipas presentes na RoboCup MSL;
- → Identificar a orientação do robô no campo de jogo;
- → Auto localização com recurso ao processamento de imagem e odometria;
- → Criação/Contrução do *world model*:
    - → Identificação e *tracking* da bola de jogo;
    - → Identificação de obstáculos na vizinhança do robô e obter a sua localização;
- → Envio de todas as informações relevantes, como as coordenadas do robô no campo, as coordenadas da bola bem como as coordenadas das entidades identificadas, utilizando o ROS, *Robot Operating System*.

É também objetivo desenvolver uma ferramenta capaz de tratar de todos os aspetos referentes à calibração do robô. Pretende-se com o desenvolvimento desta, oferecer a qualquer utilizador que deseje calibrar o robô uma ferramenta simples de utilizar e sem a necessidade de recompilar código para aplicação das alterações realizadas. O software em questão possui os seguintes objetivos:

- → Calibração do centro da câmara em relação ao espelho;
- → Calibração da *look up table*, necessária para a classificação dos pixéis da imagem obtida como entidades que o robô conhece (bola, campo, linhas, obstáculos, entre outros);
- → Alteração ou criação da máscara para cada robô;
- → Calibração dos valores das distâncias calculadas em relação aos pixéis;
- → Alteração dos valores de referência utilizados no método de auto calibração da câmara, bem como dos parâmetros PID do controlador.
- → Alteração do valor das propriedades da câmara caso se pretenda realizar algum tipo de teste;
- → Alteração dos parâmetros das linhas de pesquisa*;*
- → Calibração dos parâmetros do algoritmo de deteção das linhas, obstáculos e bola;
- → Calibração do vetor utilizado para filtragem de pontos de linha não válidos;
- → Alteração de algumas variáveis referentes à criação do *world model*, como raio de pesquisa para criação dos obstáculos que rodeiam o robô;
- → Alteração dos pesos do filtro de Kalman referentes à fusão da odometria com a localização por visão;

Um dos objetivos fulcrais desta dissertação de mestrado é que todo o software que venha a ser desenvolvido seja modular, isto é, caso se pretenda acrescentar ou retirar funcionalidades, estas sejam fáceis de implementar pois, pretende-se com isto criar uma base estável sobre a qual, equipas futuras, possam vir a acrescentar novas funcionalidades ou métodos.

1.4. MOTIVAÇÃO

O que motivou o desenvolvimento desta dissertação foi o facto da equipa MINHO TEAM apresentar falhas na localização dos seus robôs, mas também pela vontade que a equipa demonstrou possuir em termos de voltar a competir e de mostrar o seu valor tanto a nível nacional como internacional. Para tal, é necessário que a localização e todas as

tarefas adjacentes a esta, sejam o mais robustas, estáveis e precisas possível. A motivação prende-se também com o objetivo principal da RoboCup, que visa a educação e investigação em áreas de conhecimento, como a robótica e a Inteligência Artificial [2].

1.5. ESTRUTURA DA DISSERTAÇÃO

No primeiro capítulo, Introdução, é realizada uma breve introdução à RoboCup, dando a conhecer algumas das divisões da mesma e também onde o tema desta dissertação se insere. São também explicados os problemas referentes a esta dissertação de mestrado e ainda são definidos os objetivos e o que motivou a realização desta dissertação.

No capítulo dois, Revisão Bibliográfica, constam os conceitos que serão utilizados no desenvolvimento desta dissertação, que métodos de localização são utilizados na liga MSL, identificando as vantagens e desvantagens de cada um e ainda que métodos de auto calibração das câmaras poderiam ser utilizados.

No terceiro capítulo, Metodologia, são apresentadas que metodologias que irão ser utilizadas e como são aplicadas no trabalho desenvolvido nesta dissertação. Este aborda o sistema de visão do robô pois, este é de enorme importância, uma vez que é a partir desta que se obtém o maior número de informação, é também analisado todo o processo de localização do robô no campo, desde a definição das linhas de pesquisa até à obtenção das coordenadas do robô no campo. O terceiro capítulo aborda ainda a criação/modelação do mundo que rodeia o robô, explicada a arquitetura da rede de comunicação desenvolvida e apresenta o *Vision Calib* dando a conhecer todas as suas funcionalidades.

No capítulo Resultados, quarto capítulo, são apresentados os resultados obtidos, bem como são explicadas algumas alterações que necessitaram de ser introduzidas de forma a colmatar falhas encontradas.

Por fim, quinto capítulo, são apresentadas as conclusões obtidas, bem como as sugestões de trabalho futuro que se propõe para a equipa.

2. REVISÃO BIBLIOGRÁFICA

No capítulo 2, Revisão Bibliográfica, são apresentados o Estado da Arte e os Fundamentos teóricos.

No subcapítulo Fundamentos teóricos, 2.2, são apresentados alguns conceitos chave que serão utilizados na implementação e desenvolvimento da dissertação, sendo que no subcapítulo Estado da Arte, 2.1, são apresentados os diversos métodos de obtenção da posição do robô no campo, desenvolvidos até aos dias de hoje e utilizados na MSL. Neste é feita uma comparação entre os diferentes métodos de forma a perceber qual o método que apresenta melhores resultados e no subcapítulo, 2.1.2, é feita uma apresentação acerca dos métodos de auto calibração da câmara e qual o método escolhido e as razões que levaram à sua escolha.

2.1. ESTADO DA ARTE

2.1.1. Métodos de localização

Como já fora supracitado, no subcapítulo Métodos de localização são apresentados alguns métodos de obtenção da posição do robô no campo e é realizada uma comparação entre eles de forma a averiguar qual o melhor método a ser aplicado.

2.1.1.1. Filtro de Partículas

Filtro de Partículas também denominado de método de Monte Carlo, é um método estocástico que se baseia em executar sucessivas simulações, um elevado número de vezes, de forma a calcular probabilidades heuristicamente. Este método é utilizado em simulações estocásticas, sendo que é muito aplicado em áreas como a física, matemática e biologia [3].

Filtro de Partículas é a distribuição de probabilidades de uma posição por um conjunto de N amostras aleatórias, onde um conjunto de partículas forma uma aproximação discreta de uma distribuição de probabilidade. Partículas são pares ordenados, possuindo posição e peso, (x^i, π^i), para $i = 1, \dots, N$, onde $x = (x, y, \theta)$ e $\pi \geq 0$ definem o fator de importância, análogo a uma distribuição de probabilidade discreta com $\sum_{i=1}^{N} \pi^i = 1$. O conjunto de partículas é escrito da seguinte forma,
$X = \{(x^1, \pi^1), (x^2, \pi^2), \dots (x^N, \pi^N)\}$ [4].

A função de distribuição de probabilidade inicial da posição do robô define, indica, o conhecimento inicial sobre a sua posição. No caso de o robô desconhecer totalmente a sua posição inicial, a distribuição de probabilidade é inicializada com uma distribuição uniforme sobre todas as posições possíveis, ou seja, o conjunto de partículas é inicializado escolhendo N partículas distribuídas uniformemente no ambiente e os pesos são também distribuídos uniformemente [4].

Quando o robô executa um comando de movimento **a,** o Filtro de Partículas gera N novas partículas que aproximam a posição do robô, após o movimento, sendo este conjunto indicado por $\bar{X}$. Cada nova partícula (x^i, π^i), para $i = 1, \dots, N$, é dada por: $x^i \sim p(x^i | x^i, a)$ [4].

Quando o robô realiza uma observação **o** do ambiente, esta observação é integrada pelo Filtro de Partículas, atualizando o valor do fator de importância, $x_t^i = p(\boldsymbol{o}|x^i)$, sendo que este ajuste atribui pesos maiores às partículas que correspondem às posições onde **o** é mais provável de ser observado [4].

A fase seguinte é a de reamostragem ou amostragem por importância, que faz com que o conjunto de partículas $\bar{X}$ se transforme num novo conjunto, possuindo o mesmo tamanho de X, em que as novas partículas são distribuídas pelos seus fatores de importância, sendo que ao integrar os pesos das partículas nesta nova distribuição, a distribuição muda, fazendo com que as partículas se concentrem em maior número nas regiões onde há maior peso.

O algoritmo de localização utilizando Filtro de Partículas apresenta várias hipóteses sobre a posição do robô, sendo este capaz de se localizar globalmente, mas este é ineficiente para recuperar de falhas de localização. Uma das características mais interessantes do Filtro de Partículas é este possuir uma forma de adaptar o peso computacional, ou seja, este permite adaptar o número de partículas que se pretende utilizar, de forma a ajustar-se aos recursos computacionais que se possui.

2.1.1.2. *Perfect Match*

O algoritmo *Perfect Match* para a localização de robôs futebolistas na MSL foi desenvolvido pela equipa *Brainstormers Tribots*, em 2004. Para o desenvolvimento deste algoritmo [5] a equipa teve de enfrentar três grandes dificuldades sendo elas, que o algoritmo devia ser robusto, a posição estimada devia ser precisa, mesmo quando existem diversas variáveis que afetam a qualidade da auto localização, e devia possuir um peso

computacional eficiente pois, a informação acerca da localização afeta a variados processos referentes a outras tarefas que o robô tem a desempenhar.

Após a captura das imagens, estas são pré-processadas utilizando um detetor, que deteta as linhas brancas do campo, tendo por base uma pesquisa eficiente ao longo de linhas radiais pré-definidas. O resultado desta pesquisa é uma lista de pontos, posição onde as linhas radiais intercetam as linhas brancas, com coordenadas relativas ao centro do robô [6].

Consecutivamente, de forma a encontrar a posição do robô em relação à informação obtida, um algoritmo é aplicado, este estima a posição do robô, avaliando a qualidade, da sua estimativa, com base numa função de erro, sendo que esta função de erro de modo a ser definida tem em conta o ruído que afeta os pontos de linha detetados.

A ideia do algoritmo é fazer com que na posição estimada, os pontos de linhas detetados correspondam o melhor possível com as linhas de campo conhecidas, sendo que desta forma, maximizando a correspondência, minimiza-se o erro produzindo assim uma melhor estimativa da real posição do robô [5].

Foram realizados alguns testes por parte da equipa, onde esta pode concluir que o algoritmo desenvolvido foi de encontro ao pretendido, sendo que este possui um baixo peso computacional, robustez e alta precisão. Como fora dito anteriormente, o desenvolvimento deste algoritmo por parte da equipa *Brainstormers Tribots*, vinha no âmbito de se pretender diminuir o peso computacional sem descurar a robustez e eficiência do processo, sendo que o objetivo foi superado pois, quando comparado com outros processos anteriores de localização, neste caso o filtro de partículas, este algoritmo apresenta um tempo de computação bastante reduzido, como pode ser verificado na tabela Tabela 1 [5].

Tabela 1 - Tempo de computação médio do algoritmo Perfect Match

Método	**Tempo de computação médio por ciclo (ms)**
Filtro de Partículas com 500 partículas	48.3
Filtro de Partículas com 200 partículas	17.9
Perfect Match	4.2

Observando a Tabela 1, é de notar que o algoritmo desenvolvido pela equipa superou por muito o tempo médio de computação dos outros métodos [5]. É de salientar que estas medições foram realizadas com as mesmas condições.

No algoritmo do *Perfect Match* o tempo de computação está intrinsecamente ligado ao número de pontos de linha que são detetados e também ao número de linhas radiais utilizadas, sendo que é possível definir tanto um número máximo de pontos de linha bem como de linhas radias pré-definidas a usar.

2.1.1.3. Conclusões

Os métodos de localização existentes, estes foram resumidos a dois métodos, sendo eles o filtro de partículas, este é usado com bastante frequência quando se fala de localização de robôs no seu geral, e o algoritmo *Perfect Match* desenvolvido, como já fora dito pela equipa *Brainstormers Tribots*.

Poderiam ser usados ainda outros métodos, como Localização de Markov ou até mesmo o Filtro de Kalman, para cálculo da posição do robô, mas estes foram postos de parte pois, o Filtro de Kalman [4] apenas apresentaria uma hipótese sobre a localização do robô, sendo ainda incapaz de se localizar globalmente bem como de recuperar de falhas, a Localização de Markov é um método com elevado peso computacional o que vai contra os objetivos propostos para o desenvolvimento desta dissertação [4].

Porém, o filtro de Kalman será utilizado para realizar a fusão das posições do robô obtidas de forma a possibilitar uma estimativa mais precisa.

Comparando o Filtro de Partículas e a Localização de Markov, como o Filtro de Partículas permite definir o número máximo de partículas que se pretende utilizar, este é capaz de apresentar menor peso comparativamente à Localização de Markov [4].

De acordo com os resultados apresentados pela equipa *Brainstormers Tribots*, o algoritmo *Perfect Match* em comparação com o Filtro de Partículas é bastante mais rápido pois, consegue resolver o problema da localização em apenas $4.2\,ms$, sendo que aplicando o filtro de partículas, com 200 partículas, este obteve um tempo de $17.9\,ms$, onde comparando os dois, o *Perfect Match* é deveras mais rápido, nunca comprometendo a sua fiabilidade [5].

Para decisão acerca de qual método de localização apresentava ser o mais eficiente e robusto, foi feita ainda uma pesquisa por equipas da MSL, onde desta forma foi possível comprovar que o algoritmo desenvolvido pela equipa *Brainstormers Tribots*, é utilizado por praticamente todas as equipas da liga. Permitindo concluir que o algoritmo *Perfect Match* é, até ao dia de hoje, o algoritmo que apresenta melhores resultados.

2.1.2. Métodos de auto calibração da câmara

Para auto calibração das cores de uma câmara existem fundamentalmente duas formas, por hardware ou por software. Porém, a auto calibração por hardware não é uma opção viável neste caso, devido à complexidade do sistema da câmara. Assim sendo, a única possibilidade restante diz respeito à auto calibração por software.

Acerca de formas de calibração automática das cores de câmaras, chegou-se à conclusão que existe alguma informação acerca da forma como se poderia realizar este processo, mas em diversos casos esses métodos envolvem tabuleiros de xadrez com diversas cores, ou outros objetos que de alguma forma possibilitam e facilitam a calibração.

Porém, o que se pretende neste caso é um método capaz de realizar a calibração de cores da câmara sem necessitar de recorrer a objetos externos, apenas baseado na imagem obtida e que possua pouco tempo computacional, isto para ser possível a sua utilização em tempo real uma vez que a distribuição de luz num campo de jogo pode variar.

Portanto, devido ao explicado acima, optou-se por utilizar um método desenvolvido pela equipa CAMBADA da Universidade de Aveiro, que também participa na liga MSL da RoboCup [7].

O método proposto configura os parâmetros colorimétricos mais importantes das câmaras, sendo eles o ganho, a exposição, o gama, o *white-balance*, o brilho, a saturação e a nitidez, sem ser necessário a interação de um humano para calibrar estes parâmetros utilizando apenas as imagens obtidas por parte da câmara. O algoritmo proposto adquire várias imagens e calcula diversas medições que visam permitir a calibração dos parâmetros colorimétricos.

De forma a calibrar todos os parâmetros, o algoritmo começa por calibrar os parâmetros relacionados com a luminância na imagem, sendo eles o ganho, exposição e gama. Para tal é calculado o histograma da luminância presente numa imagem e é calculada a sua média pois, para uma câmara devidamente calibrada o valor da sua média deve ser de 127, para 8 bits por pixel, se o valor de média obtido não se aproxime deste é ajustado a exposição e o ganho, sendo que, caso a alteração destes dois não seja suficiente só então será feita a alteração no gama [7].

Durante o ajuste da luminância com recurso aos parâmetros supracitados, é também feito o ajuste dos outros parâmetros caso a média calculada da luminância se encontre entre valores razoáveis, ou seja, entre 101 e 152, o algoritmo inicia a calibração dos outros

parâmetros. Começa por calcular o histograma da saturação e calcula também a sua média, calcula a média de U e V numa área branca pré-definida e calcula também a média de R, G, B numa área preta pré-definida [7].

A média do histograma da saturação, tal como o da luminância, deve estar o mais perto possível de 127, sendo ajustado o seu valor se necessário. O valor de U e V numa área branca devem possuir o valor de 127 e os valores de R, G, B de uma área preta devem possuir o valor de 0. Cada um destes é ajustado até que o seu valor convirja para o seu valor de referência [7].

O algoritmo é executado até que nenhum parâmetro seja alterado. É de salientar que é realizado um controlo PID sobre os parâmetros de forma a possibilitar uma melhor convergência para os valores de referência.

2.2. FUNDAMENTOS TEÓRICOS

Como já fora referido, no subcapítulo Fundamentos teóricos, 2.2, são apresentados conceitos chave aplicados nesta dissertação, como Filtro de Kalman, *Robot Operating System* e Processamento de imagem.

2.2.1. Filtro de Kalman

Rudolf Kalman foi o criador do método matemático denominado por Filtro de Kalman. Desde essa altura, este tem sido alvo de intensas investigações e aplicado em diversas áreas, como a robótica, devido à evolução dos computadores, mas também por este apresentar pequeno peso computacional e por ser um estimador ótimo para sistemas lineares com erro Gaussiano.

O Filtro de Kalman [8][9] é um algoritmo de estimação de estado, usado muito frequentemente para conseguir a fusão de dados de fontes múltiplas. O filtro associa o modelo de processo, modelos de ruído e medições em tempo real de forma a obter estimativas dos estados de um processo linear dinâmico.

A sua aplicação depende da relação linear entre as medidas e os estados, que são corrompidos pelo ruído gaussiano branco (média zero e covariância nula). O filtro fornece uma solução recursiva onde o estado é atualizado computando juntos a estimativa anterior e os dados observados.

O algoritmo deriva de dois modelos de espaço-estado, ambos importantes para entender os parâmetros de Kalman e o processo de estimativa, sendo eles o modelo de processo e o modelo de medições.

$$x_k = Ax_{k-1} + BU_{k-1} \quad (1)$$

No modelo do processo Equação 1, A é chamado de matriz do sistema e B matriz de entrada. Ambas podem ser obtidas pelo estudo do modelo matemático do processo e x é o vetor de estados e é definido como sendo o conjunto mínimo de dados que descreve unicamente o comportamento dinâmico do sistema.

$$z_k = Hx_k + v_k \quad (2)$$

No modelo de medições Equação 2 a matriz H é chamada de matriz de medidas e z é o estado observável que corresponde a uma representação de x obtido do sensor. O modelo está corrompido com ruído branco gaussiano, definido como v, sendo que este possui média zero e covariância de matriz R.

Antes de implementar o filtro, os modelos referidos devem ser concluídos, juntamento com a definição das matrizes $A, B, H, Q\ e\ R$. Q e R representam a covariância dos modelos de ruído e, portanto, um valor baixo de Q reflete mais "confiança" no processo, ou seja, modelo do processo corrompido com menos ruído, e um valor baixo de R mais confiança no sensor, modelo de medições, sendo que estes valores, tanto de Q como de R podem ser definidos inspecionando o modelo diretamente, caso este seja conhecido.

O Filtro de Kalman é constituído por duas fases de *update*, sendo elas a fase de Previsão (com equações de *update* do tempo) e a Correção (com equações de *update* das medições), Figura 3.

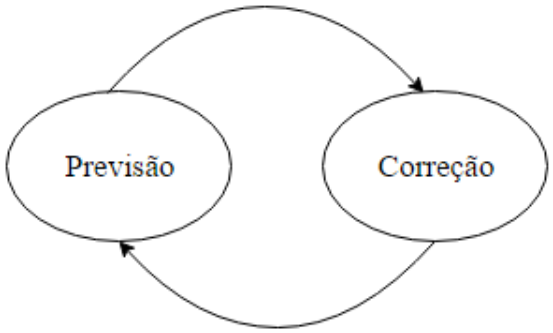


Figura 3 - Ciclos de operação do Filtro de Kalman

Na primeira fase do algoritmo de Kalman, as equações de Previsão são responsáveis por estimar o estado, $\hat{x}_k^-$, e a covariância do erro, P_k^-, à priori.

$$x_k = Ax_{k-1} + BU_{k-1} \tag{3a}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{3b}$$

As equações de Correção são responsáveis por corrigir, a estimativa obtida pelas equações de Previsão, tendo por base o modelo de medições e o modelo. Os valores obtidos anteriormente, $\hat{x}_k^-$ e P_k^- são agora usados juntamento com R para achar o ganho do Filtro de Kalman, K_k . O valor de ganho é uma medida do peso do modelo, este traduz a confiança concedida pelos valores de Q e R. O ganho, K_k , é então usado para atualizar o valor esperado $\hat{x}_k$ e também para atualizar a matriz de covariância do erro, P_k , juntamente com o seu valor anterior.

É de notar que ao fazer esta correção a covariância do estimado é minimizada, sendo esta a razão do Filtro de Kalman ser um estimador ótimo.

As equações das duas fases de *update* do Filtro de Kalman podem ser visualizadas na Tabela 2.

Tabela 2 - Equações do Filtro de Kalman

Previsão
Estimativa do estado $\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} = \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{k-1}} + \boldsymbol{BU}_{\boldsymbol{k-1}}$
Estimativa da covariância $\boldsymbol{P}_{\boldsymbol{k}}^{-} = \boldsymbol{AP}_{\boldsymbol{k-1}}\boldsymbol{A}^{\boldsymbol{T}} + \boldsymbol{Q}$
Correção
Cálculo do ganho de Kalman $\boldsymbol{K}_{\boldsymbol{k}} = \boldsymbol{P}_{\boldsymbol{k}}^{-}\boldsymbol{H}^{\boldsymbol{T}}\left(\boldsymbol{HP}_{\boldsymbol{k}}^{-}\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R}\right)^{-\boldsymbol{1}}$
Correção do estado $\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} + \boldsymbol{K}_{\boldsymbol{k}}\left(\boldsymbol{z}_{\boldsymbol{k}} - \boldsymbol{H}\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}\right)$
Correção da covariância $\boldsymbol{P}_{\boldsymbol{k}} = \left(\boldsymbol{I} - \boldsymbol{K}_{\boldsymbol{k}}\,\boldsymbol{H}\right)\boldsymbol{P}_{\boldsymbol{k}}^{-}$

No fim de cada sequência, Previsão e Correção, o processo é repetido, ou seja, $\hat{x}_k$ e P_k , da Correção, passam a ser $\hat{x}_{k-1}$ e P_{k-1} na Previsão, sendo que esta recursividade é umas das características mais interessantes do Filtro de Kalman.

2.2.2. Robot Operating System - ROS

O ROS é um sistema operativo *open-source* para robôs [10]. Em meados do ano 2000, a Universidade de Stanford criou protótipos flexíveis, em termos de sistemas de software dinâmicos, com a intenção de os usar em robôs. Em 2007 a Willow Garage, uma incubadora de robótica, proporcionou diversos recursos de forma a serem criadas implementações do ROS, já devidamente testadas em robôs [10].

Figura 4 - Logótipo do ROS

O ROS como se trata de um sistema operativo para robôs apresenta diversas características de enorme importância, destacando-se a sua flexibilidade, robustez, funcionamento distribuído, velocidade de comunicação e ainda oferecer prioridade a todas as comunicações, permitindo desta forma o desenvolvimento de sistemas complexos e modulares, oferecendo abstração e simplicidade em comparação à implementação de protocolos de comunicação já existentes.

O seu funcionamento é baseado em nós, o que permite que cada nó realize a sua tarefa independentemente dos outros. Estes nós são processos distintos.

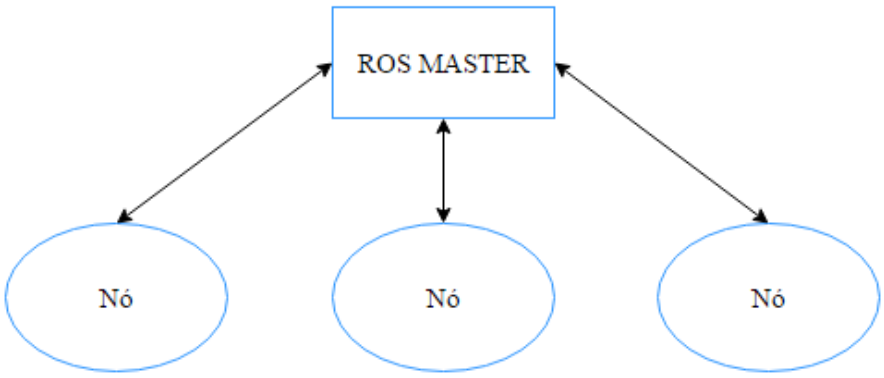


Figura 5 - Arquitetura de um sistema baseado em ROS

O sistema ROS possui um *Master*, responsável pela configuração da rede, e por um conjunto de nós, os quais são responsáveis por desempenhar as tarefas para as quais foram criados, sendo que cada nó pode ser interpretado como um processo diferente.

De forma a possibilitar a comunicação entre processos, o ROS possui duas formas distintas, sendo elas as mensagens e os serviços. Tanto as mensagens como os serviços, de forma a serem "trocados" entre processos, é necessário que exista um tópico do tipo de mensagem ou serviço a ser enviado, tanto no emissor como no recetor, definindo desta forma o "caminho" que essa informação irá tomar. O nome dos tópicos funciona como um identificador do tipo de dados que nele circulam pois, estes apenas fazem circular um tipo de dados e apenas um.

No caso das mensagens o paradigma de comunicação é de muitos para muitos, *broadcast*. Um nó que esteja interessado num certo tipo de mensagem subscreve o tópico dessa mensagem recebendo desta forma as mensagens que circulam neste tópico, ou publicam no tópico.

Um tópico pode ter vários *subscribers* e vários *publishers* e um nó pode subscrever ou publicar, ou ambos, num tópico ou em vários tópicos. Quando um tópico recebe uma mensagem por parte de qualquer um dos seus *publishers*, este envia a nova mensagem para todos os seus *subscribers*.

Os serviços funcionam com o paradigma de comunicação de *request/reply*, possibilitando a comunicação de um para um. Um nó oferece o serviço com um determinado nome e o cliente, outro nó, usa o serviço com o nome definido, enviando uma mensagem de *request* e aguarda pela resposta, *reply*. Por outra palavras o cliente envia um pedido ao servidor, usando o paradigma de *request/reply*, enviando informação acerca do que pretende que seja obtido como resposta, onde de seguida o servidor responde com a informação pretendida. O tipo de informação a ser trocada é definido aquando da definição de que tipo de serviço se pretende utilizar.

É de salientar que as mensagens utilizadas nos tópicos podem conter qualquer tipo de dados, sendo que estas mensagens podem ser parte integrante na informação trocada nos serviços, ou seja, podem ser interpretadas como um novo tipo de dados.

Como se pode observar o ROS apresenta um conjunto de características e funcionalidades deveras uteis para o desenvolvimento de robôs autónomos, possibilitando um nível de abstração e modelação formidáveis quando se pretende realizar comunicações entre baixo nível e alto nível.

Um exemplo da arquitetura de comunicação ROS entre nós pode ser visualizado na Figura 6.

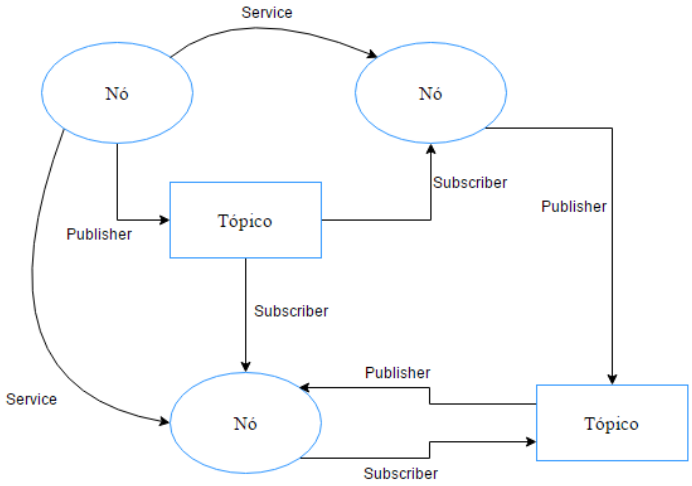


Figura 6 - Exemplo de comunicações no ROS

2.2.3. Processamento de Imagem

O processamento de imagem trata-se de uma forma de processamento de dados, onde a entrada e saída são imagens. Nesta dissertação referimo-nos mais precisamente ao processamento digital de imagem pois, além de este ser mais preciso e confiável é também mais fácil de implementar do que o processamento de imagem analógico o que requereria adição de eletrónica ao sistema de aquisição da câmara.

O processamento digital de sinal consiste no uso de algoritmos em imagens, de forma a alterar características ou obter informações da mesma, como a alteração da resolução da imagem, alteração das cores da imagem, redução do ruído, deteção de formas, construção de histogramas, aplicação de filtros, entre outras, sendo que este processamento é realizado num computador com software para o efeito. Um exemplo da sequência de processamento de imagem pode ser observado na Figura 7.

Figura 7 - Sequência do Processamento Digital de Imagem

Com o desenvolvimento de computadores cada vez mais rápidos, o processamento digital tornou-se numa mais valia para distintas áreas, sendo que é na robótica que esta se destaca possibilitando funcionalidades de uso imprescindível.

Existem diversas bibliotecas que oferecem a possibilidade de executar processamento em imagens, como CImg Library, Simd, Skia, Imagemagick, Opencv, Boost GIL, entre outas. Optou-se por trabalhar com o Opencv [11] pois, é uma biblioteca *open source*, podendo utiliza-la livremente, já deu provas das suas poderosas capacidades, é estável, robusta, e possui uma extensa lista de métodos de processamento de imagem.

É de notar que quando falamos de processamento de imagem referimo-nos a processamento de vídeo pois, o vídeo é um conjunto de imagens retiradas com pequeno espaço temporal entre cada uma.

3. METODOLOGIA

No capítulo 3, Metodologia, é apresentado todo o trabalho desenvolvido no âmbito desta dissertação. Este inicia-se com o subcapítulo Sistema de visão, 3.1.

Um dos principais fatores implicado numa boa localização é a obtenção de uma imagem com qualidade por parte do robô, uma vez que, se esta apresentar deformações ou distorções, demasiada ou comedida claridade, irá provocar erros na informação obtida. Desta feita, surge a necessidade de analisar o sistema de visão que os robôs possuem de forma a dotá-los de uma imagem que retrata a realidade do ambiente em que se encontram.

3.1. SISTEMA DE VISÃO

O sistema de visão que a MINHO TEAM utiliza é o sistema que praticamente todas as outras equipas a competir na liga MSL utilizam [12], que consiste, essencialmente, na utilização de uma câmara, *Point Grey BlackFly*, a apontar para o centro de um espelho convexo, como se pode observar na Figura 8.

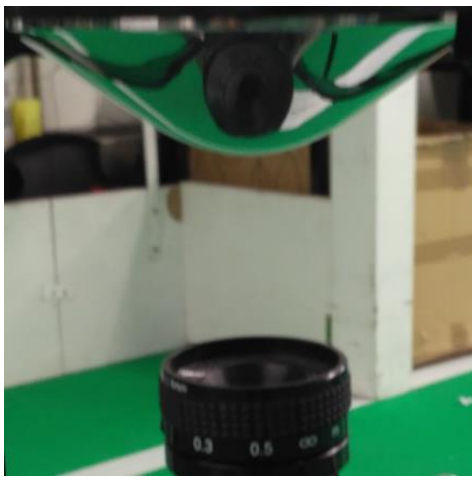


Figura 8 - Espelho catadióptrico

Como já fora dito a localização de um robô no espaço de jogo, é uma das tarefas mais importantes a ser realizadas. Contudo, este processo não é fácil de alcançar com sucesso, devido às inúmeras variáveis que o influenciam, sendo a principal a qualidade da imagem obtida.

Desta forma é necessário garantir que todas as etapas que dão origem ao resultado final, a imagem, contribuam da melhor forma possível. Posto isto, o trabalho começou por avaliar e rever o emparelhamento entre a câmara e o espelho pois, é o primeiro fator contribuinte para uma boa imagem.

3.1.1. Emparelhamento câmara – espelho

O trabalho iniciou-se pela aquisição de algumas imagens de forma a ser possível avaliar o emparelhamento. Durante este, não se obteve resultados satisfatórios pois, verificou-se que as imagens obtidas apresentavam deformações críticas que iriam influenciar o processo de localização no seu todo. Algumas das fotos podem ser observadas na figura que se segue, Figura 9.

Figura 9 - Deformações captadas pela câmara

Após realizados alguns testes, onde se fez variar a distância entre a câmara e o espelho, foi possível verificar que estes se encontravam demasiado distantes pois, como o espelho não é convexo no seu todo [13], a câmara encontrar-se-ia a captar essa transição o que estaria na origem das deformações acima observadas.

Posto isto, optou-se por desenvolver e construir um novo suporte para a câmara, uma vez que a estrutura anterior não oferecia a possibilidade de fazer variar a distância acima referida. Durante a realização dos testes verificou-se um outro problema, relacionado com o facto da estrutura utilizada ser estática, pelo que não seria possível reajustar o alinhamento entre o centro da câmara e o centro do espelho, podendo, então, causar problemas caso os suportes da estrutura e a própria estrutura em si não se encontrarem devidamente maquinados e precisos. Neste sentido, desenvolveu-se então um novo suporte para a câmara com o intuito de ultrapassar os problemas acima descritos.

Na Figura 10 pode observar-se o novo suporte da câmara desenvolvido, tanto em CAD (lado direito) como já após a sua construção (lado esquerdo).

Figura 10 - Suporte da câmara

Após a construção do novo suporte e respetiva montagem, foi então necessário ajustar a altura a que cada câmara se devia posicionar, de forma a deixar de existir deformações nas imagens obtidas, podendo os resultados obtidos ser visualizados na Figura 11.

Após o ajuste acima referido, é possível observar que as deformações já não estão presentes, passando, assim, o sistema a ser dotado de uma imagem com qualidade, retratando de forma correta o mundo que rodeia o robô.

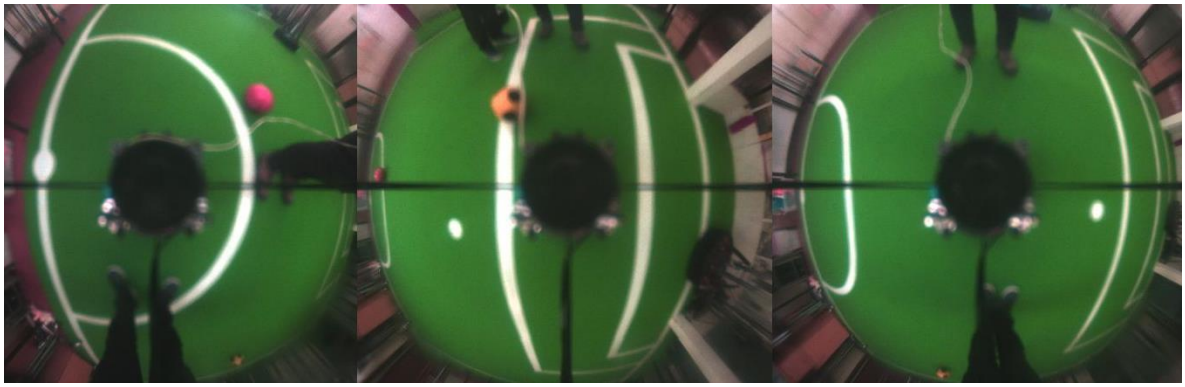


Figura 11 - Imagens obtidas após ajuste do novo suporte da câmara

3.1.2. Segmentação da imagem - *Color Look Up Table*

A etapa que se segue prende-se com o processamento de imagem, sendo o primeiro passo desta a segmentação da imagem obtida, que consiste na classificação dos pixéis da imagem de acordo com a cor que estes possuem.

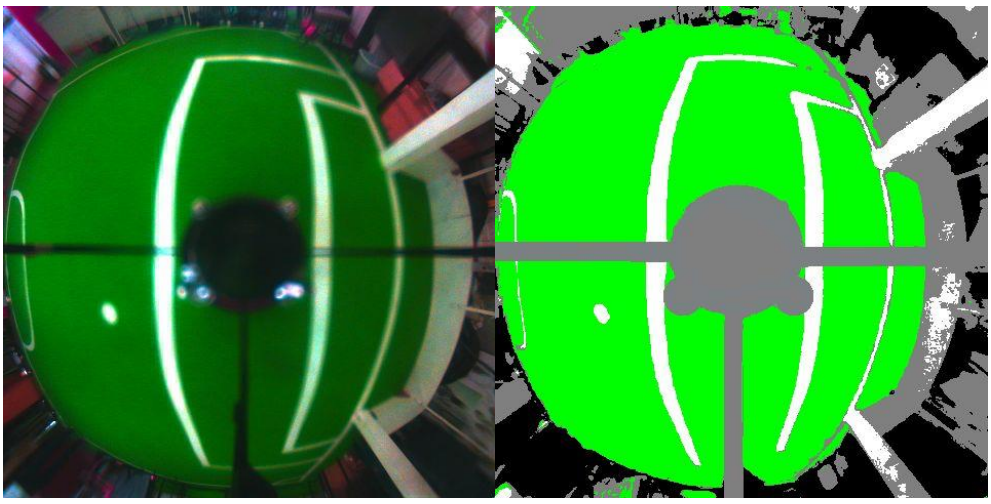
Na primeira fase, são lidos a partir de um ficheiro do tipo .txt os limites no espaço de cores HSV, para cada cor que possua significado para o processamento que será

posteriormente realizado. De seguida são percorridas todas as combinações possíveis de RGB, convertidas para HSV e classificadas em bola, robô, linha, campo ou sem cor que significa sem significado, de acordo com os limites lidos do ficheiro. Esta classificação é então guardada na LUT, na posição correspondente à combinação RGB.

É de salientar que quando referimos "cada cor que possua significado" quer-se com isto dizer cores que se traduzem em informação relevante como o preto dos outros robôs ou obstáculos, o laranja da bola, o verde do relvado e o branco das linhas, pois são estas as cores que predominam e interessam num espaço de jogo de futebol robótico. Nesta primeira fase todo o processo é realizado *offline*, ou seja, é realizado aquando da inicialização de todo o programa.

Na segunda fase, fase em tempo real, ou seja, no decorrer do programa, a segmentação é realizada com a informação já inicializada, onde, como já fora dito, a combinação RGB presente no pixel corresponde a uma posição na LUT.

A pré-criação da LUT deve-se ao facto de esta poupar imenso tempo de processamento pois, não é necessário converter e comparar cada pixel da imagem com os limites em HSV, de cada vez que se obtém uma nova imagem, usando a combinação RGB como um índice para a classificação correspondente na LUT. A conversão dos valores de RGB para HSV prende-se com o facto do espaço de cores HSV apresentar ser mais imune às variações de luminosidade que o espaço de cores RGB.



a) *b)*

Figura 12 – a) Imagem obtida; b) Imagem segmentada

Na Figura 12 a) é possível observar a imagem obtida por parte da câmara, e na Figura 12 b) a imagem após a realização da segmentação. É também possível observar que na imagem obtida, Figura 12 a), surgem a câmara e partes do corpo do robô e caso

essas partes da imagem fossem analisadas durante o processamento, poderiam dar origem a erros acerca da informação do mundo que rodeia o robô.

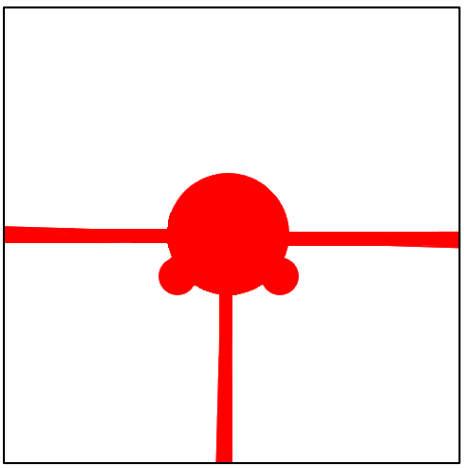


Figura 13 - Máscara para processamento de imagem

Assim, de forma a evitar o processamento sobre os pixéis que vêem partes do robô, é utilizada uma máscara que pode ser observada na Figura 13 a vermelho. Na Figura 11 é possível observar o resultado da aplicação da mesma na imagem segmentada, fazendo com que as partes referentes ao corpo do robô não estejam representadas e evitando, desta forma, o processamento das mesmas.

É de salientar que esta máscara é apenas um exemplo representativo, uma vez que os robôs em si possuem diferenças em termos de distância entre o espelho e a câmara, alterando o corpo do robô presente na imagem obtida, sendo necessário alterar então a Figura 13 em função de cada robô.

3.2. AUTO CALIBRAÇÃO DA CÂMARA

A câmara é o único sensor do robô que lhe permite adquirir informação acerca do ambiente que o rodeia, sendo desta forma a configuração dos parâmetros da câmara crucial para a deteção das linhas, dos obstáculos e da bola pois, caso a câmara não se encontre bem calibrada, pode ocorrer perda de informação ou obtenção de informação errada.

Portanto, é de enorme importância que a câmara do robô se encontre bem calibrada, sendo então necessário a auto calibração dos seus parâmetros. O desenvolvimento deste algoritmo para além de possuir um principal objetivo, obtenção de uma imagem com qualidade por todos os robôs da equipa, possui um outro, o de assegurar uma imagem

com qualidade mesmo durante o jogo, ou seja, pretende-se que o algoritmo seja executado durante o jogo, tempo real, de forma a avaliar a qualidade de imagens do robô e realizar alterações nos parâmetros da câmara se necessário pois, o ambiente em que se encontram durante um jogo é passível de sofrer alterações, como por exemplo alteração da luminosidade que se apresenta no exterior que influencia o interior.

O método desenvolvido pela equipa CAMBADA da Universidade de Aveiro será a base para o desenvolvimento do método de auto calibração para a MINHO TEAM.

Após a obtenção da primeira imagem capturada por parte da câmara inicia-se então a calibração dos parâmetros da câmara. O algoritmo começa por calcular o histograma da luminância e a sua média, foi definido um erro mínimo que a média pode estar desviada do valor de referência, 127, caso a média calculada não se encontre dentro dos limites definidos, ou seja, $média_{Lum} < Ref_{Lum} - Ref_{Lum} * 0.05$ ou $média_{Lum} > Ref_{Lum} + Ref_{Lum} * 0.05$, é feito então o ajuste dos parâmetros relacionados com a luminância. Optou-se por não utilizar o gama para calibração da câmara pois, era notório um esbater das cores da imagem e também porque após alguma pesquisa, entendeu-se que este deve ser mantido no seu valor mínimo. É também definido um valor máximo para o ganho pois, ao fim de alguns testes com os parâmetros da câmara foi notório que quando o ganho possui um valor demasiado elevado**,** as imagens obtidas começam a sofrer de alguma distorção, limitando-se o valor do ganho para $\frac{3}{4}$ do seu valor máximo.

O ajuste começa por verificar se o ganho já se encontra no seu limite ou não, caso este já não possa ser ajustado é então ajustado o valor da exposição de acordo com a diferença entre a média calculada a partir do histograma e o valor de referência utilizando um controlador PID, caso contrário será o ganho que será alterado.

De seguida a média da luminância é novamente comparada, se esta se encontrar entre 102 e 152, ou seja, $102 < média_{Lum} > 152$ procede-se ao ajuste do *white-balance*, da saturação e do brilho.

De forma a calibrar a saturação é calculado o histograma da saturação e a sua média, sendo que, para tal, foram definidos limites, tal como na luminância, $média_{Sat} < Ref_{Sat} - Ref_{Sat} * 0.05$ ou $média_{Sat} > Ref_{Sat} + Ref_{Sat} * 0.05$. Caso o valor da média não se encontre dentro dos limites estabelecidos, procede-se à calibração da saturação.

Para calibrar o *white-balance* é necessário possuir uma região branca, pelo que se optou por esta estar incluída no corpo do robô, sendo a sua posição em pixéis, na imagem, obtida aquando da inicialização do programa, no ficheiro que contém informações acerca

da calibração da câmara. Como já se possui as coordenadas bem como a área da região branca, segue-se o cálculo da média de U e V nessa região, tendo sido definidos limites de forma a validar se a média se encontra perto do valor referência, 127 para ambos. Caso o valor da média obtido não esteja dentro de um intervalo de $\pm 0{,}5\%$ do valor de referência, tanto o *white-balance blue* como o *white-balance red* serão calibrados de acordo com o erro respetivo.

Por fim, é calibrado o brilho. Tal como no *white-balance*, é necessário definir uma região, mas uma região preta e não branca, sendo que esta tal como a região anterior faz parte do corpo do robô e as suas coordenadas e área são lidas do ficheiro. Calcula-se então a média de R, G e B presentes nessa área. Caso alguma média seja maior que 6.35 acima do valor de referência, procede-se à calibração do brilho. O valor de referência para o brilho é 0.

Os valores de referência escolhidos têm por base o conhecimento adquirido acerca dos valores, que os parâmetros acima referidos devem possuir, de forma a obter-se uma imagem com qualidade. É de salientar que o algoritmo de calibração apenas deixa de ser executado quando nenhum parâmetro é alterado [7].

Na Figura 14 é apresentado um fluxograma de forma a facilitar o entendimento do funcionamento de todo o processo de calibração da câmara no seu geral, ao passo que nas Figuras 15 e 16 são apresentados mais pormenorizadamente os passos referentes aos blocos de ajuste do ganho e exposição e de ajuste da saturação, brilho e *white-balance*.

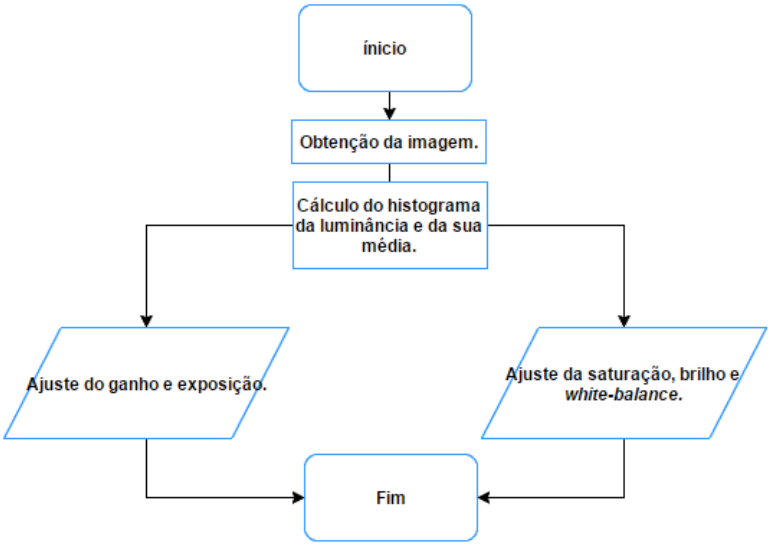


Figura 14 - Fluxograma do algoritmo de auto calibração

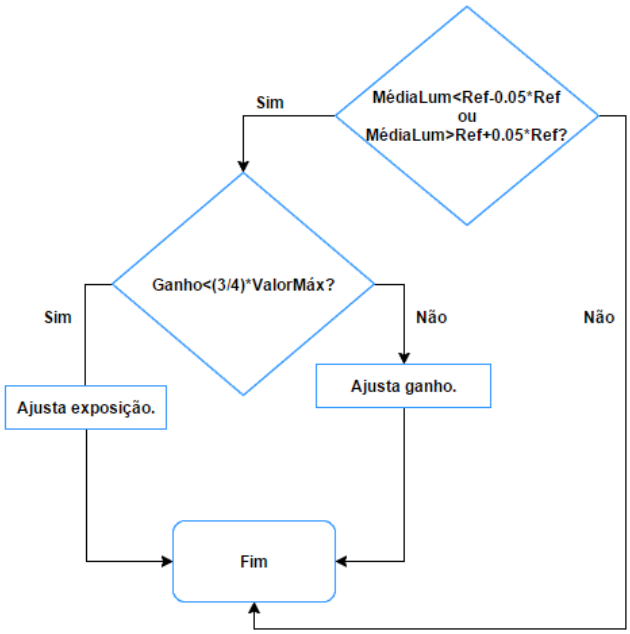


Figura 15 - Fluxograma de ajuste da exposição e brilho do algoritmo de auto calibração

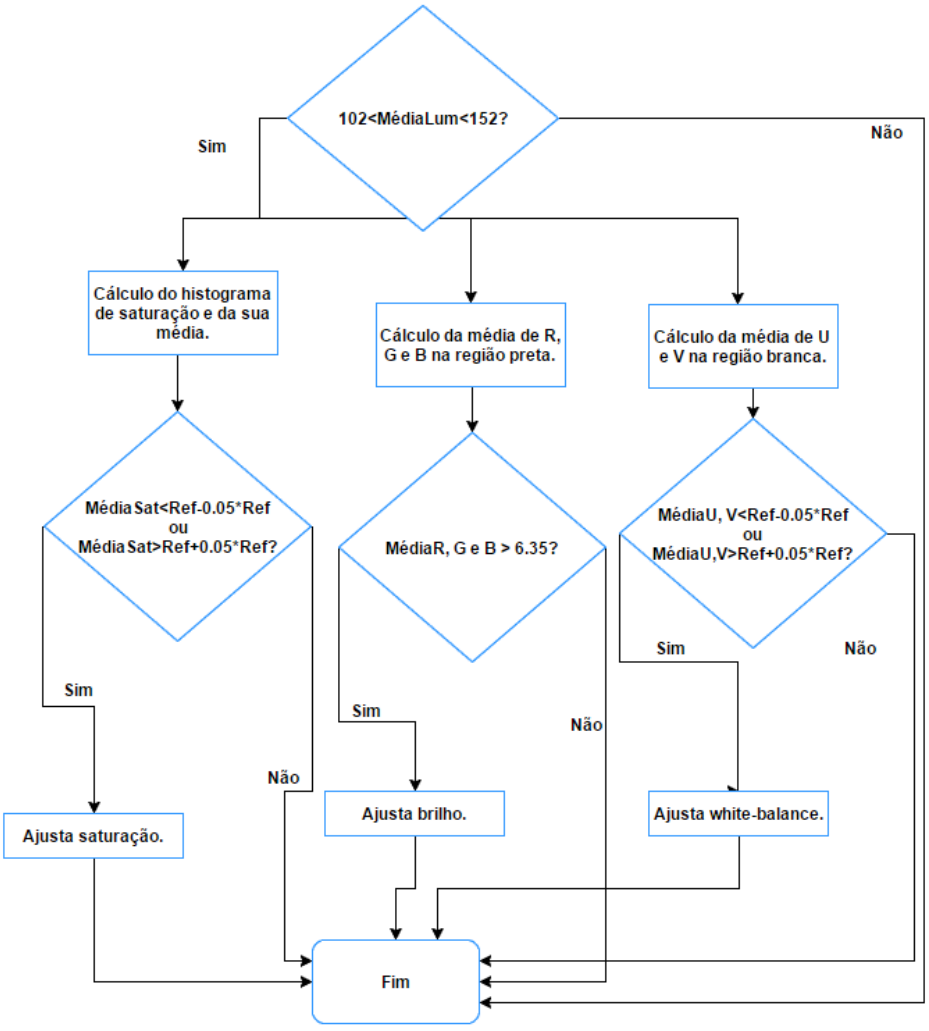


Figura 16 - Fluxograma de ajuste da saturação, brilho e white-balance do algoritmo de auto calibração

3.3. LOCALIZAÇÃO

Após o sistema de visão do robô se encontrar analisado e concluído, ou seja, o robô é dotado de uma imagem que retrata a realidade do meio onde se encontra e de uma imagem estável mesmo que existam variações de luminosidade no ambiente em que se encontra, podemos então avançar para a localização do robô em si.

Neste subcapítulo são descritos todos os passos necessários após a obtenção da imagem segmentada até à obtenção das coordenadas no campo, de forma a melhor se entender todo o processo desenvolvido para obtenção da posição do robô no campo. Este processo inicia-se por realizar um processamento de imagem, isto é, começa-se por analisar a imagem segmentada recebida, sendo esta análise explicada no subcapítulo que se segue, 3.3.1.

3.3.1. Linhas de pesquisa na imagem – *Scanlines*

Para a execução do algoritmo que possibilita a localização do robô é necessário que as linhas pertences ao campo sejam detetadas na imagem já segmentada. Porém, a análise total da imagem seria bastante morosa pois, seria necessário analisar todos os pixéis da imagem, e, uma vez que esta possui um tamanho de 480×480 pixéis, corresponderia a 230400 pixéis analisados. Para otimização do processo de deteção das linhas na imagem optou-se por usar linhas de procura radiais e axiais, *scanlines*.

São utilizadas 180 linhas de procura radiais e 30 circulares.

As linhas radiais estão espaçadas entre si de 2 graus, 4 pixéis, mas estas possuem dois tamanhos distintos, a cada par de linhas radiais, uma possui metade do tamanho da outra, onde a mais pequena apenas começa a meio da linha maior, como se pode observar na Figura 17.

Esta configuração, menor número de linhas de pesquisa perto do robô e maior número quanto mais longe do robô, deve-se ao facto das linhas do campo quanto mais perto do robô estiverem, mais fácil se torna a sua deteção, o que não acontece quanto mais afastadas estiverem.

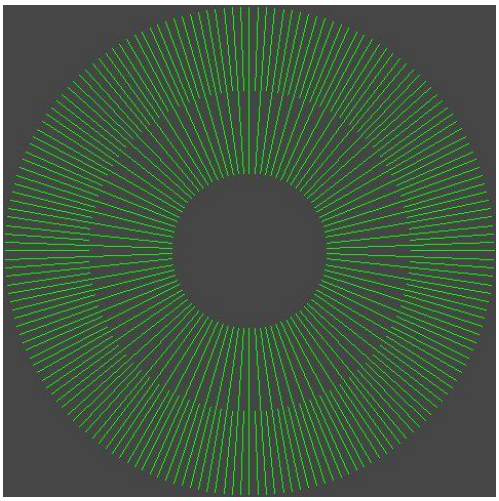


Figura 17 - Linhas de pesquisa radiais

As linhas de pesquisa circulares estão espaçadas entre si de 5 pixéis, como se pretende representar na Figura 18. Esta escolha deve-se ao facto de não se pretender um excessivo número de deteções demasiado perto umas das outras, o que poderia sobrecarregar o processamento.

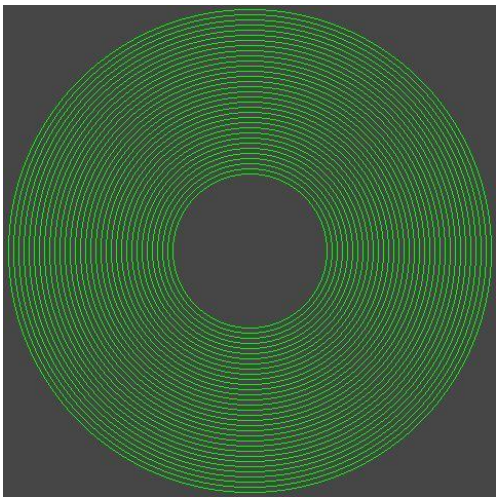


Figura 18 - Linhas de pesquisa circulares

Com este tipo de pesquisa, radial e circular, são analisados 51644 pixéis, que correspondem a 22,41% da imagem total.

Este processo, processo das linhas de pesquisa, possui duas fases, uma fase *offline*, onde é definido o número de linhas de pesquisa que se pretende e o seu tipo, e outra *real-time*. A fase *real-time* ocorre após a captura de uma imagem, onde as linhas de pesquisa são percorridas e preenchidas com a cor correspondente, ou seja, como cada ponto das linhas de pesquisa correspondem a um pixel da imagem, esse ponto fica com a mesma cor do pixel em questão, para posterior análise.

3.3.2. Deteção de linhas

Após as *scanlines* se encontrarem totalmente preenchidas é realizada, então, a procura de linhas de campo. Esta deteção é feita procurando transições de cor nas linhas de pesquisa, do tipo verde-branco ou branco-verde, sendo que esta procura por transição, pode ser alterada caso assim se pretenda.

O algoritmo de pesquisa é regrado por quatro parâmetros, sendo eles o *threshold* da cor anterior, da cor de interesse, da cor seguinte e o tamanho da janela de pesquisa. O *threshold* das cores define o número de pixéis necessários para validar que foi encontrado um segmento dessa cor e o tamanho da janela define, como o nome indica, o número de pixéis que serão avaliados no caso da cor anterior e da cor seguinte [14]. De forma a melhor se entender tudo isto, explicar-se-á de seguida o método de forma mais pormenorizada.

Num campo de jogo, a cor anterior a uma linha será sempre verde e a cor que se segue também, pelo que se procura então por uma transição verde-branco-verde. Na figura que se segue podemos observar um exemplo de *scanline*. Onde a cor anterior e a cor seguinte são representadas pelo V e a cor de interesse pelo B.

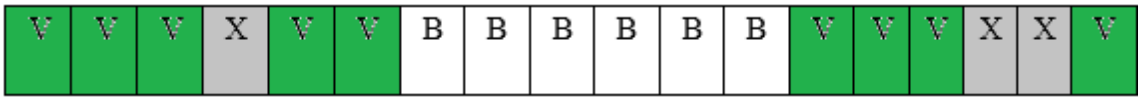


Figura 19 - Exemplo de scanline

O algoritmo começa por procurar o primeiro ponto da *scanlines* que seja da cor de interesse, neste caso branco, simbolizado com um B na Figura 19, após o encontrar utiliza o tamanho da janela de pesquisa, contando quantos pontos verdes, simbolizados com um V, se encontram antes. De seguida procura pelo último ponto da cor branca e conta-os, novamente utilizando o tamanho da janela de pesquisa. Na Figura 20 é possível visualizar o que fora explicado acima.

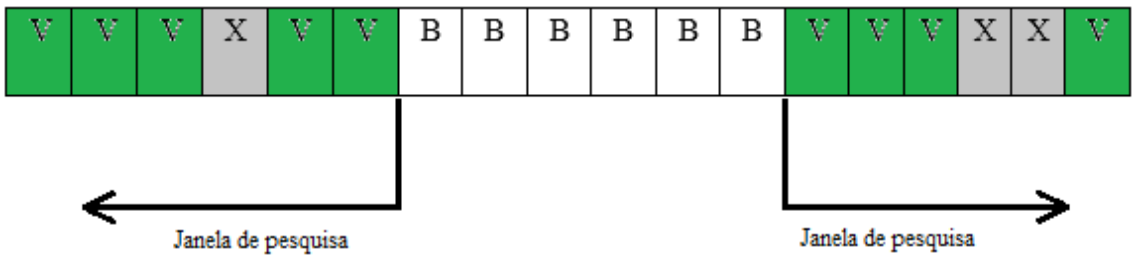
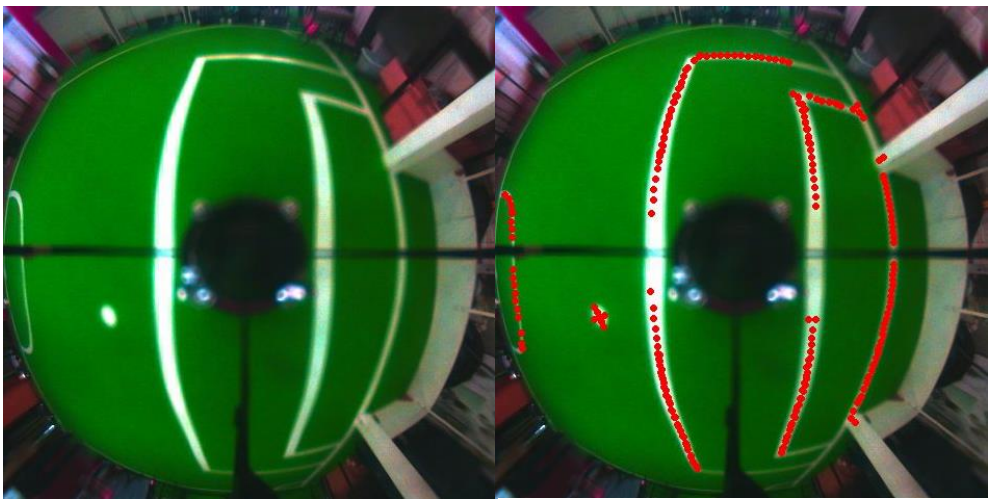


Figura 20 - Exemplo do algoritmo de pesquisa

No fim de contar quantos pontos da cor anterior, da cor de interesse e da cor seguinte, para realizar validação de que se trata de uma linha, é necessário que as contagens sejam superiores ou iguais ao respetivo *threshold* de cada uma. É de salientar, que o X na figura acima, tem o propósito de simular que a imagem pode conter pixéis que possuam informação não relevante, introduzida por situações adversas ao robô, mas que o algoritmo de pesquisa de pontos de linha está deveras preparado para esse tipo de acontecimentos.



a) *b)*

Figura 21 – a) Imagem obtida; b) Imagem com pontos de linha detetados

Na Figura 21 a) pode observar-se a imagem obtida e na Figura 21 b) as transições de pontos de linha que foram detetadas, sendo que estas são armazenados num vetor, guardando a posição do pixel (X, Y) em que se encontram na imagem, e ainda o número de pontos da cor de interesse que as constituem, neste caso número de pontos de cor branca.

A Figura 22 apresenta a reconstrução da imagem capturada por parte da câmara utilizando as linhas de pesquisa e as transições de linha detetadas nas mesmas, por parte do algoritmo.

Como se pode observar na Figura 22 as transições dos pontos de linha, não são pontos isolados, mas sim um conjunto deles aproximadamente com a mesma espessura das linhas onde foram detetados. Desta forma, é necessário traduzir este conjunto em um único ponto, pelo que foi decidido utilizar o ponto central de cada transição detetada e utilizar as coordenadas (X, Y) deste ponto como sendo a coordenada das mesmas.

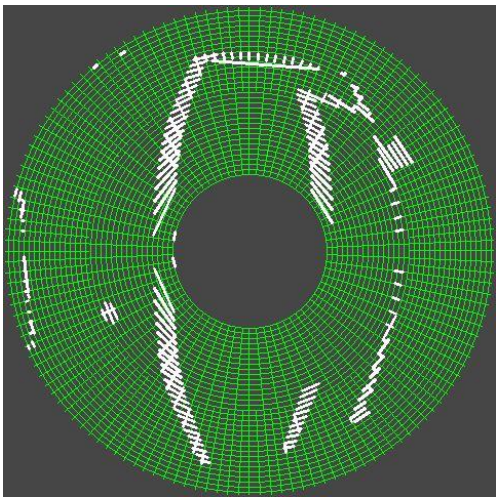
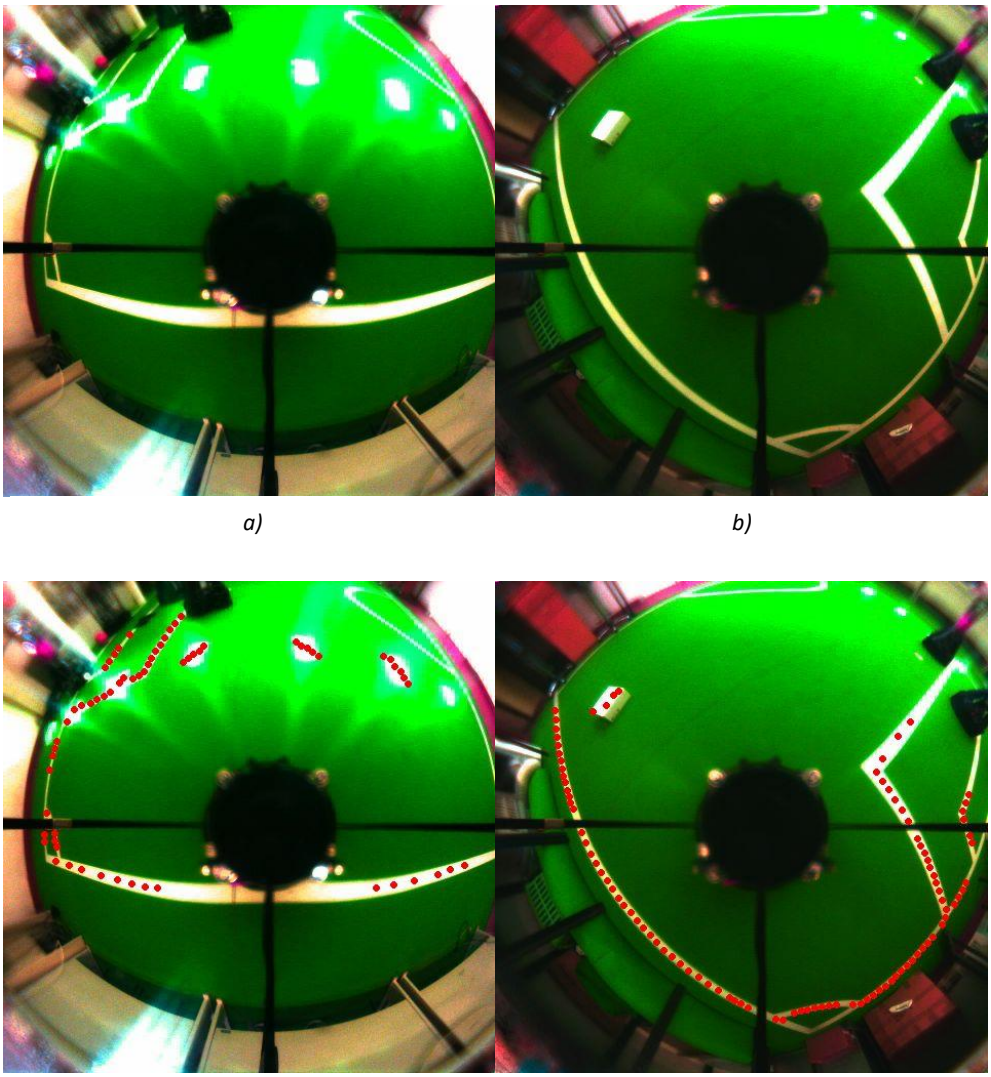


Figura 22 –Transições de linha detetadas

Portanto sempre que se fala em coordenada (X, Y) das transições referimo-nos ao ponto central das mesmas, como se pode observar na Figura 21 onde as transições estão representadas a vermelho utilizando o seu ponto central.

3.3.2.1. Filtragem

Após a pesquisa de pontos de linha, segue-se a validação de que os pontos encontrados pertencem realmente a uma linha e não a um foco de luz que possa aparecer no campo ou qualquer objeto de cor branca. Na Figura 23 pode observar-se dois casos distintos, o caso em que aparecem raios de luz a incidir no campo de jogo e, um outro caso que se refere à situação de um objeto, não pertencente a um jogo de futebol, aparecer inserido no mesmo e o resultado da deteção de linhas.



a) *b)*

c) *d)*

Figura 23 – a) Imagem com raio de luz; b) Imagem com caixa branca; c) Imagem com raios de luz e deteção desses raios como pontos de linha; d) Imagem com caixa branca e deteção da caixa como pontos de linha

De forma a verificar então a veracidade dos pontos encontrados, é feita uma filtragem tendo por base o número de pontos da cor de interesse, neste caso cor branca, cor das linhas, e pela distância, em pixéis, a que este se encontra do centro da imagem.

Para tal ser possível, é necessário executar dois passos, traduzir a posição (X, Y) da transição detetada para uma distância, em pixéis, e saber que índice de um vetor, número de pontos aproximado que uma linha deve possuir baseado na distância a que se encontra, deve ser utilizado.

A tradução de posição (X, Y) para distância em pixéis e ângulo é feita tendo em conta o sistema de eixos coordenados definido para a imagem bem como o centro da imagem. Estes estão definidos na Figura 24.

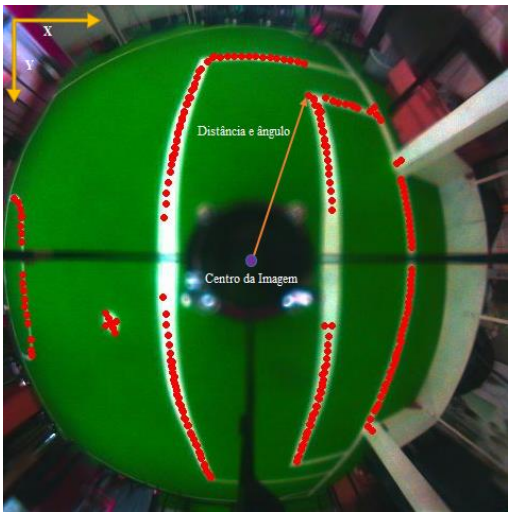


Figura 24 - Plano e eixos coordenados de uma imagem

Na Figura 24 pode observar-se a amarelo o sistema de eixos coordenados, a roxo o centro da imagem e a laranja um exemplo de distância e ângulo obtido para um ponto de linhas encontrado. É de notar que o centro da imagem e o sistema de eixos coordenados são previamente definidos na calibração de cada robô, sendo esta configuração gravada em ficheiro e carregada para o programa aquando da inicialização do mesmo.

Depois de calculada a distância a que a transição se encontra do centro da imagem, distância entre dois pontos, é necessário saber o número de pontos que a transição de linha detetada deveria possuir consoante a distância a que se encontra. Para tal, o programa possui dois vetores, o vetor que contém o número de pontos que uma linha possui de acordo com a distância a que se encontra do centro da imagem - vetor de filtragem -, e um vetor que possui as distâncias em pixéis. Ambos os vetores são previamente calibrados e guardados num ficheiro para, aquando da inicialização do programa estes serem carregados para o mesmo.

Na Tabela 3 está representado um exemplo do vetor de filtragem e na Tabela 4 o vetor de distância em pixéis.

Tabela 3 - Exemplo de vetor de filtragem

26	24	20	15	10	8	5	3	2	2

Tabela 4 - Exemplo de vetor de distância em pixéis

55	88	112	136	155	170	183	190	196	200

De forma a saber qual valor utilizar da Tabela 3, recorremos à Tabela 4 e extraímos, consoante a distância da transição ao centro da imagem, o índice que melhor se ajusta, ou seja, caso a transição se encontre a 65 pixéis de distância do centro da imagem, o valor a ser utilizado do vetor de filtragem para a verificação será o primeiro, 26, pois, 65 aproxima-se mais de 55 do que de 88 traduzindo-se isto na utilização do primeiro índice da Tabela 3. Por outras palavras, a distância a que a transição se encontra, indica que índice do vetor de filtragem deve ser utilizado. O vetor de distância em pixéis, Tabela 4, é explicado e introduzido no subcapítulo Mapeamento, 3.3.4.

Após obter o valor a utilizar da Tabela 3, este é comparado com o número de pontos que a transição a ser analisada possui, e caso os valores não sejam demasiado díspares então esta é considerada uma transição de linha válida, por outro lado, caso o sejam, esta é apagada do vetor. Este algoritmo de filtragem é aplicado a todas as transições ou pontos de linha detetados.

É de notar que o vetor das transições de linha detetadas continua a ser constituído pela posição (X, Y) em pixéis da imagem.

Após se ter realizado a validação de que realmente se trata de um ponto de linha, estão reunidas todas as condições necessárias para se realizar o cálculo da orientação do robô no campo.

3.3.3. Orientação

O cálculo da orientação do robô é um aspeto de bastante relevo, pois além da localização do robô ser (X, Y, θ), onde θ é a orientação do mesmo no campo, permite também que a obtenção da posição do robô no campo seja executada com maior rapidez, uma vez que já é conhecida a orientação deste, não sendo necessário desta forma a análise da rotação do robô em cada posição no campo.

Para o cálculo da orientação é necessário definir um sistema de eixos coordenados no campo. Na Figura 25, pode observar-se o sistema definido, sendo que quando a frente do robô se encontra alinhado com o eixo do Y o ângulo é 0º.

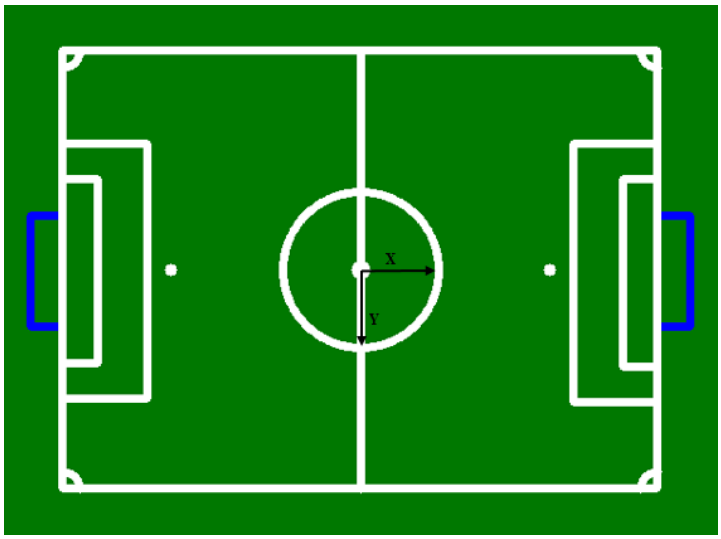


Figura 25 - Campo com sistema de eixos coordenados

A orientação do robô é obtida de três formas distintas: é usado um IMU, *Inertial Measurement Unit*, é usado um algoritmo que através das transições detetadas calcula a orientação do robô e é usado, ainda, odometria com recurso a *encoders* nas rodas omnidirecionais do robô. Fundindo a informação obtida de cada um é possível alcançar resultado mais fiável e preciso.

O algoritmo de cálculo da orientação do robô que utiliza as transições detetadas tem por base a utilização de histogramas [15].

De forma a iniciar o algoritmo é definido o intervalo de rotação em que se vai realizar a pesquisa, sendo o intervalo escolhido de 40º, ou seja, as transições detetadas vão ser rodadas de $\theta - 20º$ até $\theta + 20º$, onde θ é a última orientação conhecida.

O algoritmo inicia-se criando dois histogramas, um horizontal e outro vertical, com as transições detetadas e com a última orientação, e varre o intervalo definido, rodando os pontos de transição. A cada iteração, incrementação do ângulo de rotação, o valor máximo, soma do valor máximo vertical e horizontal, é calculado, sendo que o valor adquirido nesta iteração é comparado com o valor máximo, obtido e guardado até então, e caso este seja superior guarda-se então o novo máximo e o ângulo correspondente.

O ângulo em que a soma dos valores dos histogramas, vertical e horizontal, for máxima corresponde à rotação que o robô sofreu entre imagens. É utilizado a soma do valor máximo de cada histograma e não o valor máximo individual de cada um, pois verificou-se que em alguns casos os histogramas podem possuir mais que um máximo e desta forma mais que uma solução.

É de salientar que na primeira vez que o algoritmo acima descrito é executado não existe um valor para a última orientação conhecida, θ, sendo usado neste caso o valor obtido por parte do IMU.

Como referido, utiliza-se também um IMU para a obtenção do valor da orientação. A razão da sua utilização prende-se com o facto de o cálculo da orientação com recurso ao algoritmo supracitado depender da quantidade de transições detetadas, o que por vezes pode ser um problema. Desta forma, utilizando o IMU é possível tornar a obtenção da orientação do robô um processo mais robusto, menos oscilante e mais próximo do real. O IMU utilizado é o IMU adafruit 10 DOF, que se encontra representado na Figura 26 [16].

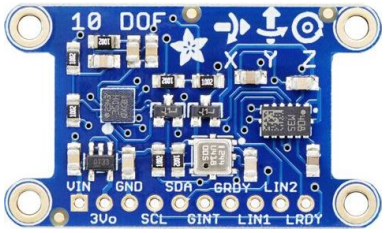


Figura 26 - IMU adafrauit 10 DOF

O IMU utiliza uma bússola eletrónica e um giroscópio de forma a dar informação acerca do ângulo em que se encontra orientado, fornecendo um valor bastante próximo da realidade. Após alguns testes verificou-se que este possui uma resolução de aproximadamente 0,5º. O IMU além de ser utilizado para obter a orientação do robô, é também utilizado para verificar se esta se encontra correta, isto é, caso o valor fornecido pelo IMU e o valor obtido pelos histogramas sejam suficientemente díspares, é então necessário realizar a relocalização do robô no campo, sendo que esta diferença entre valores foi previamente estipulada após vários testes.

De forma a obter um valor da orientação do robô o mais preciso possível é realizada a fusão do valor obtido pelo algoritmo com recurso a histogramas com o valor fornecido pelo IMU com utilização de uma equação, em que são dados pesos distintos a cada método, onde P_{IMU} é o peso dado ao valor obtido por parte do IMU e P_{Hist} o peso dado ao valor obtido por parte dos histogramas, como pode ser observado na Equação 4.

$$Orientação = P_{IMU} * Valor\ IMU + P_{Hist} * Valor\ Hist \quad (4)$$

Após a obtenção do valor da orientação por fusão da visão e do IMU é feita a fusão desse valor com o valor obtido pela odometria [17].

A odometria é um método muitas vezes utilizado em robótica móvel, permitindo estimar a posição de um robô através do uso de *encoders* nas rodas do mesmo, o que permite saber o quanto este se deslocou. Este método é dotado de uma boa precisão a curto prazo, porém é necessário saber a orientação anterior em que o robô se encontrava sempre que se pretende utilizar os valores obtidos pelos *encoders*. Desta forma, é possível obter uma estimativa precisa, uma vez que a localização por a odometria não é um método de localização absoluta [18].

De cada vez que se recebe um novo valor de *encoder* é calculado a variação do valor do mesmo. Uma vez que os valores do *encoder* variam entre -32768 e 32768 é necessário detetar quando o valor do *encoder* passa de -32768 para 32768 por forma a obter uma leitura correta da variação do *encoder*. Assim, aplica-se o seguinte algoritmo:

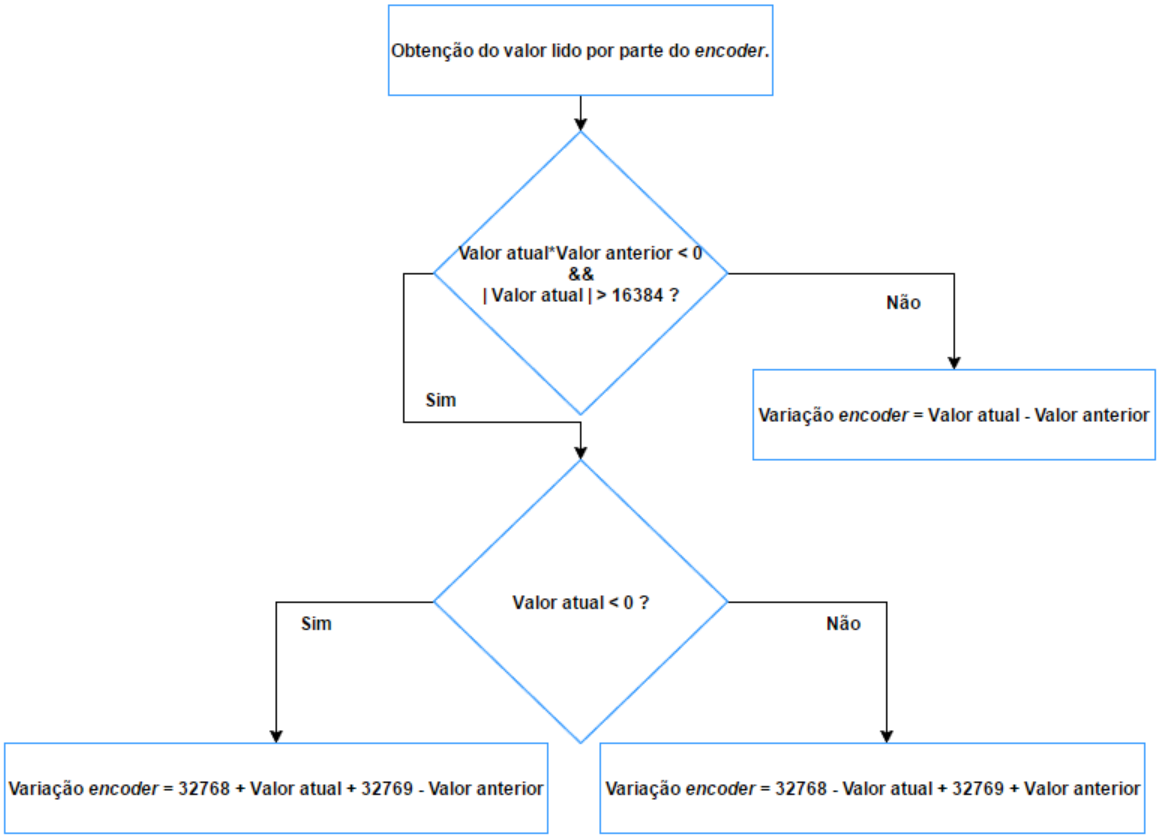


Figura 27 - Fluxograma de obtenção da variação do encoder

Como podemos ver, na Figura 27, no caso em que o valor atual do *encoder* e o valor anterior tem o mesmo sinal, a variação do *encoder* é simplesmente a sua diferença, da mesma forma, se os valores trocarem de sinal, mas o fizerem pelo 0 (daí a comparação do valor atual > 16384), continua a ser apenas a sua diferença. No caso da "volta" ser dada pelos -32768, o valor da variação é calculado tendo em conta a situação.

Após obtido o valor da variação do *encoder*, Δe, é calculada a velocidade do mesmo, sendo esta dada pela seguinte equação:

$$Velocidade = \frac{2 * \pi * r}{\Delta t * CPR} * \Delta e \quad (5)$$

Na equação acima, Δt é o intervalo de tempo entre envio dos dados por parte do *encoder*, e o CPR é o número de *ticks* do *encoder* necessários para a roda efetuar uma volta completa.

Após converter as medidas recebidas pelos *encoders* (rotações por segundo), é realizada a conversão para m/s, com a utilização da equação acima referida. De seguida realiza-se a conversão para deslocamento em X, Y, θ, possibilitando desta forma a utilização da odometria [19][20].

Porém, existem duas etapas que diferem entre si, a localização global e a localização local.

No cálculo da orientação na localização global o valor da odometria não é utilizado, uma vez que esta é realizada com o robô em repouso, mas é utilizado na localização local isto, pois, de cada vez que o robô se movimenta, são lidos valores pelos *encoders*. Quando o valor de rotação obtido pelos *encoders* é utilizado, θ, é realizada uma soma deste valor, θ, com o último valor da orientação do robô conhecido fornecendo desta forma uma estimativa da orientação atual. Esta estimativa é então depois fundida, utilizando um filtro de Kalman, com o valor obtido através da fusão anteriormente referida, fusão do ângulo da visão com o ângulo obtido pelo IMU, obtendo-se assim um valor mais preciso acerca da orientação do robô.

As equações da fusão dos valores de orientação obtidos podem ser visualizadas na Tabela 5.

É de salientar que após a utilização dos valores da odometria para cálculo da orientação do robô, estes são reiniciados a zero pois, como já fora dito, a odometria não é um método de localização absoluto, o que faz com a estimativa com recurso à odometria necessite sempre de saber a última posição do robô desde o início em que se começa a acumular valores de movimento por parte dos *encoders*.

Tabela 5 - Equações do filtro de Kalman para fusão dos valores de orientação

Previsão
Estimativa do estado
$\widehat{\boldsymbol{\theta}}^{-}_{\boldsymbol{Orientação\ Atual}} = \boldsymbol{\theta}_{\boldsymbol{Ultima\ Orientação}} + \boldsymbol{\theta}_{\boldsymbol{odometria}}$
Estimativa da covariância
$\boldsymbol{P}^{-}_{\boldsymbol{Orientação\ Atual}} = \boldsymbol{P}_{\boldsymbol{Ultima\ Orientação}} + \boldsymbol{Q}_{\boldsymbol{Orientação}}$
Correção
Cálculo do ganho de Kalman
$\boldsymbol{K}_{\boldsymbol{Orientação\ Atual}} = \boldsymbol{P}^{-}_{\boldsymbol{Orientação\ Atual}}\left(\boldsymbol{P}^{-}_{\boldsymbol{Orientação\ Atual}} + \boldsymbol{R}_{\boldsymbol{Orientação}}\right)^{-1}$
Correção do estado
$\widehat{\boldsymbol{\theta}}_{\boldsymbol{Orientação\ Atual}} = \widehat{\boldsymbol{\theta}}^{-}_{\boldsymbol{Orientação\ Atual}} + \boldsymbol{K}_{\boldsymbol{Orientação\ Atual}}\left(\boldsymbol{\theta}_{\boldsymbol{Visão\ atual}} - \widehat{\boldsymbol{\theta}}^{-}_{\boldsymbol{Orientação\ Atual}}\right)$
Correção da covariância
$\boldsymbol{P}_{\boldsymbol{Orientação\ Atual}} = \left(\mathbf{1} - \boldsymbol{K}_{\boldsymbol{Orientação\ Atual}}\right) \boldsymbol{P}^{-}_{\boldsymbol{Orientação\ Atual}}$

3.3.4. Mapeamento

Antes de se proceder ao cálculo da posição do robô no campo é necessário mapear as transições de linha detetadas em metros relativamente ao centro do robô.

Neste subcapítulo é explicado como se traduz a posição (X, Y) em pixéis das transições detetadas, para posições no campo relativamente à posição do robô no campo, isto pois, para ser possível executar este mapeamento assume-se que o robô se encontra na posição $(0,0)$ do campo, sendo esta no meio campo onde se encontra o sistema de eixos coordenados, representado na Figura 25.

A escolha deste sistema de eixos coordenados prende-se pelo facto de com este posicionamento do mesmo ser possível dividir o campo em 4 partes distintas, como esquerda e direita, e parte superior ou inferior, o que traz alguma facilidade na realização de alguns cálculos.

Para realizar o mapeamento das transições detetadas para posições no campo é necessário realizar a conversão da posição (X, Y) em pixéis na imagem para uma posição (X, Y) em metros no campo. Estas conversões assentam em dois vetores fundamentais para o bom funcionamento de todo o processo de localização em si. Estes vetores são previamente carregados para o programa aquando da inicialização, encontrando-se guardados num ficheiro, pois estes são calibrados para cada robô.

Um dos vetores é o vetor das distâncias reais, que é criado a partir de dois valores lidos, *step* e distância máxima, em metros. A distância máxima traduz-se no raio de visão do robô, ao passo que o *step* indica o intervalo de medidas entre o centro do robô até ao valor do raio de visão estipulado.

Tabela 6 - Exemplo do vetor de distância real

0,25	0,5	0,75	1	1,25	1,5	1,75	2	2,25	2,5

Pode ser observado na Tabela 6 um exemplo do vetor de distância real, os valores diferenciam entre si de $0{,}25\ m$, e a distância máxima, ou seja, o raio de visão do robô é de $2{,}5\ m$. Como o *step* é de $0{,}25\ m$ obtém-se dez medidas, desde o centro do robô até ao raio de visão do mesmo.

O outro vetor, é o vetor das distâncias em pixéis, referido no subcapítulo Filtragem, 3.3.2.1, este é em tudo semelhante com o vetor das distâncias reais, porém possui a distância correspondente em pixéis para cada valor que o outro, vetor de distância real, possua. Para um objeto que se encontre a $1\ m$ de distância, sabemos que em termos de imagem este possui 136 pixéis de distância desde o centro da imagem.

Como fora dito, aquando da inicialização do programa ambos os vetores são lidos de um ficheiro, após a sua leitura estar concluída é então construída uma LUT de distância, isto é, é criado um vetor bidimensional com o mesmo tamanho da imagem, onde cada posição deste vetor possui uma distância em metros e um ângulo, sendo que o vetor de distância em pixéis e o vetor das distâncias reais são a base para a criação desta que passo a explicar.

Começa-se por percorrer toda a LUT, cada posição desta é encarada como um pixel, onde é calculada a distância entre esse pixel e o centro da imagem, diferença entre dois pontos, para cada posição, obtendo desta forma uma distância em pixéis.

De seguida, de acordo com a distância em pixéis obtida, obtemos a distância em metros correspondente, com recurso ao vetor da distância real, é também calculado o ângulo tendo em conta a orientação da frente do robô e a sua orientação no campo. Após o cálculo estes valores são guardados na posição correspondente na LUT.

Desta forma é possível obter, a distância em metros e ângulo a que as transições de linha detetadas e validadas se encontram do centro do robô, de uma forma muito mais "leve" a nível computacional, ao invés de ter de executar o mesmo processo para cada

transição pois, a posição (X, Y) da transição de linha detetada na imagem é utilizada como índice na LUT de distâncias.

É de salientar que caso os vetores de distância em pixéis e distância real não possuam o valor que se pretende, é feita uma linearização de forma a obter valores mais próximos do real.

Após o mapeamento de todas as transições de linha detetadas e calculada a orientação do robô, estão reunidas todas as condições necessárias para se proceder ao cálculo da localização do robô.

3.3.5. Posição do robô no campo

Como o próprio nome indica, localização é o lugar ou posição em que algo se encontra, está localizado, sendo que este é dependente do espaço onde se encontra, bem como do seu sistema de eixos de coordenados. A posição do robô é um ponto (X, Y, θ) no campo, utilizando o sistema de eixos coordenados referido na Figura 25.

Porém, para ser possível o cálculo da posição do robô no campo, (X, Y), é necessário que o programa possua em memória o campo a ser utilizado. Para tal a MINHOTEAM optou por utilizar um ficheiro que contém as medidas do campo, em termos de comprimento e largura, cada posição do campo (X, Y) e a distância à linha mais próxima desde esse ponto.

Aquando da inicialização do programa lê-se o ficheiro e é criado um vetor bidimensional com o tamanho do campo. O campo de jogo possui $18\ m$ de comprimento e $12\ m$ de largura, no entanto, foi definido que o campo possui $20\ m$ de comprimento e $14\ m$ de largura, de forma a considerar campo após as linhas que limitam o mesmo, e foi escolhido um *step* de $0.1\ m^2$ entre cada posição do campo. O vetor possui então um tamanho de $200x140$, ou seja, 28000 posições, onde cada posição do vetor possui a posição (X, Y) em metros do campo e a distância à linha mais próxima.

Na Figura 28 pode observar-se um exemplo de campo carregado para o programa, ou seja, pode observar-se o vetor bidimensional criado, onde, apontado pelas setas de cor preta, pode observar-se o que cada posição do vetor contém, posição (X, Y) do campo real, bem como a distância à linha mais próxima.

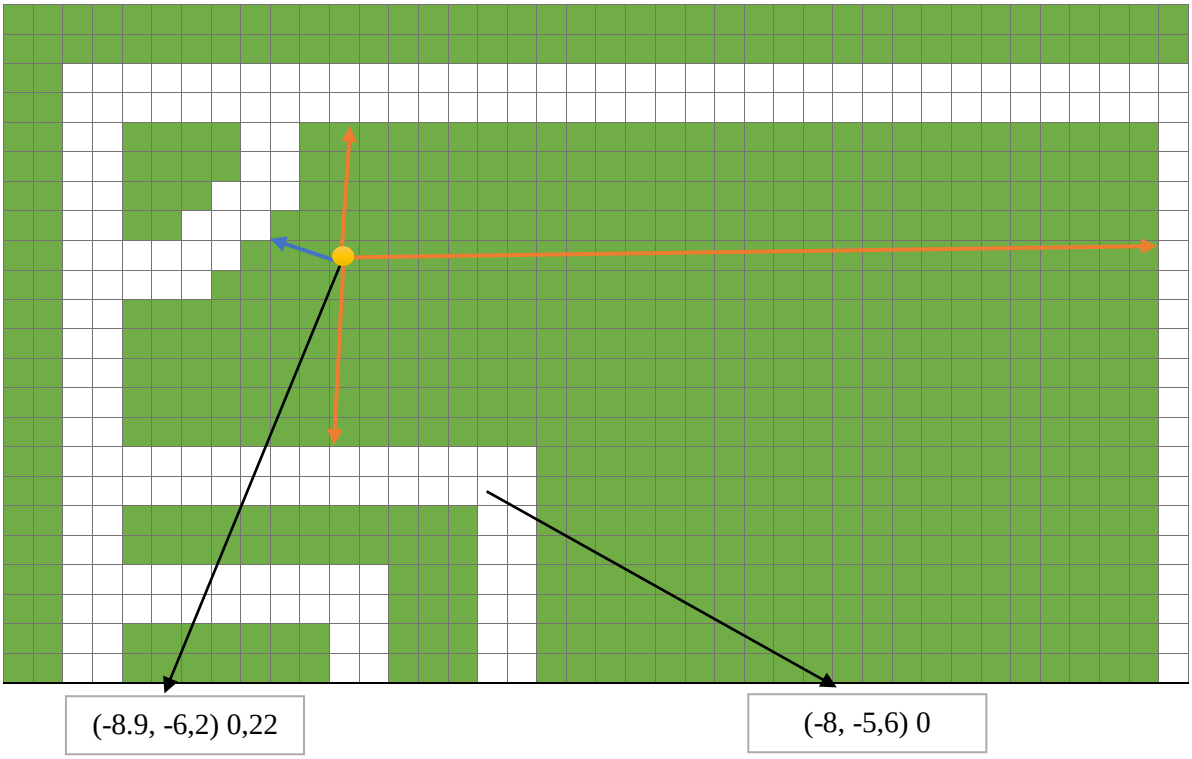


Figura 28 - Exemplo do campo carregado para o programa

Pode observar-se também que estão representadas outras quatro setas, três de cor laranja e uma de cor azul, sendo que as de cor laranja representam as possíveis distâncias às linhas desse ponto, onde poderiam ainda existir mais que as apresentadas, mas como se pode observar a seta de cor azul é a distância que se pretende pois, representa a menor distância de uma linha desde essa posição de campo, $(-8.9, -6{,}2)$ neste caso. Quando uma posição do campo lido é parte integrante de uma linha, é de notar que a sua distância é então 0, como se pode visualizar.

Desta forma é possível reduzir o custo computacional quando se pretende calcular a posição do robô no campo, uma vez que se possui o campo guardado num vetor ao invés de ler o ficheiro de cada vez que se necessite de informação que este contém.

A localização do robô no campo possui duas fases distintas, a fase global e fase local, contudo bastante semelhantes pois, ambas têm como base as transições de linha detetadas e utilizam o algoritmo *Perfect Match* [5] de forma a obter as coordenadas da posição do robô no campo.

Após terem sido inicializadas todas as componentes do programa, o processo de localização do robô será executado pela primeira vez. Dado que se trata da primeira vez este não possui qualquer informação acerca da posição anterior ou do local que se

encontra posicionado no campo, sendo este fator que diferencia se é uma localização global ou uma localização local. Portanto irá ser executado uma localização global.

3.3.5.1. Global

De forma a realizar o cálculo da localização global do robô, é apenas percorrida uma das partes do campo isto, pois, antes do jogo ter início sabe-se previamente qual será o lado do campo que a nossa equipa começará, tal como num jogo de futebol com humanos.

Portanto, em vez de percorrer todo o campo, é fornecida informação ao programa de que lado do campo se inicia o jogo. Começa-se então por definir em que lado do campo o robô se encontra, e o vetor bidimensional que contém o campo é percorrido consoante essa informação, a cada interação, cada posição (X, Y) do campo, todas as transições de linha detetadas são mapeadas, ou seja, como estas já se encontram mapeadas para a posição $(0,0)$, como fora dito no subcapítulo Mapeamento, 3.3.4, apenas se soma o (X, Y) da posição de campo que está a ser analisada à posição da transição. Após se executar esta soma, é verificado se a transição se encontra dentro dos limites do campo, pois pode acontecer que ao somar as coordenadas da posição que está a ser analisada à posição da transição em relação ao robô, esta fique fora dos limites definidos do campo.

A cada transição de linha que seja validada, com as suas coordenadas (X, Y) obtém-se a distância à linha mais próxima, que se encontra armazenada no vetor do campo, sendo esta utilizada pela Equação 6 de forma a modelar o erro consoante o valor obtido.

$$e = 1 - \frac{c^2}{c^2 + e^2} \tag{6}$$

A Equação 6 tem como função a penalização de desvios das transições de linha detetadas, pois o mapeamento das transições pode não as fazer coincidir perfeitamente com uma linha no modelo de campo ou devido ao ruído da imagem e ao pré-processamento da mesma podem surgir transições de linha detetadas erradas, sendo desta forma necessário a minimização do erro [5].

Na equação, e é o valor da distância à linha mais próxima e c é um parâmetro a partir do qual o erro é atenuado, sendo este parâmetro definido com um valor de 0.5. Este valor foi definido tendo em conta a gama de valores possíveis do erro.

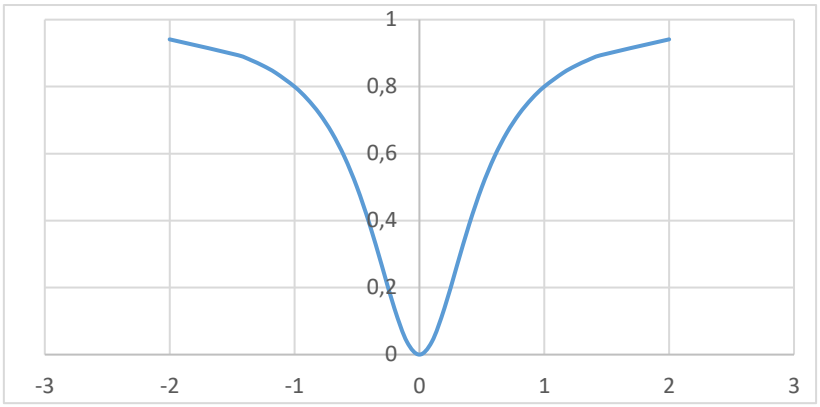


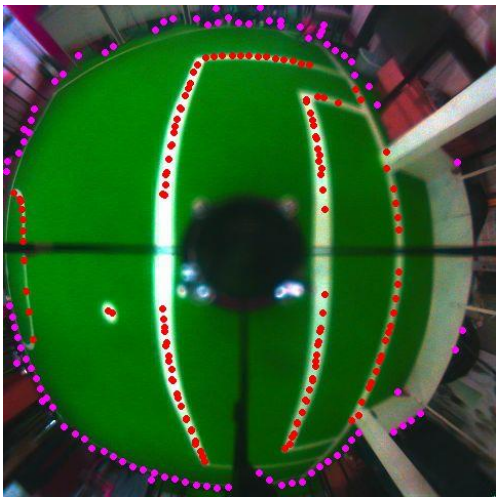
Figura 29 - Gráfico da equação do erro

Na Figura 29 está representado o gráfico da Equação 6, no qual se pode observar que quanto maior a distância à linha mais próxima, representado pelo eixo horizontal, maior será a contribuição para a incrementação do erro.

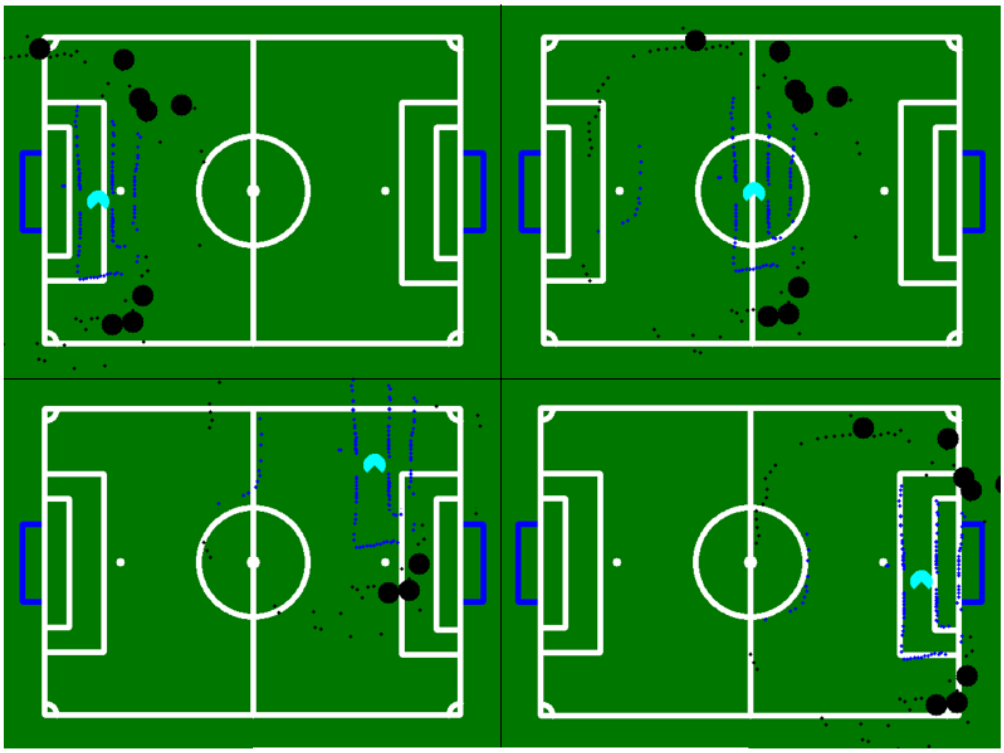
Por fim, quando todas as transições tiverem sido analisadas, obtém-se duas variáveis, o erro final, que é a soma do erro de todas as transições validadas, e a percentagem de transições que foram validadas. A posição que apresentar um mínimo de 90% de transições validadas, ou seja, 90% das transições dentro de campo e o menor erro será então a posição em que o robô se encontra.

Na Figura 30 a) pode observar-se a imagem obtida com as deteções, tanto de pontos de linha como de obstáculos, e na Figura 30 b) observa-se o funcionamento do algoritmo, percorrendo todas as posições do campo, onde a última posição corresponde à posição que apresenta menor erro, ou seja, é a posição onde o robô se encontra.

É de notar que caso o robô se encontre nas extremidades do campo o número de transições de linha dentro de campo poderá ser menor, uma vez que várias transições ao serem mapeadas podem encontrar-se fora dos limites do campo.



a)



b)

Figura 30 – a) Imagem obtida; b) Funcionamento do algoritmo de Localização Global

De forma a melhor entender o processo de localização é apresentado um esquema, Figura 31, que pretende retratar o processo de localização global, ou seja, a primeira vez que o processo de localização é realizado.

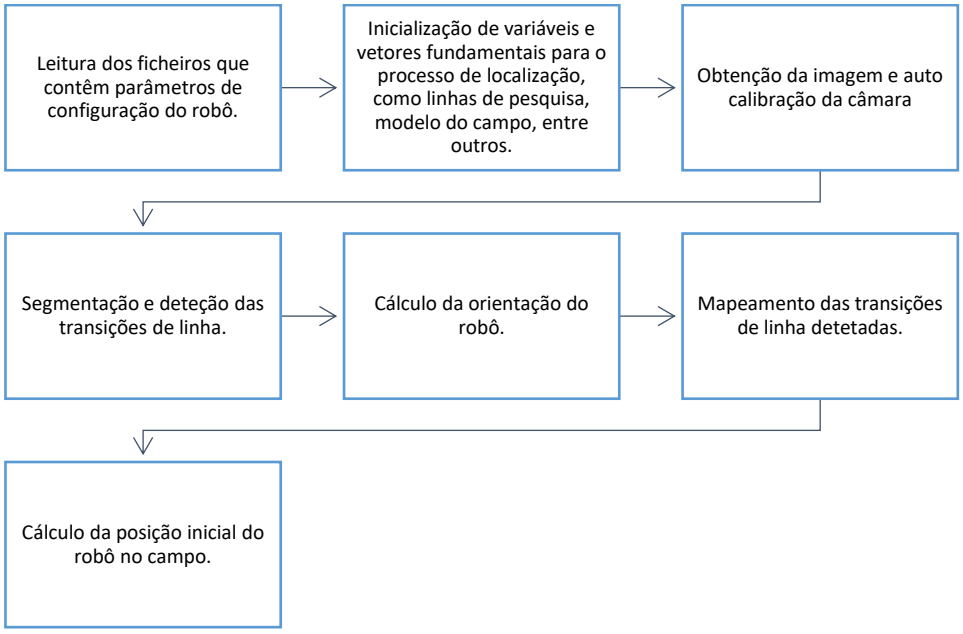


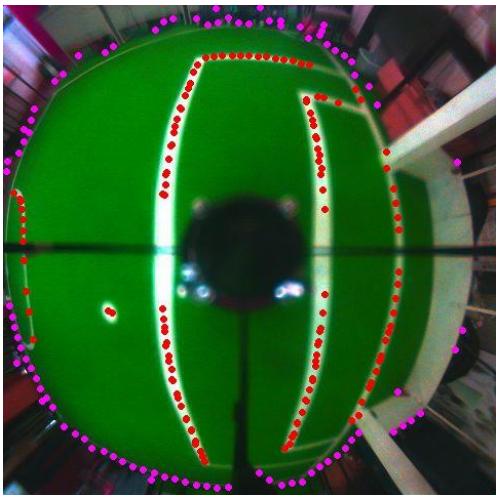
Figura 31 - Etapas do processo de localização global

Após a conclusão de qual a posição inicial do robô no campo, segue-se então a localização local, pois já se possui informação acerca da sua posição.

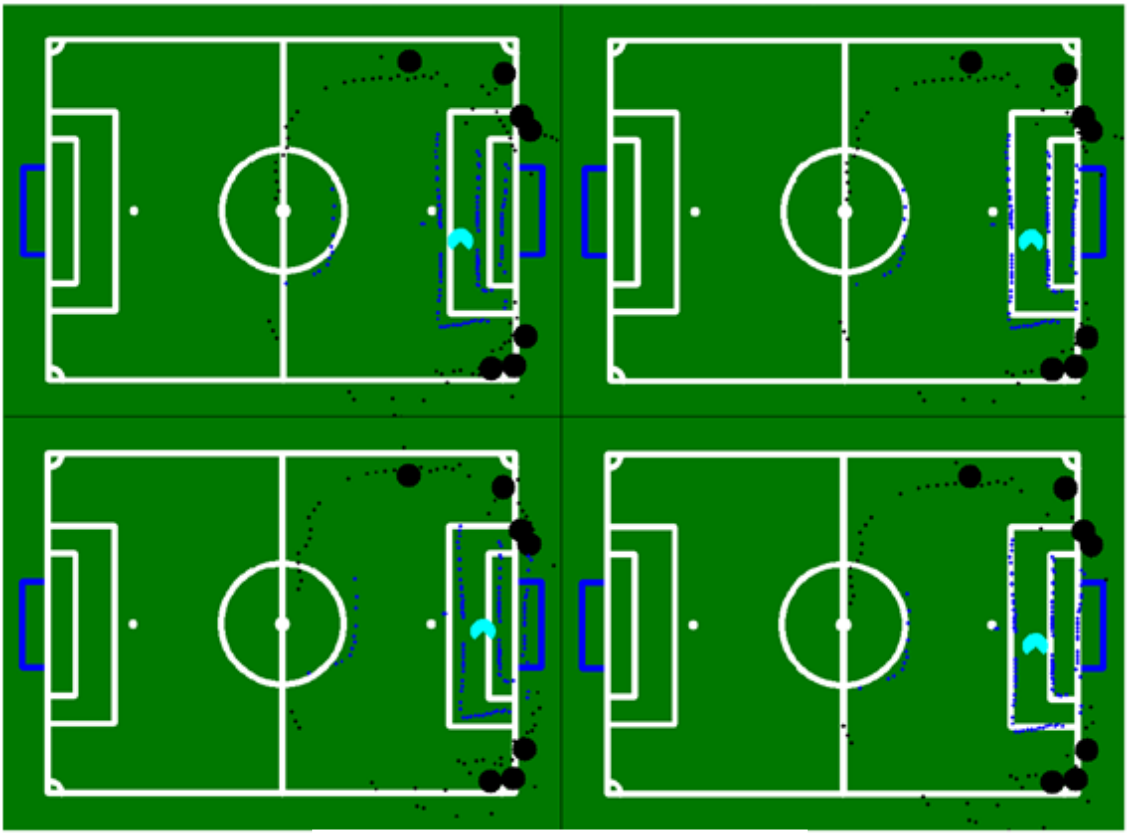
Como já fora referido, a localização local ou posicionamento local, tem também por base as transições de linha detetadas e o algoritmo de processamento, procurando a posição que melhor se ajusta ao modelo do campo de acordo com as transições de linha detetadas.

3.3.5.2. Local

O cálculo do posicionamento local utiliza o mesmo algoritmo do posicionamento global, porém, ao invés de percorrer a metade do campo em que o robô se encontra, executa uma procura na vizinhança da localização anterior do robô. Por outras palavras, de acordo com a posição anterior do robô no campo, é executada uma procura num quadrado de $0{,}6\ m\ x\ 0{,}6\ m$ em torno da mesma, sendo que é utilizado o mesmo processo que na localização global para achar a nova posição no campo limitando-se apenas a área em que este é aplicado.



a)



b)

Figura 32 – a) Imagem obtida; b) Funcionamento do algoritmo de Localização Local

A Figura 32 demonstra o processo realizado na localização local, onde apenas as posições pertencentes ao quadrado de procura são percorridas, de forma a obter a nova posição no campo em que o robô se encontra. É possível observar o deslocamento do robô entre imagens e o diferente mapeamento de cada ponto, pretendendo-se retratar o

funcionamento do algoritmo à procura da posição que melhor se ajusta ao modelo do campo, sendo que a última representação indica a posição correta do robô no campo.

De modo a melhor compreender o processo da localização local, é apresentado um esquema, Figura 33, retratando todas as etapas pertencentes à mesma.

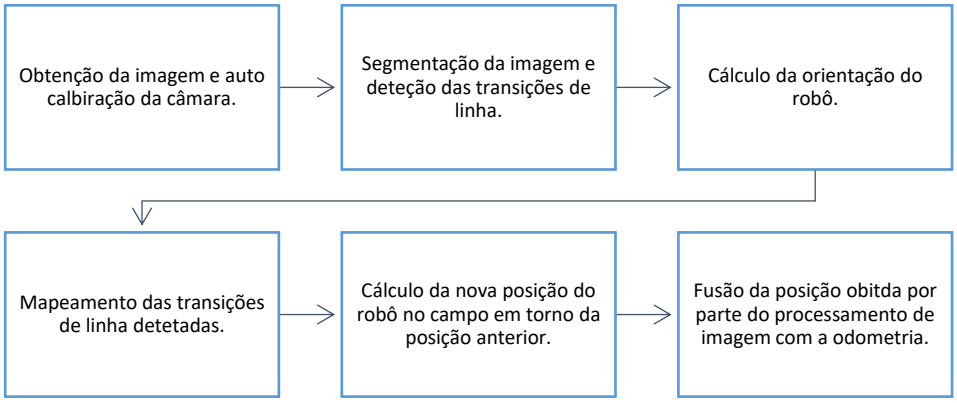


Figura 33 - Etapas do processo de localização local

No caso da localização global o tempo de processamento não é um fator preponderante, uma vez que esta é executado anteriormente ao início do jogo. Porém quando se trata do processo de localização local o tempo é um fator de enorme importância pois, tem de ser executado o máximo de vezes de forma a, tanto a *BaseStation* como o robô, possuírem uma localização o mais correta possível do posicionamento do mesmo no campo.

Portanto, de forma a otimizar e garantir que a localização local demora o mínimo tempo possível, sem nunca comprometer a sua eficiência, é necessário definir o quadrado de procura, sendo que para tal é preciso ter em conta o número de imagens que se obtém por parte da câmara por segundo, bem como da velocidade máxima do robô.

A câmara fornece imagens a uma frequência de $30 H_z$, ou seja, fornece uma nova imagem a cada $33\ ms$, e a velocidade máxima do robô é de $2\ m/s$. De forma a garantir uma maior precisão no cálculo da posição do robô no campo, optou-se então por utilizar todas imagens enviadas por parte da câmara, possuindo assim $33\ ms$ para a realização de um processo de localização local. Como a velocidade máxima do robô é de $2\ m/s$, sabemos que entre cada imagem o robô no máximo move-se $0{,}06\ m$ em qualquer direção.

Porém, decidiu-se definir o quadrado de pesquisa para um valor dez vezes maior, $0{,}6\ m\ x\ 0{,}6\ m$, devido à possibilidade de não sincronismo entre o processo de localização e a obtenção de imagens o que poderia levar à perda da posição do robô no campo.

Após a obtenção da posição do robô no campo com recurso ao processamento de imagem, é feita uma fusão com a odometria usando o filtro de Kalman, como pode ser observado na Figura 33, na última etapa, tal como acontece no cálculo da orientação do robô onde se executa a fusão da orientação obtida por parte do IMU e do processamento da imagem com a odometria. As etapas de obtenção dos valores de odometria para o deslocamento em X e Y, é exatamente igual ao citado no subcapítulo Orientação, 3.3.3, ou seja, é utilizado o mesmo algoritmo para obtenção das velocidades bem como da sua tradução para deslocamento.

Como se possui duas variáveis para realizar a fusão, movimento em X e Y, é feito um filtro de Kalman para cada uma delas, por outras palavras, cria-se um Kalman de duas variáveis sem correlação, um filtro de Kalman para o deslocamento em X e um para o deslocamento em Y, sendo esta fusão em tudo semelhante à fusão que ocorre no subcapítulo Orientação. As equações para fusão do deslocamento em X podem ser observadas na Tabela 7.

Tabela 7 - Equações do filtro de Kalman para fusão dos valores de deslocamento em X

Previsão
Estimativa do estado
$\widehat{\boldsymbol{X}}^{-}_{\boldsymbol{Posição\ Atual\ X}} = \boldsymbol{X}_{\boldsymbol{Ultima\ Posição\ X}} + \boldsymbol{X}_{\boldsymbol{odometria\ X}}$
Estimativa da covariância
$\boldsymbol{P}^{-}_{\boldsymbol{Posição\ Atual\ X}} = \boldsymbol{P}_{\boldsymbol{Ultima\ Posição\ X}} + \boldsymbol{Q}_{\boldsymbol{Deslocamento\ X}}$
Correção
Cálculo do ganho de Kalman
$\boldsymbol{K}_{\boldsymbol{Posição\ Atual\ X}} = \boldsymbol{P}^{-}_{\boldsymbol{Posição\ Atual\ X}}\left(\boldsymbol{P}^{-}_{\boldsymbol{Posição\ Atual\ X}} + \boldsymbol{R}_{\boldsymbol{Deslocamento\ X}}\right)^{-1}$
Correção do estado
$\widehat{\boldsymbol{X}}_{\boldsymbol{Posição\ Atual\ X}} = \widehat{\boldsymbol{X}}^{-}_{\boldsymbol{Posição\ Atual\ X}} + \boldsymbol{K}_{\boldsymbol{Posição\ Atual\ X}}(\boldsymbol{X}_{\boldsymbol{Visão\ atual\ X}} - \widehat{\boldsymbol{X}}^{-}_{\boldsymbol{Posição\ Atual\ X}})$
Correção da covariância
$\boldsymbol{P}_{\boldsymbol{Posição\ Atual\ X}} = (\boldsymbol{1} - \boldsymbol{K}_{\boldsymbol{Posição\ Atual\ X}})\ \boldsymbol{P}^{-}_{\boldsymbol{Posição\ Atual\ X}}$

A fusão com recurso ao filtro de Kalman permite obter uma estimativa da posição do robô no campo mais precisa e robusta, uma vez que este é um estimador ótimo, como já fora referido. É de notar que este passo não ocorre na localização global, pois a odometria não possuiria qualquer valor e também porque como se trata da primeira vez

que se obtém a posição do robô e, devido à forma como foi aplicado, é impossível utilizar o filtro de Kalman pois, este necessita de saber a posição anterior.

3.4. *WORLD MODEL*

O processo de localização, como já fora mencionado, não se refere apenas à obtenção da posição no campo, mas também à obtenção da posição da bola no campo bem como dos obstáculos que rodeiam o robô, sendo estes obstáculos outros robôs, tanto da própria equipa como da equipa adversária.

Portanto, o processo de localização no seu todo tem de ser capaz de identificar e recriar o ambiente que rodeia o robô. Este é um aspeto fundamental pois, a *BaseStation* define o papel de cada robô consoante o estado de jogo, sendo este estado de jogo construído pela informação enviada por cada robô para a mesma, e ainda por o controlo executado sobre o robô necessitar também da informação de onde se encontram obstáculos de forma a evitar colisões, o que poderia causar dano nos robôs, mas também faltas durante o jogo.

Desta forma a criação de um *World Model* é de enorme importância, sendo que neste subcapítulo são descritas todas as etapas necessárias para a criação do mesmo. A criação do ambiente que rodeia o robô inicia-se pela sua deteção, que passo a explicar mais pormenorizadamente.

A construção do ambiente que rodeia o robô utiliza as *scanlines* que são utilizadas para análise da imagem, tal como se utilizam para a deteção das linhas de campo. A deteção, tanto da bola como dos obstáculos, é realizada utilizando o mesmo método que se utiliza para deteção das linhas de campo. São procuradas transições nas *scanlines*, mas em vez de se procurar transições do tipo verde-branco ou branco-verde, como se faz para as linhas, neste caso são procuradas transições do tipo verde-preto ou preto-verde para deteção dos obstáculos e transições do tipo verde-laranja ou laranja-verde para a bola.

3.4.1. Bola

Como fora supracitado, o método utilizado para deteção da bola é o mesmo que se utiliza para deteção das linhas de campo, ou seja, é feita uma procura nas linhas de pesquisa por transições de cores, neste caso verde-laranja ou laranja-verde. Na Figura 34 pode observar-se um exemplo de linha de pesquisa onde esta representada um exemplo do que poderia ser considerado uma transição válida de um ponto de bola.

Figura 34 - Exemplo de scanline para deteção da bola

Tal como acontece na deteção de pontos de linha, são detetados vários pontos de bola, onde após a sua deteção, tal como acontece com os pontos de linha, estes são mapeados e guardados num vetor, sendo guardadas as coordenadas (X, Y) bem como o comprimento da cor de interesse, neste caso cor laranja.

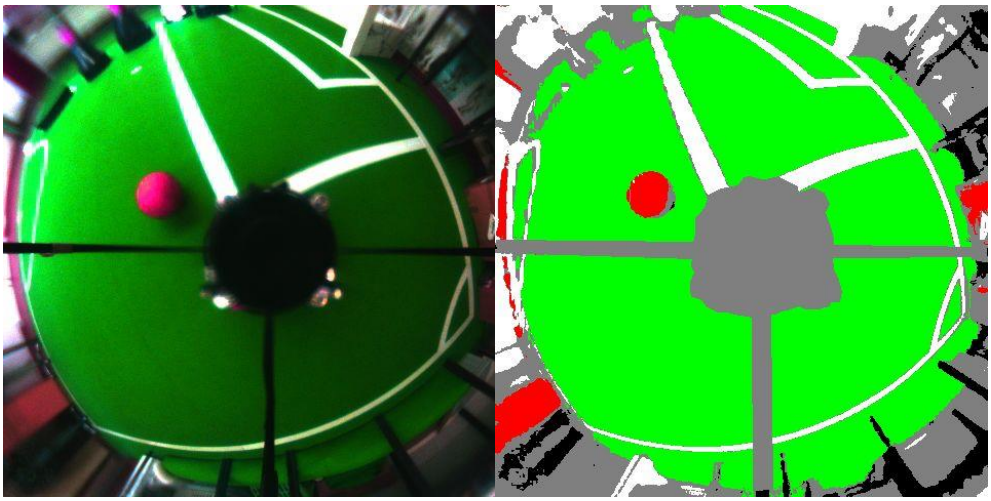
Porém os pontos de bola detetados não podem ser considerados como sendo uma bola pois, um ponto apenas não demonstra qualquer tipo de fiabilidade.

Portanto, decidiu-se tentar fazer um *merge* dos pontos de bola detetados isto porque caso um ponto de bola seja encontrado e este seja mesmo pertence à bola, deverão aparecer mais pontos perto deste. Esta fusão de pontos consiste em percorrer todos os pontos de bola detetados e verificar se na sua vizinhança se encontra mais algum ponto, caso se encontre é feito o *merge* dos dois pontos, calculando o ponto central entre os dois, criando assim uma possível bola. Na eventualidade de se encontrar um terceiro ponto, este é fundido aos dois pontos anteriores calculando-se o ponto central entre os três, sendo que este processo se repete para qualquer que seja o número de pontos.

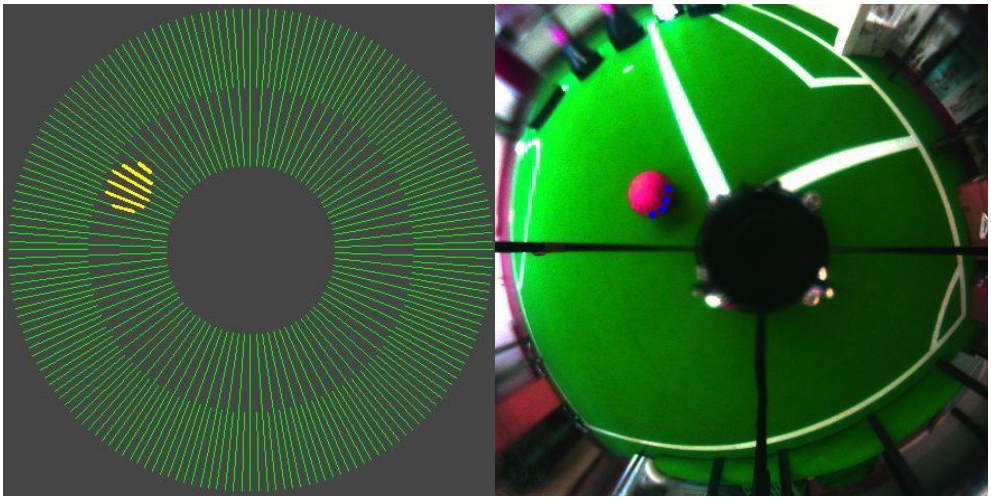
De forma a executar uma validação da bola, os números de pontos é a chave pois, foi definido que caso nenhuma das possíveis bola possua mais de três pontos então diz-se que o robô não tem informação acerca da posição da bola. É de salientar que o número de pontos de forma a validar a bola é dependente da calibração do robô, sendo este lido aquando da inicialização do processo de localização no seu geral.

Na Figura 35 é possível observar a imagem obtida por parte da câmara, a), a imagem segmentada em que a bola se apresenta vermelha alaranjada, b), a imagem com a representação da informação contida nas linhas de pesquisa, c), onde a informação referente à bola está representada pela cor amarela, a representação das transições detetadas na imagem obtida, d), e por fim o mapeamento da mesma para o campo de jogo.

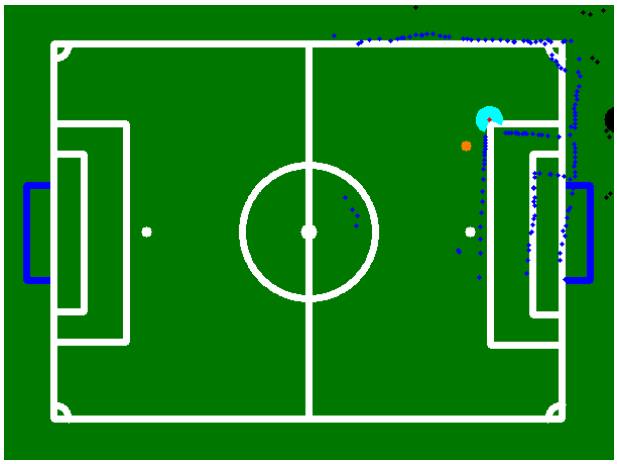
Tal como as transições de linha detetadas são um conjunto de pontos, as transições de bola detetadas também são um conjunto de pontos. No caso das linhas foi escolhido o ponto central da transição para ser a coordenada (X, Y) da mesma.



a) *b)*



c) *d)*



e)

Figura 35 – a) Imagem obtida; b) Imagem Segmentada; c) Imagem com informação das scanlines; d) Imagem obtida com representação das transições detetadas; e) Mapeamento da bola no campo de jogo

Porém no caso da bola a abordagem foi ligeiramente diferente uma vez que é preferível obter a coordenada do primeiro ponto da transição pois, obtemos um dos pontos da superfície da bola que se encontra em contacto com o solo, possibilitando desta forma estimar a posição a bola no campo com uma maior precisão do que se utilizar o ponto central da transição detetada.

Para deteção da bola apenas são utilizadas as linhas de pesquisa radiais, pois apenas estas detetam os pontos da bola que se encontram em contacto com o solo.

Após a deteção da bola e se encontrar validada, caso já exista uma outra bola na imagem anteriormente enviada pela câmara, é calculada a distância entre as duas, caso esta distância seja superior a um valor já estipulado aquando da inicialização do programa, a nova bola é descartada mantendo-se a bola encontrada anteriormente como a bola real. Caso a distância entre ambas as bolas seja inferior ao valor estipulado, a bola é então atualizada, guardando a posição da nova bola. Mas se não existir qualquer bola já guardada, ou seja, em memória e surjam duas bolas distintas na mesma imagem, a bola que contiver mais pontos de bola detetados, será a bola a ser considerada.

Uma bola depois de validada, pode ser sobreposta por outra bola, sendo necessário a validação da distância entre as duas, como explicado acima, ou então pode ser perdida, isto é, o robô passa a não saber a posição da bola caso não apareça nenhum candidato desta num número máximo de imagens consecutivas.

De forma a melhor entender todo o processo, é apresentado um fluxograma na Figura 36, que pretende retratar o funcionamento do algoritmo quando existem várias transições de bola detetadas.

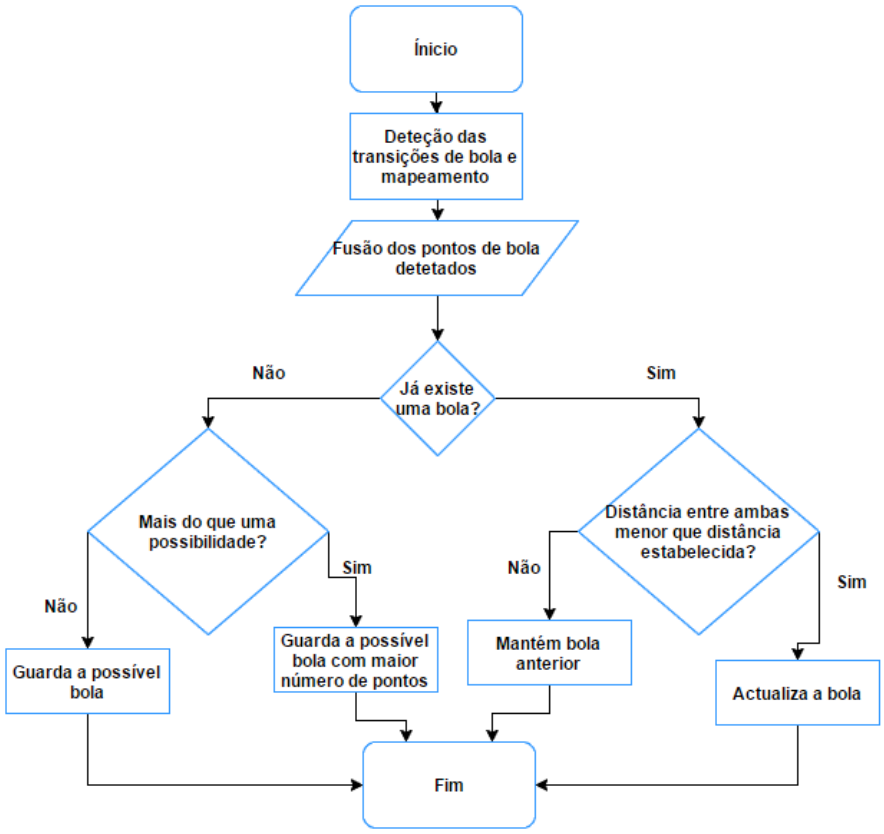


Figura 36 - Fluxograma de validação da bola

3.4.2. Obstáculos

O método utilizado para deteção dos obstáculos é o mesmo que se utiliza para deteção das linhas de campo e da bola, mas neste caso a procura nas linhas de pesquisa é feita procurando transições do tipo verde-preto e preto-verde.

Na Figura 37 visualiza-se um exemplo de uma transição válida contida numa linha de pesquisa quando se efetua a procura para obstáculos, onde a cor castanha está a representar a cor preta, a cor do obstáculo detetado na *scanline*.

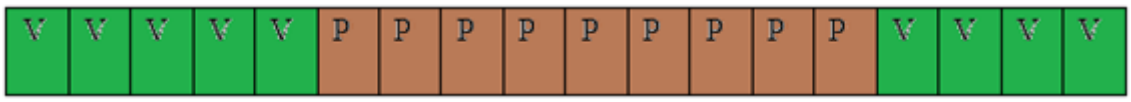


Figura 37 - Exemplo de scanline para deteção de obstáculos

A deteção de obstáculos é em tudo muito semelhante à deteção das linhas bem como da bola. Na análise das linhas de pesquisa vários pontos de obstáculos são detetados, onde estes posteriormente são mapeados e guardados num vetor.

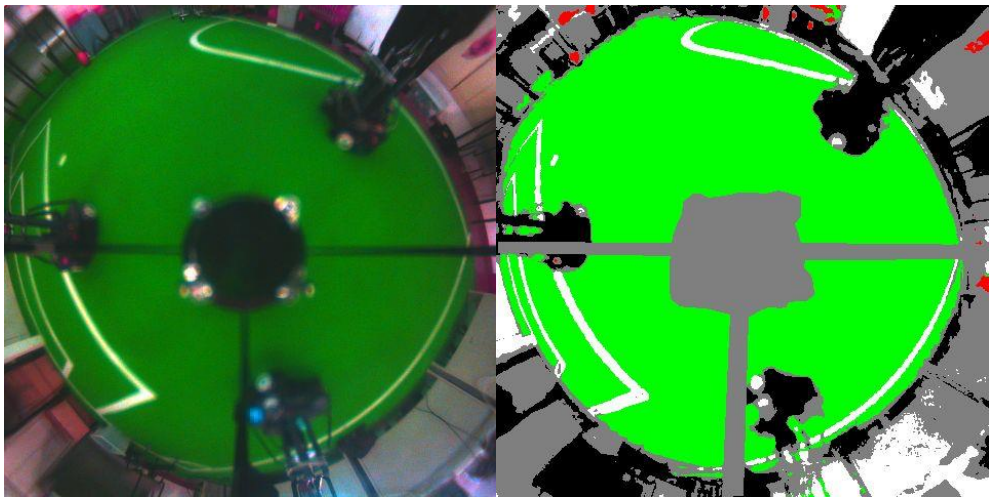
Num jogo podem existir obstáculos de dois tipos, outros robôs ou os limites físicos do campo, sendo que, de forma a diferenciar ambos é feita uma verificação se o ponto de obstáculo detetado está dentro dos limites do campo ou não, onde de forma a limitar as

posições por onde o robô se pode movimentar optou-se por utilizar o primeiro ponto da transição detetada, ao invés de qualquer outro, pois, este é o ponto que se encontra mais perto do robô.

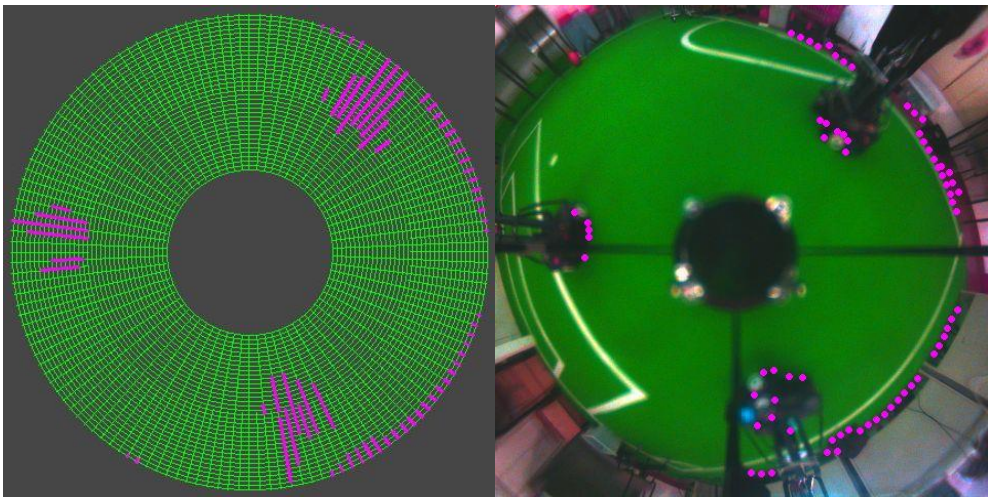
Após a verificação se os pontos se encontram fora ou dentro do campo, os pontos que se encontram dentro do campo são analisados novamente. Tal como acontece nos pontos de bola detetados, é feito um *merge* dos pontos de obstáculos detetados pois, devido à forma dos robôs quando um ponto de obstáculo é detetado e este pertence efetivamente a um obstáculo, mais pontos de obstáculos devem estar presentes na sua vizinhança.

Portanto, o vetor de obstáculos é percorrido e é verificado se algum outro ponto de obstáculo se encontra perto do mesmo, caso se encontre e esteja a uma distância máxima, definida aquando da inicialização do programa, então é feita a fusão dos dois pontos, como acontece no caso dos pontos de bola. Por fim quando todos os pontos tiverem sido analisados, apenas os possíveis obstáculos que contenham mais de três pontos são considerados obstáculos, todos os outros são descartados devido a puderem introduzir informação errada acerca do posicionamento dos obstáculos. É de salientar que o número de pontos que possibilitam a formação de um obstáculo válido pode ser alterado nos ficheiros de configuração do robô ou via *Vision Calib*.

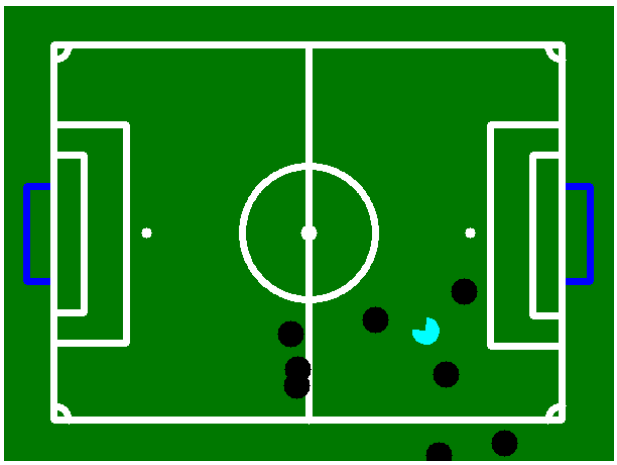
Na Figura 38 pode visualizar-se a imagem obtida, a imagem segmentada com os obstáculos representados pela cor preta, a representação da informação contida nas *scanlines* onde a cor roxa se encontra as transições de obstáculos detetadas, os pontos de obstáculos detetadas e representados na imagem obtida pela câmara e por fim a fusão e representação dos obstáculos identificados no campo de jogo.



a) *b)*



c) *d)*



e)

Figura 38 – a) Imagem obtida; b) Imagem Segmentada; c) Imagem com informação das scanlines; d) Imagem obtida com representação das transições detetadas; e) Mapeamento dos obstáculos no campo de jogo

3.5. COMUNICAÇÃO

A comunicação no robô é uma das peças fulcrais pois, possibilita a troca de informação entre os diversos robôs bem como com a *BaseStation*, sendo que esta desempenha o papel de treinador durante um jogo de futebol. Para tal, devido às capacidades demonstradas no subcapítulo 2.2, Fundamentos teóricos, o ROS foi a escolha para implementação das comunicações.

De forma a melhor entender toda a arquitetura, é necessário possuir uma ideia acerca de todos os processos que são executados num robô. Um robô é capaz de agir autonomamente pois, possui um sistema de localização, sendo responsável por saber onde o robô se encontra no campo bem como modelar o mundo em que se encontra, com recurso a câmaras, sensores, etc. Possui um sistema de controlo do hardware que lhe permite controlar os atuadores, ou seja, permite-lhe mover, chutar e driblar a bola e ainda obter leituras dos sensores da base, detém um sistema de inteligência artificial e controlo, que consoante o papel e a tarefa que lhe foi atribuída, calcula o melhor trajeto que o robô tem de efetuar e tem também um sistema de comunicação que permite o envio de informação, mas também a receção.

Portanto, após o conhecimento de todas as tarefas presentes num robô, é possível gerar a arquitetura da rede ROS desenvolvida.

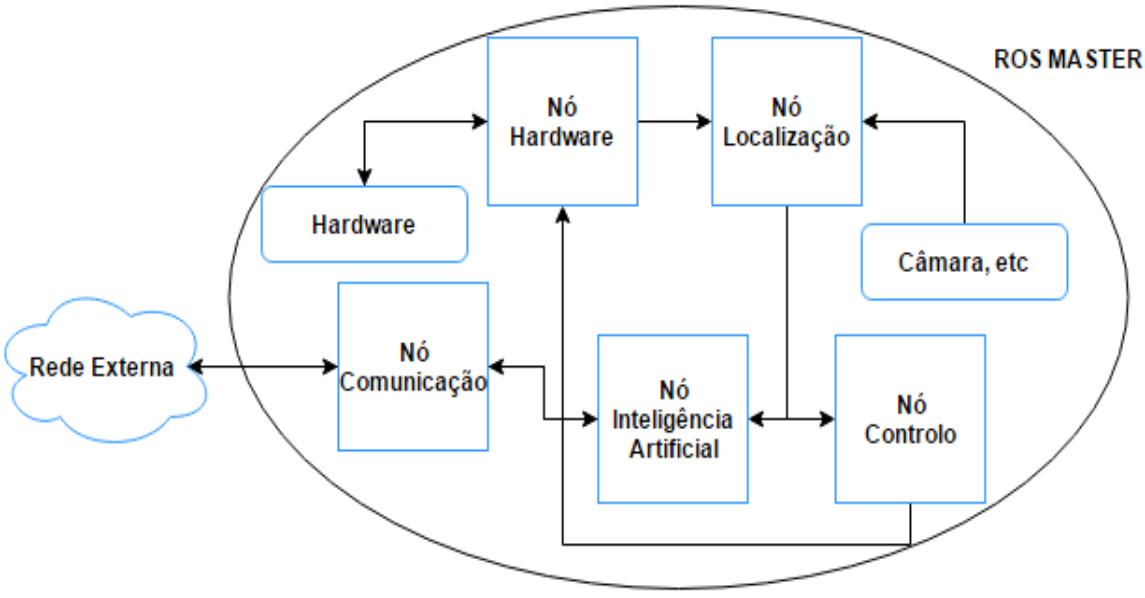


Figura 39 - Arquitetura da rede ROS num robô

Na Figura 39 é possível visualizar a arquitetura da rede ROS num robô. Esta, a rede implementada, utiliza diversas mensagens e serviços de forma a responder às necessidades apresentadas, tanto pelo utilizador como pelo robô.

Após melhor se entender a rede implementada, é possível perceber a comunicação que acontece no nó da localização. Este recebe informações do nó de hardware, da câmara e outros sensores, processa essa informação, obtém a posição do robô no campo e ainda modela o mundo que rodeia o robô.

De seguida, através de mensagens de ROS criadas especificamente para esta função, envia as informações adquiridas, como por exemplo a sua posição, posição da bola, número de obstáculos e a sua posição, entre outras, para os nós de controlo e inteligência artificial através da publicação da mensagem no tópico desse tipo de mensagens, para que, de acordo com a informação recolhida estes definam o próximo passo que o robô deve tomar.

Durante a realização de testes sobre os métodos desenvolvidos, surgiu a necessidade de efetuar calibrações no robô a partir do computador. Uma vez que o ROS permite a comunicação entre diferentes máquinas que se encontrem na mesma rede, TCP-ROS, foram então desenvolvidas as ferramentas de configuração, como o *Vision Calib*. Com a presente arquitetura, cada robô corre a sua própria rede ROS independente, ou seja, cada um possui um Master, sendo que para se ligar a este a partir de um computador, de forma a possibilitar a configuração, apenas é necessário saber o IP e a porta do Master que está no robô com que se pretende comunicar.

Desta forma, é possível comunicar diretamente do computador com o robô que se pretende, facilitando a comunicação entre ambos pois, tira partido dos mecanismos oferecidos pelo ROS e sem causar grande congestionamento na rede.

3.6. *VISION CALIB* – FERRAMENTA DE CALIBRAÇÃO

O *Vision Calib* é o software, desenvolvido no âmbito desta dissertação, que permite a alteração de praticamente todos os parâmetros do robô referentes à localização. Este foi desenvolvido com o intuito de qualquer utilizador poder calibrar os robôs em vários parâmetros, sem necessitar de alterar código nem de o recompilar, onde com o simples deslizamento de algumas barras, a escrita de alguns valores e o pressionar de alguns botões é possível alterar quase por completo toda a configuração do robô.

Quando o software é inicializado dá-se uma troca de mensagens de ROS entre o robô, a qual se conecta, e o software, de forma a este obter todas as informações acerca da calibração do robô, como por exemplo os valores dos limites da *look up table*, o centro da imagem, a localização das áreas utilizadas para a auto calibração da câmara bem como das suas áreas, entre outros. Como já fora referido, os parâmetros que podem ser alterados de forma a melhor calibrar o robô, encontram-se guardados sob a forma de mensagens de ROS, uma vez que este tipo de mensagens possibilitam o uso dos dados das suas mensagens de uma forma simples e útil.

O software possui três opções distintas de calibração, sendo que cada uma delas é utilizada para calibrar diferentes parâmetros do robô. Quando este, o software, é inicializado é apresentada a primeira opção, LUT&MIRROR que é explicado de seguida.

3.6.1. LUT&MIRROR

A primeira opção a ser apresentada no *Vision Calib* é o LUT&MIRROR, esta pode ser visualizada na Figura 40.

O LUT&MIRROR possibilita a calibração de vários aspetos relacionados com o sistema de visão, a segmentação e mapeamento dos pontos de interesse detetados. É possível alterar os valores limite que dão origem à *look up table*, é possível alterar o centro da imagem, alterar o tamanho das linhas de pesquisa, definir a frente do robô, criar uma nova máscara de forma ao corpo do robô não ser processado.

Neste é também possível alterar em tipos de imagens que pretendemos visualizar, como imagem obtida, segmentada, imagem com pontos de linhas, obstáculos ou bola detetados, imagem com informação armazenada nas linhas de pesquisa e ainda uma imagem onde os pontos de interesse detetados são previamente mapeados em função da distância ao robô. O LUT&MIRROR permite também alterar a *frame rate* com que a câmara obtém imagens, possibilita também, de uma forma bastante simples, a calibração dos dois vetores cruciais para o bom funcionamento da localização, o vetor de distância real e o vetor de distância em pixéis e ainda dá informação acerca da distância em pixéis, desde o centro da imagem até ao ponto em que o rato do computador se encontrar na imagem.

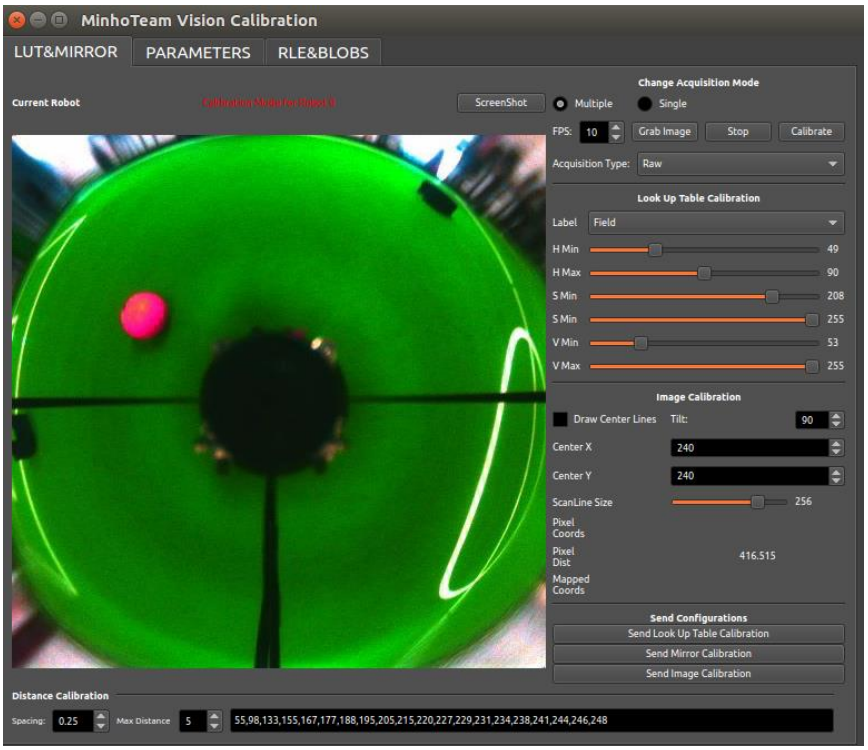


Figura 40 - Vision Calib tab LUT&MIRROR

3.6.2. PARAMETERS

A segunda opção é o PARAMETERS, Figura 41, é responsável por possibilitar a calibração de aspetos relacionados com o algoritmo de auto calibração da câmara.

Neste é possível ativar ou desativar a auto calibração da câmara, alterar o valor das propriedades da mesma, como brilho, saturação, ganho, exposição, entre outros, bem como alterar os ganhos do controlador PID de cada uma e até mesmo reiniciar o controlador.

Permite ainda definir a posição na imagem das regiões, branca e preta, necessárias no algoritmo de auto calibração, bem como definição das suas áreas, sendo também possível alterar os valores de referência tanto da saturação como da luminância, possibilitando desta forma um maior controlo da qualidade da imagem obtida.

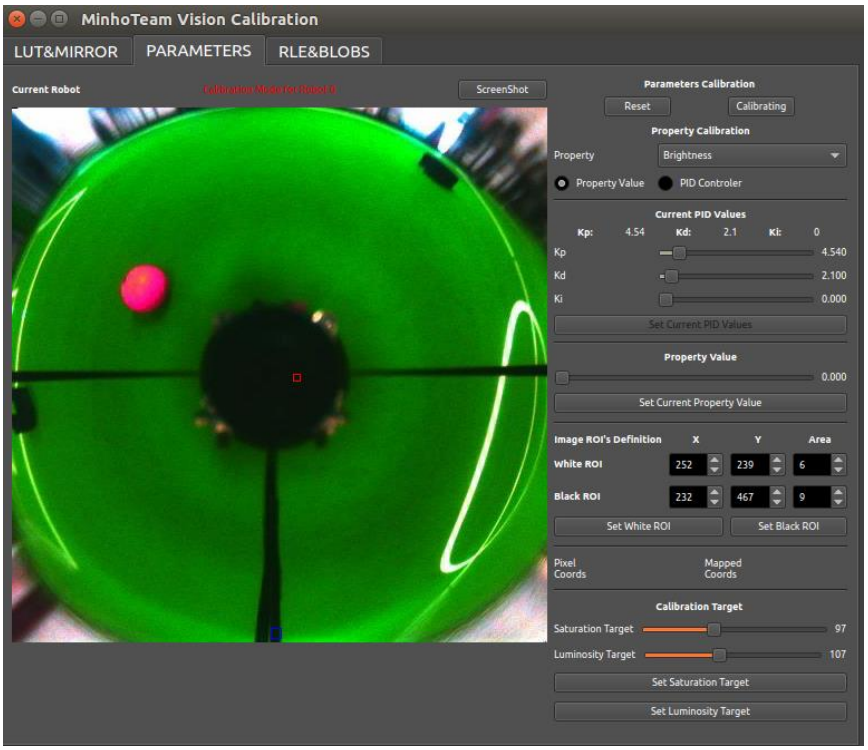


Figura 41 - Vision Calib tab PARAMETERS

3.6.3. RLE&BLOBS

A terceira opção, e última, Figura 42, RLE&BLOBS permite a calibração de vários parâmetros relacionados com a criação do *World Model*.

Neste é possível realizar a alteração das distâncias de fusão, tanto da bola como dos obstáculos, para criação do modelo do mundo em que o robô se encontra, permite alterar os parâmetros para procura de pontos de bola nas linhas de pesquisa. É possível também calibrar o filtro de Kalman referente à fusão da odometria com localização obtida por processamento de imagem, alternar entre os vários tipos de imagem, tal como no *tab* LUT&MIRROR.

É também possível calibrar o vetor utilizado pelo filtro para validação dos pontos de linha detetados, através da possibilidade de ativar ou desativar uma linha de pesquisa na imagem que fornece informação acerca de quantos pixéis a transição de linha detetada possui, e ainda definir a intensidade com que o filtro será aplicado.

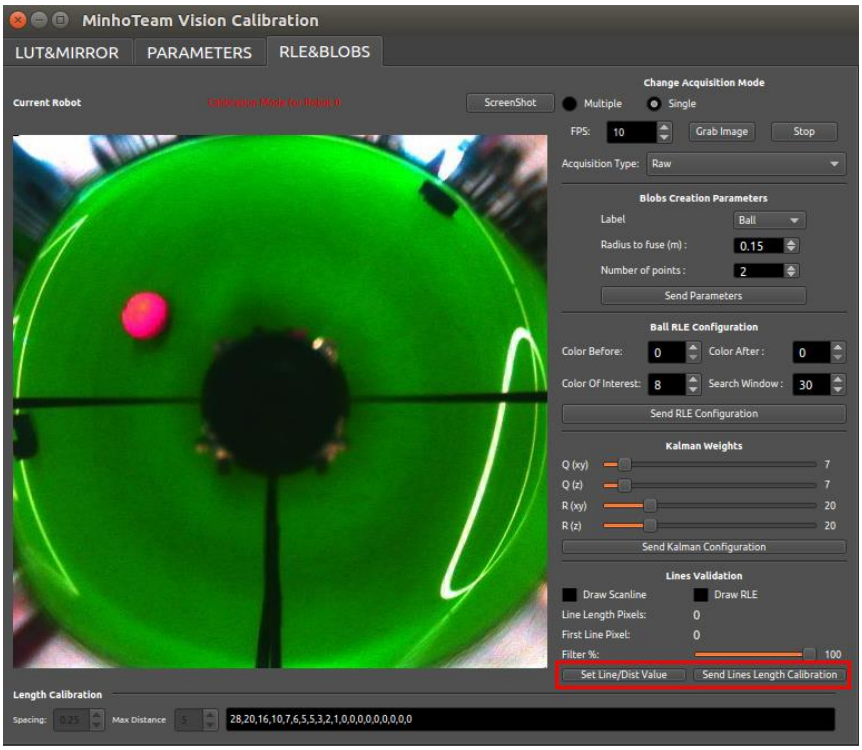


Figura 42 - Vision Calib tab RLE&BLOBS

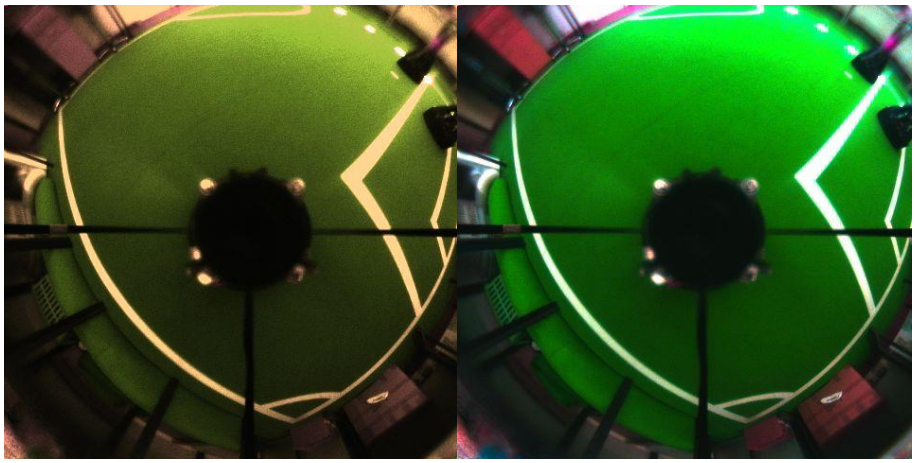
Em todas as opções é possível guardar a imagem que está a ser apresentada. É de salientar que sempre que se pretende guardar alguma configuração realizada, basta premir os botões de *Set* ou *Send*, destacados por um retângulo vermelho na figura acima, e as alterações são enviadas imediatamente para o robô através de mensagens de ROS, sendo que este guarda a configuração recebida, e a escreve para o ficheiro correspondente informação recebida, fazendo com que desta forma todos os parâmetros estejam atualizados aquando da inicialização do processo de localização.

4. RESULTADOS

Neste capítulo, são apresentados os resultados obtidos após o desenvolvimento de todas as etapas necessárias para a obtenção da localização do robô no campo bem como da modelação do mundo que rodeia o robô e de todas as tarefas realizadas que permitiram uma melhoria geral para a equipa MINHO TEAM.

4.1. AUTO CALIBRAÇÃO DA CÂMARA

Para testar o algoritmo desenvolvido, foram realizados diversos testes com diferentes configurações iniciais. Na Figura 43 pode observar-se que após a execução do algoritmo sobre as configurações da câmara, a imagem obtida apresenta uma melhor qualidade de imagem no seu geral.



a) *b)*

Figura 43 – a) Imagem obtida antes do algoritmo de auto calibração ser aplicado; b) Imagem obtida depois do algoritmo de auto calibração ser aplicado

O tempo médio que o algoritmo demora a obter uma imagem correta é de aproximadamente 4 segundos, isto com a câmara a 30fps e a partir de configurações parecidas com as apresentadas na imagem. Este tempo apenas foi medido de forma a possuir-se uma ideia acerca do tempo para obtenção de uma boa imagem pois, este processo é realizado aquando do inicio do jogo não sendo relevante o tempo que este demora a ser realizado. Porém, este pode ser diminuído com ajustes no controlador PID de cada parâmetro da câmara.

Um dos objetivos pretendidos com o desenvolvimento deste algoritmo é a sua utilização em *real-time*, isto é, durante o jogo pois, as condições no campo podem alterar-

se um pouco durante o mesmo, podendo ser necessário pequenos ajustes na calibração dos parâmetros da câmara. Desta forma foram realizados diversos testes, medindo o tempo médio que um ciclo do algoritmo demora a ser executado, realizando o ajuste de todos os parâmetros, sendo que o tempo obtido foi, em média, de 13 milissegundos. Este é demasiado elevado uma vez que apenas dispomos de 33 milissegundos para o processo de localização no seu geral, desde a calibração até à comunicação da posição do robô e modelação do mundo.

Após uma nova análise de todo o algoritmo, de forma a diminuir o peso computacional requerido, ou seja, reduzir o tempo necessário, foi decidido diminuir o número de pixéis da imagem avaliados para a obtenção dos histogramas utilizados pelo algoritmo de auto calibração. Como a imagem possui 230400 pixéis decidiu-se realizar testes utilizando metade, um quarto e um décimo do tamanho da mesma.

O resultado em termos de qualidade da imagem é praticamente idêntico uma vez que algoritmo contínua a convergir para os valores especificados, porém, os tempos obtidos são apresentados na Tabela 8.

Tabela 8 - Tempos de ciclo do algoritmo de auto calibração com análise de diferentes quantidades de pixéis

Pixéis analisados	**115200**	**57600**	**23040**
Tempo (milissegundos)	10,6	8,5	7

Os tempos obtidos continuam a ser significativamente elevados quando comparados com o tempo total que se dispõem para realizar o processo de localização geral, por outras palavras, o tempo necessário para obtenção da posição do robô e de modelação do mundo é de 17 milissegundos em média.

Porém pode demorar mais um pouco dependendo do número de transições detetadas, tanto de linhas como dos restantes objetos presentes num jogo de futebol robótico, sendo que se definiu então que fossem 22 milissegundos, no pior dos casos. Retirando este tempo aos 33 milisegundos, tempo total para o processo, apenas restam 11 milissegundos para a auto calibração da câmara, o que se torna demasiado pouco comparativamente aos tempos obtidos na Tabela 8 pois, existe a possibilidade de em alguns casos acontecer de o algoritmo demorar mais de 11 milissegundos.

No entanto decidiu-se diminuir o número de parâmetros calibrados pelo algoritmo, sendo que para tal a arquitetura do mesmo foi alterada. O algoritmo possui duas fases, na primeira fase, todos os parâmetros são calibrados, tal como já era realizado, mantendo-se

o algoritmo inicial. Na segunda fase, tal como acontece na primeira o algoritmo é igual, porém foi retirado o *white-balance* da calibração. O motivo de retirar o *white-balance* prende-se pelo facto de, após o estudo inicial se ter entendido que após a calibração do mesmo, muito dificilmente este se alteraria durante o jogo pois, a causa principal de alteração do seu valor ocorre no desligar e ligar da câmara.

A alteração que se propõe visa diminuir o tempo de processamento requerido pelo algoritmo, de forma a avaliar se a sua aplicação em *real-time* é possível, mas sem nunca comprometer a qualidade da imagem obtida.

De forma a testar a alteração proposta, realizaram-se exatamente os mesmos testes que se realizaram para a avalização do algoritmo inicialmente. A tabela que se segue apresenta os tempos obtidos para a análise de todos os pixéis da imagem, para metade, um quarto e um décimo do tamanho total da imagem.

Tabela 9 - Tempos de ciclo do algoritmo de auto calibração com análise de diferentes quantidades de pixéis quando não calibrado o white-balance

Pixéis analisados	**230400**	**115200**	**57600**	**23040**
Tempo(milissegundos)	11,3	8,1	6,4	5

Comparando os tempos obtidos apresentados na Tabela 8 com os tempos obtidos apresentados na Tabela 9, chega-se à conclusão que é percetível a redução de tempo que se pretendia, retirando em média cerca de 2 milissegundos a cada tempo obtido na Tabela 8. Com a utilização de um décimo da totalidade dos pixéis da imagem é possível manter a qualidade da imagem e possuir um tempo de 5 milissegundos de processamento, onde no pior dos casos demorará cerca de 10 milissegundos.

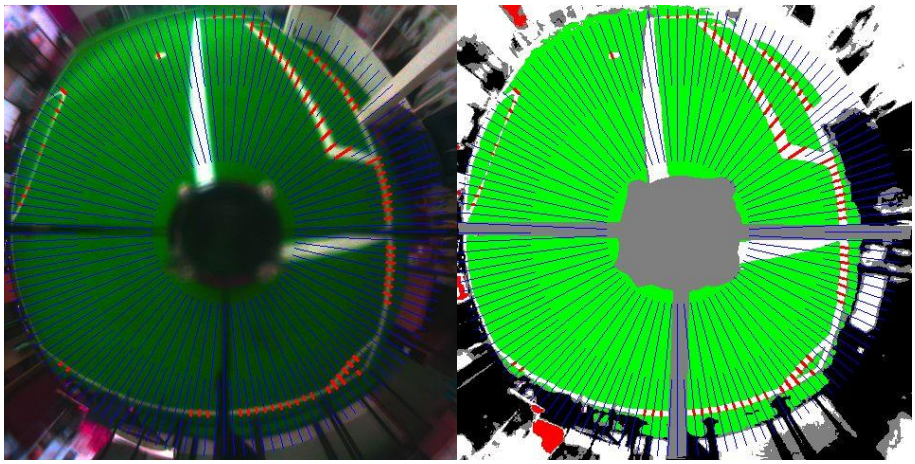
Portanto, pode concluir-se que sem comprometer a qualidade da imagem obtida é possível fazer com que o tempo de processamento requerido pelo algoritmo possibilite a sua utilização em tempo real. Decidiu-se, de acordo com os resultados obtidos, utilizar o algoritmo com a nova arquitetura desenvolvida, ou seja, com duas fases distintas, onde na primeira fase são calibrados todos os parâmetros da câmara utilizando a totalidade dos pixéis da imagem e na segunda fase são calibrados todos os parâmetros, exceto o *white-balance*, e são utilizados um décimo da totalidade dos pixéis.

Por fim é possível concluir, que tanto o objetivo da utilização *real*-time do algoritmo desenvolvido bem como da utilização do mesmo quando se liga o robô foram cumpridos. Desta forma é possível garantir que a qualidade da imagem, antes e durante,

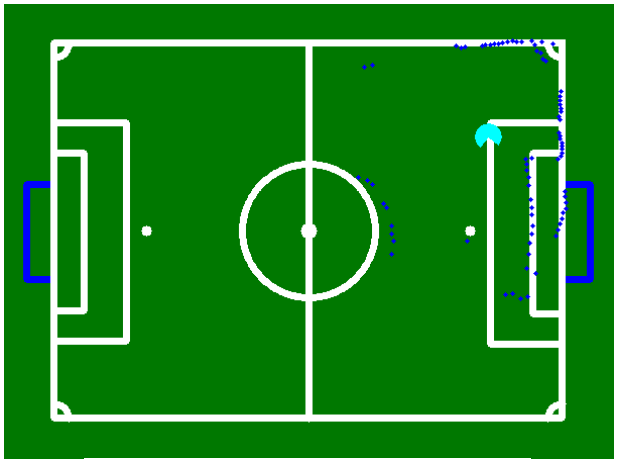
o jogo não é afetada, mas também por permitir que o operador do robô não necessite de saber como calibrar a câmara, nem os parâmetros que definem uma boa imagem, podendo este especializar-se noutra função, oferecendo também mais tempo ao operador para calibrar outros aspetos importantes para o robô, como a LUT de distâncias ou a LUT de cores.

4.2. DETEÇÃO DAS LINHAS

Na Figura 44 são apresentados os resultados obtidos para deteção de linhas por pesquisa radial.



a) *b)*



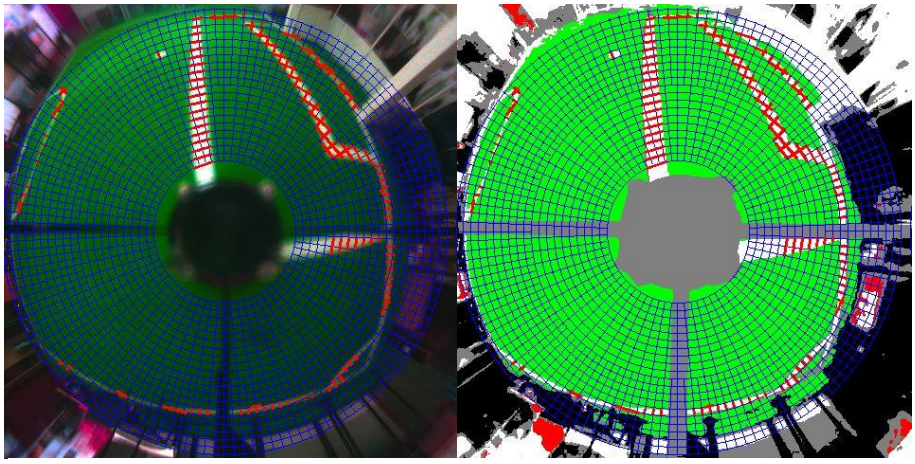
c)

Figura 44 – a) Imagem obtida com scanlines radiais representadas e informação das mesmas; b) Imagem segmentada com scanlines radiais representadas e informação das mesmas; c) Mapeamento dos pontos de linha no campo de jogo

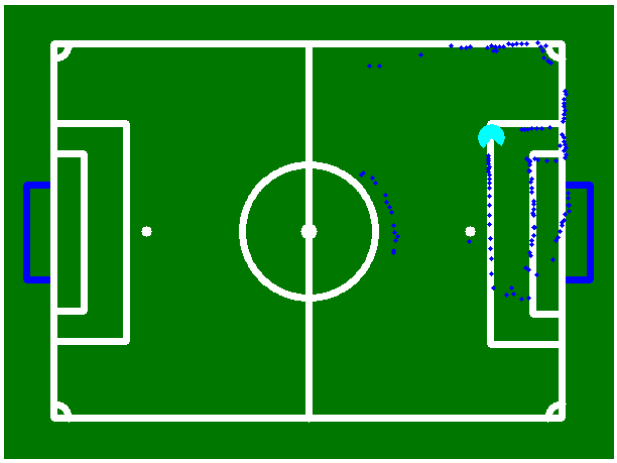
Pode-se observar a imagem obtida, onde representados a azul estão as linhas de pesquisa radiais, a imagem segmentada onde se encontram representadas novamente as

linhas de pesquisa, mas também as transições de linha detetadas representadas a vermelho. Por fim é possível observar-se uma representação do mapeamento dos pontos no campo de jogo, Figura 44 sendo desta forma possível obter uma representação da informação contida nas linhas de pesquisa radiais.

Visualizando a Figura 44 com maior atenção, é possível observar que caso a linha de pesquisa, *scanline*, se encontre inteiramente sobre uma linha de campo ou tenha início sobre a mesma, como acontece em algumas linhas de pesquisa que se encontram na imagem, estas não detetam qualquer transição, ou seja, caso o robô se encontre por cima de linhas, o que pode causar que linhas de pesquisa estejam contidas em algumas linhas de campo ou estas se encontrem alinhadas com alguma linha de pesquisa, não são detetadas transições dessa linha, ficando desta forma, vários pontos de linha por detetar.



a) *b)*



c)

Figura 45 - a) Imagem obtida com scanlines radiais representadas e informação das mesmas; b) Imagem segmentada com scanlines radiais representadas e informação das mesmas; c) Mapeamento dos pontos de linha no campo de jogo

Portanto, devido a este problema decidiu-se utilizar também a pesquisa axial, linhas de pesquisa circulares, sendo possível desta forma resolver o problema acima referido.

Na Figura 45 pode observar-se a utilização dos dois tipos de pesquisa, radial e axial bem como todas as transições de linha detetadas e a respetiva representação do mapeamento dos pontos de linha detetados para o campo de jogo.

Como já fora referido no subcapítulo Deteção de linhas, 3.3.2, as transições dos pontos de linha detetados são um conjunto de pontos aproximadamente com a mesma espessura que estas, as linhas, possuem na imagem. Optou-se por utilizar então o ponto central da transição pois, como as linhas do campo possuem uma espessura constante, no modelo do campo carregado para o robô aquando da inicialização do mesmo, existe mais do que uma coordenada em que o ponto central da transição encontrada, pode ser mapeado.

Desta forma, utilizando o ponto central da transição, diminui-se a possibilidade do ponto de linha detetado ser mapeado para uma coordenada que no modelo de campo não represente uma linha, aumentando então a certeza que o ponto de linha, ao ser mapeado, se encontre numa linha do modelo de campo.

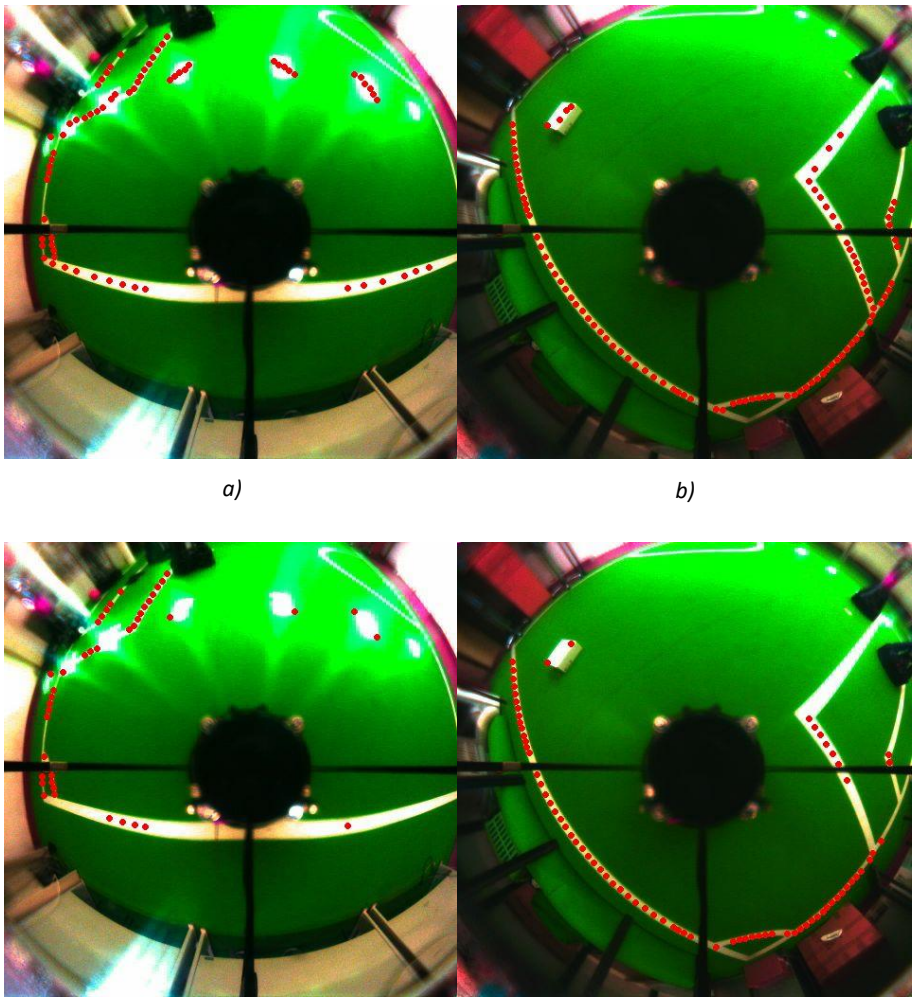
4.2.1. Filtragem

Após a implementação do método de filtragem estar concluída, realizou-se alguns testes de forma a perceber até que ponto este seria um método fiável.

Durante os testes apercebemo-nos de uma particularidade que viria a limitar a aplicação do filtro de pontos de linhas. As linhas de pesquisa axiais, por norma, quando detetam transição de linhas, ou seja, quando detetam pontos de linhas possuem um maior comprimento da cor de interesse pois, estas são capazes de detetar, como já fora referido, as linhas que se encontram junto ao robô ou por baixo do mesmo.

Portanto, quando se aplicava o filtro às mesmas, *scanlines* axiais, bastantes pontos de linha detetados eram descartados, porém decidiu-se utilizar o método nas linhas de pesquisa radiais uma vez que os resultados obtidos se apresentavam satisfatórios.

Na Figura 46 pode observar-se que vários são os pontos que são eliminados devido ao filtro implementado, porém existem sempre casos em que não o são. A não eliminação de alguns pontos prende-se com o facto de as linhas de pesquisa não intercetarem o feixe de luz ou a caixa na sua plenitude, fazendo com que por vezes estes possuam o tamanho correto de uma linha àquela distância.



a) *b)*

c) *d)*

Figura 46 – a) Imagem obtida com pontos de linha errados devido ao feixe de luz; b) Imagem obtida com pontos de linha errados devido à presença da caixa branca no campo; c) Aplicação do filtro sobre a imagem com o feixe de luz; d) Aplicação do filtro sobre a imagem com a caixa no campo

Desta forma, é possível concluir que o método implementado funciona parcialmente devido à perda de alguns pontos que são realmente pontos de linhas e por não ser possível a eliminação total dos pontos errados. A perda de pontos de linha corretos poderia apresentar-se como sendo um problema, mas como são utilizadas linhas radiais e axiais para deteção das linhas de campo, em conjunto obtêm-se sempre um número de pontos que não prejudica de todo a localização do robô no campo.

Decidiu-se utilizar o método, mas alterando os seus parâmetros, ou seja, o filtro continuará a ser utilizado, mas com uma maior abrangência possibilitando desta forma descartar objetos que apresentem um maior tamanho, comparativamente aos utilizados para testes, e também descartar qualquer objeto de cor branca que se apresente demasiado próximo do robô.

4.3. POSIÇÃO DO ROBÔ NO CAMPO

Neste subcapítulo serão testados os métodos implementados para o cálculo da orientação, do mapeamento dos pontos de linhas e da obtenção da posição do robô no campo. Decidiu-se apresentar os resultados de todos os métodos em conjunto pois, durante a obtenção da posição no campo, caso se possua uma orientação errada ou um mau mapeamento, quer na proximidade quer longe do robô, estes irão impossibilitar o cálculo da posição de forma correta.

Durante a realização de alguns testes foi notada uma particularidade do algoritmo de obtenção da posição do robô. O algoritmo utiliza a função de erro, Equação 6, de forma a perceber, de acordo com as transições detetadas, qual a posição do robô que melhor se ajusta ao modelo do campo.

Porém, em casos que fossem detetados pontos perto e pontos longe, por exemplo pontos das linhas da área e pontos das linhas do meio campo era notado um pequeno deslocamento do robô da posição em que se devia encontrar pois, o algoritmo tentava posicionar as transições o melhor possível, de acordo com o modelo do campo, de forma a possuir o menor erro possível. Este pequeno deslocamento prende-se por a diferença em pixéis para distâncias superiores a $3m$ ser demasiado ténue, tornando difícil a calibração do mapeamento para distâncias superiores a esta.

Portanto, como o mapeamento para pequenas distâncias apresenta uma maior fiabilidade, desenvolveu-se um método simples que possibilita contornar este pequeno problema. O método desenvolvido faz uma atribuição de pesos consoante a distância a que se encontra o ponto de linha detetado, sendo que quanto mais longe este se encontra do robô menor peso terá. Desta forma é possível contornar a dificuldade de obter um bom mapeamento a longas distâncias.

De forma a testar todas as componentes, foram realizados diversos testes, sendo que estes decorreram em dois campos distintos, o campo que a MINHO TEAM possui no seu laboratório, um campo bastante mais pequeno que o oficial, e o campo da equipa CAMBADA em Aveiro, campo com o tamanho oficial dos jogos da liga MSL ($18\ m$ de comprimento e $12\ m$ de largura).

Pretendia-se com o teste em diferentes campos obter algumas conclusões acerca da fiabilidade e robustez dos métodos implementados, sendo que para tal foram escolhidas algumas posições do campo. Para distinção entre posições no campo do laboratório da MINHO TEAM e do campo de Aveiro, decidiu-se adicionar à letra da posição um M ou

A de forma a distinguir, por exemplo A_M é posição A no campo da MINHO TEAM e A_A é a posição A no campo de Aveiro, as posições escolhidas podem ser observadas na Figura 47.

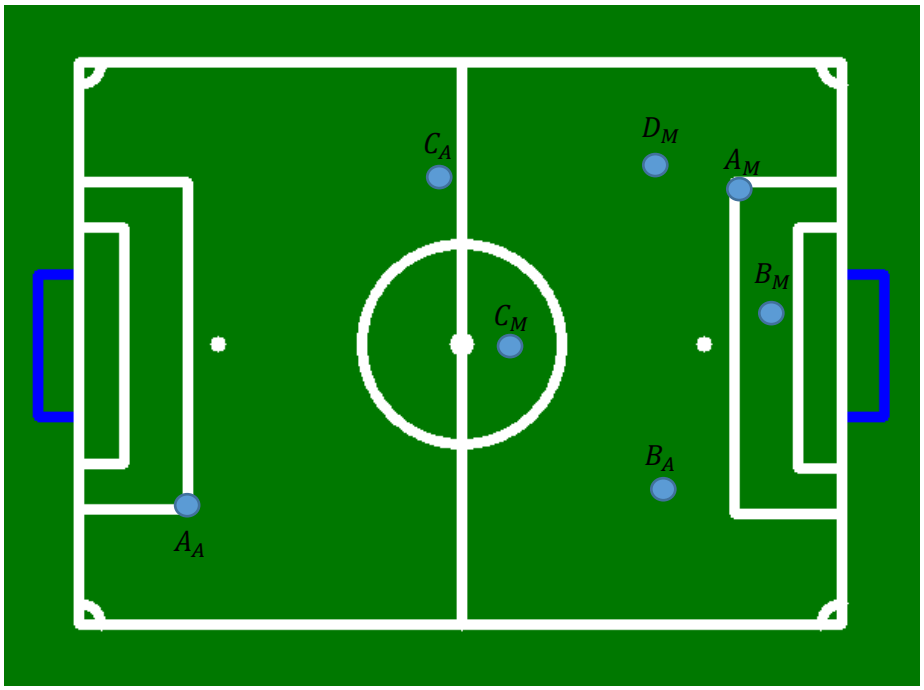


Figura 47 - Posições escolhidas para testes

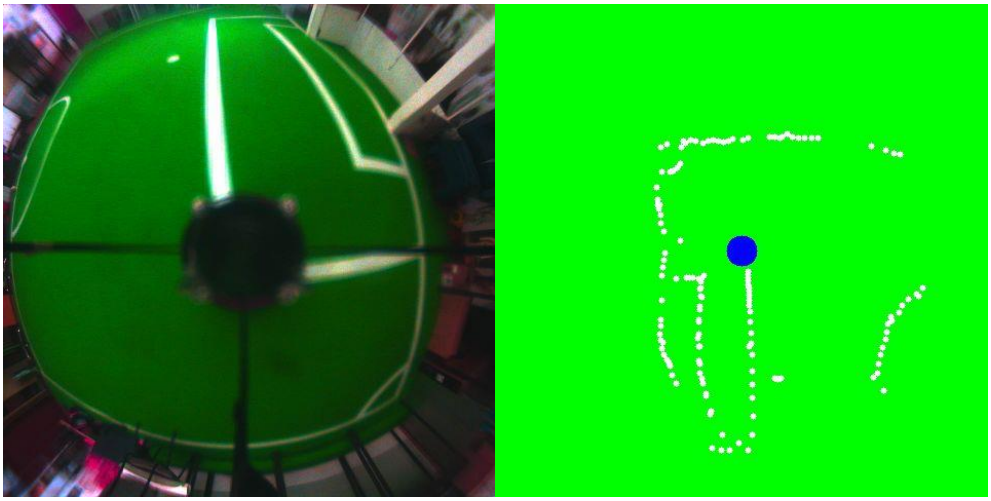
As figuras que se seguem (Figura 48 a Figura 51) apresentam os resultados obtidos de todo o algoritmo de localização, ou seja, orientação, mapeamento e posição do robô no campo. São apresentadas em todos os testes três imagens, a imagem obtida, uma imagem com o mapeamento dos pontos em função do robô e por fim a localização do robô no campo.

A Figura 48 apresenta o resultado obtido na posição A_M, é possível observar que o algoritmo foi capaz de achar a orientação, mapeou os pontos de linha e calculou a posição correta do robô no campo. O ângulo obtido por cada um dos métodos, visão, IMU e odometria, pode ser visualizado na Tabela 10.

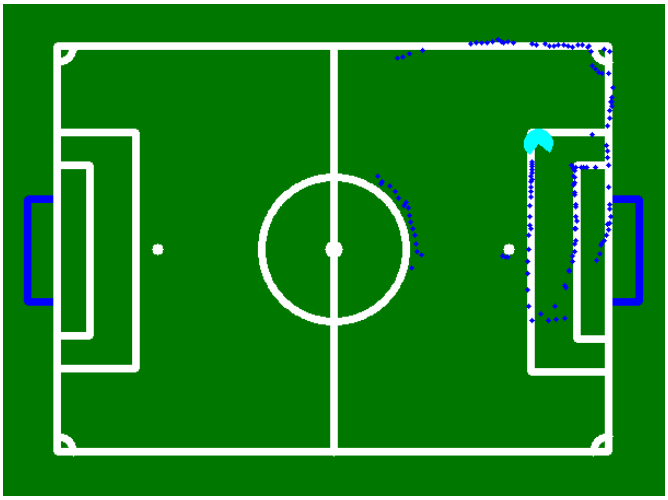
Tabela 10 - Valores de orientação de cada método na posição A_M

$\boldsymbol{\theta_{Visão}}$	$\boldsymbol{\theta_{IMU}}$	$\boldsymbol{\theta_{Odometria}}$	$\boldsymbol{\theta_{Final}}$
352	344	354	349

Os valores de orientação obtidos por cada um dos métodos, encontram-se próximos entre si e próximos do valor final, traduzindo-se num grau de fiabilidade bastante bom por parte de todos os métodos.



a) *b)*



c)

Figura 48 – a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição A_M

É possível de se perceber que o algoritmo funcionou de forma correta pois, observando a Figura 48, ao invés de tentar sobrepor os pontos de linha detetados perto do meio campo ou até mesmo tentar colocar os pontos salientados a vermelho mais perto da linha a que correspondem, este sobrepôs os pontos que se encontram junto dele, do robô, obtendo assim a posição correta do mesmo e demonstrando que a alteração imposta, pesos consoante a distância do ponto, oferece o resultado esperado.

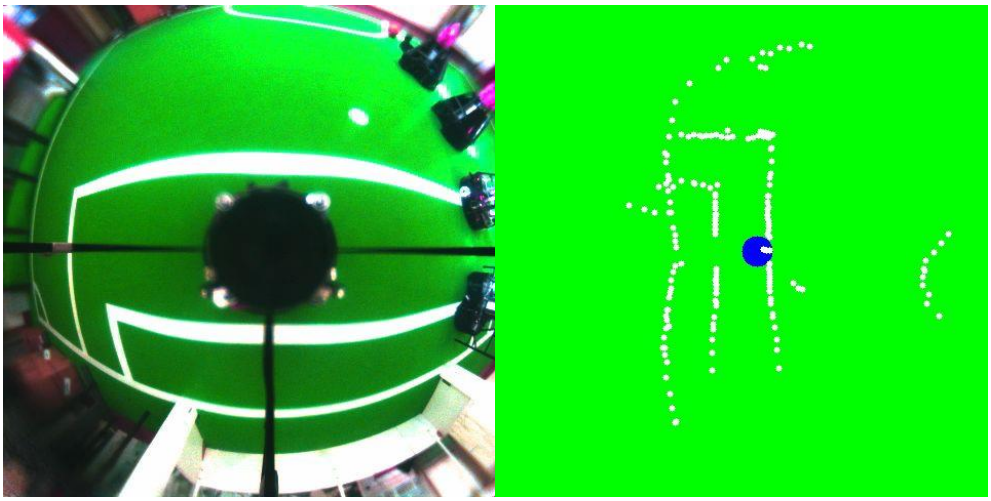
Na Figura 49 pode observar-se o resultado obtida na posição B_M, o algoritmo foi novamente capaz de calcular corretamente a posição do robô bem como a sua orientação e mapear os pontos de linha detetados.

Os valores dos ângulos obtidos pelos diferentes métodos, Tabela 11, mais uma vez, apresentam-se pouco dispares entre si, fazendo com que o valor final seja um valor de grande fiabilidade.

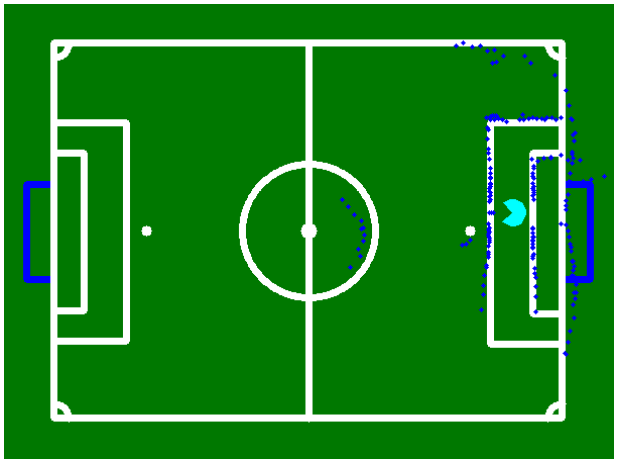
Tabela 11 - Valores de orientação de cada método na posição B_M

$\boldsymbol{\theta_{Visão}}$	$\boldsymbol{\theta_{IMU}}$	$\boldsymbol{\theta_{Odometria}}$	$\boldsymbol{\theta_{Final}}$
89	90	97	91

Pode observar-se novamente que alguns pontos não se encontram sobrepostos a nenhuma linha de campo, devendo-se esta particularidade à dificuldade, já referida, em mapear pontos com distâncias superiores a $3m$.



a) *b)*



c)

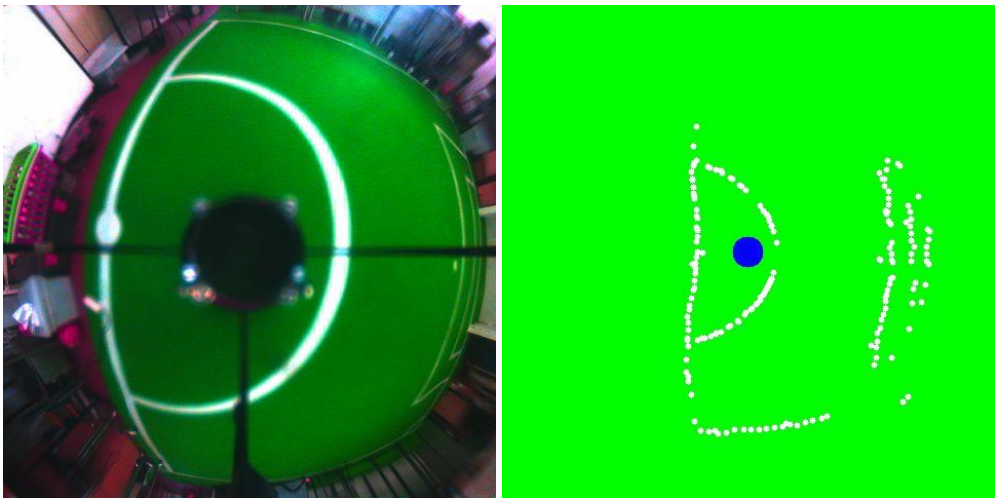
Figura 49 – a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição B_M

Para o teste na posição C_M, decidiu-se alterar a calibração do mapeamento do robô, fazendo com que a distâncias superiores a $1{,}5m$ o mapeamento dos pontos de linha esteja incorreto, de forma a testar se a atribuição dos pesos é suficiente quando o robô não se encontra devidamente calibrado. Na Figura 50 pode observar o resultado do teste realizado.

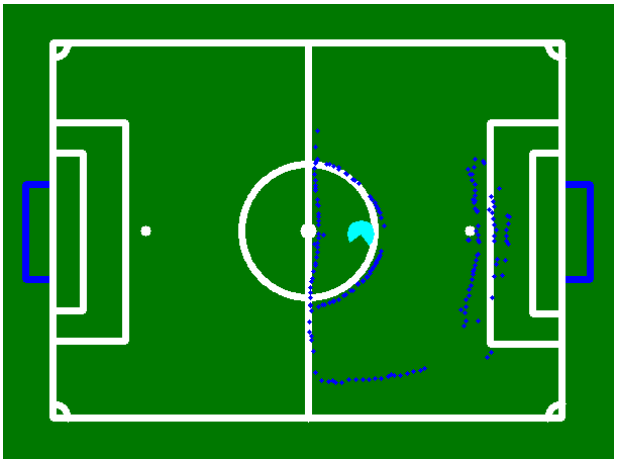
É percetível, com grande clareza, que a atribuição de pesos, bem como o algoritmo de localização do robô apresentam resultados deveras satisfatórios pois, mesmo uma má calibração do mapeamento, mas ponderada, o algoritmo contínua a encontrar a posição correta do robô no campo. A obtenção da orientação do robô continua também correta, uma vez que o resultado final se encontra muito perto da realidade, Tabela 12.

Tabela 12 - Valores de orientação de cada método na posição C_M

$\theta_{Visão}$	θ_{IMU}	$\theta_{Odometria}$	θ_{Final}
5	13	17	12



a) *b)*



c)

Figura 50 - a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição C_M

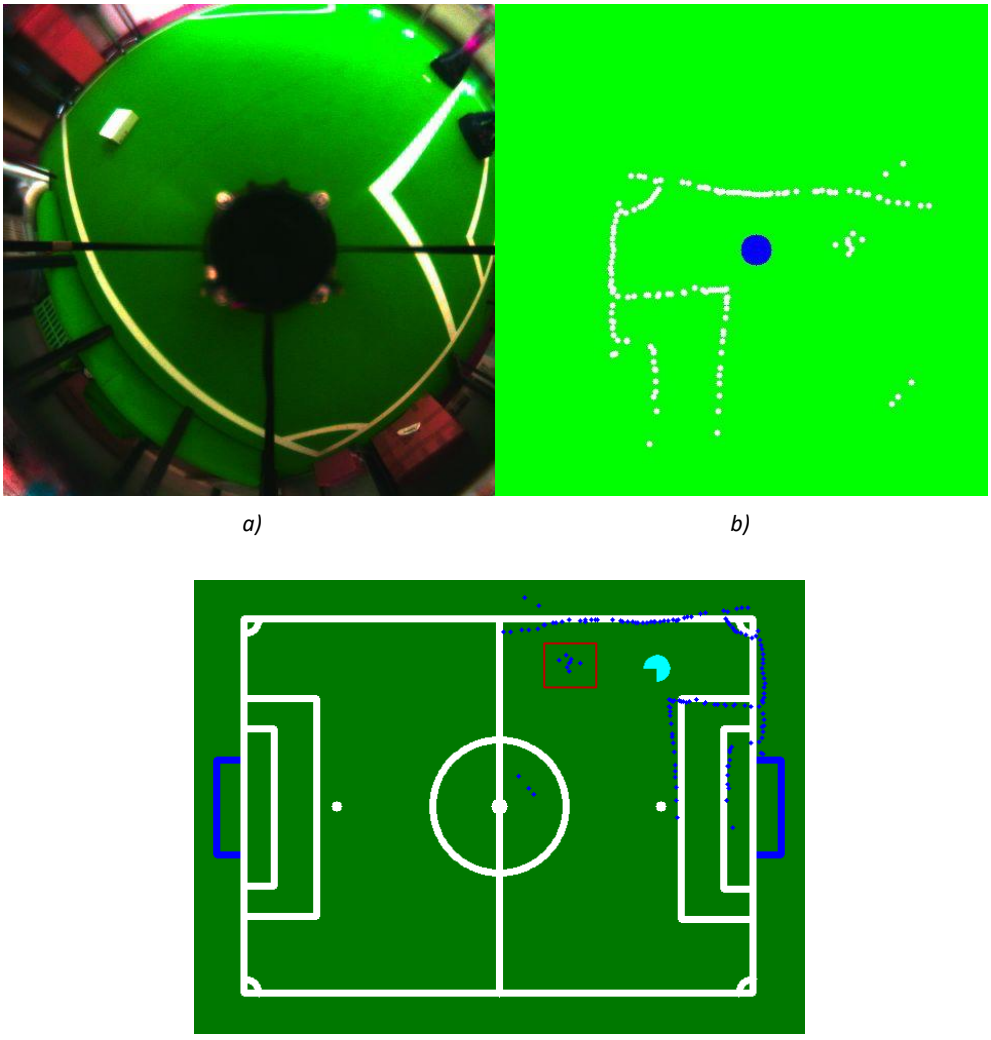
Por forma a testar todas as situações, foi realizado um teste no caso em que existe um feixe de luz ou um objeto não pertencente a um jogo de futebol dentro de campo, de forma a testar a robustez do algoritmo de localização. Este teste foi realizado na posição D_M, sendo que o resultado obtido é apresentado na Figura 51.

Como se pode observar, o algoritmo localizou o robô na posição correta do campo, mesmo quando se possuí interferências dentro do espaço de jogo que sejam interpretadas como linhas de campo, pontos destacados a vermelho. O cálculo da orientação do robô

também não foi afetado, uma vez que os valores obtidos se encontram perto entre si e perto do valor final como se pode observar na Tabela 13.

Tabela 13 - Valores de orientação de cada método na posição D_M

$\boldsymbol{\theta_{Visão}}$	$\boldsymbol{\theta_{IMU}}$	$\boldsymbol{\theta_{Odometria}}$	$\boldsymbol{\theta_{Final}}$
45	40	49	44



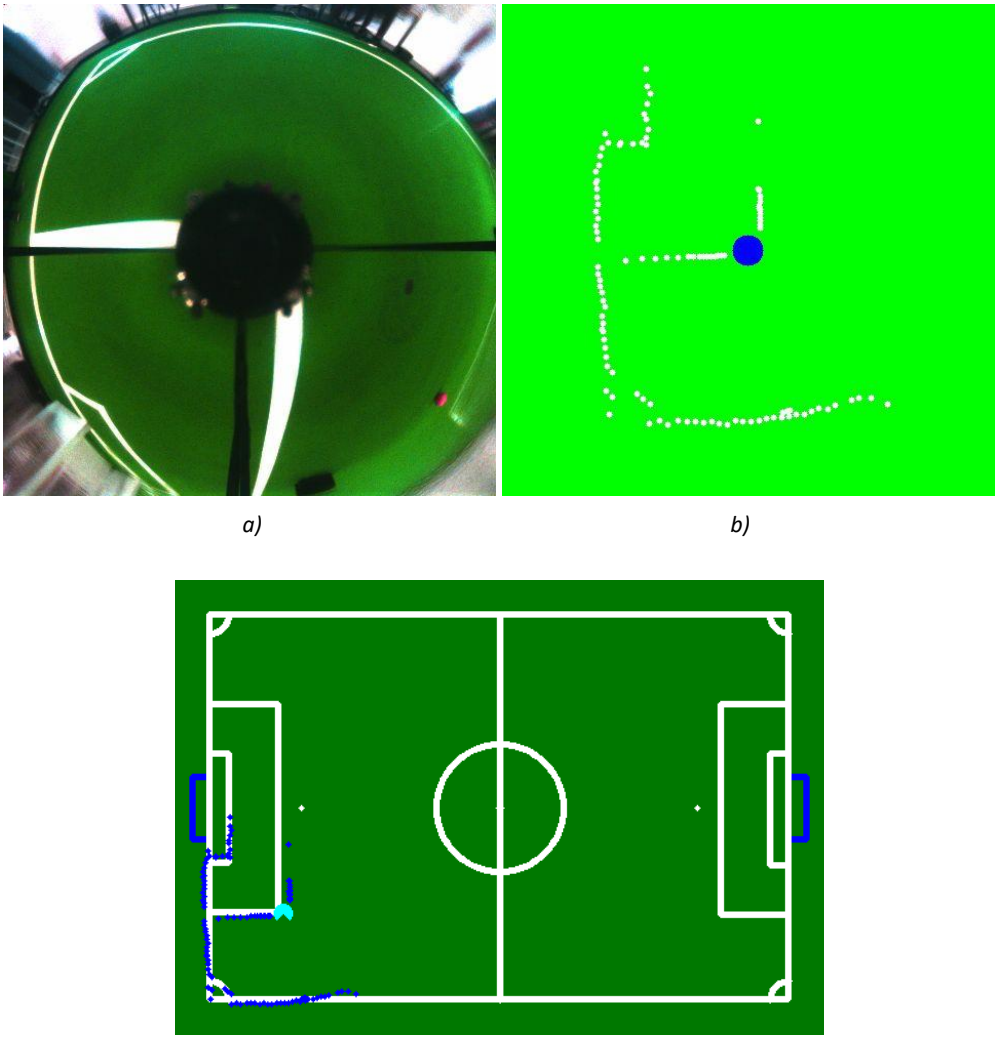
a) *b)*

c)

Figura 51 - a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição D_M com ruído propositado no campo

Como se pode comprovar com os testes realizados, o algoritmo de localização apresenta os resultados esperados, sempre com uma boa orientação e com precisão em relação à posição do robô no campo, podendo-se concluir que o objetivo, localização do robô, no campo da MINHO TEAM está de longe ultrapassado.

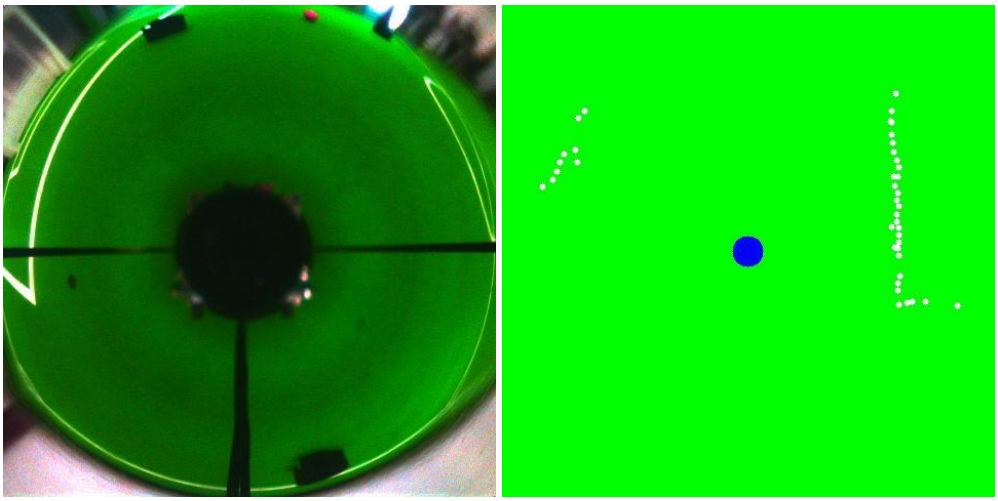
Após realizados os testes no campo da MINHO TEAM seguem-se então os testes realizados num campo com o tamanho oficial, o campo da equipa CAMBADA em Aveiro. Os testes no campo oficial visam testar a robustez e precisão do algoritmo desenvolvido para futuras competições que a equipa venha a participar e também de forma a poder comprovar que o algoritmo desenvolvido funciona em campos distintos. Estes podem ser visualizados nas figuras seguintes (Figura 52 a Figura 54).



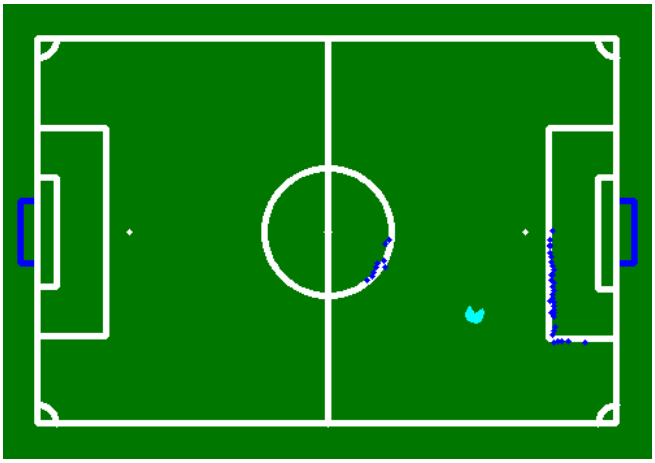
a) *b)*

c)

Figura 52 - a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição A_A

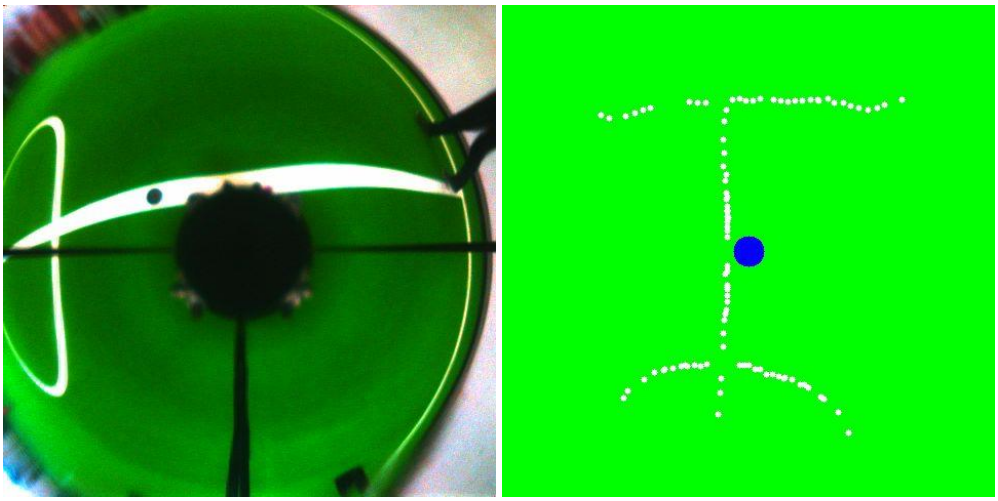


a) *b)*

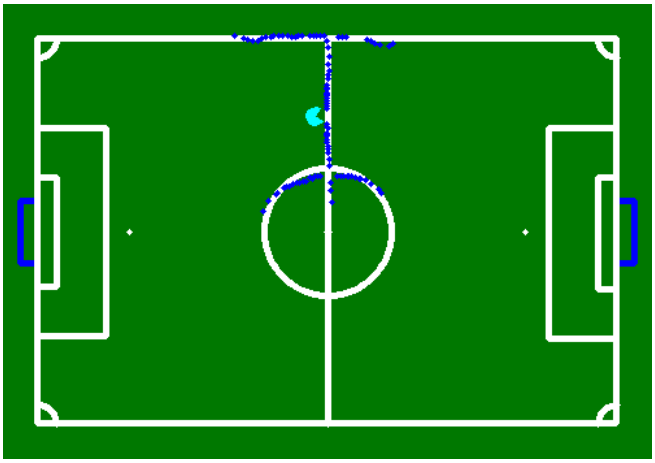


c)

Figura 53 - a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição B_A



a) *b)*



c)

Figura 54 - a) Imagem obtida; b) Mapeamento dos pontos de linha em relação ao robô; c) Teste do algoritmo de localização do robô na posição C_A

Observando as figuras acima (Figura 52 a Figura 54), pode concluir-se que o algoritmo desenvolvido funciona como esperado, apresentando robustez e precisão no que diz respeito à orientação e posição do robô no campo. Porém através da análise da Tabela 14 é notório a discrepância do valor do ângulo obtido pela visão na posição B_A.

Tabela 14 - Valores de orientação de cada método nas posições A_A, B_A, C_A

$\boldsymbol{\theta_{Visão}}$	$\boldsymbol{\theta_{IMU}}$	$\boldsymbol{\theta_{Odometria}}$	$\boldsymbol{\theta_{Final}}$
357	4	9	3
155	181	190	177
258	262	267	263

Esta diferença deve-se ao facto de, naquela posição do campo, B_A, devido ao raio de visão limitado que o robô possui, são detetados poucos pontos de linha fazendo com que a precisão e fiabilidade do ângulo obtido pela visão desça, mas como se realiza a fusão, esta variação não é praticamente sentida no valor final da orientação.

Como já fora referido, o algoritmo da localização no seu geral apenas pode demorar cerca de 33 ms, de forma a analisar todos as imagens capturadas pela câmara.

Portanto, durante a realização dos testes, foram tirados diversos valores de tempo de forma a obter uma média mais precisa do tempo de processamento necessário. Foram realizadas leituras tanto nos testes realizados no campo da MINHO TEAM como no campo da equipa CAMBADA, campo oficial, sendo notória uma diferença de tempo de processamento quando se executa uma localização global.

Porém, este resultado já era esperado uma vez que um campo maior possui mais posições para analisar pois, o campo oficial é deveras maior que o campo no laboratório da equipa. Na localização local o tempo necessário apenas varia com o número de pontos de linha detetados e não com o número de posições a analisar pois, neste caso, localização local, é apenas analisado o quadrado pré-definido.

Na Tabela 15 pode observar-se os tempos de processamento necessários, tanto para localização global como local bem como no campo da equipa CAMBADA ou no campo da MINHO TEAM.

Tabela 15 - Tempo de processamento da localização

	Localização Global	**Localização Local**
Campo LAR (milissegundos)	110	16
Campo Oficial(milissegundos)	154	17

As leituras de tempo retiras são referentes ao processo de localização, ou seja, desde a segmentação da imagem até ao envio da posição do robô e das coordenadas dos objetos detetados à volta do mesmo, modelação do mundo que o rodeia.

Através da análise da Tabela 15 é possível verificar que os tempos de processamento requeridos são baixos, verificando-se que o intervalo de tempo estabelecido, 33 ms, para realização da localização local é devidamente respeitado. Nos tempos da localização local existe diferença pois, as imagens obtidas, do campo da MINHO TEAM para o campo oficial, são diferentes, podendo fazer variar o número de

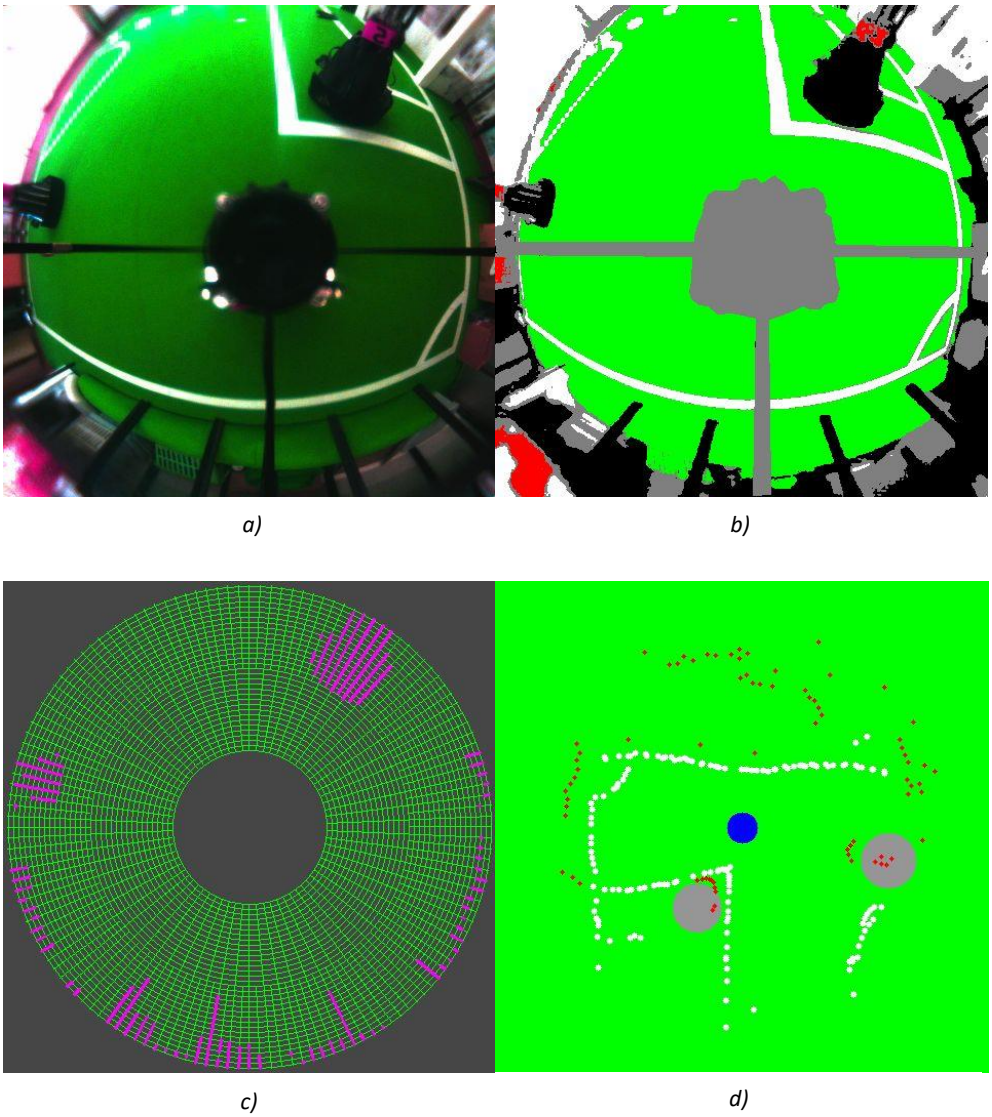
pontos de transições detetadas bem como outros parâmetros, fazendo com que exista uma pequena variação dos tempos.

É de salientar que para realização de testes e obtenção dos tempos de processamento foi utilizada a *board* que se encontra no robô, sendo esta uma MSI Cubi com um processador Intel Pentium 3805u.

4.4. WORLD MODEL

De forma a testar a modelação do mundo que rodeia o robô criaram-se diversos cenários. Começou-se por posicionar dois obstáculos à distância de 1 m e 2 m do robô.

Na Figura 55 é possível observar a imagem obtida, a imagem segmentada, a imagem com a informação obtida pelas *scanlines* e por fim o mapeamento do obstáculo em função do robô e no campo.

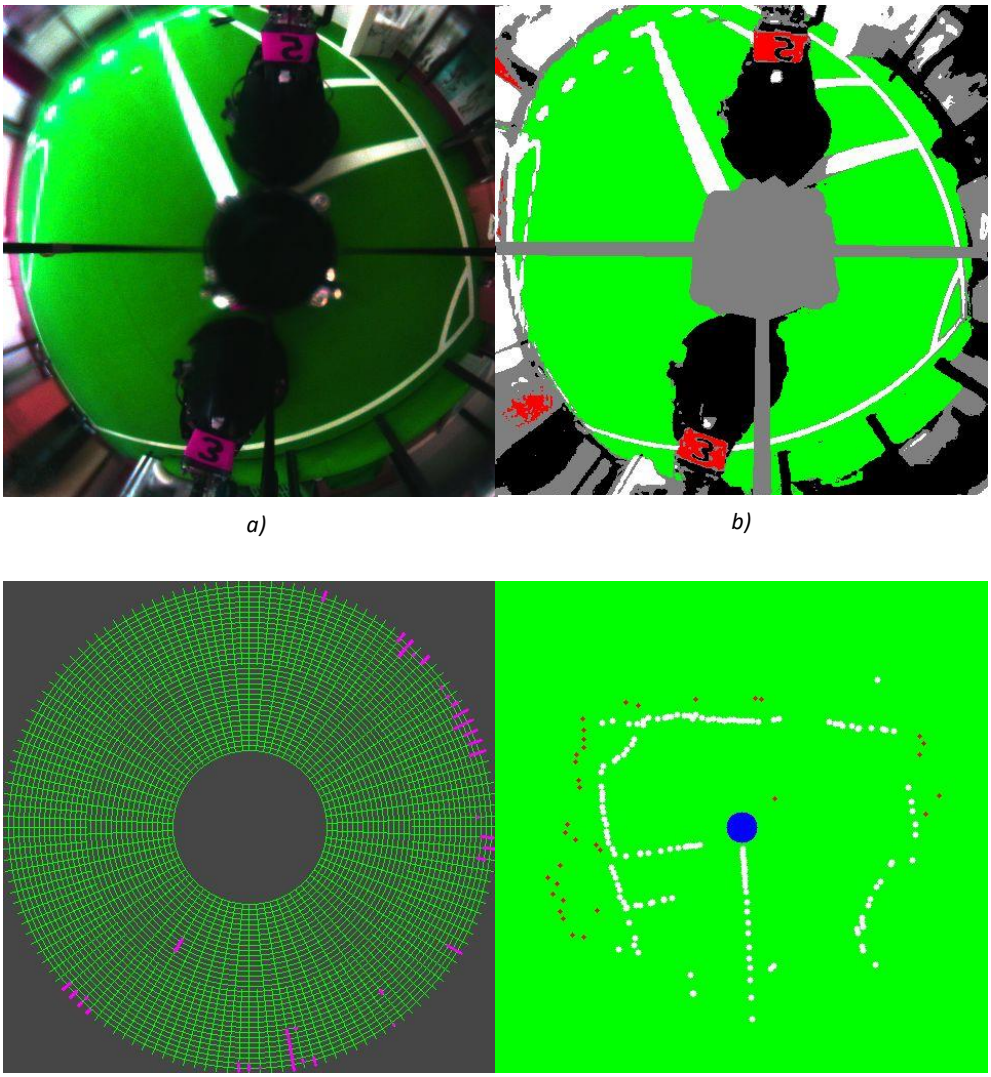


a) *b)*

c) *d)*

Figura 55 – a) Imagem obtida; b) Imagem segmentada; c) Imagem com a informação contida nas scanlines; d) Mapeamento dos obstáculos a $1m$ e $2m$ em relação ao robô

Durante a execução de diversos testes foi detetada uma limitação do programa até então desenvolvido no que diz respeito à criação do *World Model*. Caso os obstáculos se encontrassem demasiado perto do robô estes não eram detetados, como se pode observar na Figura 56.



a) *b)*

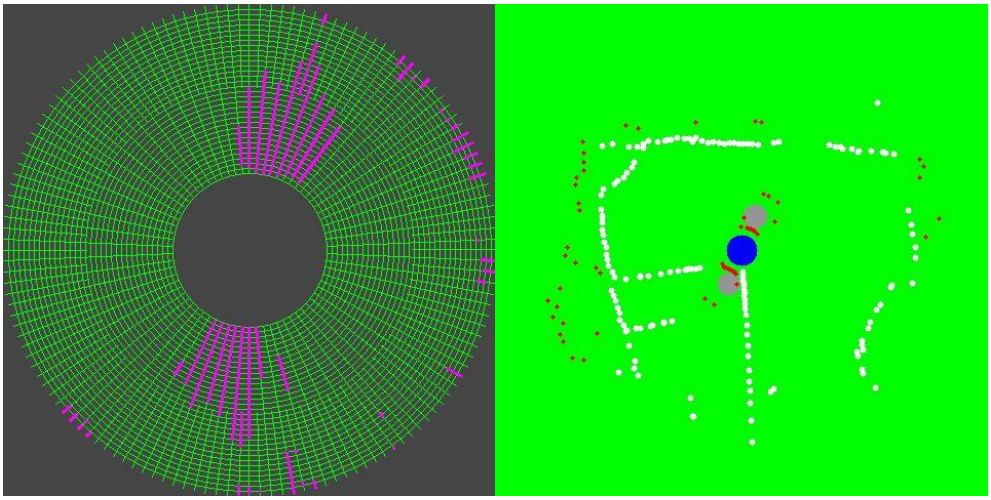
c) *d)*

Figura 56 - a) Imagem obtida; b) Imagem segmentada; c) Imagem com informações das scanlines comprovando a falha na deteção dos obstáculos ao perto; d) Mapeamento em relação ao robô; e) Mapeamento dos obstáculos no campo de jogo

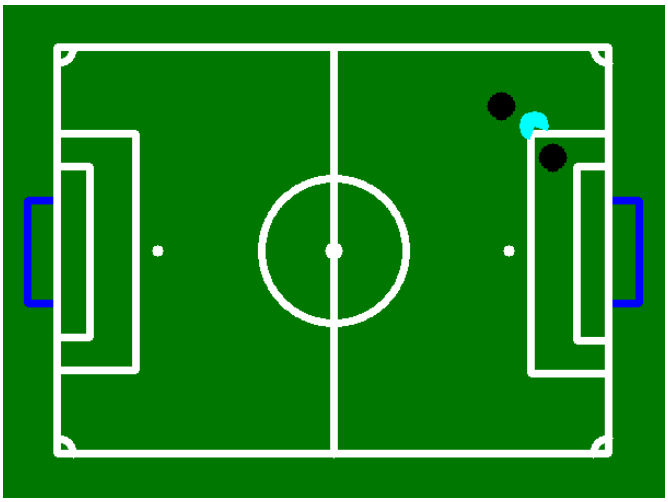
Como pode ser visualizado na Figura 56 os obstáculos, outros robôs, não são detetados pois, não existe qualquer espaço verde, campo, entre o robô e os obstáculos ou este espaço é demasiado reduzido. Desta forma, decidiu-se criar um novo tipo de pesquisa, sendo que este apenas irá procurar pela cor preta, mas para ser validado deverá conter um mínimo elevado de pontos devido à proximidade que o obstáculo pode apresentar.

Após esta nova implementação, como se pode observar na figura que se segue, Figura 57, a limitação foi ultrapassada, sendo que todos os obstáculos que se encontram no raio de visão do robô são detetados com sucesso.

a) *b)*



c) *d)*

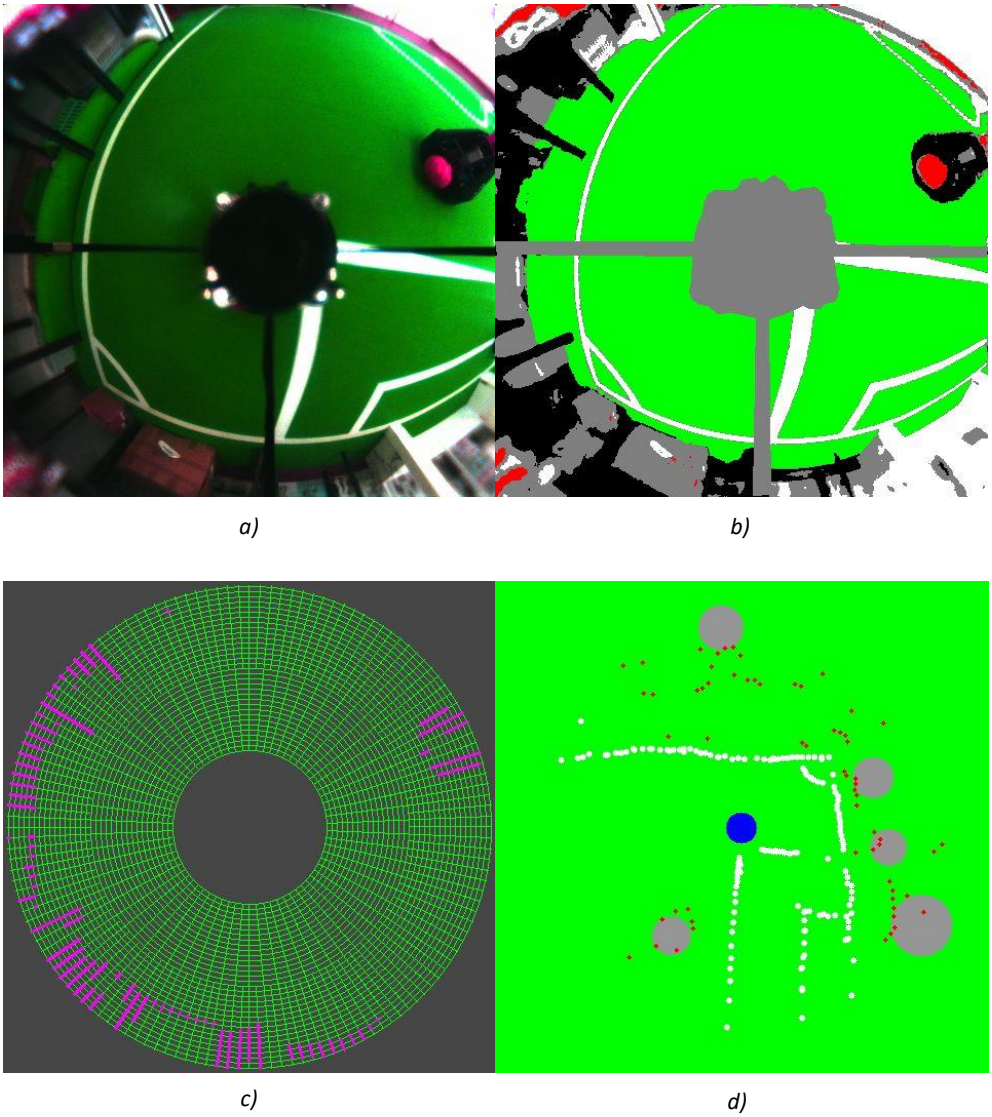


e)

Figura 57 – a) Imagem obtida; b) Imagem segmentada; c) Imagem com informações das scanlines; d) Mapeamento em relação ao robô; e) Mapeamento dos obstáculos no campo de jogo

No entanto, após a correção da limitação acima mencionada, foi detetada uma nova limitação no decorrer da realização de alguns testes. Esta nova limitação ocorre na pesquisa da bola, como fora dito a bola é procurada pelas transições de cores verde-laranja ou laranja-verde.

Porém, caso a bola se encontre na posse de um outro robô, esta não é detetada pois, não é encontrada a transição que se procura, laranja-verde, sendo encontrada a transição laranja-preto, porém o robô que possui a bola é detetado. Um exemplo deste cenário, bola não detetada quando na posse de um robô, pode ser observado na Figura 58.



a) *b)*

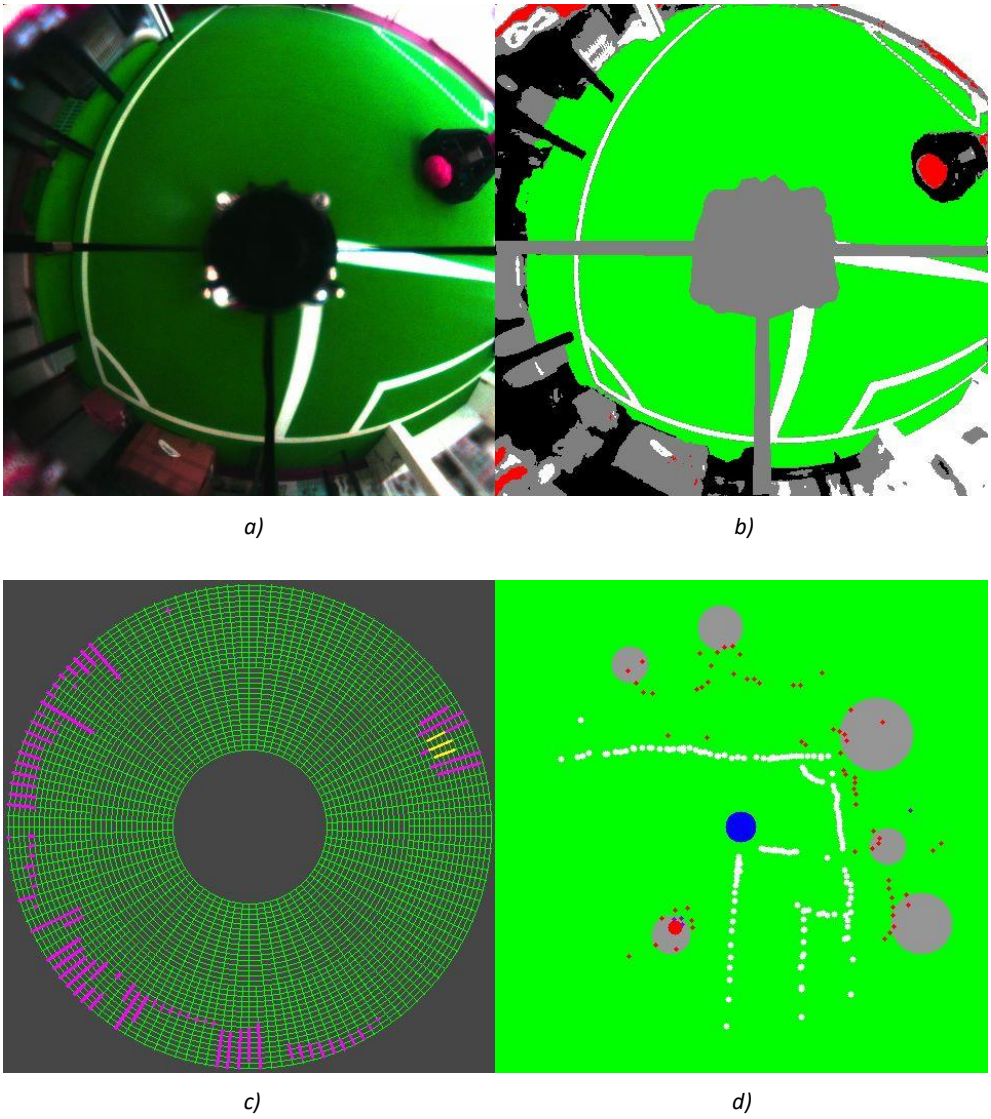
c) *d)*

Figura 58 – a) Imagem obtida; b) Imagem segmentada; c) Imagem com a informação contida nas scanlines; d) Mapeamento em relação ao robô

Portanto, tal como aconteceu na deteção de obstáculos, foi implementada uma nova pesquisa. Esta ao invés de procurar pela transição laranja-preto, procura apenas pela cor

laranja, mas para uma transição ser válida esta deve conter um número mínimo de pontos de cor laranja, de forma a validar que realmente se trata de um ponto de bola.

Na Figura 59 é possível observar que esta nova pesquisa implementada apresenta os resultados pretendidos, deteção da bola quando na posse de outro robô, pois, observando mais pormenorizadamente a figura é possível perceber que dentro do robô encontra-se representada a bola detetada, círculo vermelho perto de pontos azuis.



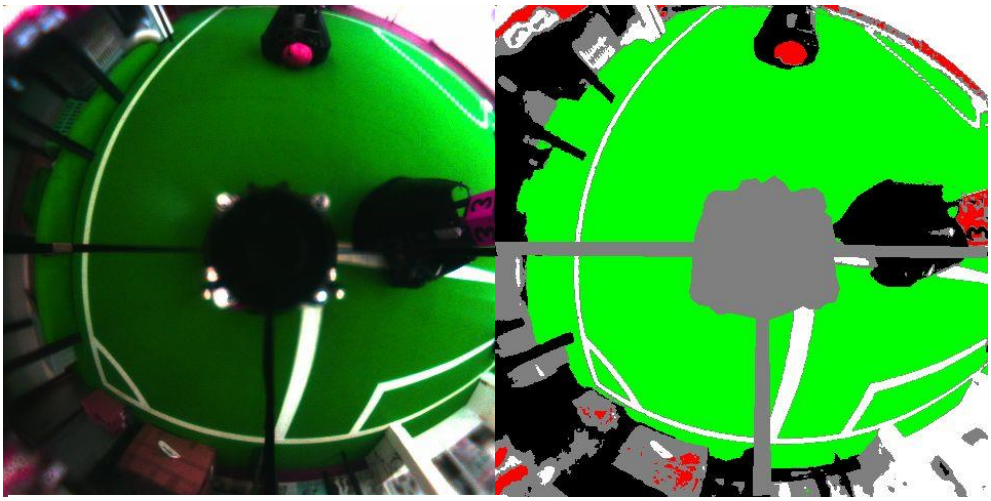
a) *b)*

c) *d)*

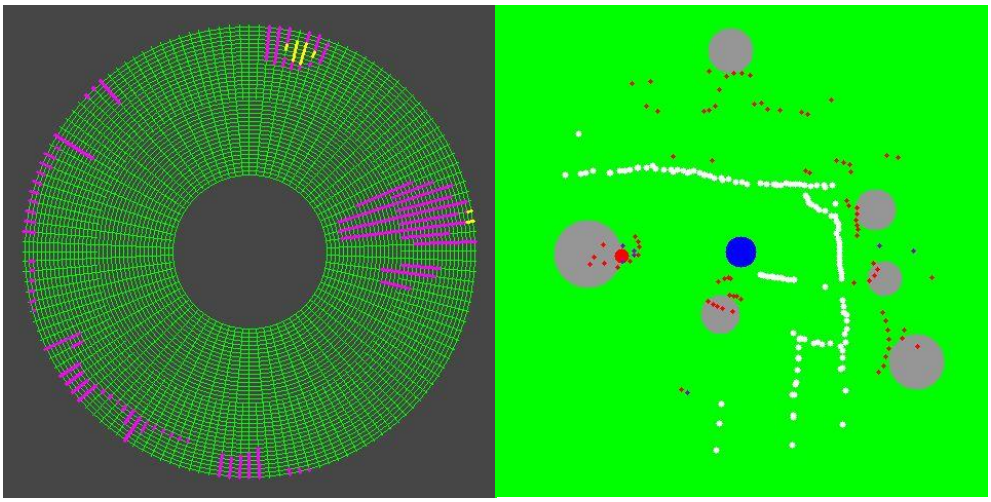
Figura 59 – a) Imagem obtida; b) Imagem segmentada; c) Imagem com informação contida nas scanlines; d) Mapeamento em relação ao robô

Por fim, após a resolução dos problemas acima referidos, foram realizados diversos testes, tendo os resultados obtidos sido deveras satisfatórios tal como se pode visualizar na imagem que se segue, Figura 60, onde foi criado um cenário para teste de várias possibilidades.

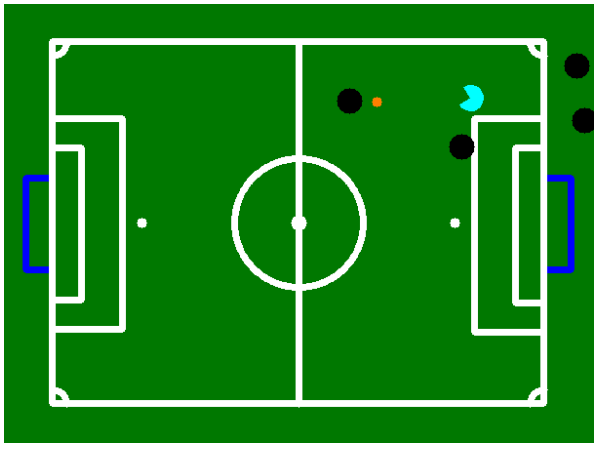
Neste cenário foram colocados dois robôs na imagem, um perto do robô a partir do qual se captura as imagens e um outro robô com a bola em sua posse de forma a testar então as novas implementações. Foram criados muitos outros cenários, como a bola em cima da linha de jogo, dois robôs perto um do outro, entre muitos outros e como esperado as mudanças realizadas apresentaram os resultados esperados, uma modelação do mundo que rodeia o robô muito próxima da realidade.



a) *b)*



c) *d)*



e)

Figura 60 - a) Imagem obtida; b) Imagem Segmentada; c) Imagem com informação das scanlines; d) Imagem do mapeamento em relação ao robô; e) Mapeamento dos obstáculos e da bola no campo de jogo

4.5. *VISION CALIB*

A ferramenta *Vision Calib* foi desenvolvida anteriormente ao desenvolvimento de todo o processo de localização pois, com isto pretendia-se poder testar a ferramenta criada logo desde o início.

Durante o desenvolvimento do processo de localização bem como da realização de testes, o *Vision Calib* demonstrou ser uma ferramenta de enorme valor, possibilitando alterar praticamente tudo, no que diz respeito à localização no seu geral, obter a imagem do que o robô estava a ver, da informação contida nas linhas de pesquisa entre muitas outras possibilidades já citadas.

O software desenvolvido apresentou ser sempre de elevada robustez e com uma grande fiabilidade de comunicação, sendo que a sua comunicação falha apenas se a rede onde o robô está inserido estiver com um elevado tráfego de dados.

Pode-se então concluir que a criação desta ferramenta foi uma mais valia para toda a equipa e é mais um impulsionador para o desenvolvimento continuo, sendo o que se pretende para o futuro da MINHO TEAM.

4.6. ROBÓTICA 2017

De 26 a 30 de abril a equipa MINHO TEAM participou uma vez mais numa competição, Robótica 2017 na liga MSL [21]. Esta competição ocorre anualmente, em cidades diferentes de ano para ano, sendo que este ano ocorreu em Coimbra e é uma competição reconhecida a nível internacional. No que diz respeito à liga MSL as principais equipas que competem no RoboCup participam com grande regularidade no Robótica.

Figura 61 - Símbolo do Robótica 2017

Aquando do início da competição toda a equipa apresentava algumas expectativas e objetivos, sendo que o maior objetivo de toda a equipa era conseguir realizar um jogo

de principio a fim com alguma interação dos robôs entre si, como por exemplo sermos capazes de efetuar um *kick-off* ou um *free-kick*.

Durante os dias da competição a equipa trabalhou afincadamente de forma a superar os objetivos traçados e colmatar alguns problemas conhecidos. Após a correção dos problemas conhecidos a equipa decidiu participar então no seu primeiro jogo de forma a testar todo o trabalho desenvolvido até então.

Porém, novos problemas foram encontrados, como, logo no inicio do jogo os robôs têm de se conectar à *RefBox, Referee Box*, pois, é esta que envia os comandos acerca dos cantos, faltas, etc, mas estes não conseguiram conectar-se, outros não permitiam a compilação do código, entre outros problemas.

No terceiro dia da competição, dia 28 de abril, após muitas horas de trabalho contínuo e muitos problemas e limitações resolvidas a equipa decidiu novamente jogar. Foi então um momento de grande alegria e satisfação para toda a equipa pois, fomos capazes de realizar um jogo completo, executamos *kick-off* e *free-kick* e acima de tudo a equipa MINHO TEAM esteve na eminência de marcar um golo, mas por questões mecânicas tal não foi possível.

A participação nesta competição, Robótica 2017 demonstrou ser deveras importante pois, para além de ter sido uma possibilidade para teste de todas as ferramentas criadas demonstrarem a sua utilidade, e foram sem dúvida uma mais valia devido à sua necessidade quase constante, mas por permitir perceber que a equipa se encontra no caminho certo, onde com mais tempo e trabalho alcançará o nível de competitividade das melhores equipas da liga MSL.

Em suma pode dizer-se que os objetivos para esta competição foram amplamente ultrapassados, todas as ferramentas desenvolvidas demonstraram ser cruciais e acima de tudo é notável uma enorme evolução na equipa. É de salientar que após a competição toda a equipa se encontra muito mais motivada e focalizada no melhoramento da performance da equipa.

5. CONCLUSÃO E TRABALHO FUTURO

5.1. CONCLUSÃO

Com a realização desta dissertação, criaram-se ferramentas capazes de calibrar e configurar os robôs, foram corrigidas deformações nas imagens obtidas, desenvolveu-se um algoritmo de auto calibração das cores da câmara. O objetivo principal foi alcançado com o desenvolvimento de um processo de localização para os robôs da MINHO TEAM.

Foi analisada a qualidade da imagem obtida em termos de deformações e investigado a forma de as corrigir, tendo-se chegado à conclusão que seria necessário desenvolver um novo suporte para a câmara que possibilitasse uma alteração do posicionamento da mesma com precisão. Desenvolveu-se, ainda, um algoritmo de auto calibração, o qual obteve um resultado deveras satisfatório, sendo possível a sua utilização aquando do início de um jogo, mas também durante o mesmo, garantindo que a qualidade da imagem não é afetada.

O processo de localização desenvolvido é bastante rápido, necessitando apenas de 154 milissegundos para realizar a localização global e de 17 milissegundos para a localização local, tendo estes tempos sido retirados com o processo de localização a funcionar totalmente no campo oficial, ou seja, desde obtenção da imagem até à modelação do mundo e comunicação de todas as informações, e não apenas a obtenção da posição do robô no campo,

Criou-se o *Vision Calib*, software capaz de calibrar praticamente todos os aspetos referentes ao processo de localização, desde a alteração dos parâmetros da câmara, se necessário, bem como de parâmetros da auto calibração, até à alteração de valores de variáveis referentes à modelação do mundo que rodeia o robô.

Por fim, pode afirmar-se que todos os objetivos propostos nesta dissertação foram alcançados com sucesso e que a MINHO TEAM está agora dotada de ferramentas e processos robustos, que lhe permitirá um desenvolvimento contínuo e sustentado, com vista a atingir o principal objetivo da equipa que é voltar a ser capaz de competir com qualquer equipa ao mais alto nível.

5.2. TRABALHO FUTURO

No decorrer da realização desta dissertação surgiram diversas ideias passíveis de ser caso de estudo no futuro. Algumas das soluções que são propostas necessitam de utilizar mais hardware, bem como a alteração de alguns componentes do robô.

A primeira alteração que deverá ser realizada é a mudança do espelho do robô e da câmara, responsáveis pela visão do mesmo. A razão desta sugestão prende-se com o facto de $480x480$ pixéis ser uma resolução de imagem bastante reduzida, tornando difícil a deteção de obstáculos a grandes distâncias. Como foi referido, quando se executa uma localização global, se esta for realizada no campo oficial, o robô tem de se encontrar perto das linhas de campo pois, caso esteja em zonas com poucas linhas devido ao raio de visão reduzido, $4m$, como já fora dito, não vão existir praticamente pontos de linha na imagem. Durante o Robótica 2017, observou-se as imagens obtidas pelas outras equipas da liga MSL, onde, em média, todas as equipas possuíam um raio de visão o dobro do apresentado pela MINHO TEAM, ou seja, um raio de visão a rondar os $8m$ e em alguns casos mais ainda.

Portanto, melhorar o espelho e a câmara deveria ser a primeira alteração a ser realizada.

Utilizar um acelerómetro seria uma alteração deveras importante, pois permitiria eliminar o erro introduzido pelo acumular dos *encoders*, bem como compensar o caso em que o robô derrapa devido à alta velocidade.

A utilização de uma câmara Kinect em cada robô e não só no guarda-redes seria igualmente vantajoso, uma vez que seria possível não só estimar a posição da bola no ar, como também ajudar na modelação do mundo que rodeia o robô.

De forma a melhorar todo o programa, julgo que a arquitetura do software deveria sofrer algumas alterações. A calibração e obtenção da imagem, e também a modelação do mundo deveriam ser processos completamente distintos da localização, pois possibilita a criação de algoritmos mais complexos e rigorosos.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] “RoboCup Federation Official Site” The Robocup Federation. [Online]. Disponível em: http://www.robocup.org/.

[2] “RoboCup Federation Official Site - Objective” The Robocup Federation. [Online]. Disponível em: http://www.robocup.org/objective.

[3] “Método de Monte Carlo.” [Online]. Disponível em: https://pt.wikipedia.org/wiki/Método_de_Monte_Carlo.

[4] Odakura, V. V. V. A., Costa, A. H. R., Chaimowicz, L., Cozman, F. G., Lizarralde, F. C., & Sichman, J. S, "Localização de Markov para multirrobôs cooperativos". Universidade de São Paulo, 2006

[5] Lauer M, Lange S, Riedmiller M. Calculating the perfect match: an efficient and accurate approach for robot self-localization. In: Bredenfeld A, Jacoff A, Noda I, Takahashi Y, editors. RoboCup 2005: robot soccer world cup IX. Lecture notes in computer science, vol. 4020. Springer; 2006. p. 142–53.

[6] “Tribots selflocalization.” [Online]. Disponível em: http://ml.informatik.uni-freiburg.de/research/tribots/selflocalization.

[7] Alina Trifan A. J. R. Neves and Bernardo Cunha, "Self-calibration of colormetric parameters in vision systems for autonomous soccer robots". In in Proc. of RoboCup, 2014 Symposium, 2014.

[8] Welch, G., and Bishop, G., “An introduction to the Kalman Filter,” Tech. Report TR 95- 041 (Update May 23, 2003), Department of Computer Science, Univ. of North Carolina at Chapel Hill, 2003.

[9] Kalman, R.E., “ A New Approach to Linear Filtering and Prediction Problems”, Trans. of the ASME - Journal of Basic Engineering, Vol. 82: pp. 35-45, 1960.

[10] “ROS.org | Powering the world’s robots.” [Online]. Disponível em: http://www.ros.org/.

[11] “OpenCV | OpenCV.” [Online]. Disponível em: http://opencv.org/.

[12] A. J. R. Neves, G. Corrente, and A. J. Pinho. An omnidirectional vision system for soccer robots. In Proc. of the EPIA 2007, volume 4874 of Lecture Notes in Artificial Inteligence, pages 499–507. Springer, 2007.

[13] G. Lopes and F. Ribeiro, “Catadioptric system optimisation for omnidirectional RoboCup MSL robots”, T. Röfer et al. (Eds.): RoboCup 2011, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7416 LNCS, pp. 318-328. Springer-Verlag Berlin Heidelberg 2012.

[14] A. Trifan, A. J. R. Neves, B. Cunha, and J. L. Azevedo, “UAVision: A modular time-constrained vision library for soccer robots,” in Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), vol. 8992, pp. 490–501, 2015.

[15] P. Trigueiros, F. Ribeiro, and L. Paulo Reis, “Robot Orientation with Histograms on MSL”, (2012) T. Röfer et al. (Eds.): RoboCup 2011, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7416 LNCS, pp. 507-514. Springer-Verlag Berlin Heidelberg 2012

[16] “Adafruit 10-DOF IMU Breakout - L3GD20H + LSM303 + BMP180 ID: 1604.” [Online]. Disponível em: https://www.adafruit.com/product/1604.

[17] “Odometria.” [Online]. Disponível em: https://pt.wikipedia.org/wiki/Odometria.

[18] Jüngel M., Risler M., "Self-localization Using Odometry and Horizontal Bearings to Landmarks". In: Visser U., Ribeiro F., Ohashi T., Dellaert F. (eds) RoboCup 2007: Robot Soccer World Cup XI. RoboCup 2007. Lecture Notes in Computer Science, vol 5001. Springer, Berlin, Heidelberg

[19] F. Ribeiro, I. Moutinho, P. Silva, C. Fraga, and N. Pereira, “Controlling Omni-directional Wheels of a MSL RoboCup Autonomous Mobile Robot” Sci. Meet. Port. Robot. Open, 2004.

[20] F. Ribeiro, I. Moutinho, P. Silva, C. Fraga, and N. Pereira, “Three Omni-Directional Wheels Control on a Mobile Robot” Sci. Meet. Port. Robot. Open, 2004.

[21] “Robótica 2017 - Bem-vindos.” [Online]. Available: http://robotica2017.isr.uc.pt/index.php/pt/.