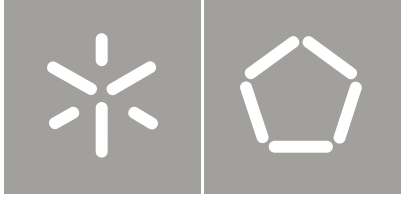
Universidade do Minho
Escola de Engenharia

Bruno Miguel da Silva Pereira

Localização de robô futebolista MINHO TEAM

Dezembro de 2011

Bruno Miguel da Silva Pereira Localização de robô futebolista MINHO TEAM
UMinho | 2011



Universidade do Minho
Escola de Engenharia

Bruno Miguel da Silva Pereira

Localização de robô futebolista MINHO TEAM

Tese de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de
Mestre em Engenharia Electrónica Industrial e Computadores

Trabalho efetuado sob a orientação do
Professor Doutor Fernando Ribeiro

Dezembro de 2011

"Põe quanto tu és no mínimo que fazes."

(Fernando Pessoa)

AGRADECIMENTOS

Gostaria de prestar os mais sinceros agradecimentos a algumas pessoas que de alguma maneira contribuíram para a realização deste trabalho.

Em primeiro lugar quero agradecer a todos os elementos do LAR, em especial ao Sérgio que desenvolveu no seu trabalho um *software* importante para a realização dos testes de Localização; ao João Costa pelo desenvolvimento da comunicação entre o *software* e o *hardware*. Agradeço igualmente ao João Paulo que desenvolveu o *software* para o movimento do robô permitindo assim testar de forma mais robusta a localização do robô; ao João Rodrigues pela ajuda dada na montagem dos robôs. Não poderia esquecer também o Doutor Gil Lopes incansável na ajuda prestada e o meu orientador, o Doutor Fernando Ribeiro que me dirigiu e ajudou para que esta dissertação pudesse ser concluída.

Quero ainda mostrar gratidão à minha família pelo apoio dado durante estes anos de estudo, bem como a todos os meus amigos, que de alguma forma, sempre me apoiaram e me deram confiança para que chegasse a bom porto.

Aproveito também para agradecer à equipa do ISEP pela disponibilidade demonstrada em ceder o seu campo de futebol robótico para a realização dos testes, que contribuíram para o desenvolvimento e melhoramento do trabalho desenvolvido.

Localização de robô futebolista MINHO TEAM

RESUMO

A localização de robôs futebolistas no campo é uma peça fulcral para o robô conseguir jogar de forma cooperante com a restante equipa, sendo assim necessário que a localização dos robôs seja confiável e robusta.

Este trabalho de investigação tem como objectivo o desenvolvimento de um algoritmo que permita saber a localização de robôs futebolistas num campo de futebol robótico com as dimensões utilizadas na liga MSL do *RoboCup*.

São propostos dois algoritmos com baixo custo computacional. Um que permite calcular a orientação do robô com base em histogramas e outro que possibilita calcular a posição do robô. Os dois algoritmos recorrem ao processamento de imagem para extrair a informação necessária para o cálculo da orientação e da posição.

A abordagem proposta para o cálculo da posição do robô consiste na utilização das distâncias às linhas de campo brancas mais próximas. O processamento de imagem assenta na detecção de transições linha-campo ou vice-versa, sendo esta detecção feita com recurso à análise de linhas radiais e de linhas axiais, permitindo uma redução significativa no tempo de processamento.

Palavras-chave: Localização de robôs, orientação de robôs, *RoboCup*, Robôs futebolistas, Processamento de imagem.

Localization of soccer robot MINHO TEAM

ABSTRACT

Soccer Robot localization in the MSL football field is fundamental for team play and cooperation between robots, which makes it a requirement to use reliable and robust robots localization.

This research work aims to develop an algorithm that allows robots localization in a MSL football field with the standard RoboCup MSL league dimensions.

Two low computational cost algorithms are proposed. One calculates the robot orientation based on histograms and the other one to calculate the robot's position. Both algorithms use computer vision and image processing to grab the relevant information required to calculate the orientation and position.

The proposed approach for calculating the robot's position consists on the usage of the nearest white lines distances. The image processing is based on the detection of line-field transitions or vice-versa, and this detection is carried out using the radial and axial lines analysis, allowing a significant processing time reduction.

Keywords: Robot localization, Robot orientation, RoboCup, Soccer robots, Image processing

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ACRÓNIMOS

AP → Access Point

Cm → Centímetros (unidade de comprimento)

Dm → Decímetros (unidade de comprimento)

FNR → Festival Nacional de Robótica

Fps → Frames por segundo

GHz → Gigahertz (unidade de frequência)

ISEP → Instituto Superior de Engenharia do Porto

LAR → Laboratório de Automação e Robótica

LUT → Look Up Table

M → Metros (unidade de comprimento)

Ms → Milissegundos (unidade de tempo)

MSL → Middle Size League (Liga de tamanho médio)

CAPÍTULO 1

1. INTRODUÇÃO

A equipa Minho começou as suas actividades no fim de 1997 e participou activamente em vários eventos tendo conseguido alguns resultados de relevo. Foi três vezes campeã do FNR (2004-2006), vice-campeã do FNR em 2007, três quintos lugares no *RoboCup* (2004-2006). Conquistou ainda um terceiro lugar no *GermanOpen* em 2004, um segundo lugar e um quarto lugar no *GermanOpen* em 2005 e 2007 respectivamente, tendo conseguido também um segundo lugar no *RoboLudens* em 2006. Em 2007 a equipa Minho interrompeu a sua participação nas competições para desenvolver uma nova equipa de robôs mais rápidos. [**1**] Na Figura 1.1 é possível ver a evolução dos robôs da equipa Minho desde 1998 até 2005.

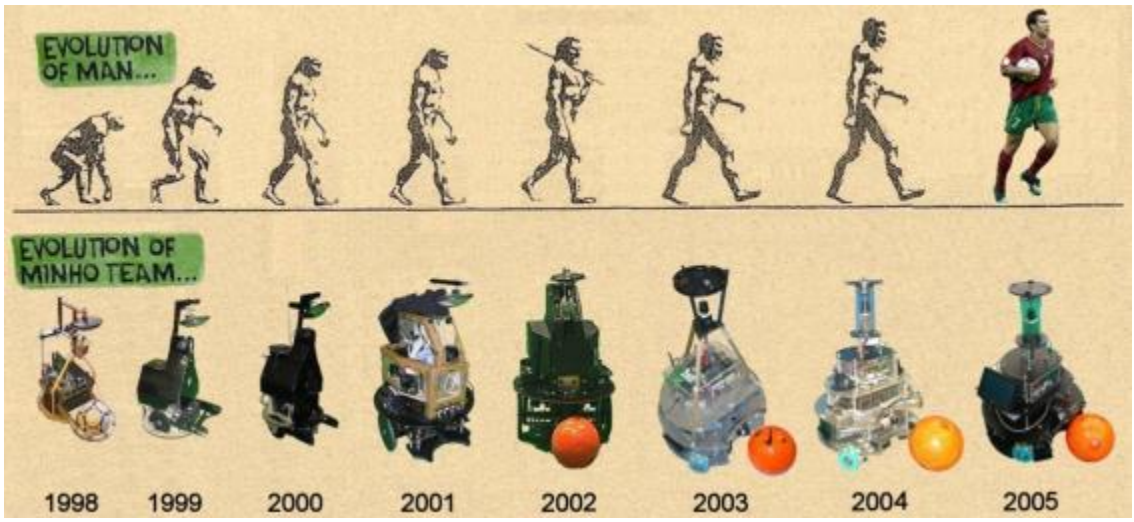


Figura 1.1 - Evolução da equipa Minho TEAM

Este projecto recomeçou em Setembro de 2010 com uma nova equipa, cujos robôs foram construídos totalmente de raiz. Na Figura 1.2 é apresentada a equipa MINHO TEAM.

Figura 1.2 - Equipa MINHO TEAM

Neste primeiro capítulo é apresentado o *RoboCup*, o Festival Nacional de Robótica bem como uma breve explicação do que é um robô. É ainda apresentado o problema que origina esta dissertação, assim como os objectivos da mesma.

1.1. Robô

Robô é um dispositivo, ou grupo de dispositivos electromecânicos capazes de realizar trabalhos de forma autónoma, pré-programada ou com controlo humano. O termo robô deriva da palavra checa *robota*, que significa "trabalho forçado". [**2**]

Os robôs são bastante utilizados para realizarem tarefas em locais mal iluminados ou na execução de tarefas perigosas para o ser humano. Os robôs industriais são o tipo de robôs mais conhecidos devido às tarefas que executam. Estes podem ser encontrados em linhas de montagem, embalamentos e até transporte de cargas. Actualmente é possível encontrar robôs a realizar tarefas mais "banais", sendo designados de robôs de serviços, cujas tarefas mais comuns são cortar a relva e aspirar. Também é possível encontrar robôs noutros ambientes, onde se pode destacar a área da saúde, na qual já se utilizam robôs para fazer certos tipos de cirurgia onde é necessária bastante precisão.

Robô móvel autónomo é um robô dotado de movimento capaz de realizar tarefas de forma autónoma, isto é, sem intervenção humana.

1.2. RoboCup

O *RoboCup* é uma iniciativa científica internacional criada em 1997. É realizado anualmente num país diferente, sendo o seu objectivo desenvolver a inteligência artificial e a robótica. Este evento tem como objectivo o desenvolvimento de robôs futebolistas capazes de em 2050 derrotarem a selecção campeã do mundo.

Desde então o *RoboCup* foi-se expandindo para outras áreas relevantes com base nas necessidades da sociedade. Actualmente é composto pelos seguintes temas: [**3**]

- *RoboCup Rescue;*
- *RoboCup @home*;
- *RoboCup Junior*;
- *RoboCup Soccer;*

1.2.1. RoboCup Rescue

O *RoboCup Rescue* é uma competição de busca e salvamento, a qual tem como desafio o desenvolvimento de um robô capaz de andar num cenário de desastre.

O resgate em cenários de desastre é uma das questões mais graves que envolve um grande número de agentes heterogéneos no ambiente hostil. O objectivo do *RoboCup Rescue* é promover a pesquisa e desenvolvimento a vários níveis, como a coordenação de trabalho de equipa envolvendo multiagentes e também agentes físicos robóticos para busca e salvamento. [**4**]

O robô deve ser capaz de detectar possíveis vítimas do desastre através de fontes de calor, som ou mesmo movimento.

1.2.2. RoboCup @home

O *RoboCup @Home* é uma competição que visa o desenvolvimento de tecnologia para um robô de serviços e assistência, com grande relevância para o futuro pessoal de aplicações domésticas. É a maior competição internacional para robôs autónomos de serviços. O robô deve ser capaz de se movimentar numa casa, seguir uma certa

pessoa. Nesta competição o robô deve ser capaz de se apresentar, bem como interpretar comandos de voz, visando assim a interacção humano-robô. [**5**]

1.2.3. RoboCup Junior

O *RoboCup Junior* é uma iniciativa orientada para projectos educacionais para jovens estudantes. Tem como objectivo introduzir estudantes do ensino básico e secundário no *RoboCup*, bem como estudantes sem recursos para as actividades das ligas sénior. O *RoboCup Junior* fornece uma introdução emocionante para o campo da robótica, sendo uma nova forma de desenvolver habilidades técnicas por meio de experiências com electrónica, *hardware* e *software*. O *RoboCup Júnior* é uma oportunidade altamente motivadora para aprender a trabalhar em equipa de modo a alcançar um objectivo comum, enquanto se compartilha tecnologia com os amigos. As provas que constituem o *RoboCup Junior* são: [**6**]

- Futebol;
- Dança;
- Busca e salvamento;

1.2.4. RoboCup Soccer

O *RoboCup soccer* foi a primeira competição do *RoboCup*, estando a mesma relacionada com o objectivo principal do evento. A principal preocupação desta liga é a investigação sobre sistemas multiagente e cooperação entre vários robôs em ambientes dinâmicos. Um requisito da liga de futebol robótico é que todos os robôs envolvidos na competição têm que ser completamente autónomos. A competição de futebol robótico está dividida em várias ligas sendo elas: [**7**]

- *Middle Size League* (MSL);
- *Small Size League* (SSL);
- *Humanoid League*;
- *Simulation League*;

Na MSL os robôs têm dimensões máximas de 52 cm de lado, com uma altura máxima de 80 cm, sendo o peso máximo de 40 kg. O jogo disputa-se num campo de futebol similar a um campo de futebol humano, cujas dimensões oficiais actuais são de 18 m x 12 m. Actualmente o número de jogadores permitidos na competição são 5 elementos por equipa. [**8**]

O futebol é um jogo em que é imperativo recorrer à cooperação entre os jogadores, sendo para isso necessário que o jogador tenha conhecimento da sua posição e orientação no recinto de jogo, bem como dos restantes elementos da equipa. É ainda importante saber a posição dos adversários para melhor definir as acções que cada jogador deve tomar.

1.3. Festival nacional de robótica

O festival nacional de robótica teve a sua primeira edição em 2001, sendo o principal objectivo a promoção da Ciência e da Tecnologia junto dos jovens do ensino básico, secundário e superior através de competições de robôs. Este evento promove também um encontro científico, no qual investigadores estrangeiros e nacionais da área da robótica se reúnem para apresentar os mais recentes resultados da sua actividade. [**9**]

O festival nacional de robótica tem várias competições, sendo algumas delas iguais às do *RoboCup*, estas encontram-se assinaladas entre parêntesis: [**10**]

- Busca e Salvamento Júnior (*RoboCup Junior*)
- Dança Júnior (*RoboCup Junior*)
- Futebol Robótico Júnior (*RoboCup Junior*)
- Condução Autónoma
- Liga INFAIMON Futebol Robótico Médio (*RoboCup*)
- Freebots
- Robot@Factory

1.4. O problema

Os robôs futebolistas são uma categoria de robôs da competição mundial de robótica conhecida como *RoboCup*. Esta competição tem várias ligas sendo a principal a liga MSL. Nesta liga os robôs são autónomos, têm dimensão máxima de 52 cm x 52 cm x 80 cm e jogam, no máximo, 5 robôs por equipa. Os robôs podem comunicar via rede sem fios (WiFi) com um computador externo conhecido como monitor, que pode agir como treinador da equipa. O meio onde o robô se movimenta é o campo de futebol, cujas dimensões são 18 m x 12 m. O campo é constituído por uma alcatifa de cor verde, com marcações ou linhas de campo e balizas brancas, tem alguns círculos pretos e brancos em posições específicas do campo. Para que o robô possa jogar, é necessário que este cumpra um conjunto de regras básicas, tal como evitar o embate noutros robôs, localizar a bola de jogo e rematar à baliza adversária. Sem a noção da sua posição no campo, um robô futebolista consegue localizar a bola e até pode marcar golo, mas não consegue efectuar passes, repor a bola em jogo e desenvolver qualquer estratégia de jogo. O conhecimento da sua localização é por isso importante e necessário.

1.5. Objectivos

O principal desafio deste projecto consiste no desenvolvimento de um algoritmo e sua implementação em *software* capaz de efectuar a localização do robô no campo de futebol. A localização deve ser fiável e robusta.

Os objectivos propostos no início do projecto consistiram em:

- Estudar os algoritmos de localização já desenvolvidos por outras equipas
- Implementar esses algoritmos no robô futebolista e efectuar os testes necessários
- Identificar a posição de outras entidades na vizinhança e linha de vista do robô
- Enviar as suas próprias coordenadas, bem como as coordenadas das entidades identificadas ao monitor

1.6. Motivação

A motivação para o desenvolvimento deste trabalho prendeu-se com o facto da equipa Minho ter imensa vontade de voltar a participar nos campeonatos de futebol robótico. Para se poder concretizar esse desejo havia a necessidade de construir uma equipa, sendo necessário implementar um algoritmo capaz de localizar (posição, orientação) o robô no recinto de jogo. Espera-se com o desenvolvimento e implementação do algoritmo, que a equipa Minho continue nas competições a fim de evoluir o *software*, permitindo também melhorar a sua prestação nas competições em que venha a participar.

CAPÍTULO 2

2. ESTADO DA ARTE

A robótica móvel autónoma é uma área de pesquisa cujo principal objectivo é a operação de um robô sem intervenção humana. Para um robô operar autonomamente é necessário que o mesmo seja capaz de responder a uma pergunta importante, “onde estou?”, a resposta a esta pergunta é a localização do robô.

Este segundo capítulo apresenta alguns princípios fundamentais que permitem fazer a localização de um robô, bem como uma comparação do que algumas equipas utilizam para calcular a localização dos seus robôs no futebol robótico na secção 2.1 e 2.2 respectivamente. No fim de cada secção há um pequeno resumo que faz uma síntese dos métodos apresentados. De entre os princípios fundamentais que são apresentados tem-se o método de Monte Carlo, a Localização de *Markov* e a localização com base no Filtro de Kalman. Quanto às equipas analisadas optou-se por três equipas: a equipa *Brainstormers Tribots*, porque desenvolveu um algoritmo que permite fazer a localização com base na imagem com um tempo de computação baixo (cerca de 4ms), a equipa CAMBADA pois actualmente é a melhor equipa portuguesa, e por fim a equipa *Water*, que apesar de ser uma equipa recente, foi campeã nas duas últimas edições do RoboCup.

2.1. Princípios fundamentais

2.1.1. Método de Monte Carlo

O método de Monte Carlo é um método estatístico utilizado em simulações estocásticas com diversas aplicações. Este método tem várias aplicações, sendo as

mais comuns na área da computação numérica para avaliar integrais. A ideia do método é escrever o integral que se deseja calcular como um valor esperado. [**11**]

A localização de Monte Carlo, também denominada por filtro de partículas, representa a distribuição de probabilidades de uma determinada posição por um conjunto de N amostras aleatórias. Um conjunto de partículas constitui uma aproximação discreta de uma distribuição de probabilidades. Partículas são pares ordenados de posição e peso, (x^i, π^i), para $i = 1, \ldots, N$, onde $x = (x, y, \theta)$ e $\pi \geq 0$ é o factor de importância, análogo a uma distribuição de probabilidade discreta com $\sum_{i=1}^{N} \pi^i = 1$. O conjunto de partículas é indicado por $X = \{(x^1, \pi^1), (x^2, \pi^2), \ldots, (x^N, \pi^N)\}$. [**12**]

A função de distribuição de probabilidade inicial da posição do robô representa o conhecimento inicial sobre a sua posição. No caso do robô não conhecer a sua posição inicial a distribuição de probabilidade é inicializada com uma distribuição uniforme sobre todas as posições possíveis. O conjunto de partículas é inicializado escolhendo N partículas distribuídas uniformemente no ambiente e atribuindo pesos também distribuídos uniformemente, ou seja $\pi_0^i = 1/N$ para $i = 1, \ldots, N$. [**12**]

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 **a**, sendo este conjunto indicado por $\bar{X}$. Cada nova partícula (x^i, π^i), para $i = 1, \ldots, N$, é dada por: $x^i \sim p(x^i | x^i, a)$. Quando o robô realiza uma observação **o** do ambiente, esta observação é incorporada pelo filtro de partículas, actualizando o valor do factor de importância de forma que: $\pi_t^i = p(o|x^i)$. Este ajuste atribui pesos maiores para as partículas que correspondem às posturas onde **o** é mais provável de ser observado. [**12**]

A fase seguinte é de reamostragem ou amostragem por importância, que transforma o conjunto de partículas $\bar{X}$ num novo conjunto do mesmo tamanho X, em que as novas partículas são ponderadas pelos seus factores de importância. Ao incorporar os pesos das partículas, a distribuição de partículas muda, concentrando-se nas regiões em que há maior peso. [**12**]

2.1.2. Localização de Markov

A localização de Markov é uma técnica probabilística que guarda uma distribuição de probabilidades sobre o espaço de todas as hipóteses onde o robô pode estar, em vez de guardar uma única hipótese sobre a sua posição. Como o robô não conhece a sua posição exacta, ele possui apenas uma certeza de onde poderia estar, ou seja, a distribuição de probabilidade sobre o espaço de posições. [**12**]

Este método é denominado de Localização de Markov, pois segue a suposição de Markov, em que a posição do robô apenas depende da posição anterior e das observações actuais, sendo independente das observações passadas. [**12**]

A Localização de Markov precisa de dois modelos probabilísticos para manter a estimação da posição: o modelo de movimento e o modelo de observação. Esta técnica apresenta um custo computacional superior em relação ao filtro de partículas. [**12**]

2.1.3. Filtro de Kalman

O filtro de Kalman é um método matemático criado por Rudolf Kalman. Este filtro foi publicado em 1960 como uma solução recursiva para o problema de filtragem linear de sistemas discretos. É um conjunto de equações matemáticas que possibilita estimar recursivamente o estado de um sistema linear, com variância mínima segundo um dado critério de optimalidade, por exemplo, o quadrado do erro médio. As informações de interesse no sistema podem ser representadas por um vector de estado $x(t)$, n-dimensional, no tempo t, no caso da localização este vector será do tipo: $x(t) = (x(t), y(t), \theta(t))$. O filtro de Kalman, estima o estado do sistema utilizando os conhecimentos de: [**12**]

- Modelo da dinâmica do sistema – no caso de robôs móveis, que é o ponto de interesse, representa o modelo do movimento do robô, medido pelos encoders;
- Modelo de observação – representa o modelo dos sensores externos do robô, como por exemplo, uma câmara;

- Estatísticas dos ruídos da dinâmica e erros de observação – incerteza associada ao movimento e aos sensores externos de localização do robô;
- Informações das condições iniciais – localização inicial do robô.

Para iniciar o filtro de Kalman é necessário conhecer os valores estimados para a posição inicial e a matriz de covariância. Assume-se que os erros do movimento do robô e da observação são sequências de erro não correlacionadas no tempo. [**12**]

O filtro de Kalman opera num ciclo de três fases, **predição**, **emparelhamento** e **actualização**. Na primeira fase, **a predição**, a posição do robô e as observações são estimadas num tempo futuro, utilizando o modelo do movimento do robô e o modelo de observação, respectivamente; na segunda fase, **o emparelhamento**, as observações reais são correspondidas com as observações da fase de predição, seleccionando as observações válidas; na terceira e última fase, **a actualização**, a predição da posição é corrigida utilizando as observações válidas. [**12**]

Na Figura 2.1 pode-se ver o ciclo de operação (Predição-emparelhamento-actualização) do filtro de Kalman para a Localização.

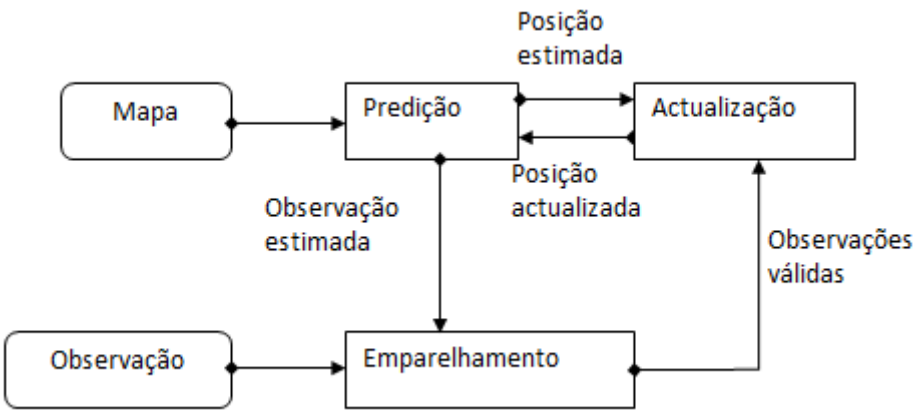


Figura 2.1 - Filtro de Kalman para a localização

As equações que resumem o filtro de Kalman são apresentadas na Tabela 2.1:

Tabela 2.1 - Equações do Filtro de Kalman

Modelos
Modelo de Movimento $x(t+1) = F(t)x(t) + w(t), w(t) = N(0, Q(t))$ **Modelo de Observação** $z(t) = H(t)x(t) + v(t), v(t) = N(0, R(t))$
Predição
Estimação da Posição $\bar{x}(t+1\|t) = F(t)\bar{x}(t\|t)$ **Covariância da Posição** $P(t+1\|t) = F(t)P(t\|t)F^T(t) + Q(t)$ **Estimação das Observações** $\bar{z}(t+1\|t) = H(t)\bar{x}(t+1\|t)$
Emparelhamento
Inovação $r(t+1) = z(t+1) - \bar{z}(t+1\|t)$ **Covariância da Inovação** $S(t+1) = H(t+1)P(t+1\|t)H^T(t+1) + R(t+1)$
Actualização
Cálculo do Ganho de Kalman $K(t+1) = P(t+1\|t)H^T(t+1)S^{-1}(t+1)$ **Estimação do Estado** $\bar{x}(t+1\|t+1) = \bar{x}(t+1\|t) + K(t+1)r(t+1)$ **Covariância da Posição** $P(t+1\|t+1) = P(t+1\|t) - K(t+1\|t)H(t+1)P(t+1)$

2.1.4. Conclusões

A localização de Markov, que utiliza grades de ocupação, representa uma hipótese sobre a posição do robô para cada célula da grade, é capaz de efectuar uma localização global bem como recuperar de falhas. [**12**] Tem a desvantagem de ter um custo computacional elevado.

A localização com base em filtro de partículas representa várias hipóteses sobre a posição do robô, é capaz de efectuar uma localização global. [**12**] A localização com base em filtro de partículas apesar de apresentar um custo computacional inferior à localização de Markov ainda assim é significativo para o objectivo pretendido.

A localização com base no filtro de Kalman representa apenas uma hipótese sobre a posição do robô, sendo este incapaz de efectuar uma localização global ou recuperar de uma perda de localização. O filtro de Kalman tem como vantagem o facto de ser eficiente e preciso. [**12**]

2.2. Comparação entre equipas

2.2.1. Brainstormers *Tribots*

A equipa *Brainstormers Tribots*, foi uma equipa de futebol robótico que terminou o seu projecto em 2009, foi criada em 2002 tendo vencido o RoboCup em 2006 e 2007, e quatro vezes campeã europeia (2004 - 2007). O nome *Brainstormers Tribots* é uma combinação da equipa de simulação (*Brainstormers*) e a equipa da MSL (*Tribots*). O seu logo, representado na Figura 2.2 simboliza essa ideia "*Brainstormers Tribots – two classes, one team*". [**13**]

Figura 2.2 - Logo Brainstormers Tribots

A equipa *Brainstormers Tribots* desenvolveu em 2004 um algoritmo para a localização de robôs futebolistas na MSL. Este algoritmo foi testado pela primeira vez no *RoboCup* 2005. [**14**] O algoritmo desenvolvido baseia-se na qualidade de uma estimativa usando um termo de erro com objectivo de minimizá-lo numericamente. Segundo os testes realizados pela equipa foi possível concluir que o algoritmo desenvolvido possui uma alta precisão, sendo também robusto e computacionalmente eficiente. O principal objectivo da equipa para o algoritmo de localização era reduzir o tempo de computação para menos de 15 ms, este objectivo foi largamente conseguido uma vez que o tempo de computação necessário é de cerca de 4 ms. Importa salientar que estas medições de tempo realizadas pela equipa *Brainstormers Tribots* foram realizadas num computador JVC com um processador Pentium de 1GHz. O algoritmo foi implementado em C++ e foi testado num campo com as dimensões utilizadas no RoboCup 2004. Na Tabela 2.2 são apresentados alguns tempos de computação em ms para o filtro de partículas e para o algoritmo desenvolvido por esta equipa. [**15**]

Tabela 2.2 - Tempo de computação médio em milissegundos

Abordagem	**Tempo de computação médio por ciclo (ms)**
Filtro de Partículas com 500 partículas	48.3
Filtro de Partículas com 200 partículas	17.9
Minimizar o erro de auto localização	4.2

O algoritmo desenvolvido pela equipa *Brainstormers Tribots* utiliza principalmente as linhas brancas do campo para efectuar a localização. O primeiro passo do algoritmo é detectar na imagem as linhas brancas, tendo como resultado uma lista de pontos com coordenadas em relação ao centro do robô. De seguida, é aplicado um algoritmo de correspondência que encontra a posição do robô em que as linhas encontradas combinam perfeitamente com o conhecimento das linhas do campo. Ao mesmo tempo esta abordagem também produz medidas de fiabilidade que podem ser usadas num processo de integração temporal para obter trajectórias suaves do movimento do robô ao longo do tempo. [**14**]

Como reduzir o tempo de processamento era um dos principais objectivos da equipa *Brainstormers Tribots*, esta decidiu utilizar linhas radiais para analisar a imagem capturada da câmara, permitindo assim reduzir o tempo de computação. [**16**]

2.2.2. CAMBADA

CAMBADA é um acrónimo para *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*. A equipa CAMBADA é uma das cinco equipas portuguesas que participam na MSL, pertencendo à Universidade de Aveiro. É uma equipa relativamente recente, foi criada em 2003, tendo iniciado a sua participação nos campeonatos de futebol robótico em 2004. Desde então conquistou um *RoboCup* em Suzhou (China) no ano de 2008, três terceiros lugares também no *RoboCup* em 2009, 2010 e 2011, um terceiro lugar no FNR em 2006, cinco vezes campeões do FNR (2007 - 2011), e vice-campeões do *GermanOpen* em 2010. [**17**]

O algoritmo de localização da equipa CAMBADA é baseado nas linhas do campo detectadas, com fusão sensorial da odometria e de uma bússola electrónica. A localização é feita com base no algoritmo desenvolvido pela equipa *Brainstormers Tribots*, com algumas alterações. O algoritmo utilizado pode ser visto como uma tarefa de minimização de erros, com uma medida derivada da fiabilidade da posição calculada, de modo que um processo de fusão sensorial estocástico pode ser aplicado para aumentar a precisão da estimativa. [**18**]

A ideia do algoritmo utilizado pela equipa CAMBADA é analisar os pontos de linha detectados, estimando uma posição, e através de uma função de erro descrever a aptidão da estimativa. Este processo é feito através da redução do erro da correspondência entre as linhas detectadas e as linhas do campo conhecidas. A função de erro deve ser definida considerando a quantidade substancial de ruído que afecta os pontos da linha detectados, tal provocaria uma distorção da estimativa de representação. [**18**]

A equipa CAMBADA refere ainda que a qualidade da medição de odometria é bastante afectada com o tempo, sendo que dentro dos tempos de ciclo reduzidos conseguidos na aplicação, e com leituras consecutivas consegue-se obter resultados

aceitáveis e, portanto, fundindo a estimativa visual com os valores de odometria consegue-se aprimorar a estimativa. O filtro de Kalman é utilizado para fazer a fusão sensorial entre a posição estimada do robô através da odometria e a posição estimada do robô através da informação visual. Esta abordagem permite ao agente estimar a sua posição mesmo que a informação visual esteja indisponível. No entanto, não é confiável usar apenas valores de odometria para estimar a posição por mais que alguns ciclos, pois deslizamentos e atritos nas rodas podem produzir grandes erros nas estimativas num curto espaço de tempo. [**18**]

A equipa refere ainda que a orientação estimada visualmente pode ser ambígua, isto é, cada ponto no campo de futebol tem uma posição simétrica, relativamente ao centro do mesmo, e o robô detecta exactamente as mesmas linhas do campo. Para eliminar a ambiguidade é utilizada uma bússola electrónica. A orientação estimada pelo robô é depois comparada com a orientação dada pela bússola e se o erro entre eles é maior do que um limiar pré-definido, são tomadas acções que variam conforme o erro. [**18**]

O erro é a diferença entre a direcção dada pela imagem e a direcção dada pela bússola. Na Figura 2.3 pode ver-se a certificação da localização e a classificação da diferença dos ângulos. Importa salientar que a zona verde representa uma localização correcta, a zona amarela representa uma localização correcta mas invertida, enquanto as zonas vermelhas representam uma localização errada, sendo por isso necessário efectuar uma relocalização. [**19**]

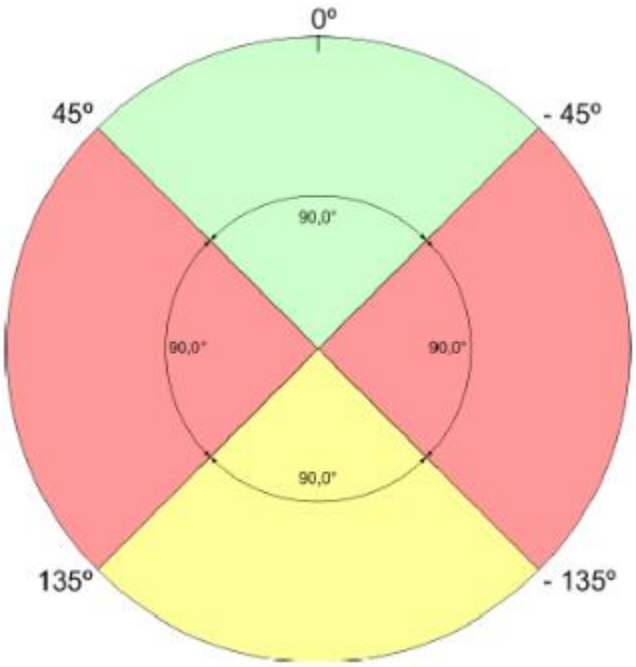


Figura 2.3 - Detecção de erros de localização baseado numa bússola

2.2.3. Water

A equipa *Water* foi fundada em 2003 e pertence à Universidade de Pequim (*Beijing Information Science and Technology University*). Esta equipa participa no *China Open* desde 2006, tendo ficado em segundo lugar em 2008 e 2009, sendo que em 2010 venceu a competição. Participa no *RoboCup* desde 2009, tendo no seu ano de estreia ficado em sétimo lugar, vencendo as duas edições seguintes 2010 e 2011. [**20**] [**21**]

A equipa *Water* utiliza as linhas do campo para saber a posição e orientação do robô no campo, utiliza também um modelo de linha do campo. Através das linhas do campo e do modelo de linha têm a localização em relação à linha do campo. Na Figura 2.4 pode-se ver a azul o modelo da linha do campo e a cor-de-rosa a linha do campo detectada. A função do algoritmo é fazer coincidir a linha do campo com o modelo de linha do campo. A coordenada absoluta no campo é calculada através da matriz de transformação de ângulo e distância. [**20**]

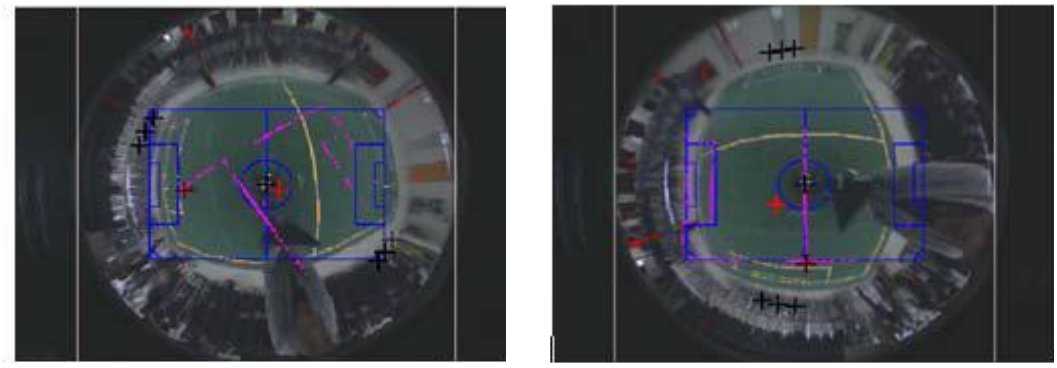


Figura 2.4 - Método de localizaçao da equipa Water

Na Figura 2.5 pode-se visualizar o fluxograma do algoritmo que a equipa *Water* utiliza para fazer coincidir as linhas do campo com o modelo de linha do campo.

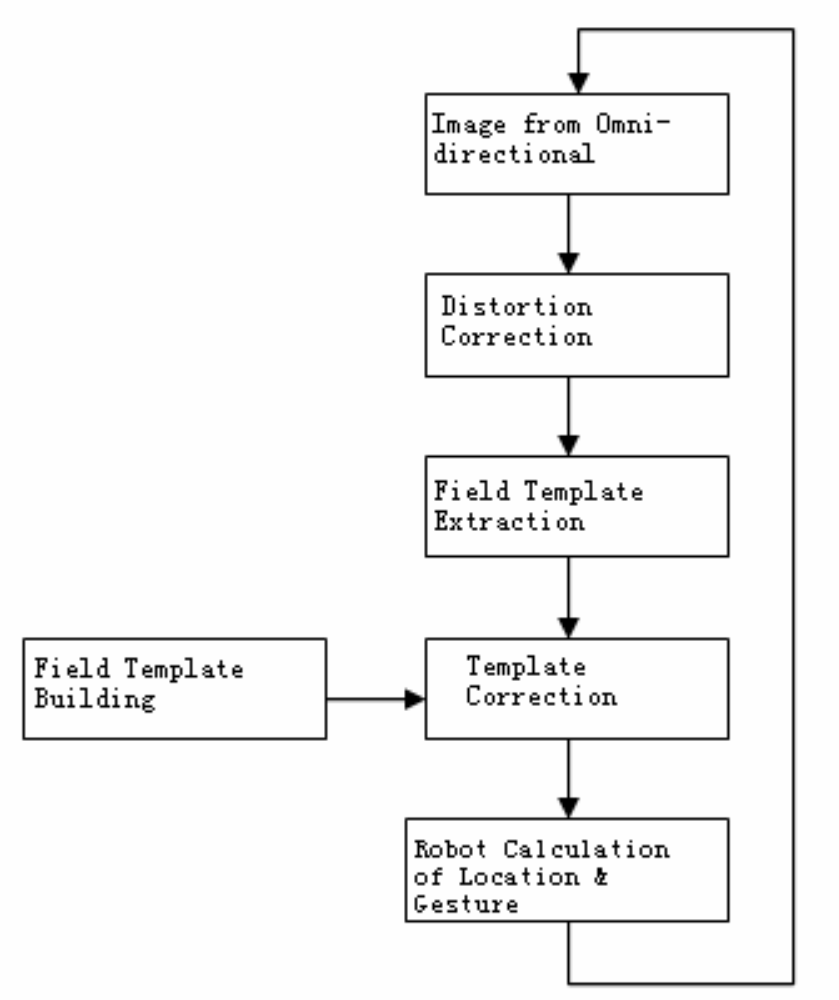


Figura 2.5 - Fluxograma de correspondência

2.2.4. Conclusões

O algoritmo desenvolvido pela equipa *Brainstormers Tribots* apresenta-se como um algoritmo robusto e rápido uma vez que consegue resolver o problema da localização do robô de forma fiável com um tempo de computação médio de 4,2 ms, que comparado com o filtro de partículas é bastante reduzido uma vez que para o caso de 200 partículas demora 17,9 ms. [**15**]

Apesar dos testes efectuados pela equipa *Brainstormers Tribots* terem sido realizados em 2005, altura em que o campo tinha dimensões mais pequenas (8 m x 12 m) quando comparado com as actuais dimensões (12 m x 18 m), o algoritmo continua a funcionar na actual configuração do campo, o que ajuda a comprovar a robustez do mesmo. [**14**]

Pode-se também concluir que as equipas CAMBADA e *Water* utilizam algoritmos semelhantes ao desenvolvido pela equipa *Brainstormers Tribots*.

CAPÍTULO 3

3. O ROBÔ

Neste terceiro capítulo são abordados temas como os materiais utilizados para a construção do robô, a sua forma e a estrutura da cabeça de visão utilizada.

3.1. A forma e materiais

A forma escolhida para a construção do robô foi a circular, uma vez que já era a forma utilizada anteriormente, permitindo também ocupar um maior espaço em campo sem criar pontos críticos na estrutura, ou seja, pontos que podem fracturar com embates em outros robôs ou mesmo com outros objectos. Esta forma permite igualmente maior espaço para distribuição de toda a electrónica utilizada no funcionamento do robô.

A estrutura utilizada foi a anteriormente desenvolvida com alguns ajustes, optando-se por utilizar alumínio, pois é um material que proporciona maior robustez ao robô ao contrário da anterior abordagem testada. [**22**] Na Figura 3.1 é possível ver-se o desenho CAD do robô ainda sem a parte da cabeça e sem chuto vertical.

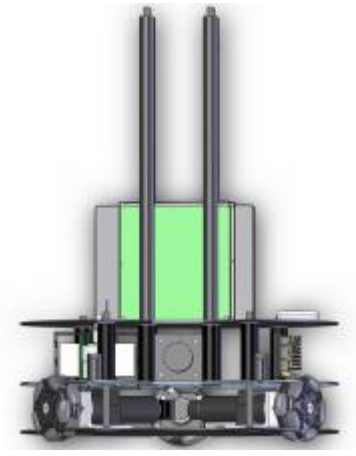


Figura 3.1 - Desenho CAD do robô

3.2. Forma física da cabeça

Optou-se por desenvolver outra cabeça de visão em relação à proposta em [**22**], uma vez que na estrutura anterior tornava-se complicado garantir que as hastes verticais ficavam no sítio, sendo também complicado garantir a robustez da cabeça. Um outro problema seria fixar o suporte da câmara na cabeça proposta em [**22**], por isso, desenvolveu-se uma nova cabeça que resolve os problemas da versão anterior. Na Figura 3.2 pode-se verificar a cabeça anterior e a nova cabeça desenvolvida.

a) b)

Figura 3.2 - a) CAD da cabeça antiga; b) CAD da cabeça nova

Na Figura 3.3 é apresentado o robô final da equipa Minho. Este robô foi utilizado no *RoboCup 2011* em Istambul. De salientar que foi acrescentada em relação ao CAD, anteriormente apresentado, uma estrutura metálica logo abaixo da cabeça apelidada de mão-de-amigo para ajudar a dar robustez à torre que suporta a cabeça.

Figura 3.3 - Actual robô MINHO TEAM

Importa também referir que toda a estrutura metálica (as 3 bases, mão-de-amigo, e cabeça) foi lacada de preto com dois objectivos, primeiro para isolar o alumínio de possíveis contactos com placas electrónicas, uma vez que o alumínio é um bom condutor eléctrico; o segundo objectivo pode-se dizer que é um pré-requisito, uma vez que o robô deve ser maioritariamente preto. [**8**]

3.3. Conclusões

Com a participação no *RoboCup* e com os testes anteriormente efectuados foi possível concluir que a estrutura actual é mais robusta que a apresentada em [**22**], essencialmente devido ao material utilizado.

Foram também detectados alguns pontos na estrutura que podem ser melhorados em eventuais *upgrades* que se venham a fazer. Esses pontos irão ser apresentados, mais à frente, quando forem abordadas sugestões para trabalho futuro.

CAPÍTULO 4

4. LOCALIZAÇÃO

Neste quarto capítulo é descrito o algoritmo para o cálculo da localização dos robôs no campo.

A localização de um robô futebolista no campo de futebol consiste num algoritmo capaz de detectar a posição e orientação do robô no campo em relação a um sistema de eixos coordenados. O sistema de eixos coordenados utilizado é o sistema de eixos do campo, tal como está representado na Figura 4.1.

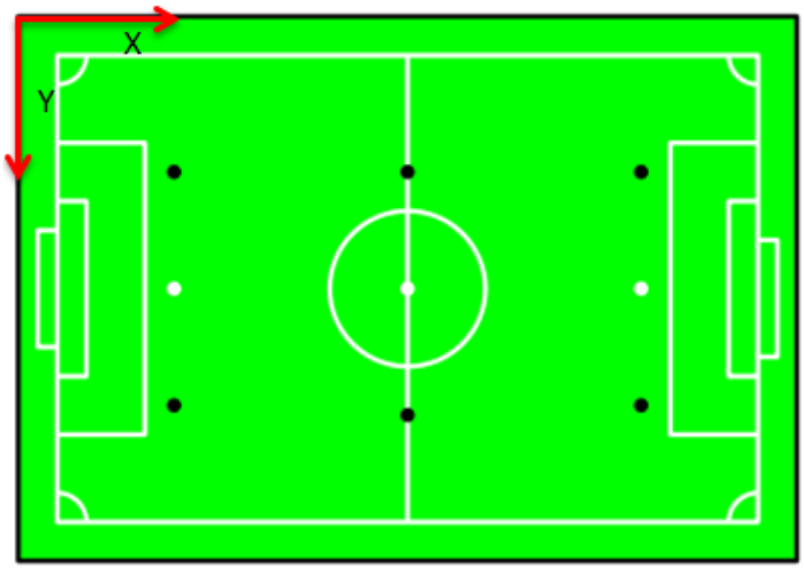


Figura 4.1 - Campo e respectivo sistema de eixos

Para realizar a localização é necessário ter em memória o campo utilizado, sendo que a configuração escolhida pela equipa Minho é uma configuração binária a qual é apresentada na Figura 4.2 (note-se que o '0' corresponde à cor verde e o '1' corresponde à cor branca). Importa ainda realçar que cada algarismo ('0' ou '1') corresponde a uma área no campo de 1 dm^2.

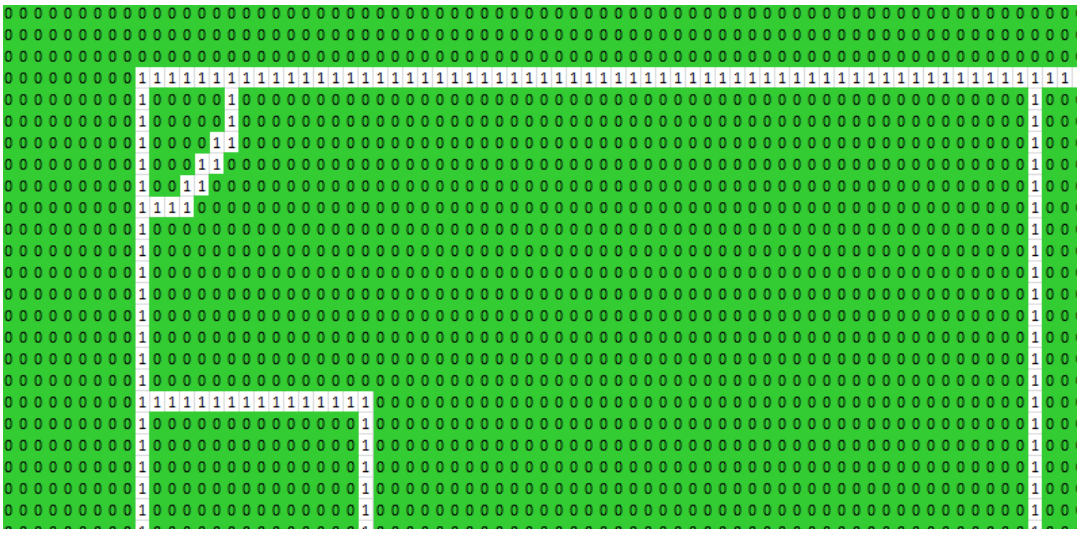


Figura 4.2 - Exemplo de ficheiro do campo em memória

Baseado neste ficheiro do campo (o qual fica alojado em memória para optimizar o processamento) é calculada para cada ponto do campo a distância à linha mais próxima (em dm), sendo este cálculo feito *offline*, isto é, na inicialização do programa. Esta informação também é guardada em ficheiro apenas para fazer alguns testes de *debug*, estando um exemplo importado para Excel visível na Figura 4.3.

Figura 4.3 - Campo gerado em Excel

Através da Figura 4.3 é possível perceber que junto às linhas é mais fácil para o algoritmo saber a localização do robô, sendo que nas zonas onde o gráfico tem os mínimos é mais difícil para o algoritmo saber a localização do robô.

Na Figura 4.4 está representada uma imagem com várias transições detectadas e as distâncias à linha mais próxima para cada transição marcadas pela linha vermelha.

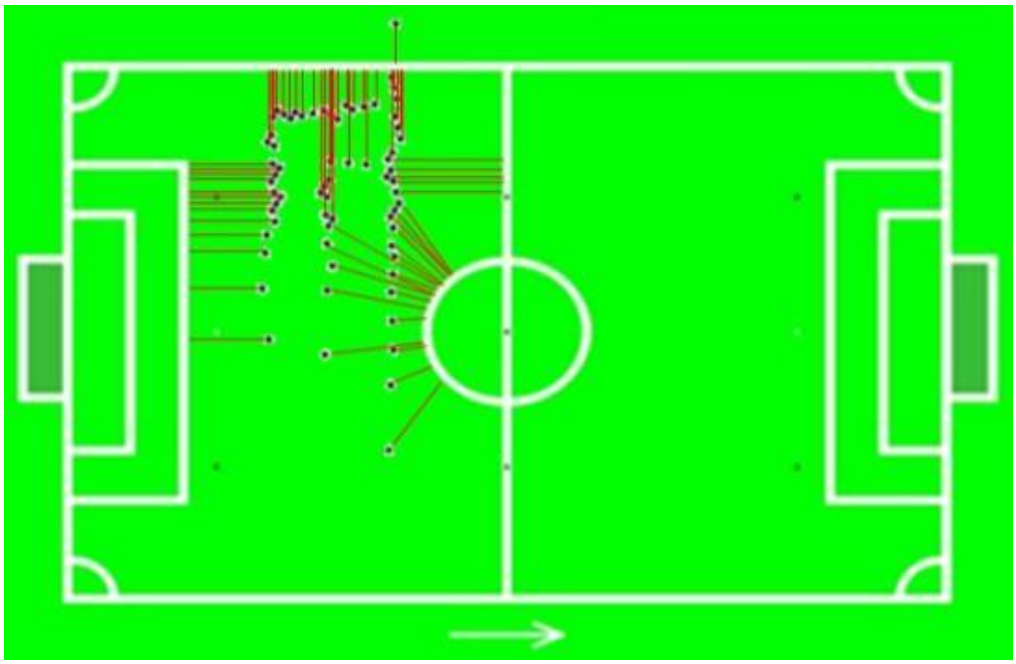


Figura 4.4 - Distância à linha mais próxima

O cálculo da localização do robô assenta nas transições de linha do campo detectadas. Para melhor perceber o algoritmo é necessário perceber o sistema de visão do robô, bem como todo o processamento de imagem necessário.

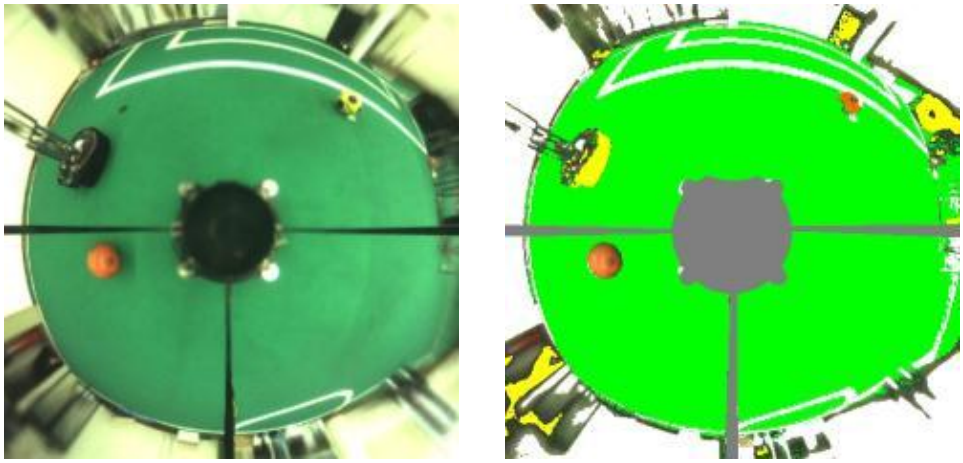
4.1. O sistema de visão

O robô está dotado de um sistema de visão omnidireccional catadióptrico como mostrado na Figura 4.5. O sistema é constituído por uma câmara, *Point Grey Flea 2*, com ligação *Firewire* direccionada para um espelho, possibilitando ver a toda a volta do robô.

Figura 4.5 - Sistema de visão catadióptrico

O primeiro passo do processamento de imagem é a realização da segmentação da mesma. A segmentação da imagem consiste em classificar os pixéis da imagem de acordo com a cor. Este processo é realizado em duas partes: uma feita *offline* e outra em tempo real. No modo *offline* são definidos os limites de HSV para cada cor interveniente, sendo também criada uma LUT em que é feita uma conversão dos limites de HSV para RGB16 de modo a acelerar o processo de segmentação. Em tempo real é feita a segmentação da imagem, de modo a definir as cores utilizadas durante o jogo. A segmentação é feita em RGB16, no entanto os limites são definidos em HSV, uma vez que o este é mais imune às variações de luminosidade que o sistema de cor RGB. Todo o processamento de imagem é feito agora sobre a imagem segmentada.

Na Figura 4.6 é visível uma imagem adquirida directamente da câmara, e a mesma imagem segmentada.



a) b)

Figura 4.6 - a) Imagem adquirida pela câmara; b) Imagem segmentada

Na Figura 4.6 é possível ver que a área cinzenta da imagem resulta da aplicação de uma máscara. Na Figura 4.7 pode ver-se a máscara utilizada, esta tem como função não processar os pixéis que vêem a própria estrutura metálica.

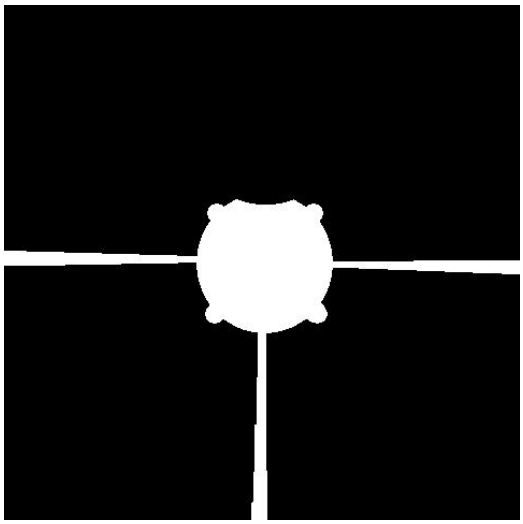


Figura 4.7 - Máscara utilizada

A Figura 4.7 representa as zonas da imagem a serem processadas. Importa salientar que a parte preta da máscara é a parte da imagem que se irá analisar, uma vez que o branco é o que pertence à estrutura do robô.

4.2. Detecção de transições de linha

Para detecção das linhas do campo é efectuada pesquisa radial e axial, poupando assim no tempo de processamento. Se fosse analisada a imagem toda, isto é, 480x480 pixéis, seriam analisados 230400 pixéis, com a pesquisa radial e axial são analisados 28616 pixéis, que representa cerca de 12,4% do total da imagem. Para a detecção de transições são utilizadas radiais com tamanhos diferentes, umas mais curtas que as outras, evitando assim que sejam encontradas bastantes transições junto do centro do robô. As radiais mais compridas estão com um intervalo de 10°, intercaladas pelas radiais mais curtas desfasadas das outras 5°, como a Figura 4.8 sugere, o intervalo entre cada axial é de 10 pixéis. Na Figura 4.8 é apresentada a máscara utilizada para a pesquisa radial e axial, sendo que esta máscara é criada *offline*. Na Figura 4.8 é possível ver a azul a zona de pesquisa radial e a vermelho a zona de pesquisa axial.

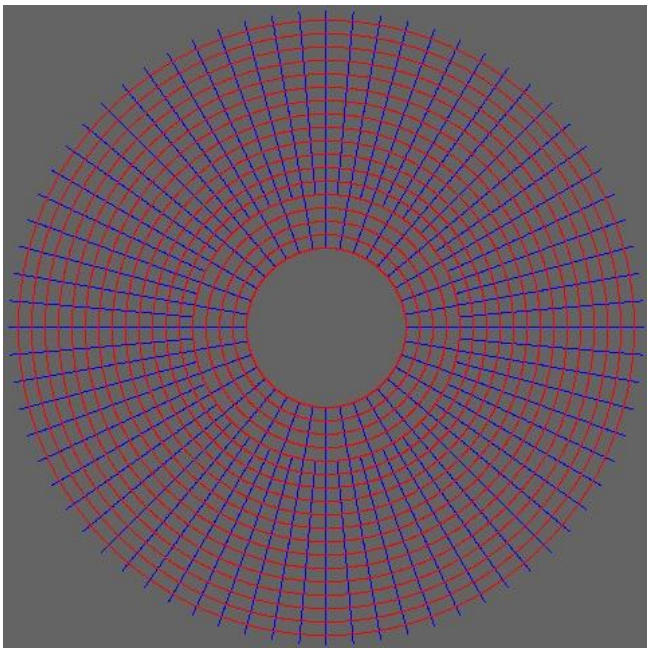


Figura 4.8 - Máscara de pesquisa radial e axial

Utilizando apenas pesquisa radial poderia conseguir-se efectuar o pretendido, no entanto optou-se por se adicionar também a pesquisa axial para o caso em que o robô esteja em cima de linhas. Na Figura 4.9 é verifica-se que na linha em que o robô se encontra, praticamente não detecta transições, ficando bastantes transições por detectar, que ajudariam no cálculo da localização.

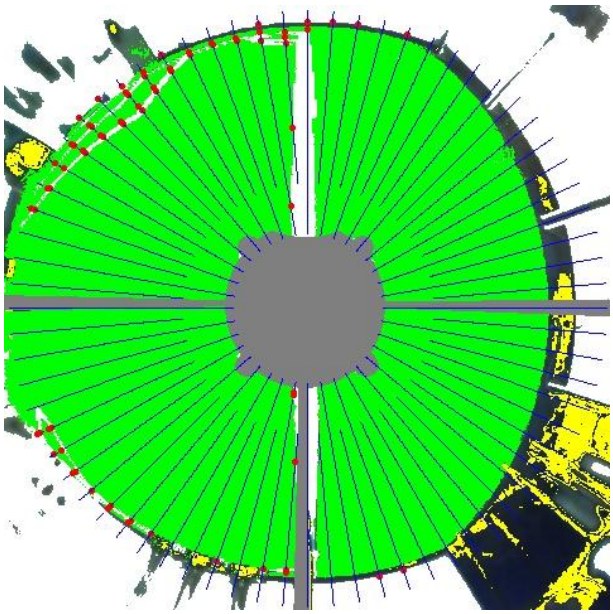
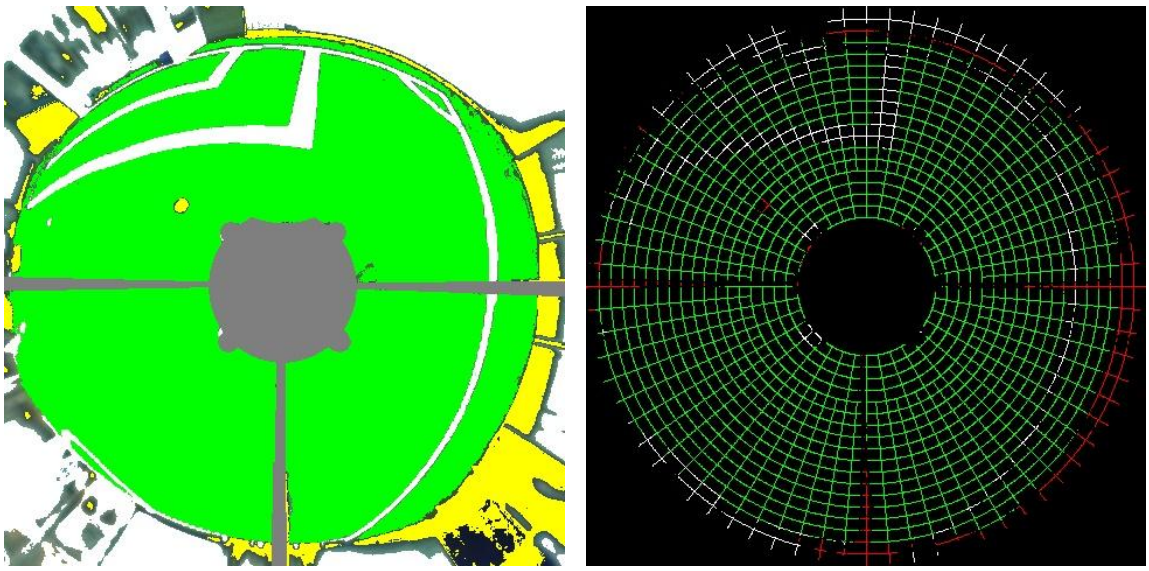


Figura 4.9 - Problema de só utilizar radiais

A Figura 4.10 b) apresenta uma imagem em que é possível ver o que é aproveitado da imagem da Figura 4.10 b) para calcular a localização do robô no campo.



a) b)

Figura 4.10 - a) Imagem segmentada; b) Imagem utilizada para calcular localização

De notar que na Figura 4.10 b) as linhas a vermelho correspondem ao que está a amarelo na Figura 4.10 a).

4.2.1. Como se detectam?

Antes de se efectuar a detecção de linhas, é necessário que haja uma LUT em que é feita a correspondência entre o número de pixéis e a distância em cm.

A detecção de linha é feita percorrendo a máscara da Figura 4.8, sendo de seguida verificado se existe transição de cores na imagem, verde-branco e vice-versa. O algoritmo só considera linha se na transição não detectar mais de três pixéis de outra cor que não branco e verde, como sugere a Figura 4.11, em que L corresponde a linha (branco), o C corresponde a campo (verde) e o X é outra cor.

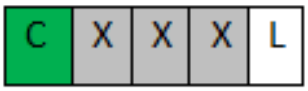


Figura 4.11 - Exemplo de detecção de linha

A Figura 4.12 representa as transições detectadas, bem como a informação que é extraída da imagem para ser guardada no vector, sendo que os pontos a azul dizem respeito às transições detectadas pela pesquisa radial, enquanto os pontos a vermelho dizem respeito às transições detectadas pela pesquisa axial.

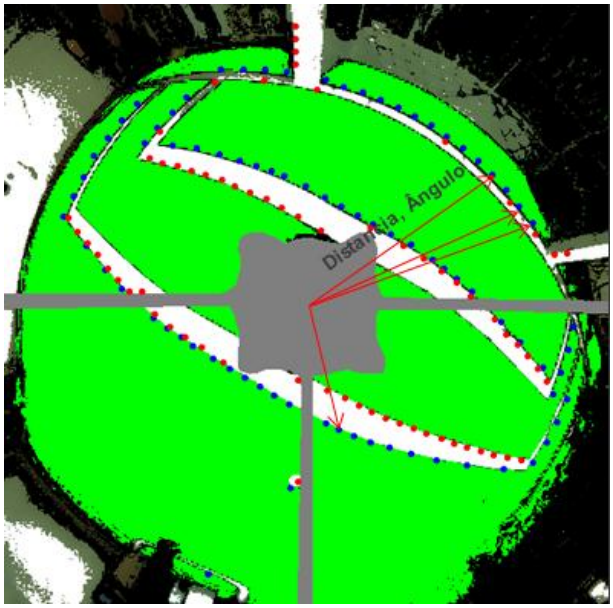


Figura 4.12 - Transições detectadas

Depois de se ter detectada a transição, esta é guardada num vector bidimensional sob a forma de distância em dm e ângulo em graus. Na Tabela 4.1 pode ser visto um exemplo de como são guardadas as transições.

Tabela 4.1 - Exemplo de transições detectadas

Distância (dm)	41	54	41	50	36	48	47
Ângulo (graus)	5	5	10	10	14	14	20

As transições são guardadas no vector ordenadas por ordem ascendente de ângulo. O algoritmo utilizado para a ordenação foi o *Insertion sort* [**23**].

Tendo todas as transições guardadas no vector estão reunidas condições para se iniciar o cálculo da localização do robô, principiando pelo cálculo da orientação, seguido do cálculo da posição do robô.

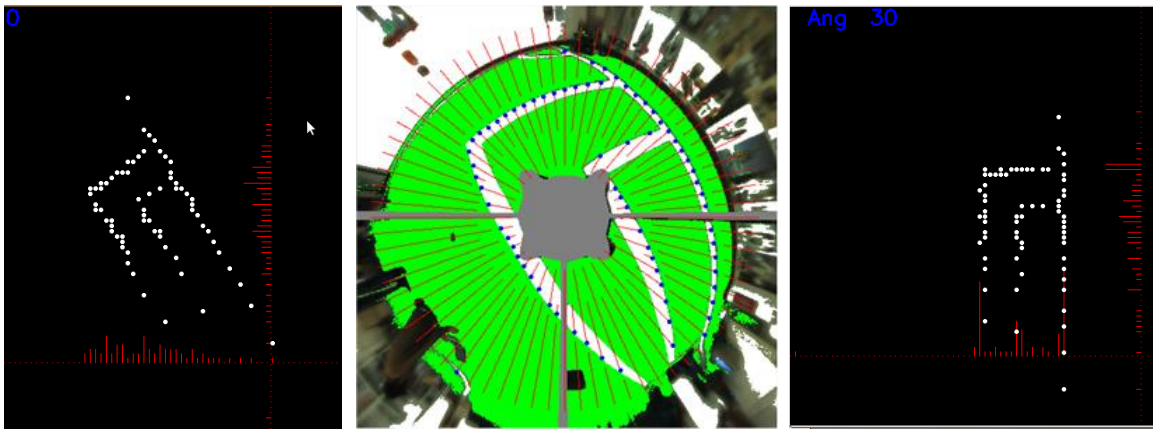
4.3. Orientação do robô

Antes de calcular a posição do robô é necessário saber a sua orientação, isto faz com que o processo do cálculo da posição seja mais rápido, uma vez que já é conhecida a orientação do robô no campo.

A orientação do robô é o ângulo que este faz com o sistema de eixos coordenados da Figura 4.1.

4.3.1. Como se calcula?

Uma vez que neste ponto já existem transições é chegada a hora do algoritmo calcular a orientação do robô, para isso, as transições anteriormente detectadas vão ser rodadas num intervalo de 20°, isto é, entre $-10° + \hat{a}ngulo_{ant}$ *e* $10° + \hat{a}ngulo_{ant}$ em que $\hat{a}ngulo_{ant}$ é o ângulo que o robô tinha na *frame* anterior. Na primeira *frame* da execução do algoritmo é necessário que o robô esteja orientado segundo o eixo do x, isto é, com um ângulo próximo de zero em relação ao sistema de eixos da Figura 4.1, pois $\hat{a}ngulo_{ant} = 0$. Para o jogo foi definido que os zero graus dos robôs são com os mesmos orientados para a baliza adversária, tornando necessário que ao intervalo se reinicialize a orientação dos robôs. O facto de os zero graus serem com os robôs orientados para a baliza adversária, faz com que no intervalo todo o modelo de campo e respectivo sistema de eixos sofra uma rotação de 180°. Por cada incremento de ângulo as transições vão sendo mapeadas em (x, y), permitindo assim ter as transições em imagem assemelhando-se ao formato da zona do campo em que o robô se encontra, como pode ser visto na Figura 4.13 a) e c). Em simultâneo é incrementado cada (x, y) originando dois histogramas, um vertical e outro horizontal, representativos das transições como se pode ver na Figura 4.13 a) e c). A Figura 4.13 b) mostra a zona do campo em que o robô se encontra, sendo que a Figura 4.13 a) representa as transições detectadas na Figura 4.13 b) mapeadas em (x, y) e os respectivos histogramas. Já a Figura 4.13 c) representa como ficam as transições detectadas depois de se saber a orientação do robô, é possível ver ainda que nesta situação os histogramas possuem máximos maiores que os histogramas da Figura 4.13 a).



a) b) c)

Figura 4.13 - a) Histograma sem rotação; b) Transições detectadas; c) Histograma com rotação

Pela análise da Figura 4.13 a) e da Figura 4.13 c) é possível ver no canto superior esquerdo o valor de rotação dos respectivos histogramas.

Em cada incremento de ângulo compara-se se o máximo dos histogramas é superior ao máximo já guardado, caso se verifique actualiza-se o máximo de cada histograma assim como o ângulo a que ocorreu o máximo. O máximo final de cada histograma indica o ângulo de orientação relativo a cada histograma. Os dois ângulos devem ser iguais ou muito semelhantes, mas devido a problemas de arredondamentos ou em situações em que existam poucos pontos, nota-se que há diferenças entre os ângulos calculados. Para corrigir a diferença de ângulos foi criado um algoritmo que ajuda a filtrar qual o ângulo final do robô de acordo com a amplitude dos histogramas. Se a amplitude do histograma horizontal é maior que a amplitude do histograma vertical, o ângulo escolhido é o determinado pelo máximo do histograma horizontal, caso contrário, o ângulo escolhido é o determinado pelo máximo do histograma vertical.

De forma a tornar o algoritmo mais robusto, optou-se por utilizar uma bússola electrónica para verificar o ângulo devolvido pelo algoritmo de orientação, esta encontra-se representada na Figura 4.14. Esta bússola possui uma leitura com resolução de 0,5° e um valor médio de precisão de 2,5°. [**24**]

Figura 4.14 - Bússola electrónica HMC6352

A bússola pode ajudar a verificar se a orientação do robô é a correcta. Para isso faz-se a diferença entre o ângulo da bússola e o ângulo do histograma e utiliza-se o método utilizado pela equipa CAMBADA na Figura 2.3, sendo assim possível averiguar se a orientação calculada pelos histogramas é a correcta.

4.4. Posição do robô

A posição do robô no campo é o ponto (x, y) em que o robô se encontra, sendo calculada com base nas transições de linha detectadas, tal como o cálculo da orientação.

Existem duas possibilidades para o cálculo da posição do robô em tudo semelhantes, sendo que essas possibilidades depois de combinadas com a orientação do robô dão a localização do mesmo, isto é, de acordo com os limites definidos é possível calcular a posição global, ou a posição local do robô.

4.4.1. Como se calcula?

O cálculo da posição do robô é uma função que recebe por parâmetro os limites de pesquisa para o cálculo da melhor posição, que como foi referido acima, permite calcular a posição global ou local do robô. Na Figura 4.15 e na Figura 4.16 pode ver-se a chamada da função de cálculo da posição global, bem como da função de cálculo da posição local respectivamente.

```
localiza->calcPosition(localiza->getCoordenadas()->angulo,
                       0,COMPRIMENTO, 0, LARGURA_CAMPO);
```

Figura 4.15 - Chamada para cálculo da posição global

```
localiza->calcPosition(localiza->getCoordenadas()->angulo,
                       localiza->getCoordenadas()->xcampoAnt-RAIO_PROCURA,  // InicioX
                       localiza->getCoordenadas()->xcampoAnt+RAIO_PROCURA,  // FimX
                       localiza->getCoordenadas()->ycampoAnt-RAIO_PROCURA,  // InicioY
                       localiza->getCoordenadas()->ycampoAnt+RAIO_PROCURA   // FimY
                       );
```

Figura 4.16 - Chamada para cálculo da posição local

Como na primeira execução do programa o primeiro cálculo de localização que é feito é o cálculo da posição global, será abordado este, relembrando que o cálculo da posição global e o cálculo da posição local são em tudo semelhantes.

Para o cálculo da posição global é percorrido todo o campo em dm, e para cada ponto (X, Y) do campo, mapeia-se as transições em (x, y), e soma-se às coordenadas do ponto do campo que se está a analisar. Esta soma é onde o algoritmo “acha” que detectou a transição. De seguida verifica-se se a transição detectada é valida, isto é, verifica-se se a transição está dentro das dimensões do campo (entenda-se os 18 m x 12 m, mais a zona de segurança que existe para além das linhas que delimitam o campo). Se a transição é válida acede-se então à posição de memória que contém a distância à linha mais próxima para a coordenada detectada e incrementa-se o erro com essa distância de acordo com a Equação (1), sendo que isto é feito para cada transição válida. Para despistar eventuais *bugs*, gravaram-se os valores de erro para cada coordenada em ficheiro, estando um exemplo da matriz de erros sobreposta ao modelo do campo na Figura 4.17. Depois de se ter percorrido todas as transições vê-se qual é a coordenada do campo que tem o menor erro, sendo que a coordenada que possui o menor erro é a posição global do robô.

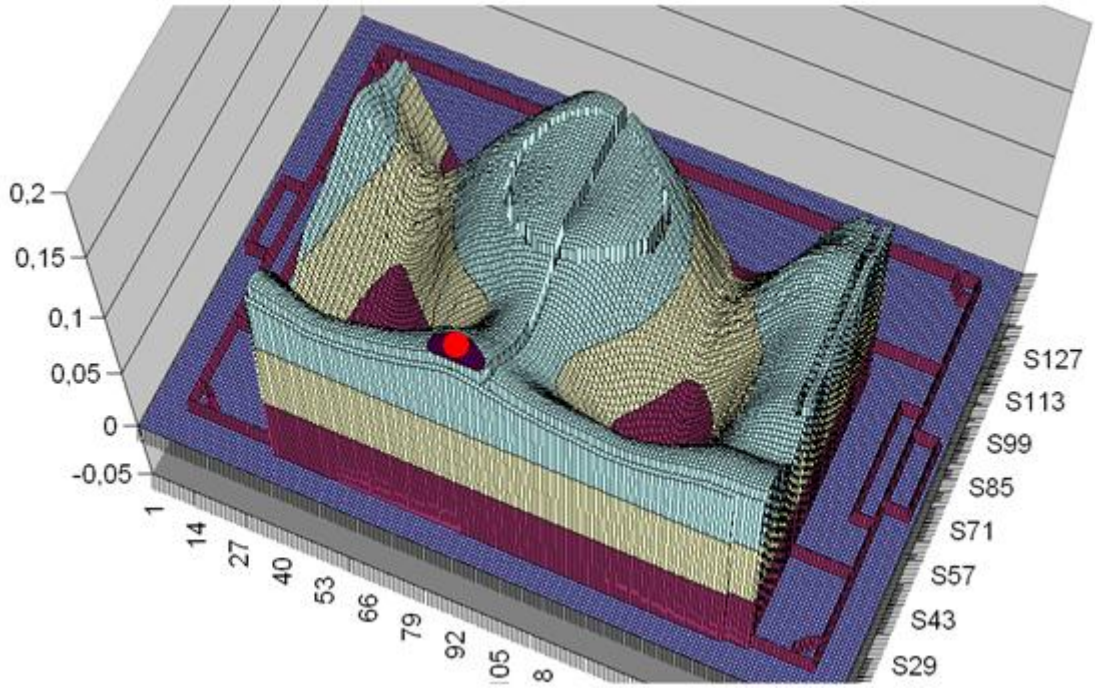


Figura 4.17 - Matriz de erros com modelo do campo

Pela análise da Figura 4.17 pode ver-se que o robô se encontra na zona onde a linha lateral intersecta a linha de meio-campo, sendo que o círculo a vermelho indica a posição do robô. Importa referir que a matriz de erros representada está invertida para facilitar o debug, logo ao estar invertida faz com que a posição correcta seja o maior erro que está representado pelo círculo vermelho.

A Figura 4.18 pretende reproduzir as transições a percorrer todo o campo, de forma a coincidirem o mais possível com o mesmo, este método é utilizado para o cálculo da posição global.

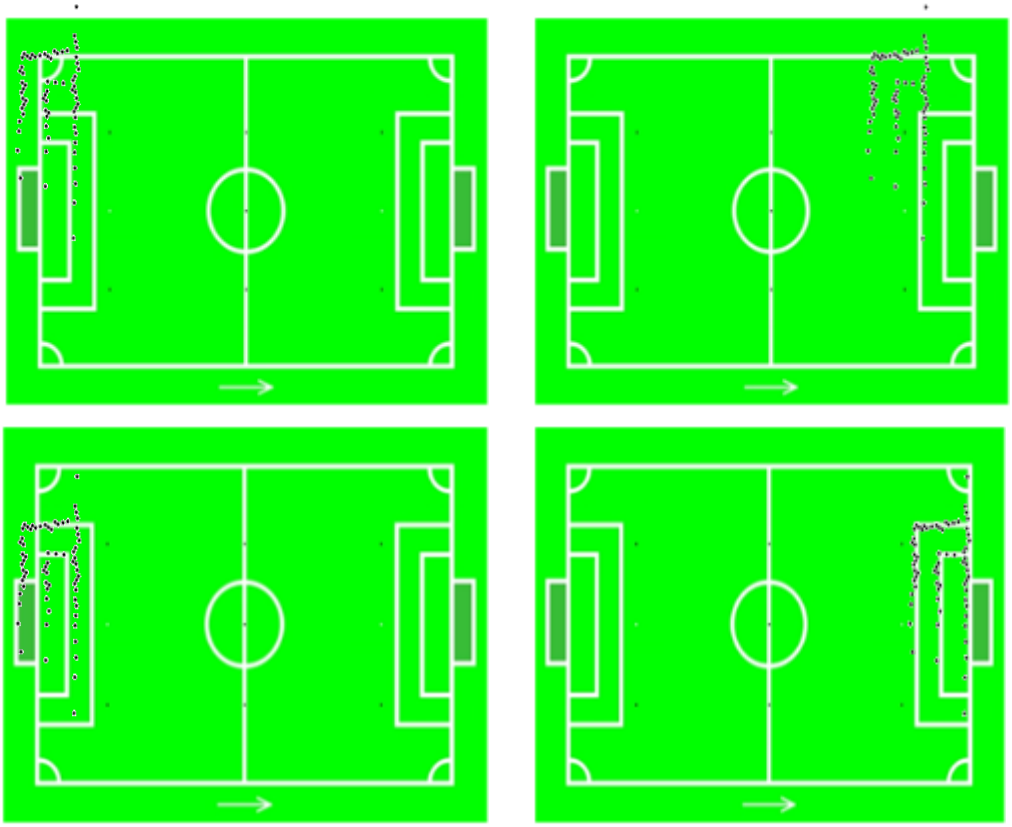


Figura 4.18 – Varrimento do campo para o cálculo da posição global

A Equação (1) é uma função que penaliza desvios entre as transições detectadas e o modelo de linha utilizado, o seu objectivo é minimizar o erro em que c é um parâmetro a partir do qual o erro é atenuado, este parâmetro foi definido como 500, já o e é o valor da distância à linha mais próxima.

$$e = 1 - \frac{c^2}{c^2 + e^2} \quad \textbf{(1)}$$

Na Figura 4.19 pode ver-se a Equação (1) representada por uma linha contínua, enquanto a linha a tracejado representa uma equação em função do erro ao quadrado. Por análise do gráfico pode concluir-se que para valores de $e < c$ a linha a tracejado aproxima-se do gráfico da linha contínua. [**15**]

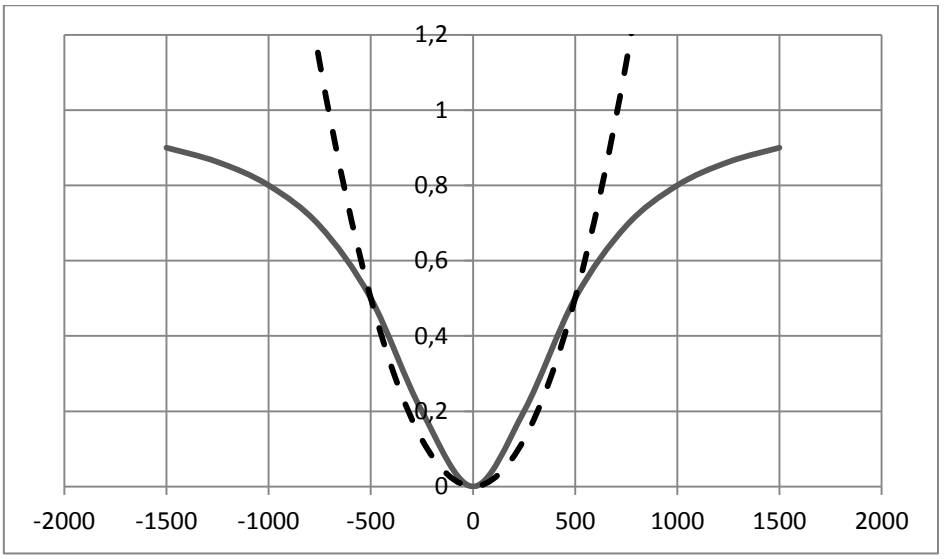


Figura 4.19 - Gráfico da Equação (1)

Para o cálculo da posição local, em vez de se percorrer o campo todo, utiliza-se a posição anteriormente detectada (em dm) e analisa-se um quadrado de 11 dm x 11 dm, em que o ponto central do quadrado é a posição calculada na *frame* anterior, depois o processo é igual ao explicado para a posição global. A Figura 4.20 pretende retratar as transições a percorrer o quadrado vermelho, de forma a coincidirem o mais possível com o modelo do campo, este método é utilizado para o cálculo da posição global.

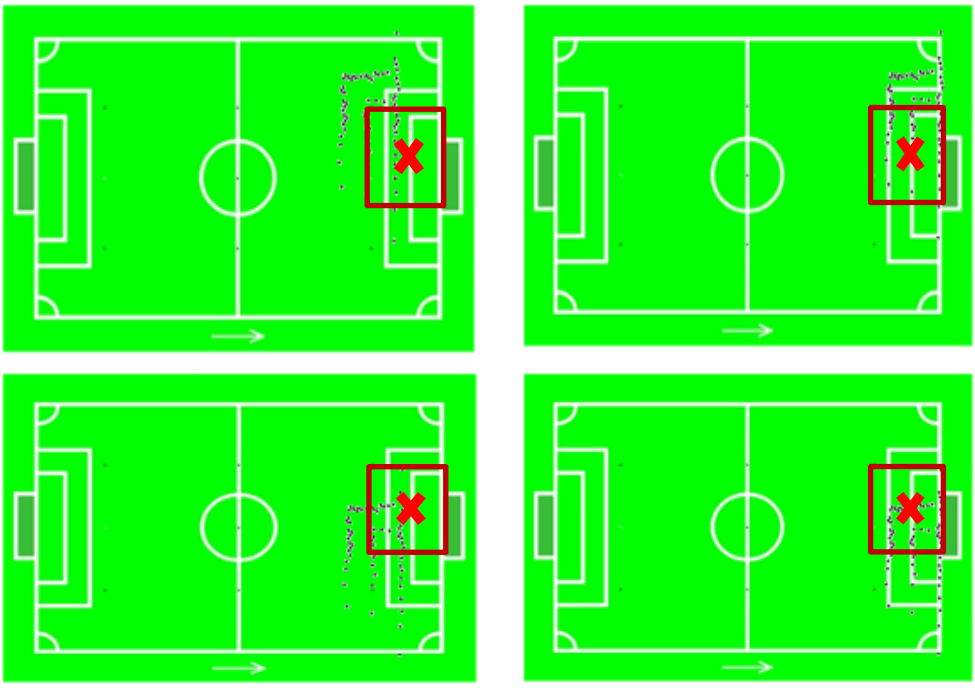


Figura 4.20 – Varrimento para o cálculo da posição local

4.5. Comunicação com o monitor

Depois de se ter efectuado a localização do robô é necessário informar o monitor acerca da localização do robô bem como de outras entidades. Para informar o monitor é utilizado rede sem fios, sendo que os robôs utilizam um AP para poder comunicar com o monitor. O monitor é uma das peças fundamentais de uma equipa de futebol robótico. Tem um conjunto de ferramentas que permite controlar e monitorizar o estado dos robôs durante o jogo, além de permitir a criação e selecção de tácticas para os robôs. O monitor tem desenhado um campo onde cada robô é projectado de acordo com a posição e orientação obtidos a partir do algoritmo de localização. Por tudo isto, o monitor é considerado um elemento chave numa equipa de futebol robótico, quer no controlo e monitorização dos robôs durante o jogo, quer na criação de tácticas para um conjunto de situações. O monitor é um computador que está fora do campo e funciona como treinador, uma vez que tal como um treinador humano também o monitor dá instruções sobre tarefas a executar em determinados momentos, ou qual a melhor táctica a aplicar. [**25**]

A Figura 4.21 representa a estrutura de dados utilizada para comunicar com o monitor, isto é, representa os dados que são enviados pelo robô para o monitor. O campo "LarBd" é o cabeçalho da trama e serve para o monitor validar a trama; os campos "X_{ROBOT}"e "Y_{ROBOT}" representam a posição do robô; o campo "θ_{ROBOT}" representa a orientação do robô; os campos "X_{BALL}"e "Y_{BALL}" representam a posição da bola; o campo "V_{BALL}" representa a velocidade da bola; o campo "DIR_{BALL}" representa a direcção que a bola tem; o campo "U_{BAT}" representa a tensão da bateria; o campo "ACT_{BEH}" representa o papel que o robô tem no momento, isto é, se o robô é guarda-redes, defesa, médio ou avançado.

LarBd	X_{ROBOT}	Y_{ROBOT}	θ_{ROBOT}	X_{BALL}	Y_{BALL}	V_{BALL}	DIR_{BALL}	U_{BAT}	ACT_{BEH}

Figura 4.21 - Estrutura de dados utilizada para comunicação

A Figura 4.22 representa a topologia de comunicação que a MINHO TEAM implementou para troca de dados entre o monitor e os robôs.

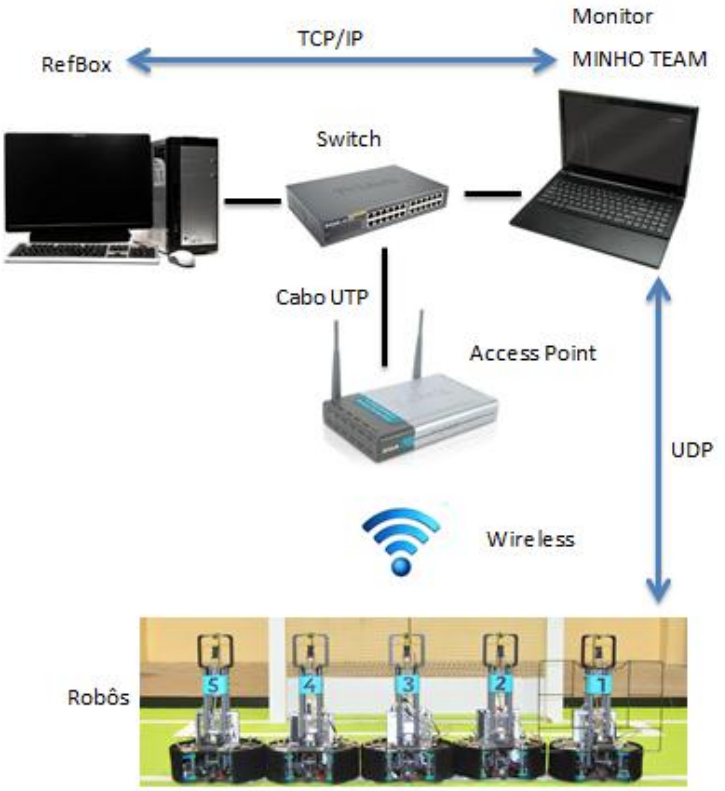


Figura 4.22 - Comunicação entre robôs e monitor

CAPÍTULO 5

5. RESULTADOS

Neste quinto capítulo são apresentados os resultados obtidos com a implementação dos algoritmos desenvolvidos. Estes algoritmos foram desenvolvidos utilizando a distribuição Ubuntu 10.10 do Linux, e implementados em C++ na plataforma QT Creator da Nokia [**26**].

5.1. Detecção de transições

5.1.1. Transições axiais

Inicialmente o método de detecção de transições através de pesquisa axial não era contínuo, o que fazia com que as transições detectadas não fossem no sítio da transição como se pode ver na Figura 5.1. Na Figura 5.1 é possível ver a vermelho os pontos que se são analisados e a azul as transições detectadas.

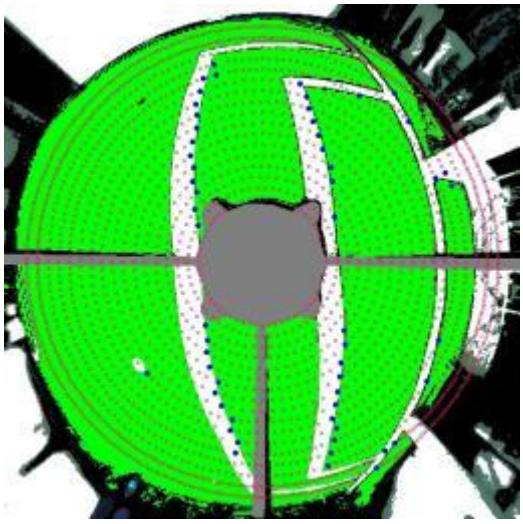


Figura 5.1 - Primeira aproximação para pesquisa axial

Este desfasamento nas transições detectadas é susceptível de introduzir erros quer no cálculo da orientação, quer no cálculo da posição, o que se torna um factor que diminui a robustez da localização. Optou-se então por fazer uma pesquisa contínua, isto é, os pixéis a analisar estão juntos. Esta alteração fez com que as transições detectadas sejam na transição verde-branco ou vice-versa conforme o tipo de transição pretendido. Na Figura 5.2 pode ver-se o exemplo de uma detecção de transições verde-branco.

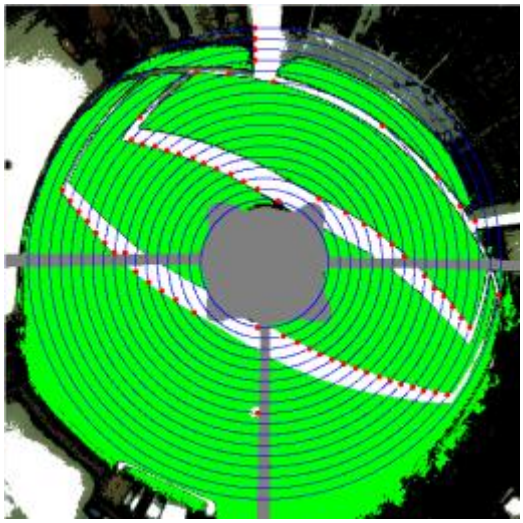


Figura 5.2 - Nova aproximação para pesquisa axial

Tal como na Figura 5.1, na Figura 5.2 é visível a azul os pontos a analisar e a vermelho as transições detectadas. Nesta abordagem, como já foi referido, a linha azul que são os pixéis a analisar é uma linha contínua, fazendo com que os erros sejam menores que na primeira aproximação. O facto de os erros serem menores contribui para um melhor cálculo da orientação, bem como um melhor cálculo de posição, tornando o algoritmo mais robusto.

5.1.2. Transições radiais

A Figura 5.3 representa os resultados obtidos para a detecção de transições através de pesquisa radial, pode ver-se a vermelho os pontos a analisar e a azul as transições detectadas. É possível também ver na Figura 5.3 que as transições detectadas estão no sítio onde ocorre a transição o que faz com que haja poucos erros e contribua para um melhor cálculo da localização do robô.

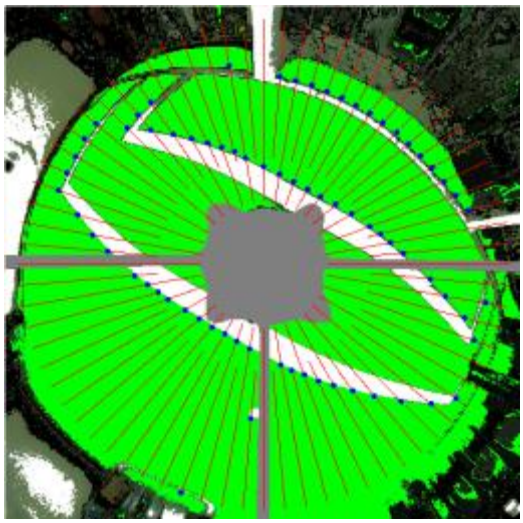


Figura 5.3 - Pesquisa radial

5.1.3. Conclusões

Para um algoritmo mais robusto optou-se por utilizar os dois tipos de pesquisa (radial e axial). O algoritmo de detecção de transições em vez de detectar só um tipo de transições (verde-branco ou branco-verde) detecta os dois tipos de transição.

Na Figura 5.4 é possível ver todas as transições detectadas, isto é, transições verde-branco, branco-verde, quer para a pesquisa radial, quer para a pesquisa axial.

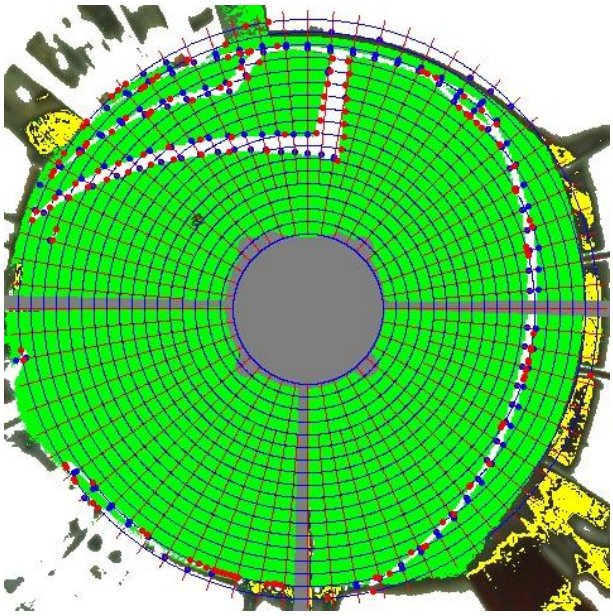


Figura 5.4 - Transições detectadas

5.2. Orientação do robô

O algoritmo antes de ser utilizado no *RoboCup* foi testado no campo do ISEP, onde foi possível tirar algumas conclusões acerca da sua robustez, visto que o campo tem as dimensões oficiais do campo de futebol robótico, ou seja, (18 m x 12 m). Na Figura 5.5, é possível ver algumas posições que foram utilizadas para efectuar testes de orientação do robô.

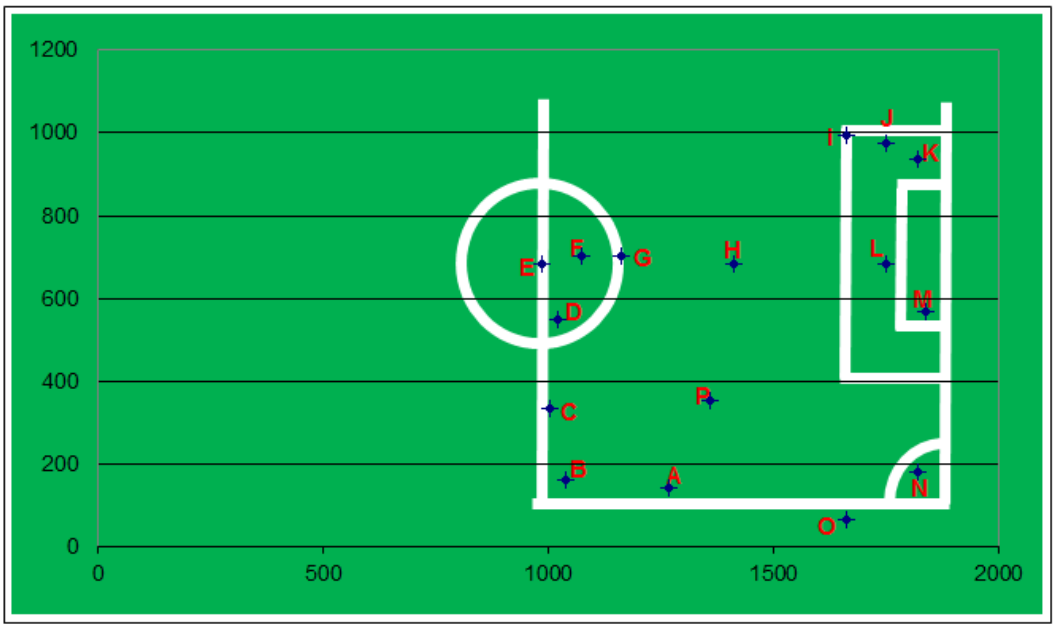


Figura 5.5 - Posições de teste utilizadas

As figuras seguintes (Figura 5.6, Figura 5.7, Figura 5.8, Figura 5.9, Figura 5.10, Figura 5.11 e Figura 5.12) mostram histogramas de alguns pontos analisados. Estes histogramas foram construídos em Excel, com base nos máximos de cada histograma criado no cálculo da orientação.

Ponto A

Na Figura 5.6 é possível perceber que a linha cor-de-rosa representa a linha lateral do campo, isto porque não existe muita variação. Pode concluir-se que neste caso o histograma horizontal será o melhor histograma para a escolha do ângulo, uma vez que tem uma amplitude considerável, sendo por isso o resultado do ângulo fiável.

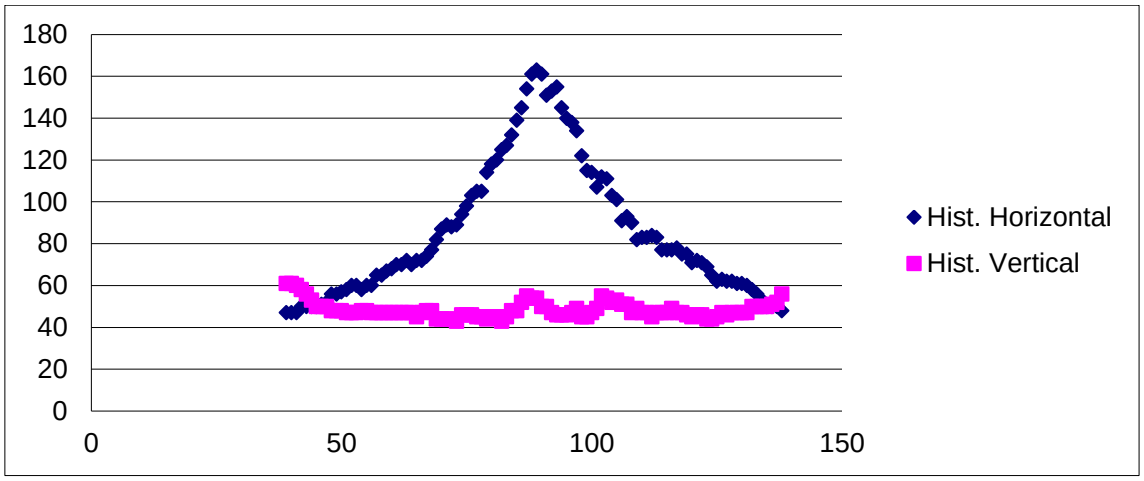


Figura 5.6 - Histograma do Ponto A

Ponto B

Na Figura 5.7, não é fácil fazer uma interligação entre os histogramas e as linhas do campo, no entanto é possível concluir que ambos os histogramas têm uma amplitude considerável, com o máximo aproximadamente no mesmo ponto. O ângulo escolhido é o devolvido pelo histograma horizontal, sendo que o resultado é fiável.

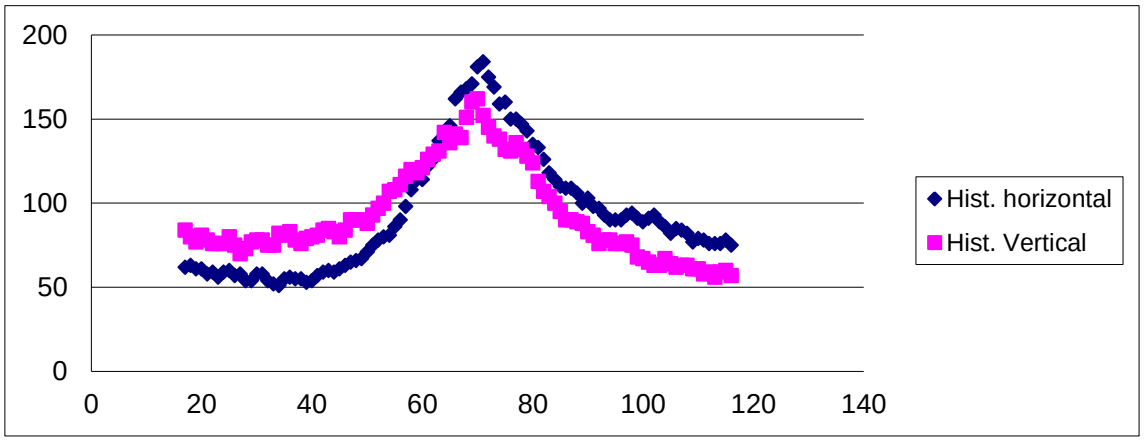


Figura 5.7 - Histograma do Ponto B

Ponto E

Na Figura 5.8, é possível perceber que a linha cor-de-rosa representa a linha do meio-campo, enquanto a linha azul representa o círculo central. Pode concluir-se que neste caso o histograma vertical será o melhor histograma para a escolha do ângulo,

uma vez que tem uma amplitude considerável, sendo por isso o resultado do ângulo fiável.

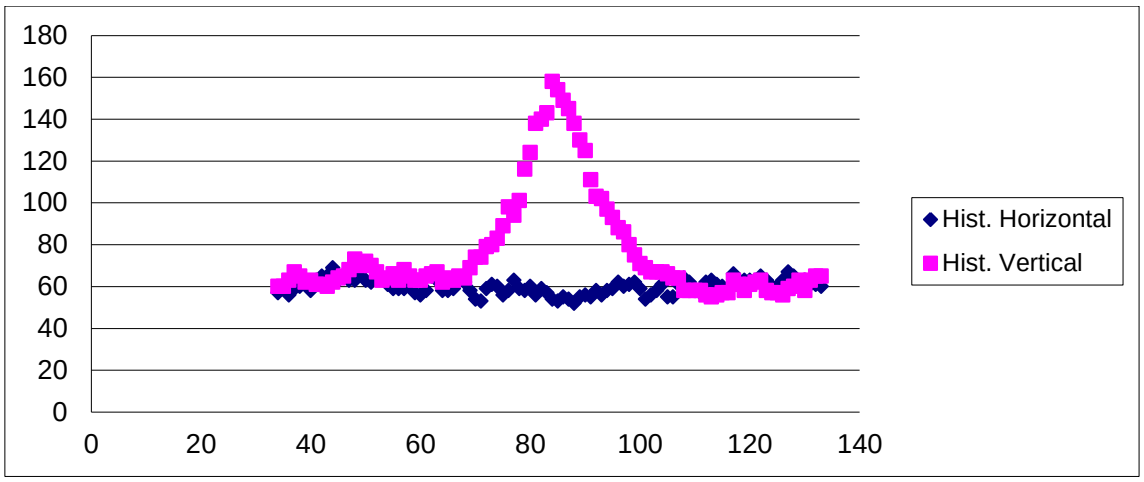


Figura 5.8 - Histograma do Ponto E

Ponto H

Na Figura 5.9, não é fácil fazer uma interligação entre os histogramas e as linhas do campo, neste caso é possível perceber que o histograma vertical será o melhor histograma para a escolha do ângulo. No entanto, o ângulo pode não ser o mais correcto, uma vez que a amplitude do histograma é baixa, sendo por isso aconselhado o uso de outros meios para ajudar a definir a orientação do robô, como por exemplo, uma bússola electrónica.

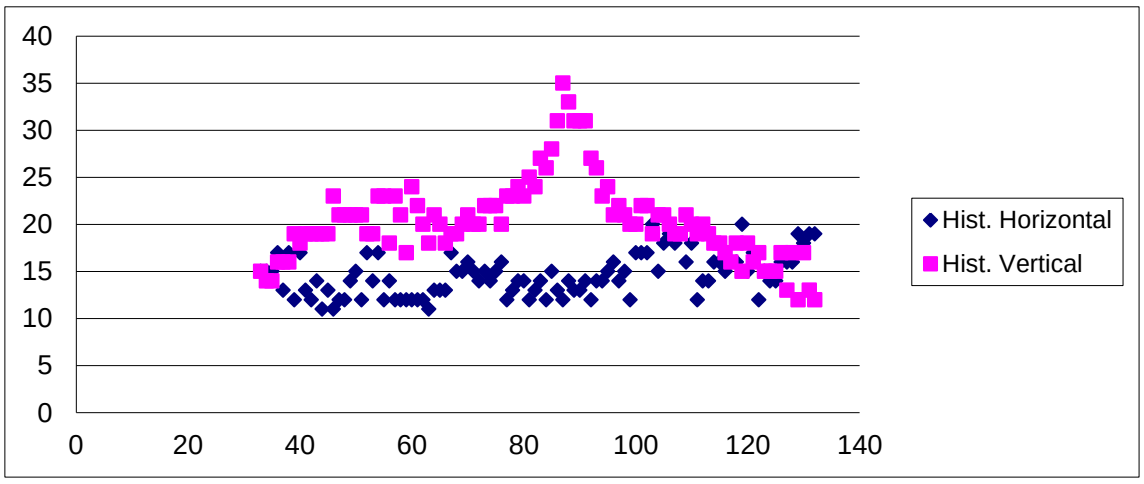


Figura 5.9 - Histograma do Ponto H

Ponto L

Na Figura 5.10, não é fácil fazer uma interligação entre os histogramas e as linhas do campo, contudo, é possível concluir que o histograma vertical tem uma amplitude considerável, sendo por isso o ângulo fiável.

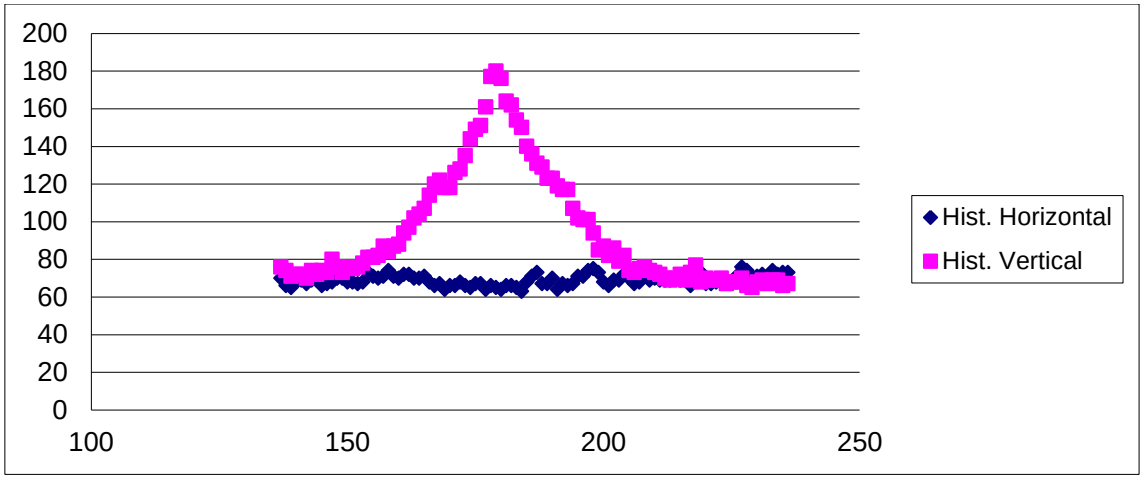


Figura 5.10 - Histograma do Ponto L

Ponto N

Na Figura 5.11, não é fácil fazer uma interligação entre os histogramas e as linhas do campo, no entanto é possível concluir que ambos os histogramas têm uma amplitude considerável. Como a amplitude histograma vertical é mais expressiva, é escolhido o ângulo devolvido pelo histograma vertical, este ângulo é também fiável.

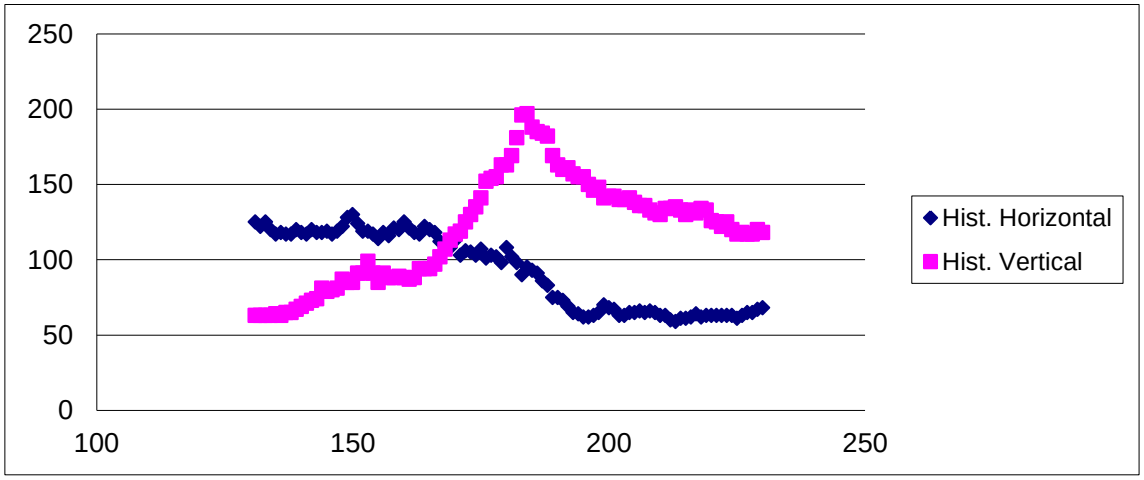


Figura 5.11 - Histograma do Ponto N

Ponto P

Na Figura 5.12 é impossível fazer uma interligação entre os histogramas e as linhas do campo, sendo possível concluir que ambos os histogramas não são adequados para o cálculo da orientação do robô devido à sua baixa amplitude. Neste caso é imprescindível o uso de outros meios para obter a orientação do robô.

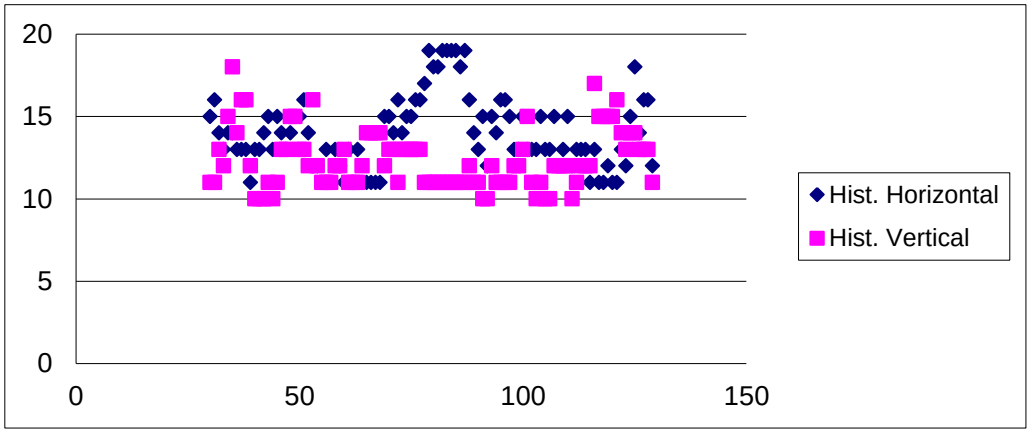


Figura 5.12 - Histograma do Ponto P

A Tabela 5.1 é uma síntese das posições analisadas, na qual se apresenta o ângulo dado pelo histograma horizontal (ângulo horizontal), o ângulo dado pelo histograma vertical (ângulo vertical) e qual o ângulo que o algoritmo escolhe. É também apresentado o número de transições detectadas, bem como o grau de fiabilidade do histograma nas posições analisadas.

Tabela 5.1 - Grau de fiabilidade do ângulo dos pontos analisados

Posição no Campo	Ângulo horizontal	Ângulo vertical	Ângulo escolhido	Número de transições	Fiabilidade do ângulo
A	89	39	89	105	Muito Boa
B	71	70	71	186	Muito Boa
E	44	84	84	233	Muito Boa
H	103	87	87	60	Aceitável
L	227	179	179	256	Muito Boa
N	150	184	184	203	Boa
P	79	35	79	41	Muito Fraca

Através da análise da Tabela 5.1 é possível concluir que o número de transições detectadas está muito relacionado com o nível de fiabilidade. É possível verificar que no caso dos pontos H e P, o número de transições é baixo e o grau de fiabilidade não é o desejado. Já nas restantes posições analisadas (A, B, E, L e N) o número de transições detectadas é alto, o que faz com que o grau de fiabilidade seja o pretendido.

Apesar da melhoria introduzida com a alteração na detecção de transições através da pesquisa axial, a robustez do algoritmo no cálculo da orientação do robô não é a desejável, pois há pontos no campo em que o número de transições detectadas é baixo, o que faz com que a orientação detectada possa não ser a mais correcta.

O algoritmo desenvolvido para o cálculo da orientação do robô é bastante rápido. Este algoritmo foi testado em dois computadores, um portátil Tsunami Replacer 655T com um processador Intel Core 2 Duo T5600 a 1,83 GHz e 1 GB DDR2 de RAM; o outro computador é o utilizado durante os jogos, possui uma motherboard mini-itx MB890F, esta motherboard está dotada com um processador Intel Pentium M a 1.6 GHz e com 512 MB DDR de RAM.

Na Tabela 5.2 é possível ver os tempos de computação do algoritmo no cálculo da orientação com os dois computadores diferentes. Estes valores foram tirados fazendo a média de 10 leituras. É de salientar que os valores obtidos, foram adquiridos no campo instalado no LAR.

Tabela 5.2 - Comparativo de tempos de computação da orientação

Computador	**Tempo de computação (ms)**
Portátil Tsunami (Intel Core 2 Duo T5600 a 1,83 GHz)	1,06
Motherboard MB890F (Intel Pentium M a 1.6 GHz)	1,25

Este algoritmo foi também utilizado no *RoboCup* onde foi possível concluir que este não era suficientemente robusto, apesar dos testes realizados no ISEP apontarem

em sentido contrário. A falta de robustez observada no *RoboCup* prendeu- -se essencialmente com o facto de não ser possível utilizar a bússola para ajudar no cálculo da orientação do robô. Não foi possível utilizar a bússola, porque o campo destinado aos jogos da equipa MINHO TEAM era atravessado por um campo magnético extremamente elevado. Este campo magnético era sentido essencialmente numa linha que dividia o campo em duas partes iguais e que unia as duas balizas, além disso fazia com que a bússola nessa zona variasse cerca de 180° em relação ao ângulo que o robô efectivamente tinha.

5.3. Posição do robô

Como se pôde verificar com os resultados do algoritmo de orientação, existem pontos no campo que implicam o uso de meios complementares à visão, para o cálculo da orientação. Como o cálculo da posição depende do cálculo da orientação, foi adicionada uma bússola para corrigir a orientação dada pela visão, de forma a melhorar a qualidade dos testes de posição.

O algoritmo de posição, assim como o de orientação também foi testado no campo do ISEP pelas razões já antes apresentadas. Nas figuras, que são a seguir apresentadas, é possível ver para além da posição do robô as transições de linha detectadas, assim como a orientação do robô na respectiva posição. Antes de o algoritmo ser testado no ISEP, foi testado no LAR. O campo que está instalado no LAR é um campo bastante pequeno, sendo que apenas existe a metade do lado direito da Figura 5.13, mas mesmo esta metade do campo não está com as dimensões reais, só assim é possível que na posição em que o robô se encontra na Figura 5.13 este consiga detectar transições quer no círculo central quer na linha de pequena área.

Figura 5.13 - Teste de posição efectuado no LAR

Os resultados que a seguir se apresentam, já são resultados de todo o algoritmo de localização, isto é, posição mais orientação, além disso estes resultados foram obtidos no ISEP.

Na Figura 5.14, que representa um dos testes efectuados, percebe-se que o algoritmo foi capaz de descobrir a orientação do robô, bem como calcular a sua posição. Esta conclusão prende-se com o facto de as transições detectadas estarem com um erro mínimo em relação ao modelo do campo como se pode ver na Figura 5.14.

Figura 5.14 - Teste de localização efectuado no ISEP

Na Figura 5.15 é possível ver que existem transições falsas, isto é transições que de facto não existem. Estas transições são as que estão assinaladas e devem-se ao excesso de luminosidade. Transições falsas são bastante susceptíveis de provocar erros quer de posicionamento quer de orientação, o que não se veio a verificar, pois o algoritmo conseguiu calcular a localização do robô como se pode perceber pela Figura 5.15.

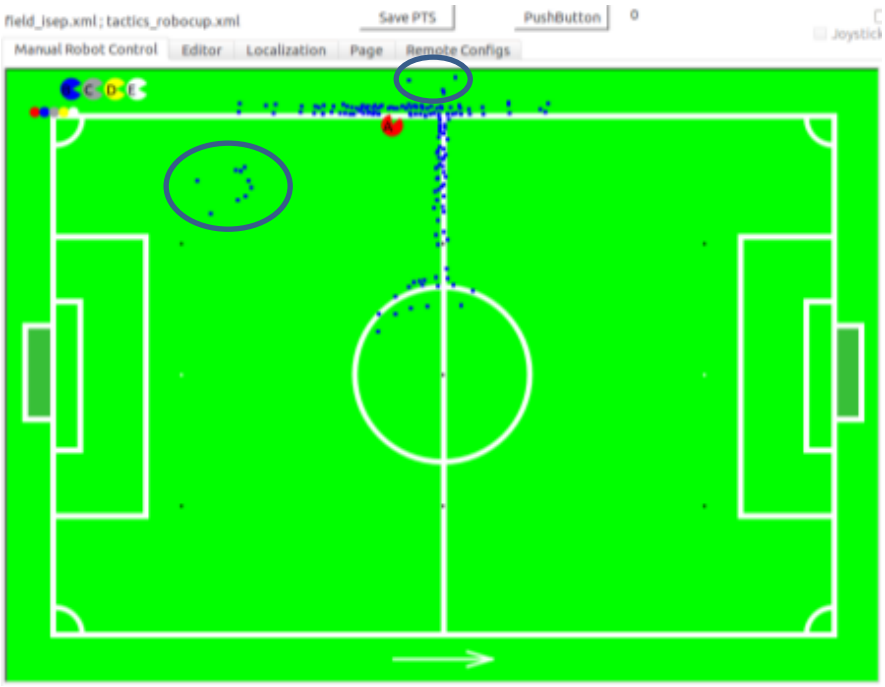


Figura 5.15 - Teste de localização efectuado no ISEP

Na Figura 5.16, tal como na Figura 5.15 é possível ver que existem transições falsas que estão também assinaladas, mas o algoritmo conseguiu, mais uma vez, calcular a localização do robô como se pode perceber pela Figura 5.16.

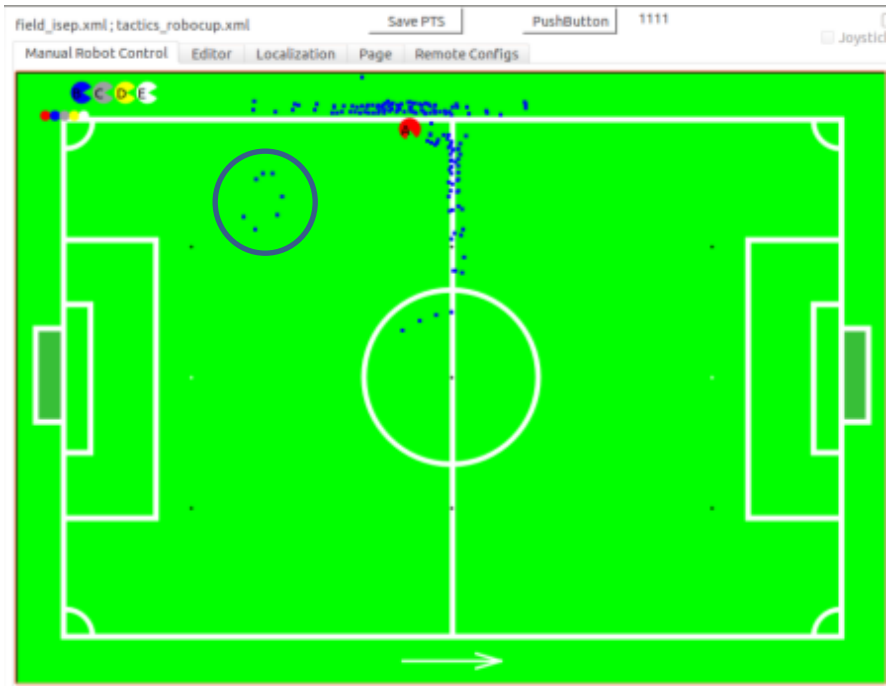


Figura 5.16 - Teste de localização efectuado no ISEP

Os desvios de transições junto do círculo central verificados na Figura 5.16 podem ser derivados de uma má calibração do espelho, isto é, a distância entre a câmara e o espelho pode ter sido alterada. Uma má calibração do espelho implica uma leitura errada das distâncias, o que faz com que os resultados obtidos não sejam os mais correctos.

A posição do robô depende bastante da sua orientação, uma vez que a posição do robô é calculada com base na sua orientação, assim sendo a robustez do cálculo da posição depende directamente da robustez do cálculo da orientação.

Com os testes realizados antes do *RoboCup* foi possível verificar que o algoritmo desenvolvido para o cálculo da localização funciona e até parece ser robusto, o que não se veio depois a verificar durante o *RoboCup*, mais uma vez, devido aos campos magnéticos presentes debaixo do campo.

O algoritmo desenvolvido para o cálculo da posição do robô, assim como o algoritmo de orientação também se apresenta como um algoritmo bastante rápido. Este algoritmo foi testado com os mesmos dois computadores utilizados para os testes de orientação, sendo que os valores dos testes são apresentados na Tabela 5.3. Os valores foram tirados fazendo a média de 10 leituras. Importa referir que os valores obtidos foram adquiridos no campo do LAR. Ao fazer as leituras no LAR, o tempo de processamento necessário para o cálculo da posição global é mais baixo, em relação a um campo com as dimensões reais. Já o tempo de processamento para o cálculo da posição local não é afectado.

Tabela 5.3 - Comparativo de tempos de computação da posição

Computador	**Posição Global (ms)**	**Posição Local (ms)**
Portátil Tsunami (Intel Core 2 Duo T5600 a 1,83 GHz)	153,84	2,44
Motherboard MB890F (Intel Pentium M a 1.6 GHz)	149,85	2,61

Com as tabelas anteriores (Tabela 5.2 e Tabela 5.3), é possível verificar que os tempos de computação variam de acordo com o computador, mais precisamente de acordo com o processador, que se está a utilizar, conseguindo-se mesmo assim tempos de computação bons quando se utiliza o computador de jogo. Nas duas tabelas anteriores (Tabela 5.2 e Tabela 5.3) foi possível ver os tempos de computação para as duas situações distintas, isto é, orientação e posição. A Tabela 5.4 representa o tempo de computação de todo o algoritmo de localização, pode ver-se que estes tempos são aproximadamente a soma dos tempos das duas tabelas anteriores. Estes valores foram tirados fazendo a média de 10 leituras. Importa referir que os valores obtidos tal como os anteriores, também foram adquiridos no campo do LAR.

Tabela 5.4 - Comparativo de tempos de computação da localização

Computador	**Localização Global (ms)**	**Localização Local (ms)**
Portátil Tsunami (Intel Core 2 Duo T5600 a 1,83 GHz)	150,17	3,48
Motherboard MB890F (Intel Pentium M a 1.6 GHz)	154,15	3,82

Como se pode ver pela análise da Tabela 5.4 os tempos de computação para o robô se localizar é baixo (menos de 4 ms), isto faz com que o algoritmo desenvolvido possa ser utilizado noutro tipo de aplicações que também exijam tempos de computação reduzidos.

5.4. Conclusões

Antes da participação no *RoboCup* foram realizados testes, cujos resultados já foram apresentados. Estes resultados faziam prever que o algoritmo era robusto, uma vez que todos os testes realizados apontavam nesse sentido. Porém importa relembrar que nos testes realizados utilizou-se uma bússola electrónica.

Com a participação no *RoboCup* foi possível concluir que o algoritmo não é suficientemente robusto, uma vez que este não consegue ser imune a campos

magnéticos que facilmente fazem variar a bússola, fazendo com que o robô perca a sua orientação e consequentemente perca também a sua localização.

CAPÍTULO 6

6. CONCLUSÕES E TRABALHO FUTURO

Com este trabalho é possível concluir que o algoritmo desenvolvido é bastante rápido, necessitando de cerca de 155 ms para se localizar globalmente e menos de 4 ms para calcular a sua localização local. Como o tempo de computação gasto pelo algoritmo é reduzido, é possível dentro do número mínimo de *frames* pretendido (entre 30 a 40 fps), desenvolver algoritmos que visem melhorar a robustez da localização, bem como de forma a melhorar o seu comportamento.

Durante a execução deste trabalho foram surgindo algumas ideias alternativas, com potencial aplicação, sugerindo-se que sejam testadas como trabalho futuro. A solução apresentada provou ser bastante robusta, excepto na presença de campos magnéticos, os quais influenciam bastante o resultado da localização. Nestes casos, a solução proposta passa pelo cálculo da direcção com recurso à visão por computador.

No decorrer desta dissertação foram sendo apresentados aspectos a melhorar, tendo este capítulo como objectivo apresentar sugestões para as melhorias que se podem realizar.

Nas Conclusões do capítulo 3, foi dito que havia pontos na estrutura do robô que poderiam ser melhorados. Esses pontos são essencialmente a estrutura da torre que suporta a cabeça, pois a actual estrutura ocupa pouca área, sendo fácil para o adversário fazer passar a bola a meia altura. A estrutura proposta para a torre consiste numa forma de pirâmide, esta seria fixa aos parafusos que unem as três bases do robô. A estrutura em pirâmide permitiria utilizar um portátil apoiado na base superior do robô, bem como uma maior facilidade em separar a torre das bases do robô.

As sugestões para melhorar o algoritmo de localização do robô podem fazer com que seja necessário utilizar mais *hardware*. As soluções propostas passam pela utilização de odometria e utilização de sensores inerciais.

Com o uso de odometria será necessário fazer fusão sensorial entre o algoritmo de localização e os valores de odometria, mas para isso é necessário que a placa de controlo dos motores devolva os valores lidos pelos encoders. O uso de sistemas inerciais, ou seja, giroscópios e acelerómetros, seria um complemento à odometria. O giroscópio serviria para ajudar na orientação, pois no caso em que a bússola sofre interferências de campos electromagnéticos o giroscópio não altera o seu valor, logo informaria o sistema que não houve alteração na orientação do robô. O acelerómetro seria para o caso em que o robô derrape; neste caso os encoders devolvem um valor errado de deslocamento, sendo que o acelerómetro ajuda a corrigir esse problema. Aconselha-se ainda um Filtro de Kalman para suavizar os dados quer dos encoders quer do sistema inercial.

Também é necessário que o algoritmo de localização seja capaz de perceber que o robô está perdido, se acontecer, e que o próprio algoritmo decidisse fazer o cálculo da localização global do robô.

O cálculo da localização global deveria ser feito pelo menos 3 ou 4 vezes com o robô parado, de modo a diminuir a probabilidade do algoritmo por algum motivo calcular uma localização errada do robô.

O algoritmo de localização deve ser capaz de calcular o grau de fiabilidade da posição calculada.

É importante também que o robô seja capaz de identificar os outros robôs como obstáculos e que envie para o monitor a posição dos mesmos, de modo a que o monitor possa definir qual a melhor estratégia a utilizar conforme a disposição dos intervenientes no campo.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] MINHO TEAM. RESULTS OF MINHO TEAM. [Online]. http://www.robotica.dei.uminho.pt/robocup/RES.htm

[2] Merriam-Webster. Merriam-Webster Dictionary. [Online]. http://www.merriam-webster.com/dictionary/robot

[3] RoboCup Federation. RoboCup. [Online]. http://www.robocup.org/about-robocup/

[4] RoboCup Federation. RoboCup. [Online]. http://www.robocup.org/robocup-rescue/

[5] RoboCup Federation. RoboCup. [Online]. http://www.robocup.org/robocup-home/

[6] RoboCup Federation. RoboCup. [Online]. http://www.robocup.org/robocup-junior/

[7] RoboCup Federation. RoboCup. [Online]. http://www.robocup.org/robocup-soccer/

[8] MSL Technical Committee, "Middle Size Robot League Rules and Regulations for 2011," RoboCup, 2010.

[9] IST. Robótica 2011. [Online]. http://robotica2011.ist.utl.pt/

[10] IST. Robótica 2011. [Online]. http://robotica2011.ist.utl.pt/pt/competicoes/

[11] (2011, Julho) Wikipédia. [Online]. http://pt.wikipedia.org/wiki/M%C3%A9todo_de_Monte_Carlo

[12] Valguima Victoria Viana Odakura, "Localização de Markov para Multirrobôs cooperativos," São Paulo, Tese de Doutoramento 2007.

[13] Universität Freiburg. Machine Learning Lab. [Online]. http://ml.informatik.uni-freiburg.de/research/tribots

[14] Universität Freiburg. Machine Learning Lab. [Online]. http://ml.informatik.uni-freiburg.de/research/tribots/selflocalization

[15] Martin Lauer, Sascha Lange, and Martin Riedmiller, "Calculating the Perfect Match: an Efficient and Accurate Approach for Robot Self-Localization," in *RoboCup-2005: Robot Soccer World Cup IX*, 2005, p. 12.

[16] Universität Freiburg. Machine Learning Lab. [Online]. http://ml.informatik.uni-freiburg.de/research/tribots/vision

[17] CAMBADA. CAMBADA Robotic Soccer. [Online]. http://www.ieeta.pt/atri/cambada/index.htm

[18] António J. R. Neves, José Luís Azevedo, Bernardo Cunha, Nuno Lau, João Silva, Frederico Santos, Gustavo Corrente, Daniel A. Martins, Nuno Figueiredo, Artur Pereira, Luís Almeida, Luís Seabra Lopes, Armando J. Pinho, João Rodrigues e Paulo Pedreiras, "CAMBADA soccer team: from robot architecture to multiagent coordination," in *Robot Soccer*.: Vladan Papic, 2010, pp. 19-45.

[19] Gustavo Abdul da Fonseca Ussemane Pires Corrente, "Arquitectura de controlo/coordenação de uma equipa de futebol robótico," Universidade de Aveiro, Tese de mestrado 2008.

[20] Min Huang, Xinsheng Ge, Shiyong Hui, Xueyan Wang, Song Chen, Xinxin Xu, Wanjie Zhang,Ye Lu, Xiaoming Liu, Liang Zhao, MiaoWang, Zhe Zhu, Chenyu Wang, Bin Huang, Libo Ma, Biao Qin, Fangyu Zhou and Changyun Wang, "Water Team Description Paper 2011," Beijing Information Science & Technology University, China, 2011.

[21] RoboCup. Results World Championships 2011: Istanbul, Turkey. [Online]. http://wiki.robocup.org/wiki/Results_World_Championships_2011:_Istanbul,_Turkey#Results

[22] Tomé Pereira da Silva, "Desenvolvimento de Plataforma Móvel para Futebol Robótico," Universidade do Minho, Tese de Mestrado 2010.

[23] scvalex. (2007, Maio) DZone Snippets. [Online]. http://snippets.dzone.com/posts/show/3972

[24] Honeywell International Inc., "Digital Compass Solution HMC6352," Datasheet 2006.

[25] Fernando Ribeiro, Gil Lopes, João Costa, João Pedro Rodrigues, Bruno Pereira, João Silva, Sérgio Silva, Paulo Ribeiro, Paulo Trigueiros, "Minho MSL - A New Generation of soccer robots," TDP 2011.

[26] Nokia Corporation. Qt. [Online]. http://qt.nokia.com/products/developer-tools/