Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2015

Fred Gomes

Monitorização de objetos em 3-D com base em múltiplas câmaras

3-D objects monitoring based on multiple cameras

Fred Gomes

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Automação Industrial, realizada sob a orientação científica do Professor Doutor Paulo Miguel de Jesus Dias e do Professor Doutor António José Ribeiro Neves, Professores Auxiliares do Departamento de Eletrónica,Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor Armando José Formoso de Pinho
Professor Associado c/ Agregação da Universidade de Aveiro

vogais / examiners committee

Doutor Miguel Armando Riem de Oliveira
Bolseiro Pós-Doutoramento, Inesc - Porto (Arguente)

Professor Doutor Paulo Miguel de Jesus Dias
Professor Auxiliar da Universidade de Aveiro (Orientador)

agradecimentos / acknowledgements

Ao meu orientador e co-orientador, Professores Paulo Dias e António Neves, por toda a orientação e motivação ao longo deste último ano. Agradeço também à equipa CAMBADA, pela oportunidade que me foi dada de trabalhar neste projeto. Aos meus pais, por toda a confiança depositada em mim e no meu trabalho em todas as fases da minha vida. Deixo também um especial agradecimento aos meus amigos e colegas de curso, por todo o apoio e companheirismo prestado durante o meu percurso académico. Por fim, queria agradecer a toda a minha família, que também fizeram parte desta fase da minha vida, festejando comigo todas as pequenas conquistas e orgulhando-se do meu percurso. Um muito obrigado a todos.

Palavras chave

Ground truth, visão por computador, futebol robótico, sistema de monitorização, calibração de câmaras, *OpenCV*

Resumo

No ano de 2014 foi concluído um novo edifício na Universidade de Aveiro, para o desenvolvimento de projetos em robótica. Neste edifício está localizado um campo de futebol robótico com dimensões oficiais, para o desenvolvimento do projeto CAMBADA.

O futebol robótico representa um banco de ensaio inovador e atraente para os mais recentes avanços em sistemas multiagente, inteligência artificial, perceção, navegação e *biped walking*. O principal elemento sensor de um robô de futebol deve ser o seu sistema de perceção, muitas das vezes baseado numa câmara digital, através da qual o robô analisa o mundo circundante. Até à presente data, a validação de um sistema de visão de um jogo de futebol pode apenas ser relacionado com a forma como o robô e os seus companheiros de equipa interpretam o ambiente à sua volta.

Esta dissertação tem como objetivo desenvolver um sistema de visão externo de monitorização que pode atuar como um sistema de *ground truth* para a validação dos objetos de interesse de um jogo de futebol robótico, principalmente os robôs e a bola. O sistema é constituído por duas a quatro câmaras, estrategicamente posicionadas no campo de futebol.

O trabalho dividiu-se em quatro partes fundamentais: a seleção do método mais adequado para a aplicação; o desenvolvimento de software para efetuar a calibração das câmaras RGB; o desenvolvimento de software para obter a posição 3-D de objetos e a validação dos resultados obtidos, em termos de precisão e fiabilidade.

Os resultados indicam que um sistema deste tipo parece adequado para ser utilizado como *ground truth* das posições da bola no campo, durante um jogo de futebol robótico.

Key words

Ground Truth, computer vision, robotic soccer, monitoring system, camera calibration, OpenCV

Abstract

A new building was concluded at the University of Aveiro for the development of projects in robotics, in 2014. In this building is located a robotic soccer field with official dimensions, for the development of the CAMBADA project.

Robotic soccer represents an innovative and appealing test bed for the most recent advances in multi-agent systems, artificial intelligence, perception, navigation and biped walking. The main sensorial element of a soccer robot must be its perception system, most of the times based on a digital camera, through which the robot analyses the surrounding world and performs accordingly. Up to this date, the validation of the vision system of a soccer robots can only be related to the way the robot and its team mates interpret the surroundings, relative to their owns.

This Master's thesis aims the development of a external monitoring vision system that can act as a ground truth system for the validation of the objects of interest of a robotic soccer game, mainly robots and ball. The system is made of two to four digital cameras, strategically positioned above the soccer field.

The developed work was divided into four main parts: the selection of the most appropriate method for the application; the development of a software to perform the calibration of the RGB cameras; the development of a software to obtain the 3-D information of objects and the validation of the obtained results, in terms of accuracy and reliability.

The results prove that such a system can indeed be used as a provider for ground truth ball positions on the field during a robotic soccer game.

Conteúdo

Capítulo 1

Introdução

Sistemas para obter informação 3-D têm sido alvo de investigação e desenvolvimento ao longo dos últimos anos, esta informação pode ser obtida a partir de múltiplas câmaras RGB.

No ano de 2014 foi concluído um novo edifício na Universidade de Aveiro, para o desenvolvimento de projetos de robótica. Neste edifício está localizado um campo de futebol robótico com dimensões oficiais, na Figura 1.1 é apresentada uma fotografia do campo de futebol presente no laboratório, para o desenvolvimento do projeto CAMBADA.[1]

Figura 1.1: Fotografia do campo de futebol robótico existente no laboratório de robótica da Universidade de Aveiro.

Em aplicações de robótica, a informação 3-D é cada vez menos indispensável, permitindo aos robôs obterem uma melhor perceção do ambiente à sua volta e facilitando a sua tomada de decisões.

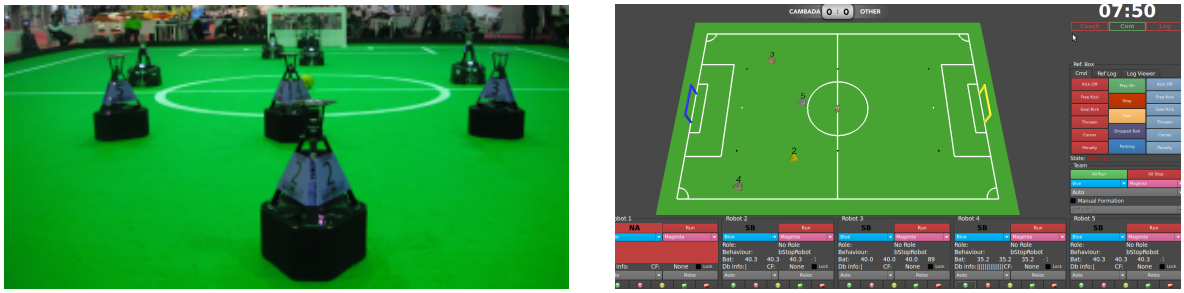
É bastante útil a existência de sistemas externos de *ground truth*,[2] que permitam efetuar a validação dos dados provenientes dos robôs e a realidade. Nesta dissertação pretende-se desenvolver um sistema de *ground truth*, que permita efetuar a monitorização de objetos em 3-D com base em múltiplas câmaras RGB. Aplicando o sistema no campo de futebol robótico

[1] Acrónimo de *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*. Esta equipa visa o desenvolvimento de robôs para a participação em torneios de futebol robótico, da designada liga média.

[2] Em robótica, este termo refere-se à existência de um valor de referência para comparar com a informação dos robôs.

existente no laboratório, permitindo efetuar a monitorização de um jogo de futebol robótico com alguma precisão.

Esta informação é essencial para se conseguir validar muitos dos algoritmos implementados nos robôs, dado que até agora muita dessa informação só pode ser obtida a partir dos próprios robôs, com os erros e problemas associados. Na Figura 1.2 é apresentado um *setup* dos robôs (esquerda) e o mundo como o robô o entende, representado numa *basestation* (direita).



(a) (b)

Figura 1.2: (a) *Setup* dos robôs durante um jogo de futebol. Visão do mundo como o robô o entende, representado na *basestation* [1].

1.1 Motivação e objetivos

A informação 3-D apresenta uma elevada importância em aplicações de robótica, permitindo aos robôs obterem informação 3-D do ambiente à sua volta. Aplicações como *bin picking*, *object tracking* e *product profiling* necessitam de sistemas capazes de obter informação 3-D [2].

A informação 3-D pode ser obtida recorrendo a métodos passivos ou ativos. Os métodos passivos, como a visão *stereo*, apenas dependem da ilumunição do meio ambiente, pois para que ocorra a formação da imagem em câmaras RGB é necessário haver iluminação. Já os métodos ativos, tais como *structured laser light* e *pattern-based lighting systems* , usam fontes de luminosidade externas para efetuar a reconstrução 3-D do meio ambiente [2].

O facto dos métodos ativos não dependerem da quantidade de luminosidade do meio ambiente não significa que seja obrigatoriamente uma vantagem, face aos métodos passivos, pois a luminosidade naturalmente existente no meio ambiente pode interferir com a fonte de luminosidade externa destes sistemas, interferindo com o bom funcionamento dos mesmos.

Pretende-se assim desenvolver um sistema capaz de obter a informação 3-D de objetos, aplicando o mesmo num jogo de futebol robótico. O projeto integra várias câmaras digitais para monitorizar o campo, sendo proposto utilizar quatro câmaras digitais, desenvolvendo-se assim um sistema do tipo passivo.

1.2 Estrutura

Esta dissertação encontra-se estruturada da seguinte forma:

- No Capítulo 2 é efetuado um levantamento acerca de sistemas existentes para efetuar o *tracking* de objetos.

Posteriormente, é efetuado um estudo dos diferentes métodos existentes para obter informação 3-D a partir de múltiplas imagens, obtidas com câmaras digitais RGB.

Por fim, são explorados os principais métodos utilizados para efetuar a deteção de objetos em imagens.

- No Capítulo 3 é explorado o princípio de funcionamento de uma câmara digital RGB, sendo esta um dos tipos de sensores mais utilizados em visão por computador. É apresentado o modelo das câmaras utilizadas, bem como as suas principais características. É também apresentada a interface para o *driver* desenvolvida, compatível com a biblioteca apresentada pelo fabricante das câmaras. Permitindo assim efetuar o controla das câmaras via software.

- No Capítulo 4 é apresentado o processo de calibração das câmaras. Para verificar a qualidade da calibração, diferentes métodos são aplicados. Estes são descritos ao longo deste capítulo.

- No Capítulo 5 é apresentada a aplicação desenvolvida para obter as coordenadas 3-D de objetos, com base em múltiplas câmaras RGB. É ainda efetuada a análise dos resultados obtidos.

- No capítulo 6 são integradas as múltiplas câmaras no sistema, métodos para obter uma maior precisão e fiabilidade nos dados obtidos são aqui descritos.

- No Capítulo 7 é efetuada uma análise crítica do sistema desenvolvido, efetuando-se uma avaliação do mesmo em termos de qualidade. São ainda descritas algumas ideias de trabalho futuro, tendo com vista a melhoria do trabalho desenvolvido.

Capítulo 2

Estado de arte

Esta dissertação tem como objetivo desenvolver um sistema de monitorização de um jogo de futebol robótico através de múltiplas câmaras RGB, desenvolvendo-se assim um sistema de *ground truth* para a validação dos algoritmos implementados nos robôs.

O problema consiste em encontrar a posição de um objeto no espaço, através da sua posição em duas (ou mais) imagens, obtidas por duas (ou mais) câmaras. Neste capítulo são apresentados sistemas disponíveis no mercado e são estudados diferentes métodos existentes para este propósito, assim como a seleção do método mais vantajoso para a aplicação.

Visto que se pretende monitorizar os objetos de interesse no campo, o primeiro passo num sistema deste tipo é a deteção de *features* nas imagens provenientes das diferentes câmaras. Neste capítulo são apresentadas diversas soluções aplicadas em visão por computador para a deteção de objetos de interesse em imagens.

2.1 Sistemas comerciais para efetuar o *tracking* de objetos

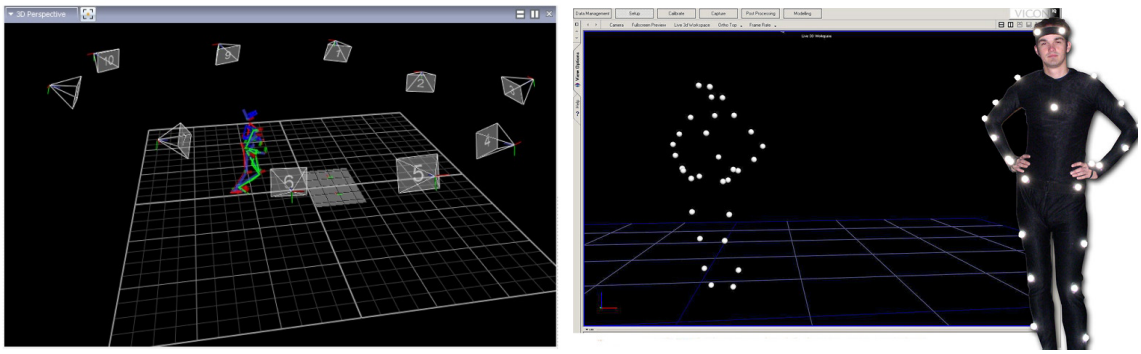
Existem alguns sistemas no mercado para efetuar o *tracking* de objetos. Estes são muito utilizados em um grande número de aplicações, tais como: aplicações militares, entretenimento, medicina, robótica, validação da visão de computador, entre outras. De seguida são abordados alguns destes sistemas.

2.1.1 Sistemas baseados em marcadores de infravermelhos

Sistemas como o *Vicon*, *Qualisys* e *Optitrack* são sistemas que efetuam o *tracking* de objetos baseando-se em marcadores de infravermelhos. Estes sistemas incorporam várias câmaras de infravermelhos com uma elevada resolução e marcadores. Os marcadores são colocados nos objetos que irão ser monitorizados, estes são pequenas esferas cobertas com um material retrorreflexivo.

Os marcadores são detetados pelas múltiplas câmaras de infravermelhos. Efetuando a triangulação entre as diferentes câmaras é determinado o posicionamento 3-D dos marcadores com bastante precisão, abaixo dos 20 µm [3].

O método de triangulação irá ser explicado com detalhe mais à frente nesta dissertação. Na Figura 2.1 é mostrado um exemplo do sistema *Vicon*, sendo que todos os mencionados anteriormente funcionam de forma semelhante a este.



(a) (b)

Figura 2.1: Sistema de captura de movimento 3-D. (a) O corpo de uma pessoa é coberto com vários marcadores retrorreflexivos que irão ser localizados através das múltiplas câmaras. Na figura é possível ver o *setup* destas [4]. (b) Visualização da posição 3-D dos marcadores retrorreflexivos, localizados pelo software [5].

2.1.2 Sistemas baseados em imagens

Sistemas como o *APAS*, *Simi Motion* ou *Kinovea*, permitem efetuar o *tracking* da forma humana tendo como base a sua silhueta. Ao contrário dos sistemas apresentados anteriormente, estes não utilizam tecnologia de infravermelhos, todavia funcionam de forma semelhante esses. O sistema *Simi Motion* funciona da seguinte forma [6]:

1. A pessoa é capturada por oito câmaras.
2. As silhuetas são extraídas (separação de pessoas e fundo).
3. Montagem de silhuetas para o modelo virtual 3-D para cada pessoa.
4. Montagem ideal de silhuetas com um modelo 3-D que permite a extração de posições comuns em 3-D e ângulos articulares.
5. Os resultados são posições comuns em 3-D e ângulos articulares (sistemas baseados nos sistemas de marcadores, mas sem o uso destes).

A silhueta é detetada tendo como base as articulações do corpo humano. De forma a facilitar este processo, marcadores adicionais podem ser colocados para aumentar a precisão. Na Figura 2.2 é mostrada uma imagem deste software.

Os sistemas referidos anteriormente são impraticáveis na aplicação pretendida, dado não ser possível colocar marcadores de infravermelhos na bola de futebol, ou mesmo para os sistemas baseados na silhueta do corpo humano, pois teriam de ser adaptados para efetuar o *tracking* de robôs.

2.2 3-D a partir de múltiplas câmaras RGB

Existem diversas técnicas para obter a posição 3-D de um objeto a partir de múltiplas câmaras digitais, que serão descritas de seguida.

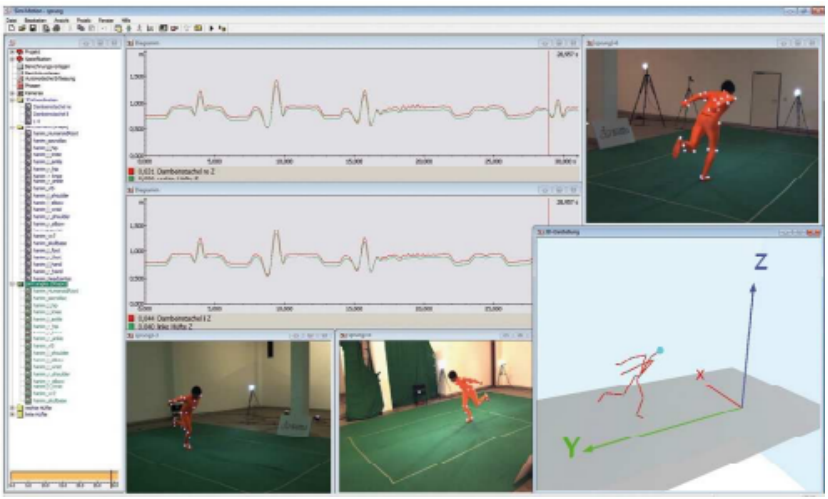


Figura 2.2: Processamento de dados com o *SIMI Motion* [6].

2.2.1 Visão *stereo*

A visão *stereo* é inspirada no funcionamento da visão humana. Os olhos humanos encontram-se horizontalmente espaçados por aproximadamente 60 mm e cada olho tem uma visão do mundo diferente. A linha de visão encontra um ponto no espaço que é projetado para a retina dos olhos. A diferença posicional da projeção dos pontos nas duas retinas é denominada de disparidade binocular.

A disparidade contribui de forma acentuada para o entendimento da informação de profundidade. Esta ideia surgiu no século XIX por *Charles Wheatstone* [7], que mostrou que o cérebro utiliza a disparidade horizontal para estimar a informação de profundidade. A disparidade binocular é definida pela diferença da projeção de um ponto nos dois olhos, como apresentado Figura 2.3.

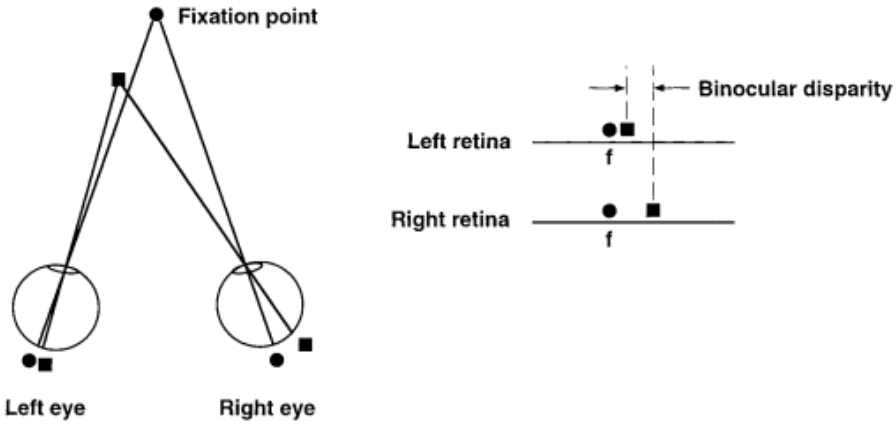


Figura 2.3: Demonstração detalhada de como a disparidade binocular é processada pelo cérebro humano [7].

A equação 2.1 relaciona a disparidade, d, com a distância, Z:

$$d = \frac{b \cdot F}{Z} \tag{2.1}$$

- onde F é a distância focal;
- b é a distância entre as câmaras (ou olhos).

Baseando-se neste princípio, é possível efetuar a reconstrução 3-D de uma cena utilizando duas câmaras RGB. Este processo é conhecido como visão *stereo*.

Na Figura 2.4 é mostrada a geometria de um sistema de visão *stereo*. A imagem apresenta o modelo mais simples de um sistema deste tipo, em que as duas câmaras encontram-se separadas apenas horizontalmente.

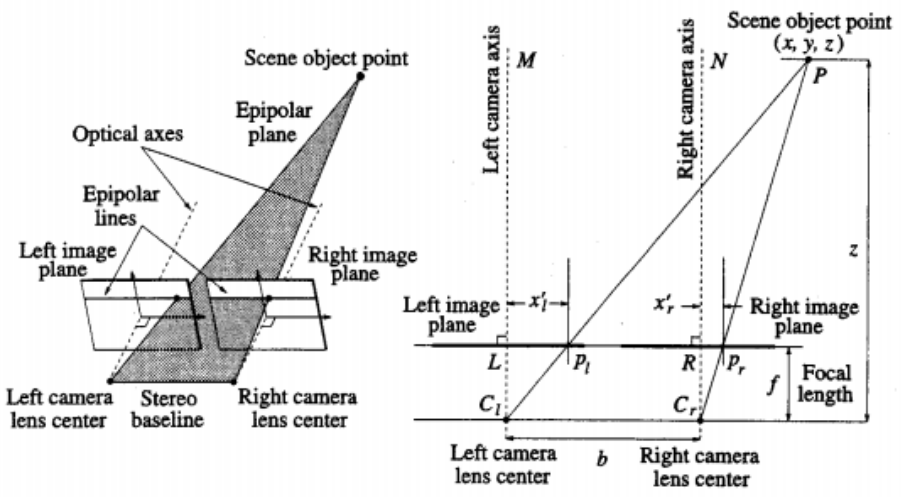


Figura 2.4: Qualquer ponto presente na cena, que seja visivel por ambas as câmaras, é projetado no sensor de ambas as câmaras. A diferença posicional entre os pontos é denominada de disparidade [8].

Na Figura 2.4 os planos da imagem são coplanares. O plano que passa através do centro das câmaras e o ponto 3-D da cena é denominado de plano epipolar. Um determinado ponto da imagem proveniente da primeira câmara irá corresponder na imagem proveniente da segunda câmara ao longo de uma linha, denominada de linha epipolar. Assim, para um determinado ponto numa imagem não é necessário efetuar uma pesquisa intensiva do ponto correspondente na segunda imagem.

Caso a câmara não se encontre alinhada com o eixo das abcissas, o modelo da visão *stereo* vai ser mais complicado, pois o cálculo da linha epipolar torna-se mais complexo, como será explicado de seguida.

Cálculo de linhas epipolares

As epipoles e_1 e e_2 podem ser determinadas pela interseção da *baseline*[1] e os planos das imagens. Se estes forem paralelos, as linhas epipolares vão tender para infinito, não existindo

[1]Linha definida pela interseção do centro ótico das duas câmaras.

epipoles. Para estas condições, as linhas epipolares são paralelas ao eixo das abcissas. Na Figura 2.5 é mostrado como as linhas epipolares são obtidas com os planos das imagens não paralelos, em (a), e paralelos, em (b).

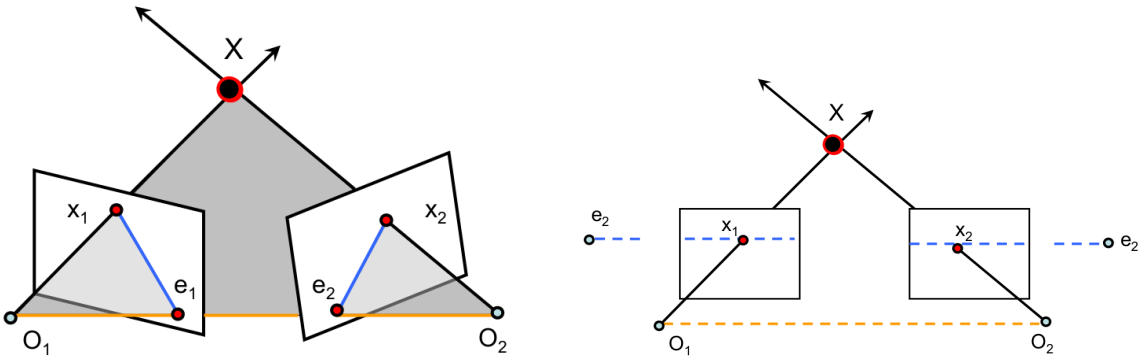


Figura 2.5: Geometria epipolar, (a) planos não paralelos e (b) planos paralelos [9].

Depois de efetuada a calibração das câmaras, os parâmetros intrínsecos e extrínsecos são conhecidos. O significado destes parâmetros e o processo de calibração das câmaras são abordados no Capítulo 4. Esta informação permite encontrar um vetor 3-D projetado em direção a um determinado pixel, X_i. O objeto no mundo, X, deve pertencer algures ao longo desta linha, como apresentado na Figura 2.6.

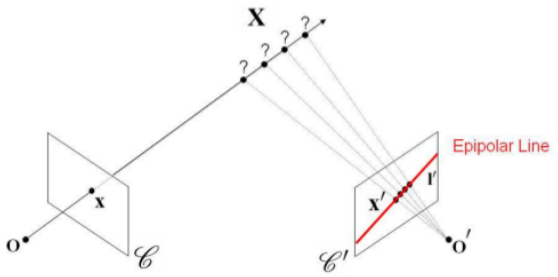


Figura 2.6: Correspondência entre pontos recorrendo às linhas epipolares. Para um ponto x na câmara C, o ponto correspondente em C', deve encontrar-se ao longo da linha epipolar [9].

A relação matemática entre os pontos nas duas imagens pode ser expressada pela seguinte equação: $x'^T E \cdot x = 0$ [10]. Onde E corresponde à matriz essencial. Esta matriz corresponde à transformação entre duas câmaras e pode ser utilizada para encontrar a linha epipolar correspondente numa imagem, em relação a um determinado ponto de outra imagem: $l = x' \cdot E$ ou $l' = x \cdot E$.

Conhecida a matriz essencial, é possível efetuar a correspondência de pontos de uma imagem para outra. Existem alguns métodos para calcular a matriz essencial, alguns bastante comuns são: o algoritmo *eight-point* [11] e o método *median of squares (LMedS)* [12].

Visto que se pretende que a distância entre as câmaras seja elevada, dado ser necessário monitorizar o campo de futebol na totalidade, o método apresentado não é o mais adequado.

Este funciona bem na obtenção da informação 3-D para *baselines* entre câmaras reduzidas. Além disso, este método é bastante útil para efetuar a reconstrução 3-D de uma cena, que não é o pretendido.

2.2.2 Método da triangulação

O método da triangulação baseia-se na interseção de duas (ou mais) linhas no espaço. As linhas consistem em vetores projetados desde o centro ótico das câmaras (como indicado na Figura 2.7) em direção ao objeto de interesse. Este método apresenta algumas desvantagens, tais como:

- É assumido que existe sempre a correspondência de um ponto de uma imagem para outra. Visto que em amplas *baselines* o número de pontos vistos pelas duas câmaras diminui, isso pode nem sempre acontecer.

- A triangulação consiste em calcular a interseção de dois vetores no espaço. À primeira vista, isto parece um problema de fácil resolução. No entanto, devido à presença de ruído ou outros fatores, os vetores podem não se chegar a intersetar, pelo que é necessário calcular o ponto mais próximo entre os dois vetores, o que pode gerar um erro significativo.

O problema principal da triangulação consiste na correspondência entre dois pontos de duas imagens. Pela Figura 2.7 verifica-se que numa correspondência mal efetuada, os pontos $x \longleftrightarrow x'$ encontram-se próximos da correspondência real, $\hat{x} \longleftrightarrow \hat{x}'$. Todavia, para estes casos a interseção dos vetores irá falhar, como apresentado na Figura 2.7. Para satisfazer este problema existem alguns métodos propostos na literatura, tais como: *Homogeneous method (DLT)* e *Inhomogeneous method*, para calcular o ponto mais próximo esntre dois vetores no espaço. Em [13] é efetuado um estudo detalhado sobre a performance destes métodos, em termos de precisão.

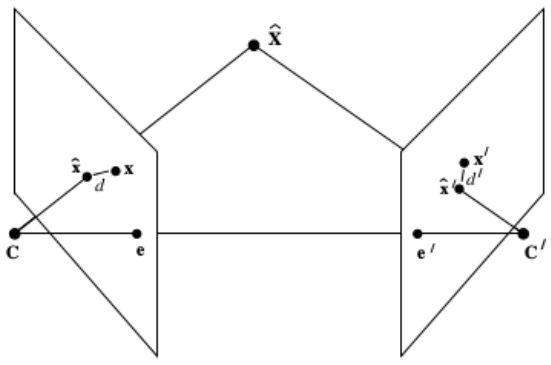


Figura 2.7: Estimativa da posição 3-D de um ponto recorrendo ao método da triangulação [13].

2.2.3 Correspondência de múltiplas imagens

O método proposto em [14] consiste em utilizar várias câmaras em campos de desporto para efetuar o *tracking* de jogadores.

O método consiste não só na calibração de cada sequência das câmaras, mas também tirando vantagem das múltiplas correspondências para tornar o *matching* dos jogadores mais robusto. Novas câmaras podem ser adicionadas ao sistema em qualquer instante de tempo, adicionando mais robustez ao sistema.

Um dos desafios consiste em estabelecer correspondências entre múltiplas câmaras, inicializando um *bag-of-features* para uma frame calibrada. As correspondências entre as diferentes câmaras são efetuadas em dois passos fundamentais:

1. Calcular descritores invariantes para cada *feature* que se pretende efetuar o *tracking*, transcrevendo-os para um sistema de coordenadas comum a todas as câmaras. É efetuada uma correspondência grosseira entre os descritores armazenados nesta sequência e na anterior, atualizando o *bag-of-features*. Isto acrescenta uma maior precisão e fiabilidade na deteção das *features*.

2. Descritores invariantes a fatores de escala são extraídos das diferentes *features*, armazenadas na visualização comum a todas as câmaras, o que leva a correspondências mais precisas e robustas.

   Sempre que uma sequência nova é adicionada ao sistema, as novas *features* são registadas utilizando homografias.[2] O mapeamento entre as *features* é efetuado em todas as imagens, recorrendo ao método MSER.[3] Sendo de seguida armazenadas numa única imagem de referência.

É demonstrado que esta aproximação consegue lidar muito bem com amplas *baselines*. Na Figura 2.8 são mostradas duas imagens calibradas de um campo de *hockey*. Estas foram obtidas por duas câmaras diferentes, com uma ampla *baseline*.

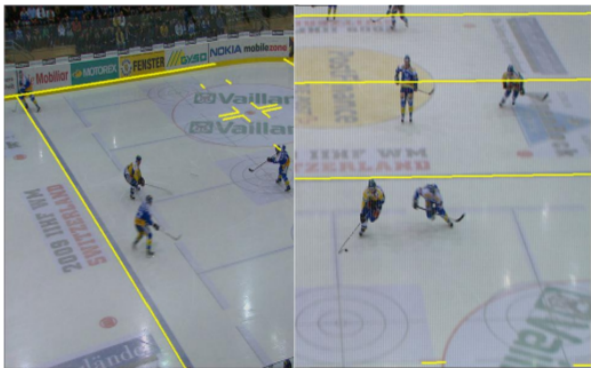


Figura 2.8: As figuras foram capturadas por duas câmaras distintas, com uma ampla *baseline*. A calibração bem sucedida é demonstrada pela projeção das linhas amarelas no campo [14].

[2] Em visão por computador, duas imagens no mesmo plano podem ser relacionadas utilizando homografias.
[3] Acrónimo de *maximally stable extremal regions*.

2.2.4 Equações da velocidade

O método proposto em [15] descreve uma forma de efetuar o *tracking* de jogadores e da bola de futebol tendo como base as equações da velocidade.

O método proposto é aplicado em transmissão de televisão, baseando-se na rotação e *zoom* de uma câmara. Visto que em cada frame adquirida o *zoom* e rotação da câmara alteram, é necessário efetuar a calibração da câmara para cada frame adquirida. Isto é feito extraindo pontos de interesse no campo, tais como linhas e círculos, como apresentado na Figura 2.9.

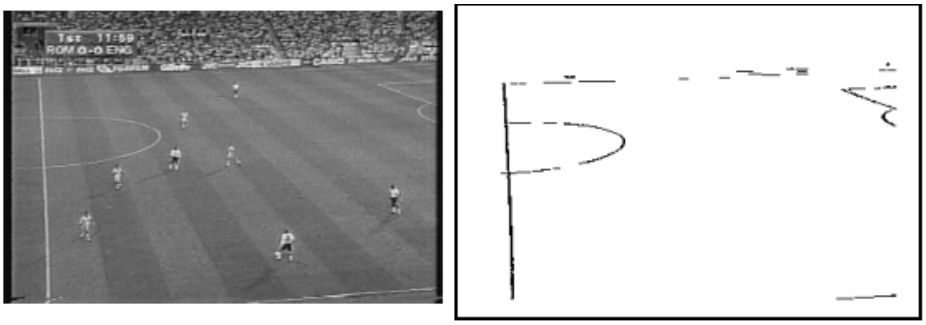


Figura 2.9: À esquerda encontra-se a imagem original e à direita a deteção das linhas do campo na imagem.

2.3 Deteção de *features* em imagens

Para efetuar a monitorização de objetos em 3-D com múltiplas câmaras RGB, inicialmente é necessário efetuar a deteção dos objetos de interesse nas imagens provenientes das diferentes câmaras.

A deteção de *features* em imagens é uma das tarefas mais importantes em visão artificial. Esta é bastante difícil em muitas aplicações, visto ser necessário efetuar a deteção de objetos de interesse com alguma precisão. Nesta secção são apresentados diferentes tipos de algoritmos aplicados em visão por computador para efetuar a deteção de *features* em imagens.

2.3.1 Segmentação por cor

A informação de cor é usada em vários algoritmos de visão, nomeadamente nas seguintes aplicações:

1. verifica se a cor de um determinado objeto corresponde à cor que o sistema de visão está programado para encontrar;
2. ordena vários objetos tendo como base a sua cor;
3. utilizado para procurar defeitos em objetos.

Os espaços de cor são modelos matemáticos utilizados para caracterizar as cores. Estes podem ser vistos como uma base representativa das componentes.

A luminância é definida como uma medida da luminosidade fotométrica por unidade de área. Esta é utilizada para descrever a quantidade de luz que atinge um determinado objeto.

De modo a representar a luminosidade, cada pixel é convertido em uma média com pesos diferentes para cada uma das cores. Esses pesos assumem que o verde representa 59% da luminosidade percebida, enquanto que o vermelho e o azul representam apenas 30% e 11%, respetivamente.

A crominância é uma medida que descreve a forma como a luz é distribuída no espetro visível. A crominância é independentemente da luminância. Esta consiste em dois parâmetros independentes: a tonalidade da cor, *hue*(h), e a intensidade da cor, também conhecida como a saturação (s).

- RGB (*Red*,*Green*,*Blue*): Sistema utilizado na visão humana, em monitores e em televisões (espaço de cor aditivo).
- CMYK (*Cyan*,*Magenta*, *Yellow*,*Black*): Sistema utilizado em impressoras (espaço de cor subtrativo).
- HSV (*Hue*,*Saturation*, *Value*): Sistema apropriado para a segmentação por cor. Separa a luminância da crominância.
- YCbCr: Sistema utilizado em vídeo digital. É utilizado como forma de codificação da informação RGB.

Normalmente os espaços de cor são entidades tridimensionais. O espaço de cor RGB é criado pelo mapeamento de três componentes de cor (*Red*, *Green*, *Blue*) num espaço tridimensional. De seguida é apresentado e explicado um software, desenvolvido para a equipa CAMBADA para a deteção de objetos coloridos [16].

2.3.1.1 Deteção de objetos coloridos recorrendo à biblioteca UAVision

A biblioteca UAVision incorpora um conjunto de módulos que permite o desenvolvimento de sistemas de visão para a segmentação de cor [1]. Na Figura 2.10 é apresentado um diagrama, que retrata as fases desde que a imagem é adquirida até à deteção dos objetos de interesse.

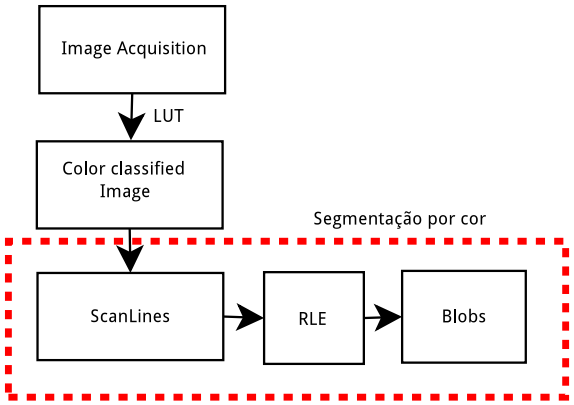


Figura 2.10: Sistema de visão para deteção de um objeto numa imagem, implementado para a equipa CAMBADA [17].

A biblioteca é composta por quatro sub-módulos, que são apresentados de seguida:

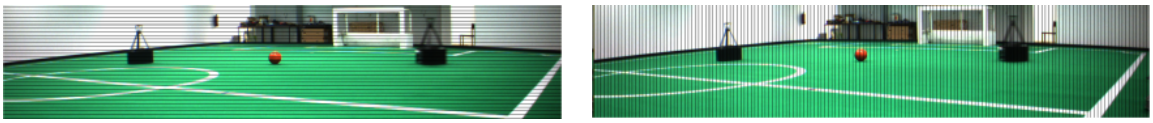
- ***Look-Up Table*:** Para que a classificação por cor seja rápida, é utilizada uma *Look-Up Table* (LUT). Esta representa uma estrutura de dados, utilizada para substituir o cálculo computacional por uma operação básica de indexação de um array. Esta aproximação permite obter uma redução significante do tempo de processamento.

  As imagens podem ser adquiridas em RGB, YUV ou formato *Bayer* e são convertidas para uma *index image* (*image of labels*), utilizando uma LUT apropriada para cada uma das três possibilidades.

  A tabela consiste em 16,777,216 entradas (2^{24}, 8 bits para o R, 8 bits para o G e 8 bits para o B). O tamanho da tabela é o mesmo para as outras duas possibilidades (YUV e *Bayer*), mas o significado de cada componente altera. Cada bit na tabela representa uma das cores de interesse (branco, verde, azul, amarelo, laranja, vermelho, céu azul, cinzento - sem cor) caso a cor se encontre dentro da classe correspondente.

  Para classificar um pixel, primeiro o valor da cor do pixel é lida e em seguida utilizada como um índice para a tabela. O valor de 8 bits lido a partir da tabela é denominado de "máscara de cor"do pixel.

- ***ScanLines*:** Para extrair a informação de cor nas imagens foram implementadas três tipos de linhas de pesquisa: radiais, lineares (horizontais ou verticais) e circulares. Neste trabalho apenas serão utilizadas as *ScanLines* do tipo linear, como mostrado na Figura 2.11.



(a) *ScanLines* horizontais (b) *ScanLines* verticais

Figura 2.11: *ScanLines* lineares [16].

  Cada objeto do tipo *ScanLine* é utilizado pelo módulo RLE, como será explicado de seguida.

- ***Run Length Encoding* (RLE):** Para cada *ScanLine*, um algoritmo de *Run Length Encoding* é aplicado. Este tem como objetivo obter informação da existência de uma cor específica na *ScanLine*. A deteção é efetuada iterando através dos pixels ao longo de cada *ScanLine*, como apresentado na Figura 2.12.

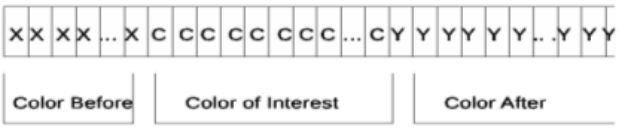


Figura 2.12: Ilustração de uma RLE.

  Na definição das *Run Lengths*, o utilizador pode especificar a cor de interesse, a cor antes e depois do objeto a detetar, assim como valores de *threshold* para as respetivas cores.

- ***Blob formation*:** Para a deteção de objetos de uma determinada cor é necessário detetar regiões na imagem com a respetiva cor, estas são denominadas de manchas (*blobs*).

Estes podem ser posteriormente validados como objetos de interesse, depois de extraídas algumas propriedades que são relevantes para definir o que as manchas realmente representam, mais do que apenas informação de cor.

Para construir os *blobs* é utilizada informação acerca da posição onde uma cor específica ocorre, baseado na informação do módulo *Rung Length*, anteriormente descrito.Todas as *Run Lengths* de uma cor específica são percorridas e um algoritmo de agrupamento das manchas é aplicado.

Na Figura 2.13 é mostrada a deteção das linhas do campo recorrendo à biblioteca apresentada. Na Figura 2.14 é detetada a bola de futebol recorrendo à mesma biblioteca.

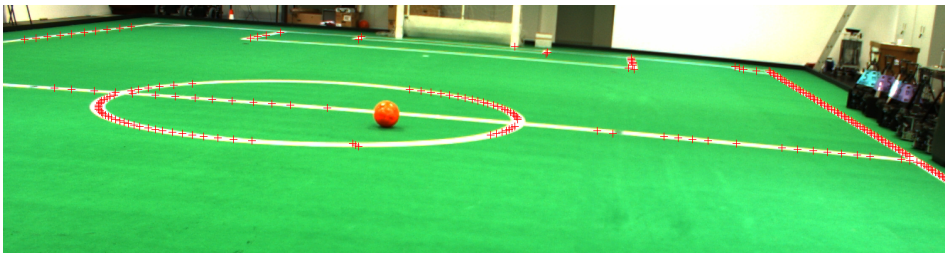
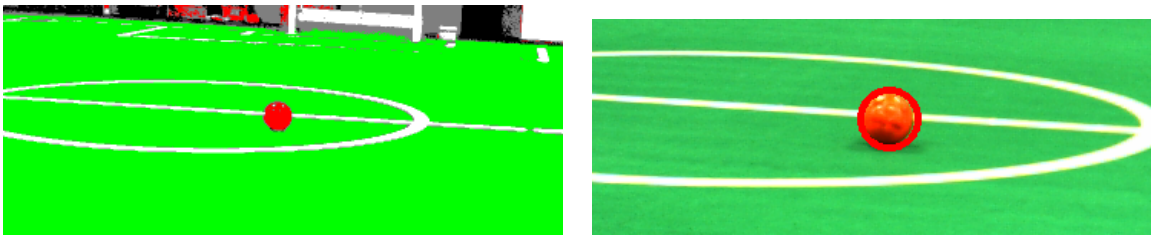


Figura 2.13: Deteção das linhas do campo. As *ScanLines* utilizadas foram do tipo linear vertical e linear horizontal, sendo que foram criadas *ScanLines* com um incremento de 2 em 2 pixels, tanto na vertical, como na horizontal. Os parâmetros utilizados para a RLE foram: 2 pixels de cor antes (verde), 2 pixels cor de interesse (branca) e 2 pixels de cor depois (verde).



(a) (b)

Figura 2.14: (a) Imagem classificada pela cor. as *ScanLines* utilizadas foram do tipo linear vertical e linear horizontal. Foram criadas *ScanLines* com um incremento de 2 em 2 pixels, tanto na vertical, como na horizontal. Os parâmetros utilizados para a RLE foram: 4 pixels de cor antes (verde), 4 pixels de cor de interesse (laranja) e 4 pixels de cor depois (verde). (b) Bola detetada na imagem original.

2.3.2 *Template matching*

Uma das técnicas para a deteção de *features* em imagens consiste em comparar "sub-regiões"de uma imagem com uma imagem de referência. Desta forma procura-se encontrar a correspondência de um modelo numa pequena parte da imagem global. Este processo de deteção de imagens é denominado de *template matching*.

Na Figura 2.15 é mostrado um exemplo da aplicação do *template matching*.

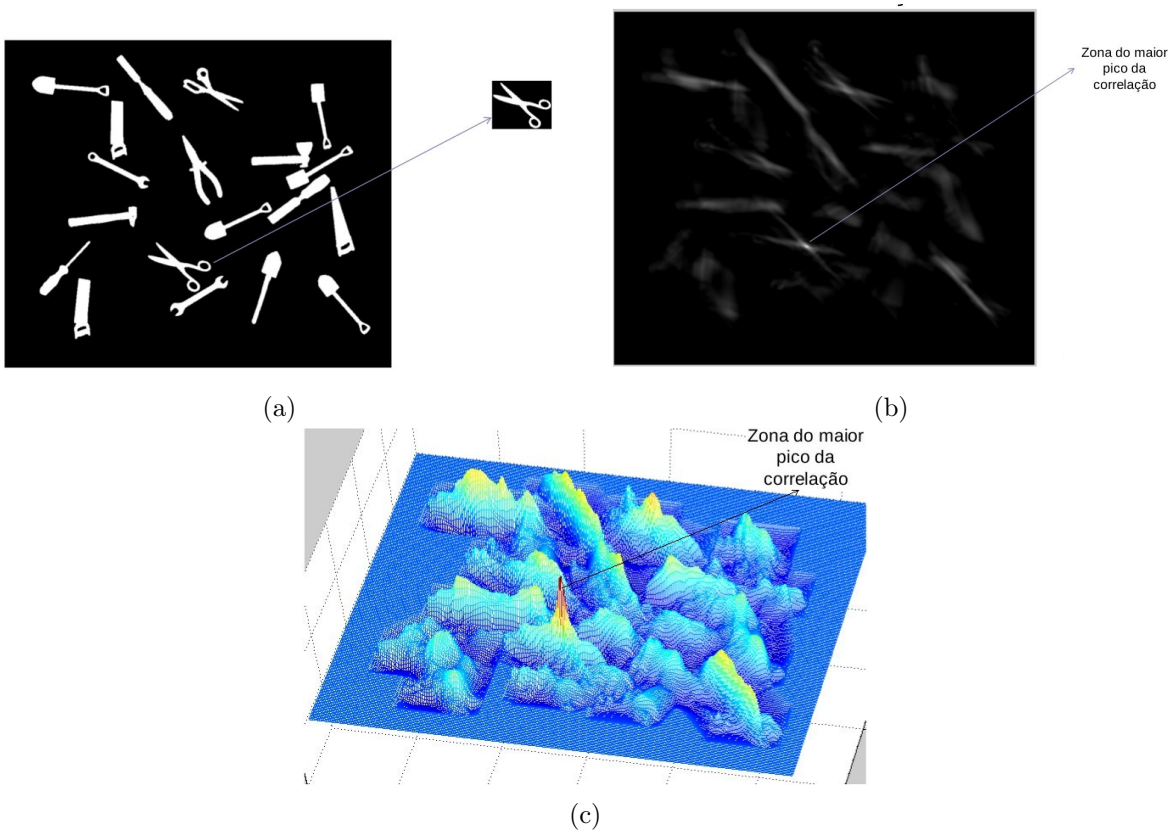


Figura 2.15: *Template matching* e resultados da corelação em 2-D e 3-D [18].

2.3.3 Descritores invariantes

A correspondência entre imagens pode ser efetuada recorrendo ao uso de descritores locais para a representação do objeto de interesse.

Os descritores não são nada mais do que vetores que caracterizam uma imagem, ou uma região específica desta. De seguida são apresentados alguns dos algoritmos baseados em descritores invariantes mais utilizados.

2.3.3.1 SIFT

O SIFT[4] permite efetuar a deteção e extração de descritores locais, estes são invariantes a mudanças de luminosidade, ruído, rotação, escala e pequenas mudanças de perspetiva.

Recentemente, as técnicas SIFT têm sido bastante utilizadas em reconhecimento facial e correspondências em amplas *baselines* [19]. O SIFT deteta regiões específicas em imagens recorrendo ao DoG,[5] que extrai regiões circulares dos pontos de interesse.

Para calcular as *SIFT features*, quatro principais passos são necessários [20]:

1. ***Scale-space peak detection***: O primeiro passo consiste numa pesquisa do objeto a detetar, para todos os fatores de escala e orientações possíveis. Este passo é implementado

[4] Acrónimo de *Scale Invariant Feature Transform*
[5] Acrónimo de *Difference-of-Gaussian*

Tabela 2.1: Comparação dos algoritmos SIFT e SURF [23]

Método	**Tempo**	**Escala**	**Rotação**	**Imagens desfocadas**	**Iluminação**	**Precisão**
SIFT	Normal	Melhor	Melhor	Melhor	Normal	Bom
SURF	Melhor	Bom	Normal	Bom	Melhor	Bom

de forma eficiente usando a função DoG, que localiza potenciais pontos de interesse, invariantes a fatores de escala e orientação.

2. ***Keypoint localization and Filtering***: Para cada candidato, um modelo detalhado é executado para determinar a sua localização e escala, os *keypoints* são selecionados baseando-se em medidas da estabilidade destes. Este passo melhora a deteção dos *keypoints* e exclui deteções mal efetuadas no passo anterior.

3. ***Orientation assignment***: Uma ou mais orientações são atribuídas à localização de cada *keypoint*, baseando-se nas direções da imagem local de gradiente. Todas as operações futuras serão tomadas relativamente aos dados da imagem que foi o resultado da transformação e orientação atribuída, escala, e localização de cada *feature*, proporcionando assim invariância a estas transformações. Este passo remove efeitos de variação à rotação e fatores de escala.

4. ***Keypoint descriptor***: Os gradientes das imagem locais são medidos, à escala selecionada na região à volta de cada *keypoint*. Estas são transformações para uma representação que permite a deteção de objetos de interesse, mesmo com níveis elevados de distorção local ou mudanças de iluminação.

2.3.3.2 SURF

O algoritmo SURF[6] é focado na performance (em termos de rapidez) para a extração de objetos. É invariante a fatores de escala e rotação. Baseia-se em imagens integrais para a deteção de regiões, reduzindo o tempo de computação (*Fast-Hessiam detector*) [21]. Uma desvantagem das imagens integrais é a capacidade para calcular *box filters* de todos os tamanhos a uma velocidade constante [22]. O conceito de imagens integrais foi originalmente proposto como *summed area table*, esta estrutura de dados consiste num rápido algoritmo para gerar a soma de valores em uma área rectangular [4].

O algoritmo SURF é definido por um procedimento de três passos, muito semelhante ao algoritmo SIFT: *keypoint detection*, *orientation assignment* e *description*.

Na Tabela 2.1 são apresentadas as vantagens de desvantagens do SIFT, face ao SURF [23].

2.3.3.3 FAST

O FAST[7] é um método para identificar pontos de interesse em imagens. Um ponto de interesse diz respeito a um pixel com uma posição bem definida e que pode ser detetado de forma robusta e eficaz, devendo possuir bastante informação local.

[6] Acrónimo de *Speeded-Up Robust Features*.
[7] Acrónimo de *Features from accelerated segment test*.

O método FAST apresenta uma eficiência computacional muito elevada. Foi desenvolvido para efetuar a deteção de pontos de interesse para utilizar em aplicações de tempo real, tais como SLAM,[8] onde os recursos computacionais são limitados [24].

Como mostrado na Figura 2.16, o algoritmo começa por criar um circulo de 16 pixels à volta de cada pixel da imagem. Cada pixel é considerado um canto se existir um conjunto de n pixels no circulo que apresentem um determinado valor de intensidade. O algoritmo considera o valor da intensidade do pixel p mais um valor de *threshold*, que pode ser negativo ou positivo.

Além disso, para tornar o algoritmo mais rápido, um procedimento é tomado de forma a não efetuar a verificação anteriormente referida intensivamente para todos os pixels da imagem [24].

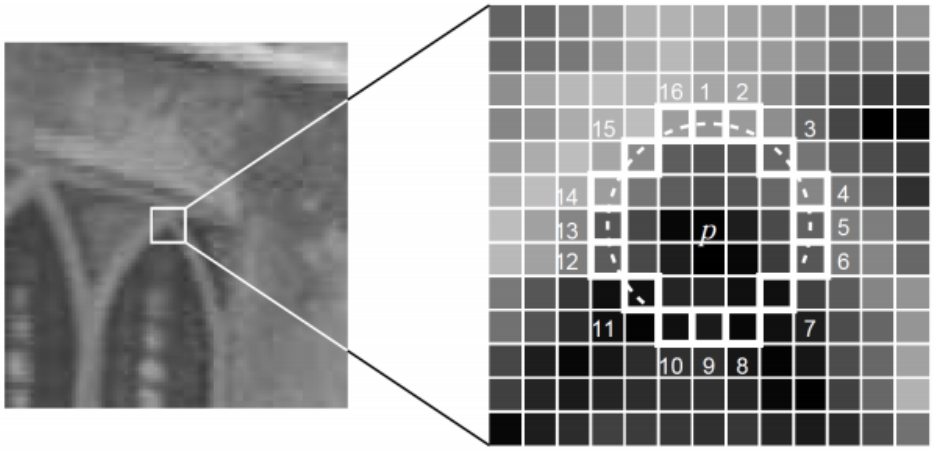


Figura 2.16: Processo para deteção de um pixel de uma imagem através de pontos de interesse. Na figura do lado direito, os 16 pixels em destaque correspondem aos pixels utilizados para a deteção do ponto de interesse, este corresponde à posição p na imagem [25].

Para evitar a deteção de múltiplos pontos de interesse adjacentes a apenas um canto, uma supressão dos pontos não máximos (*non-maximal supression*) é tomada. Para cada pixel considerado um canto no primeiro passo, o somatório das diferenças de intensidade dos 16 pixels do circulo com o pixel em análise é calculada. Dos pixels adjacentes, apenas o que apresentar o máximo valor da soma efetuada é considerado o correspondente ao canto.

Na Figura 2.17 é apresentada uma imagem onde o algoritmo FAST é aplicado, afim de efetuar a deteção de pontos de interesse, nesta é possível ver os pontos referentes excluídos pela supressão dos pontos não máximos (a verde) e os pontos referentes aos pontos de interesse (a vermelho).

Na presente dissertação utilizou-se a biblioteca *UAVision* para a deteção de *features*. Visto que se pretende efetuar o *tracking* de objetos coloridos (a bola) e esta biblioteca já ser utilizada pela equipa CAMBADA. Todavia, para efetuar o *tracking* de outros objetos, como os robôs, pode vir a ser útil explorar os algoritmos abordados.

[8] Acrónimo de *Simultaneous localization and mapping*.

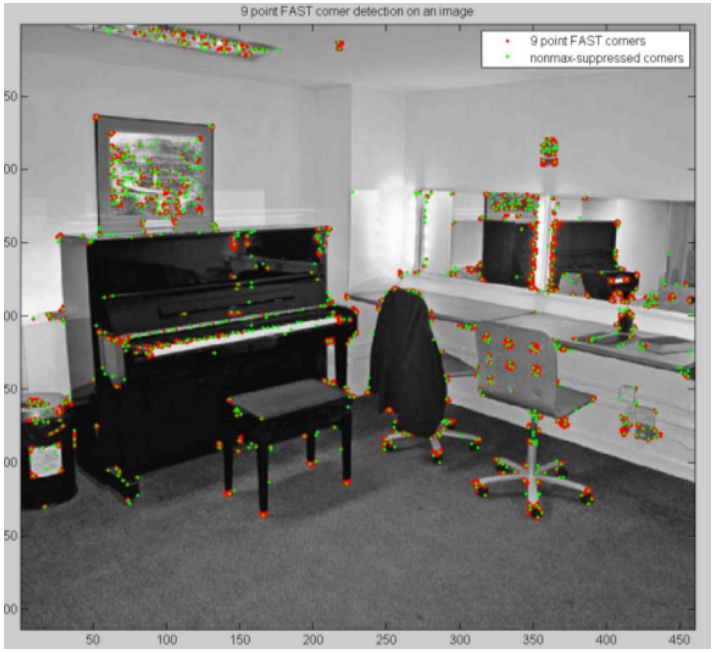


Figura 2.17: Imagem com os pontos de interesse detetados. Os pontos a verde mostram os *Non-maximally surpressed corners* [26].

Capítulo 3

Câmara utilizada no sistema desenvolvido

A visão por computador é um campo que inclui técnicas para a aquisição, processamento e análise de imagens digitais. Têm sido bastante explorados nos últimos anos algoritmos para o processamento de imagens digitais.

Neste capítulo é apresentado um estudo acerca do princípio de funcionamento das câmaras digitais, bem como da formação da imagem nestas. De seguida, é apresentado o modelo das câmaras utilizadas, assim como as suas principais características. Por fim, é descrita a interface para o *driver* das câmaras que foi desenvolvido, de forma a ser possível efetuar o controlo das mesmas via software.

3.1 Princípio de funcionamento de uma câmara digital

Uma câmara digital pode ser vista como um dispositivo que captura luz do ambiente para formar imagens digitais. O componente eletrónico responsável por efetuar esta transformação corresponde a um *chip* constituído por uma área sensível à luz, que absorve fotões e converte-os em eletrões, através do efeito fotoelétrico.

Existem dois tipos principais de *chips* em câmaras digitais: CCD (*charge-coupled device*) e CMOS (*complementary metal-oxide semiconductor*). Pode-se ver estes sensores como um array 2-D de milhões de pequenos fotodiodos, em que cada um destes converte a luz duma pequena parte da imagem em eletrões.

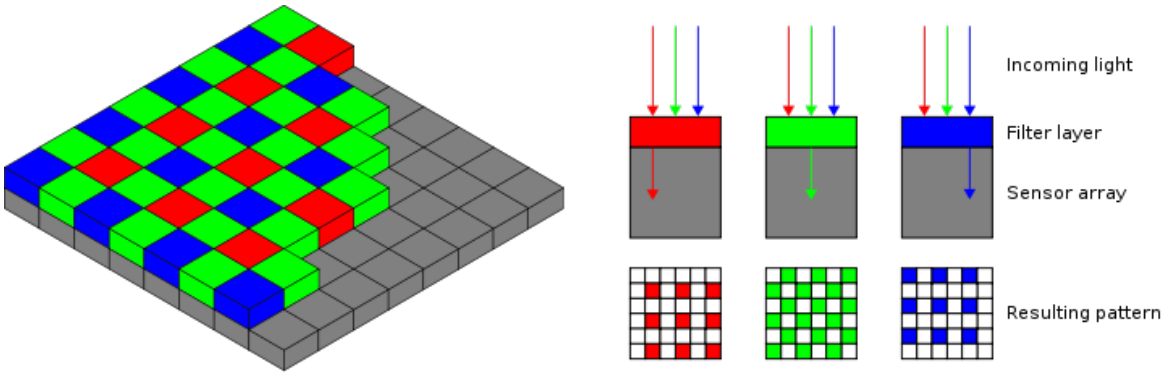
O processo de funcionamento destes sensores é semelhante, diferindo essencialmente na forma como a carga é convertida em tensão e na transferência da mesma para fora do sensor. São dois tipos de tecnologias diferentes, apresentando vantagens e desvantagens, uma face à outra, sendo que a escolha do tipo de sensor deve ser efetuada consoante a aplicação. Na Tabela 3.1 são apresentadas as principais características destes sensores.

O *shutter* é um componente da câmara que abre e fecha para controlar o tempo que o plano focal é exposto à luz. Quando uma câmara captura uma imagem, o *shutter* abre instantaneamente e cada pixel no sensor da imagem armazena a quantidade de luz que insidiu sobre este, através da acumulação de uma carga elétrica. Quando o tempo de exposição termina, o *shutter* fecha e a carga armazenada por cada pixel é medida e convertida num valor digital. No sensor da imagem, os pixels apenas capturam o brilho, e não a cor. Nas câmaras RGB são colocados filtros: vermelhos, verdes e azuis, em cada pixel, sendo possível efetuar a

Tabela 3.1: Comparação entre os sensores CCD e CMOS [27].

Sensor	CCD	CMOS
Nível de ruído	Baixo	Moderado - Alto
Gama dinâmica	Alto	Moderado
Uniformidade	Alto	Baixo
Resolução	Baixo - Alto	Alto
Velocidade	Moderado - Alto	Alto
Consumo de energia	Moderado - Alto	Baixo

reconstrução da imagem à posteriori. Este filtro é conhecido como *Bayer matrix* e é ilustrado na Figura 3.1.



(a) (b)

Figura 3.1: (a) Arranjo dos pixels pelas cores, no array de pixels. (b) Secção da cor vermelha, verde e azul [28].

3.1.1 Formação da imagem na câmara digital

Quando a luz atinge um objeto, grande parte desta é absorvida pelo objeto e a restante é refletida, fornecendo a perceção de cor.

A luz refletida que atinge a câmara (ou o olho) forma a imagem. Um simples modelo que procura ilustrar o funcionamento deste processo é o modelo de um *pinhole* de uma câmara. Este consiste num pequeno orifício em que os raios de luz irão passar através deste. Se tratarmos o *pinhole* como um único ponto, apenas um raio de luz para cada ponto da imagem pode entrar na câmara. Na Figura 3.2 é ilustrado o modelo referido.

3.2 Câmara digital utilizada

Para efetuar a monitorização do jogo de futebol robótico, vai-se recorrer a quatro câmaras RGB, cujo o modelo é o seguinte: PointGrey Zebra - ZBR2-PGEHD-20S4C-CS, apresentada na Figura 3.3. Estas foram escolhidas de acordo com os seguintes fatores:

1. Resolução da imagem: Permitem adquirir imagens com uma elevada resolução (1600 x 1200 pixels). Este fator é importante, pois para o mesmo tamanho do sensor da imagem,

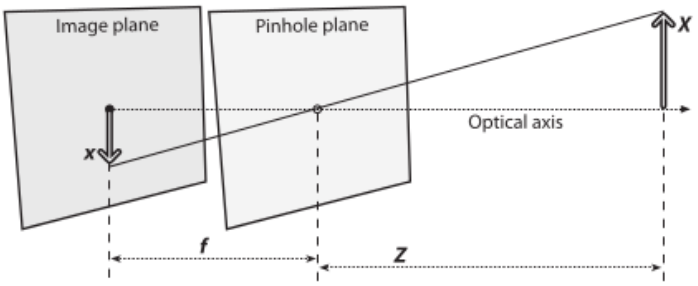


Figura 3.2: Modelo *pinhole* de uma câmara. Este descreve a relação geométrica entre um ponto no espaço e a sua correspondência em um plano 2-D [29].

Figura 3.3: Câmara PointGrey Zebra - ZBR2-PGEHD-20S4C-CS.

---

quanto maior a resolução menor é o espaçamento entre os pixels. Visto que os objetos a detetar podem encontrar-se a uma distância bastante elevada da câmara, em imagens com uma maior resolução consegue-se efetuar o reconhecimento destes com uma maior precisão, face a imagens cuja a resolução é menor. Além disso, o tamanho dos pixels tem influência no ruído da imagem, sendo que para tamanhos de pixels inferiores o ruído na imagem é mais baixo.

2. Sensor do tipo CCD: Os sensores CCD permitem obter imagens com menos ruído, face aos CMOS, este fator resulta em parte devido à elevada gama dinâmica dos sensores CCD.

   Embora nestes sensores a carga armazenada nos pixels seja limitada à velocidade que estes podem ser transferidos, isto leva a que exista uma elevada consistência entre cada pixel, apresentando assim uma maior uniformidade entre estes.

   Visto que se pretende efetuar o *matching* de objetos entre diferentes câmaras, estas vantagens dos sensores CCD são bastante relevantes, pois pequenos erros no *matching* do objeto a detetar podem resultar em elevados erros na obtenção das coordenadas 3-D do mesmo.

3. Suportar ligação *Gigabit Ethernet*: O facto das câmaras suportarem ligação *Gigabit Ethernet* permite que a distância da câmara ao local de processamento seja elevada, o que pode ser útil, dada a elevada dimensão do campo de futebol. Caso as câmaras apenas suportassem ligação USB, a distância entre estas e o local de processamento seria muito limitada, na ordem dos cinco metros.

4. Apresentam um mecanismo de sincronização externo. Esta é uma característica obrigatória na escolha das câmaras, dado que é necessário garantir que as frames obtidas a

partir destas sejam tiradas exatamente no mesmo instante de tempo.

5. Apresentam uma boa relação custo / qualidade.

Estas câmaras são bastante utilizadas em aplicações de vigilância, instalações de processamento de monitorização, visão *stereo*, entre outras. Na Tabela 3.2 são apresentadas as suas principais características.

Tabela 3.2: Principais características da câmara ZBR2-PGEHD-20S4C [30].

Canais de memória	2 canais de memória para configurações de câmara personalizados
Sensor de imagem	Sony ICX274 CCD, 1/1.8", 4.4 µ m
Memória flash	1 MB
Dimensões	44 mm x 44 mm x 87.5 mm
Consumo de energia	8-30 V, $<$ 6 W
Controlo da Câmara	Via *FlyCapture SDK*
Temperatura	Operação: 0ºC a 45ºC; Armazenamento: -30º to 60ºC

Na Figura 3.4 é mostrado o exemplo de uma imagem adquirida por estas câmaras.

Figura 3.4: Exemplo de uma imagem adquirida por uma câmara PointGrey Zebra - ZBR2-PGEHD-20S4C-CS.

3.3 Interface para o *driver* das câmaras

O fabricante das câmaras disponibiliza um SDK (*Software Development Kit*) para controlar e adquirir imagens das câmaras *Point Grey*, designado por *Flycapture SDK*. Este permite configurar diversos parâmetros, tais como: ganho, *frame rate*, ângulo de exposição (*exposure*), brilho, *sharpness*, entre outras propriedades. Na Figura 3.5 é mostrado o SDK fornecido.

Para se conseguir efetuar o controlo das câmaras por software foi necessário implementar uma interface para o *driver* das câmaras, adicionando esta à biblioteca UAVision. Os métodos desenvolvidos encontram-se descriminados no Anexo A.

Para se adquirir uma imagem da câmara basta executar o fragmento de código indicado abaixo, sendo que a verde encontram-se os métodos desenvolvidos necessários para esta função. No Anexo B encontram-se descriminado o código referente as estes métodos.

```
1  Camera *cam;
2  cv::Mat image;
3  //Define subnet mask and default gateway
4  FlyCapture2::IPAddress subnetMask = ((255<<24)+(255<<16)+(255<<8)+0);
5  FlyCapture2::IPAddress DG = ((192<<24)+(168<<16)+(1<<8)+1);// Default Gateway
6
7  ///*************************************
8  //Define IP Address
9  FlyCapture2::IPAddress ipAddress = ((192<<24)+(168<<16)+(1<<8)+141); //IP Address
10 FlyCapture2::MACAddress macAddress = (((00<<8)+(176)),((157<<24)+(217<<16)+
11 (48<<8)+(106))); //MAC Address
12
13 cam = new CameraZebra(ipAddress,macAddress,subnetMask,DG,numCamera);
14 cam->initCamera(config.camSettings);
15 cam->readFrame(image);
16 cam->setAutoMode(config.camSettings);
17 printf("CONFIGURATION FINISHED\n");
```

3.4 Ferramentas para aquisição de imagens

Desenvolveu-se software para trabalhar em modo *offline*, sendo possível trabalhar sem que as câmaras estejam ligadas ao computador. O software que deve ser executado para trabalhar neste modo é denominado de *visionOffMode*. Baseou-se na gravação de vídeos para as diferentes câmaras, sendo que posteriormente é efetuada a leitura dos mesmos, efetuando a análise das frames contidas nestes. Desta forma têm-se imagens previamente armazenadas em vídeos, não sendo necessário ter as câmaras ligadas para a obtenção das imagens.

O software para gravar o vídeo é denominado de *recordVideo*, este deve ser executado indicando o número da câmara e o número de frames que se pretende adquirir:

```
recordVideo -numCamera NUM_CAMERA -numImages NUM_IMAGES
```

De salientar que cada câmara possui um número (de 1 a 4), sendo que através desta indicação o software conhece o *MAC Adrress* e *IP Address* da câmara que se pretende adquirir imagens.

Depois de gravados os vídeos é possível trabalhar em modo *offline*. Como tal, basta executar o software *visionOffMode*. Este vai ler as frames dos vídeos, em vez de adquirir as frames diretamente das câmaras.

3.5 Ferramentas para a deteção da bola

Para a deteção da bola recorreu-se à biblioteca UAVision, apresentada anteriormente na Secção 2.3. Na Figura 3.6 é mostrada uma imagem que retrata o processo para a deteção da bola. Em (a) é mostrada a imagem original, em (b) a imagem de indexação (*index image*) obtida a partir da LUT, e em (c) a imagem classificada pela cor.

3.5.1 Tempo de processamento

Em aplicações robóticas o tempo de processamento é um fator de elevada importância, pois o desempenho do sistema depende fortemente deste. Nesta secção é apresentado o tempo de processamento do processo de deteção da bola numa imagem RGB, recorrendo à biblioteca UAVision (apresentada na Secção 2.3) para um computador com as características apresentadas na Tabela 3.3.

Tabela 3.3: Características do computador de desenvolvimento e teste do sistema.

Memória RAM	6 GiB
Processador	Intel® Core™ i5-4200M CPU @ 2.50GHz x 4
Sistema operativo	Ubuntu 14.04, 64 - bit

O tempo de processamento foi obtido por software. Recolheu-se uma amostra de 10 frames e mediu-se o tempo de processamento em partes chave do código. Na Tabela 3.4 são mostrados os tempos obtidos, estes resultam da média de todas as medições efetuadas.

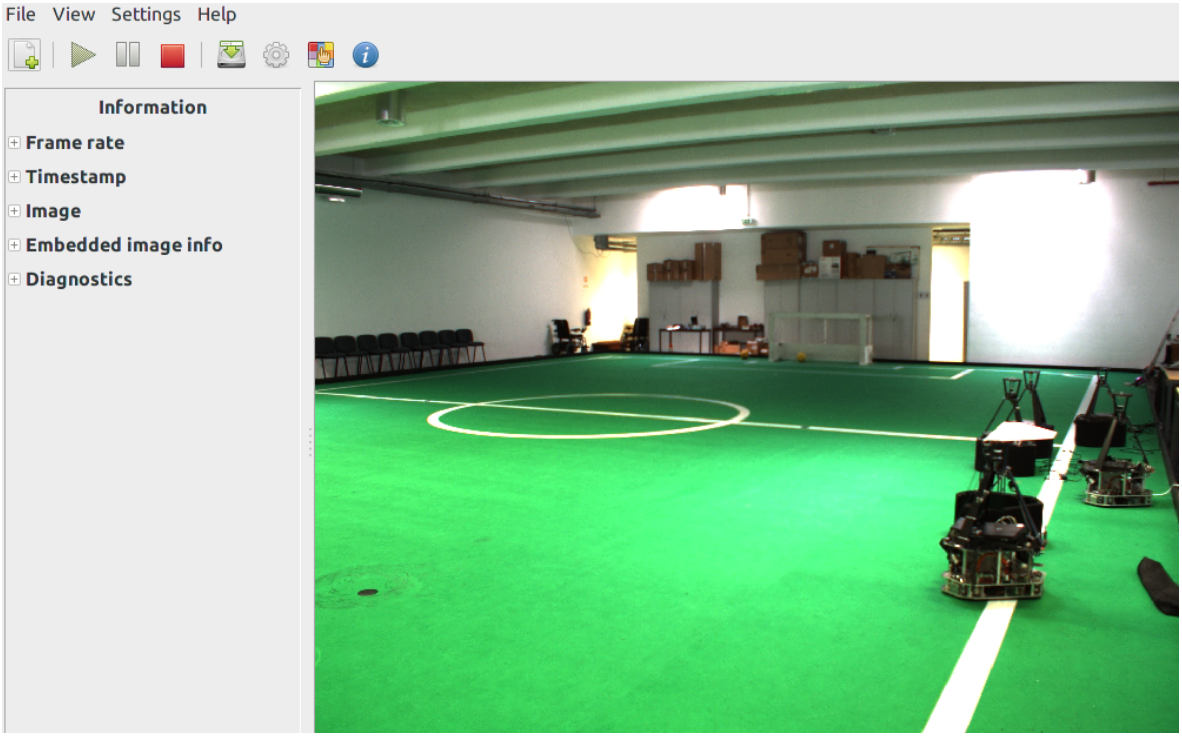
Tabela 3.4: Tempo de processamento do sistema de visão, em μs.

Aquisição da imagem	Deteção da bola	*Display*	Total
8078	52623	6721	68490.8

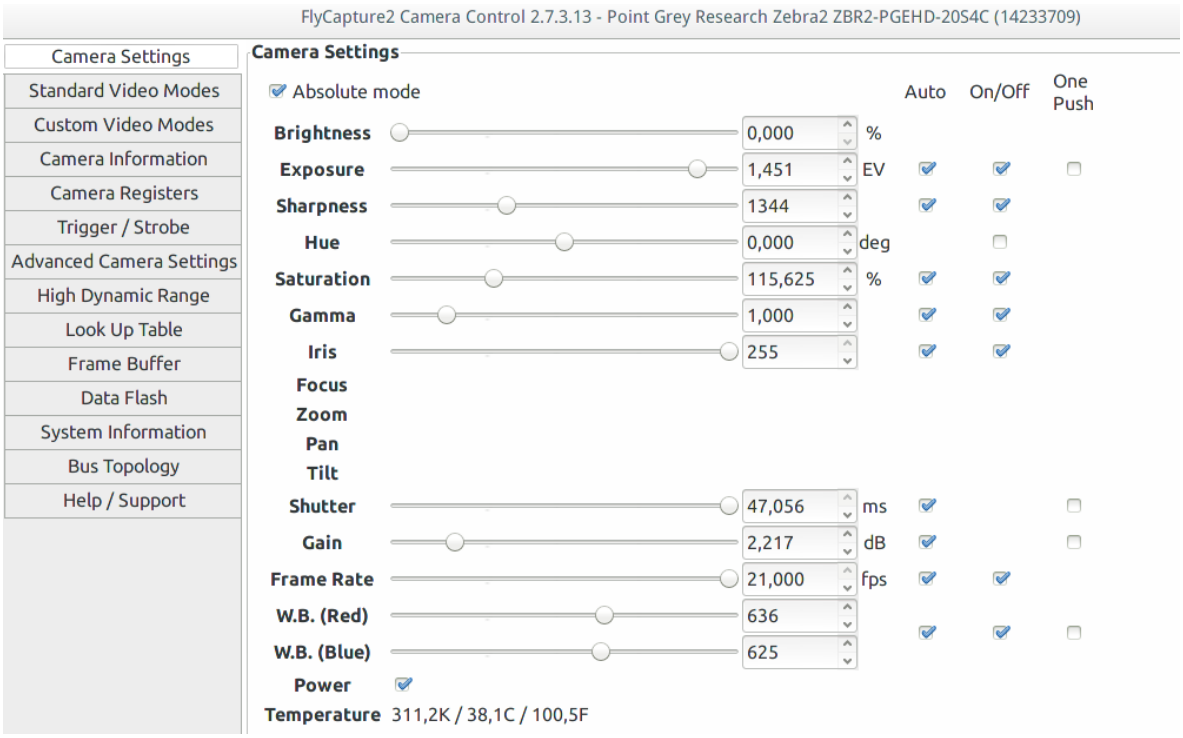
A deteção da bola corresponde à maior parte do tempo de processamento. Na Tabela 3.5 são apresentados os tempos de processamento correspondentes aos módulos da biblioteca UAVision (introduzida na Secção 2.3) necessários para a deteção da bola.

Tabela 3.5: Tempo de processamento dos módulos do UAVision para a deteção da bola, em μs.

LUT	ScanLines	RLE	Blobs	Validação	Total
21983	939.5	10608	2337	16756	35867



(a)



(b)

Figura 3.5: Software *Flycapture* da *PointGrey*, disponibilizado para controlar e adquirir imagens da câmara. Em (b) é possível visualizar os parâmetros da câmara que este permite controlar.

(a) (b) (c)

Figura 3.6: Deteção da bola recorrendo à biblioteca UAVision. Em (a) é mostrada a imagem original, em (b) a imagem binária filtrada por cor e em (c) a imagem classificada pela cor, aplicando a "máscara de cor"em cada pixel.

Capítulo 4

Calibração de câmaras

A calibração de câmaras permite obter a relação entre as unidades das câmaras (pixels) e unidades físicas (por exemplo: metros). Em aplicações em que se pretenda obter informação 3-D, recorrendo a câmaras digitais, a calibração das câmaras é um um passo essencial neste processo.

A calibração consiste em dois passos fundamentais: a obtenção dos parâmetros intrínsecos e extrínsecos das câmaras. Os parâmetros intrínsecos descrevem as propriedades geométricas das câmaras e os extrínsecos relacionam o posicionamento destas em relação a um determinado referencial.

Na presente dissertação irá-se recorrer ao método da triangulação para se obter as coordenadas 3-D dos objetos de interesse, como será explicado de seguida. Na Figura 4.1 é mostrado o diagrama de blocos do processo de funcionamento do sistema em desenvolvimento, sendo que o primeiro passo consiste em efetuar a calibração dos parâmetros intrínsecos e extrínsecos das câmaras.

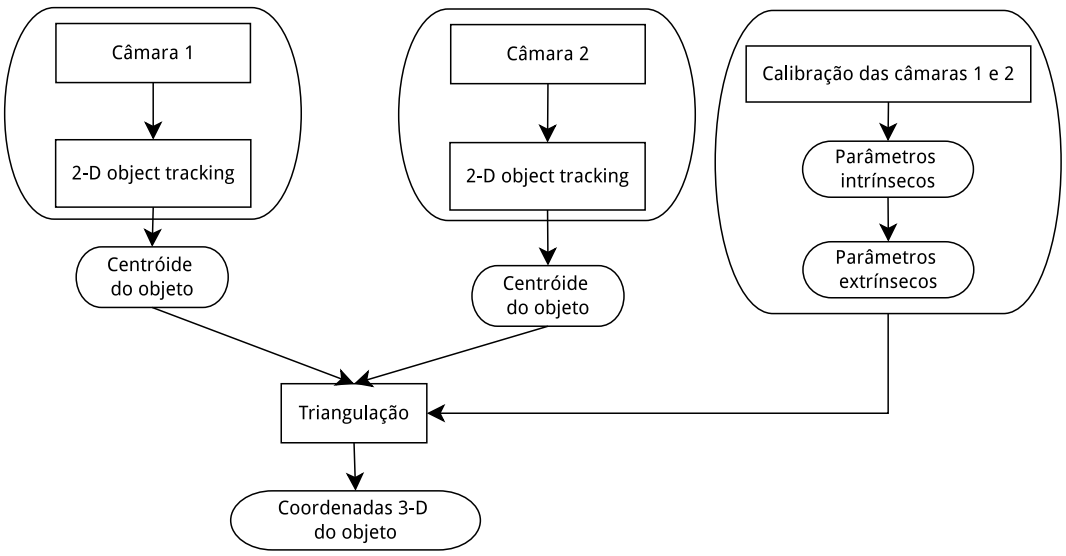


Figura 4.1: Diagrama de blocos do sistema em desenvolvimento, para determinar as coordenadas 3-D de objetos de interesse.

4.1 Parâmetros intrínsecos

Os parâmetros intrínsecos definem as coordenadas de um determinado ponto (pixel) em relação ao sistema de coordenadas da câmara. Estes dizem respeito não só à distância focal, ao centro ótico e ao coeficiente de distorção, mas também aos parâmetros para a correção da distorção da lente.

Um modelo muito utilizado para descrever a geometria de uma câmara é o modelo de um *pinhole*, anteriormente abordado no Capítulo 3. Este é interessante para se obter um melhor entendimento acerca do significado dos parâmetros intrínsecos das câmaras digitais. Na Figura 3.2 é apresentado o modelo referido. Nesta figura é possível visualizar que pela similaridade de triângulos, $-\frac{x}{f} = \frac{X}{Z}$, ou:

$$-x = \frac{X}{Z} \cdot f \tag{4.1}$$

O modelo de um *pinhole* pode ser transcrito para um modelo equivalente, em que é mais fácil deduzir as expressões matemáticas a partir deste. A Figura 4.2 mostra o modelo de um *pinhole* com o plano da imagem e do *pinhole* vistos de outra perspetiva.

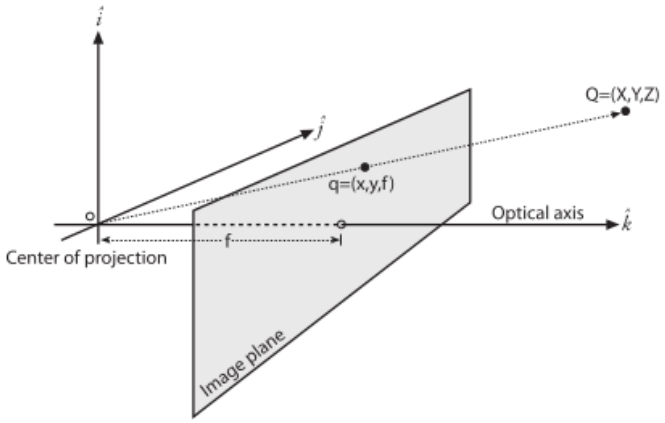


Figura 4.2: Modelo de um *pinhole* com o plano da imagem e do *pinhole* trocados [29].

Desta perspetiva, a similaridade de triângulos é mais direta do que anteriormente: $\frac{x}{f} = \frac{X}{Z}$. O sinal negativo desaparece porque nesta fase a imagem já não se encontra invertida.

Olhando desta perspetiva, cada ponto do objeto é diretamente projetado em direção ao centro ótico. O ponto da interseção do plano da imagem com o eixo ótico corresponde ao ponto principal (centro ótico), idealmente este deve localizar-se no centro da imagem.

Visto que as câmaras não são perfeitas, o centro ótico dificilmente se irá localizar no centro da imagem. Desta forma, é necessário introduzir dois tipos de parâmetros, c_x e c_y, para caracterizar esta variação, como mostrado nas equações em 4.2.

$$\begin{aligned} x_{screen} &= f_x \cdot \left(\frac{X}{Z}\right) + c_x \\ y_{screen} &= f_y \cdot \left(\frac{Y}{Z}\right) + c_y \end{aligned} \tag{4.2}$$

De notar que foram introduzidas duas distâncias focais diferentes. Isto ocorre porque muitas das vezes os pixels são retangulares, invés de quadrangulares. Na equação 4.3 é apresentada

a conversão da distância focal, em termos de distância métrica para pixels.

$$\begin{aligned} \alpha_x &= f_x \cdot m_x \\ \alpha_y &= f_y \cdot m_y \end{aligned} \tag{4.3}$$

Em que:

- f representa a distância focal em termos de distância métrica;
- m_x, m_y são os fatores de escala que relacionam pixels com a distância métrica.

O sistema de equações em 4.4 mostra a relação entre pixels com as coordenadas da câmara, recorrendo aos parâmetros intrínsecos.

$$\begin{bmatrix} x_{pix} \\ y_{pix} \end{bmatrix} = \begin{bmatrix} \alpha_x & \gamma & c_x \\ 0 & \alpha_y & c_y \end{bmatrix} \cdot \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} \tag{4.4}$$

O sistema de equações em 4.4 é impossível resolver diretamente. Obter informação 2-D a partir de informação 3-D é trivial, mas o inverso não. Para resolver este sistema de equações é necessário conhecer um dos seguintes parâmetros: x_s, y_s ou z_s. Tendo mais do que uma imagem da mesma cena é possível conhecer o parâmetro em falta, recuperando assim a informação 3-D anteriormente perdida. Por este motivo, é necessário utilizar múltiplas imagens retiradas da mesma cena a partir de diferentes câmaras, posicionadas em zonas distintas. Existem diversos algoritmos para obter informação 3-D a partir de múltiplas câmaras digitais, como foi apresentado na Secção 2.2.

Distorção da lente

A lente introduz distorção na imagem, sendo que existem dois tipos principais de distorção: radial e tangencial. A distorção radial resulta da forma da lente, enquanto que a tangencial surge a partir da montagem da câmara e da lente como um todo.

Quanto mais afastados se encontram os pixels do centro da imagem, mais acentuada é a distorção, como apresentado na Figura 4.3. Este fenómeno é muito conhecido como efeito "barril", ou "olho de peixe" [29]. Estes fatores de distorção podem ser caracterizados por aproximações polinomiais, como apresentado nas equações em 4.5.

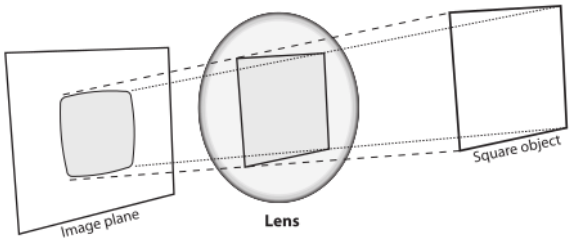


Figura 4.3: Efeito da distorção das lentes numa imagem [31].

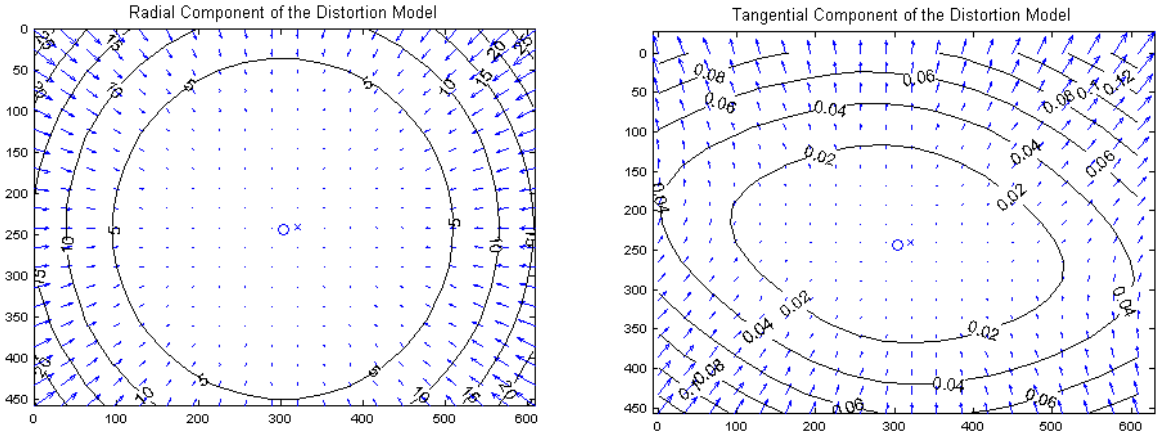
$$\begin{aligned} x_{corrected} &= x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ y_{corrected} &= y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \end{aligned} \tag{4.5}$$

A distorção tangencial ocorre devido às lentes não se encontrarem exatamente paralelas com o plano da imagem. À semelhança da distorção radial, esta também pode ser caracterizada por parâmetros adicionais, p_1 e p_2, como apresentado nas equações em 4.6.

$$\begin{aligned} x_{corrected} &= x + \left[2p_1 y + p_2\left(r^2 + 2x^2\right)\right] \\ y_{corrected} &= y + \left[p_1\left(r^2 + 2y^2\right) + 2p_2 x\right] \end{aligned} \tag{4.6}$$

Em que x e y correspondem à localização original do pixel, $x_{corrected}$ e $y_{corrected}$ correspondem à sua nova localização, corrigindo o efeito da distorção.

Na Figura 4.4 é mostrado o impacto que a distorção radial, (a) e tangencial, (b) tem numa imagem. A imagem da distorção radial mostra que esta é muito semelhante ao apresentado na distorção total (mostrado na Figura 4.5). Verifica-se assim que a componente da distorção tangencial apresenta um impacto baixo na distorção total de uma imagem.



(a) Distorção radial

(b) Distorção tangencial

Figura 4.4: Efeito da distorção observada para uma determinada câmara. Cada seta representa o deslocamento efetivo de um pixel, devido à distorção radial e tangencial da lente. A cruz indica o centro da imagem e o circulo a localização do ponto principal [29].

Na Figura 4.5 é mostrado o impacto da distorção radial e tangencial (distorção total) numa única imagem. Cada seta representa o deslocamento efetivo de cada pixel induzido pela distorção da lente. Observa-se que quanto mais afastados se encontram os pixels do centro da imagem, mais acentuado é o erro proveniente da distorção. No exemplo apresentado, para os pontos dos cantos da imagem, o deslocamento dos pixels chega a atingir os 25 pixels.

Existem muitos outros tipos de distorção, todavia, o impacto da distorção radial e tangencial numa imagem é bastante mais considerável, face aos restantes.

4.2 Parâmetros extrínsecos

Os parâmetros extrínsecos definem a localização e orientação da câmara em relação ao sistema de coordenadas do mundo.

Recorrendo aos parâmetros extrínsecos, a relação entre um ponto no mundo, P_W, e da

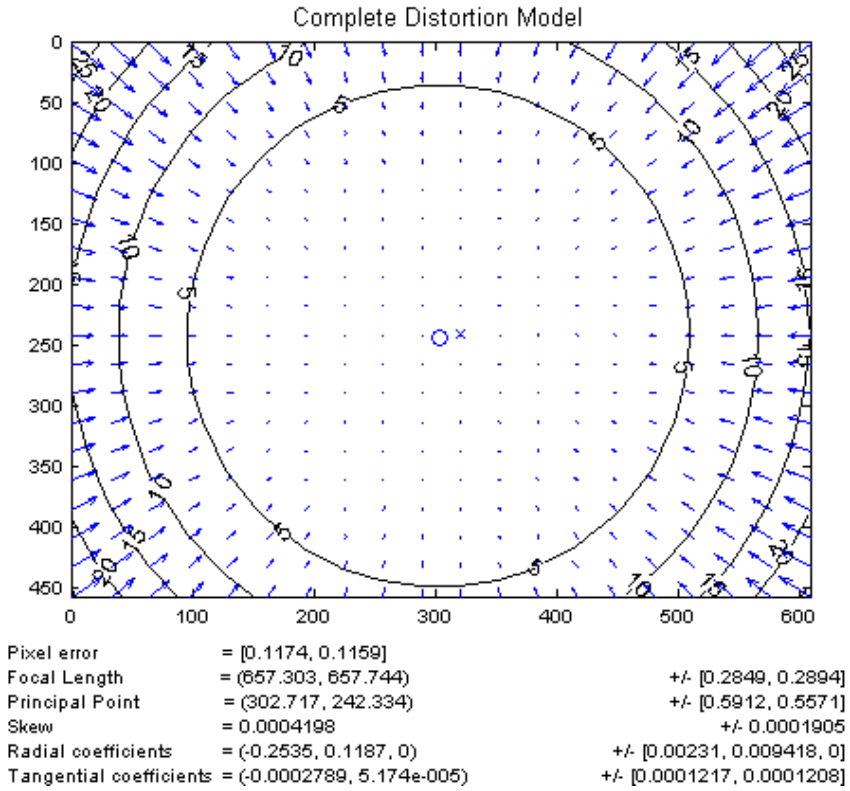


Figura 4.5: Efeito da distorção radial e tangencial (distorção total) de uma câmara. Cada seta representa o deslocamento efetivo de um pixel, devido à ditorção introduzida pela lente. A cruz indica o centro da imagem e o circulo a localização do ponto principal [29].

---

câmara, P_C, é conhecida, como apresentado na equação 4.7.

$$P_C = R \cdot (P_W - T) \tag{4.7}$$

Onde R e T dizem respeito às matrizes de rotação e translação, estas denotam a transformação geométrica do sistema de coordenadas 3-D do mundo em relação ao sistema de coordenadas 3-D da câmara. Na Figura 4.6 é exemplificada a transformação geométrica entre a câmara e o referencial global.

O processo de calibração consiste em determinar os parâmetros intrínsecos e extrínsecos. No total, existem 5 parâmetros intrínsecos e 6 extrínsecos. São necessárias pelo menos 11 equações para determinar todos estes parâmetros.

Além disso, como referido anteriormente, as lentes introduzem distorção na imagem. Para corrigir esta distorção é necessário determinar mais 5 parâmetros de distorção das lentes. Para estimar todos estes parâmetros, existem vários algoritmos, um dos algoritmos mais utilizados foi desenvolvido por Jean-Yves Bouguet [32], que será explicado de seguida.

4.3 Algoritmo para calibração de câmaras

Existem várias técnicas para a calibração de câmaras. Podendo esta ser feita recorrendo a um objeto de calibração 3-D, 2-D ou mesmo 1-D [33]. Um dos algoritmos mais utilizados

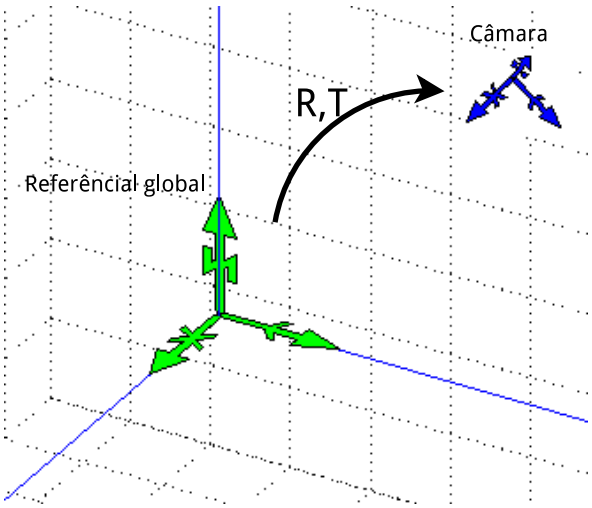


Figura 4.6: Transformação geométrica da câmara em relação ao referencial global.

para a calibração de câmaras foi introduzido por Jean-Yves Bouguet [32]. Qualquer objeto adequadamente caracterizado pode ser usado como objeto de calibração. Frequentemente utiliza-se um *chessboard* (quadrados brancos e pretos alternados) porque apresenta um padrão regular e é muito fácil detetar os seus cantos com algoritmos de visão por computador.

Os parâmetros intrínsecos são estimados de forma iterativa, fornecendo ao algoritmo múltiplas visualizações de um objeto adequadamente caracterizado são determinados os parâmetros intrínsecos da câmara.

4.4 Obtenção dos parâmetros intrínsecos

Para efetuar a calibração dos parâmetros intrínsecos recorreu-se ao método *cv::calibrateCamera*, do *OpenCv*. A função estima os parâmetros intrínsecos e extrínsecos para cada imagem do *chessboard* fornecida. O algoritmo é baseado nos algoritmos apresentados em [32] e [34].

As coordenadas 3-D e correspondências 2-D em cada visualização devem ser especificadas. Isto pode ser conseguido utilizando um objeto com uma geometria conhecida e pontos característicos facilmente detetáveis. Para o efeito, utilizou-se um *chessboard*, dado que que o *OpenCv* também disponibiliza suporte para a deteção dos cantos deste (método *cv::findChessboardCorners*). Na Figura 4.7 é exemplificado o funcionamento deste processo.

De forma a quantificar a qualidade da calibração dos parâmetros intrínsecos, efetuou-se a projeção das coordenadas 3-D dos cantos do *chessboard* no mesmo. Para cada imagem fornecida são devolvidos os parâmetros extrínsecos do *chessboard*, que correspondem a duas matrizes que definem o posicionamento do *chessboard* relativamente à câmara, uma de rotação e outra de translação.

Visto que as coordenadas 3-D dos cantos do *chessboard* são um dado conhecido, pode-se efetuar a projeção destas na imagem, recorrendo aos parâmetros extrínsecos previamente obtidos. A média da distância euclidiana entre os pontos projetados e os pixels detetados para todas as imagens fornecidas indica uma medida do erro médio, em pixels, da calibração dos parâmetros intrínsecos.

Para se obter uma medida da variação da distância entre os pontos projetados e detetados

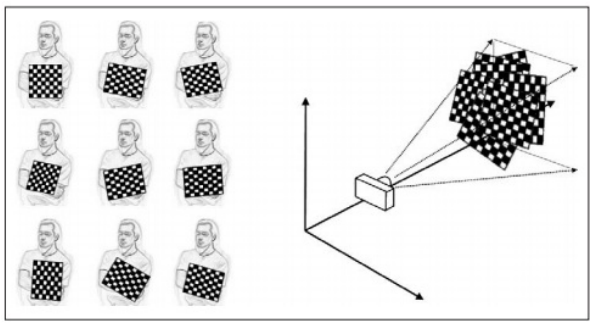


Figura 4.7: Imagens do *chessboard* retiradas com diferentes orientações [29].

calculou-se o desvio padrão. Na Figura 4.8 é possível visualizar a deteção dos pixels correspondentes aos cantos de uma imagem de um *chessboard* (indicados com um círculo) e a projeção das coordenadas 3-D destes na imagem (indicados com uma cruz).

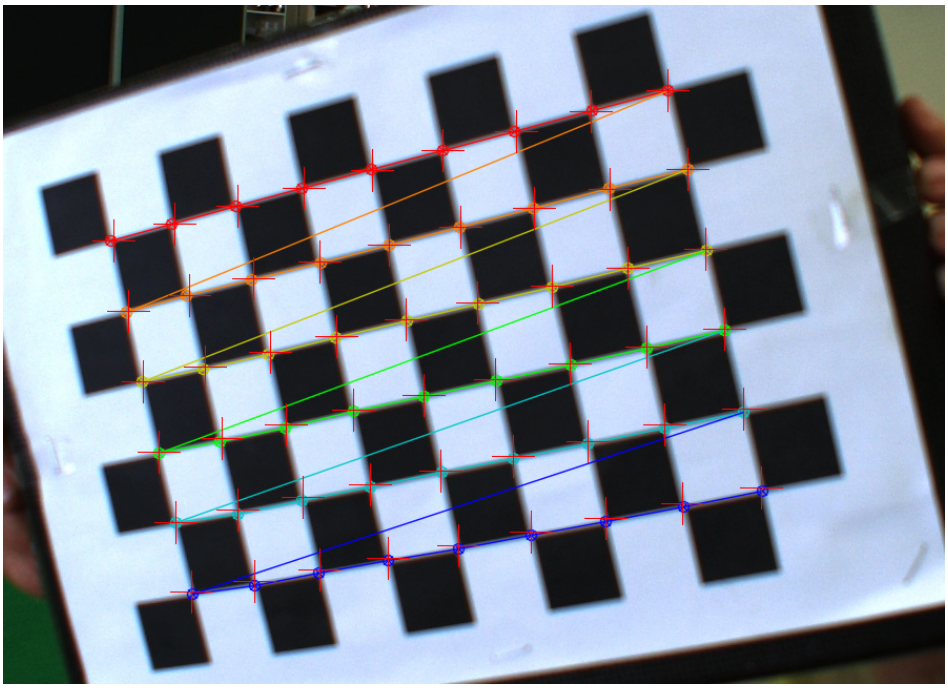


Figura 4.8: Deteção dos pixels correspondentes aos cantos do *chessboard* (indicados com um círculo) e projeção das coordenadas 3-D dos cantos na imagem (indicados com uma cruz).

Na Figura 4.9 é apresentado um fluxograma relativo ao algoritmo desenvolvido para a calibração dos parâmetros intrínsecos. Inicialmente é necessário efetuar a captura de várias imagens de um *chessoboard* com a câmara que se pretende calibrar. Visto que se conhece a distância entre cada canto do *chessboard* e assumindo que o plano de z=0 corresponde ao próprio tabuleiro do *chessboard*, as coordenadas 3-D dos cantos são conhecidas. Efetuando-se a deteção dos pixels correspondentes aos cantos do *chessboard*, para todas as imagens obtidas, e indicação das coordenadas 3-D destes ao método *cv::calibrateCamera*, são estimados os parâmetros intrínsecos da câmara.

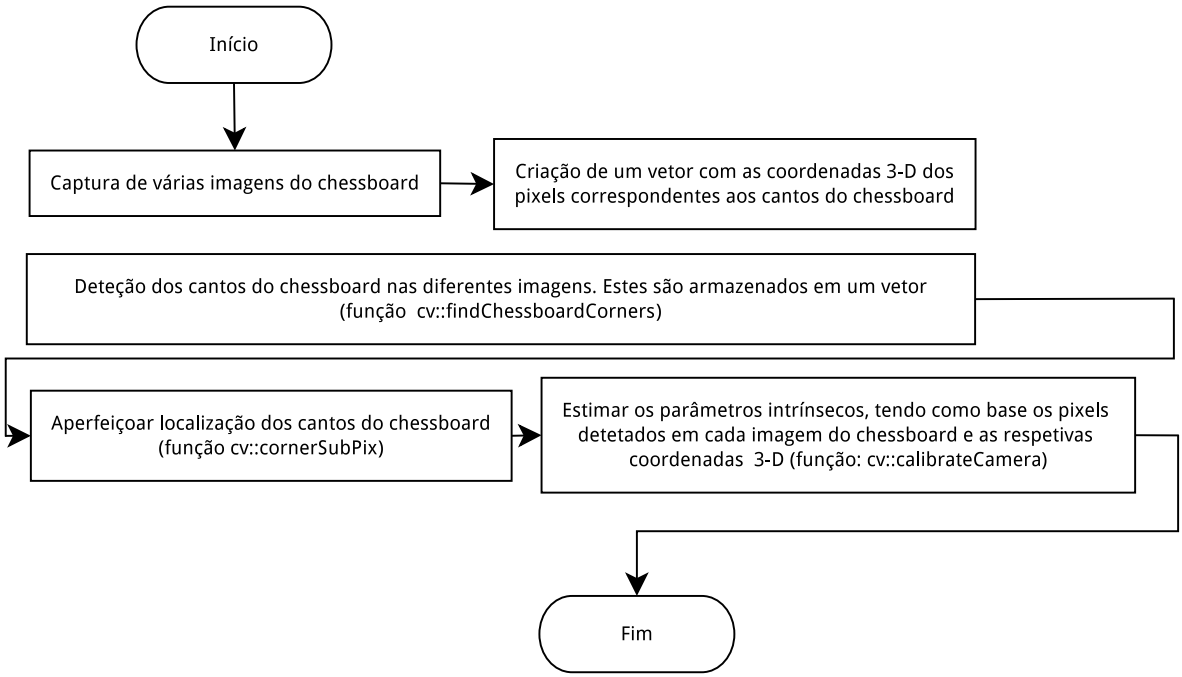


Figura 4.9: Fluxograma do algoritmo para a calibração dos parâmetros intrínsecos.

Afim de se quantificar o impacto que a calibração dos parâmetros intrínsecos tem no sistema em desenvolvimento, efetuou-se duas calibrações diferentes: uma má e uma boa calibração.

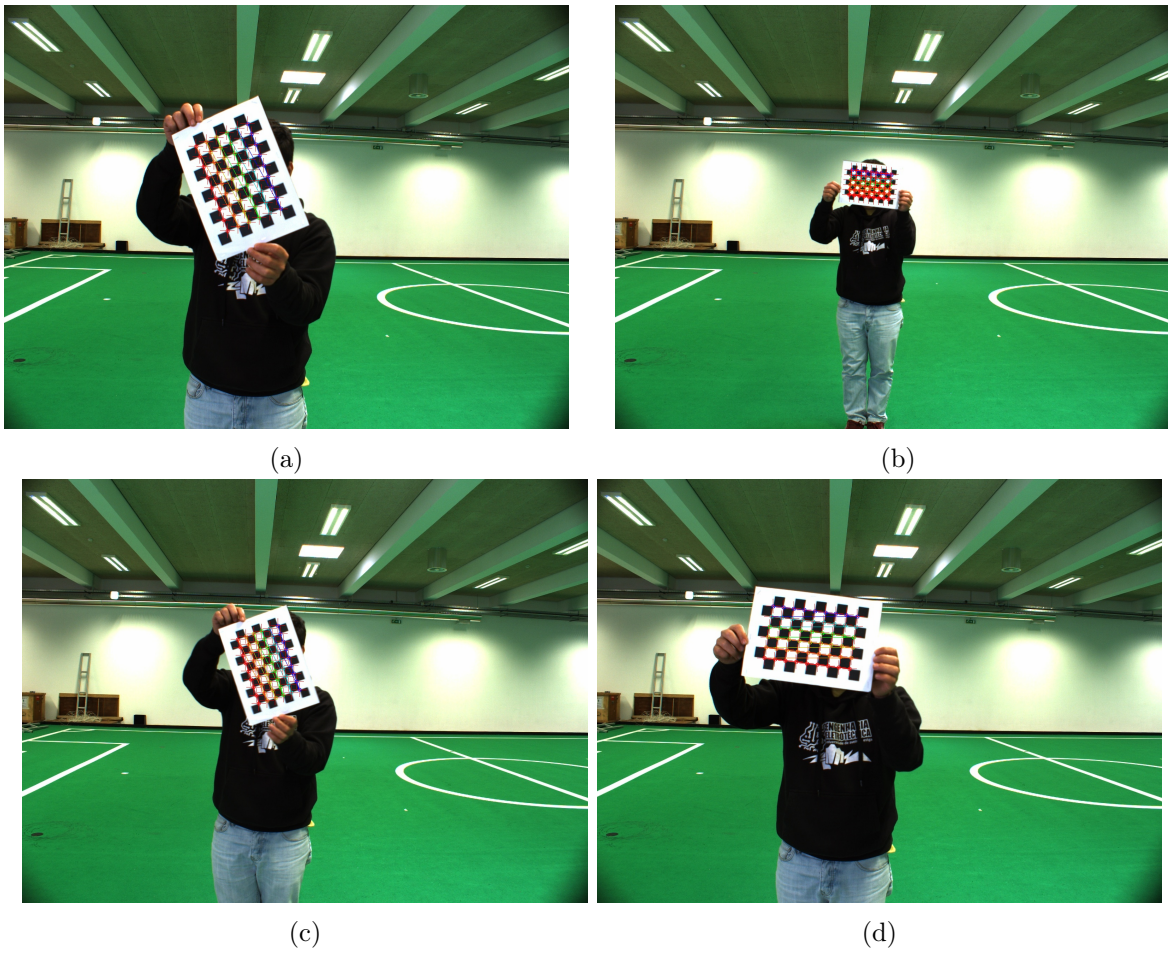
A calibração mal efetuada consistiu em retirar várias imagens do *chessboard* com diferentes orientações e com o *chessboard* a uma distância relativamente elevada da câmara. Efetuou-se a calibração para duas câmaras, afim de ser possível aplicar o método da triangulação.

A boa calibração diferenciou-se da primeira em dois fatores: teve-se a preocupação com que o *chessboard* ocupasse a maior parte da imagem possível e com que a distância entre o *chessboard* e a câmara não variasse. Nas Figuras 4.10 e 4.11 são apresentadas algumas das imagens do *chessboard* para a má e boa calibrações, respetivamente.

Na Tabela 4.1 é apresentado o erro entre a deteção dos pixels correspondentes aos cantos do *chessboard* e projeção das coordenadas 3-D dos mesmos ($\bar{x}$), bem como o desvio padrão da calibração dos parâmetros intrínsecos (σ), correspondentes às duas câmaras, para as duas calibrações efetuadas.

Tabela 4.1: Quantificação da qualidade da calibração dos parâmetros intrínsecos. $\bar{x}$: Erro médio entre a deteção dos pixels correspondentes aos cantos do *chessboard* e projeção das coordenadas 3-D dos mesmos, em pixels. σ: Desvio padrão da calibração dos parâmetros intrínsecos, em pixels.

	Erro médio (pixels)			
Número da câmara	1		2	
Tipo de erro	$\bar{x}$	σ	$\bar{x}$	σ
Calibração mal efetuada	0.29	3.21	0.35	3.86
Calibração bem efetuada	1.45	8.56	0.60	4.27



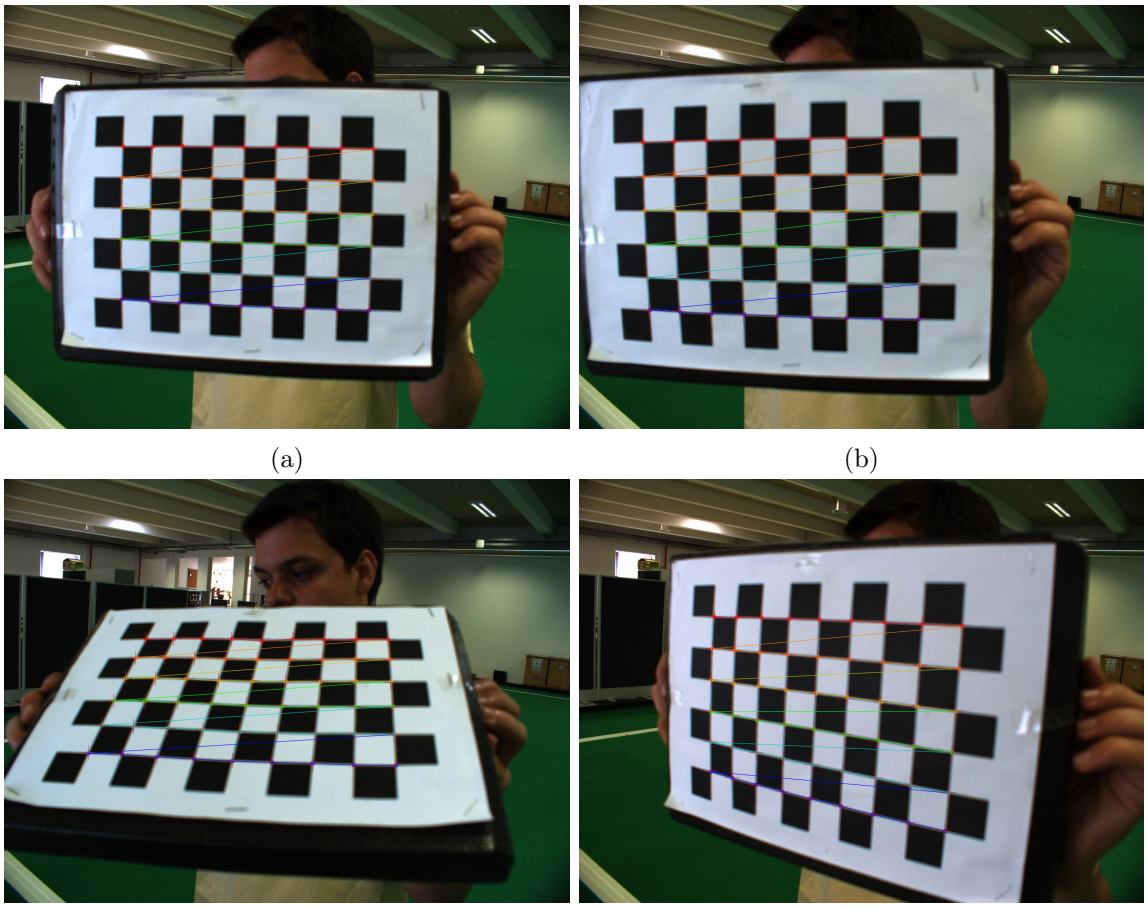
(a) (b)

(c) (d)

Figura 4.10: Exemplo de uma má calibração dos parâmetros intrínsecos. Na figura encontram-se algumas das imagens fornecidas ao algoritmo para a calibração dos parâmetros intrínsecos.

Face à primeira calibração, o erro médio entre a deteção dos pixels correspondentes aos cantos do *chessboard* e a projeção das coordenadas 3-D no mesmo, bem como o desvio padrão são bastante inferiores aos da segunda. Isto justifica-se porque nas imagens desta calibração, o *chessboard* encontra-se bastante mais longe da câmara do que na segunda calibração, pelo que a diferença entre a deteção dos pixels dos cantos do *chessboard* e a projeção dos pontos 3-D neste é bastante mais inferior. Desta forma, o interesse destas medições é relativo, dado que valores inferiores destas medidas nem sempre correspondem à melhor calibração. Todavia, para calibrações dos parâmetros intrínsecos efetuadas de forma semelhante, em que o *chessboard* ocupa grande parte da imagem, esta ferramenta pode vir a ser útil para se obter uma medida da qualidade das mesmas.

Na Secção 5.3 é efetuado um estudo acerca do impacto da calibração dos parâmetros intrínsecos na obtenção das coordenadas 3-D de uma bola no campo (objeto de interesse), onde são utilizados os dois exemplos de calibrações efetuadas neste capítulo, sendo assim possível verificar o impacto da calibração dos parâmetros intrínsecos no sistema em desenvolvimento.



(a) (b)

(c) (d)

Figura 4.11: Exemplo de uma boa calibração. Na figura encontram-se algumas das imagens fornecidas ao algoritmo para a calibração dos parâmetros intrínsecos.

Parâmetros extrínsecos

Os parâmetros extrínsecos definem a posição da câmara em relação a um referencial global. Na equação 4.8 é apresentada a expressão matemática que indica o posicionamento da câmara em relação a um referencial global.

$$C = -R^{\intercal} \cdot T \tag{4.8}$$

À semelhança dos parâmetros intrínsecos, também os parâmetros extrínsecos foram obtidos recorrendo ao *OpenCv* (método *cv::solvePnP*). Efetuando-se uma correspondência entre pixels na imagem e coordenadas 3-D, consegue-se determinar o posicionamento da câmara em relação ao referencial global.

Para se efetuar a calibração dos parâmetros extrínsecos, é necessário indicar a correspondência entre pixels e coordenadas 3-D. Deste modo, desenvolveu-se uma aplicação em que sempre que se clicar num pixel da imagem é efetuado um *Zoom in* em redor deste, sendo assim possível definir o posicionamento do pixel com uma maior precisão, como apresentado na Figura 4.12.

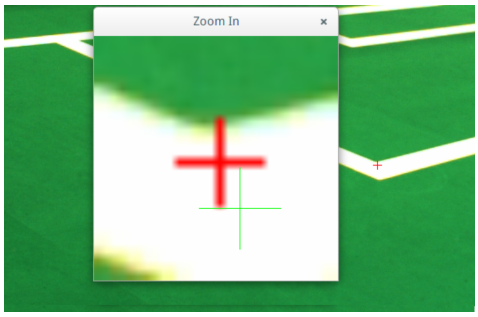


Figura 4.12: Correspondência entre pixels e coordenadas 3-D com opção de *Zoom in*.

Na Figura 4.13 é apresentado um fluxograma do algoritmo desenvolvido para a calibração dos parâmetros extrínsecos.

A correspondência entre pixels e coordenadas do mundo é efetuada manualmente, clicando com o rato no pixel pretendido da imagem e indicação das suas coordenadas 3-D. No Anexo C são apresentadas coordenadas do campo em pontos específicos.

4.5 Ferramentas de apoio à calibração

Depois de se efetuar a calibração das câmaras, desenvolveram-se diversas ferramentas para verificação da qualidade desta, descriminadas de seguida.

Começou-se por construir um modelo do campo de futebol presente no laboratório. A escala utilizada foi de 1 pixel para 20 milímetros. Pretende-se detetar os pixels correspondentes às linhas do campo de futebol e efetuar a projeção destes no plano do campo previamente construído. Comparando a distância entre as linhas desenhadas no modelo do campo e as linhas efetivamente detetadas, obtém-se uma medida do erro devido principalmente à calibração.

De notar que para a deteção das linhas do campo recorreu-se à biblioteca UAVision, abordada na Secção 2.3.

Para quantificar o erro devido da calibração construiu-se uma LUT ,[1] que consiste numa estrutura de dados. Para cada pixel da imagem do modelo do campo é calculada a distância euclidiana com o pixel branco mais próximo, sendo armazenado o valor obtido no índice do pixel em análise. Na Figura 4.14 é apresentada em (a) um exemplo do modelo com a projeção das linhas detetadas e em (b) o correspondente mapa de distâncias para quantificação do erro.

Na Figura 4.15 é apresentada a projeção das linhas do campo para as duas calibrações efetuadas. Para a má calibração verifica-se que alguns pixels das linhas do campo têm um erro muito acentuado na projeção, não se chegando a perceber a que zona do campo estes realmente pertencem. Estima-se que estas correspondam às linhas mais afastadas da câmara, o facto destes não estarem a ser bem projetados pode significar uma má estimativa dos parâmetros de distorção da lente. Este fenómeno foi estudado com mais detalhe na Secção 4.1, onde se verificou que em zonas mais afastadas do centro da imagem o efeito da distorção da lente é mais acentuado. Já na boa calibração verifica-se que este fenómeno não acontece, pelo que se conclui que a calibração dos parâmetros intrínsecos foi bem sucedida.

De forma a verificar o impacto que a calibração dos parâmetros intrínsecos, face aos parâmetros extrínsecos, tem na calibração das câmaras, realizou-se o seguinte teste: efetuou-se a

[1] Acrónimo de *Look-Up Table*.

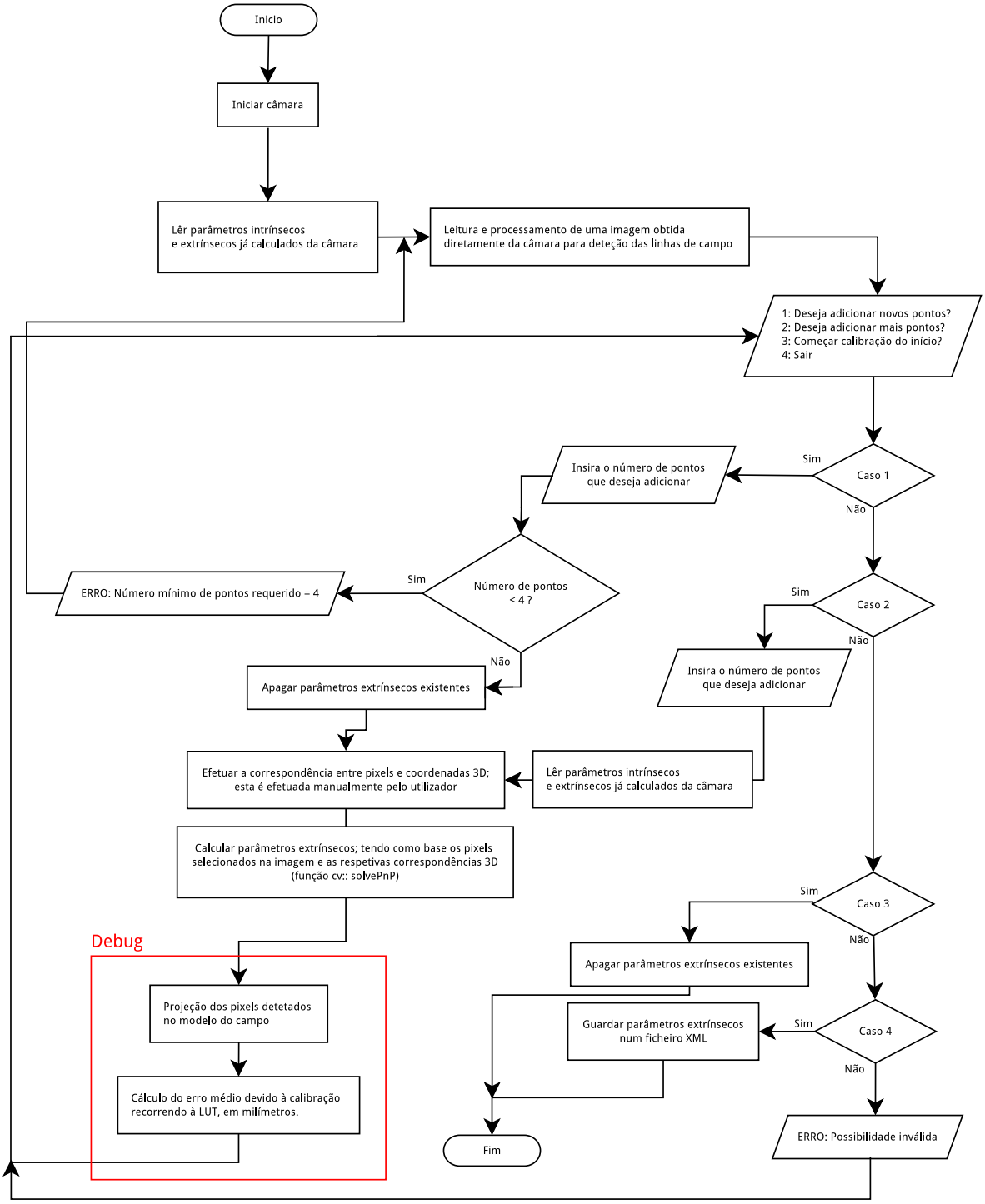
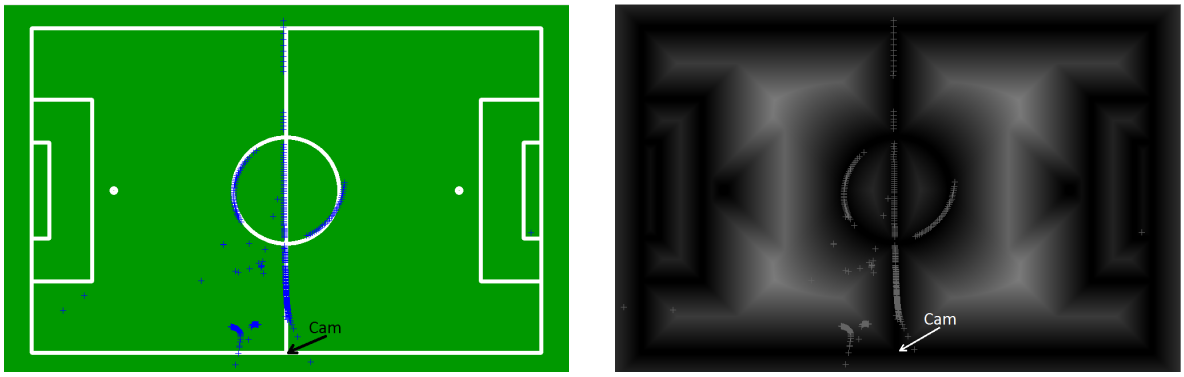


Figura 4.13: Fluxograma do algoritmo desenvolvido para calibração dos parâmetros extrínsecos.

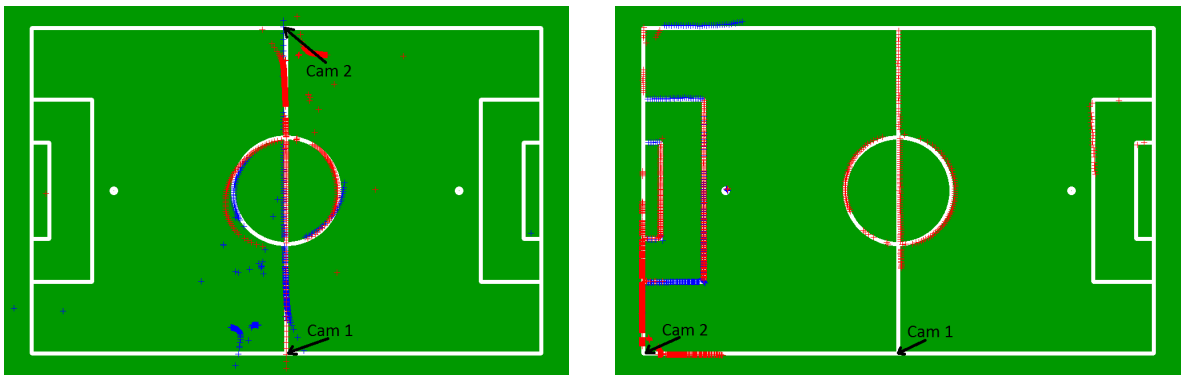
calibração dos parâmetros intrínsecos e extrínsecos, utilizando uma má calibração dos parâmetros intrínsecos na primeira fase.

De seguida, realizou-se uma boa calibração dos parâmetros intrínsecos e utilizaram-se os



(a) (b)

Figura 4.14: (a) Projeção das linhas brancas detetadas no modelo do campo de futebol. (b) Obtenção do erro médio recorrendo ao mapa de distâncias euclidianas. O erro médio é de 12.05 pixels, 241.05 mm. Neste exemplo utilizou-se a calibração dos parâmetros intrínsecos mal efetuada.



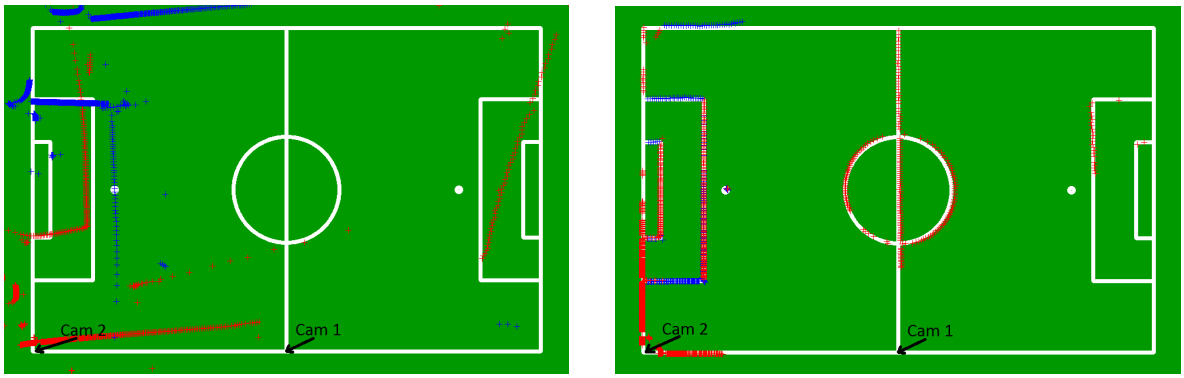
(a) (b)

Figura 4.15: (a) Projeção das linhas brancas detetadas no modelo do campo de futebol para a calibração mal efetuada. (b) Projeção das linhas brancas detetadas no modelo do campo de futebol para a calibração bem efetuada.

---

mesmos parâmetros extrínsecos, para as duas calibrações efetuadas. Efetuou-se a deteção das linhas do campo e a projeção destas no modelo do campo criado. Na Figura 4.16 são apresentados os resultados obtidos para as duas calibrações dos parâmetros intrínsecos efetuadas, que dizem respeito à diferença entre o posicionamento real das linhas do campo e a projeção das linhas do campo detetadas no modelo do campo.

Conclui-se assim que a calibração dos parâmetros intrínsecos é bastante mais crítica que a dos extrínsecos. Pelo que é necessário ter uma preocupação acrescida aquando na calibração dos parâmetros intrínsecos, procurando-se obter a melhor calibração possível.



(a) (b)

Figura 4.16: (a) Projeção das linhas do campo para a primeira calibração efetuada, o erro médio da calibração da primeira câmara é de 18.92 pixels, 378.47 mm, e da segunda câmara é de 22.75 pixels, 455.03 mm. (b) Projeção das linhas do campo para a segunda calibração efetuada. O erro médio proveniente da calibração para a primeira câmara é de 0.35 pixels, 6.97 mm, e da segunda câmara é de 0.57 pixels, 11.41 mm. Neste exemplo utilizou-se a calibração dos parâmetros intrínsecos mal efetuada, em (a) e bem efetuada, em (b), mantendo-se os parâmetros extrínsecos para as duas calibrações.

Capítulo 5

Sistema de monitorização de objetos em 3-D

Neste capítulo é apresentado o algoritmo desenvolvido para efetuar a monitorização de objetos com base em múltiplas câmaras RGB. Nesta fase o objeto em monitorização diz respeito à bola de futebol, dado já existir implementado software para a deteção desta, anteriormente abordado na Secção 2.3.

Na Secção 2.2 foram apresentados diferentes métodos existentes para efetuar a deteção 3-D de objetos a partir de múltiplas câmaras, verificando-se que o método mais vantajoso para a aplicação é o da triangulação, pois permite efetuar o *tracking* de objetos com bastante precisão para amplas *baselines* entre as câmaras. Assim, vai-se recorrer a este método para efetuar a monitorização de objetos em 3-D com as múltiplas câmaras.

5.1 Visualizador em VTK

Para se ilustrar o posicionamento da bola e das câmaras no campo de futebol, bem como o cálculo da interseção entre os vetores, recorreu-se à biblioteca VTK. Esta é uma biblioteca gratuita para computação gráfica, modelação, processamento de imagem, entre outros [35]. Fornece um conjunto de ferramentas em C ++ que permite aos utilizadores construirem aplicações pela combinação de diferentes objetos numa única aplicação.

A aplicação desenvolvida mostra o posicionamento das câmaras no campo. Além disso, permite visualizar a projeção dos vetores em direção à bola, para as diferentes câmaras, bem como a localização da bola no campo, como ilustrado na Figura 5.1.

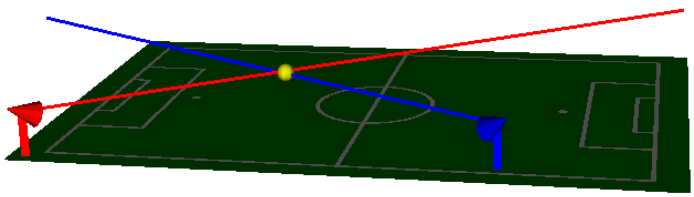


Figura 5.1: Visualização do processo de deteção 3-D da bola utilizando o programa desenvolvido em VTK. Na imagem são mostradas as câmaras posicionadas no campo, bem como os vetores projetados a partir destas em direção à bola e o posicionamento 3-D da bola.

Tal como se apresentou no Secção 2.2, na triangulação entre dois vetores pode não se verificar a interseção entre estes, pois a deteção da bola nas imagens pode não ser suficientemente precisa para que os vetores se intersetem. Para contornar este problema, em vez de se calcular a interseção entre dois vetores, calcula-se o ponto mais próximo entre estes.

Recorrendo ao método *vtkLine::DistanceBetweenLineSegments*, incluído no VTK, calculou-se a mínima distância entre dois vetores. Este recebe como parâmetros de entrada dois vetores 3-D, devolvendo as coordenadas correspondentes à mínima distância entre estes.

5.2 Algoritmo desenvolvido para a obtenção das coordenadas 3-D

O algoritmo desenvolvido para a obtenção das coordenadas 3-D da bola foi baseado no método da triangulação entre dois vetores, tal como referido anteriormente. Na Figura 5.2 é apresentado o fluxograma do algoritmo.

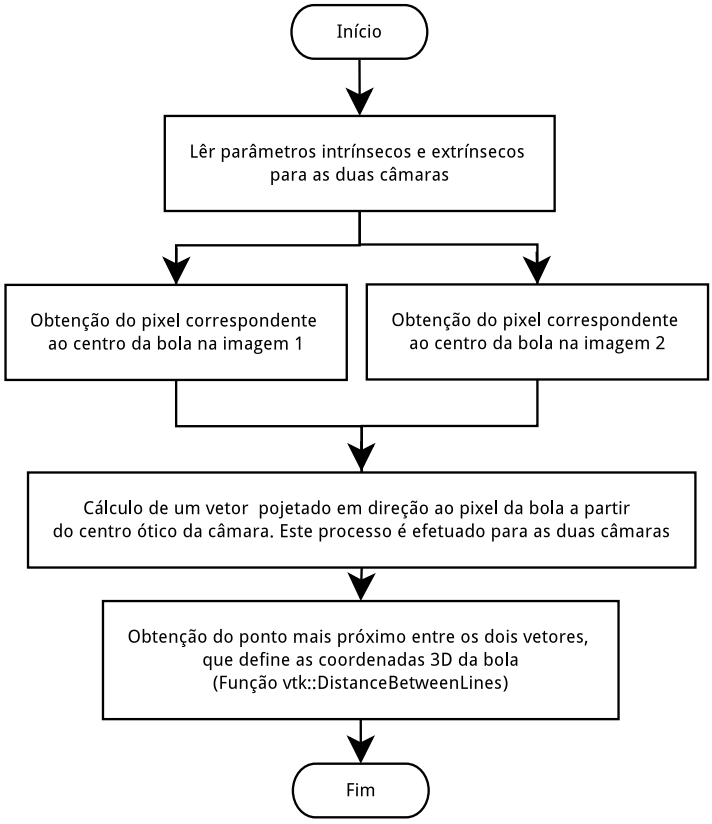


Figura 5.2: Fluxograma do algorítmo desenvolvido para a obtenção das coordenadas 3-D da bola.

Na Figura 5.3 (a) é mostrada a deteção da bola numa imagem, esta encontra-se circunscrita a vermelho. Em (b) é apresentada a projeção de um vetor tridimensional em direção ao pixel correspondente ao centro da bola. A interseção deste vetor com o plano do chão define as coordenadas 3-D da bola.

Verifica-se que as coordenadas 3-D da bola obtidas pelo sistema não correspondem às coordenadas reais desta. Isto justifica-se devido à altura da bola, sendo que o pixel correspondente à deteção desta diz respeito ao seu centro, encontrando-se acima do plano do chão.

Uma vez que a altura da bola é um dado conhecido, seria possível corrigir o erro na obtenção das suas coordenadas 3-D devido à sua altura. Todavia, num jogo de futebol robótico a bola nem sempre se encontrará no plano do chão, nesses casos, esta aproximação deixa de ser válida. Assim, é necessário introduzir uma segunda câmara, o ponto correspondente à interseção dos dois vetores projetados pelas duas câmaras define as coordenadas 3-D da bola.

Na Figura 5.4 é mostrada a projeção de dois vetores no espaço, para duas câmaras distintas. O ponto mais próximo destes define as coordenadas 3-D da bola.

(a) (b)

Figura 5.3: (a) Deteção da bola recorrendo à biblioteca UAVision. (b) Projeção de um vetor, desde o centro ótico da câmara em direção à bola.

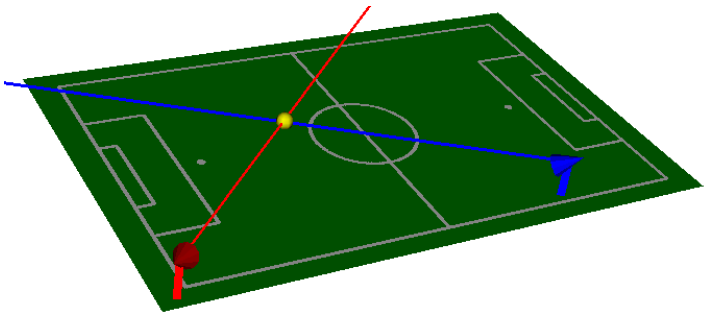


Figura 5.4: Projeção de dois vetores em direção à bola. O ponto da interseção destes define as coordenadas 3-D da bola.

5.3 Resultados experimentais com objetos estáticos

Nesta secção é estudado o impacto da calibração na obtenção das coordenadas 3-D da bola. Além disso, é efetuado um estudo acerca da influência da distância da bola em relação às câmaras, no erro do sistema.

Um dos problemas para efetuar a validação dos resultados obtidos é a falta de um sistema externo de *ground truth*, que permita efetuar a validação dos dados obtidos com precisão. Como tal, a validação é efetuada manualmente, colocando-se a bola em posições definidas no campo, onde as coordenadas 3-D são conhecidas, comparando as coordenadas onde a bola foi colocada com as coordenadas 3-D obtidas pelo sistema.

5.3.1 Efeito da calibração

Para se verificar o impacto que a calibração tem na obtenção das coordenadas da bola foram colocadas duas câmaras posicionadas em zonas distintas do campo. De seguida efetuou-se a calibração dos parâmetros extrínsecos das mesmas e colocou-se a bola em diferentes zonas do campo, onde as coordenadas 3-D são conhecidas, e obteve-se as coordenadas da bola recorrendo ao sistema em desenvolvimento. Este processo foi realizado para as duas calibrações dos parâmetros intrínsecos referidas no Capítulo 4: uma mal e outra bem efetuada. Na Figura 5.5 é apresentada uma fotografia com o posicionamento das câmaras no campo para a realização deste teste.

Figura 5.5: Fotografia da configuração das câmaras utilizada para efetuar o estudo do impacto que a calibração dos parâmetros intrínsecos tem na obtenção das coordenadas 3-D da bola.

Na Figura 5.6 são apresentadas as coordenadas da bola medidas no campo (pontos a azul) e obtidas pelo sistema (pontos a vermelho), para as má e boa calibrações. Na Tabela 5.1 são apresentadas as distâncias, em mm, entre os valores medidos e obtidos, para as duas calibrações.

Nas Figuras 5.7 e 5.8 são apresentadas algumas das imagens da interface gráfica desenvolvida, recorrendo ao VTK, é mostrado o processo de obtenção das coordenadas 3-D da bola, com a má e boa calibrações, respetivamente. São mostradas as projeções dos vetores para as duas câmaras, bem como a localização da bola no campo, sendo que esta corresponde às coordenadas da mínima distância entre os dois vetores, projetados pelas duas câmaras.

Verifica-se que, para a calibração bem efetuada, o erro entre as coordenadas da bola medidas e obtidas pelo sistema é bastante inferior ao da mal efetuada. Dada a elevada disparidade dos resultados obtidos reforça-se as conclusões obtidas anteriormente, que efetuar uma boa calibração dos parâmetros intrínsecos é essencial para o projeto em desenvolvimento, como esperado.

Nos testes que se seguem utilizou-se a boa calibração dos parâmetros intrínsecos, anteriormente efetuada no Capítulo 4.

5.3.2 Efeito da distância da bola em relação às câmaras

Para verificar o impacto que a distância da bola, em relação às câmaras, tem na obtenção das coordenadas 3-D desta, colocou-se duas câmaras em dois cantos do campo. De seguida foram retiradas várias imagens da bola em posições conhecidas no campo, em que a bola vai-se afastando gradualmente das câmaras.

Efetuando-se a comparação entre a posição 3-D obtida pelo sistema e a posição real em que a bola se encontra, relativamente às câmaras, pretende-se estudar a precisão das coordenadas 3-D da bola obtidas pelo sistema em função da distância desta às câmaras. Na Figura 5.9 é apresentada a configuração das câmaras utilizada e na Figura 5.10 são apresentadas as coordenadas da bola medidas no campo (pontos a azul) e obtidos pelo sistema (pontos a vermelho). Na Tabela 5.2 são apresentadas as distâncias, em mm, entre os valores medidos e obtidos.

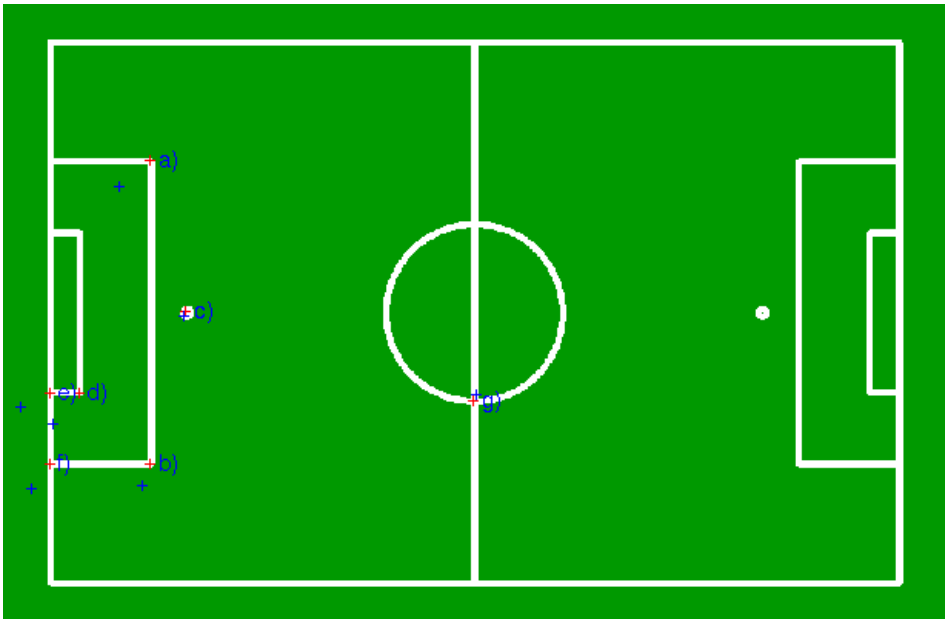
Verifica-se que à medida que a bola se vai afastando das câmaras o erro não é linear. Não se verificando assim nenhuma tendência para que o erro aumente à medida que a bola se vai afastando das câmaras, que seria de esperar. De salientar que a bola encontra-se a distâncias bastante elevadas da câmara, indo desde os 2.880 metros, ponto a), até aos 17.875 metros, ponto p), obtendo-se erros bastante reduzidos nesta gama de distâncias.

5.3.3 Teste geral

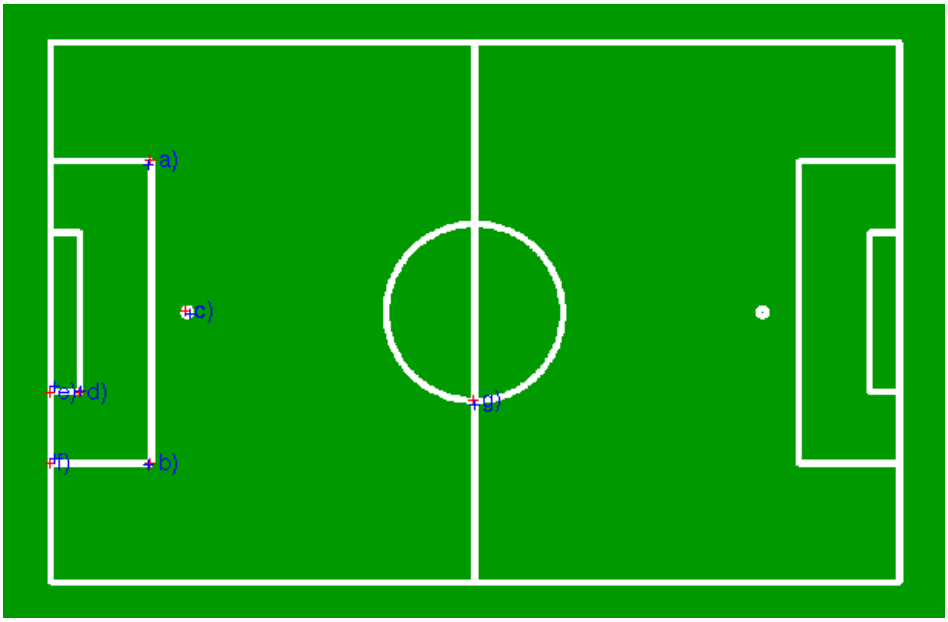
Efetuou-se uma outra experiência, que visa efetuar uma avaliação global do sistema. Colocou-se duas câmaras posicionadas no campo, com uma ampla *baseline* entre estas. À semelhança dos exemplos anteriores, colocou-se a bola em posições conhecidas no campo e obteve-se as suas coordenadas, recorrendo ao sistema em desenvolvimento. Na Figura 5.11 é apresentada uma fotografia do posicionamento das câmaras para a realização desta experiência. Utilizou-se a boa calibração dos parâmetros intrínsecos neste teste.

Na Figura 5.12 são apresentadas as coordenadas da bola medidas no campo (pontos a azul) e obtidas pelo sistema (pontos a vermelho), para o teste efetuado. Na Tabela 5.3 são apresentadas as distâncias, em mm, entre os valores medidos e obtidos.

Verifica-se que o erro na obtenção das coordenadas 3-D da bola é bastante reduzido, sendo este variável na ordem dos milímetros. Visto que existe um erro associado à colocação das posições da bola nos diferentes pontos do campo, conclui-se que com este método não é possível quantificar o erro proveniente do sistema abaixo dos milímetros. Para se obter uma melhor quantificação do erro seria necessário recorrer a um sistema externo de *ground truth*.



(a) Má calibração.



(b) Boa calibração.

Figura 5.6: Testes efetuados para a obtenção das coordeanadas 3-D da bola com uma má e boa calibrações. São indicadas as coordenadas 3-D da bola medidas no campo (pontos a vermelho) e coordenadas obtidas pelo sistema (pontos a azul).

Tabela 5.1: Distância euclidiana entre as coordenadas medidas no campo e as coordenadas obtidas, para as calibrações mal e bem efetuadas, em mm.

	a)	b)	c)	d	e)	f)	g)	Média
Má calibração	393.67	365.91	37.96	688.25	664.60	550.12	119.77	402.90
Boa calibração	6.16	34.93	67.25	36.64	143.95	131.82	94.16	73.56

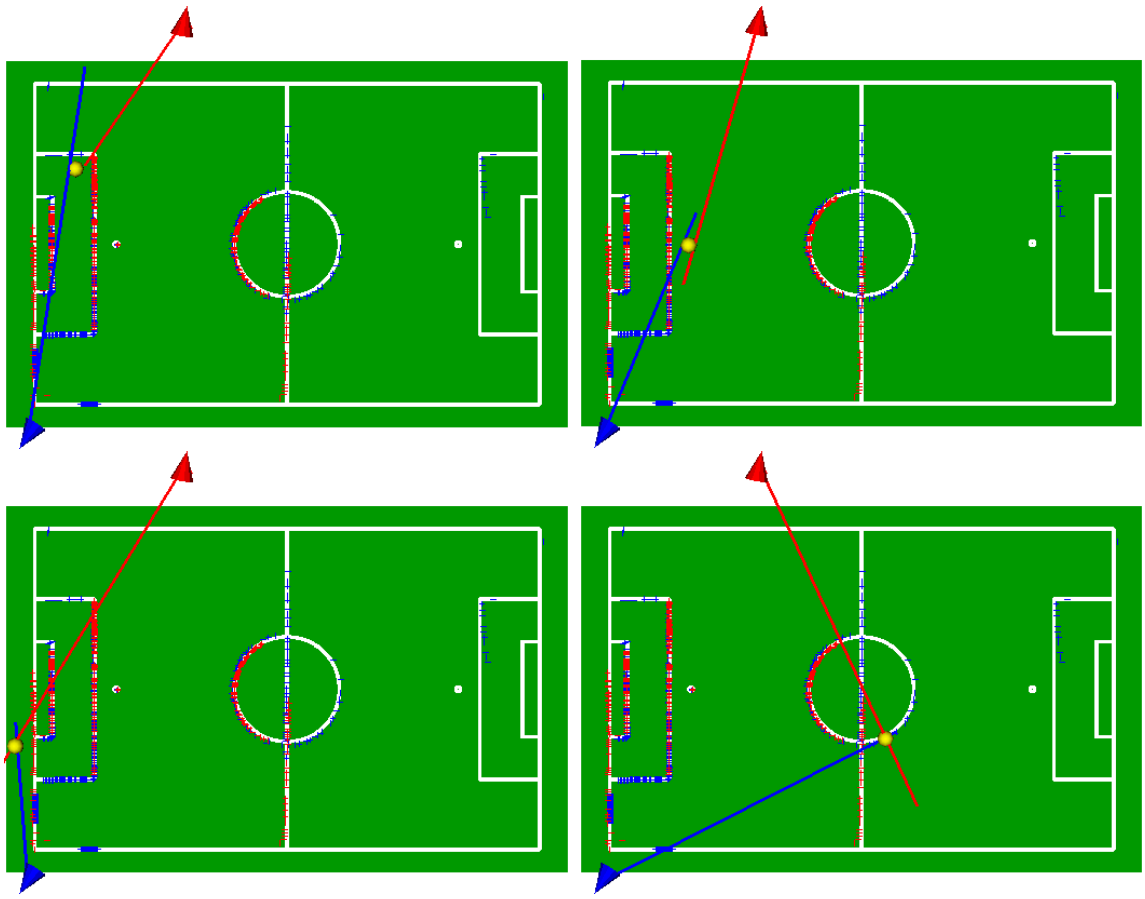


Figura 5.7: Obtenção das coordenadas 3-D da bola recorrendo ao método da triangulação, exemplos retirados para a calibração dos parâmetros intrínsecos mal efetuada. As posições da bola correspondem a alguns pontos indicados na Figura 5.6 (a).

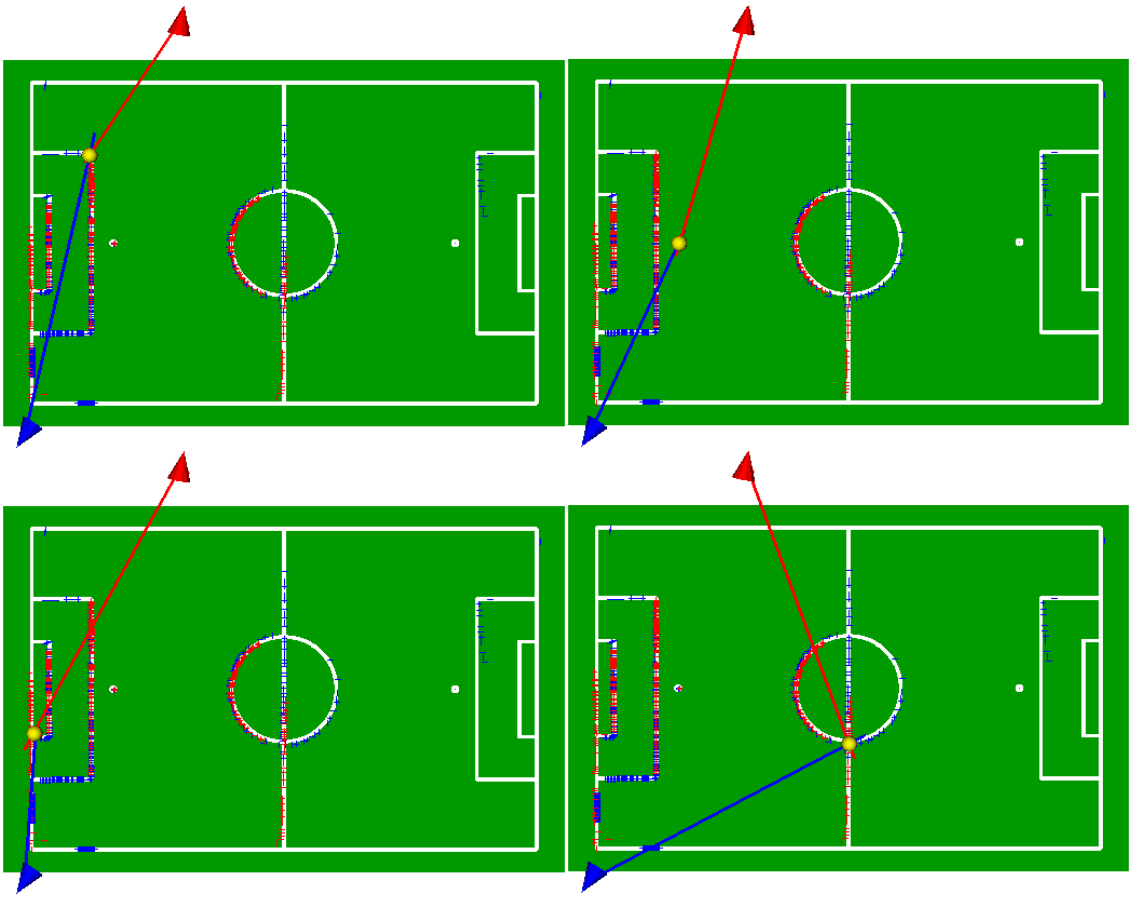


Figura 5.8: Obtenção das coordenadas 3-D da bola recorrendo ao método da triangulação, para a boa calibração dos parâmetros intrínsecos efetuada. As posições da bola correspondem a alguns pontos indicados na Figura 5.6 (b).

Figura 5.9: Fotografia da configuração das câmaras utilizada para efetuar o estudo do impacto que a distância da bola às câmaras tem na obtenção das suas coordenadas 3-D.

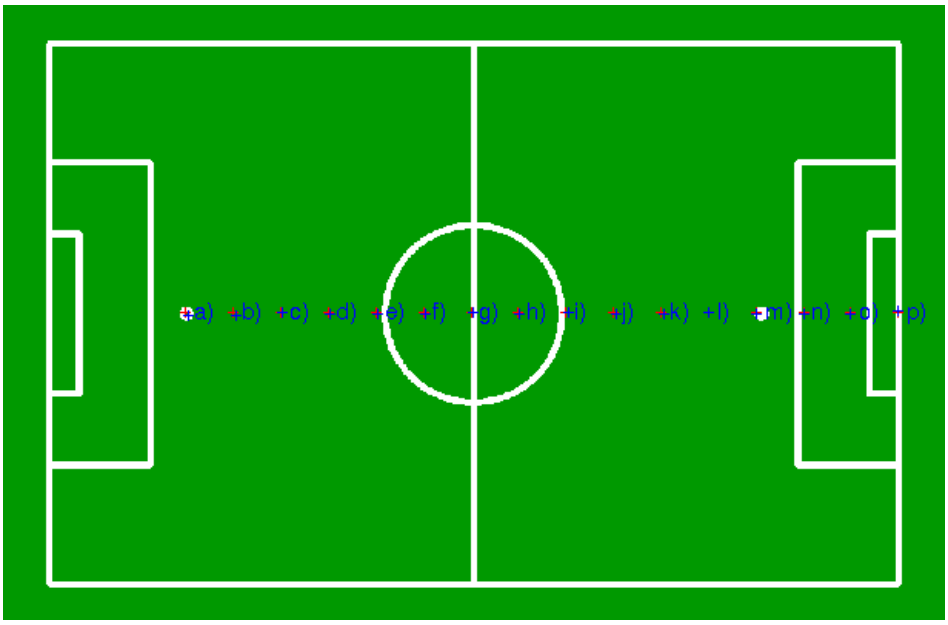


Figura 5.10: Coordenadas da bola medidas no campo (pontos a azul) e obtidas (pontos a vermelho).

Tabela 5.2: Distância euclidiana entre as coordenadas medidas no campo e as coordenadas obtidas, em mm.

a)	b)	c)	d)	e)	f)	g)	h)
50.30	40.03	37.90	41.65	40.16	47.12	5.15	31.52
i)	j)	k)	l)	m)	n)	o)	p)
48.15	69.09	56.01	21.11	14.21	22.73	9.30	7.53
Média: 33.88							

Figura 5.11: Fotografia do posicionamento das câmaras utilizada para a realização do teste geral, que visa efetuar uma avaliação global do sistema no que diz respeito à precisão da obtenção das coordenadas 3-D da bola.

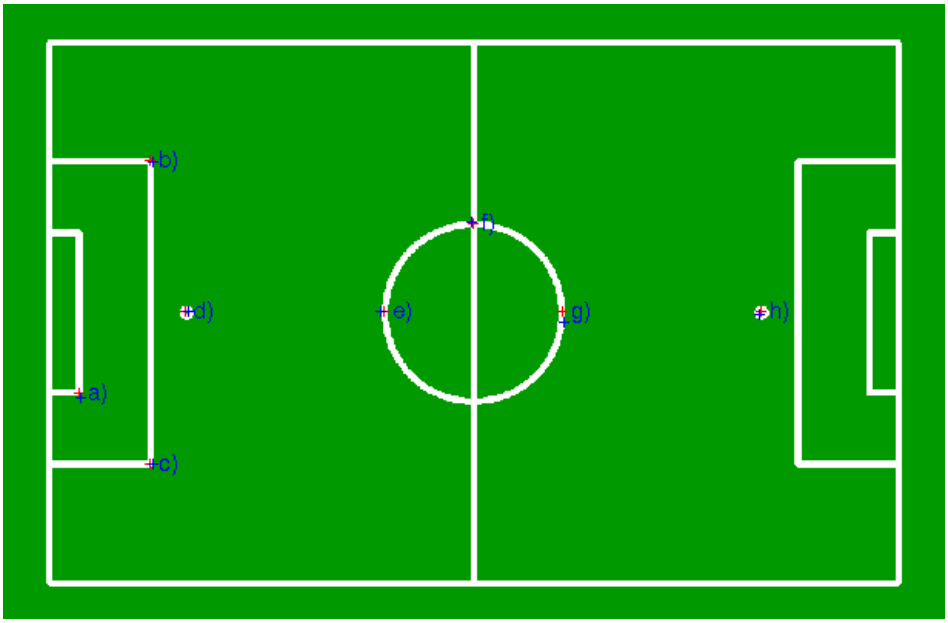


Figura 5.12: Coordenadas da bola medidas no campo (pontos a azul) e obtidas (pontos a vermelho), que corresponde à validação da obtenção das coordenadas 3-D da bola, em termos de precisão, para o teste geral.

Tabela 5.3: Distância euclidiana entre as coordenadas medidas no campo e as coordenadas obtidas, em mm.

a)	b)	c)	d)	e)	f)	g)	h)	Média
10.53	70.74	67.59	55.83	72.01	35.24	33.04	44.25	48.65

Capítulo 6

Integração de múltiplas câmaras no sistema

De forma a conseguir-se obter a posição 3-D da bola em todo o campo deve-se garantir que a bola é vista pelo menos por duas câmaras, para que seja possível aplicar o método da triangulação.

Para se conhecer as zonas do campo vistas pelas diferentes câmaras, foi desenvolvido um algoritmo que devolve um mapeamento do campo de visão destas, tendo como base o seu posicionamento.

Inicialmente, o algoritmo estabelece um conjunto de pontos 3-D ao longo de todo o campo. Depois, tendo como base os parâmetros extrínsecos da câmara, é efetuada a projeção dos pontos 3-D na imagem da câmara. Os pontos 3-D cuja projeção se encontre dentro da imagem são vistos por essa mesma câmara. Na Figura 6.1 é apresentado um exemplo do resultado da projeção de um ponto 3-D numa imagem. Os pontos a serem projetados são o resultado do varrimento de pontos 3-D ao longo de todo o campo, representados pela grelha a preto na figura.

O resultado do algoritmo é uma imagem em pseudo cor. Para cada câmara foi atribuída uma cor, sendo que para todos os pontos 3-D que pertençam a uma dada câmara é pintada na imagem do campo a cor correspondente a esta. Para os pontos que são vistos por duas, ou mais, câmaras, é efetuada uma sobreposição de cores, sendo possível obter assim uma visualização das zonas do campo vistas por múltiplas câmaras. Na Figura 6.2 é apresentada em (a) uma imagem com o campo de visão de duas câmaras posicionadas em zonas distintas do campo, como mostrado em (b).

Foi testada uma configuração base com três câmaras no campo. Na Figura 6.3 é apresentado em (a) o posicionamento das diferentes câmaras e em (b) é mostrado o campo de visão destas. Para a presente configuração, uma percentagem bastante reduzida do campo é visto por apenas uma câmara.

Visto que uma percentagem bastante reduzida do campo é vista por apenas uma câmara, pode-se à priori assumir que três câmaras é o suficiente para se efetuar a monitorização do jogo de futebol robótico. Todavia, não se estão a considerar dois fatores fundamentais:

1. Durante o jogo de futebol ocorrem oclusões entre os robôs e a bola. Nestas condições, e caso a zona do campo onde se encontre a bola esteja apenas a ser monitorizada por duas câmaras, não é possível aplicar o método da triangulação.

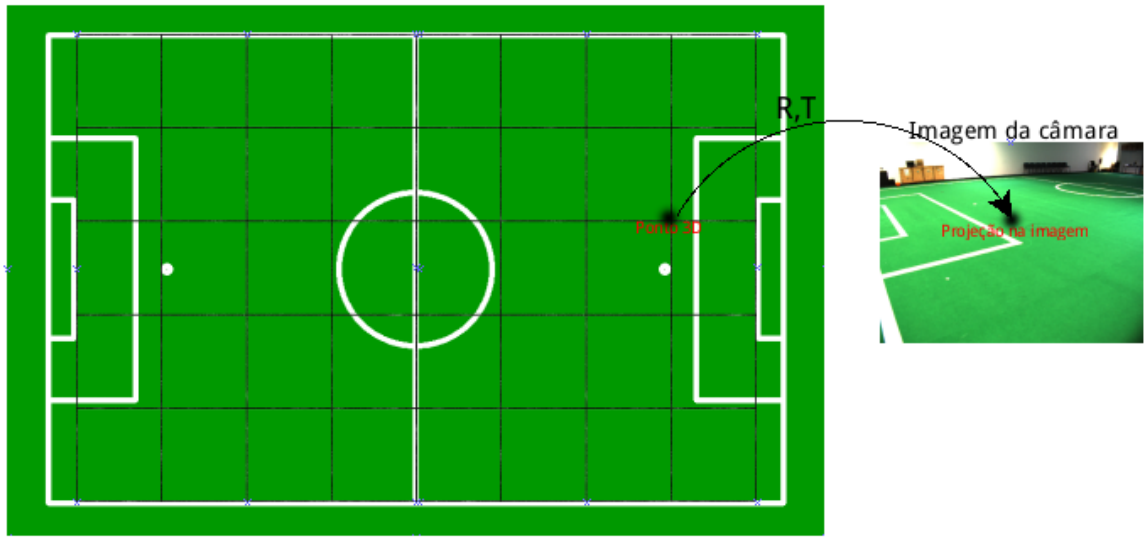
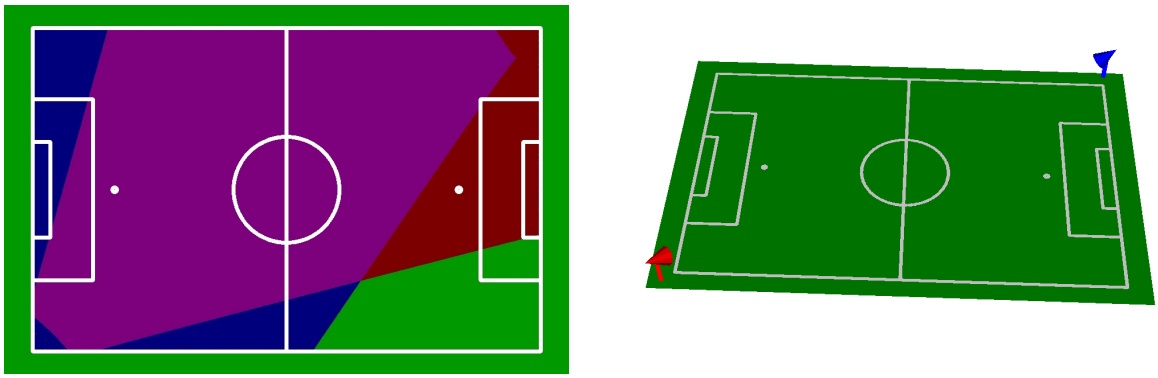


Figura 6.1: Projeção de pontos 3-D numa imagem. Os pontos a serem projetados são o resultado do varrimento de pontos 3-D ao longo de todo o campo, tendo como base uma dada resolução, representada pela grelha a preto. A projeção é efetuada através dos parâmetros extrínsecos da câmara. A resolução é um parâmetro de entrada, definido pelo utilizador.



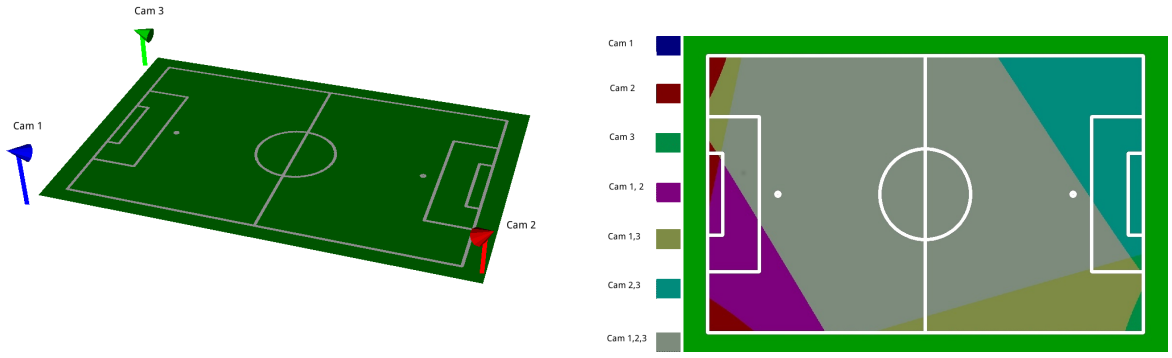
(a) (b)

Figura 6.2: Em (a) é apresentado o campo de visão de duas câmaras e em (b) o posicionamento das câmaras no campo.

---

2. É expetável que o ângulo entre as câmaras a efetuar a triangulação interfira com a precisão dos dados obtidos. Estima-se que o ângulo ideal entre estas se encontre à volta dos 90°, pois é quando existe uma maior ortogonalidade entre as câmaras.

Na próxima secção é apresentado um estudo, que tem como objetivo verificar em que medida o ângulo entre as câmaras afeta a precisão dos dados obtidos pelo sistema, bem como o ângulo ideal entre as câmaras, para efetuar a triangulação.



(a) (b)

Figura 6.3: (a) Integração de três câmaras no campo. (b) Ilustração do campo de visão das diferentes câmaras. Para a presente configuração, 2.28 % do campo é visto por apenas 1 câmara, 31.56 % por 2 câmaras e 66.16 % por 3 câmaras.

6.1 Influência do ângulo entre as câmaras utilizadas para a triangulação

Para se verificar o impacto que o ângulo entre as câmaras tem na obtenção das coordenadas 3-D da bola efetuaram-se dois testes: o primeiro consistiu num estudo do erro da posição 3-D da bola obtido pelo sistema, para diferentes ângulos entre câmaras. No segundo teste foi forçado um erro na deteção da bola, verificando-se em que medida este afeta a variação do erro na obtenção das coordenadas 3-D da bola, para os diferentes ângulos entre as câmaras.

6.1.1 Posição das câmaras

Para se verificar o impacto que o ângulo entre as câmaras tem na obtenção das coordenadas 3-D da bola, procurou-se variar o ângulo destas entre os 0° e 180°. Para tal, foram testadas as sete configurações apresentadas na Figura 6.4.

Em cada configuração colocou-se a bola em posições distintas no campo, obtendo-se as coordenadas 3-D desta e calculando-se o respetivo ângulo entre as câmaras. Na Figura 6.5 são apresentados os pontos onde a bola foi colocada. As coordenadas destes pontos encontram-se descriminadas no Anexo C.

Para as configurações das câmaras apresentadas na Figura 6.4, efetuou-se a calibração dos parâmetros extrínsecos e colocou-se a bola nas posições indicadas na Figura 6.5. Calculando-se de seguida o ângulo entre as câmaras e determinando-se a posição 3-D da bola. No gráfico da Figura 6.6 é apresentado um diagrama de extremos e quartis, relativamente ao erro obtido para os diferentes ângulos entre as câmaras. A partir deste obtém-se uma ideia acerca da variação do erro para os diferentes ângulos entre as câmaras, assim como o grau de confiança entre os valores obtidos.

Com base nestas medidas, chega-se à conclusão que quando o ângulo entre as câmaras se encontra compreendido entre os 40° e 120° existe uma tendência para que o erro e desvio padrão sejam menores. Todavia, estes são ainda testes preliminares, realizados com poucos dados. Pelo que foram realizados mais alguns testes, que têm como objetivo estudar a influência do erro na deteção da bola em função do ângulo entre as câmaras, estes são apresentados de

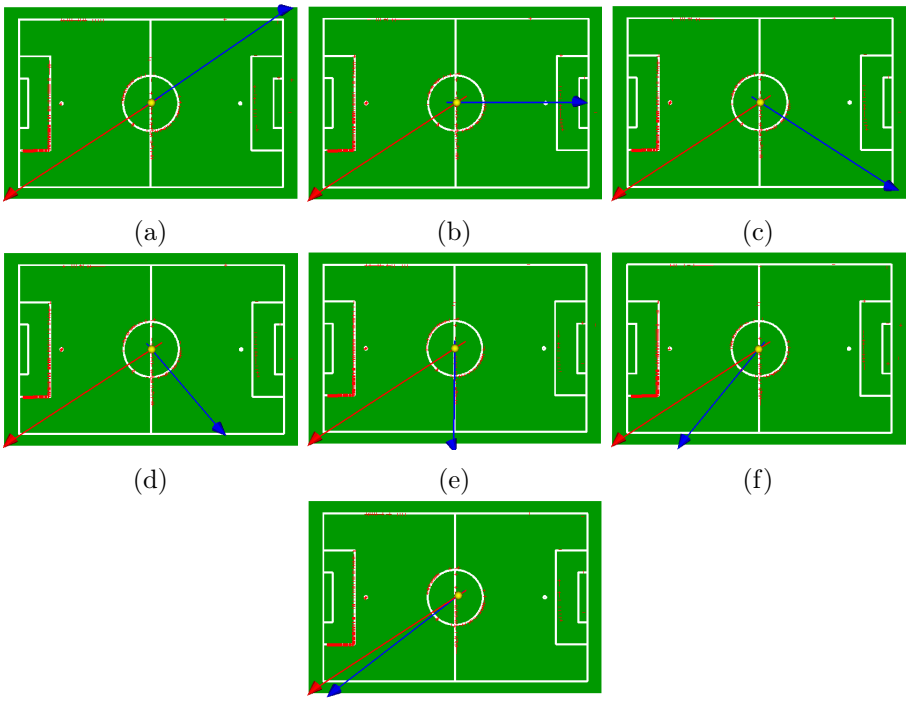


Figura 6.4: Configurações das câmaras utilizadas para verificação do impacto da ortogonalidade dos vetores, entre a bola e as duas câmaras, na obtenção das coordenadas 3-D da bola.

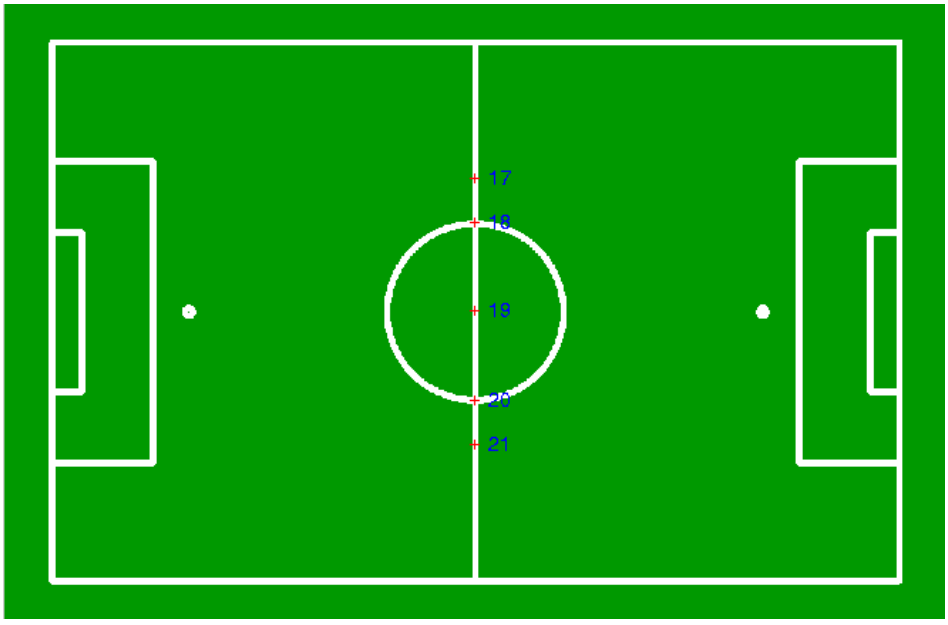


Figura 6.5: Pontos do campo onde foi colocada a bola. Para verificação da precisão das coordenadas obtidas em função do ângulo entre as câmaras.

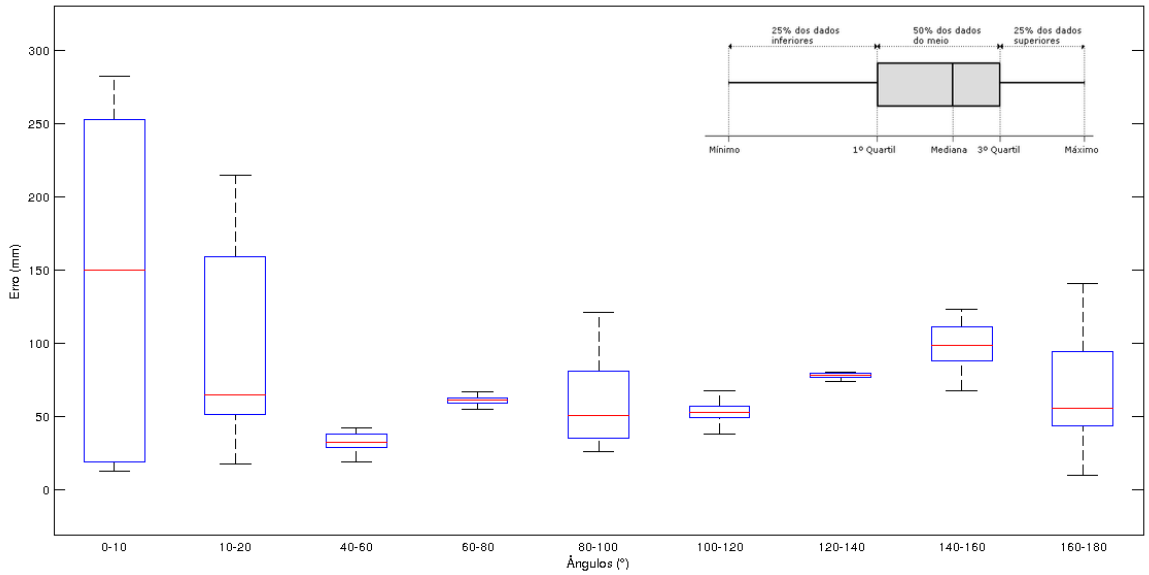


Figura 6.6: No diagrama de quartis é mostrada a variação do erro para os diferentes ângulos entre as câmaras. É possível visualizar também uma medida da confiança nos valores obtidos, bem como do surgimento de *outliers* em algumas configurações.

seguida.

6.1.2 Influência do erro na deteção da bola

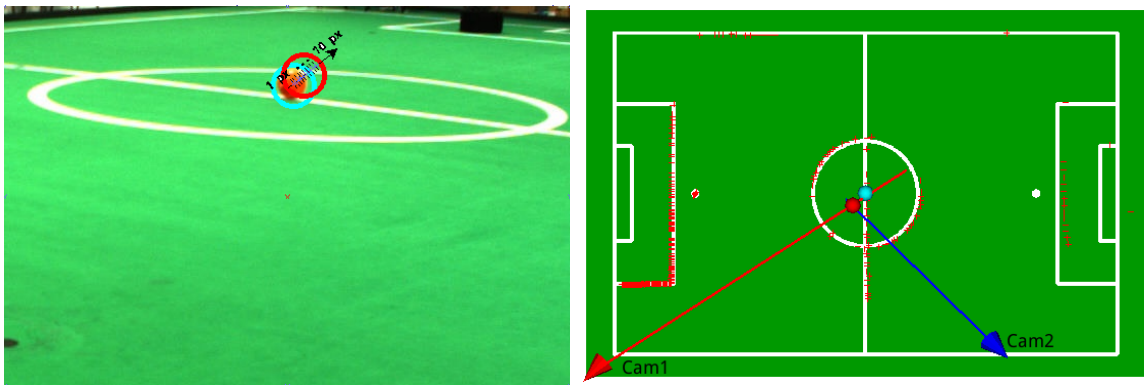
Muitas das vezes não se consegue efetuar a deteção de objetos em movimento em imagens RGB com bastante precisão. No instante da captura da imagem pela câmara, caso o objeto se encontre em movimento, muitas das vezes este fica desfocado na imagem, dificultando o processo de deteção do mesmo.

Esta experiência tem como objetivo estudar os ângulos entre as câmaras em que a variação do erro na obtenção das coordenadas 3-D do objeto é menor, com um erro na deteção do objeto nas imagens. Assim, a experiência consistiu em forçar um erro na deteção da bola, verificando-se em que medida este afeta a variação do erro na obtenção das coordenadas 3-D desta, para os diferentes ângulos entre as câmaras.

Na Figura 6.7 é apresentado um exemplo de uma imagem em que se forçou um erro na deteção da bola, assim como o impacto deste na obtenção das coordenadas 3-D da bola. O erro foi forçado na deteção da bola da segunda câmara, variando entre 1 a 10 pixels no eixo das abcissas e ordenadas, como mostrado em (a). Em (b) é apresentado o resultado da posição 3-D da bola obtida pelo sistema. A deteção da bola sem o erro adicionado encontra-se representado a vermelho e com a adição do erro a azul.

Para todas as configurações de câmaras (indicadas na Figura 6.4), determinou-se o erro na obtenção das coordenadas 3-D da bola. Na Figura 6.8 é apresentado um gráfico com os resultados obtidos, visto de duas perspetivas diferentes.

Observa-se que para ângulos muito baixos, entre os 0° e 20°, o erro da posição 3-D da bola é bastante variável com a variação do erro na deteção da bola na imagem, apresentando assim uma baixa confiança nas coordenadas 3-D da bola obtidas. Para ângulos entre as câmaras



(a) (b)

Figura 6.7: (a) Adição de um erro na deteção da bola da segunda câmara. (b) Informação da posição 3-D da bola no VTK. A cor azul representa a deteção real da bola e a cor vermelha representa a deteção da bola com o erro. Neste exemplo, o ângulo entre as câmaras é de 114°.

---

compreendidos entre 40° e 120°, o erro na obtenção das coordenadas 3-D da bola é menor, encontrando-se sempre abaixo dos 100 mm. Além disso, para esta gama de ângulos, o erro na obtenção das coordenadas da bola varia pouco com a adição do erro na deteção da bola da segunda imagem.

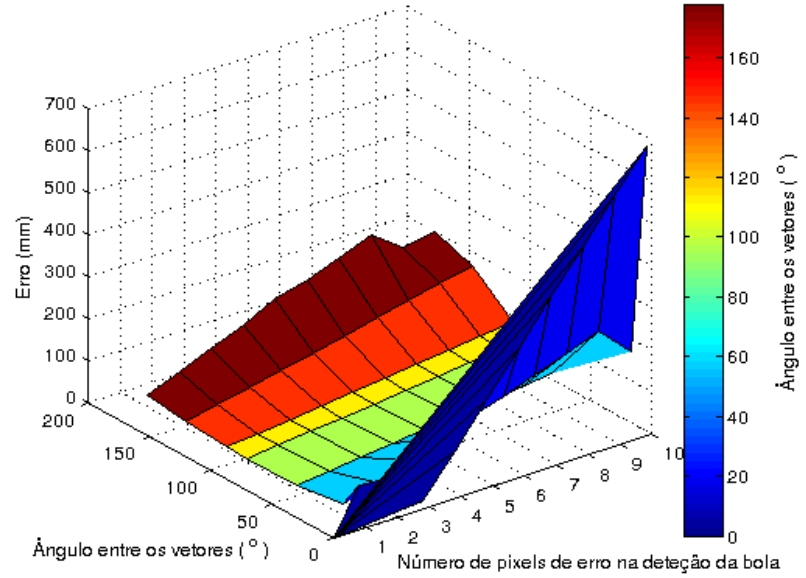
Reforçam-se as conclusões obtidas no primeiro teste, onde se verificou que entre os 40° e 120° existia uma tendência para que o erro e desvio padrão fossem menores. Conclui-se, assim, que a projeção dos vetores pelas duas câmaras deve apresentar um ângulo compreendido entre os 40° e 120°, de forma a se conseguir obter um menor erro e uma máxima confiança nos resultados obtidos.

Pela Figura 6.3 verifica-se que com apenas três câmaras uma percentagem considerável do campo é vista por apenas duas câmaras (31.56%), introduziu-se uma quarta câmara no sistema. Aumentando-se assim a percentagem de pontos no campo vistos por três ou mais câmaras e consequentemente obtendo-se um maior número de combinações entre as câmaras para efetuar a triangulação. Procurando-se sempre que o ângulo entre estas se encontre entre os limites ideais, 40° e 120°.

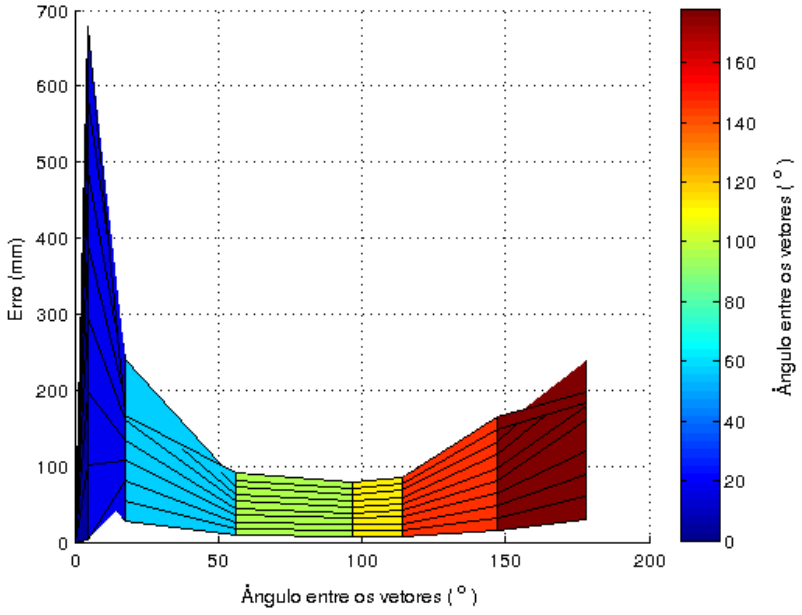
Na Figura 6.9 é apresentada em (a) uma configuração com as quatros câmaras e em (b) é apresentado o campo de visão destas. Verifica-se que para esta configuração uma percentagem considerável do campo é vista por três ou mais câmaras. Nestas condições, poder-se-ia efetuar a triangulação entre três ou quatro câmaras, em vez de duas, este assunto é abordado na próxima secção. Na Figura 6.10 é apresentado o mapa de cores correspondente ao campo de visão das quatro câmaras.

6.2 Triangulação entre múltiplas câmaras

Afim de se estudar a viabilidade do sistema, caso a triangulação seja efetuada com três câmaras, em vez de apenas duas, efetuou-se o seguinte teste: posicionaram-se três câmaras em três posições distintas no campo. De seguida, colocou-se a bola nas posições apresentadas na Figura 6.5 e determinou-se a posição 3-D desta, recorrendo do método da triangulação,



(a)



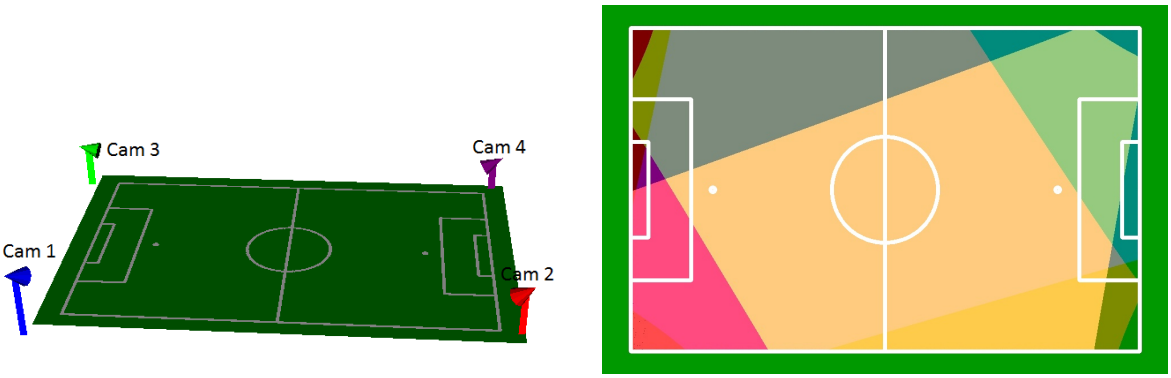
(b)

Figura 6.8: Erro na obtenção das coordenadas 3-D da bola, em função da introdução de um erro na deteção da bola na imagem de uma das câmaras e do ângulo entre as câmaras.

---

para as diferentes configurações de câmaras. Na Figura 6.11 são apresentadas as diferentes configurações de câmaras utilizadas para este teste.

Tendo como base o ângulo entre as câmaras, estudado na Secção 6.1, a configuração apresentada na Figura 6.11 (a) é inapropriada. De acordo com o estudo efetuado, as configurações com duas câmaras em que se consegue obter uma melhor precisão são as apresentadas na Figura 6.11 (b) e (c), em que o ângulo entre as câmaras é de 67 ° e 114 °, respetivamente.



(a) (b)

Figura 6.9: (a) Integração de quatro câmaras no campo. (b) Ilustração do campo de visão das diferentes câmaras. Para a presente configuração 1.30 % do campo é visto por apenas 1 câmara, 7.36 % por 2 câmaras, 44.04 % por 3 câmaras e 47.30 % por 4 câmaras.

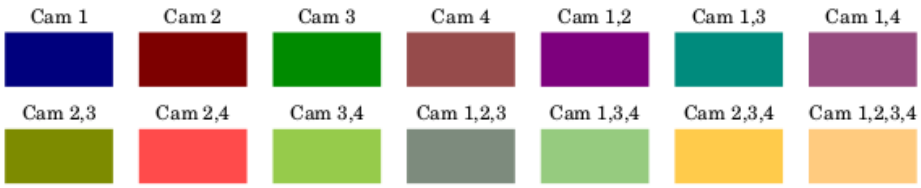


Figura 6.10: Mapa de cores correspondente ao campo de visão das quatro câmaras no campo.

Assim, apenas se verificou o impacto na obtenção das coordenadas 3-D da bola para estas configurações e a que utiliza as três câmaras.

Na Figura 6.12 é apresentado em (a) um gráfico com os resultados obtidos. Em (b) é apresentado um diagrama de quartis. Sendo o eixo "Posição da bola" relativo às posições apresentadas na Figura 6.5. O eixo "Configurações de câmaras" corresponde às configurações apresentadas na Figura 6.11. No diagrama de quartis é mostrado o erro médio, bem como o grau de confiança para as diferentes experiências. De salientar que a triangulação entre as três câmaras foi o resultado da média dos valores obtidos na triangulação entre duas câmaras, correspondentes às configurações indicadas na Figura 6.11 (a), (b) e (c).

Verifica-se que nas configurações (b) e (c) o erro médio é menor, existindo também uma menor variação nos valores obtidos. Conclui-se que o impacto de se efetuar a triangulação por três câmaras, face a apenas duas, parece não trazer vantagens. Contudo, isto ainda são ainda resultados preliminares que carecem de um estudo mais cuidado. De acordo com o estudo efetuado, verificou-se que para se obter uma boa precisão nas coordenadas 3-D do objeto de interesse, é necessário efetuar a escolha apropriada das duas câmaras para se efetuar a triangulação, tendo como base o ângulo entre as câmaras.

Visto que com o uso de quatro câmaras quase todo o campo é visto por três, ou mais, câmaras, conclui-se que com quatro câmaras consegue-se efetuar a monitorização de um jogo de futebol robótico com uma maior precisão, face a utilizar apenas três câmaras. Embora se tenha verificado que a triangulação entre três câmaras não apresenta vantagens, face à triangulação com duas câmaras, é necessário garantir que todo o campo seja visto por três, ou mais, câmaras. Sendo possível desta forma escolher as duas câmaras para efetuar a triangulação

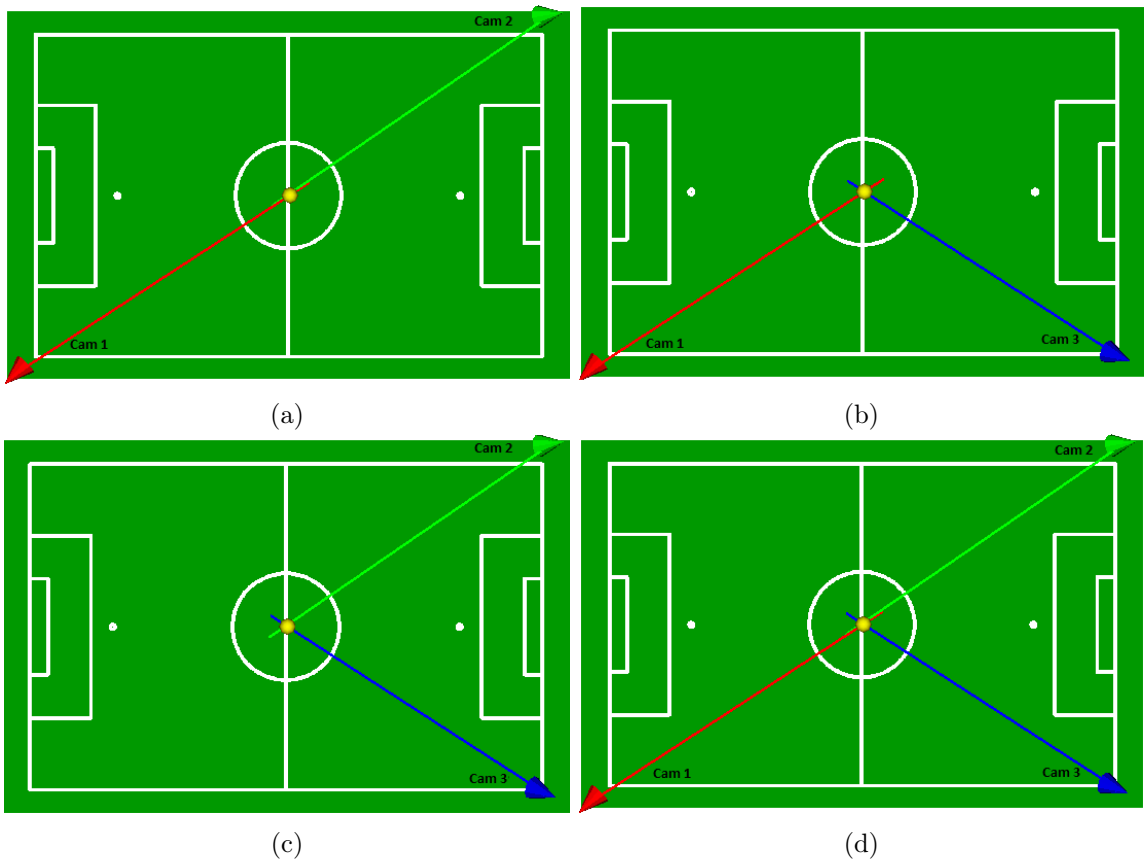


Figura 6.11: Obtenção das coordenadas 3-D da bola para diferentes combinações de câmaras, colocadas em três pontos distintos no campo. O ângulo entre as câmaras em (a) é de 180 °, em (b) de 114 ° e em (c) de 67 °.

---

que apresentam o melhor ângulo. Além disso, desta forma consegue-se contornar possíveis problemas de oclusão dos robôs com a bola, em que a bola deixa de ser vista numa das câmaras.

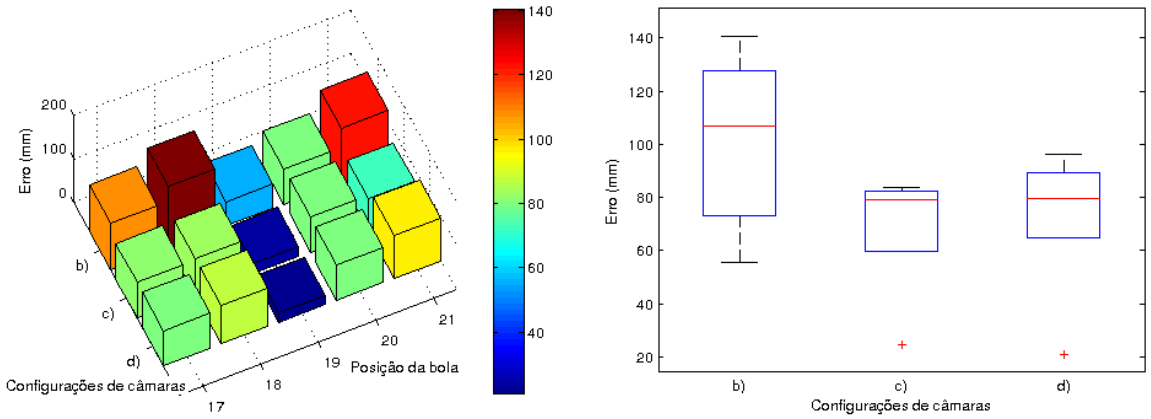
6.3 Estudo do melhor posicionamento das câmaras

Nesta secção é efetuado um estudo acerca do melhor posicionamento das quatro câmaras para se conseguir obter as coordenadas 3-D dos objetos de interesse com uma maior precisão. O algoritmo tem como base o ângulo ideal entre as câmaras, para se conseguir obter as coordenadas 3-D do objeto de interesse com uma maior precisão. Na Secção 6.1 verificou-se que para ângulos entre as câmaras compreendidos entre 40° e 120° consegue-se obter as coordenadas 3-D da bola com maior fiabilidade.

O algoritmo funciona da seguinte forma:

1. Estabelecimento de um conjunto de pontos 3-D ao longo de todo o campo.

2. Para todos os pontos 3-D, é verificado os que são vistos pelas diferentes câmaras.



(a) (b)

Figura 6.12: No lado esquerdo é apresentado um gráfico que mostra o erro na obtenção das coordenadas da bola. No eixo "Configurações de câmaras" é apresentado o erro para as configurações de duas câmaras onde se espera obter uma maior precisão (de acordo com o estudo efetuado na Secção 6.1) e a configuração com três câmaras, sendo estas apresentadas na Figura 6.11. O eixo "Posição da bola" apresenta o erro para a bola nas diferentes posições do campo (apresentadas na Figura 6.5). No lado direito, é apresentado um diagrama de quartis, onde é mostrado o erro médio das diferentes experiências, bem como uma medida da confiança dos valores obtidos.

---

3. Para os pontos vistos por duas ou mais câmaras, é efetuada a projeção dos vetores, desde o centro ótico de cada câmara até ao ponto 3-D. Neste passo é calculado o ângulo entre todas as combinações de duas câmaras a efetuar a triangulação, guardando o ângulo numa estrutura de dados para todas as combinações de câmaras.

4. Para cada ponto 3-D deve ser escolhido o ângulo mais próximo de 90°. Na imagem do modelo do campo são pintados os pontos de acordo com a Figura 6.13.

Para se realizar o estudo colocou-se as quatro câmaras em posições diferentes do campo, efetuando-se a triangulação para todos os pontos do campo visíveis por múltiplas câmaras. Nas Figuras 6.14, 6.15, 6.16 e 6.17 são apresentadas as quatro configurações de câmaras testadas, bem como o respetivo campo de visão. O mapa de cores correspondente ao campo de visão das quatro câmaras encontra-se apresentado na Figura 6.10.

Na Tabela 6.1 é mostrada a percentagem do campo vista por um número diferente de câmaras.

Na Figura 6.19 é mostrado um mapeamento colorido do campo, cada ponto do campo é pintado de acordo com o melhor ângulo entre câmaras que se consegue efetuar a triangulação. Na Tabela 6.2 é mostrada a percentagem do campo para os diferentes ângulos.

Pelos dados obtidos na Tabela 6.2, verifica-se que para a "Configuração 1" de câmaras a percentagem do campo onde o ângulo entre as câmaras não se encontra entre os valores ideais é mais reduzida, 22 %. Todavia, segundo a Tabela 6.1, para esta configuração existe uma elevada percentagem do campo vista por 1 câmara, 13 %, ou 2 câmaras, 43 %. Sendo que na Secção 6.1 chegou-se à conclusão de que se deve garantir que o campo seja visto por mais do que duas câmaras.

Tabela 6.1: Percentagem do campo de visão do campo visto pelas quatro câmaras, para as diferentes configurações de câmaras testadas.

	Percentagem do campo de visão do campo				
Configuração	1 câmara	2 câmara	3 câmara	4 câmara	Nenhuma câmara
1	13 %	43 %	20 %	14 %	0 %
2	34 %	34 %	16 %	16 %	0 %
3	13 %	43 %	20 %	14 %	10 %
4	13 %	26 %	27 %	34 %	0 %

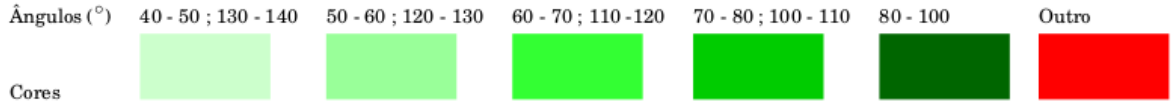
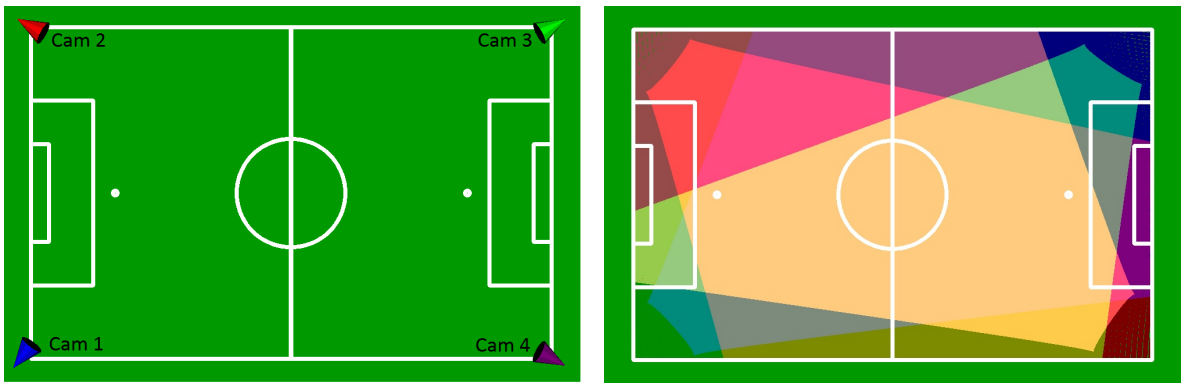


Figura 6.13: Cores em que se pintou o modelo do campo de futebol, em função do melhor ângulo entre as câmaras (mais próximo de 90°). Os tons a verde correspondem aos ângulos compreendidos entre os 40° e 120°, sendo que nesta gama as coordenadas 3-D da bola obtidas pelo sistema apresentam uma maior confiança, de acordo com o estudo efetuado anteriormente. A vermelho encontram-se os ângulos que não estão compreendidos entre estes valores.



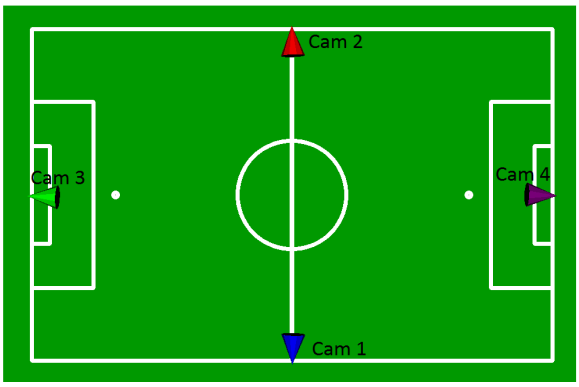
(a) Configuração 1

(b) Campo de visão

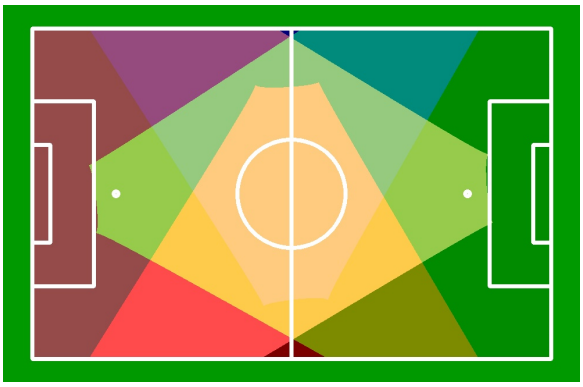
Figura 6.14: (a) Configuração das quatro câmaras no campo. (b) Ilustração do campo de visão das quatro câmaras. Nesta configuração 10 % do campo é visto por apenas 1 câmara, 27 % por 2 câmaras, 21 % por 3 câmaras e 42 % por 4 câmaras.

Já a "Configuração 4" apresenta uma elevada zona do campo onde o ângulo entre as câmaras não se encontra dentro dos valores ideais, 32.8 %. No entanto, nesta configuração a percentagem do campo vista por uma ou duas câmaras é inferior, face à "Configuração 1".

Embora a "Configuração 1" apresente uma elevada zona do campo monitorizada por apenas uma ou duas câmaras, prevê-se que de todas as configurações de câmaras apresentadas, esta é a que se consegue os melhores resultados, pois nesta a percentagem do campo onde os ângulos entre câmaras se encontram entre os valores ideais é consideravelmente maior que as restantes. como indicado na Tabela 6.2.

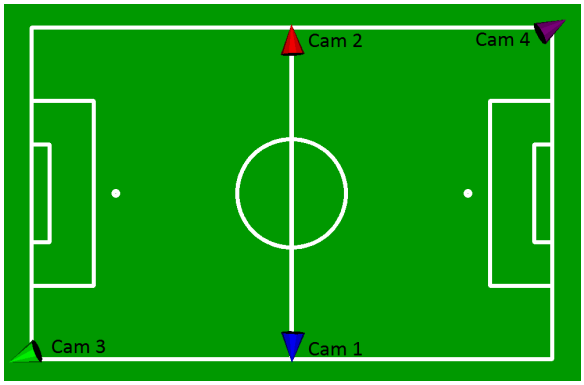


(a) Configuração 2

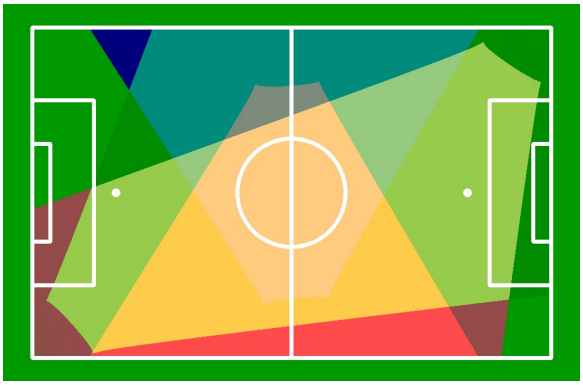


(b) Campo de visão

Figura 6.15: (a) Configuração das quatro câmaras no campo. (b) Ilustração do campo de visão das quatro câmaras. Nesta configuração 34 % do campo é visto por apenas 1 câmara, 34 % por 2 câmaras, 16 % por 3 câmaras e 16 % por 4 câmaras.



(a) Configuração 3



(b) Campo de visão

Figura 6.16: (a) Configuração das quatro câmaras no campo. (b) Ilustração do campo de visão das quatro câmaras. Nesta configuração 13 % do campo é visto por apenas 1 câmara, 43 % por 2 câmaras, 20 % por 3 câmaras, 14 % por 4 câmaras e 10 % do campo não é visto por nenhuma câmara.

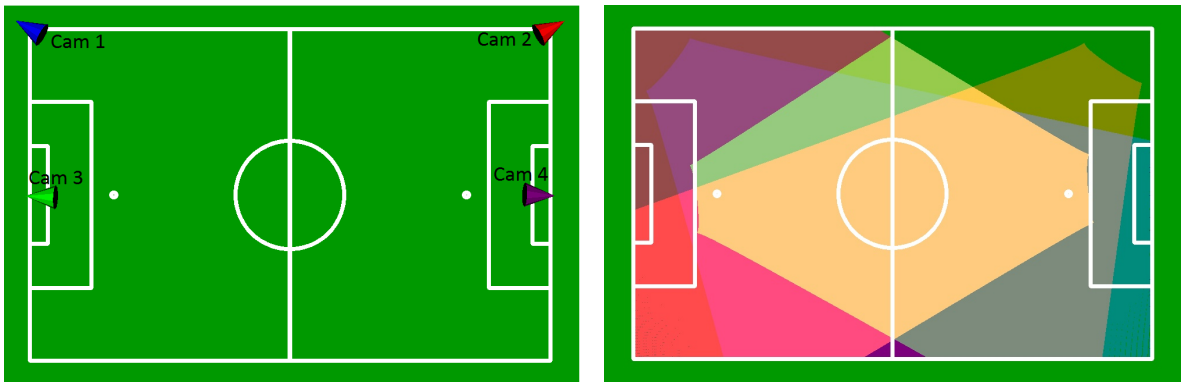
6.4 Sincronismo entre as diferentes câmaras

Cada câmara foi instalada no campo com um computador diferente, sendo que a comunicação entre os diferentes computadores é efetuada através de uma ligação do tipo cliente-servidor.

Inicialmente é executada a aplicação do servidor, que fica à escuta por clientes que se pretendam conectar à rede, no porto 500. Sempre que um novo cliente se pretende conectar à rede, é efetuado um pedido de estabelecimento de ligação do cliente para o servidor. Quando esta é aceite pelo servidor é estabelecida uma nova ligação.

Na Figura 6.20 é apresentado um desenho esquemático da ligação implementada. De notar que um dos computadores executa em simultâneo a aplicação do servidor e cliente, tendo assim uma câmara conectada ao mesmo, poupando-se o uso de um 5º computador.

Cada computador é responsável por executar uma aplicação cliente. Esta incorpora o



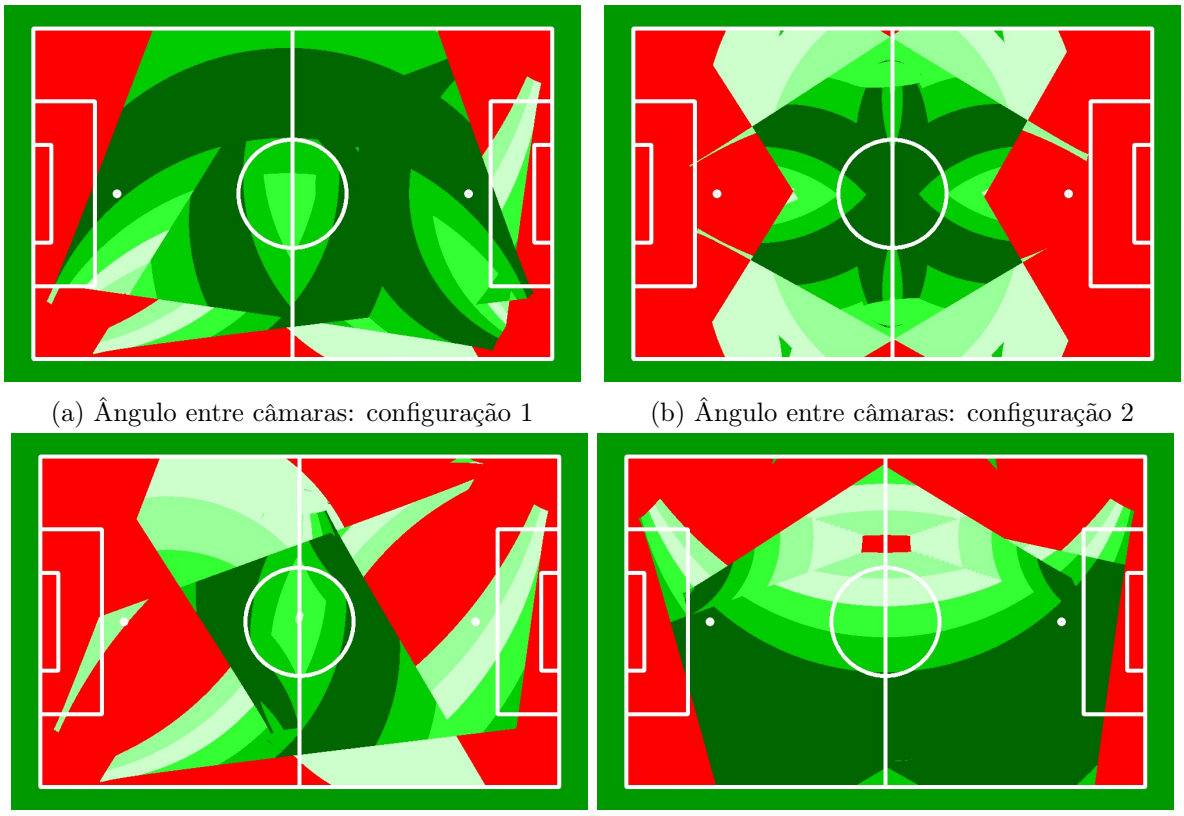
(a) Configuração 4 (b) Campo de visão

Figura 6.17: (a) Configuração das quatro câmaras no campo. (b) Ilustração do campo de visão das quatro câmaras. Nesta configuração 13 % do campo é visto por apenas 1 câmara, 26 % por 2 câmaras, 27 % por 3 câmaras e 34 % por 4 câmaras.

---

software para a deteção da bola, sendo que sempre que o servidor efetue um pedido para os clientes, estes respondem com a informação do número da câmara e coordenadas da bola, em pixels, da imagem correspondente à câmara conectada a este.

Nesta fase ainda não existe implementado um sistema de sincronismo externo nas câmaras. Pelo que o sistema está limitado a determinar a posição 3-D de objetos estáticos, garantindo-se que os objetos se encontram na mesma posição aquando da captura das imagens nas diferentes câmaras. Para se conseguir efetuar o *tracking* de objetos em movimento, é necessário implementar um sistema de sincronismo externo nas câmaras, garantindo-se que as frames das diferentes câmaras sejam obtidas no mesmo instante de tempo.



(a) Ângulo entre câmaras: configuração 1

(b) Ângulo entre câmaras: configuração 2

(c) Ângulo entre câmaras: configuração 3

(d) Ângulo entre câmaras: configuração 4

Figura 6.19: Ângulo entre câmaras a efetuar a triangulação, em função do posicionamento das mesmas. Os tons a verde dizem respeito aos ângulos compreendidos entre os 40° e 120° (apresentados na Figura 6.13), sendo que nesta gama as coordenadas 3-D da bola obtidas pelo sistema apresentam uma maior confiança, tal como estudado anteriormente. A vermelho encontram-se todos os ângulos que não se encontram dentro desta gama de valores.

Tabela 6.2: Percentagem dos ângulos das câmaras a efetuar a triangulação, para as diferentes configurações de câmaras testadas.

	Figura 6.19			
Ângulo entre câmaras (°)	(a)	(b)	(c)	(d)
40 - 50 ; 130 - 140	1%	13.5%	14.8%	8.2%
50 - 60 ; 120 -130	1.4%	9.6%	10.5%	6.3%
60 - 70 ; 110 - 120	2.5%	5.1%	7.3%	7.8%
70 - 80 ; 100 - 110	23.3%	9%	8%	8.9%
80 - 100	49.8%	17.8%	10.9%	36%
Outro	22%	45%	48.5%	32.8%

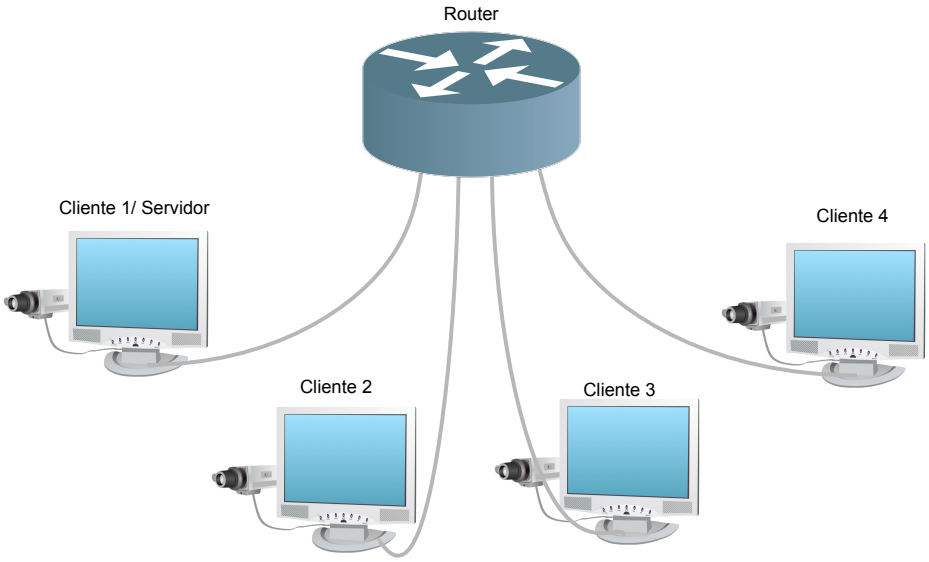


Figura 6.20: Ligação do tipo cliente-servidor. Esta foi utilizada para efetuar a comunicação entre os diferentes computadores, que se encontram distribuidos ao longo do campo.

Capítulo 7

Conclusão

No ano de 2014 foi concluído um novo edifício na Universidade de Aveiro, para o desenvolvimento de projetos na área da robótica. Neste edifício está instalado um campo de futebol robótico com dimensões oficiais, para o desenvolvimento do projeto CAMBADA. Nesta dissertação desenvolveu-se um sistema para efetuar a monitorização de objetos em 3-D com base em múltiplas câmaras RGB, sendo que se pretende aplicar o sistema desenvolvido no campo de futebol robótico do laboratório. Permitindo assim efetuar a monitorização de um jogo de futebol robótico.

Foram apresentados diferentes métodos que existem para o propósito, bem como sistemas comerciais. O trabalho desenvolvido dividiu-se em três passos fundamentais: a seleção do método mais apropriado para a aplicação; o desenvolvimento do software para determinar a posição 3-D de objetos com base em múltiplas câmaras; a validação dos resultados obtidos, em termos de precisão.

Selecionou-se o método da triangulação, que segundo a literatura é o mais adequado em aplicações para elevadas *baselines* entre as câmaras. Além disso, os sistemas comerciais existentes para este fim baseiam-se neste método para efetuar o *tracking* de objetos.

Para a triangulação entre duas câmaras, foi estudada em que medida o ângulo entre estas afeta o grau de confiança dos dados obtidos. Verificou-se ainda o impacto em efetuar a triangulação com mais do que duas câmaras, face a apenas duas. Concluiu-se que com a triangulação entre apenas duas câmaras consegue-se um sistema bastante preciso e fiável, devendo-se por isso garantir que o ângulo entre estas esteja compreendido entre os 40° e 120°.

Em aplicações robóticas, o tempo de processamento é um fator de elevada importância. Pretende-se que o tempo de processamento do sistema de *ground truth* seja suficientemente rápido para conseguir acompanhar a atualização da informação nos robôs. Mediu-se o tempo de processamento do sistema de visão, responsável pela deteção da bola nas imagens, sendo que este ocupa grande parte do tempo do sistema desenvolvido. O tempo medido ronda os 70 ms, sendo que os robôs atualizam a informação a cada 100 ms, pelo que se conclui que o tempo de processamento do sistema é aceitável.

7.1 Ferramentas desenvolvidas

Na presente dissertação foram desenvolvidas diversas ferramentas, sendo estas indicadas de seguida. Os argumentos de cada uma destas aplicações é apresentado no Anexo D.

1. Desenvolvimento de uma aplicação para adquirir imagens das câmaras *pointGrey Zebra*.

Esta é denominada de ***getImagesZebraPoint2Grey*** e pode ser utilizada para adquirir imagens do *chessboard* para efetuar uma posterior calibração dos parâmetros intrínsecos, por exemplo.

2. Desenvolvimento de um software que permite efetuar a calibração dos parâmetros intrínsecos e extrínsecos das câmaras. O software para a calibração dos parâmetros intrínsecos é denominado de ***intrinsicCalibration*** e dos extrínsecos de ***visionExtrinsic***. Estes parâmetros são armazenados em ficheiros *XML*, sendo criado um ficheiro deste tipo para cada câmara.

   Desenvolveu-se ferramentas de apoio à calibração, de forma a quantificar a qualidade da mesma. Através da deteção das linhas do campo e da projeção das mesmas num modelo do campo, obtém-se uma medida do erro devido à calibração. O erro é obtido através do cálculo da média da distância entre os pontos projetados e o ponto da linha branca mais próximo, para todos os pontos projetados.

   A calibração dos parâmetros extrínsecos das câmaras é efetuada pela correspondência de pixels com as respetivas coordenadas 3-D. Foi desenvolvida uma aplicação que permite efetuar esta calibração, clicando com o rato nos os pixels da imagem e indicação das respetivas coordenadas 3-D. O software indica o valor do erro médio da calibração (em pixels e mm) em tempo real, sendo que o utilizador pode adicionar mais pontos à calibração, tendo como vista melhorar a mesma, ou caso prefira começar a calibração de novo. Esta aplicação é denominada de ***visionExtrinsic***.

3. Desenvolvimento de um programa que efetua um mapeamento colorido das zonas do campo vistas pelas diferentes câmaras, devolvendo ainda a percentagem do campo vista por um número diferente de câmaras. Para que este consiga efetuar o mapeamento do campo de visão das câmaras deve-lhe ser fornecido como parâmetros de entrada os parâmetros intrínsecos e extrínsecos das diferentes câmaras, armazenados previamente em ficheiros *XML* aquando efetuada a calibração. O programa desenvolvido para este propósito é denominado de ***fieldOfViewCams***.

4. Verificou-se que a fiabilidade dos dados obtidos pelo sistema está diretamente relacionada com o ângulo entre as câmaras a efetuar a triangulação. Concluiu-se que para uma maior confiança estas devem encontrar-se espaçadas entre os 40° e 120°. Baseando-se neste estudo desenvolveu-se uma aplicação onde é possível verificar o melhor posicionamento das câmaras no campo, afim de se obter uma maior fiabilidade nos dados obtidos ao longo de todo o campo. Esta aplicação é denominada de ***optimizeCameraPosition***.

5. Implementação de um programa denominado de ***visionOffMode*** que permite trabalhar em modo *off-line*, sem que as câmaras estejam obrigatoriamente ligadas ao computador. Como tal, deve ser gravado previamente um vídeo para cada câmara presente no campo. Neste modo é efetuado o processamento das imagens contidas nos vídeos gravados para as diferentes câmaras, não tendo estas de estar necessariamente ligadas, para se obter imagens.

6. A aplicação para gravação de um vídeo com as câmaras *pointGrey Zebra* desenvolvida é denominada de ***recordVideo***. A esta deve ser especificada o número da câmara e de imagens que se pretende concatenar no vídeo. Cada câmara possui um número diferente,

de 1 a 4, através deste número o software conhece o *MAC Address* e *IP Address* da mesma.

7. A aplicação ***client*** desenvolvida deve ser executada em todos os computadores clientes. Quando esta aplicação é executada é enviado um pedido de estabelecimento de ligação do cliente para o servidor, depois do pedido ser aceite é estabelecida uma nova ligação. Esta aplicação contém o software de visão necessário para a deteção da bola no campo de futebol robótico, sendo que sempre que o servidor enviar um pedido para os clientes, estes enviam ao servidor o pixel correspondente ao centro da bola da câmara conectada ao computador.

8. Desenvolvimento de uma aplicação, ***server***, responsável por efetuar a gestão da informação de todos os clientes. Com base na informação enviada pelos clientes, é determinada a posição 3-D da bola no campo de futebol robótico. Esta aplicação é responsável ainda por efetuar a demonstração 3-D dos diferentes intervenientes no jogo de futebol robótico.

7.2 Trabalho futuro

A validação dos dados obtidos pelo software desenvolvido foi efetuada manualmente, colocando a bola em pontos específicos do campo e obtenção das respetivas coordenadas 3-D pelo sistema, não permitindo quantificar o sistema desenvolvido com uma precisão abaixo dos milímetros. Para melhorar este processo pode-se recorrer a um sistema de *ground truth* externo, como por exemplo o *Vicon*, conseguindo-se assim quantificar o sistema com uma precisão na ordem dos μm.

O sistema não se encontra a funcionar em tempo real, pois é necessário garantir que as frames provenientes das diferentes câmaras sejam obtidas exatamente no mesmo instante de tempo. Embora as câmaras suportem um mecanismo de sincronismo externo, este não foi implementado. O motivo deste não ter sido implementado é porque houve um atraso na implementação do hardware para a sincronização das câmaras, ao que estava fora do âmbito desta dissertação.

Nos objetivos iniciais desta dissertação consta efetuar o *tracking* da bola e dos robôs com alguma precisão. Apenas foi efetuado o *tracking* da bola, visto que já existe um software desenvolvido para a equipa CAMBADA capaz de efetuar a deteção de objetos coloridos com bastante precisão e fiabilidade, sendo este facilmente aplicável à bola de futebol.

O sistema desenvolvido apenas tem em consideração o pixel correspondente ao centro da bola, para as diferentes câmaras, para determinar o posicionamento 3-D da mesma. Assim sendo, o software é facilmente aplicável a qualquer outro objeto. Para isso, apenas lhe deve ser indicado o pixel correspondente ao centro do objeto que se pretende monitorizar, podendo corresponder este a um robô, ou qualquer outro objeto.

A calibração dos parâmetros extrínsecos das câmaras é efetuada manualmente, clicando com o rato sobre os pixels na imagem e indicação das respetivas coordenadas 3-D. Visto que as coordenadas 3-D da linhas do campo são fixas, através da deteção destas e indicação "automática" das suas coordenadas 3-D consegue-se tornar este processo automático.

Apêndice A

Interface para o *driver* das câmaras

Para se conseguir efetuar o controlo das câmaras por software foi necessário implementar uma interface para o *driver Flycapture* das câmaras, fornecido pelo fabricante; adicionando esta à biblioteca UAVision. A biblioteca incorpora uma interface comum a todos os tipos de câmaras que se pretenda adaptar a esta. Na Tabela A.1 são apresentados os métodos implementados, bem como a respetiva função.

Tabela A.1: Métodos da interface do UAVision para controlo de câmaras.

Métodos:	**Função:**
CameraZebra	Construtor
`void startCamera(CameraSettings *camSettings)`	Iniciar a câmara
`void stopCamera(CameraSettings *camSettings)`	Interromper o envio de imagens
`void shutDownCamera()`	Parar a câmara e terminar a ligação
`void setParameters(CameraSettings *camSettings)`	Definir os parâmetros da câmara
`void setParameter(unsigned int parameter, unsigned value)`	Definir um único parâmetro da câmara
`void getParameter(unsigned int parameter, double *value)`	Obter o valor de um parâmetro da câmara
`void getParameters(CameraSettings *camSettings)`	Obter todos os parâmetros da câmara
`void getParameterRanges(unsigned parameter,ParameterRange p)`	Obter limites de um parâmetro
`void getParametersRanges(ParameterRange p[])`	Obter limites de todos os parâmetros
`int readFrame cv::Mat &imageOpenCV)`	Ler uma frame da câmara
`void setAutoMode(CameraSettings *camSettings)`	Configurar a câmara no modo automático
`void setManualMode(CameraSettings *camSettings)`	Configurar a câmara no modo manual
`void printParameters()`	Imprimir as configurações da câmara
`void printParameter(unsigned int parameter)`	Imprimir apenas um parâmetro da câmara

Apêndice B

Aquisição de imagens das câmaras *PointGrey Zebra*

Foi necessário desenvolver software para se conseguir obter imagens das câmaras *Point Grey Zebra*. Abaixo encontra-se descriminado o código desenvolvido para este propósito.

```
// Construtor
CameraZebra::CameraZebra(FlyCapture2::IPAddress ipAddress, FlyCapture2::MACAddress
macAddress, FlyCapture2::IPAddress subnetMask, FlyCapture2::IPAddress DG,int numCamera)
{
        cam = new FlyCapture2::Camera;
        FlyCapture2::Error error;
        FlyCapture2::BusManager bsr;

        error = bsr.ForceIPAddressToCamera(macAddress,ipAddress,subnetMask,DG);
        if (error != PGRERROR_OK)
                PrintError( error );

        this->numCam = numCamera;
        this->iPaddress = ipAddress;
}
```

```
// Camera initialization
int CameraZebra::initCamera(CameraSettings *camSettings)
{
        if(numCamerasDetected()==0){
                printf("ERROR: No cameras connected");
        exit(1);}
        if(camSettings->getCameraSetting(UAV_CAMERAUSED) == 0){
                startCamera(camSettings);
                camSettings->setCameraSetting(UAV_CAMERAUSED, 1);
        }
        else{
                stopCamera(camSettings);
                startCamera(camSettings);
```

```
                camSettings->setCameraSetting(UAV_CAMERAUSED, 1);
        }
        return 0;
}
```

```
// Read frame
int CameraZebra::readFrame(cv::Mat &imageOpenCV)
{
        FlyCapture2::Error error;

        if(imageOpenCV.empty())
                imageOpenCV = cv::Mat(cv::Size(1200,1600),CV_8UC3);
        if(!cam->IsConnected())
        {
                FlyCapture2::BusManager busMgr;

                error = busMgr.GetCameraFromIPAddress(iPaddress, guid);

                if (error != PGRERROR_OK)
                        PrintError( error );
        }

        RunSingleCamera(imageOpenCV);

    return 0;
}

// RunSingleCamera method
int CameraZebra::RunSingleCamera(cv::Mat &imageOpenCV)
{
        FlyCapture2::Error error;

        // Connect to a camera
        if(!cam->IsConnected())
        {
                error = cam->Connect(guid);
                FlyCapture2::Property prop;
                error = cam->StartCapture();
                if (error != PGRERROR_OK)
                        PrintError( error );
        }
        FlyCapture2::Image rawImage;

        // Retrieve an image
        error = cam->RetrieveBuffer( &rawImage );
        if (error != PGRERROR_OK)
                PrintError( error );
```

```
43
44          std::cout << "Grabbed image "  << std::endl;
45
46          //Create a converted image
47          FlyCapture2::Image convertedImage;
48
49          //Convert the raw image
50          error = rawImage.Convert(cv::PIXEL_FORMAT_RGB, &convertedImage );
51          if (error != PGRERROR_OK)
52                  PrintError( error );
53
54          //Creatte a unique filename
55          memcpy(imageOpenCV.ptr(), convertedImage.GetData(), convertedImage.GetDataSize()
56
57          if (error != PGRERROR_OK)
58                  PrintError( error );
59          return 0;
60  }
```

Apêndice C

Coordenadas do campo em pontos específicos

Na Tabela C.1 são apresentadas as coordenadas do campo nos pontos referentes à Figura C.1.

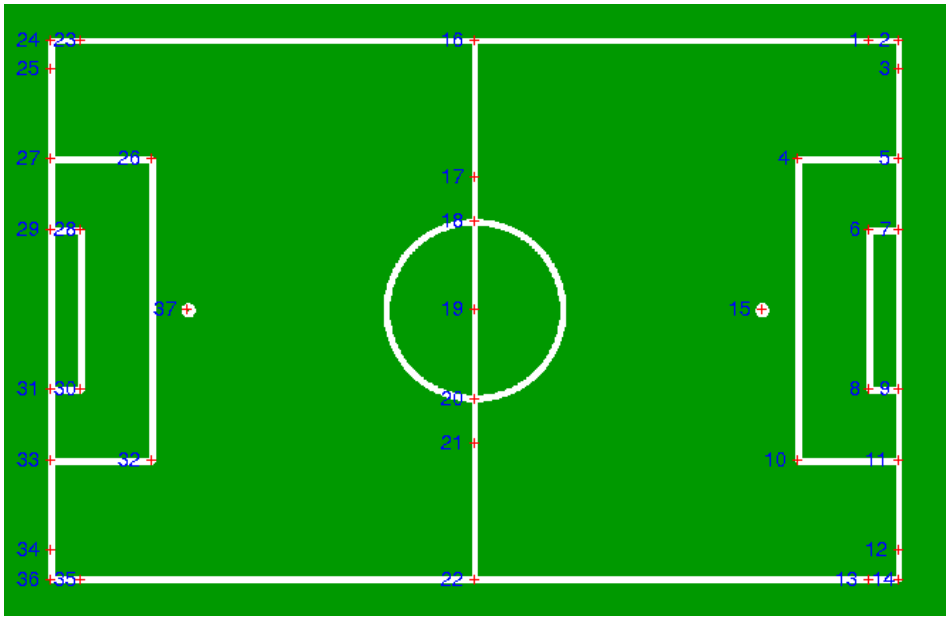


Figura C.1: Numeração das coordenadas do campo em pontos específicos; estas encontram-se especificadas na Tabela C.1.

Tabela C.1: Coordenadas do campo correspondentes aos pontos indicados na Figura C.1.

idx	X (mm)	Y (mm)		X (mm)	Y (mm)		X (mm)	Y (mm)
1	8312.5	5687.5	23	-8312.5	5687.5	16	0	5687.5
2	8937.5	5687.5	24	-8937.5	5687.5	17	0	2800
3	8937.5	5062.5	25	-8937.5	5062.5	18	0	1875
4	6812.5	3187.5	26	-6812.5	3187.5	19	0	0
5	8937.5	3187.5	27	-8937.5	3187.5	20	0	-1875
6	8312.5	1687.5	28	-8312.5	1687.5	21	0	-2800
7	8937.5	1687.5	29	-8937.5	1687.5	22	0	-5687.5
8	8312.5	-1687.5	30	-8312.5	-1687.5			
9	8937.5	-1687.5	31	-8937.5	-1687.5			
10	6812.5	-3187.5	32	-6812.5	-3187.5			
11	8937.5	-3187.5	33	-8937.5	-3187.5			
12	8937.5	-5062.5	34	-8937.5	-5062.5			
13	8312.5	-5687.5	35	-8312.5	-5687.5			
14	8937.5	-5687.5	36	-8937.5	-5687.5			
15	6057.5	0	37	-6057.5	0			

Apêndice D

Ferramentas desenvolvidas

Na Tabela D.1 são apresentadas as ferramentas desenvolvidas nesta dissertação, bem como os respetivos parâmetros de entrada.

Tabela D.1: Indicação dos parâmetros de entrada para as diversas ferramentas desenvolvidas.

Programa	**Parâmetros de entrada**
getImagesZebraPoint2Grey	-numCamera NUM_CAM
intrinsicCalibration	-numCamera NUM_CAMERA -numImages NUM_IMAGES
visionExtrinsic	-cf x.conf -nocalib -numCamera NUM_CAM
fieldOfViewCams	-resolution RESOLUTION
optimizeCameraPosition	-resolution RESOLUTION
recordVideo	-numCamera NUM_CAMERA -numImages NUM_IMAGES
visionOffMode	-
client	-
server	-

Bibliografia

[1] António J R Neves, Alina Trifan e Bernardo Cunha. "UAVision: A modular time-constrained vision library for color-coded object detection". Em: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8641 LNCS (2014), pp. 351–362.

[2] Vision Systems Design. *Choosing a 3D vision system for automated robotics applications - Vision Systems Design*. 2014. URL: `http://www.vision-systems.com/` (acedido em 08/09/2015).

[3] OptiTrack. *Motion Capture for Biomechanics - OptiTrack*. 2015. URL: `http://www.optitrack.com/` (acedido em 05/09/2015).

[4] Franklin C Crow. "Summed-Area Tables for Texture Mapping". Em: *Proceedings of SIGGRAPH* 18.3 (1984), pp. 207–212.

[5] Digital production middle east. *Tracking motion capture*. 2015. URL: `http://www.digitalproductionme.com` (acedido em 08/09/2015).

[6] *Markerless Motion Capture and Analysis - Silhouette-Tracking - Simi Shape 3D*. URL: `http://www.simi.com/` (acedido em 17/11/2015).

[7] Ning Qian. "Binocular Disparity and the Perception of Depth". Em: *Cell Press* 18 (1997), pp. 359–368.

[8] Brian G Schunck Ramesh Jain Rangachar Kasturi. *Machine Vision*. Ed. por Eric M Munson. McGraw-Hill, Inc, 1995.

[9] R. Hartley Zisserman e A. "Multiple View Geometry in Computer Vision". Em: *Academic Press* (2012).

[10] Dr. Simon J D Prince. *Two or More Cameras*. Department of Computer Science University College of London. 2008.

[11] Richard I Hartley. "In Defense of the Eight-Point Algorithm". Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.6 (1997), pp. 580 –593.

[12] Erik Valdemar Cuevas Jimenez. "Intelligent Robotic Vision". Tese de doutoramento. Universitat Berlin, 2006.

[13] Andrew Zisserman Richard Hartley. *Multiple View Geometry in computer vision*. Ed. por Cambrige University Press. 2004, pp. 312–313.

[14] Jens Puwein et al. "Robust multi-view camera calibration for wide-baseline camera networks". Em: *2011 IEEE Workshop on Applications of Computer Vision, WACV 2011* (2011), pp. 321–328.

[15] Shirai Y Miura J Yamada A. "Tracking Players and a Ball in Video Image Sequence and Estimating Camera Parameters for 3D Interpretation of Soccer Games". Em: *Pattern Recognition, 2002. Proceedings. 16th International Conference on (Volume:1)* vol.1.1 (2002), pp. 303–306.

[16] Alina Trifan et al. "Real-Time Color Coded Object Detection Using a Modular Computer Vision Library". Em: *Journal of Real-Time Image Processing manuscript No* (2014).

[17] B Cunha et al. "CAMBADA'2015: Team Description Paper". Em: (2015).

[18] Vitor Santos. *Reconhecimento de imagem: Modelos e padroes*. 2014.

[19] Kai-Bin Wei YongZhang. "Research on Wide Baseline Stereo Matching Based on peA-SIFT". Em: *3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)* 5 (2010), pp. 137–150.

[20] Scale-invariant Keypoints e David G Lowe. "Distinctive Image Features from". Em: 60.2 (2004), pp. 91–110.

[21] Tinne Tuytelaars Luc Van Gool Herbert Bay Andreas Ess. "SURF: Speeded Up Robust Features". Em: *Computer Vision and Image Understanding (CVIU)* 110.3 (2008), pp. 346–359.

[22] Klaus D McDonald-Maier Shoaib Ehsan. "Exploring Integral Image Word Length Reduction Techniques for SURF Detector". Em: *IEEE Computer Society* (2009), pp. 635–639.

[23] Luo Juan e O Gwun. "A comparison of sift, pca-sift and surf". Em: *International Journal of Image Processing (IJIP)* 3.4 (2009), pp. 143–152.

[24] Edward Rosten e Tom Drummond. "Fusing points and lines for high performance tracking". Em: *Proceedings of the IEEE International Conference on Computer Vision* II (2005), pp. 1508–1515.

[25] Edward Rosten, Reid Porter e Tom Drummond. "Faster and better: A machine learning approach to corner detection". Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (2010), pp. 105–119.

[26] Dg Viswanathan. *Features from Accelerated Segment Test (FAST)*.

[27] Spectral Properties Imaging et al. *Imaging Electronics 101 : Understanding Camera Sensors for Machine*. 1992. URL: `http://www.edmundoptics.com` (acedido em 02/10/2015).

[28] John Compton and John Hamilton, John Compton Hamilton e John. *Color Filter Array 2.0*. 2007.

[29] Adrian Kaehler e Bradski Gary. *Learning OpenCV*. Ed. por Mike Loukides. O'Reilly Media, 2013, p. 575.

[30] Point Grey. *PoE HD-SDI Digital Camera - Technical Reference*. 2014.

[31] Augmented Reality e Opencv Camera Calibration. "Augmented Reality and Camera Calibration". Em: (1994), pp. 1–5.

[32] A FETIC, D JURIC e D OSMANKOVIC. "The procedure of a camera calibration using Camera Calibration Toolbox for MATLAB". Em: *Proceedings of the 35th International Convention MIPRO*. 2012, p. 1752.1757.

[33] Ahmed Elgammal. *Computer Vision: Camera Calibration*. 2014.

[34] Zhengyou Zhang. “A flexible new technique for camera calibration”. Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334.

[35] *VTK - The Visualization Toolkit*. URL: `http://www.vtk.org/` (acedido em 06/10/2015).