Método de correspondência para sistemas de visão multi-câmara

João Pedro Mendes Pereira Ribeiro,
Nº 1081694

Mestrado em Engenharia Eletrotécnica e de Computadores -
Área de Especialização de Sistemas Autónomos

14 de Julho de 2015

Dissertação, para satisfação parcial dos requisitos do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: João Pedro Mendes Pereira Ribeiro,
Nº 1081694

Orientador: Eduardo Alexandre Pereira Da Silva

Mestrado em Engenharia Eletrotécnica e de Computadores -
Área de Especialização de Sistemas Autónomos

Agradecimentos

Nesta secção queria agradecer ao meu orientador Eng.º Eduardo Silva pela ajuda disponibilizada ao longo do projeto e pelas oportunidades apresentadas ao longo do mestrado.

Quero agradecer ao Eng.º José Almeida por todo o apoio e acompanhamento disponibilizado ao longo deste ciclo, bem como todo conhecimento que transmitiu.

A todos os membros do laboratório de sistemas autónomos (LSA) por toda a experiência e oportunidades que me foram concedidas ao longo destes anos em que frequentei o LSA.

Um agradecimento ao João Ribeiro e à Joana Moreno por toda ajuda e disponibilidade oferecida ao longo desta etapa.

A todos os meus amigos de curso que percorreram comigo esta etapa da minha vida, nomeadamente ao Flávio Lopes, André Faria, André Neves, Renato Ribeiro, José Araújo, Bruno Matias, Joel Oliveira, Tiago Fernandes, Nelson Campos e Filipe Silva. Um enorme agradecimento aos meus grandes amigos Delmiro Costa, Rita Fonseca, Ricardo Santos e Pedro Silva pela amizade demonstrada ao logo deste anos.

Um especial agradecimento aos meus pais, ao meu irmão e à minha tia, por todo o apoio incondicional e por me terem proporcionado a oportunidade de efetuar esta etapa académica assim como todas as anteriores.

Para terminar, quero agradecer à minha namorada, Rita Moreno, pelo apoio constante, pela paciência, a ternura e força que me deu ao longo desta etapa. Mesmo quando tudo ia abaixo, a voz que me dizia que ia conseguir estava sempre presente e perseverante até

ao fim.

Resumo

Os sistemas de perceção visual são das principais fontes de informação sensorial utilizadas pelos robôs autónomos, para localização e navegação em diferentes meios de operação. O objetivo passa por obter uma grande quantidade de informação sobre o ambiente que a câmara está a visualizar, processar e extrair informação que permita realizar as tarefas de uma forma eficiente. Uma informação em particular que os sistemas de visão podem fornecer, é a informação tridimensional acerca do meio envolvente. Esta informação pode ser adquirida recorrendo a sistemas de visão monoculares ou com múltiplas câmaras. Nestes sistemas a informação tridimensional pode ser obtida recorrendo à técnica de triangulação, tirando partido do conhecimento da posição relativa entre as câmaras.

No entanto, para calcular as coordenadas de um ponto tridimensional no referencial da câmara é necessário existir correspondência entre pontos comuns às imagens adquiridas pelo sistema. No caso de más correspondências a informação 3D é obtida de forma incorreta.

O problema associado à correspondência de pontos pode ser agravado no caso das câmaras do sistema terem características intrínsecas diferentes nomeadamente: resolução, abertura da lente, distorção. Outros fatores como as orientações e posições das câmaras também podem condicionar a correspondência de pontos. Este trabalho incide sobre problemática de correspondência de pontos existente no processo de cálculo da informação tridimensional.

A presente dissertação visa o desenvolvimento de uma abordagem de correspondência de pontos para sistemas de visão no qual é conhecida a posição relativa entre câmaras.

Palavras chave:

Robôs, ISePorto, *Features*, Detetores de *features*, *Template matching*, Geometria epipolar, Retificação de imagens, *Stereo*, *Matchers*, Triangulação *stereo*, Correspondência de pontos, Câmaras;

Abstract

Visual Perception Systems are one of the most utilized means in robotic sensing and perception for localization and navigation tasks. The visual systems main purpose is to acquire visual information from the robot environment, and process it in an efficient manner. One commonly used information that can be processed, is the 3D information acquired by a monocular or stereo based vision system. If the relative position of the cameras is known, 3D information can be processed using a computer vision method denoted as stereo triangulation.

One important aspect for solving the triangulation problem, is to have a good match correspondence between image points of both cameras. In case the correspondence fails, 3D information will be wrongly computed. Furthermore the correspondence problem will be more severe in case cameras don't have enough overlap, and also in case of high image distortion or cameras with different image resolutions.

In this thesis we will addresses the correspondence problem for the stereo case, and propose a method to increase the number of valid correspondence matches. The proposed approach is demonstrated in robotic soccer application scenario.

Esta página foi intencionalmente deixada em branco.

Conteúdo

Esta página foi intencionalmente deixada em branco.

Lista de Figuras

Esta página foi intencionalmente deixada em branco.

Lista de Tabelas

Esta página foi intencionalmente deixada em branco.

Lista de Acrónimos

3D 3 Dimensions

2D 2 Dimensions

BFL Bayesian Filtering Library

BP Belief Propagation

BRIEF Binary Robust Independent Elementary Features

CC Cross-Correlation

CPU Central Processing Unit

DP Dynamic Programming

DoG Diferences of Gaussians

FAST Features from Accelerated Segment Test

FOV Field of View

GC Graph Cut

GPS Global Position System

GPU Graphics Processing Unit

ID3 Iterative Dichotomiser 3

INS Inercial Navigation System

ISEP Instituto Superior de Engenharia do Porto

LASER Light Amplification by Stimulated Emission of Radiation

LiDAR Light Detection and Ranging

LSA Laboratório de Sistemas Autónomos

MSL Middle Size League

NCC Normalized Cross Correlation

NMS Non Maximum Suppression

OpenCV Open Source Computer Vision Library

ORB Oriented FAST and Rotated BRIEF

PAN Panning

PPM Matriz de Projeção Projetiva

RAM Random Access Memory

SAD Sum of absolute differences

SIFT Scale Invariant Feature Transform

SO Scanline Optimization

SSD Sum of Squared Differences

SURF Speeded-Up Robust Features

TILT Tilting

ZNCC Zero-Mean Normalized Cross-Correlation

Capítulo 1

Introdução

1.1 Âmbito da dissertação

A utilização dos sistemas robóticos têm vindo a expandir-se em várias áreas de aplicação que vão desde, aplicações no campo da medicina, a fins militares ou exploração de meios marinhos. Alguns exemplos de sistemas da robótico são: braços robóticos, sistemas de transporte autónomos, veículos aéreos autónomos ou semi-autónomos, submarinos operados remotamente.

Os sistemas autónomos são sistemas que necessitam de informação adquirida pelos mesmos, de forma a conseguirem deliberar sobre as ações a tomar para conseguir desempenhar as suas funções[8]. Assim sendo, estes tipo de sistemas robóticos depende em grande escala de informação sensorial para desempenhar as suas funções. Alguns dos sensores utilizados em sistemas de robóticos são:

- Sensores de visão (câmaras);
- Sistemas *Light detection and Ranging* (LiDAR);
- Sistemas de *Global Positioning System* (GPS);
- Sistemas *Inercial Navigation System* (INS).

Umas das principais fontes de informação sensorial utilizada em muitos sistemas robóticos são sistemas baseados em visão. Através destes sistemas, é possível obter vastas quantidades de informação acerca do meio envolvente do robô, permitindo-lhe assim navegar nesse meio e efetuar as suas tarefas.

A extração da informação para a navegação com base nos dados das câmaras é alcançada recorrendo a técnicas de visão computacional. Através do processamento das imagens é possível detetar objetos, reconhecer marcas no terreno, e obter informação acerca da posição de objetos relativamente ao robô. Existem múltiplas abordagens em visão computacional para a obtenção de informação tridimensional, sendo uma delas a visão *stereo*. Para obter a informação tridimensional de um objeto, é necessário um par de imagens do objeto com uma câmara em perspetivas diferentes, ou de múltiplas câmaras em posições diferentes. Extraindo um conjunto de pontos do objeto em ambas as imagens e, admitindo que esses pontos são iguais em ambas as imagens, ou seja, existe correspondência entre os pontos de ambas as imagens, é possível determinar a posição tridimensional desses pontos, recorrendo a técnicas, como por exemplo: triangulação e retificação.

No entanto, a correspondência automática de pontos entre imagens é um processo complexo, no qual nem sempre é possível obter uma boa correspondência entre os pontos comuns às imagens. Existem múltiplas abordagens que vão desde, a aplicação de métodos de correspondência denso ou dispersos, utilização da restrição epipolar, entre outros.

Este processo pode ser dificultado no caso da utilização de imagens com resolução diferente, ou quando existe um *overlap* reduzido entre as perspetivas das imagens.

Nesta dissertação foi desenvolvido um método de correspondência de pontos para aplicação em sistemas de visão multi-câmara, no qual sejam utilizados câmaras com resolução de imagem diferente ou cujas perspetivas das imagens contêm um *overlap* reduzido.

1.2 Enquadramento e Motivação

O Laboratório de Sistemas Autónomos[1] (LSA) do Instituto Superior de Engenharia do Porto(ISEP), tem vindo a desenvolver diversos projetos nas áreas da robótica marinha, terrestre e aérea, como é possível observar na Figura 1.1.

Um aspeto comum a todos os meios de operação dos robôs é o facto de todos necessitarem de informação sensorial para navegar, sendo os sistemas de visão computacional um dos meios de perceção sensorial mais utilizado. Desta forma, é possível extrair grandes quantidades de informação do meio envolvente, contribuindo para uma melhor navegação.

[1] http://www.lsa.isep.ipp.pt

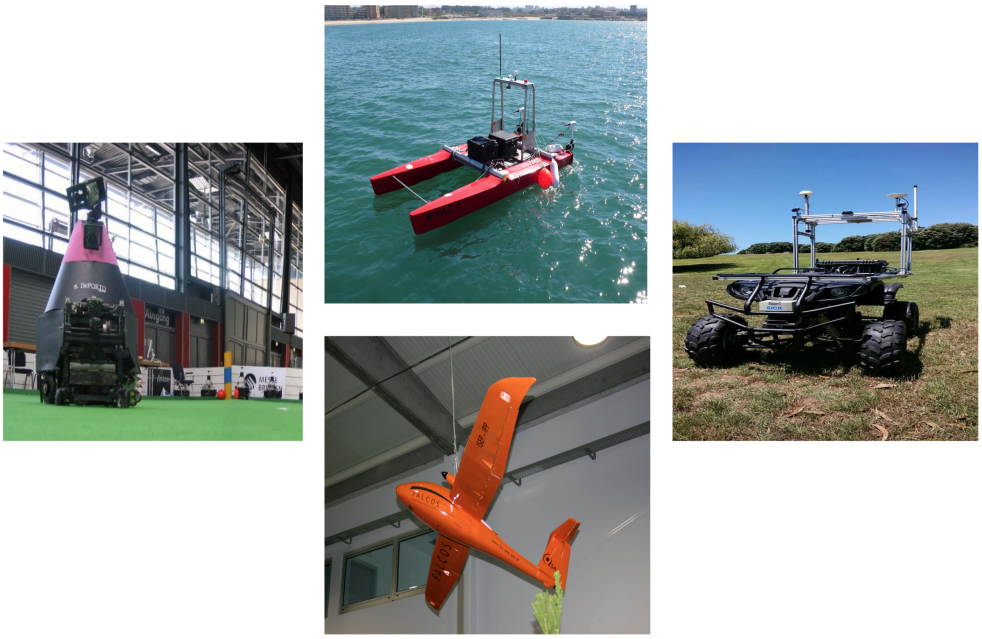


Figura 1.1: Exemplos de alguns robôs existentes no LSA.

De forma a melhorar navegação do robô no meio envolvente, utiliza-se a informação tridimensional, uma vez que esta permite relacionar a posição de um objeto no mundo com o robô. Para obter a informação tridimensional é necessário que o sistema de visão esteja calibrado e seja utilizado um método para estabelecer a correspondência entre pontos.

Como tal, o problema adjacente a esta dissertação surge da necessidade de obter uma boa correspondência entre pontos de imagens adquiridas nos sistemas de visão multi-câmara. A complexidade da correspondência de pontos pode aumentar com os seguintes aspetos:

- Resolução de imagem diferentes;
- Câmaras com características intrínsecas diferentes;
- Imagens com perspetivas diferentes e *overlap* reduzido.

A plataforma no qual se integrou o trabalho desta dissertação foi a equipa de robôs futebolistas do ISePorto, que competem na *middle size league* (MSL). O futebol robótico é uma área da robótica que tem evoluído significativamente em diversos campos nos últimos anos, nomeadamente nos campos da inteligência artificial e da perceção. A

competição da liga MSL consiste num jogo de futebol entre duas equipas de cinco robôs autónomos, sendo quatro deles jogadores de campo e um guarda-redes, como é possível observar na Figura 1.2.

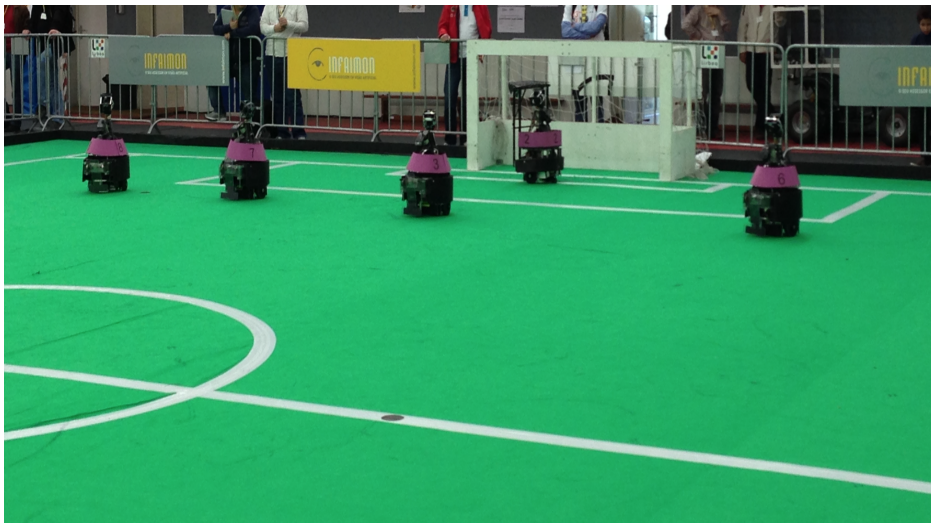


Figura 1.2: Equipa de futebol robótico ISePorto.

Uma vez que os robôs futebolistas utilizam a informação tridimensional dos sistemas de visão multi-câmara para a navegação e deteção de obstáculos[7], a utilização de um método de correspondência eficiente irá permitir um melhor desempenho nas tarefas a elaborar. Deste modo, o principal objetivo deste trabalho consistiu no desenvolvimento de uma abordagem para melhorar a correspondência de pontos nos sistema de visão multi-câmara.

1.3 Cenários de aplicação

A abordagem proposta para aplicação foi concebida para ser integrada com sistemas de visão multi-câmara, com o intuito de melhorar o método de correspondência de pontos entre imagens, que por sua vez permite melhorar o desempenho do sistema 3D.

Algumas áreas de utilização deste método na área robótica podem ser:

- Integração na plataforma de futebol robótico;
- Sistemas de visão multi-câmara;

- Sistemas robótico aéreos, terrestres e marítimos, dotados de meios de perceção visual.

1.4 Objetivos

Os objetivos desta dissertação abordam a problemática da correspondência de pontos entre imagens. Assim sendo, os objetivos delineados para esta tese são:

- Estudo de métodos de deteção e correspondência de *features*;
- Avaliação de métodos de deteção de *features* para implementar na dissertação;
- Desenvolvimento de uma abordagem de correspondência de pontos com baixa taxa de correspondências incorretas;
- Validação da abordagem desenvolvida no cenário do futebol robótico;
- Caracterização dos resultados gerados pela abordagem desenvolvida;
- Avaliação da abordagem desenvolvida comparando com um método de correspondência de *features*.

1.5 Estrutura do Relatório

No segundo capítulo é apresentado um estudo acerca de diferentes métodos de correspondência *stereo*, métodos de extração de *features* e correspondência de *features*, por forma a relacionar o trabalho a desenvolver nesta dissertação com trabalhos ou aplicações já existentes.

O terceiro capítulo aborda os conceitos fundamentais para a compreensão e desenvolvimento da abordagem de correspondência apresentada na presente dissertação.

No capítulo quatro é descrita a abordagem conceptual, desenvolvida nesta dissertação.

A implementação do projeto é descrita no quinto capítulo, no qual são mencionadas as diferentes etapas do software implementado.

No sexto capítulo, são apresentados os resultados obtidos ao longo do processo de implementação e teste.

Por último, no sétimo capítulo são apresentadas as conclusões e o trabalho futuro.

Capítulo 2

Estado da Arte

O estudo inicial de um determinado tema é essencial para uma melhor compreensão do mesmo, sendo que representa um grande impacto no rumo a tomar no desenvolvimento do projeto. Desse modo, foram estudadas diferentes abordagens de correspondência *stereo*, bem como alguns trabalhos relacionados com a deteção e *matching* de *features*.

2.1 Correspondência *stereo*

A perceção da localização e das características tridimensionais de objetos através de imagens com diferentes perspetivas, tem uma importância significativa na reconstrução tridimensional do objeto. Para extrair a informação tridimensional do objeto, é necessário conseguir estabelecer uma relação entre as duas imagens, ou seja, é necessário garantir que o mesmo objeto é visualizado em ambas as imagens. A relação entre um objeto ou qualquer ponto do mundo projetado em duas imagens pode ser classificado como um problema de correspondência. Através dos pontos correspondentes nas imagens, é possível obter a informação tridimensional do objeto, podendo essa informação ser extraída recorrendo mapas de disparidade, tirando partido do conhecimento da posição e orientação relativas entre câmaras [9].

A problemática de correspondência *stereo* pode ser definida como a procura de pares de pontos correspondentes em duas imagens, no qual esses pontos correspondem a um ponto no mundo [10].

Com o intuito de obter correspondências corretas, foram geradas restrições com base nos princípios físicos associados do *stereo* [11, 12]. Algumas das restrições mais comuns são:

- Restrição epipolar: os pontos correspondentes devem encontrar-se posicionados sobre as respetivas linhas epipolares(Figura 2.1(b));
- Restrição de continuidade: os valores da disparidade têm tendência a variar pouco ao longo da superfície incidente;
- Restrição de exclusividade: um ponto numa imagem deve ter apenas um ponto correspondente na outra imagem;
- Restrição de ordenação: a ordem dos pontos situados sobre as linhas epipolares é a mesma(Figura 2.1(a));
- Restrição de Oclusão: a descontinuidade numa imagem corresponde à oclusão na outra imagem e vice-versa.

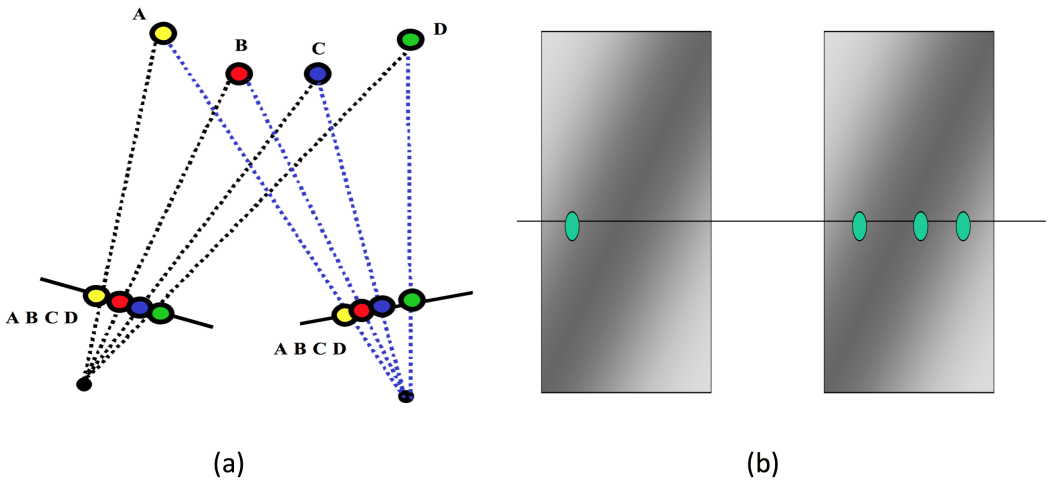


Figura 2.1: (a) Restrição de ordenação[1]. (b) Restrição epipolar[2].

Como forma de superar problemática em causa, foram desenvolvidos diversos métodos de correspondência *stereo*, podendo estes ser agrupados em dois tipos: métodos de correspondência dispersa e métodos de correspondência densa [13].

[1]http://www.cs.ubc.ca/ lowe/425/slides/6-Stereo-6up.pdf, acedido em 15/02/2015
[2]http://www.cse.psu.edu/ rtc12/CSE486/lecture09_6pp.pdf, acedido em 15/02/2015

Os métodos de correspondência dispersa [14, 15] são baseados na correspondência de *features* em imagens, sendo estas *features* provenientes de detetores de pontos de interesse, limites (*edges*) ou cantos (*corners*). Com este método, apenas é possível obter correspondências fiáveis nos píxeis situados em zonas da imagem com textura [16]. A vantagem deste tipo de métodos reside na produção de resultados precisos em situações em que ocorrem grandes deslocamentos entre imagens [17].

Os métodos de correspondência densa procuram estabelecer correspondência entre todos ou quase todos os píxeis nas imagens, de forma a gerar estimativas de disparidade densa. O intuito destes métodos consiste na propagação das zonas de estimação da disparidade, de forma a ser possível inferir zonas de elevada e baixa textura. A motivação para aplicação dos métodos de correspondência densa é que, maioria dos píxeis na imagem podem ter correspondência [16]. Nos últimos anos, os métodos de correspondência densa tornaram-se muito populares, pelo motivo que muitas aplicações requerem reconstrução e modelação 3D.

Os algoritmos de correspondência densa podem ser divididos em duas grandes categorias [18]: locais e globais.

2.1.1 Algoritmos locais e algoritmos globais

Tal como foi mencionado anteriormente, os algoritmos de correspondência podem ser divididos em duas categorias.

Os algoritmos de correspondência local comparam a semelhança dos valores de intensidade dos píxeis numa janela, em torno de pontos dos quais se pretende obter a respetiva correspondência [12]. Um parâmetro de custo é utilizado para determinar qual a melhor correspondência. Nesta abordagem, a escolha do tamanho da janela de comparação é importante para obter o mapa de disparidade, levando a uma determinação de correspondência mais precisa.

Com este tipo de algoritmos obtêm-se resultados consistentes, com tempo de execução relativamente reduzido, permitindo ser utilizado em muitas aplicações em tempo real [19].

O processo de aplicação dos algoritmos passa por primeiro definir duas janelas, uma janela de referência e uma janela de pesquisa. Quando é efetuada a comparação dos píxeis, a janela de referência é movida ao longo da janela de pesquisa, efetuando a comparação dos píxeis da janela de referência a cada posição. Tal como foi mencionado anteriormente, a escolha do tipo de janela de pesquisa é um aspeto importante, sendo

que muitos algoritmos utilizam uma janela com um tamanho fixo. Contudo, existem outros algoritmos que utilizam janelas de tamanho variável cujo intuito é otimizar o processo de correspondência [20].

A escolha do parâmetro de custo encontra-se associado à velocidade de aplicação do algoritmo de correspondência. Algumas técnicas utilizadas são *Sum of absolute differences* (SAD) e o *Sum of squared differences* (SSD) [21, 22]. Ao contrário das técnicas anteriores, as baseadas em correlação, tais como o *normalized cross-correlation* (NCC) e o *zero-mean normalized cross correlation* (ZNCC), têm em consideração as diferenças dos ganhos nas janelas de correspondência devido à normalização. Estas técnicas são invariantes variações de intensidade nos píxeis, no entanto o ZNCC consegue ser mais robusto que o NCC, conseguindo compensar variações de intensidade uniforme [12, 23]. Estas propriedades são muito úteis em algoritmos de visão *stereo*, especialmente quando são aplicados a cenários exteriores, no qual as condições de iluminação não são constantes.

Os algoritmos globais seguem uma abordagem que consiste na propagação da informação da disparidade em torno da vizinhança de um ponto. De forma a realizar propagação, é necessário aplicar a minimização de uma função de energia a uma grande região da imagem [24].

Os resultados gerados por este tipo de algoritmos contêm grande precisão, no entanto o custo computacional destes algoritmos é significativamente alto, deixando de ser viáveis para aplicações em tempo real [19].

Tal como foi referido, as abordagens de otimização global consideram o problema de correspondência como a procura de uma zona de disparidade D que minimiza a equação de energia que se segue [25]:

$$E(D) = E_{data}(D) + E_{smooth}(D) \tag{2.1}$$

Sendo que, o termo $E_{data}(D)$ mede em que grau a função de disparidade encontra-se de acordo com o par de imagens de entrada. O termo $E_{smooth}(D)$ codifica as diferenças de disparidade entre os píxeis vizinhos em torno de cada ponto p.

Alguns dos algoritmos com melhor desempenho são: *Graph Cut* (GC) e *Belief Propagation* (BP) [25]. Apesar dos algoritmos dos tipos referidos [26, 27, 28] obterem ótimo desempenho, estes são normalmente lentos, devendo-se principalmente à sua natureza

iterativa e aos requisitos de memória elevados, tornando-se inadequados para dispositivos com limitações ao nível de memória e de capacidade de processamento. No entanto, algoritmos como *Scanline Optimization* (SO) [18, 29] e *Dynamic Programming* (DP) [18, 30] permitem obter resultados relativamente precisos, todavia, estes algoritmos apenas podem ser aplicados quando o problema de correspondência é de apenas uma dimensão. O tempo de execução destes algoritmos é menor face aos algoritmos globais enunciados inicialmente, tornando possível a aplicação dos mesmo em sistemas embebidos [31].

2.1.2 Resumo de correspondência *stereo*

Nesta secção foram abordados diferentes métodos de correspondência *stereo*, sendo realizada uma breve descrição dos tipos de métodos e apresentadas algumas vantagens e desvantagens.

Os métodos de correspondência dispersa permitem obter correspondências fiáveis em zonas com textura, sendo estes métodos mais adequados para situações no qual ocorreram grandes deslocamentos entre imagens.

No que diz respeito aos algoritmos a utilizar nos métodos de correspondência densa, foram mencionadas duas categorias. Os algoritmos locais conseguem ser mais rápidos que os algoritmos globais, no entanto os algoritmos globais oferecem melhores resultados face aos algoritmos locais. É de realçar que, apesar dos seus resultados serem inferiores aos algoritmos globais, os algoritmos locais oferecem resultados mais ou menos precisos, permitindo ser aplicados em tempo real. O tempo de processamento do algoritmo local encontra-se associado ao parâmetro de custo e ao tamanho das janelas de referência e de pesquisa. Os parâmetros de custo que obtêm melhores resultados face a mudanças de intensidade são o NCC e o ZNCC.

Os algoritmos globais obtêm melhor desempenho na correspondência quando comparado com algoritmos locais [25], uma vez que os algoritmos com a melhor classificação na avaliação de Middlebury[3] são os globais. Apesar de terem um grande desempenho, estes algoritmos são geralmente mais lentos e têm requisitos de memória elevados. O SO e DP são algoritmos globais cujo tempo de execução é mais curto quando comparado com outros algoritmos globais, no entanto apenas podem ser aplicados quando o problema de correspondência é apenas numa dimensão.

As restrições de correspondência podem ser integradas com os métodos de corres-

[3] http://vision.middlebury.edu/stereo/eval/, acedido em 15/02/2015

pondência dispersa e densa, permitindo simplificar o problema de correspondência. Um exemplo de simplificação é o uso da restrição epipolar, em que o problema de correspondência passa apenas a existir numa dimensão, dado que, com a retificação das imagens, as linhas epipolares ficam paralelas ao plano da imagem. Com a integração de um algoritmo local sobre as linhas epipolares paralelas, seria possível obter uma correspondência mais precisa num espaço de tempo menor.

2.2 Detetores e descritores de pontos de interesse

A designação de *feature* em visão computacional, pode ser atribuída a um ponto da imagem que representa uma determinada particularidade de interesse, nomeadamente valores de intensidade de píxeis diferentes numa determinada zona, cantos (*corners*), zonas de fronteira (*edges*), linhas ou curvas.

Os detetores e descritores de *features* desempenham um papel importante em aplicações de visão computacional, tais como: reconhecimento de objetos, reconstrução de imagens, mapeamento para navegação, entre outras.

Estes detetores assentam no princípio de que os pontos detetados nas imagens definem uma dada estrutura, na qual esses pontos são invariantes à rotação, translação, escala e transformação *affine* numa sequência de imagens. Esses pontos são denominados *local invariant features*, sendo que estes permitem caracterizar as diferentes imagens, conseguindo correlacionar objetos entre imagens [3].

Alguns aspetos importantes relativos ao processo de deteção de pontos, é que este processo deve ser preciso e repetitivo, ou seja, o mesmo ponto deve ser detetado sempre na mesma zona da imagem. Estas *features* devem também ser distintas para que seja possível distinguir diferentes objetos em imagens diferentes.

Uma vez que as *features* foram extraídas das imagens, estas devem ser transformadas num formato adequado, para que possam ser comparadas pelo *matcher*. O formato referido é denominado de *descriptor*[3]. O principal objetivo do *descriptor* consiste em mapear uma pequena janela em torno do *keypoint*, guardando informação relevante para a sua caracterização [32]. Esta informação, que pode consistir apenas numa medida de distância, é utilizada para estabelecer correspondência entre *keypoints*.

Os descritores devem ser distintos e ao mesmo tempo robustos às mudanças nas condições de visualização e aos erros provenientes do detetor [33]. Para garantir que o processo de *matching* seja o mais rápido possível, é necessário que os descritores apre-

sentem um consumo de memória eficiente.

O processo denominado por *matcher* é responsável por estabelecer a correspondência entre *keypoints* de diferentes imagens, com base nas informações dos respetivos descritores de cada *keypoint*.

De modo a identificar métodos para a deteção de *features*, referindo as suas vantagens e desvantagens, foi efetuado um estudo com base em artigos já existentes [1, 34, 35].

2.2.1 SIFT

O *Scale Invariant Feature Transform*(SIFT) foi desenvolvido por Lowe[4] apresentando uma combinação entre um detetor de regiões de interesse, *Diferences of Gaussians* (DoG) e o correspondente descritor.

Este método pode ser dividido em 4 etapas principais: *space extrema detection*, *keypoint localization*, *orientation assignment* e *keypoint descriptor*[1]. Das etapas referidas, apenas três fazem parte do detetor, sendo que a última etapa faz parte do descritor.

Na primeira etapa são obtidos os *keypoints* a partir da imagem. O detetor procura em escalas diferentes da imagem, aplicando uma convolução com um filtro gaussiano nas diferentes escalas das imagens. O *keypoint* é obtido a partir dos máximos ou dos mínimos retornados pelo DoG, tal como é ilustrado na Figura 2.2.

[4]https://scholar.google.com/citations?user=8vs5HGYAAAAJ, acedido em 15/02/2015

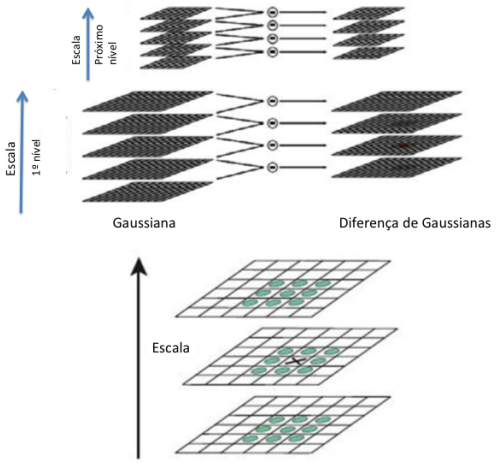


Figura 2.2: Construção do espaço de escala obtendo a DoG (pirâmide), adaptado de [1]

A segunda etapa consiste em rejeitar os pontos obtidos na fase anterior que sejam suscetíveis a ruído. Uma possível técnica consiste na realização de um estudo dos valores dos píxeis em redor do *keypoint*, de forma a identificar e posteriormente rejeitar os pontos com baixo nível de contraste, sendo esses pontos mais sensíveis a ruído[2].

Na fase denominada por *orientation assignement* é atribuída a componente de orientação a cada *keypoint*, sendo que esta pode apresentar várias orientações associadas. A escala no qual o *keypoint* foi extraído é obtida através de uma imagem suavizada(L) por um *kernel* gaussiano, de forma a que todos os pontos serem invariantes à escala.

Assim sendo, para cada imagem suavizada numa dada escala é obtida a magnitude do *keypoint* e a orientação, para cada píxel na vizinhança do ponto de interesse. Através da orientação é obtido um histograma com os gradientes das orientações, sendo que as orientações atribuídas ao *keypoint* correspondem aos maiores picos do histograma[2].

O descritor é criado a partir do gradiente da magnitude e da orientação na região em torno do *keypoint*, ao qual é aplicado um *kernel* gaussiano com o mesmo nível da escala do *keypoint*. A recolha da amostra é realizada com uma janela de 16x16 em torno da região de interesse. As amostras obtidas são então acumuladas em histogramas de orientação que sumarizam o seu conteúdo numa janela de 8x8, no qual o tamanho das setas correspondem à soma dos gradientes da magnitude com mesma direção na região. Na Figura 2.3 é possível observar esse processo.

O SIFT oferece um desempenho otimizado na deteção de *features*, uma vez que obtém

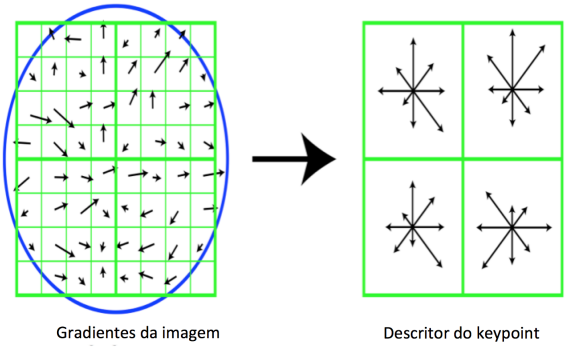


Figura 2.3: Construção do descritor de *keypoints*, adaptado de[2]

uma boa correspondência entre objetos de imagens diferentes. No entanto, este não permite o seu uso em aplicações que exijam uma execução em tempo real, sendo esta a sua maior desvantagem [1]. Esta desvantagem deve-se ao elevado custo computacional, principalmente ao gerar o descritor, pois este apresenta um tamanho considerável para cada *keypoint*, levando a que o seu tempo de processamento seja elevado. Contudo, aplicações do SIFT em GPU permitem reduzir significativamente o tempo de processamento do método [36].

2.2.2 SURF

Um outro detetor que pode ser designado como uma alternativa ao SIFT é o *Speeded Up Robust Features*(SURF) [3]. Este método combina um detetor de zonas de interesse Hessian-Laplace com um descritor de *features* com base em gradientes de orientação. O detetor faz uso de filtros *box* 2D simples(*Haar wavelets*) para o seu processamento, como alternativa a derivadas gaussianas, como se pode observar na Figura 2.4.

Os filtros de *box* têm efeitos aproximados aos *kernels* com filtros de derivadas, embora os filtros de *box* possam ser avaliados eficientemente utilizando imagens integrais, ou seja, estes filtros podem ser aplicados na imagem normal.

Esta avaliação requer o mesmo número de constante de pesquisas, independentemente da escala, sendo que deixa de ser necessário aplicação de uma pirâmide gaussiana. Apesar desta simplificação, o SURF consegue obter uma boa repetibilidade em comparação com detetores que fazem uso de derivadas Gaussianas normais.

O descritor do SURF apresenta uma estratégia de *binning* espacial semelhante ao

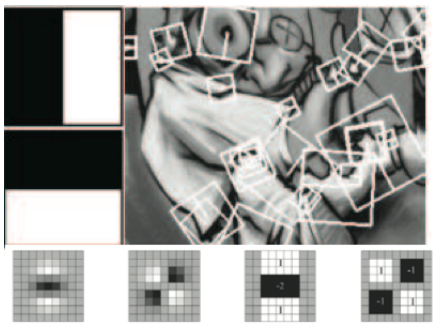


Figura 2.4: Processamento de imagem com recurso a filtros de *box* 2D do SURF[3]

SIFT, sendo que, o descritor do SURF divide a zona em torno da *feature* numa grelha de 4x4 [3]. No entanto, o SURF apenas processa um conjunto de sínteses estatísticas($\sum dx$, $\sum |dx|$, $\sum dy$, $\sum |dy|$), resultando num descritor com uma dimensão de nx64, no qual n é numero de *keypoints*. Contrariamente, o SIFT realiza um histograma com gradientes de orientação para zona de cada *keypoint*, resultando num descritor cujas dimensões são nx128.

Este método consegue obter um desempenho ao nível do SIFT, no entanto o SURF consegue ser mais rápido que o SIFT em termos de processamento [34]. Apesar de tudo, o SURF continua a ser um método pouco viável para aplicações em tempo real.

2.2.3 FAST e BRIEF

O *Features from Accelerated Segment Test*(FAST) é um detetor de *features* utilizado primariamente para a deteção de cantos [35, 4]. Ao contrário dos métodos referidos anteriormente, o FAST apenas funciona como um detetor de *features*, não tendo a funcionalidade de elaboração do descritor da imagem.

Este método avalia o valor de intensidade dos píxeis num círculo de raio r, em torno de um ponto p, sendo este o ponto que se pretende validar como *feature*. O círculo tem um raio igual a 3 sendo composto por 16 píxeis, como se pode observar na Figura 2.5.

A comparação dos píxeis do círculo com o ponto a testar como possível *feature* irá resultar numa classificação dos píxeis do círculo. No caso do valor de intensidade de um píxel do círculo ser maior que o valor de intensidade do ponto p mais o valor de limiar(t), então este píxel do círculo é considerado "claro". Na eventualidade do valor de intensidade do píxel do círculo ser menor que o valor de intensidade do ponto p mais

o limiar, então o píxel é considerado "escuro". Na situação do valor de intensidade do píxel do circulo encontra-se entre as condições mencionadas anteriormente, então o píxel é considerado "semelhante". O limiar(t) é definido pelo utilizador e permite aceitar ou rejeitar mais pontos consoante o seu valor.

O ponto p é aceite como *feature* se o conjunto de n píxeis em torno do círculo forem considerados "claros"ou "escuros", em que $n = 9$. O valor da variável n pode variar dependendo do tipo de algoritmo FAST a aplicar, no qual variam o raio do círculo e o número de píxeis que compõem o círculo.

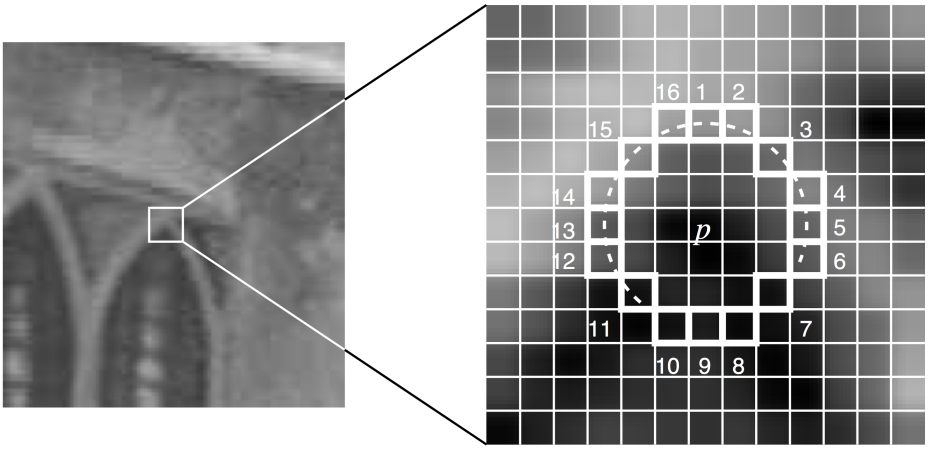


Figura 2.5: Teste de segmentação de 12 pontos do detetor de cantos numa janela da imagem [4]

O algoritmo *Iterative Dichotomiser 3*(ID3) é utilizado para otimizar a ordem no qual os píxeis do círculo são testado, resultando num processamento mais rápido.

O resultado do método descrito até agora produz alguns pontos positivos próximos uns dos outros, sendo que para resolver esse problema, é aplicado um *Non Maximum Suppression*(NMS), cujo intuito é refinar os dados e rejeitar esses pontos próximos uns dos outros. A aplicação deste método não compromete a sua velocidade, visto que este só é aplicado a um conjunto de pontos que passaram no teste de segmentação.

Como foi referido anteriormente, o FAST não tem implementado um descritor de *features*, o que faz com que este necessite de menos tempo para a extração de *features*, uma vez que apenas produz os pontos no qual encontrou as *features*, não produzindo informação adicional acerca da *feature* tal como a orientação.

Apesar do FAST não incorporar um descritor de *features*, este pode ser implementado com outros descritores. Um possível descritor de *features* a implementar é o *Binary Robust Independent Elementary Features*(BRIEF) [37]. Este descritor é composto por strings binárias, no qual permite o uso da distância de *Hamming* no *matcher*, que por sua vez é mais rápido a processar que a distância euclidiana. Antes de se proceder ao teste que vai gerar o descritor, é necessário aplicar um *kernel* gaussiano para suavizar a janela no qual o será testado o *keypoint*. O motivo pelo qual se aplica a suavização à janela de teste é por este ser sensível ao ruído. Através da janela elaborada em torno do *keypoint* a testar, são selecionados dois pontos dentro desta, de modo a comparar os valor de intensidade desses pontos. Na eventualidade do valor de intensidade do segundo ponto ser maior do que a intensidade do primeiro ponto, então é colocado o valor "1", caso contrário é colocado o valor "0".

O teste é repetido n vezes, sendo que esta variável pode tomar os seguintes valores: 128, 256 ou 512. O processo de escolha dos pontos implementado no BRIEF consiste num método que recolhe amostras de uma distribuição gaussiana isotrópica. O número de bytes que são requeridos para armazenar o descritor é dado por $n/8$, para cada *feature*.

O descritor BRIEF destaca-se face a outros abordados anteriormente, dado que apresenta melhor performance ao nível do tempo de processamento. Contudo, este apenas é parcialmente invariante à rotação, ou seja, quando existem grandes rotações, este apresenta uma eficácia reduzida [35].

2.2.4 ORB

O *Oriented FAST and Rotated BRIEF* (ORB) é um método que incorpora um detetor e um descritor de *features* [38]. Como o próprio nome indica, este método é baseado no detetor de *features* FAST e no descritor BRIEF. Ambos os métodos apresentam um bom desempenho e baixo custo computacional, no entanto estes têm algumas limitações, sendo mais notória a ausência parcial de invariância à rotação por parte do BRIEF, ou seja, este apenas consegue ser invariante a pequenas rotações. Face às presentes adversidades existentes nestes métodos, foi desenvolvido o ORB.

Na etapa de deteção de *features* é utilizado o detetor FAST, visto que este é dotado de um baixo custo computacional. Este detetor produz uma grande quantidade de *features* que dependem do seu limiar, embora muitas das *features* obtidas se encontrem situadas em extremidades. Para resolver essa situação, foi aplicada uma medida de

cantos de Harris, ordenando os *keypoints*(*features*) produzidos pelo FAST, de forma a serem escolhidos os melhores N pontos, sendo N definido pelo utilizador.

O FAST não produz *features* invariantes à escala, sendo desta forma aplicada uma pirâmide à imagem previamente à aplicação do detetor FAST.

A principal contribuição do ORB no detetor FAST consiste na adição de uma componente de orientação às *features* geradas pelo FAST. A orientação é obtida calculando o momento sobre a janela no qual o *keypoint* se encontra, sendo obtido o centroíde. Aplicando a função arco tangente às dimensões do centroíde, é possível obter a orientação do *keypoint*.

O descritor do ORB contempla as mudanças aplicadas ao detetor FAST. Como tal, este permite a utilização da informação da orientação do *keypoint*. Esta característica é alcançada aplicando uma rotação com a orientação do *keypoint*, aos pares de pontos utilizados no teste binário.

A aplicação deste método para introduzir a orientação no descritor, levou a uma perda da variância no descritor e a um aumento da correlação nos testes binários. De forma a solucionar este problema, foi desenvolvido um algoritmo de aprendizagem para a amostragem dos pontos utilizados no teste binário. Este algoritmo é desenvolvido através de uma série de possíveis testes binários sem repetições, recorrendo a várias imagens do conjunto de PASCAL 2006 [39]. O conjunto de aprendizagem foi obtido utilizando uma janela de 31x31, à qual se aplicou uma sub janela 5x5, sendo os pares de pontos obtidos dessas janelas. As mesmas especificações das janelas são aplicadas nos testes para o descritor.

Por fim, é realizada uma comparação entre os testes binários atuais e os testes binários do conjunto de aprendizagem, ao qual são descartados os testes com correlação acima de um limiar, até no final se obter 256 testes. Se, eventualmente, na primeira execução do algoritmo não forem conseguidos os 256 testes, o limiar de aceitação é ajustado e o processo de seleção dos testes é repetido.

O ORB é um detetor e um descritor relativamente rápido quando comparado com o SURF e com SIFT, chegando a ser duas vezes mais rápido que o SIFT [38].

Em adição, o ORB é ainda capaz de extrair um maior número de features em menos tempo, quando em comparação com o SIFT e SURF. Relativamente ao tempo de processamento do detetor, o ORB não é tão rápido quanto o detetor FAST, no entanto o ORB consegue obter *features* em escalas diferentes e ao mesmo tempo gerar informação acerca da orientação da *features*. Apesar do ORB implementar uma pirâmide para escalar a

imagem, este não é totalmente invariante à escala.

2.2.5 Resumo dos detetores e descritores

Nesta secção foram abordados vários detetores e descritores de *features*, tendo sido descritos os seus princípios de funcionamento e referidos os seus pontos fortes e fracos, principalmente em relação à aplicação destes em situações de tempo real. Nesta subsecção é realizada uma comparação com base em trabalhos já realizados sobre os detetores apresentados.

No que diz respeito ao tempo de processamento, o detetor mais rápido é o FAST, sendo que o SIFT é o detetor que apresenta um tempo de processamento [1]. O SURF, apesar de ser mais rápido que o SIFT, é lento em comparação com o FAST e com o ORB, sendo que o ORB é o detetor que mais se aproxima do FAST em termos de tempo de processamento.

Relativamente ao número de *features* detetadas nas imagens, o ORB produz grandes quantidades de *features*, contudo este é sensível ao ruído e a cantos nas imagens. O FAST também é um algoritmo que produz grandes quantidades de *keypoints*. Quanto à sensibilidade do detetor a diferentes luminosidades, o ORB e o FAST conseguem um bom desempenho, no entanto o SIFT obtém piores resultados.

No que refere ao par detetor/descritor, o FAST é um detetor que obtém um desempenho razoável com qualquer descritor, sendo que o FAST-BRIEF é a melhor opção em termos do tempo de processamento [35]. É importante ter em conta o descritor a implementar com o detetor no contexto do cenário da aplicação.

Com esta secção foi possível entender quais os métodos a utilizar para deteção de *features* e quais as suas potenciais aplicações. Os métodos como o SIFT e o SURF, oferecem um desempenho sólido, no entanto estes requerem um tempo de processamento maior. Já o ORB e do FAST cujo tempo de processamento é significativamente menor, sendo que o ORB obtém um bom desempenho na deteção de objetos.

2.3 Matchers

Nesta secção são abordados os *matchers* de *features* utilizados para estabelecer a

correspondência entre *features*.

2.3.1 *Matchers* de pontos de interesse

Após serem extraídas as *features* das imagens e ser obtido o respetivo descritor, é necessário implementar o processo de correspondência entre os *keypoints* de ambas a imagens, fazendo uso do *matcher* [5]. Esta fase pode ser dividida em duas partes, sendo que a primeira consiste em selecionar uma estratégia de correspondência, na qual são determinados os pontos caraterizados como correspondentes, realizando a comparação entre descritores. A segunda parte consiste na aplicação de algoritmos que permitam estabelecer os pares de pontos que são possíveis correspondências para, posteriormente passar à fase de comparação. Estes algoritmos devem ser eficientes para que o processo de correspondência seja rápido e preciso.

Uma estratégia utilizada para comparar os descritores, em situações no qual existem pequenos movimentos da câmara, consiste na aplicação da distância euclidiana. Com esta estratégia é necessário definir um limiar(distância máxima), sendo apenas retornadas as correspondências abaixo do limiar definido. Na eventualidade do limiar ser muito alto, vai existir um aumento no número de falsos positivos, ou seja, associações incorretas de pontos. Se o limiar for muito baixo, vão surgir muitos falsos negativos, sendo ignorados muitas associações corretas de pontos, como é possível observar na Figura 2.6. Este método apenas pode ser utilizado com descritores que estejam preparados para a sua implementação.

Outro método semelhante consiste na utilização da distância de Hamming, que apenas pode ser aplicada a descritores binários. Este método deve ser aplicado a *strings* com o mesmo tamanho, realizando uma comparação entre as duas *strings* e verificado qual o número de alterações necessárias para transformar uma *string* na outra.

Uma outra estratégia consiste em estabelecer uma correspondência com o vizinho mais próximo no espaço da *feature*. Uma vez que algumas *features* não têm correspondência, pode ser utilizado um limiar para reduzir o número de falsos positivos.

Tendo explorado os métodos para comparar os descritores, é então necessário um algoritmo para auxiliar na decisão sobre quais descritores devem passar para a fase de comparação. O método mais simples para encontrar as correspondências consiste na comparação de todos os pontos numa imagem com todos os pontos noutra imagem [5]. A grande desvantagem deste método é resultar num grande número de comparações,

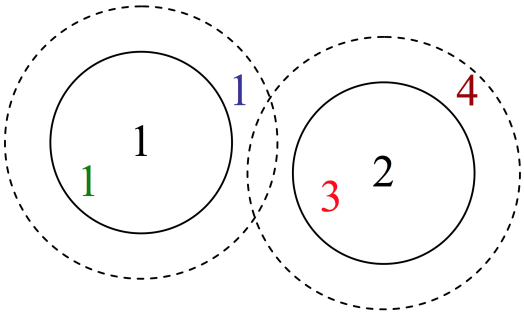


Figura 2.6: Exemplo de regulação do limiar de aceitação de correspondências [5]

aumentando o tempo de processamento consoante o número de *features* em ambas as imagens. Desta forma, este método não é viável para aplicações no qual seja necessário um processo de correspondência rápido.

Uma abordagem com um leque maior de aplicações consiste na utilização de estruturas de indexação, tais como árvores multi-dimensionais ou tabelas *hash*, tornando possível uma procura mais rápida de *features* próximas. Estas estruturas de indexação podem ser construídas para cada imagem ou podem ser globais para todas as imagens numa base de dados, tornando o processo mais rápido.

O método de tabelas de *hash* multi-dimensionais é um dos métodos mais simples. Este consiste num mapeamento do descritor em *"buckets"* de tamanho fixo com base numa função aplicada a cada vetor do descritor. No processo de correspondência, cada nova *feature* é "*hashed*"para dentro de um *bucket*, sendo que depois é realizada uma procura nos *buckets* mais próximos para encontrar potenciais correspondências. Existem outras técnicas com base nas referidas anteriormente, tais como o *locality sensitive hashing* [40, 41] ou o *parameter-sensitive hashing*[42].

Um outro método utilizado são as árvores multi-dimensionais, também conhecidas por *k-d trees* ou *kd-trees*. Este método divide o espaço multi-dimensional das *features* em eixos alternados, ou seja, as *features* são organizadas num diagrama semelhante à ramificação de uma árvore. A cada eixo da árvore é aplicado um limiar para otimizar a procura da correspondência, tal como se pode observar na Figura 2.7. Outras técnicas como árvores métricas [43] ou *hierarchical k-means trees* [44] são abordagens semelhantes à referida anteriormente.

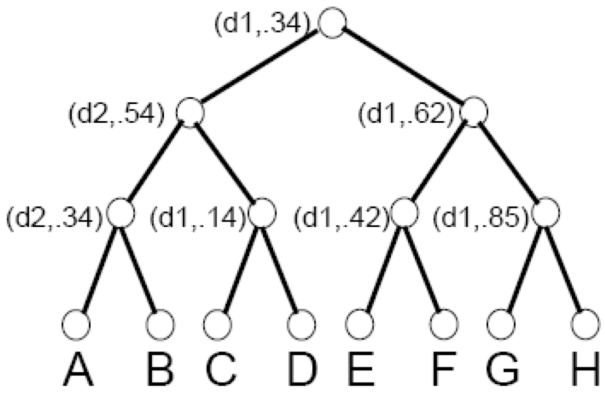


Figura 2.7: Exemplo de subdivisão dos pontos para correspondência com o algoritmo de árvore[5].

2.3.2 Resumo dos *matchers*

A escolha do algoritmo de correspondência depende principalmente do cenário de aplicação do *matcher* e dos seus requisitos. A abordagem de comparação de todas as *features* umas com as outras, garante que não é excluída nenhuma hipótese de correspondência. Dependendo do número de *features* o processo pode ser muito demorado, ao contrário dos métodos por indexação que são mais rápidos, dado que estes não realizam uma associação de *features* extensiva embora neste método possam ser ignoradas possíveis correspondências. Apesar desta desvantagem, estes métodos de indexação, especialmente os métodos com indexação em árvore, oferecem um bom desempenho.

Em relação à estratégia de correspondência, a sua escolha irá incidir principalmente no tipo descritor utilizado, sendo que no caso de descritores binários, a melhor estratégia a utilizar na comparação é a distância de Hamming.

Esta página foi intencionalmente deixada em branco.

Capítulo 3

Fundamentos Teóricos

Neste capítulo são apresentados alguns conceitos e fundamentos necessários para compreensão das temáticas abordadas na presente dissertação, tais como conceitos associados à visão computacional, nomeadamente visão *stereo*, extração de *features* e correspondência de *features*.

3.1 Introdução à visão computacional

O sistema de visão presente nos animais e nos seres humanos é dotado de grandes capacidades, dado que permite obter vastas quantidades de informação sobre o ambiente incidente sem que seja necessário contacto físico. Desta forma, é possível identificar e determinar a posição de objetos e características relevantes num dado cenário, permitindo assim uma interação dinâmica com o mundo, auxiliando na perceção e na navegação.

Afim de atribuir estas capacidades a máquinas que têm vindo a desempenhar tarefas, que outrora eram realizadas pelo ser humano, tem-se intensificado o estudo da problemática de visão, por forma a colmatar as diferenças que existem entre os sistemas de visão presente nos animais e o das máquinas. Deste modo, surgiram inúmeras soluções comerciais e industriais, que aplicam sistemas de visão em diversas áreas, tais como o controlo de qualidade em linhas de produção, diagnóstico médico, reconhecimento facial em câmaras de vigilância, entre outros. Uma área onde a visão computacional tem grande impacto é na robótica móvel, onde desempenha funções como a localização e a navegação dos robôs, a deteção e identificação de obstáculos, o mapeamento do ambiente incidente do robô e os algoritmos de aprendizagem com base na informação visual do robô.

O processo de obtenção da informação sensorial visual consiste inicialmente na aquisição de uma ou mais imagens provenientes de, pelo menos, uma câmara. O processo de identificação de objetos ou características de interesse depende do método a aplicar, sendo que alguns deles consistem na deteção de pontos de interesse ou conjuntos de píxeis com determinadas características, que permitem relacionar objetos ou padrões no mundo.

3.2 Projeção perspetiva

O processo que permite o mapeamento de pontos do mundo em pontos na imagem é denominado projeção perspetiva. O cálculo do modelo de projeção é efetuado tendo em conta o modelo do sensor de imagem.

Com base nas características do sensor de imagem, o cálculo da projeção perspetiva pode ser realizada com base no modelo *pinhole*. Este modelo de câmara é o mais comum, sendo que considera que a passagem dos raios de luz é efetuada através de um pequeno orifício antes destes serem projetados no plano da imagem. Com esta afirmação torna-se possível assumir que um ponto no mundo corresponde apenas a um ponto no plano da imagem. O modelo *pinhole* caracteriza a câmara através do centro ótico $C.O.$, o plano do centro ótico R^c e o plano de imagem R^i, como se pode observar na Figura 3.1. A distância que separa os dois planos é designada distância focal f.

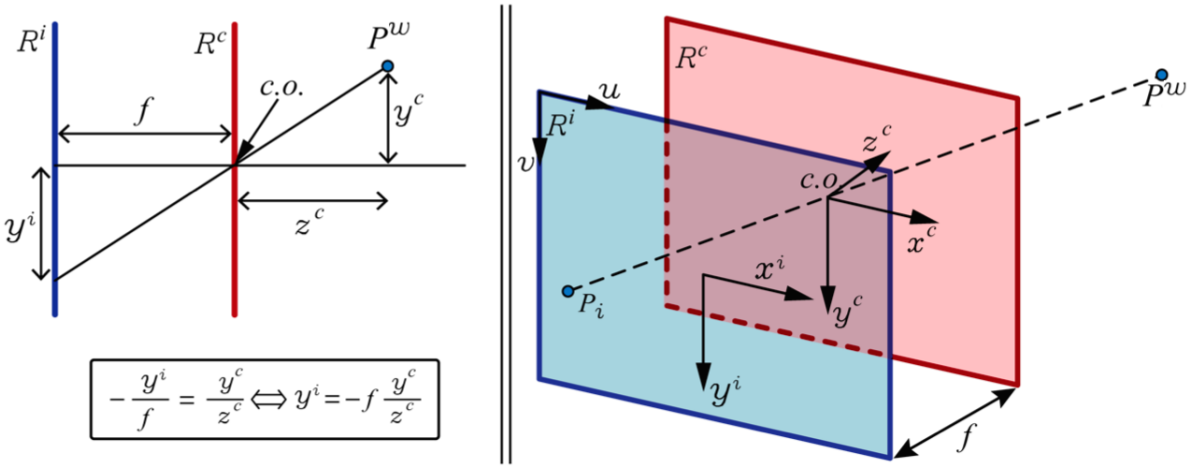


Figura 3.1: Modelo da câmara *pinhole*. À esquerda encontra-se uma representação 2D do modelo *pinhole*. À direita está uma representação 3D com o plano da câmara e da imagem.[6]

Dado que existe uma relação entre os dois planos, a projeção perspetiva pode ser descrita pela equação linear 3.1:

$$\begin{bmatrix} x^i \\ y^i \end{bmatrix} = -\frac{f}{Z^c} \begin{bmatrix} X^c \\ Y^c \end{bmatrix} \tag{3.1}$$

Onde X^c, Y^c e Z^c são as coordenadas do ponto no mundo P^w no referencial da câmara e x^i e y^i são as coordenadas desse ponto no plano da imagem.

Com base no modelo *pinhole* da câmara, o raio de luz ao passar no orifício do centro ótico vai produzir uma inversão na imagem, sendo esta a razão por detrás do sinal negativo no segundo membro da equação 3.1. De forma a colmatar esta situação, realiza-se uma inversão à imagem, o que corresponde a colocar o plano da imagem entre o centro ótico e o ponto no mundo P^w[6]. Assim, reformula-se a equação anterior, obtendo a equação 3.2.

$$\begin{bmatrix} x^i \\ y^i \end{bmatrix} = \frac{f}{Z^c} \begin{bmatrix} X^c \\ Y^c \end{bmatrix} \tag{3.2}$$

A relação entre os pontos no referencial do mundo(X^w, Y^w, Z^w) e os pontos no referencial da câmara podem ser obtidoss através de uma transformação composta por uma rotação e uma translação, em coordenadas homogéneas:

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \end{bmatrix} = \begin{bmatrix} R|\mathbf{t} \end{bmatrix} \begin{bmatrix} X^w \\ Y^w \\ Z^w \end{bmatrix} \tag{3.3}$$

A matriz de rotação(R) contém a orientação da câmara num dado referencial e $\mathbf{t}$ é o vetor de translação que contém a posição da câmara nesse mesmo referencial. A matriz R e o vetor $\mathbf{t}$ são resultantes do processo de calibração dos parâmetros extrínsecos da câmara abordados posteriormente na secção 3.3.2.

A agregação das equações 3.2 e 3.3 gera a equação 3.4 que permite relacionar um ponto no mundo com o referencial normalizado da câmara, recorrendo à Matriz de Projeção Projetiva(PPM);

$$Z^c \begin{bmatrix} x^i \\ y^i \\ 1 \end{bmatrix} = PPM \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} \tag{3.4}$$

Em que a PPM pode ser representada por:

$$PPM = A \left[R|\mathbf{t} \right] \tag{3.5}$$

A PPM é composta pelos parâmetros extrínsecos referidos na equação 3.3 e pela matriz A que depende dos parâmetros intrínsecos da câmara, sendo estes parâmetros abordados na secção 3.3.1.

Com o cálculo da matriz PPM é possível mapear os pontos do mundo no referencial 2D da imagem, tendo por base a relação presente na equação 3.6, em que m^w representa um ponto no referencial do mundo e m^I representa o mesmo ponto no referencial da imagem.

$$m^I = PPMm^w \tag{3.6}$$

3.3 Calibração da câmara

Com a utilização dos sistemas de visão, é necessário obter as relações entre os pontos no plano do mundo para os pontos no plano da imagem e vice-versa. Para tal, torna-se necessário obter os parâmetros que permitem realizar essas transformações e que caracterizam o modelo da câmara, sendo estes designados de parâmetros intrínsecos e extrínsecos. Com os parâmetros intrínsecos é possível definir as características câmara e da lente, enquanto que os parâmetros extrínsecos definem o referencial da câmara face ao mundo num sistema de coordenadas pré-definido.

Por forma a obter os parâmetros da câmara é necessário realizar um procedimento de calibração, sendo um dos procedimentos o algoritmo de Tsai [45]. Este algoritmo, para além de gerar os parâmetros da câmara, fornece também os coeficientes de distorção da lente. Este método gera a informação de calibração com base no modelo *pinhole*.

3.3.1 Parâmetros intrínsecos

As características físicas da câmara, como a sua geometria interna e tipo de lente,

são representadas pela matriz dos parâmetros intrínsecos e pelo vetor de distorção da imagem, sendo apresentados nas equações 3.7 e 3.8. Estes parâmetros contêm informação referente à resolução da imagem, alinhamento do sensor de imagem e coeficientes de distorção da lente.

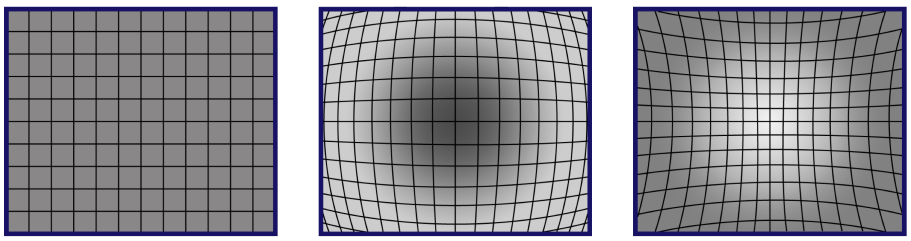
$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$K = (k_1, k_3, k_3, k_4, k_5) \tag{3.8}$$

Em que f_x e f_y são as distâncias focais, c_x e c_y são as coordenadas dos centros óticos e k_1, k_3, k_3, k_4 e k_5 representam os coeficientes associados à distorção da imagem.

Com a matriz A é possível relacionar as coordenadas do plano da câmara com o plano da imagem, ou seja, torna-se possível passar um ponto no referencial da câmara normalizado para um ponto na imagem em píxeis.

De acordo com o modelo *pinhole*, um ponto no mundo e um ponto na plano da imagem seriam colineares. Desta forma, um conjunto de retas no mundo ao serem projetadas no plano da imagem deveriam apresentar o mesmo aspeto, no entanto estas linhas sofrem uma deformação causada pela distorção da lente, como se pode observar na Figura 3.2. Este efeito pode ser mais visível quando a distância focal da lente é maior [46].



(a) Ausência de distorção

(b) Distorção radial em forma de barril

(c) Distorção radial em forma de almofada

Figura 3.2: Projeção de retas do mundo na imagem dependendo do tipo de distorção causada pela lente[6].

A distorção da lente pode ser caracterizada segundo duas componentes: radial e

tangencial. A distorção radial tem como causa a variação do ângulo de refração que varia consoante o aumento da distância ao centro da lente. Tendo uma refração menor na extremidade da lente, leva a que a distorção radial na imagem se manifeste em forma de barril, Figura 3.2(b). Ambiguamente, quando a refração na extremidade da lente é maior, a deformação na imagem é na forma de almofada, Figura 3.2(c).

A distorção tangencial é causada por um desalinhamento físico de elementos que constituem a lente. Nas câmaras atuais o impacto desta perturbação é reduzida, principalmente em lentes com distância focal fixa, sendo que nesses casos a distorção tangencial pode ser desprezada.

A presença da distorção na imagem gera incertezas no mapeamento dos pontos do mundo no plano da imagem, tornando necessário remover a distorção presente nas imagens recorrendo aos coeficientes de distorção radial e tangencial obtidos no processo de calibração. A remoção da distorção nas imagens pode ser feita recorrendo às seguintes equações [13]:

$$radial : x_{undistort} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.9}$$

$$radial : y_{undistort} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.10}$$

$$tangencial : x_{undistort} = x + (2k_4 y + k_5(r^2 + 2x_2) \tag{3.11}$$

$$tangencial : y_{undistort} = y + (k_4(r^2 + 2y_2) + 2k_5 x) \tag{3.12}$$

Onde x e y são os pontos na imagem com distorção e $x_{undistort}$ e $y_{undistort}$ são os pontos na imagem sem distorção. Os parâmetros k_1, k_2, k_3 são os coeficientes de distorção radial, enquanto que k_4 e k_5 são os coeficientes de distorção tangencial. A distância ao centro ótico é dada por r.

3.3.2 Parâmetros extrínsecos

Os parâmetros extrínsecos permitem transformar pontos do referencial do mundo para o referencial da câmara [46]. Para tal é necessário conhecer as relações de rotação e translação($[R|\mathbf{t}]$) entre os referenciais, por forma a ser possível estabelecer uma relação. Assim sendo, esta transformação é caracterizada por uma matriz de rotação (R) e por

um vetor de translação (**t**), como se pode observar na Figura 3.3.

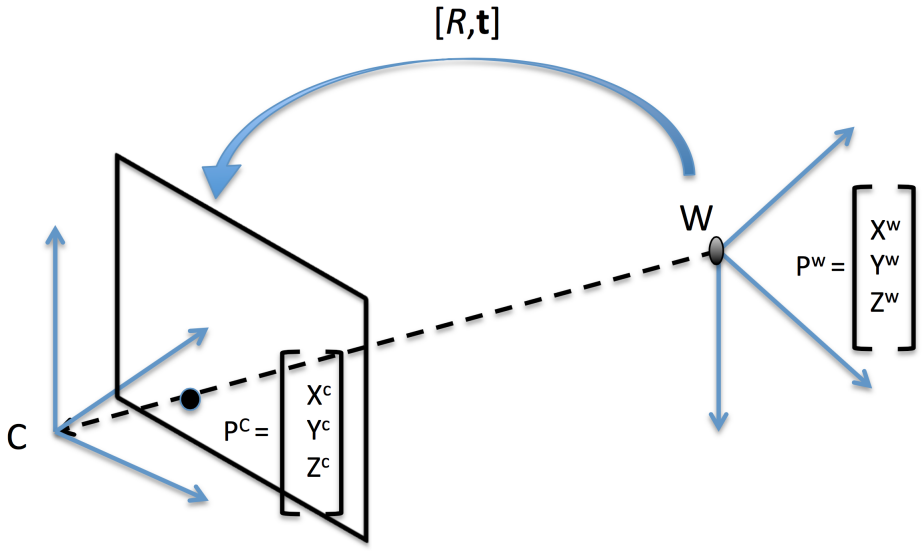


Figura 3.3: Projeção das coordenadas de um ponto para coordenadas no referencial da câmara.

A matriz de rotação pode ser representada pelos ângulos de Euler(ϕ, θ e ψ), através da seguinte equação:

$$R = R^T_{(x,\phi)} R^T_{(y,\theta)} R^T_{(z,\psi)} \tag{3.13}$$

A equação 3.13 é composta por três rotações puras, em que uma rotação pura consiste numa rotação que ocorre em apenas um eixo. As rotações puras que compõe a matriz de rotação são dadas pelas equações 3.14, 3.15 e 3.16, onde a rotação em torno do eixo X é dada por *roll*(ϕ), a rotação em torno do eixo Y é dada por *pitch*(θ) e a rotação em torno do eixo Z é dada por *yaw*(ψ) [13].

$$R_{(x,\phi)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{3.14}$$

$$R_{(y,\theta)} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.15}$$

$$R_{(z,\psi)} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

Na Figura 3.4 é possível observar o comportamento dos ângulos de Euler no referencial de uma câmara.

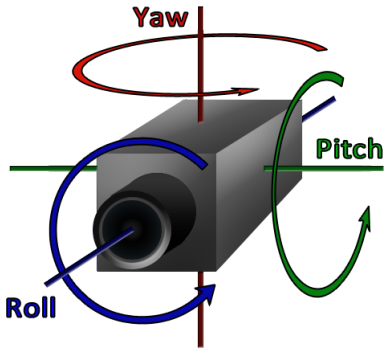


Figura 3.4: Representação dos ângulos de Euler no referencial de uma câmara, adaptado de [4].

A matriz de rotação R pode ser descrita em função dos ângulos de Euler na equação que se segue:

$$R = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\phi\sin\theta - \cos\theta\sin\psi & \sin\psi\sin\phi\sin\theta + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\psi\sin\theta\cos\phi + \sin\phi\sin\psi & -\sin\phi\cos\psi + \sin\psi\sin\theta\cos\theta & cos\theta\cos\phi \end{bmatrix} \tag{3.17}$$

O vetor de translação $\mathbf{t}$, que contém o ponto de origem do referencial do mundo no referencial da câmara, é dado pela seguinte equação:

$$\mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{3.18}$$

3.4 Visão *stereo*

Na secção 3.2 foi referido como mapear um ponto tridimensional no referencial do

[4] http://aldream.net/article/2013-04-13-painter-s-algorithm, acedido em 02/03/2015

mundo para o referencial da imagem, com a projeção perspetiva. Contudo, nesse processo perdeu-se a informação de profundidade desse ponto, sabendo apenas que esse ponto se encontrava sobre uma linha no espaço correspondendo a um píxel na imagem. Desta forma, o inverso da projeção perspetiva de um píxel gera um segmento de reta semi-infinito, que interceta as coordenadas do ponto na imagem com origem no centro ótico[6].

O método utilizado para obter as coordenadas 3D de um ponto no referencial da imagem é denominado por triangulação. Esta consiste na visualização de um ponto no mundo a partir de duas imagens com perspetivas diferentes. As imagens com perspetiva diferente podem ser obtidas por diferentes câmaras em posições diferentes ou apenas com a mudança de posição de uma câmara. O cálculo da posição tridimensional de um ponto consiste na interceção de dois segmentos de reta, que resultam da projeção inversa de cada uma das imagens.

Para aplicar a triangulação, é importante que o sistema de visão tenha uma boa calibração. Para além da calibração, é necessário que os pontos a usar na triangulação sejam os mesmos nas imagens com diferentes perspetivas, para que o resultado do método seja o mais exato possível. Assim sendo, para assegurar uma correta correspondência, aplica-se a geometria epipolar como forma de validar os pontos nas diferentes imagens.

3.4.1 Geometria Epipolar

A geometria epipolar permite relacionar geometricamente imagens de um único ponto (P) obtidas de perspetivas diferentes. Pode ser utilizada para representar a circunstância na qual duas câmaras observam a mesmo ambiente, ou uma única a adquire múltiplas imagens de dois pontos de vista diferentes. Na Figura 3.5 é ilustrado o princípio da geometria epipolar, em que o ponto 3D P é projetado na imagem das duas câmaras, sendo identificados por p_l e p_r.

Com base na Figura 3.5, é possível identificar os pontos O_l e O_r como sendo os centros óticos das câmaras da esquerda e da direita, respetivamente. A linha que une os centros óticos das duas câmaras é designada de *baseline*, sendo a interceção entre a *baseline* e o plano de imagem denominado de epipólo. Um epipólo é a projeção do centro ótico de uma câmara no plano da imagem da outra câmara. Assim, o epipólo na imagem da esquerda corresponde a e_l e o epipólo na imagem da direita é e_r. O plano epipolar é o plano que contém a *baseline* e um ponto no mundo, sendo que o cruzamento entre o plano epipolar e o plano das imagens origina as linhas epipolares, em que todas elas intercetam os respetivos epipólos.

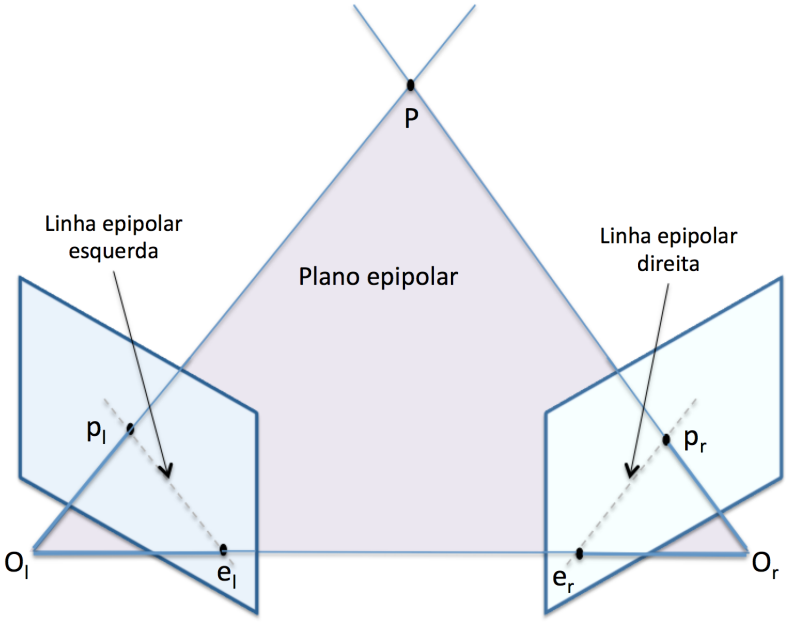


Figura 3.5: Modelo de geometria epipolar.

A propriedade da geometria epipolar que permite validar a correspondência de pontos entre imagens é denominada de restrição epipolar. Esta propriedade define que um ponto na imagem da esquerda (p_l) encontra-se posicionado ao longo de uma linha na imagem da direita.

Com base na propriedade da restrição epipolar é possível validar se um ponto na imagem da esquerda existe na imagem da direita, aplicando a seguinte expressão:

$$p_r^T F p_l = 0 \tag{3.19}$$

Onde F é a matriz fundamental 3x3 e p_l e p_r são os pontos nos planos da imagem da esquerda e da direita, respetivamente. Quanto mais o resultado da equação tender para valores próximos de zero, maior é a probabilidade do ponto da imagem da direita existir na imagem da esquerda, logo este vai estar posicionado sobre a linha epipolar da imagem da esquerda.

A geometria epipolar pode ser representada algebricamente pela matriz Fundamental (F) ou pela matriz essencial [47]. Com a matriz Fundamental é possível calcular os epipólos com as equações 3.20 e 3.21.

$$Fe_l = 0 \tag{3.20}$$

$$F^T e_r = 0 \tag{3.21}$$

Com a matriz Fundamental e os pontos nos planos das imagens em píxeis p_l e p_r, é possível calcular as linhas epipolares com as equações:

$$l_l = F^T p_r \tag{3.22}$$

$$l_r = F p_l \tag{3.23}$$

A matriz Fundamental pode ser obtida de várias formas, por exemplo através do algoritmo de 8 pontos [47] ou através da relação existente entre a matriz Fundamental e a matriz Essencial, sendo que a matriz Fundamental é obtida com a seguinte equação:

$$F = A_r^{-T} E A_l^{-1} \tag{3.24}$$

Onde a matriz E é a matriz Essencial e as variáveis A_l e A_r são os parâmetros intrínsecos das câmaras da esquerda e da direita.

A matriz Essencial também pode ser usada para validar a restrição epipolar, no entanto esta usa os pontos no plano da câmara ao contrário da matriz Fundamental que usa os pontos no plano da imagem[47]. Assim sendo, a restrição epipolar com a matriz Essencial é dada pela seguinte equação:

$$A_r^{-T} p_r^T E A_l^{-1} p_l = 0 \tag{3.25}$$

Onde $A_r^{-T} p_r$ apresenta o ponto no referencial da câmara da direita e $A_r^{-T} p_r$ o ponto no referencial da câmara da esquerda.

Por sua vez, a matriz Essencial pode ser calculada a partir dos parâmetros que relacionam as duas câmaras, ou seja, a matriz de rotação R e o vetor de translação $\mathbf{t}$, sendo dada pela equação 3.26.

$$E = T_\times R \tag{3.26}$$

Sendo que T_x é a matriz *skew-symetric* de $\mathbf{t}$ dada pela equação 3.27.

$$T_x = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{3.27}$$

A matriz de rotação R e o vetor de translação $\mathbf{t}$ relacionam a orientação e a posição do sistema *stereo* em relação ao referencial do mundo, sendo representadas pelas seguintes equações[48]:

$$R = R_r R_l^T \tag{3.28}$$

$$\mathbf{t} = \mathbf{t}_l - R^T \mathbf{t}_r \tag{3.29}$$

Onde as matrizes de rotação R_l e R_r são as matrizes de rotação da câmara da esquerda e da direita. Os vetores $\mathbf{t}_l$ e $\mathbf{t}_r$ dizem respeito à translação das câmaras da esquerda e da direita. Estas matrizes de rotação e vetores de translação são os parâmetros resultantes da calibração dos parâmetros extrínsecos de cada câmara referidos no secção 3.3.2.

3.4.2 Triangulação

Recorrendo à triangulação, é possível determinar um ponto 3D a partir das projeções desse ponto nas imagens de ambas as câmaras. O método de triangulação consiste na interceção de duas retas, com as orientações de p_l e p_r, que são a projeção de um ponto P na câmara da esquerda e da direita. O resultado dessa interceção seria o ponto 3D calculado. No entanto, dado que apenas existem aproximações dos parâmetros da câmara e da localização das imagens, as duas retas não se vão intercetar no espaço [48]. Posteriormente, a interceção das duas retas pode ser aproximada pelo ponto médio da distância entre as duas retas, sendo que esse vai ser o ponto 3D dado por P', como se pode observar na Figura 3.6.

Tendo o conhecimento da matriz de rotação R e do vetor translação $\mathbf{t}$ bem como dos pontos projetados em ambas as câmaras(p_l e p_r) é possível determinar o valor dos escalares a, b e c, a partir da seguinte equação:

$$ap_l - bR^T p_r + c(p_l \times R^T p_r) = \mathbf{t} \tag{3.30}$$

Onde a representa a distância segundo p_l, b representa a distância segundo p_r e c a distância com a direção perpendicular a p_l e p_r [7].

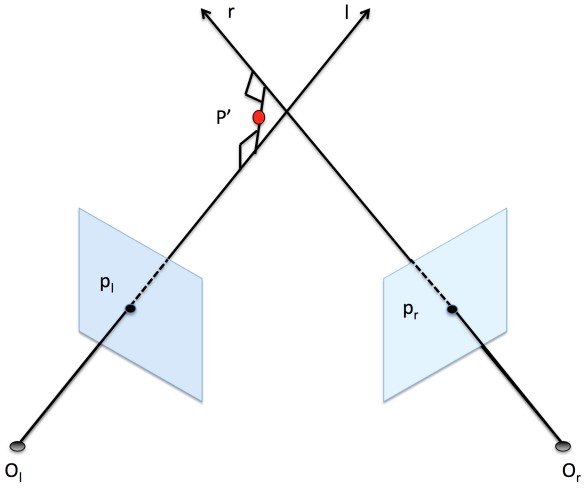


Figura 3.6: Método de triangulação *stereo*.

Através do cálculo dos escalares a, b e c é então possível determinar o ponto P', que representa o ponto médio da interceção das duas câmaras, que é dado por:

$$P' = ap_l + \frac{c}{2}(p_l \times R^T p_r) \tag{3.31}$$

3.4.3 Retificação

Uma outra técnica utilizada para obter a informação tridimensional a partir de um par de pontos de duas imagens com perspetivas diferentes, é a retificação *stereo*[48, 49, 7]. O processo de retificação consiste em determinar a transformação para cada imagem, de modo a simular o alinhamento dos centros focais e dos planos das imagens em ambas as câmaras, tornando os planos de imagem coplanares e perpendiculares ao plano horizontal. Desta forma, os epipólos são projetados para o infinito, fazendo com que as linhas epipolares sejam horizontais e colineares. Assim, é reduzido o problema de procura de correspondência para apenas uma dimensão.

Na Figura 3.7 é possível observar a geometria de retificação do sistema *stereo*.

A transformação a aplicar às imagens originais pode ser considerada como uma aquisição de um par de imagens com uma nova configuração [48]. Estas imagens são obtidas rodando as câmaras originais em torno dos respetivos centros óticos.

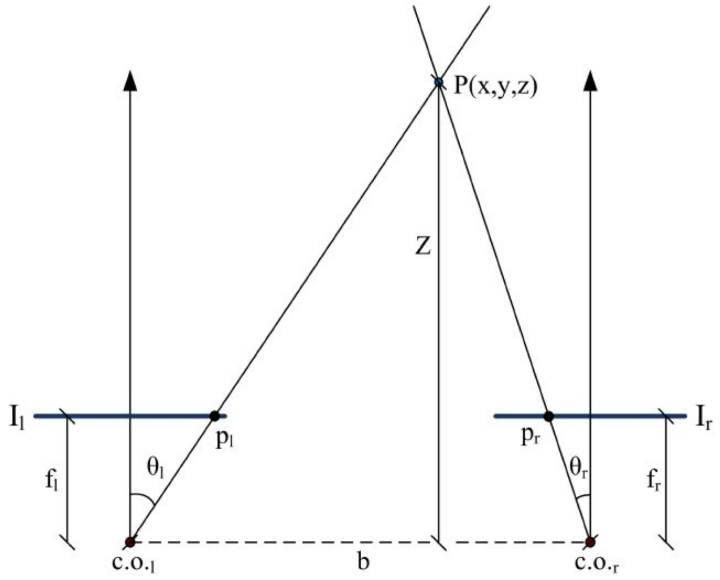


Figura 3.7: Geometria de retificação[7].

Esta metodologia [49] é alcançada calculando três vetores ortogonais $\mathbf{e}_1$, $\mathbf{e}_2$ e $\mathbf{e}_3$, recorrendo às equações 3.32, 3.35 e 3.36.

$$\mathbf{e}_1 = \frac{c_l - c_r}{\|c_l - c_r\|} \tag{3.32}$$

Onde c_l é o centro ótico da câmara da esquerda e c_r é o centro ótico da câmara da direita, em que as respetivas equações são:

$$c_l = -R_l^T t_l \tag{3.33}$$

$$c_r = -R_r^T t_r \tag{3.34}$$

O segundo vetor apresenta como restrição ser obrigatoriamente ortogonal ao vetor $\mathbf{e}_1$, sendo calculado pelo *cross-product* entre $\mathbf{k}$ e $\mathbf{e}_1$, apresentando a seguinte equação:

$$\mathbf{e}_2 = \mathbf{k} \times e_1 \tag{3.35}$$

Onde $\mathbf{k}$ é um vetor arbitrário, que fixa a posição do novo eixo do Y no plano ortogonal

ao eixo do X. Nesta situação, o vetor **k** toma como valores a terceira linha da matriz de rotação da esquerda.

O terceiro vetor é dado pelo *cross-product* de $\mathbf{e}_1$ com $\mathbf{e}_2$, sendo que este deve ser ortogonal ao eixo X e Y, resultando na seguinte equação:

$$\mathbf{e}_3 = \mathbf{e}_1 \times \mathbf{e}_2 \tag{3.36}$$

Com os vetores calculados nas equações 3.32, 3.35 e 3.36 e possível construir a matriz R_{rect} da seguinte forma:

$$R_{rect} = \begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \\ \mathbf{e}_3^T \end{bmatrix} \tag{3.37}$$

É importante notar que, quando o eixo ótico é paralelo à *baseline*, ocorre um movimento puro para a frente, o algoritmo de retificação falha.

Após o cálculo da matriz R_{rect}, são calculadas as novas matrizes dos parâmetros intrínsecos das imagens retificadas, sendo que ambas as matrizes contêm os valores da matriz de parâmetros intrínsecos de uma das câmaras.

Através da nova matriz dos parâmetros intrínsecos A_{New} e da matriz R_{rect}, é possível calcular as novas matrizes de projetivas de cada câmara(PPM_{lnew} e PPM_{rnew}), recorrendo às seguintes equações:

$$PPM_{lnew} = A_{new} \begin{bmatrix} R_{rect} & -R_{rect}c_l \end{bmatrix} \tag{3.38}$$

$$PPM_{rnew} = A_{new} \begin{bmatrix} R_{rect} & -R_{rect}c_r \end{bmatrix} \tag{3.39}$$

Para conseguir retificar os pontos no referencial de uma imagem ou até mesmo retificar a imagem, é necessário calcular a matriz que permite realizar esta transformação. Esta transformação é alcançada através das matrizes H_l e H_r, correspondendo a primeira à transformação em relação à câmara da esquerda e a segunda transformação em relação à câmara da direita. Estas matrizes são apresentadas nas seguintes equações:

$$H_l = PPM_{lnew} \begin{bmatrix} 3 \times 3 \end{bmatrix} PPM_l \begin{bmatrix} 3 \times 3 \end{bmatrix}^{-1} \tag{3.40}$$

$$H_r = PPM_{rnew} \left[3 \times 3\right] PPM_r \left[3 \times 3\right]^{-1} \tag{3.41}$$

Para obter os pontos retificados em ambas as imagens (p'_l e p'_r), é necessário aplicar as equações 3.42 e 3.43.

$$p'_l = H_l p_l \tag{3.42}$$

$$p'_r = H_r p_r \tag{3.43}$$

Após ser realizada a retificação das imagens, estas vão ficar no mesmo plano como é possível observar na Figura 3.8.

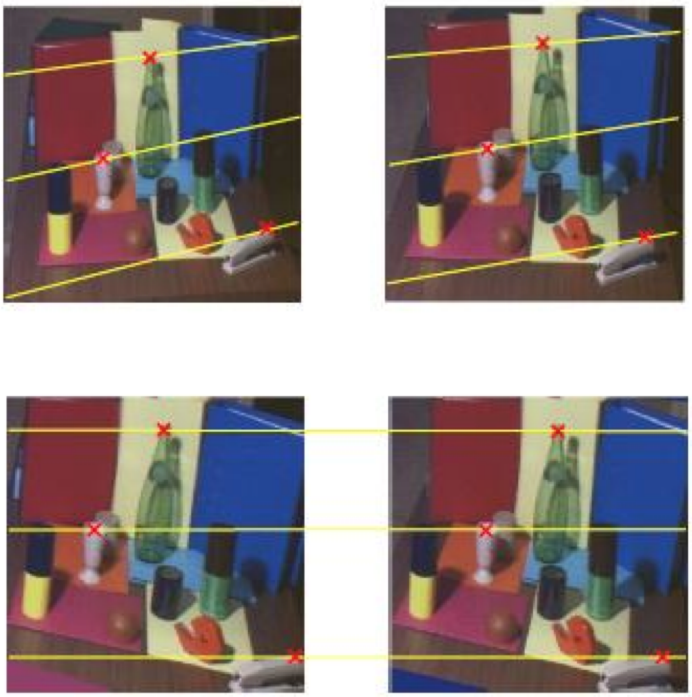


Figura 3.8: Retificação de um par de imagens *stereo* e representação das linhas epipolares, adaptado de[5].

Com as imagens no mesmo plano torna-se simplificada a tarefa de obter a informação 3D de um ponto no mundo, sendo que coordenadas do ponto 3D dadas pelas equações 3.44, 3.45 e 3.46.

$$Z^w = \frac{fb}{x_l^i - x_r^i} \tag{3.44}$$

$$X^w = \frac{x_l^i Z}{f} \tag{3.45}$$

$$Y^w = \frac{y_l^i Z}{f} \tag{3.46}$$

Onde X^w, Y^w e Z^w são as coordenadas do ponto 3D no referêncial da câmara, sendo as coordenadas no plano da imagem representadas por x_l^i e y_l^i na câmara da esquerda e x_r^i e y_r^i na câmara da direita. Sendo b a baseline e f a distância focal.

3.5 *Template matching*

O método de *Template Matching* consiste numa procura de identificação de zonas idênticas em imagens, através da comparação da intensidade dos píxeis nas imagens em questão. Como tal, este método pode ser utilizado em aplicações nos quais seja necessário identificar um dado objeto numa imagem onde esse objeto esteja presente, como é possível observar na Figura 3.9. Uma outra aplicação para este método seria a validação da correspondência de *features* entre imagens.

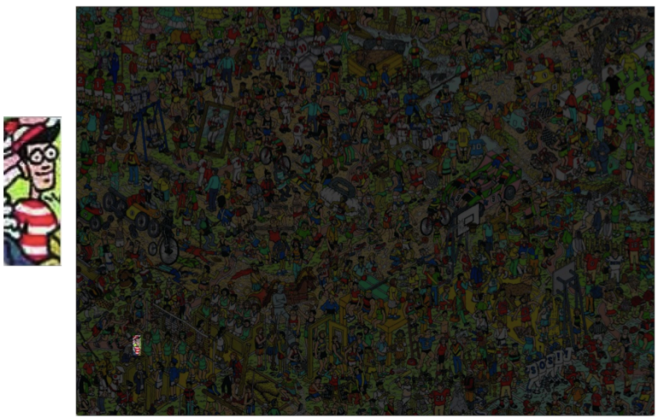


Figura 3.9: Exemplo de aplicação do *template matching*, adaptado de[6].

[5]http://campar.in.tum.de/twiki/pub/Chair/TeachingWs10Cv2/3D_CV2_WS_2010_Rectification_Disparity.pdf, acedido em 03/03/2015

Para iniciar o processo de *template matching* para a correspondência de *features* entre imagens, primariamente é necessário definir o *template* que se pretende procurar na imagem. Este *template* deve ser uma janela quadrada centrada em torno da posição da *feature* da imagem de referência.

O passo seguinte consiste em deslocar o *template* um píxel de cada vez sobre a imagem alvo, da esquerda para a direita e de cima para baixo, sendo calculado em cada posição, um valor que indicará homogeneidade entre os conjuntos de píxeis sobrepostos.

Para obter este valor resultante da sobreposição dos conjuntos de píxeis, é aplicado um método de correlação linear como o *Sum of Squared Differences*(SSD) [50]:

$$SSD(x,y) = \sum_{i,j \in W} (I(i,j) - T(x+i, y+j))^2 \tag{3.47}$$

Onde T corresponde ao *template* com uma dimensão W centrado no píxel na posição (x, y) da imagem I. As variáveis i e j representam os índices de localização de um elemento do *template*.

O SSD calcula o quadrado da distância euclidiana entre o *template* e a parte da imagem sobreposta pelo *template*. Quanto mais próximo de zero for o resultado do SSD, maior é a medida de semelhança entre o *template* e o zona sobreposta pelo *template*.

Um outro método utilizado para *template matching* é o *Cross-Correlation*(CC). Este método é motivado pelo quadrado da distância euclidiana dado pela equação:

$$d^2(x,y) = \sum_{i,j} ((I(i,j) - T(i-x, j-y))^2 \tag{3.48}$$

Através da expansão de d^2 é possível observar que o termo $\sum T^2(x-i, y-j)$ é constante. Como tal, se o termo $\sum T^2 x, y$ também for aproximadamente constante, então o resultado da *Cross-Correlation* seria dado pela equação 3.49[51].

$$CC(x,y) = \sum_{i,j \in W} (I(i,j) - T(i-x, j-y)) \tag{3.49}$$

No entanto o uso deste método pode falhar se a energia da imagem $\sum I^2(i,j)$ variar com a sua posição. De forma adversa, o CC não é invariante às mudanças de intensidade na imagem causadas pela variação das condições de iluminação no cenário no qual se ob-

[6]http://machinelearningmastery.com/using-opencv-python-and-template-matching-to-play-wheres-waldo/, acedido em 03/03/2015

teve a imagem. O método que permite colmatar estes problemas é dado pela *Normalized Cross-Correlation coefficient*(γ) [51], apresentado na equação 3.50.

$$\gamma(x,y) = \frac{\sum_{i,j}\left(I(i,j) - \overline{I}_i, j\right)(T(i-x, j-y) - \overline{T})}{\sqrt{\sum_{i,j}\left(I(i,j) - \overline{I}_i, j\right)^2 \sum_{i,j}\left(T(i-x, j-y) - \overline{T}\right)^2}} \tag{3.50}$$

Onde $\overline{T}$ é o valor médio de T e $\overline{I}_i, j$ é o valor médio de $I(x,y)$ quando este se encontra posicionado sobre o *template* T. O valor encontra-se -1 e 1, sendo que valores próximos de 1 indicam uma boa correspondência e os valores próximos de -1 indicam uma má correspondência.

Apesar deste método ter um custo computacional superior, existem algoritmos que permitem acelerar este processo[51, 52], eliminando assim a principal desvantagem deste método.

Esta página foi intencionalmente deixada em branco.

Capítulo 4

Abordagem Conceptual

Anteriormente, foi caracterizado a problemática adjacente à correspondência de pontos em sistemas de visão multi-câmara. Neste capítulo é realizada uma descrição generalizada da presente dissertação, sendo efetuada uma descrição da plataforma de aplicação e descrita a arquitetura do software a implementar.

4.1 Plataforma de aplicação

Os robôs futebolistas da equipa do ISePorto são robôs autónomos que possuem um sistema perceção, de forma a conseguir desempenhar as suas funções num ambiente muito dinâmico. O bloco de perceção destes robôs é composto por um sistema de visão, utilizado paras as tarefas de navegação e deteção de objetos. A tarefa de navegação ou localização consiste na deteção das linhas do campo, sendo que na deteção de objetos em campo, prende-se distinguir diferentes objetos existentes no jogo, tais como: robôs adversários ou aliados, a bola de futebol ou as balizas do campo.

O sistema de visão dos robôs futebolistas é composto por duas câmaras: uma câmara no topo do robô denominada *head* e uma câmara no torso do robô denominada *kicker*. O robô com a posição de guarda-redes tem uma câmara adicional na base do*kicker*, sendo esta a câmara do *keeper*. Atualmente as câmaras implementadas têm características diferentes para além das lentes, sendo que a câmara da *head* tem uma resolução de 1294x964 e a câmara do *kicker* tem uma resolução de 640x480. Na Figura 4.1 é possível visualizar a posição de cada câmara no robô.

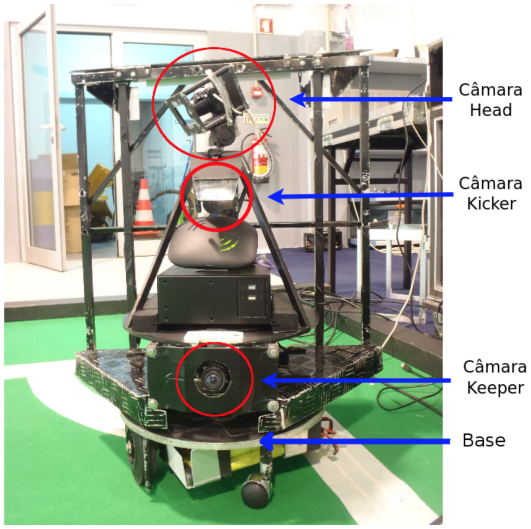


Figura 4.1: Imagem do guarda-redes do ISePorto, com posição das câmaras assinalada [46]

As câmaras do robô encontram-se posicionadas numa plataforma móvel, em que o seu movimento consiste numa rotação em torno do eixo do Z. Para além dos robôs conseguirem rodar sobre eles mesmos, estes conseguem rodar as câmaras permitindo a cada câmara desempenhar uma função específica. Através da rotação da plataforma da câmara da *head* torna possível o robô visualizar todo o campo, não sendo necessário rodar sobre si mesmo. A rotação da plataforma do *kicker* permite ao robô um controlo da bola mais diversificado, fazendo com que a câmara do *kicker* detete sempre a posição da bola. Na Figura 4.2 é ilustrado os eixos de rotação do robô.

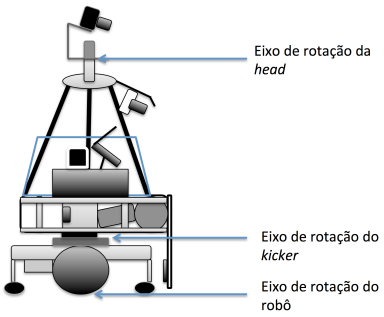


Figura 4.2: Eixos de rotação do robô do ISePorto.

A deteção de objetos relevantes para o jogo é realizado principalmente através da análise de cores em imagens obtidas por um processo de segmentação[53]. Isto porque, as linhas do campo, a bola e robôs de ambas as equipas têm cores distintas umas das outras, o que possibilita designar um objeto com base na sua cor. Um exemplo é a deteção da bola, uma vez que a bola pode ter a cor amarela ou laranja, ao analisar a imagem segmentada é apenas necessário procurar uma zona circular amarela ou laranja, como é possível observar na Figura 4.3.

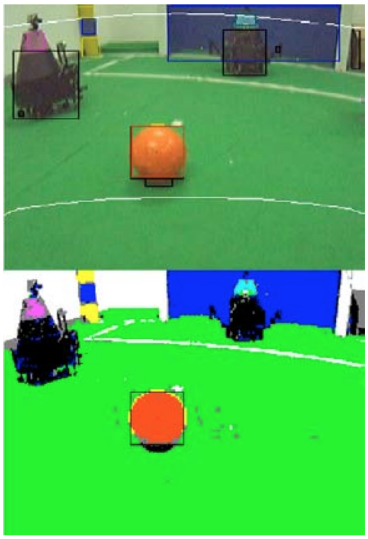


Figura 4.3: Sistema de deteção baseado na segmentação de imagem, adaptado de [53].

Tendo em conta as características dos robôs futebolistas da equipa do ISePorto, torna a aplicação de um método de correspondência de *features* para obtenção de informação tridimensional um desafio interessante, uma vez que seria possível tirar partido da abordagem de segmentação da imagem, dado que, é uma método computacionalmente leve. Para além da abordagem de segmentação, as câmaras dos robôs têm resoluções diferentes, o que aumenta a dificuldade do processo de correspondência.

O uso da plataforma do ISePorto para validar a abordagem a desenvolver é vantajoso, uma vez que permite validar o método em ambientes *indoor*, sendo possível controlar as condições do ambiente de teste.

4.2 Arquitetura do software

Durante esta secção é apresentada a arquitetura do software desenvolvida para o método de correspondência de *features* entre imagens. Desta forma, foi desenvolvido um método dividido em múltiplas fases, abordando tópicos como a deteção de *features*, *template matching*, *matchers* de *features* e retificação de imagem.

Esta arquitetura foi desenvolvida tendo em consideração o problemática de correspondência de *features*, mas também levando em conta os requisitos para aplicação do método em tempo real.

No diagrama apresentado na Figura 4.4 é ilustrada a arquitetura do método implementado na presente dissertação.

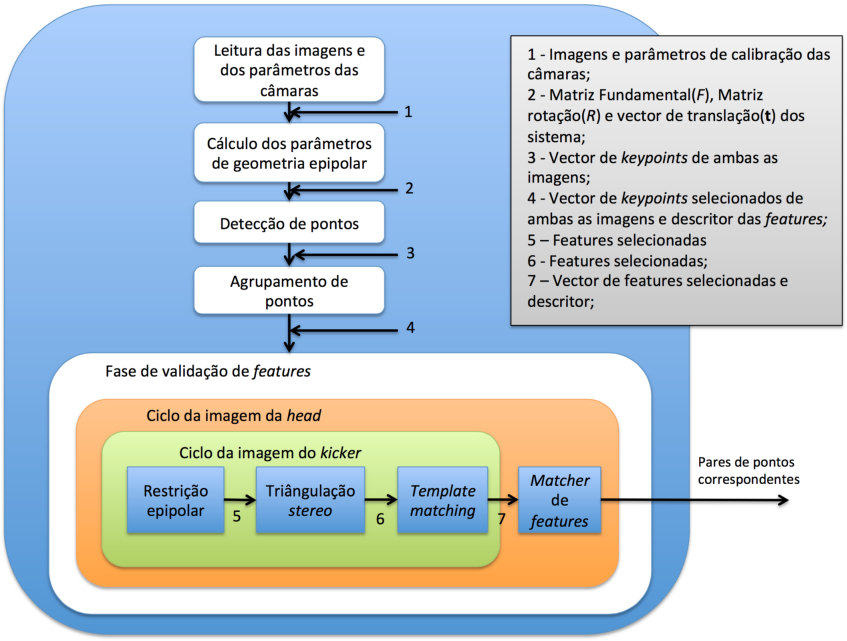


Figura 4.4: Diagrama do processo de correspondência de *features*.

Na primeira fase do processo de correspondência é realizada a leitura das imagens provenientes das câmaras da head e do kicker e os respetivos parâmetros de calibração.

Na fase posterior são calculados os parâmetros necessários para aplicação da geometria epipolar, nomeadamente a aplicação da restrição epipolar.

A deteção de pontos consiste na aplicação de um detetor de *features* às imagens da fase de leitura. Previamente à aplicação do detetor, é realizado o redimensionamento da imagem com maior resolução, de forma a que as imagens de ambas as câmaras têm um campo de visão semelhante. Desta forma, a probabilidade do detetor extrair *features* comuns às duas imagens é maior, contribuindo para um aumento de possíveis correspondências.

Na quarta fase é aplicado um algoritmo de *buckting*, este consiste em agrupar o grupo de *features* que se encontram muito próximas umas das outras, filtrando essas *features* resultando num vetor de *features* com apenas uma *feature* por zonas de *features* agrupadas.

A fase de validação de *features* é dividida em 4 etapas que decorrem em ciclos que percorrem os vetores de pontos de cada imagem, sendo que nesta fase é estabelecida a correspondência de *features* entre imagens.

A primeira etapa consiste na aplicação da restrição epipolar para validar os *features* que se encontram na zona da linha epipolar.

Na etapa seguinte são calculados os escalares a, b e c de triangulação com o intuito de validar quais as *features* que são possíveis de estar nas zonas comuns a ambas as imagens.

Após a aplicação da triangulação *stereo*, é aplicado um método de *template matching* para validar quais são os pares de *features* correspondentes, sendo extraído um *template* da imagem da *head* e uma janela de pesquisa na imagem do *kicker*. Tanto o *template* como a janela de pesquisa são janelas de tamanhos diferentes obtidas em torno das respetivas *features* das imagens, aplicando uma região de interesse. Com os resultados do *template matching*, é aplicado um algoritmo para avaliar quais são as possíveis correspondências são corretas.

As três etapas iniciais da fase de validação de *features* decorrem no ciclo da imagem do *kicker*.

A última etapa apenas decorre no caso da etapa de *template matching* não obter uma única correspondência para os pares de *features*. Nesta fase é aplicado um *matcher* de *features* que compara as *features* das duas imagens com base na informação fornecida pelo descritor. Desta forma é possível conseguir destacar o par de *features* com maior probabilidade de ser uma correspondência, dos vários pares obtidos na etapa de *template matching*.

Durante o procedimento de comparação dos pares de detetores e descritores foram selecionados o FAST/BRIEF e o ORB. No entanto para aplicação no método desen-

volvido foi selecionado o detetor ORB, uma vez que cumpria os requisitos da fase de comparação, mas também era um método invariante à escala e a rotações.

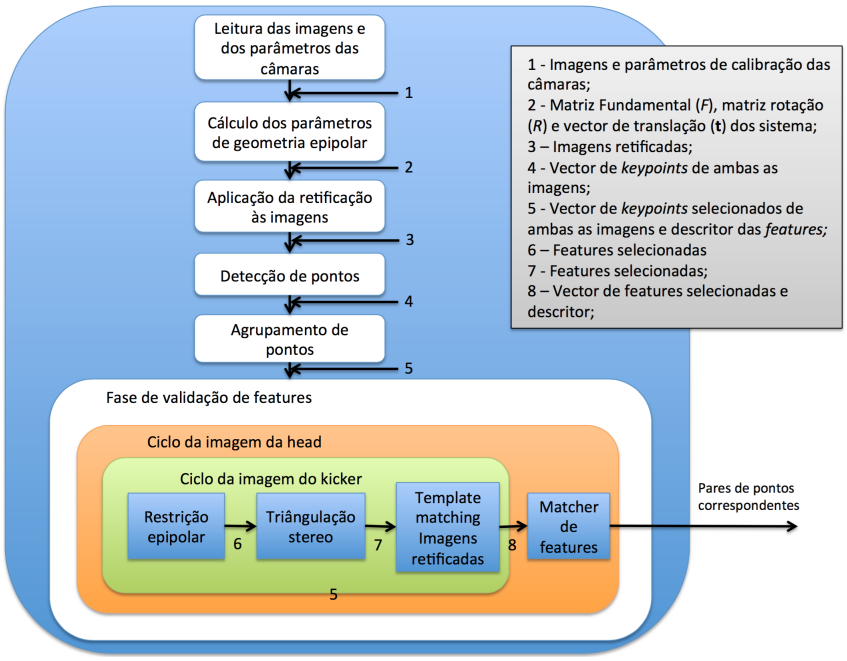


Figura 4.5: Diagrama do processo de correspondência de *features* com retificação de imagem.

Uma variante da arquitetura anterior foi proposta e implementada nesta dissertação, sendo que nesta nova arquitetura é aplicada a retificação de imagens. Esta aplicação surge com sentido de verificar se imagens com a mesma escala iriam sortir um efeito melhorado na etapa de aplicação do *template matching*.

Como se pode observar na Figura 4.5, a implementação da retificação foi efetuada em dois blocos. O primeiro consiste na introdução do bloco de aplicação da retificação às imagens da fase de leitura, sendo calculados os parâmetros de retificação para de seguida aplicar a retificação às imagens da *head* e do *kicker*.

A segunda alteração introduzida foi na etapa de *template matching*, no qual foram usadas as imagens retificadas para extrair o *template* e a janela de pesquisa. Desta forma é necessário aplicar a retificação às *features* obtidas, permitindo extrair o *template* e a janela de pesquisa de forma correta nas imagens retificadas.

Capítulo 5

Implementação

Após ter sido formulado o problema relativo à correspondência de *features* entre imagens e apresentado a arquitetura do novo processo de correspondência de *features*, procedeu-se então à implementação.

5.1 Processo de correspondência de *features* com imagens originais

Nesta secção são detalhadas todas as etapas do processo de correspondência de pontos, apresentadas na Figura4.4. Como tal, pretende-se salientar os pontos que contribuíram para o bom desempenho deste método. Este processo foi implementado em C++, recorrendo à biblioteca de visão computacional OpenCV.

5.1.1 Leitura das imagens e parâmetros das câmaras

Nesta fase é realizada a leitura de imagens com ambas as câmaras do robô(*head* e *kicker*), no qual estas imagens contêm elementos presentes no ambiente de operação dos, tais como outros robôs ou bolas, como se pode observar na Figura 5.1.

Na leitura das imagens são lidas duas imagens, uma imagem da *head* e outra do *kicker*, em que ambas as imagens encontram-se num formato(*png, jpg, ...*) que permite ser lido pela função *imread* do OpenCV.

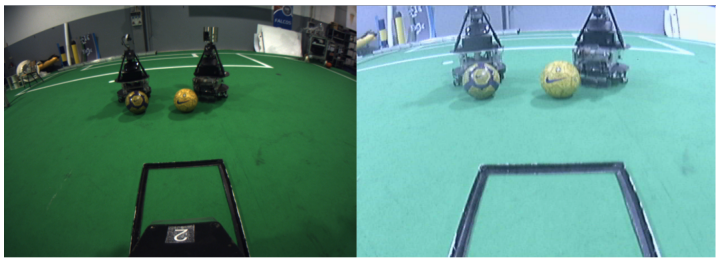


Figura 5.1: Imagens adquiridas pelas câmaras da *head* e do *kicker*.

Associado a cada imagem encontra-se um ficheiro gerado pelo processo de calibração, sendo que esse ficheiro contém os parâmetros intrínsecos e extrínsecos. Os parâmetros lidos do ficheiro são os seguintes:

- **fc** - Vetor com a distância focal no eixo do x e no eixo do y;
- **cc** - Vetor com as coordenadas do centro ótico no eixo do x e no eixo do y;
- **kc** - Vetor com os coeficientes de distorção;
- *Rckk1* - Matriz com os dados de orientação da câmara em relação ao referencial considerado;
- **Tckk1** - Vetor com a posição relativa da câmara em relação ao referencial considerado.

5.1.2 Cálculo dos parâmetros da geometria epipolar

Concluído o processo de leitura das imagens e dos parâmetros das câmaras, prosseguiu-se para o cálculo das variáveis utilizadas de modo a aplicar a restrição epipolar às duas imagens. Nesta etapa são necessários os parâmetros intrínsecos e extrínsecos de cada câmara. O objetivo desta etapa é obter a matriz fundamental F, possibilitando numa fase posterior a aplicação da restrição epipolar. As funções utilizadas nesta etapa para realizar os cálculos enunciados na secção , foram desenvolvidas em Maple[8].

Inicialmente foi calculado a posição dos centros óticos de cada câmara no referencial da câmara(c_{head} **e** c_{kicker}), sendo estes parâmetros obtidos pelas equações 3.33 e 3.34, nas respetivas câmaras da *head* e do *kicker*.

[8]http://www.maplesoft.com/products/Maple/, acedido em 01/03/2015

Desta forma as matrizes de rotação a usar no cálculo dos centros óticos, são dadas por: $R_l = R_{head}$ e $R_r = R_{kicker}$. No qual, os vetores de translação são apresentados por: $t_l = t_{head}$ e $t_r = t_{kicker}$.

A matriz de rotação do sistema de câmaras (R_{stereo}) tem como origem o referencial da câmara da *head*, sendo dada pela equação 3.28. Através da posição dos centros óticos é possível calcular o vetor de translação do sistema de câmaras $\mathbf{t}_{stereo}$ com o mesmo referencial de R_{stereo}, sendo este dado pela equação 5.1.

$$\mathbf{t}_{stereo} = R_{head}(c_{kicker} - c_{head}) \tag{5.1}$$

Com o vetor de translação t_{stereo} é possível obter a matriz *skew-symetric* de t_{stereo}, sendo esta apresentada na equação 3.27. Através da matriz *skew-symetric* $T_\times$ e a matriz de rotação R_{stereo} é possível obter a matriz essencial E, que permite relacionar dois pontos de câmaras distintas no referencial da câmara, sendo calculada recorrendo à equação 3.26.

Através do cálculo da matriz essencial E é possível obter a matriz fundamental F uma vez que também são conhecidas as matrizes dos parâmetros intrínsecos da câmara da $head(A_{head})$ e os parâmetros intrínsecos da câmara do $kicker(A_{kicker})$. A matriz fundamental F é dada pela equação 5.2.

$$F = A_{head}^{-T} E A_{kicker}^{-1} \tag{5.2}$$

Com a matriz fundamental torna-se possível relacionar um ponto na imagem da câmara da *head* com um ponto na imagem da câmara do *kicker*, sendo aplicada numa etapa posterior.

5.1.3 Deteção de *features*

Nesta fase é implementado o detetor de *features* nas imagens lidas na fase de leitura, com o objetivo de extrair *features* em objetos relevantes para o cenário de aplicação dos robôs futebolistas, nomeadamente outros robôs ou bolas.

Previamente é necessário realizar uma alteração à imagem proveniente da câmara da *head*. Esta alteração é necessária devido à resolução das imagens serem diferentes, sendo a resolução da imagem da *head* maior. Uma vez que a imagem da *head* tem

uma resolução maior, faz com existam zonas visualizadas na imagem da *head* que não são comuns à imagem do *kicker*, como se pode observar pelos quadrados a vermelho na Figura 5.2. Desta forma, algumas das *features* detetadas numa imagem podem não ter correspondência na outra imagem, pelo facto de não haver a possibilidade dessa *feature* existir na outra imagem.

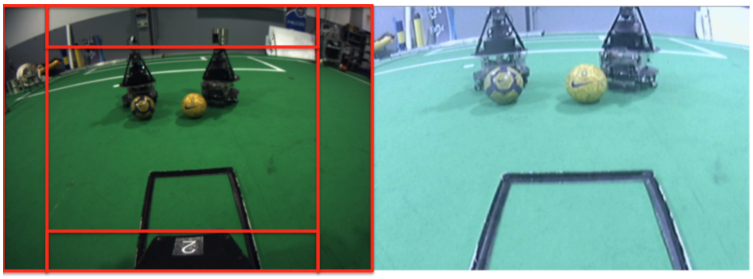


Figura 5.2: Imagem da câmara da *head*(a) com marcação a vermelho de zonas que não existem na imagem da câmara do *kicker* (b).

Devido a este motivo, o número de correspondências possíveis entre os pontos obtidos pelo detetor diminui. De forma a ultrapassar esta situação, foi extraída uma região de interesse da imagem da câmara da *head*, de forma a que esta região contemple uma área aproximada à visualizada na imagem do *kicker*, como se pode observar na Figura 5.3. Assim, a probabilidade de detetor extrair *features* em zonas comuns às duas imagens é maior, podendo gerar uma mais possíveis correspondências.

Assim, a distribuição das *features* obtidas pelo detetor será aproximada em ambas as imagens, aumentando assim a existência de possíveis correspondências.

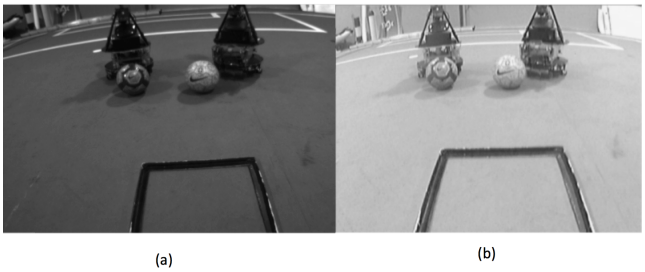


Figura 5.3: (a)Imagem reduzida da câmara da *head* (b)Imagem normal da câmara do *kicker*.

Após a redução do tamanho da imagem da câmara da *head*, é possível prosseguir para a aplicação do detetor de *features*. Nesta fase do processo é criado e inicializado o detetor e descritor de *features* ORB e o *matcher* de *features Brute-force matcher*, que foram usados ao longo da implementação, em fases diferentes. O ORB tem os seguintes parâmetros de configuração:

- *nfeatures* - Número de *features*;
- *scaleFactor* - Fator de escala da pirâmide;
- *nlevels* - Número de níveis da pirâmide;
- *edgeThreshold* - Tamanho da fronteira onde as *features* não são detectadas;
- *firstLevel* - Definição do primeiro nível da pirâmide no qual o detector é aplicado;
- *WTA_K* - Número de pontos produzidos por *feature* pelo descritor, para análise da *feature*;
- *scoreType* - Critério a usar para categorizar as *features* obtidas;
- *patchSize* - Tamanho da janela de aplicação do descritor.

Na inicialização do ORB apenas se alterou alguns parâmetros relacionados com o detetor, mantendo todos os outros com os valores pré-definidos.
O número de *features* foi um dos parâmetros alterado, ao qual o valor inicial era de 500, sendo que este foi reduzido para 200 de forma a reduzir o tempo de processamento em fases posteriores que dependem do número de *features*.

A escolha do número de níveis da pirâmide foi associada ao tempo de execução do detetor, em que quanto mais níveis existirem na pirâmide, maior será o tempo de execução do detetor. No entanto, o tempo de execução do detetor não foi único motivo para a alteração dos níveis da pirâmide. A dispersão das *features* na imagem está relacionada com o número níveis na pirâmide e com o critério de categorização das *features*(*scoreType*). O critério pré-definido é o *HARRIS_SCORE*, que aplica o algoritmo de HARRIS para categorizar e selecionar as melhores *features* em cada nível da pirâmide sem ultrapassar o número de *features* definido inicialmente. Desta forma, é possível controlar o número de *features* extraídas.

Tendo em conta que é possível uma *feature* ser detetada muito próxima de outra *feature* na mesma zona da imagem, mas em diferentes níveis da pirâmide, leva a que muitas das *features* extraídas pelo detetor estejam todas no mesmo local. Como consequência, a dispersão das *features* vai ser mais baixa, como se pode observar na Figura 5.4.

Figura 5.4: Imagem da câmara da *head* com *features* extraídas pelo ORB com uma pirâmide de 8 níveis.

Como solução para o problema de dispersão e do tempo de processamento, configurou-se o número de níveis da pirâmide para 3 níveis, sendo que o seu valor pré-definido era de 8 níveis. Na Figura 5.5 é possível observar a imagem da câmara da *head*, após ter sido executado o detetor de *features ORB* com número de níveis da pirâmide igual a 3, sendo marcadas algumas zonas no qual se observou um aparecimento de novas *features*.

Em relação ao *matcher* escolhido, este compara um vetor de descritores de ambas as imagens. Uma vez que o descritor de *features* é binário, o *matcher* deve ser capaz de processar a informação binária do descritor para poder comparar diferentes *features*. Desta forma, o *Brute-force matcher* é configurado com a variante NORM_HAMMING. O método NORM_HAMMING é uma estratégia de correspondência que faz uso da distância de Hamming para comparar *strings* binárias. No contexto da correspondência de pontos, o método NORM_HAMMING vai retornar a distância entre dois descritores correspondentes a *features* de imagens diferentes, sendo que quanto menor for o valor, maior é a probabilidade de existir correspondência entre as *features*.

Figura 5.5: Imagem da câmara da *head* com *features* extraídas pelo ORB com uma pirâmide de 3 níveis.

5.1.4 Agrupamento de *features*

Apesar do detetor ORB gerar *features* mais dispersadas umas das outras com redução do número de níveis da pirâmide, ainda assim, continuam a formar grupos de *features* muitos próximas umas das outras, como se pode observar na Figura 5.6. Essas *features* agrupadas não vão permitir uma grande caracterização dos objetos, sendo que não é relevante processar todas as *features* agrupadas, pois a contribuição de um grupo de *features* na mesma zona é igual à contribuição de uma única *feature* nessa mesma zona.

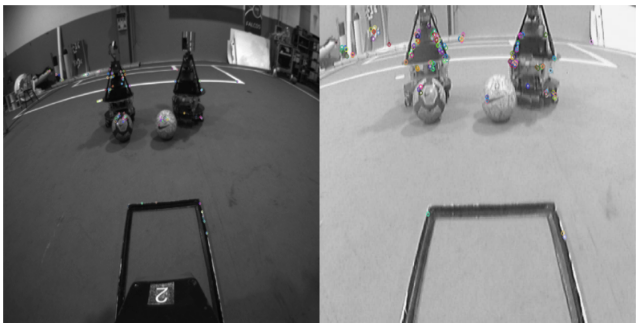


Figura 5.6: Imagens da câmara da *head* e do *kicker*, após execução do detetor de *features*.

Nesta etapa é aplicado um algoritmo que permite reduzir o número *features* muito

próximas umas das outras, substituindo um aglomerado de *features* muito próximas umas das outras por apenas uma *feature* naquela zona.
A função que implementa o algoritmo desenvolvido recebe como parâmetros de entrada um vetor de *keypoints* proveniente do detetor de *features* e um parâmetro para regular o tamanho dos grupos de *features*, retornando um vetor de *keypoints* filtrado pelo algoritmo. Na Figura 5.7 está representado o diagrama do algoritmo desenvolvido.

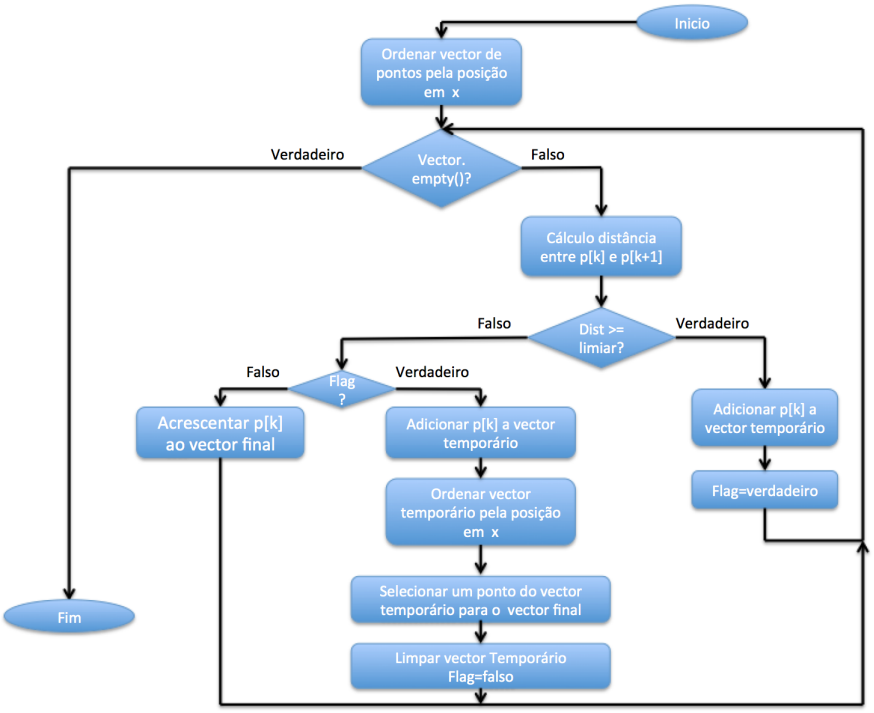


Figura 5.7: Diagrama do algoritmo de agrupamento de *features*.

O primeiro passo a realizar no algoritmo consiste em ordenar o vetor de *keypoints* por ordem crescente segundo a posição da *feature* no eixo do x.
Tendo o vetor ordenado, realiza-se um ciclo no qual é calculada a distância Euclidiana entre o *keypoint* atual e *keypoint* seguinte do vetor recorrendo à equação 5.3.

$$dist = \sqrt{(keyp_x[k+1] - keyp_x[k])^2 + (keyp_y[k+1] - keyp_y[k])^2} \tag{5.3}$$

Onde $keyp_x[k]$ é a posição atual do vetor de *keypoints* no eixo do x, $keyp_x[k+1]$ é a posição seguinte do vetor de *keypoints* no eixo do x, $keyp_y[k]$ é a posição atual do vetor de *keypoints* no eixo do y e $keyp_y[k+1]$ é a posição seguinte do vetor de *keypoints* no

eixo do y. A variável k é o índice do vetor.

No caso do valor da distância entre *keypoint* atual e o seguinte for menor que o valor do limiar definido, então adiciona-se o *keypoint* a vetor temporário, significando que o *keypoint* atual pode ser agrupado com o *keypoint* seguinte. Por fim é acionada uma *flag* com o valor '1' e retorna-se à parte do cálculo da distância Euclidiana.
Na eventualidade de acontecer a situação oposta, ou seja, o valor da distância euclidiana apresentar um valor superior ao limiar definido, então é testado o valor da *flag*, em que na situação do valor da *flag* for '0' , então o *keypoint* é adicionado ao vetor de pontos final.
No caso do valor da *flag* ser '1', adiciona-se o *keypoint* ao vetor temporário significando que este é o último *keypoint* do grupo.

De seguida é ordenado o vetor temporário por ordem crescente segundo a posição da *feature* no eixo do x. Com o vetor ordenado, prossegue-se à seleção do *keypoint* que representa o grupo de *keypoints*, em que a escolha do *keypoint* é realizado com base na equação 5.4.

$$P_{sel} = ceil(\frac{size_{vector_{temp}}}{2}) \tag{5.4}$$

Onde $size_{vector_{temp}}$ é o tamanho do vetor temporário e P_{sel} é a índice do ponto a selecionar do vetor temporário, que será guardado no vetor final.

Por fim é eliminado o conteúdo do vetor temporário e mudado o valor da *flag* para '0', repetindo novamente o cálculo da distância euclidiana.

Quando o ciclo usado para percorrer o vetor de *keypoints* de entrada chegar ao fim, a função é terminada, sendo retornado no fim um vetor com os *keypoints* selecionados pelo algoritmo da função.

A função que implementa o algoritmo desta fase foi aplicada aos vetores de *keypoints* de ambas a imagens obtidos na fase anterior.

5.1.5 Fase de validação de *features*

Nesta fase do método implementado, foi realizado um processo para validar quais as possíveis correspondências entre as *features* da imagem da *head* e as da imagem do *kicker*. Este processo foi divido em várias etapas de validação incorporadas em dois ciclos que percorrem o vetor de *keypoints* de ambas as imagens, de forma a conseguir

obter uma correspondência de pontos fiável entre imagens. Na Figura 5.8 é representado um diagrama das etapas de validação implementadas nesta fase.

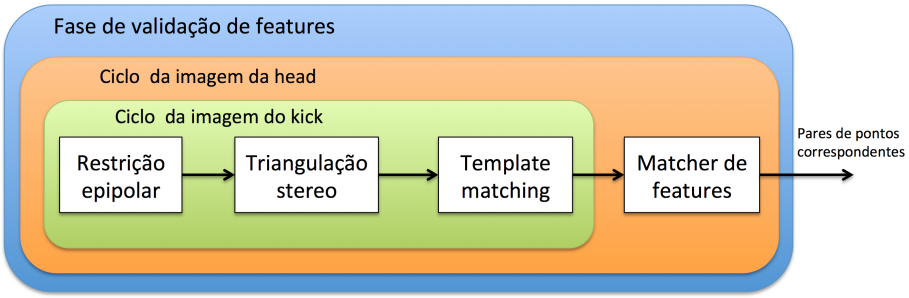


Figura 5.8: Processo de validação de *features*.

Previamente à execução da fase de validação de *features*, é necessário remover a distorção à posição da *feature* na imagem, de modo a permitir a aplicação correta da geometria epipolar. A função usada para remover a distorção aos pontos recebe como parâmetros de entrada, o vetor de *keypoints* resultante da fase anterior, sendo que o retorno da função é um vetor de pontos sem distorção.
A função para remover a distorção dos pontos das imagens é aplicada aos vetores de *keypoints* de ambas as imagens.

Restrição epipolar

Esta etapa de validação utiliza a geometria epipolar para eliminar possíveis más correspondências entre as *features* de ambas as imagens. A utilização da geometria epipolar nesta etapa consiste na aplicação da restrição epipolar, que permite determinar se um ponto da imagem da *head* encontra-se posicionado ao longo da linha epipolar da imagem do *kicker*. O cálculo da restrição epipolar de um ponto na imagem da câmara da *head* com um ponto na imagem da câmara do *kicker* é dado pela equação 5.5.

$$p_{head}^{T} F p_{kicker} = 0 \tag{5.5}$$

Onde p_{head} é um ponto na imagem sem distorção da câmara da *head*, F é a matriz fundamental e p_{kicker} é um ponto na imagem sem distorção da câmara do *kicker*.

Na realidade o valor da restrição epipolar nunca vai ser exatamente zero, pois o

pontos nunca se encontram posicionados precisamente sobre a linha epipolar. Deste modo o valor da restrição epipolar é um valor aproximado de zero, que tanto pode ser positivo como negativo. Tendo em conta o comportamento da restrição epipolar, foi definido um limite para aceitar pontos na imagem do *kicker* próximos da linha epipolar, como se pode observar na Figura 5.9.

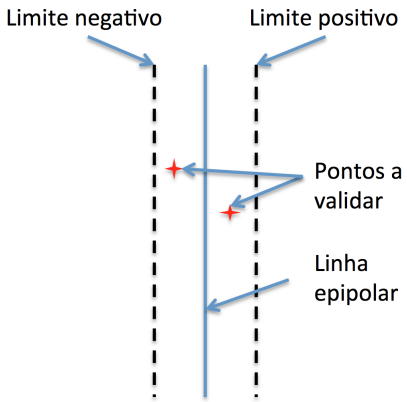


Figura 5.9: Aplicação da restrição epipolar com limites.

Ao longo da implementação, o intervalo usado para limitar a restrição epipolar foi de -0.8 a 0.8, em que um valor maior significa aceitar pontos que dificilmente podem ter correspondência. Os pontos que foram aceites pela restrição epipolar prosseguem à próxima etapa de validação.

Parâmetros de triangulação stereo

Apesar do restrição epipolar permitir reduzir o problema de correspondência de pontos entre imagens para uma dimensão, esta apenas garante que o ponto da imagem da *head* encontra-se sobre a linha epipolar da imagem do *kicker*. Desta forma, não é possível concluir que as *features* de ambas as imagens são correspondentes, dado que, podem existir múltiplas *features* do *kicker* posicionadas sobre a linha epipolar, sendo que uma das *features* pode ser correspondente ou então nenhuma é correspondente.

Noutra situação, um ponto da imagem da *head* pode encontrar-se sobre a linha epipolar da imagem do *kicker*, no entanto, não implica que o verdadeiro ponto correspondente pertença à imagem do *kicker*, significando que ainda existem zonas não são comuns a

ambas as imagens. Apesar da extração de *features* ter sido aplicada à imagem da *head* reduzida para que o campo de visão da imagem da *head* se aproximasse da imagem do *kicker*, continuam a existir zonas que não são comuns a ambas as câmaras.

A solução para esta situação implica o cálculo dos escalares(a, b e c) usados para aplicar o método de triangulação. Como foi explicado na secção 3.4.2, o parâmetro a é a distância segundo p_l, b representa a distância segundo p_r e c a distância com a direção perpendicular a p_l e p_r. Para obter os parâmetros de triangulação, é necessário converter os pontos da imagem da *head* e do *kicker* em causa, para o referencial da câmara.
A conversão dos pontos do referencial da imagem para o referencial da câmara é alcançado através das equações 5.6 e 5.7.

$$P_{head} = A_{head}^{-1} p_{head} \tag{5.6}$$

$$P_{kicker} = A_{kicker}^{-1} p_{kicker} \tag{5.7}$$

Onde P_{head} é o ponto no referencial da câmara da *head* e P_{kicker} é o ponto no referencial da câmara da *kicker*.

Através dos pontos no referencial da câmara(P_{head} e P_{kicker}), consegue-se obter os escalares(a, b e c) resolvendo a equação 3.30 em ordem a cada um dos escalares.

Com os valores das distâncias a e b, é possível determinar a que distância segundo cada câmara, os pontos a validar vão se cruzar no espaço 2D. Dependendo do valor de a e de b é possível determinar se um ponto é comum as duas imagens ou não. Na Figura 5.10 é possível observar dois tipos de situação que se pode validar com os parâmetros de triangulação.

A primeira validação consiste em verificar se os valores de a e b são superiores a '0'. No caso de os valores serem negativos, ou apenas um desses valores ser negativo, significa que os pontos intercetam-se atrás da câmara, como é ilustrado na Figura 5.10(a).

Tendo passado o primeiro critério, prossegue-se à segunda validação. Esta consiste em verificar de os valores de a e b são superiores a um dado limite, garantindo desta forma que não existe uma interceção muita curta num espaço onde não existiria nenhum objeto. Esta situação é ilustrada na Figura 5.10(b).

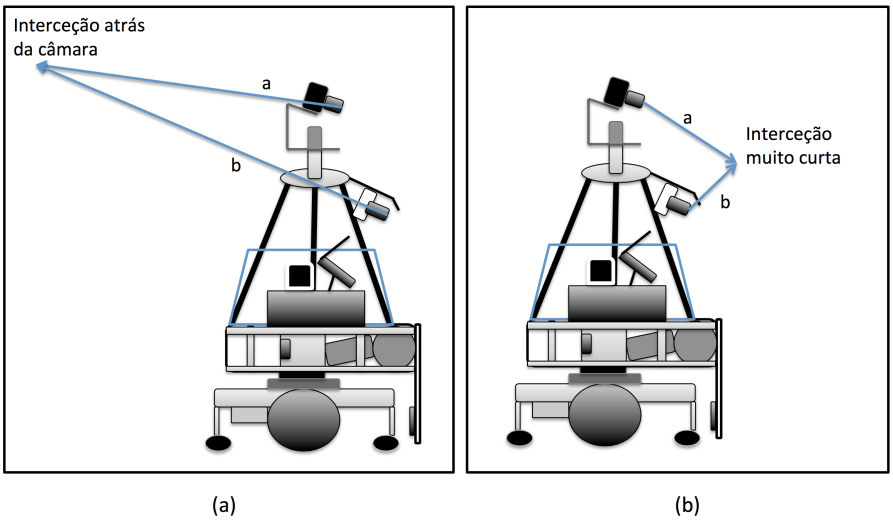


Figura 5.10: (a)Interceção de a e de b atrás da câmara. (b)Interceção muito curta.

Os valores de a e b usados para limitar os pontos a passar no segundo critério de validação são de 300 mm e 200 mm respetivamente, sendo que, nessas distâncias de interceção não existiria nenhum objeto.

Aplicação do *template matching*

Nesta fase é implementado o método de *template matching* abordado na secção3.5 para determinar com maior precisão quais as *features* na imagem do *kicker*, correspondem às *features* da imagem da *head*. Nas fases anteriores, aplicaram-se diferentes métodos para eliminar possíveis más correspondências, não tendo sido possível determinar com um grau suficiente de fiabilidade, se um par de *features* das imagens era uma boa correspondência.

A aplicação do *template matching* nesta fase visa conseguir obter uma correspondência entre as *features* de ambas as imagens, sendo para esse efeito desenvolvido um algoritmo que com base nos resultados do *template matching*, seleciona quais são as correspondências corretas ou incorretas.

Inicialmente começa-se por definir o *template* e a janela de pesquisa, em que o *template* é composto por uma janela com um tamanho de 32x32 em torno de uma *feature* da imagem da *head*. A janela de pesquisa tem um tamanho de 51x51, centrada em torno

de uma *feature* da imagem do *kicker*. Na Figura 5.11(a) é ilustrado um exemplo de um *template* em torno de uma *feature* da imagem da *head* e na Figura 5.11(b) uma janela de pesquisa centrada em torno de uma *feature* na imagem do *kicker*.

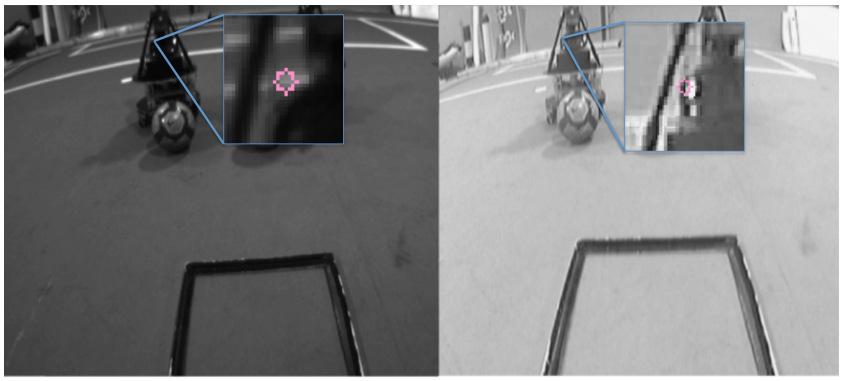


Figura 5.11: (a) *Template* centrado em torno de uma *feature* na imagem da *head*. (b)Janela de pesquisa centrada em torno de uma *feature* na imagem da *kicker*.

Concluída a extração do *template* e da janela de pesquisa, prossegue-se à aplicação da função de *template matching*, que recebe como parâmetros o *template*, a janela de pesquisa e uma *flag* a indicar o método de correlação a usar. O algoritmo usado para correlacionar *template* com a janela de pesquisa foi o coeficiente de correlação cruzada normalizada(γ), calculado recorrendo à equação 3.50.

Tendo terminado a função de *template matching* prossegue-se à analise do resultado da função, em que este resultado é composto por uma matriz com as dimensões da janela de pesquisa. O conteúdo da matriz é o resultado da operação de correlação a cada passagem do *template* na janela de pesquisa, em que os valores de correlação podem variar entre '-1' e '1', sendo que valores próximos de '1' indicam correspondências bem sucedidas e os valores próximos de '-1' correspondências incorretas.

Para analisar os resultados do *template matching* foi desenvolvido um algoritmo para avaliar quais são os pares de *features* correspondentes. Na Figura 5.12 é ilustrado um diagrama que representa o algoritmo de análise implementado.

A primeira fase rejeita os pares de correspondência cujo valor máximo do resultado do *template matching* não é superior a 0.5. No caso do valor máximo ser superior a 0.5, é calculada a diferença absoluta entre o valor máximo(Maxval) e o mínimo(Minval).

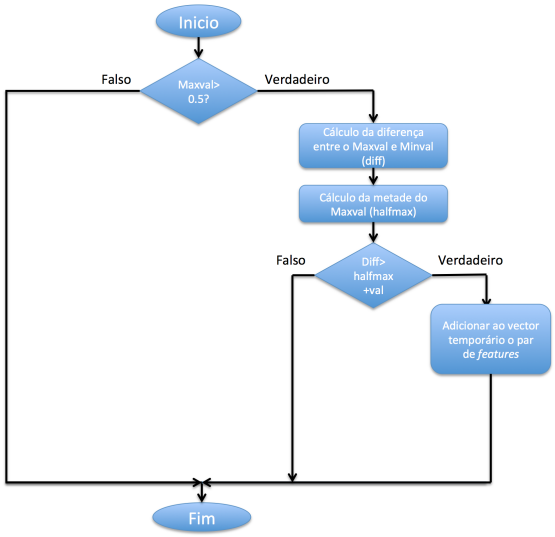


Figura 5.12: Algoritmo de validação dos resultados do *template matching*.

Na segunda fase é validado o valor do resultado da diferença anteriormente citada, e caso esse valor seja superior a metade do valor máximo a condição é verdadeira, sendo adicionado o *keypoint* da imagem do *kicker* ao vetor temporário.

Após terminar o ciclo que percorre o vetor de *features* da imagem do *kicker* e o vetor temporário contiver mais que um *keypoint*, então é possível afirmar que o algoritmo selecionou múltiplas *features* da imagem do *kicker* como possíveis correspondências para uma *feature* da imagem da *head*.

Aplicação do *matcher* de *features*

Tal como se observou na Figura 5.8, as fases de validação anteriores eram executadas no ciclo que percorre o vetor de *keypoints* do *kicker*, sendo que a fase do *matcher* de *features* é executada no ciclo que percorre o vetor de *keypoints* da *head*.

A necessidade de implementação desta fase, deve-se ao fato do resultado do *template matching* poder selecionar múltiplas correspondências na imagem do *kicker* para uma *feature* da imagem da *head*. Desta forma, surge a necessidade de aplicar um *matcher* de *features* para selecionar a melhor correspondência do conjunto selecionado na fase anterior. Tendo em conta que para aplicação do *matcher* é necessário ter um vetor com múltiplas correspondências, esta fase é executada após o ciclo que percorre as *features*

do *kicker* terminar, sendo executada no ciclo que percorre as *features* da *head*.

O matcher de features utilizado é inicializado na fase de detecção de pontos, tal como enunciado na secção 5.1.3. Tirando partido do descritor ORB que é invariante à rotação e à escala, é possível obter melhores resultados.

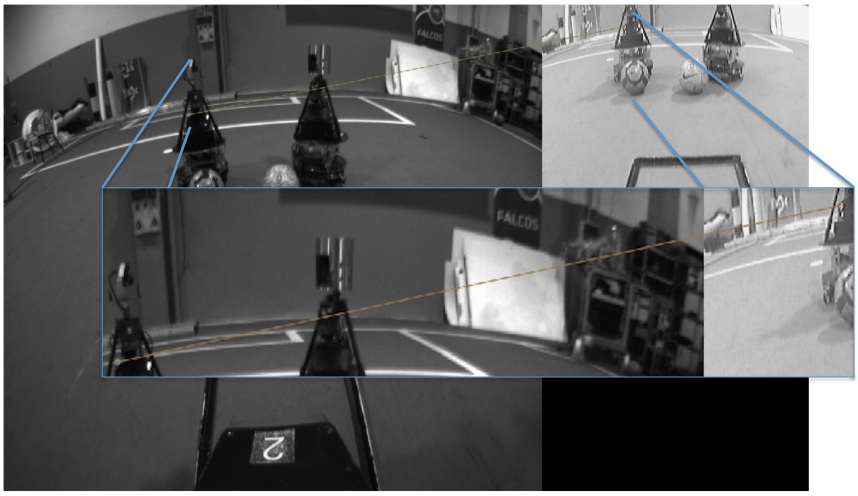


Figura 5.13: Resultado de aplicação do *matcher*.

Para a aplicação do *matcher* é executada uma função que recebe como parâmetros os descritores dos pares de *features* selecionados e as configurações do *matcher*. O resultado da função é um vetor com os índices do par de *features* que é uma correspondência válida. Na Figura 5.13 é possível observar o resultado da aplicação do *matcher*.

5.2 Processo de correspondência de *features* com imagens retificadas

Este processo de correspondência de pontos é semelhante ao processo apresentado na secção 5.1, no entanto este processo apresenta duas fases distintas, uma vez que no processo de *template matching* são utilizadas imagens retificadas. As duas fases introduzidas são: Aplicação da retificação as imagens das câmaras e Aplicação de *template matching* a imagens retificadas. Na Figura 4.5 é ilustrada a arquitetura deste processo de correspondência de *features*.

5.2.1 Aplicação da retificação às imagens das câmaras

Nesta fase é realizada a retificação das imagens lidas na fase de leitura. O método de retificação das imagens pode ser decomposto em três fases, sendo que na primeira etapa são calculados os parâmetros que possibilitam aplicação da retificação. A segunda etapa é compreendida pelo processo de cálculo das variáveis que permitem alinhar as imagens retificadas, na janela que contém cada uma das imagens retificadas. Por fim, a última etapa consiste na aplicação da retificação às imagens originais. Na Figura 5.14 é ilustrado o processo descrito anteriormente.

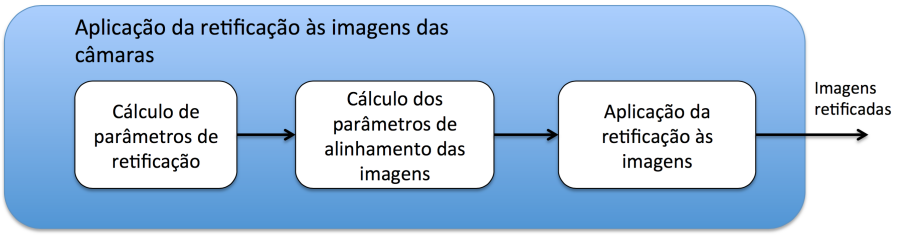


Figura 5.14: Processo de retificação das imagens.

Antes de aplicar a retificação às imagens originais, é necessário remover a distorção existente nas imagens para que o processo de retificação seja efetuado corretamente.

Cálculo dos parâmetros de retificação

Como foi referido anteriormente, nesta etapa são calculados os parâmetros necessários para aplicação da retificação às imagens. Como tal, foi desenvolvida uma função que recebe os parâmetros extrínsecos de cada câmara, a posição dos centros óticos, e as variáveis de alinhamento das imagens retificadas. Por sua vez, a função retorna as novas matrizes projetivas das imagens retificadas e as respetivas matrizes de homografia de retificação. Os parâmetros de alinhamento das imagens são enviados com o valor zero.

Inicialmente começa-se por calcular a matriz R_{rect} dada pela equação 3.37. Esta equação necessita dos vetores e_1, e_2 e e_3, sendo calculados pelas equações 3.32, 3.35 e 3.36 respetivamente, em que $c_l = c_{head}$ e $c_r = c_{kicker}$. O vetor k corresponde à terceira linha da matriz de rotação dos parâmetros extrínsecos da câmara da *head*.

Após o cálculo da matriz R_{rect} são calculadas as novas matrizes dos parâmetros intrínsecos das imagens retificadas, em que, ambas as matrizes contêm os valores da

matriz de parâmetros intrínsecos da câmara do *kicker*. Desta forma, ambas as imagens retificadas apresentam a mesma distância focal, sendo que os centros óticos podem ser alinhados somando os parâmetros d_{head} e d_{kicker}, parâmetros estes que são vetores com duas posições calculados na etapa seguinte.

Através das novas matrizes dos parâmetros intrínsecos ($A_{headNew}$ e $A_{kickerNew}$) e a matriz R_{rect} é possível calcular as novas matrizes projetivas de cada câmara ($PPM_{headRect}$ e $PPM_{kickerRect}$), recorrendo às equações 3.38 e 3.39.

Tendo calculado as matrizes projetivas das imagens retificadas ($PPM_{headRect}$ e $PPM_{kickerRect}$) e calculando as matrizes projetivas (PPM_{head} e PPM_{kicker}) das câmaras com a equação 3.5, torna-se possível obter as matrizes da homografia de retificação ($H_{headRect}$ e $H_{kickerRect}$), ou seja, obtém-se as matrizes que permitem transformar a imagem original na imagem retificada. As matrizes de homografia podem ser obtidas aplicando as equações 3.40 e 3.41.

Cálculo dos parâmetros de alinhamento das imagens

Nesta fase são calculados os parâmetros d_{head} e d_{kicker} que permitem alterar a posição das imagens retificadas na janela que contém a imagem. O processo para determinar estes parâmetros inicia-se com a definição de dois vetores de duas posições com a posição do centro das imagens originais ($Psize_{head}$ e $Psize_{kicker}$), sendo dado pelas seguintes equações 5.8 e 5.9.

$$Psize_{head} = \begin{bmatrix} Imghead_{with} & Imghead_{height} \end{bmatrix} \tag{5.8}$$

$$Psize_{kicker} = \begin{bmatrix} Imgkicker_{with} & Imgkicker_{height} \end{bmatrix} \tag{5.9}$$

Onde $Imghead_{with}$ e $Imghead_{height}$ são a largura e a altura da imagem da câmara da *head* e $Imghead_{with}$ e $Imghead_{height}$ são a largura e a altura da imagem da câmara da *kicker*.

Após o cálculo dos centros das imagens originais, é aplicada a homografia de retificação, de forma a obter o ponto calculado nas imagens retificadas. A aplicação da homografia aos centros óticos das imagens originais são calculados através das equações

5.10 e 5.11.

$$P_{headRect} = H_{headRect} Psize_{head} \tag{5.10}$$

$$P_{kickerRect} = H_{kickerRect} Psize_{kicker} \tag{5.11}$$

Através de $Psize_{head}$ e $P_{headRect}$ torna-se possível determinar d_{head} recorrendo à equação 5.12. O parâmetro de alinhamento da imagem do *kicker* retificada é obtida pela equação 5.13.

$$d_{head} = Psize_{head} - \frac{P_{headRect}(1:2)}{P_{headRect}(3)} \tag{5.12}$$

$$d_{kicker} = Psize_{kicker} - \frac{P_{kickerRect}(1:2)}{P_{kickerRect}(3)} \tag{5.13}$$

De forma a que o alinhamento vertical das duas imagens retificadas seja o mesmo, a segunda posição de d_{head} é igual à segunda posição de d_{kicker} mais um valor de ajuste. O valor de ajuste utilizado foi de 260, dado que, este valor garante que uma grande parte das imagens retificadas seja visível na janela.

Tendo calculado os parâmetros de alinhamento das imagens, realiza-se novamente o cálculo dos parâmetros de retificação implementando os parâmetros de alinhamento.

Aplicação da retificação às imagens das câmaras

Nesta etapa é realizada a retificação das imagens obtidas na fase de leitura, sendo então executada uma função que permite realizar a retificação das imagens aplicando a matriz de homografia de retificação às imagens originais.

A função utilizada nesta etapa foi a *warpPerspective* do OpenCv, visto que esta função permite aplicação de uma matriz de homografia a uma imagem. Esta função recebe como parâmetros de entrada a imagem original, a matriz da homografia de retificação, o tamanho da imagem de saída, o método de interpolação e o método de extrapolação dos pixeis. O retorno da função é imagem retificada, como se pode observar na Figura 5.15

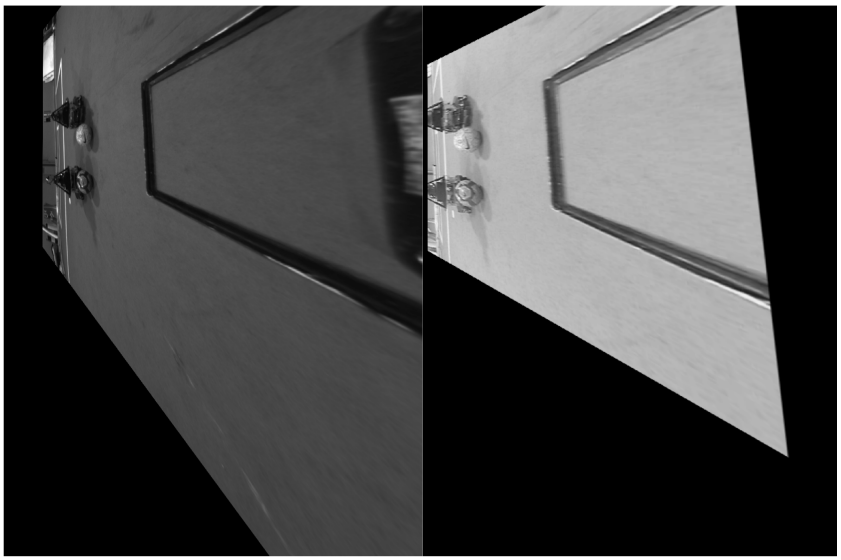


Figura 5.15: (a) Imagem retificada da câmara da *head*.(b)Imagem retificada da câmara da *kicker*.

A função *warpPerspective* foi aplicada a cada uma das imagens, sendo utilizadas as correspondentes homografias de retificação de cada câmara.

Em ambas as execuções, a *flag* do método de interpolação é INTER_LINEAR e *flag* do método de extrapolação é BORDER_CONSTANT. O tamanho da imagem de saída foi definido como sendo duas vezes o tamanho da imagem original da *head* para a retificação de ambas a imagens.

5.2.2 Aplicação do *Template matching* nas imagens retificadas

Nesta etapa é aplicado o template matching às imagens retificadas, sendo esta fase semelhante à enunciada na secção 5.1.5.

Tal como template matching descrito na secção 5.1.5, esta fase também é executada na etapa de validação, no entanto, a extração do *template* e da janela de pesquisa é diferente, dado que, as *features* das imagens originais não se encontram na mesma posição nas imagens retificadas. Deste modo, é necessário aplicar a homografia de retificação as *features* da imagem original, de forma a visualizar essas *features* nas imagens retificadas, sendo necessário aplicar a equações 5.10 e 5.11.

As features ao qual se vai aplicar a homografia devem estar despromovidas de dis-

torção, uma vez que o método de retificação exige imagens sem distorção.

Concluído o processo de retificação dos pontos, prossegue-se à extração do *template* e a janela de pesquisa. De seguida é aplicada a função do *template matching* utilizando o método de correlação referido na secção 5.1.5.

Após a função do *template matching* terminar, é aplicado o algoritmo para interpretar os resultados retornado pela função.

Na eventualidade do resultado do *template matching* gerar múltiplos pares de correspondências, é executada a fase de aplicação do *matcher* de *features* para selecionar qual a melhor correspondência dos pares selecionados na fase anterior.

Esta página foi intencionalmente deixada em branco.

Capítulo 6

Resultados

Neste capítulo são apresentados os resultados obtidos nas diversas experiências efetuadas, para a validação do trabalho desenvolvido nos capítulos anteriores. Assim sendo, este capítulo foi dividido em três grandes etapas: comparação entre múltiplos pares detetor/descritor, método de correspondência de *features* e comparação entre métodos de correspondência de *features*.

A primeira etapa consiste no processo de comparação entre múltiplos pares detetor/-descritor, com o objetivo de determinar qual seria o melhor par a utilizar no método implementado. O procedimento experimental realizou-se em quatro cenários de teste, no qual são avaliados quatro tipo de detetor/descritor.

Na segunda etapa são apresentados os resultados obtidos nas principais fases do método. Os resultados apresentados foram obtidos nos cenários do procedimento experimental enunciado na etapa anterior.

Na última etapa são comparados os resultados dos métodos de correspondência de *features* desenvolvidos com outro método de correspondência de *features* já existente.

6.1 Comparação entre múltiplos pares detetor/descritor

Nesta secção pretende-se comparar os detetores e descritores abordados no subcapítulo 2.2, de forma a determinar qual o par detetor/descritor adequado para implementar no método a desenvolver. Assim sendo, os detetores e os descritores são avaliados de acordo com os seguintes parâmetros:

- Tempo de processamento.
- Número de *features* detetadas.
- Dispersão das *features* detetadas.

O tempo de processamento de cada detetor/descritor é o fato mais significativo nos testes realizados, uma vez que é objetivo a utilização do detetor/descrito em aplicações de tempo, sendo como tal necessário que o tempo de processamento seja o mais baixo possível.

O segundo aspeto de caracterização dos testes passa por avaliar a quantidade de *features* extraídas pelo detetor, ou seja, pretende-se avaliar se detetor obtém uma quantidade de *features* razoável no cenário de aplicação.

O último aspeto de avaliação é a dispersão das *features*, sendo que o objetivo passa por classificar a quantidade de *features* extraídas em objetos relevantes do cenário experimental, como por exemplo: robôs ou bolas. Tendo em conta o cenário experimental, é importante que o detetor extraia uma quantidade suficiente de *features* de forma a ser possível caracterizar o objeto. No caso do detetor não extrair quantidade de *features* nos objetos visualizados nas duas imagens, como consequência pode existir uma probabilidade menor de existirem correspondências, não sendo viável para a presente dissertação.

Assim sendo, este parâmetro foi avaliado segundo uma escala numérica de 0 até 3, tal como é possível observar na Tabela 6.1.

Tabela 6.1: Tabela de classificação da dispersão das *features*

Número de *features* Dispersas	Classificação
n <15	0
15 $\leqslant$ n <30	1
30 $\leqslant$ n $\leqslant$ 45	2
n >45	3

6.1.1 Procedimento de teste

Os testes realizados para comparar os vários pares detetor/descritor têm como base a mesma plataforma experimental. Esta plataforma experimental contempla o uso da

câmara da zona da *head* e a câmara da zona do *kicker* do robô mencionado na secção 4.1. Estas câmaras têm orientações independentes, no entanto ambas tem um campo de visão aproximadamente comum, como se pode visualizar na Figura 4.2. Para além do campo de visão não ser exatamente o mesmo, as câmaras utilizadas têm resoluções diferentes e lentes com *Field of view* (FOV) e abertura diferentes.

O procedimento experimental pode ser descrito nos seguintes tópicos:

- Extração da região de interesse na imagem da câmara da *head*;
- Aplicação do detetor de *features* na imagem da câmara *head* e na imagem da câmara *kicker*;
- Aplicar o descritor de *features* às imagens usadas no passo anterior;
- Análise das imagens com as *features* obtidas pelo detetor.

A primeira etapa consiste na aplicação de uma região de interesse à imagem da câmara da *head*, de forma a obter uma imagem cujo campo de visão aproxima-se da imagem da câmara do *kicker*, como foi detalhado na secção 5.1.3.

Na segunda etapa é aplicado o detetor de *features* inicializado com as configurações pré-definidas da biblioteca dos algoritmos.

Na etapa seguinte é utilizado o descritor de *features* nos *keypoints* resultantes do detetor, sendo este configurado com os parâmetros pré-definidos.

A última etapa consiste numa análise dos resultados experimentais obtidos pela aplicação de teste, contendo informação do tempo de execução dos processos e o número de *features* obtidas pelo detetor, sendo que, os resultados obtidos são avaliados de acordo com os parâmetros enunciados anteriormente.

A aplicação desenvolvida para testar os algoritmos foi elaborada na linguagem C++, com integração da biblioteca de visão computacional OpenCV, sendo que esta biblioteca contém todos os algoritmos dos detetores e descritores utilizados nesta fase de experimental.

A plataforma que executa a aplicação de teste é constituída por um portátil a executar uma máquina virtual com o sistema operativo *Ubuntu 12.04 LTS*, um processador de dois núcleos a 2.0 GHz e 1GB de memória RAM.

6.1.2 Resultados experimentais

O procedimento experimental referido anteriormente foi aplicado a 4 ambientes experimentais contendo uma imagem de cada câmara, no qual estão presentes objetos comuns do cenário de aplicação da plataforma robótica, como se pode observar nas Figuras 6.1, 6.2, 6.3 e 6.4.

Os algoritmos de deteção e de descrição de *features* testados são: SIFT, SURF, FAST/BRIEF e ORB.

Cenário nº1

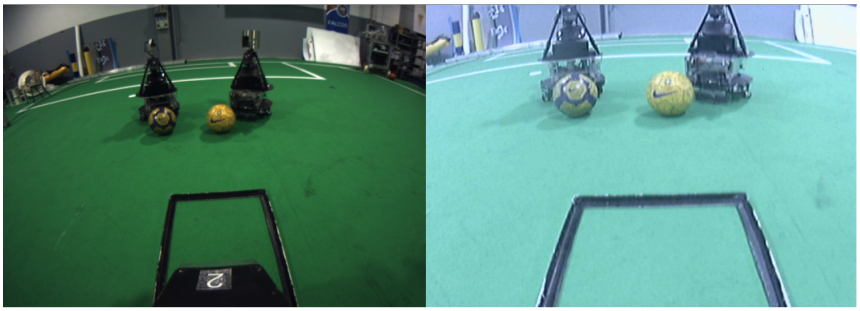


Figura 6.1: Cenário de teste nº1.

Na Tabela 6.2 são apresentados os resultados do teste no cenário ilustrado na Figura 6.1. As imagens usadas para avaliar a dispersão encontram-se no Anexo A.

Tabela 6.2: Resultados do teste no cenário nº1

	SIFT		SURF		FAST/BRIEF		ORB	
	Head	*Kicker*	*Head*	*Kicker*	*Head*	*Kicker*	*Head*	*Kicker*
Tempo de processamento detetor (s)	0.21248	0.113054	0.175771	0.105029	0.000955343	0.000998497	0.0106425	0.00633097
Tempo de processamento descritor (s)	0.40776	0.144423	0.16348	0.127659	0.0156512	0.00845194	0.0267358	0.00683117
Número de *features*	447	779	639	700	1127	1739	500	500
Dispersão das *features*	2	3	2	3	3	3	1	2

Relativamente ao tempo de processamento do detetor e do descritor o par FAST/BRIEF é o mais rápido. O par mais lento é o SIFT, seguido pelo SURF apresenta um tempo

de processamento ligeiramente mais rápido, sendo mais notável a diferença no tempo de processamento do descritor.

Em relação à dispersão das *features*, o detetor FAST produz uma grande quantidade *features* nas imagens nas zonas de interesse em ambas as imagens.

Cenário nº2

Figura 6.2: Cenário de teste nº2.

Na Tabela 6.3 são apresentados os resultados do teste no cenário ilustrado na Figura 6.2. As imagens usadas para avaliar a dispersão encontram-se no Anexo B.

Tabela 6.3: Resultados do teste no cenário nº2

	SIFT		SURF		FAST/BRIEF		ORB	
	Head	*Kicker*	*Head*	*Kicker*	*Head*	*Kicker*	*Head*	*Kicker*
Tempo de processamento detetor (s)	0.208204	0.133585	0.186434	0.122096	0.00104833	0.00132084	0.00865555	0.00694323
Tempo de processamento descritor (s)	0.403735	0.143387	0.243121	0.151432	0.0100286	0.00966573	0.0251427	0.00725198
Número de *features*	421	971	827	817	1096	2029	500	500
Dispersão das *features*	1	2	2	2	2	3	1	1

O ORB é o método com o tempo de processamento mais baixo, seguido do par FAST/BRIEF. No entanto, o tempo de processamento do descritor do ORB aumenta consoante o tamanho da imagem.

O tempo de processamento do detetor FAST é maior na imagem do *kicker*, sendo que nessa imagem é obtida um número maior de *features*, traduzindo-se num aumento dos testes realizados para validar um ponto como *feature*.

Em relação ao nível de dispersão das *features*, o SIFT oferece ótimos resultados, dado que este obtém um bom número de *features* relevantes, que permitem uma boa caracterização dos objetos. O SURF produz um número elevado de *features* em objetos relevantes, no entanto, algumas das *features* produzidas encontram-se em zonas exteriores ao objeto de interesse, não oferecendo um bom contributo para caracterização do objeto.

Cenário nº3

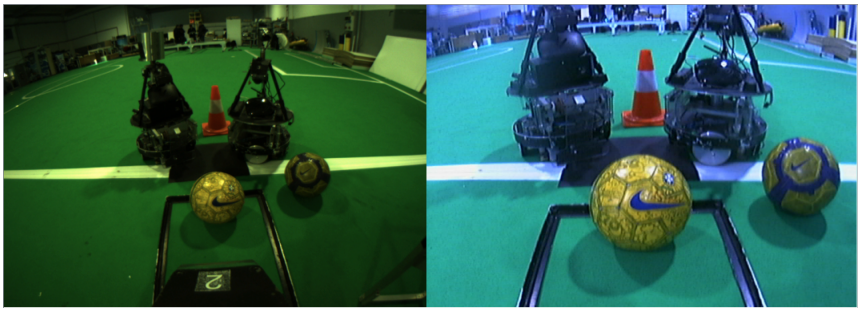


Figura 6.3: Cenário de teste nº3.

Na Tabela 6.4 são apresentados os resultados do teste no cenário ilustrado na Figura 6.3. As imagens usadas para avaliar a dispersão encontram-se no Anexo C.

Tabela 6.4: Resultados do teste no cenário nº3

	SIFT		SURF		FAST/BRIEF		ORB	
	Head	*Kick*	*Head*	*Kick*	*Head*	*Kick*	*Head*	*Kick*
Tempo de processamento detetor (s)	0.221911	0.139845	0.187696	0.126814	0.00116396	0.00196099	0.00958586	0.00859571
Tempo de processamento descritor (s)	0.41523	0.186302	0.245996	0.227181	0.013217	0.0156608	0.0252783	0.00688934
Número de *features*	666	1729	798	1245	1585	3537	500	500
Dispersão das *features*	1	2	2	3	3	3	0	1

Ao nível do tempo de processamento todos os detetores mantêm a mesma coerência sendo a diferença mínima.

O número de *features* produzidas pelos detetores é sempre maior na imagem do *kicker* com a exceção ao detetor SURF e ORB. No detetor SURF a imagem com mais *features*

é aleatória, ao contrário do detetor ORB que gera sempre um número constante de *features*.

A dispersão das *features* do ORB neste cenário foi inferior em comparação com os resultados dos cenários anteriores, nomeadamente na imagem da *head*.

O FAST contém uma grande quantidade de *features* em objetos de interesse na imagem da *head*, embora em alguns casos tenha *features* em excesso.

Cenário nº4

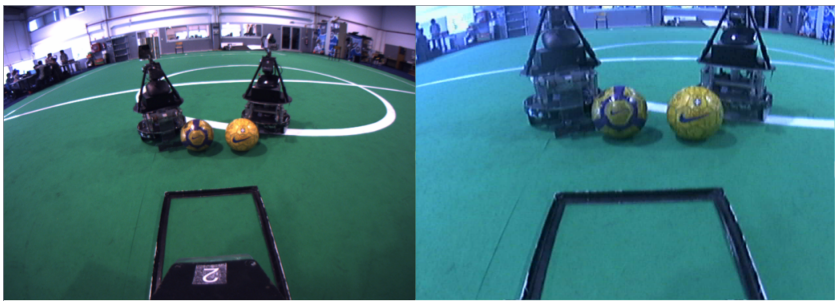


Figura 6.4: Cenário de teste nº4.

Na Tabela 6.5 são apresentados os resultados do teste no cenário ilustrado na Figura 6.4. As imagens usadas para avaliar a dispersão encontram-se no Anexo D.

Tabela 6.5: Resultados do teste no cenário nº4

	SIFT		SURF		FAST/BRIEF		ORB	
	Head	*Kick*	*Head*	*Kick*	*Head*	*Kick*	*Head*	*Kick*
Tempo de processamento detetor (s)	0.214891	0.136869	0.192968	0.111351	0.00127196	0.00121856	0.00928593	0.00883245
Tempo de processamento descritor (s)	0.420171	0.146138	0.271029	0.158257	0.0148447	0.00889659	0.0249977	0.00676489
Número de *features*	655	935	969	755	1797	1800	500	500
Dispersão das *features*	1	2	2	2	3	3	1	1

Apesar de neste cenário experimental as condições de iluminação serem diferentes, existindo um aumento da luz de fundo, os detetores comportam-se de forma semelhante ao exemplo referido anteriormente.

6.1.3 Conclusão

Com base nos testes elaborados, foi possível concluir que alguns dos resultados obtidos são coerentes com os dados mencionados no Capítulo 2.2, mais concretamente os resultados da comparação do tempo de processamento dos detetores e dos descritores. Assim sendo o detetor de *features* mais rápido é o FAST, sendo que o mais lento é o SIFT. Apesar do SIFT ser o detetor/descritor mais lento, este permite controlar o número de *features* a detetar.

Um outro detetor que permite controlar o número de *features* extraídas é o ORB, como se observou ao longo dos testes efetuados.

No que diz respeito ao descritor, o BRIEF tem vantagem relativamente ao tempo de processamento, embora esse tempo varie com o tamanho da imagem. Os restantes descritores têm um tempo de processamento superior a 0.1 s, com a exceção do descritor do ORB.

Na dispersão de *features* o SIFT foi o detetor que obteve melhor distribuição mesmo não ajustando o número de *features*, sendo que o SURF também extrai uma quantidade significativa de *features* dispersas, no entanto muitas dessas *features* situam-se em zonas exteriores aos objetos, não ajudando na caracterização dos objetos. O detetor FAST extraí muitas *features* nos objetos relevantes, embora na maioria dos casos o número de *features* extraídas seja em grande quantidade.

Para aplicações em tempo real o SIFT e o SURF não são considerados escolhas viáveis, uma vez que o tempo de processamento do detetor e do descritor são elevados. Assim sendo FAST/BRIEF e o ORB são as melhores escolhas para aplicar nesta dissertação.

6.2 Método de correspondência de *features*

Nesta secção são apresentados os resultados obtidos nas etapas do método de correspondência desenvolvido, sendo que a plataforma de teste e os cenários experimentais são iguais aos utilizados nos testes de comparação dos pares detetor/descritor, secção 6.1.

6.2.1 Agrupamento de pontos

Como referido na secção 5.1.4, nesta fase é realizada uma seleção de *features* de um conjunto inicial produzido pelo detetor, de forma a que sejam apenas selecionadas as

features dispersas umas das outras.

O processo de seleção utilizou um conjunto de 200 *keypoints* obtidos pelo detetor de *features*, ao qual foi aplicado o algoritmo mencionado com o parâmetro de agrupar *features* de aproximadamente 5 píxeis. Na Figura 6.6 são ilustrados os *keypoints* obtidos pelo detetor sem qualquer processo de seleção.

As condições iniciais mencionadas são as mesmas para todos os cenários de aplicação testados.

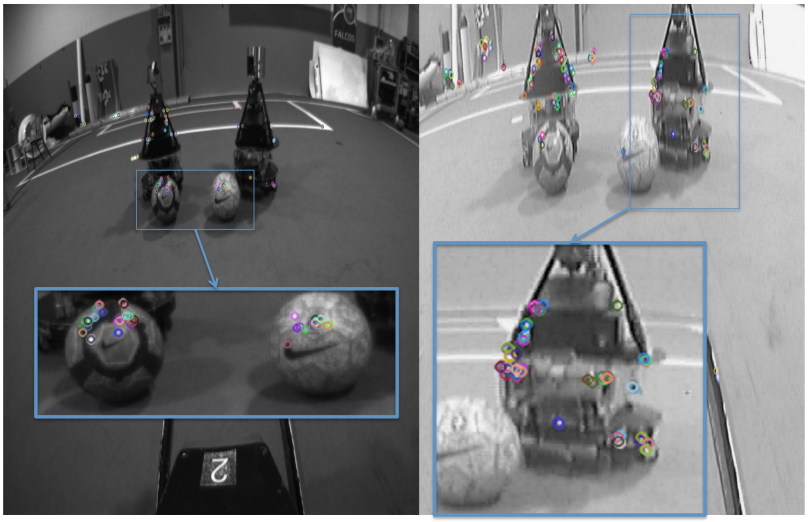


Figura 6.5: Imagens das duas câmaras com *keypoints* obtidos pelo detetor.

A quantificação do número de *features* selecionadas pelo processo e o respetivo tempo total de processamento são apresentados na Tabela 6.6.

Tabela 6.6: Tabela de resultados da fase seleção de pontos.

	Cenário nº 1		Cenário nº 2		Cenário nº 3		Cenário nº4	
	head	kicker	head	kicker	head	kicker	head	kicker
Número de keypoints	100	93	75	106	90	107	78	86
Tempo de processamento(s)	0.0001855	0.00014901	0.00016594	0.00020146	0.00029635	0.00018287	0.00017977	0.00014710

Na Tabela 6.6, é possível observar que aproximadamente 50% dos *keypoints* iniciais

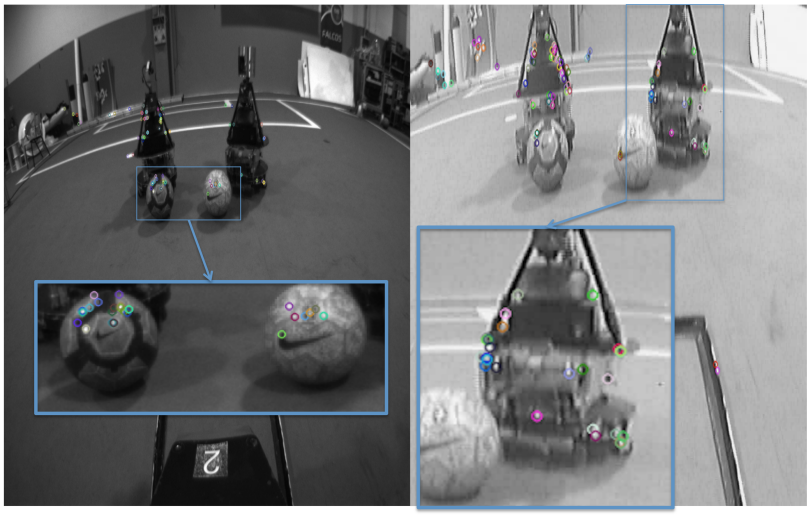


Figura 6.6: Imagens de ambas as câmaras com *keypoints* selecionados.

são filtrados, sendo que no último cenário a percentagem de pontos selecionados aumenta ligeiramente. Analisando o número de *keypoints* de cada câmara, é possível constatar que na imagem da *head* existem menos *features* selecionadas comparativamente com a imagem do *kicker*, em exceção do primeiro cenário. Uma possível explicação seria a baixa dispersão das *features* existente na imagem da *head*, uma vez que na imagem do *kicker* a dispersão é maior.

Relativamente aos tempos de processamento, é possível afirmar que o algoritmo de seleção apresenta um tempo de execução baixo, mesmo combinando o tempo de execução do algoritmo para ambas a câmaras.

6.2.2 Restrição epipolar

Tendo um conjunto de pontos selecionados da fase anterior, procedeu-se à execução do processo de correspondência de *features*. Tal como foi mencionado anteriormente, este processo foi divido em várias etapas permitindo selecionar um conjunto de pares de *keypoints*, sendo estes categorizados como sendo possíveis correspondências.

A primeira etapa deste método consiste na aplicação da restrição epipolar mencionada na secção 5.1.5, sendo que o objetivo consiste em eliminar correspondências que não se encontrem ao longo da linha epipolar.

Com o conjunto inicial de *keypoints* obtidos pelo detetor existiriam cerca de 40000 testes de possíveis correspondências, uma vez na fase de validação de *features* são testados todos os possíveis pares de correspondências. Com a aplicação do algoritmo da fase anterior, o número de possíveis combinações de correspondências é aproximadamente cerca de 10000.

Através do conjunto de *keypoints* selecionados na fase de agrupamento de pontos, é possível aplicar a restrição epipolar, sendo selecionados apenas o pontos que se encontram posicionados sobre a linha epipolar, como se pode visualizar na Figura 6.7.

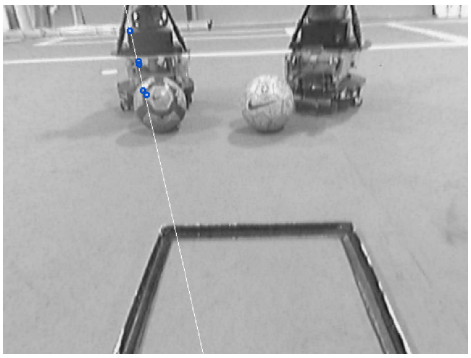


Figura 6.7: Exemplo de aplicação da restrição epipolar na imagem do *kicker*.

Comparando a Figura 6.8 resultante do processo da restrição epipolar com a Figura 6.6 contendo as *features* selecionadas na etapa anterior, é possível observar uma redução do número de *features* na bola da direita da Figura 6.8. Esta redução de pontos está consistente com a ausência de pontos na imagem do *kicker*, como é possível de observar na imagem da direita da Figura 6.6.

Através da Tabela 6.7 é possível observar o número de possíveis correspondências inicial e o numero de correspondências resultante da aplicação do método, bem como o tempo de processamento do método.

Com base nos resultados percetíveis na Tabela 6.7, é possível observar uma grande redução no número de possíveis correspondências, sendo que de uma forma geral apenas 1,6% dos valores iniciais são considerados possíveis correspondências.

O tempo de processamento médio do processo encontra-se nas dezenas de milissegundos, sendo um valor razoável face aos resultados obtidos. O tempo de processamento

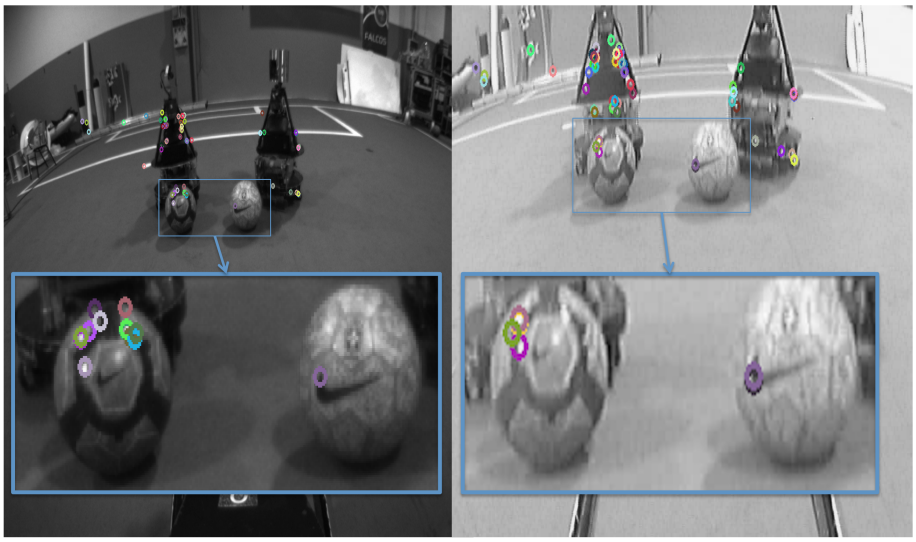


Figura 6.8: Imagens de ambas as câmaras com *keypoints* selecionados na fase de restrição epipolar.

Tabela 6.7: Tabela de resultados da fase da restrição epipolar.

	Cenário nº 1	Cenário nº 2	Cenário nº 3	Cenário nº4
Número inicial de correspondências	9300	7950	9630	6708
Número de correspondências selecionadas	153	115	151	61
Tempo de processamento(s)	0.0030005	0.00245547	0.00290513	0.0021894

aumenta consoante o número de possíveis correspondências iniciais.

6.2.3 Triangulação *stereo*

Posteriormente à etapa de restrição epipolar, procedeu-se à execução da etapa de triangulação *stereo*. Nesta etapa são calculados os parâmetros de triangulação a, b e c, com o objetivo de rejeitar possíveis falsas correspondências entre *keypoints* que não são comuns a ambas as imagens.

Através dos parâmetros a e b é possível rejeitar possíveis não correspondências, determinando se os valores dos escalares são negativos ou muito baixos. Esses escalares poderão tomar valores negativos quando existe uma interceção entre os dois *keypoints*

na parte de trás da câmara. A segunda situação é considerada pois a interceção dos dois *keypoints* ocorre numa zona próxima da câmara. Em ambos os casos rejeitam-se os pares de correspondências que não existem nas zonas comuns às duas imagens, ou as correspondências cujo o ponto tridimensional encontra-se próximo das câmaras.

Com as possíveis combinações de correspondências da fase anterior e aplicação da seleção de pontos por triangulação *stereo*, os resultados podem ser observados na Tabela 6.8.

Tabela 6.8: Tabela de resultados da fase de triangulação *stereo*.

	Cenário nº 1	Cenário nº 2	Cenário nº 3	Cenário nº4
Número inicial de correspondências	153	115	151	61
Número de correspondências selecionadas	118	99	129	57
Tempo de processamento(s)	0.00398064	0.0033524	0.00387263	0.00185037

Apesar de na etapa de deteção de *features* enunciada na secção 5.1.3 ter sido aplicado uma redução à imagem da *head*, com o objectivo de extrair um número comuns de *features* a ambas as imagens, possivelmente continuam a existir zonas incomuns a ambas as câmaras como é possível comprovar pelos resultados presentes na Tabela 6.8.

Esta fase é computacionalmente mais demorada uma vez que são realizados mais cálculos.

6.2.4 *Template matching*

Com a integração da restrição epipolar e da triangulação *stereo* foram rejeitadas possíveis correspondências erradas, no entanto, ainda não foi possível definir em concreto quais os pares corretos de possíveis correspondências. A aplicação do *template matching* visa obtenção de um conjunto de possíveis correspondências mais fidedignas, através dos pontos selecionados das fases anteriores.

Uma vez que foram desenvolvidos dois possíveis métodos no qual a principal diferença reside na aplicação do *template matching*, nesta secção são demonstrados os resultados da fase de *template matching* com as imagens originais e com as imagens retificadas. Em ambas as etapas são demonstrados o número de possíveis correspondências selecionadas, tempo de processamento e uma caracterização dos pares de correspondências.

Nas Tabelas 6.9 e 6.10 são apresentadas as possíveis combinações de correspondências e o tempo de processamento resultantes do algoritmo de interpretação dos dados do *template matching* de ambos os métodos.

Tabela 6.9: Tabela de resultados da fase de *template matching* com imagens originais.

	Cenário nº 1	Cenário nº 2	Cenário nº 3	Cenário nº4
Número inicial de Correspondências	118	99	129	57
Número de correspondências selecionadas	49	25	57	30
Tempo de processamento template matching(s)	0.0204682	0.0173118	0.0221229	0.0111079
Tempo de processamento do algoritmo(s)	0.00008845	0.00005961	0.00007749	0.00004458

Tabela 6.10: Tabela de resultados da fase de *template matching* com imagens retificadas.

	Cenário nº 1	Cenário nº 2	Cenário nº 3	Cenário nº4
Número inicial de Correspondências	118	99	129	57
Número de correspondências selecionadas	52	33	44	24
Tempo de processamento template matching(s)	0.0214128	0.0175574	0.0229881	0.0114689
Tempo de processamento do algoritmo(s)	0.00007868	0.00006106	0.00009346	0.00005341

De uma forma geral, ambos os métodos reduzem o número de possíveis correspondências para aproximadamente metade nas quatro experiências como é percetível nas tabelas anteriores.

No que diz respeito ao tempo de processamento, é possível constatar que o método de *template matching* com as imagens retificadas requer mais tempo, uma vez que para além da aplicação do método de *template matching*, é necessário retificar os *keypoints* em ambas as imagens de forma a ser possível extrair o *template* e a janela de pesquisa nas imagens retificadas.

A caracterização das correspondências é realizada de acordo com os seguintes parâmetros:

- VV - Indica que o par de correspondência é verdadeiro e foi assinalado como verdadeiro pelo algoritmo;
- FF - Indica que o par de correspondência é falso e foi assinalado como falso pelo algoritmo;
- PV - refere-se a um par de correspondência falso, no entanto encontra-se numa zona próxima de onde a correspondência verdadeira estaria, sendo assinalado pelo algoritmo.
- VF - Indica um par de correspondência que é verdadeiro mas foi assinalado como falso pelo algoritmo;
- FV- Indica um par de correspondência que é falso mas foi assinalado como verdadeiro pelo algoritmo;
- PF - Indica que um par de correspondência encontra-se próximo da zona de correspondência correta, no entanto não foi assinalada pelo algoritmo.

Os resultados da caracterização das correspondências podem ser visualizados nas Figuras 6.9 e 6.10. Os resultados obtidos por ambos os métodos, foram caraterizados comparando uma análise humana dos mesmos pontos atribuindo a classificação mencionada anteriormente.

Nos gráficos apresentados é possível observar que ambos os métodos têm uma percentagem elevada de falsas correspondências detetadas corretamente(FF), embora no processo com as imagens retificadas o número de FF seja ligeiramente superior. Tendo em conta que a percentagem de correspondências falsas, na amostra, é maior, sendo que consequentemente a percentagem de falsos positivos(FV) é menor para imagens retificadas. O *template matching* nas imagens originais aparenta ser mais suscetível à aceitação de *features* falsas mas próximas do local correto(PV). Contudo, ambos métodos ignoram algumas possíveis correspondências verdadeiras(VF) e correspondências próximas(PF).

Apesar do uso de imagens retificadas com o *template matching*, oferecer melhor percentagem nos resultados face à aplicação do *template matching* com as imagens originais, o processo de retificação acarreta custos de computacionais maiores, reduzindo o leque de aplicações do mesmo.

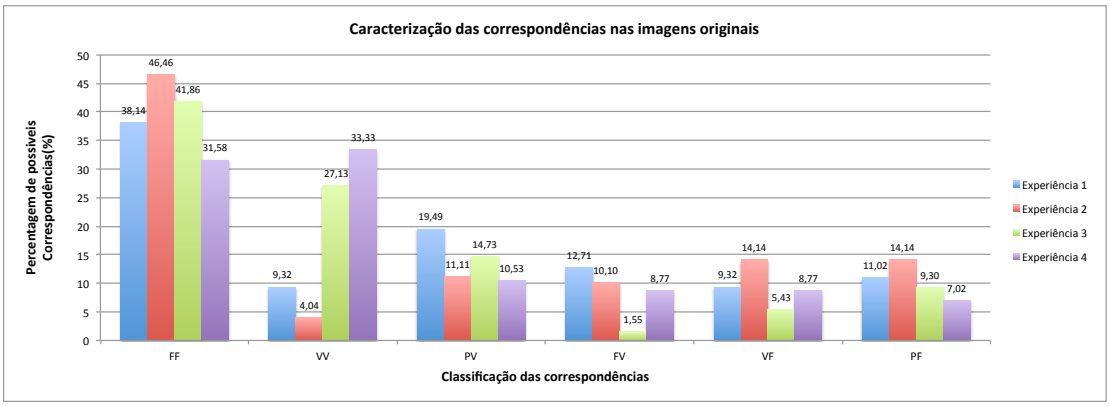


Figura 6.9: Gráfico com a caracterização das correspondências do *template matching* com as imagens originais.

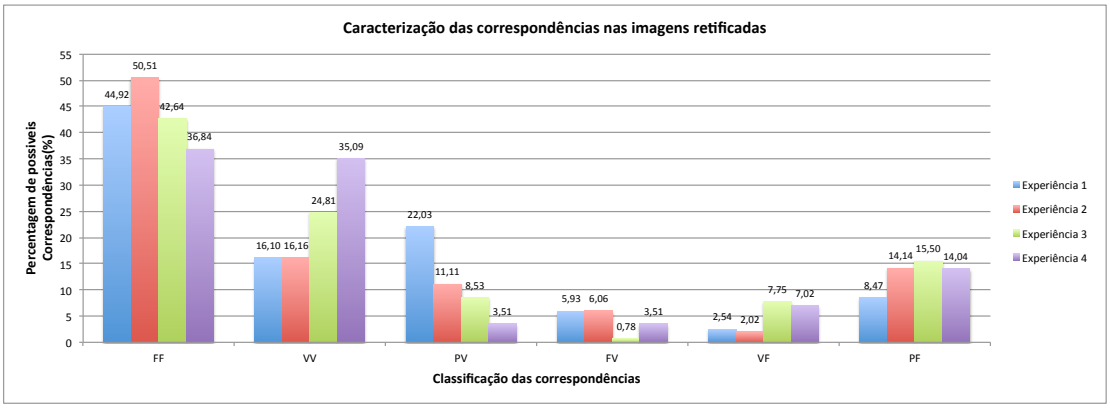


Figura 6.10: Gráfico com a caracterização das correspondências do *template matching* com as imagens retificadas.

6.2.5 *Matcher* de *features*

Uma vez que o *template match* pode gerar múltiplas possíveis correspondências para uma *feature* da imagem da *head*, tornou necessário aplicação de um *matcher* de *features* de forma a conseguir extrair apenas uma possível correspondência para cada *feature* da imagem da *head*. Desta forma o resultado do *matcher de features* pode ser observado na Figura 6.11.

Através das possíveis *features* selecionadas nas fases de *template matching*, é possível implementar o *matcher* de *features*, sendo o número de possíveis correspondências de

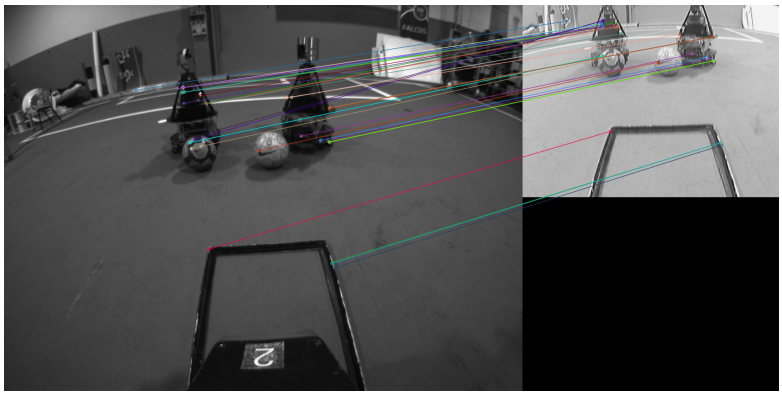


Figura 6.11: Resultado de aplicação do *matcher* de *features* aos resultados de *template matching* nas imagens retificadas.

ambos as fases apresentado na Tabelas 6.11 e 6.12.

Tabela 6.11: Tabela de resultados da fase de aplicação do *matcher* de *features* nas imagens originais.

	Cenário nº 1	Cenário nº 2	Cenário nº 3	Cenário nº4
Número inicial de correspondências	49	25	57	57
Número de correspondências selecionadas	28	20	25	19
Tempo de processamento(s)	0.000196457	0.000171661	0.000235558	0.000148296

Tabela 6.12: Tabela de resultados da fase de aplicação do *matcher* de *features* nas imagens retificadas.

	Cenário nº 1	Cenário nº 2	Cenário nº 3	Cenário nº4
Número inicial de correspondências	52	33	44	24
Número de correspondências selecionadas	34	22	25	14
Tempo de processamento(s)	0.000282526	0.000223637	0.000184298	0.0001297

Nas Tabelas 6.11 e 6.12, é possível observar que o número de possíveis correspondências baixou novamente, uma vez que apenas existe uma possível correspondência para cada *feature* na imagem da *head*.

O tempo de processamento de ambos os métodos encontra-se nas centenas de micro segundos.

A caracterização dos resultados do *matcher* nas imagens originais e nas imagens retificadas, são percetíveis nas Figuras 6.12 e 6.13, sendo que é utilizada a mesma classificação da comparação da fase anterior.

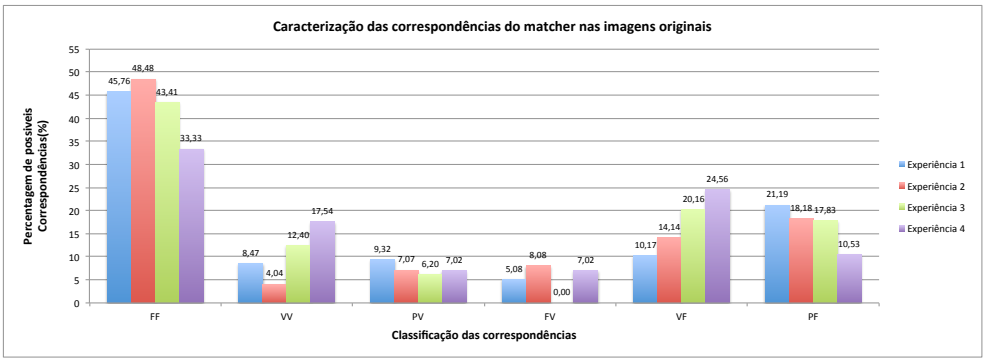


Figura 6.12: Gráfico com a caracterização das correspondências do *matcher* com as imagens originais.

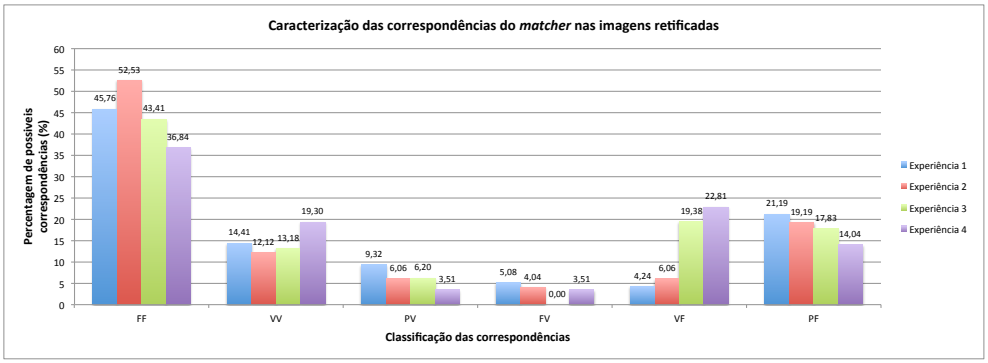


Figura 6.13: Gráfico com a caracterização das correspondências do *matcher* com as imagens retificadas.

Com aplicação do *matcher*, observou-se uma pequena redução no número de falsos positivos em ambos os métodos, no entanto ocorreu um aumento de possíveis correspondências ignoradas(VF e PF). Com esse aumento, é possível constatar que em alguns casos existem múltiplos *keypoints* na imagem do *kicker* para um *keypoint* da imagem da *head*.

6.3 Comparação entre métodos de correspondência de *features*

Com o objetivo de avaliar os resultados obtidos pelos métodos implementados, optou-se por comparar os métodos implementados com um outro método de correspondência de *features* já existente. Assim sendo, foi utilizado o detetor e descritor de *features* ORB com um *Brute-Force matcher* com distância de Hamming para comparar os descritores, e validando se as correspondências são iguais nas duas imagens (*crosscheck*).

O processo experimental é semelhante para ambos os métodos, sendo que na fase de deteção de *features* foi aplicada a redução da imagem da *head*, de forma a que extração de *features* seja realizada de igual forma em ambos os métodos.

O resultado das correspondências de *features* de ambos os métodos podem ser visualizados nas Figuras 6.14, 6.15 e 6.16, sendo que os métodos foram executados no cenário experimental nº1.

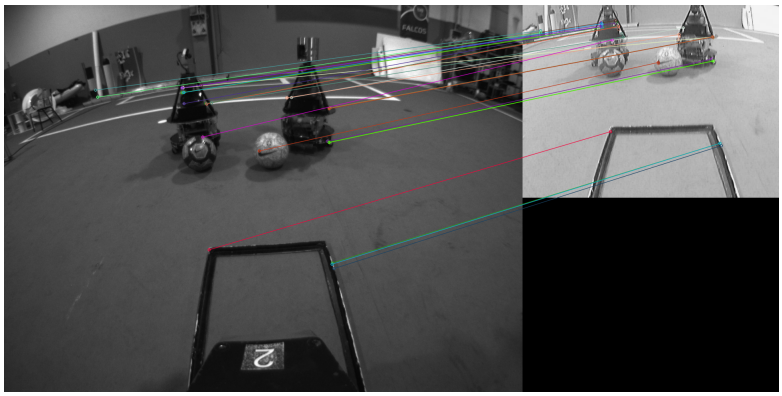


Figura 6.14: Resultado de aplicação do método correspondência de *features* desenvolvido com imagens originais.

A caracterização do número de correspondências resultante da aplicação dos três

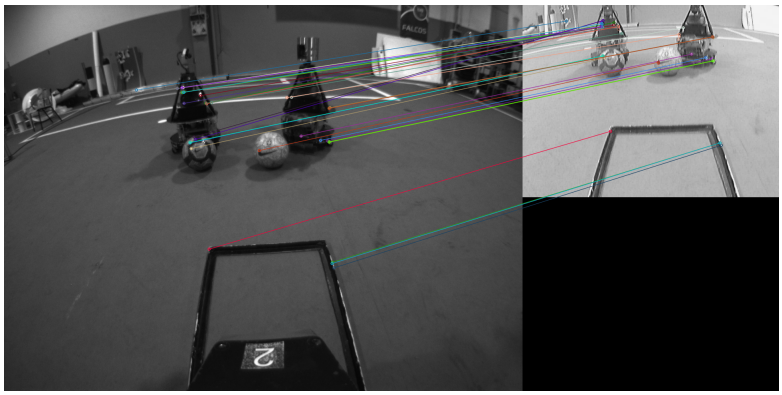


Figura 6.15: Resultado de aplicação do método correspondência de *features* desenvolvido com imagens retificadas.

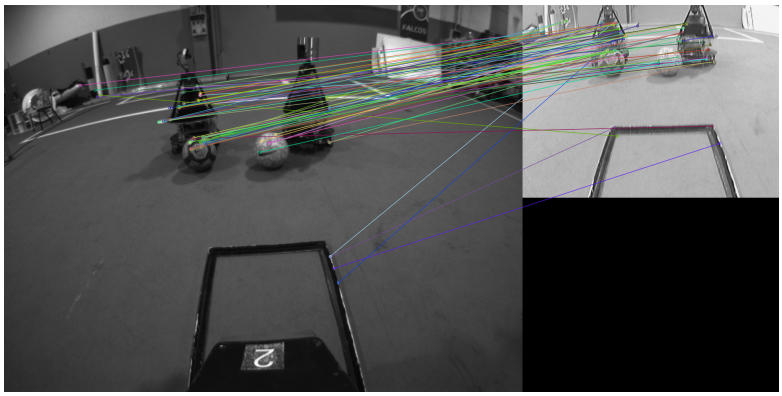


Figura 6.16: Resultado de aplicação do método correspondência de *features* referido.

métodos enunciados no cenário de aplicação, é percetível na Figura 6.17. De forma a ser possível estabelecer uma comparação mais precisa entre os diferentes métodos, foram apenas caracterizados os resultantes provenientes de cada método. Como tal, os parâmetros de caracterização dos métodos de correspondências são: VV, PV e FV.

A taxa de sucesso de cada método é representada na Tabela 6.13, no qual a taxa de sucesso é obtida recorrendo à equação 6.1.

$$Taxa_{sucesso} = \frac{N_{VV} + N_{PV}}{N_{Total}} \tag{6.1}$$

Onde N_{VV} é o número de correspondências corretas, N_{PV} é o número de corres-

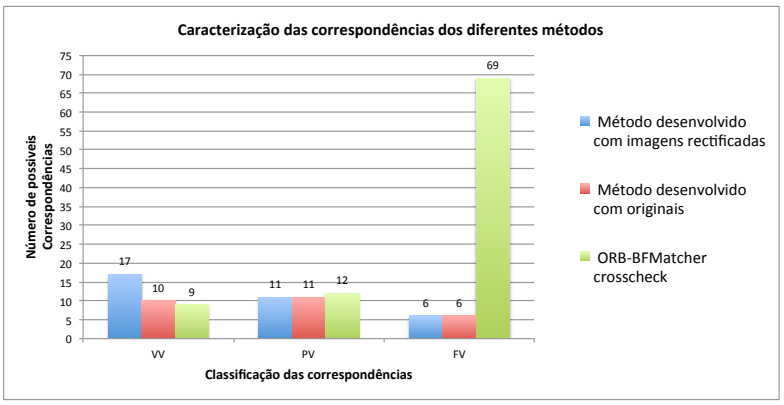


Figura 6.17: Gráfico com a caracterização das correspondências dos métodos enunciados.

pondências aproximadamente corretas e N_{Total} é número total de correspondências do método.

Tabela 6.13: Tabela com as taxas de sucesso de cada método abordado.

	Método desenvolvido com imagens Originais	Método desenvolvido com imagens retificadas	ORB BFMatcher com Crosscheck
Taxa de Sucesso(%)	77.78	82.35	23.33

Com base na Figura 6.17 é possível constatar que o ORB com *Brute Force Matcher crosscheck* é o método que gera maior quantidade de correspondências, sendo que a maioria das correspondências são falsas, traduzindo-se numa baixa taxa de sucesso, como é possível observar na Tabela 6.13.

Os métodos desenvolvidos apresentam um menor número de correspondências, no entanto o número de falsas correspondências é significativamente menor sendo que ambos os métodos apenas obtiveram 6 correspondências falsas. Entre os dois métodos desenvolvidos o método que utiliza as imagens retificadas obteve uma taxa de sucesso de 82.35%.

Esta página foi intencionalmente deixada em branco.

Capítulo 7

Conclusão e Trabalho Futuro

Esta dissertação abordou o desenvolvimento de uma abordagem de correspondência de *features* para sistemas de visão multi-câmara, sendo testado no cenário do futebol róbotico.

Numa fase inicial, realizou-se um estudo com o propósito de estudar métodos de correspondência de pontos já existentes, bem como métodos de extração de *features* e *matching* de *features*. Com a análise do estudo anterior, foi desenvolvido um método de correspondência de *features*.

Previamente à elaboração da nova abordagem de correspondência de pontos, efetuou-se um estudo experimental comparando múltiplos pares detetores/descritores, com o intuito de determinar qual o par detetor/descritor a utilizar. Com base no resultado final do estudo, decidiu-se optar pelo detetor/descritor ORB, uma vez que este cumpre os requisitos delineados.

Na abordagem desenvolvida procurou-se tirar proveito do conhecimento relativamente à posição e orientação das câmaras, fazendo uso de métodos como a geometria epipolar, a triangulação *stereo* e a retificação para ajudar no processo de correspondência.

A abordagem elaborada foi dividida em duas implementações, onde um dos métodos utiliza as imagens originais obtidas pelas câmaras e no outro método são utilizadas as imagens retificadas.

O método de correspondência passa por inicialmente extrair um grupo de *features* em ambas as imagens, ao qual, são selecionadas as *features* mais distintas do grupo

inicial. Com o grupo de *features* selecionadas de ambas as imagens prossegue-se ao processo de correspondência de pontos, implementando a restrição epipolar, triangulação *stereo*, *template matching* e um *matcher* de *features*. O método de correspondência com as imagens retificadas é semelhante ao descrito anteriormente, no entanto, este método apresenta uma etapa adicional onde é realizada a retificação das imagens originais, para serem utilizadas na fase de *template matching*.

Ambos os métodos de correspondência foram validados com sucesso no cenário do futebol róbotico, conseguindo obter uma taxa reduzida de correspondências incorretas, atingindo assim um dos objetivos propostos. As fases de restrição epipolar e de triangulação permitiram rejeitar uma quantidade de correspondências incorretas. Nas fases seguintes foram selecionadas as correspondências corretas, sendo apenas selecionado uma baixa taxa de correspondências incorretas.

Quando comparando o desempenho dos métodos desenvolvidos, é possível constatar que o método com as imagens retificadas produz um número menor de correspondências incorretas e um número ligeiramente superior de correspondências corretas.

Com o intuito de avaliar a performance dos métodos desenvolvidos face a outras soluções existentes, optou-se por comparar os métodos elaborados com um extrator e um *matcher* de *features*. O resultado da comparação comprovou que os métodos desenvolvidos obtém um número significativamente menor de correspondências incorretas face ao outro método utilizado na comparação.

Com base nos resultados obtidos na avaliação dos métodos desenvolvidos, é possível afirmar que todos os objetivos propostos, foram completados com sucesso.

Relativamente ao trabalho futuro, existem várias melhorias a aplicar ao método no sentido de melhorar a sua performance. Em termos da abordagem desenvolvida, esta pode ser otimizada de forma a ser possível integrar o método numa plataforma róbotica que funcione em tempo real, devendo ser estendido em cenários *outdoor*.

A integração desta abordagem em sistemas autónomos como veículos terrestres, aéreos ou marítimos, também permitiria validar o desempenho dos métodos de correspondências desenvolvidos, avaliando a informação tridimensional gerada pelos sistemas de visão desses veículos.

Apêndice A

Anexo A

A.1 Calibração dos parâmetros das câmaras

Começando pelo processo de calibração das câmaras que consiste na calibração dos parâmetros intrínsecos e extrínsecos referidos nos capítulos 3.3.1 e 3.3.2. Este processo é realizado recorrendo à *Camera Calibration Toolbox*[6] para Matlab[7], que é uma ferramenta para calibração de câmaras, sendo possível também testar os resultados da calibração. Na Figura A.1 é possível visualizar o menu principal da *toolbox*.

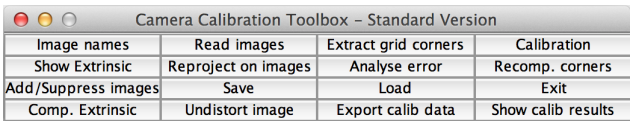


Figura A.1: Menu do *Camera Calibration Toolbox*.

Através do menu apresentado é possível executar o processo de calibração dos parâmetros intrínsecos ou extrínsecos dependendo das funções usadas, sendo que, nos subcapítulos que se seguem é explicado o procedimento de calibração.

A.1.1 Calibração dos parâmetros intrínsecos

O processo de calibração dos parâmetros intrínsecos da câmara pode ser decomposto nas seguintes etapas:

[6] http://www.vision.caltech.edu/bouguetj/calib_doc/, acedido em 01/03/2015
[7] http://www.mathworks.com/products/matlab/, acedido em 01/03/2015

- Obtenção de imagens;
- Carregamento das imagens;
- Extração dos cantos do xadrez;
- Calibração dos parâmetros intrínsecos;
- Análise do erro de calibração;
- Otimização dos parâmetros de calibração.

A primeira etapa é independente do uso da *toolbox*, uma vez que a aquisição de imagens podem ser feita recorrendo a qualquer software de aquisição, dependendo da câmara que se pretende calibrar. As imagens devem conter um alvo com um padrão de xadrez, com dimensões. O número de imagens necessárias para a calibração não é um valor fixo, sendo que se pode usar um número qualquer de imagens para calibrar, no entanto, é aconselhável ter no mínimo 15 imagens no resultado final da calibração. Assim sendo, é necessário obter mais que 15 imagens para o processo de calibração. As imagens devem ser obtidas de forma a que seja possível observar os fenómenos de distorção introduzidos pela lente, permitindo desta forma uma melhor correção desses fenómenos.

A etapa que se segue, consiste em carregar as imagens adquiridas no programa de calibração situadas num dado diretório, como se pode observar na Figura A.2.

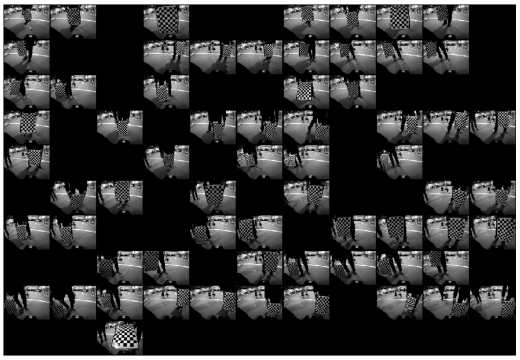


Figura A.2: Imagens usadas na calibração de uma das câmaras.

Posteriormente procede-se à extração dos cantos do alvo. Nesta etapa é necessário selecionar 4 pontos em cada imagem, sendo que estes pontos delimitam a área do xadrez considerada para a calibração da câmara. A sequência no qual a seleção dos pontos é

realizada deve ser sempre a mesma, sendo que o primeiro ponto deve ser marcado no canto superior esquerdo como se pode observar na Figura A.3(a). A partir das dimensões dos quadrados do xadrez, é possível selecionar os restantes cantos do alvo automaticamente como se pode observar na Figura A.3(b). O resultado da seleção automática dos restantes cantos do xadrez pode muitas das vezes não ser o exato, derivado à existência de muita distorção na imagem. Para reduzir o impacto da distorção, a *toolbox* permite a introdução de uma estimativa inicial para o coeficiente de distorção.

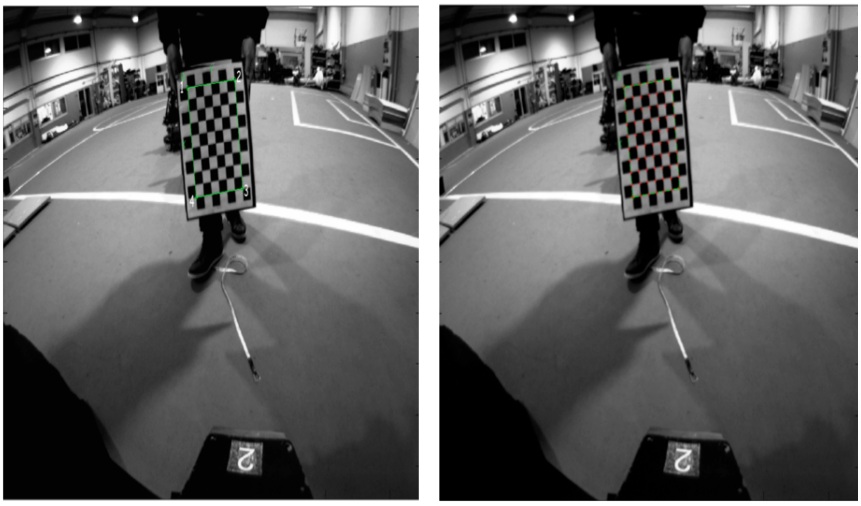


Figura A.3: (a) Seleção de 4 pontos que delimitam a área de calibração. (b) Seleção automática dos restantes quadrados do alvo.

Após a marcação do cantos em todas as imagens, executa-se a etapa da calibração, no qual, *toolbox* irá gerar os parâmetros intrínsecos da câmara composto pela distância focal, o centro ótico e os coeficientes de distorção radial e tangencial acrescentados de uma dada incerteza. Para além dos dados referidos, o processo de calibração também gera um valor de erro, que consiste no erro entre a projeção do ponto tridimensional do alvo nas imagens e os pontos detetados pela *toolbox*. Com base neste erro é possível analisar os resultados provenientes do processo da calibração.

Tendo os parâmetros de calibração gerados e o respetivo erro em pixeis é possível analisar a qualidade do processo da calibração com base no erro gerado. Ao executar o processo de análise do erro é possível visualizar num gráfico o erro em pixeis correspondente a cada ponto obtido no processo da calibração, como se pode observar na Figura

A.4. De forma a minimizar este erro, é possível remover as imagens que introduzem maus resultados na calibração, resultando numa calibração mais fiável. Cada vez que se remove uma imagem do processo de calibração, é necessário correr novamente a etapa da calibração para visualizar os resultados da ação tomada. O erro de projeção deverá ser menor a um píxel para que se obtenha uma calibração final fiável.

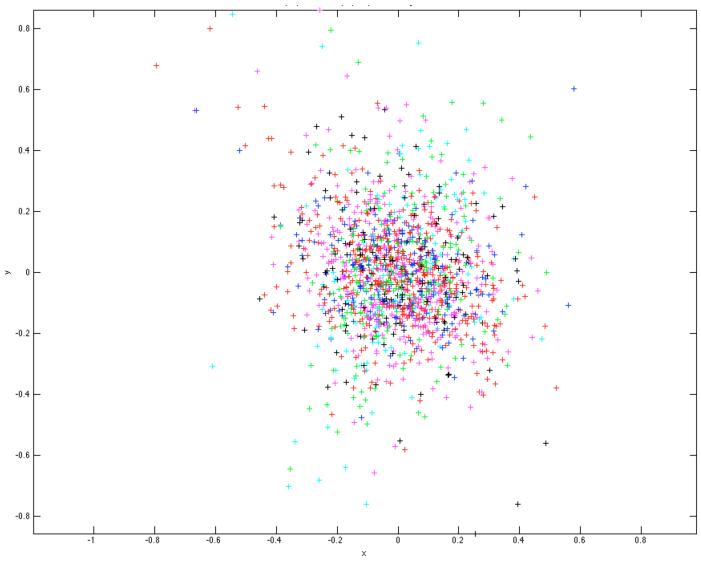


Figura A.4: Erro de projeção em píxeis.

O procedimento descrito anteriormente foi efetuado para as câmaras da plataforma experimental desta dissertação.

A.1.2 Calibração dos parâmetros extrínsecos

O processo da calibração dos parâmetros extrínsecos é mais rápido em comparação com a calibração dos parâmetros intrínsecos. Para este processo é necessário uma imagem que contenha um alvo, com o padrão de xadrez. Após a imagem ser adquirida, esta é carregada para a *toolbox* e de seguida executado o processo de calibração dos extrínsecos. Neste processo é replicado os passos feitos na etapa da calibração nos parâmetros intrínsecos, sendo que é necessário selecionar os quatro cantos que delimitam a areá do quadrado. Com a introdução das dimensões dos quadrículos do xadrez, a *toolbox* selecionará os restantes cantos do quadrado de forma automática, como se pode observar na Figura A.5.

O resultado da calibração dos extrínsecos de cada câmara é composto por um vetor

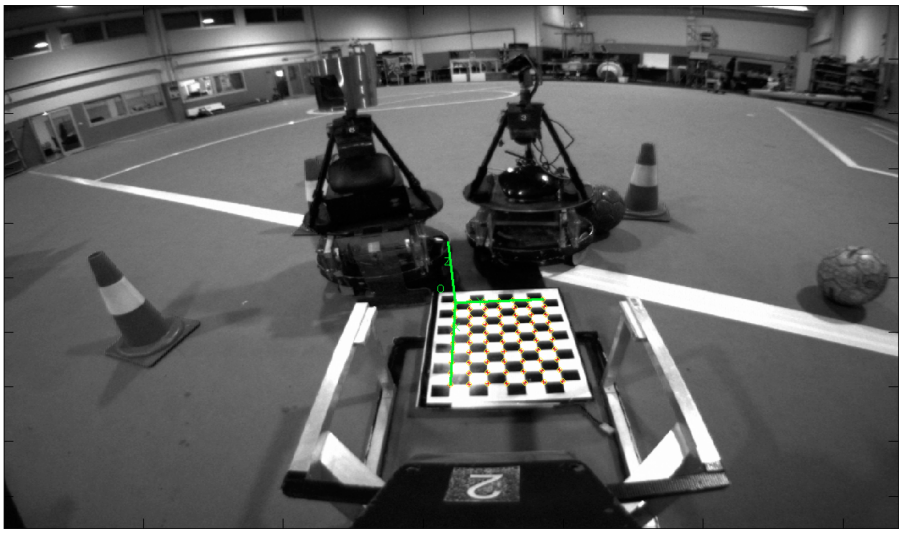


Figura A.5: Seleção dos cantos do xadrez para calibração dos parâmetros extrínsecos.

de translação(t), uma matriz de rotação(R), um vetor de rotação na notação Rodrigues e o erro em pixeis da calibração. Os parâmetros extrínsecos gerados têm em conta como ponto de referência (0.0) do xadrez, sendo que os eixos e o ponto "zero"do referencial estão ilustrados na Figura A.5.
Sempre que é alterada a posição da câmara, é necessário efetuar uma nova calibração.

Esta página foi intencionalmente deixada em branco.

Bibliografia

[1] K. M. Saipullah, "Comparison of feature extractors for real-time object detection on android smartphone," *Journal of Theoretical and Applied Information Technology*, pp. 135–142, 2013.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] K. Grauman and B. Leibe, *Visual object recognition*. Morgan & Claypool Publishers, 2010.

[4] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.

[5] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[6] A. Ferreira, "Sistema de modelização tridimensional de galerias subterrâneas," Master's thesis, Instituto Superior de Engenharia do Porto, 2011.

[7] R. Serra, "Sistema de visão stereo activo aplicado aos robôs do iseporto," Master's thesis, Instituto Superior de Engenharia do Porto, 2012.

[8] H. M. Silva, "Sistema de visão em tempo real para sistemas autónomos," Master's thesis, Instituto Superior Técnico, 2007.

[9] C. Sun, "A fast stereo matching method," in *Digital Image Computing: Techniques and Applications*. Citeseer, 1997, pp. 95–100.

[10] G. Pajares, M. Jesús, and P. J. Herrera, *Combining Stereovision Matching Constraints for Solving the Correspondence Problem*. INTECH Open Access Publisher, 2011.

[11] ——, *Combining Stereovision Matching Constraints for Solving the Correspondence Problem.* INTECH Open Access Publisher, 2011.

[12] P. Vellanki and M. Khambete, "Enhanced stereo matching technique using image gradient for improved search time," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 3, 2011.

[13] L. Nalpantidis, G. Sirakoulis, and A. Gasteratos, "Review of stereo matching algorithms for 3d vision," in *Proc. of the 16th International Symposium on Measurement and Control in Robotics (ISMCR 2007)*, 2007, pp. 116–124.

[14] K. Schauwecker, R. Klette, and A. Zell, "A new feature detector and stereo matching method for accurate high-performance sparse stereo matching," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.* IEEE, 2012, pp. 5171–5176.

[15] E. Vincent and R. Laganiere, "Matching feature points in stereo pairs: A comparative study of some matching strategies," *Machine Graphics and Vision*, vol. 10, no. 3, pp. 237–260, 2001.

[16] O. Veksler, "Dense features for semi-dense stereo correspondence," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 247–260, 2002.

[17] H. M. G. da Silva, "A probabilistic approach for stereo visual egomotion," Ph.D. dissertation, INSTITUTO SUPERIOR TÉCNICO, 2014.

[18] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[19] H. Sadeghi, P. Moallem, and S. A. Monadjemi, "Feature based dense stereo matching using dynamic programming and color," *International Journal of Computational Intelligence*, vol. 4, no. 3, pp. 179–186, 2008.

[20] O. Veksler, "Fast variable window for stereo correspondence using integral images," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1. IEEE, 2003, pp. I–556.

[21] J. Cai, "Fast stereo matching: coarser to finer with selective updating," in *Image and Vision Computing New Zealand 2007.* Image and Vision Computing New Zealand, 2007.

[22] S. Chan, Y. Wong, and J. Daniel, "Dense stereo correspondence based on recursive adaptive size multi-windowing," in *Proc. Image and Vision Computing New Zealand*, vol. 1, 2003, pp. 256–260.

[23] H. Hirschmuller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 9, pp. 1582–1599, 2009.

[24] M. E. Stivanello, E. S. Leal, N. Palluat, and M. R. Stemmer, "Dense correspondence with regional support for stereo vision systems," in *Graphics, Patterns and Images (SIBGRAPI), 2010 23rd SIBGRAPI Conference on*. IEEE, 2010, pp. 368–375.

[25] S. Mattoccia, "Improving the accuracy of fast dense stereo correspondence algorithms by enforcing local consistency of disparity fields," *3D Data Processing, Visualization and Transmission (3DPVT2010)*, 2010.

[26] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3. IEEE, 2006, pp. 15–18.

[27] Z.-F. Wang and Z.-G. Zheng, "A region based stereo matching algorithm using cooperative optimization," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[28] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 508–515.

[29] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 2, pp. 328–341, 2008.

[30] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, "High-quality real-time stereo using adaptive cost aggregation and dynamic programming," in *3D Data Processing, Visualization, and Transmission, Third International Symposium on*. IEEE, 2006, pp. 798–805.

[31] S. K. Gehrig, F. Eberli, and T. Meyer, "A real-time low-power stereo vision engine using semi-global matching," in *Computer Vision Systems*. Springer, 2009, pp. 134–143.

[32] M. Jahrer, M. Grabner, and H. Bischof, "Learned local descriptors for recognition and matching," in *Computer Vision Winter Workshop*, 2008, vol. 117.

[33] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 10, pp. 1615–1630, 2005.

[34] O. Miksik and K. Mikolajczyk, "Evaluation of local detectors and descriptors for fast feature matching," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, 2012, pp. 2681–2684.

[35] A. Schmidt, M. Kraft, M. Fularz, and Z. Domagala, "The comparison of point feature detectors and descriptors in the context of robot navigation," in *CLAWAR Workshop on Perception for Mobile Robot Autonomy (PEMRA'12), Poznan,(CD-ROM)*, 2012.

[36] M. Lu, "Fast implementation of scale invariant feature transform based on cuda," *Appl. Math*, vol. 7, no. 2L, pp. 717–722, 2013.

[37] K. Jeong and H. Moon, "Object detection using fast corner detector based on smartphone platforms," in *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on*, 2011, pp. 111–115.

[38] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[39] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge 2009," in *2th PASCAL Challenge Workshop*, 2009.

[40] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2130–2137.

[41] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," in *VLDB*, vol. 99, 1999, pp. 518–529.

[42] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 750–757.

[43] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2. IEEE, 2006, pp. 2161–2168.

[44] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." *VISAPP (1)*, vol. 2, 2009.

[45] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 4, pp. 323–344, 1987.

[46] J. C. Ribeiro, "Sistema de auto calibração visual para robots do iseporto, baseado em ekf," Master's thesis, Instituto Superior de Engenharia do Porto, 2013.

[47] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[48] E. Trucco and A. Verri, *Introductory techniques for 3-D computer vision*. Prentice Hall Englewood Cliffs, 1998, vol. 201.

[49] A. Fusiello, E. Trucco, and A. Verri, "A compact algorithm for rectification of stereo pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, 2000.

[50] S. Roma, N and Tomé, J, "A comparative analysis of cross-correlation matching algorithms using a pyramidal resolution approach," 2002.

[51] J. Lewis, "Fast normalized cross-correlation," in *Vision interface*, vol. 10, no. 1, 1995, pp. 120–123.

[52] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Aerospace/Defense Sensing, Simulation, and Controls*. International Society for Optics and Photonics, 2001, pp. 95–102.

[53] H. Silva, J. Almeida, L. Lima, A. Martins, E. Silva, and A. Patacho, "Lsavision real time framework architecture for mobile robotics," *Computational Modelling of objects represented in images Fundamentals, Methods and Applications CompIMAGE*, 2006.