**CENTRO UNIVERSITÁRIO FEI**
SAMUEL CORDEIRO DA COSTA

# UM SISTEMA DE VISÃO COMPUTACIONAL ESTEREOSCÓPICA PARA UM ROBÔ MÓVEL HUMANOIDE ATUANDO NO DOMÍNIO DO FUTEBOL DE ROBÔS DA ROBOCUP

São Bernardo do Campo
2017

SAMUEL CORDEIRO DA COSTA

# UM SISTEMA DE VISÃO COMPUTACIONAL ESTEREOSCÓPICA PARA UM ROBÔ MÓVEL HUMANOIDE ATUANDO NO DOMÍNIO DO FUTEBOL DE ROBÔS DA ROBOCUP

Dissertação de Mestrado apresentada ao Centro Universitário FEI para obtenção do título de Mestre em Engenharia Elétrica. Orientado pelo Prof. Dr. Reinaldo Augusto da Costa Bianchi

São Bernardo do Campo
2017

da Costa, Samuel Cordeiro.
UM SISTEMA DE VISÃO COMPUTACIONAL ESTEREOSCÓPICA PARA UM ROBÔ MÓVEL HUMANOIDE ATUANDO NO DOMÍNIO DO FUTEBOL DE ROBÔS DA ROBOCUP / Samuel Cordeiro da Costa. São Bernardo do Campo, 2017.
122 p. : il.

Dissertação - Centro Universitário FEI.
Orientador: Prof. Dr. Reinaldo Augusto da Costa Bianchi.

1. Robô Humanoide. 2. Visão Estéreo. 3. Reconhecimento de Objetos 3D. 4. Métodos Descritores. I. Bianchi, Reinaldo Augusto da Costa, orient. II. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

centro universitário FEI

# APRESENTAÇÃO DE DISSERTAÇÃO
# ATA DA BANCA EXAMINADORA

**Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica**

**Mestrado**

**PGE-10**

**Aluno:** Samuel Cordeiro da Costa **Matrícula:** 115125-7

**Título do Trabalho:** Um sistema de visão computacional estereoscópica para um robô móvel humanoide atuando no domínio do futebol de robôs da robocup.

**Área de Concentração:** Inteligência Artificial Aplicada à Automação

**Orientador:** Prof. Dr. Reinaldo Augusto da Costa Bianchi

**Data da realização da defesa:** 12/12/2017

ORIGINAL ASSINADA

**Avaliação da Banca Examinadora:**

São Bernardo do Campo, 12 / 12 / 2017.

| MEMBROS DA BANCA EXAMINADORA | |
|---|---|
| Prof. Dr. Reinaldo Augusto da Costa Bianchi | Ass.: _____ |
| Prof. Dr. Flavio Tonidandel | Ass.: _____ |
| Prof.ª Dr.ª Anna Helena Reali Costa | Ass.: _____ |

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒ REPROVADO ☐

**VERSÃO FINAL DA DISSERTAÇÃO**

**APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA EXAMINADORA**

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

Dedico este trabalho primeiramente à Deus pela oportunidade que Ele me concedeu. Aos meus pais e à minha esposa pela compreensão, incentivo e por estarem sempre do meu lado nos momentos que mais precisei.

## AGRADECIMENTOS

Agradeço primeiramente a Deus pela oportunidade que Ele me concedeu para escrever este trabalho.

Aos meus pais pela educação, incentivo, motivação e por sempre me conduzir à bons caminhos.

À minha esposa Jéssica pela compreensão, ajuda, incentivo e por estar sempre do meu lado nas horas mais difíceis.

Ao meu orientador Prof. Dr. Reinaldo Augusto da Costa Bianchi, pelos conselhos e ensinamentos que foram de grande proveito para concluir este trabalho. Por estar sempre presente, acessível e com grande disposição para que este trabalho fosse concluído da melhor forma possível.

Aos demais professores que contribuíram com ensinamentos ao longo do curso, sempre com muita transparência, entusiasmo e dedicação.

À toda equipe de robótica da FEI que sempre me auxiliou nos materiais que precisava, sendo que muitas vezes ficavam horas me ajudando à realizar os experimentos, sempre com muita boa vontade e disposição. Sem eles, com certeza tudo teria sido muito mais difícil.

Aos meus colegas de trabalho, engenheiro Allan Roberto Molto pelo companheirismo e por todo incentivo nos momentos difíceis; e ao Sr. Diego Rodrigues, pela ajuda em construir o suporte das câmeras, peça fundamental para o sistema de visão estéreo.

Ao meu amigo engenheiro Clóvis Pedroni Junior, pelo apoio, incentivo e disposição em auxiliar-me na finalização deste trabalho.

“O que planta e o que rega tem um só propósito e cada um será recompensado de acordo com seu próprio trabalho.”

I Coríntios 3:8

## RESUMO

Diante da constante evolução tecnológica apresentada nas últimas décadas, os robôs conquistaram um espaço muito importante na sociedade, tanto no desenvolvimento de tarefas industriais como na interação com os seres humanos. Com o intuito de estimular as pesquisas relacionadas aos robôs humanoides, surgiu a competição de futebol de robôs humanoides da RoboCup. Nesta competição, os robôs precisam atuar de forma autônoma, localizando os objetos a sua volta, tomando decisões e agindo de maneira adequada para ganhar os jogos. Pelas regras desta competição, os robôs precisam ter forma humanoide, e não podem utilizar nenhum sensor que não esteja relacionado com uma percepção humana, como lasers ou câmeras 3D. Neste contexto, a utilização de um sistema de visão estereoscópica agrega robustez e precisão no reconhecimento dos objetos envolvidos nessa competição. Apesar de sua importância, nota-se, no domínio da RoboCup, uma escassez de trabalhos que aplicam sistema de visão estéreo para reconhecimento de objetos 3D e detecção de suas respectivas distâncias, utilizando somente um par de câmeras *pin-hole*. Esta dissertação propõe, através de um sistema de visão estereoscópica, uma comparação de desempenho dos métodos descritores FPFH, SHOT e 3DSC no reconhecimento tridimensional de bolas de futebol com diferentes texturas, assim como oferece uma estimativa de sua respectiva distância, dentro do ambiente da competição RoboCup, na categoria *KidSize* da liga humanoide. Para tanto, técnicas como calibração estéreo para as câmeras, correspondência, triangulação e métodos descritores para reconhecimento de objetos 3D são abordados neste trabalho. A partir dos experimentos realizados, pôde-se concluir que o descritor FPFH foi o que obteve melhor desempenho, atingindo precisão de 100%, revocação de 78% e acurácia de 89% nos testes realizados.



Palavras-chave: Robô Humanoide, Visão Estéreo, RoboCup, Calibração Estéreo, Mapa de Disparidade, Reconhecimento de Objetos 3D, Métodos Descritores.

## ABSTRACT

In the face of the constant technological evolution presented in the last decades, robots have conquered a very important space in society, both in the development of industrial tasks and in the interaction with human beings. In order to stimulate research related to humanoid robots, the RoboCup humanoid robot soccer competition was born. In this competition, robots need to act autonomously, locating the objects around them, making decisions and acting in a proper way to win the games. By the rules of this competition, robots need to have a humanoid shape, and they can not use any sensor that is not related to human perception, like lasers or 3D cameras. In this context, the use of a stereoscopic vision system adds robustness and precision in the recognition of the objects involved in this competition. Despite its importance, in RoboCup's domain, there is a shortage of works that apply a stereo vision system for 3D object recognition and detection of their respective distances, using only a pair of pin-hole cameras. This dissertation proposes, through a stereoscopic vision system, a comparison of the performance of the descriptive methods FPFH, SHOT and 3DSC in the three-dimensional recognition of soccer balls with different textures, as well as an estimation of their respective distance, within the competition environment RoboCup, in the KidSize category of the humanoid league. For such, techniques such as stereo calibration for cameras, matching, triangulation and methods descriptors for 3D object recognition are addressed in this work. From the experiments performed, it was possible to conclude that the FPFH descriptor was the one that obtained the best performance, reaching 100% accuracy, 78% recall and 89% accuracy in the tests performed.



Keyword: Humanoid Robot, Stereo Vision, RoboCup, Stereo Calibration, Disparity Map, 3D Object Recognition, Descriptors Methods.

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| 3DSC | 3D Shape Context |
| CCD | Charge-Couple Device |
| CMOS | Complementary Metal-Oxide-Semiconductor |
| CPU | Central Processing Unit |
| FIFA | Fédération Internationale de Football Association |
| FPFH | Fast Point Feature Histograms |
| FPS | Frames por segundo |
| ISS | Intrinsic Shape Signatures |
| LSP | Local Salient Patterns |
| PC | Personal Computer |
| PCL | Point Cloud Library |
| PFH | Point Feature Histograms |
| PIXEL | Picture Element |
| RAM | Random Access Memory |
| RoPS | Rotational Projection Statistics |
| SHOT | Signature of Histograms of OrienTations |
| SI | Spin Images |
| SIFT | Scale Invariant Feature Transform |
| SPFH | Simplified Point Feature Histogram |
| TriSI | Tri-Spin-Image |
| USC | Unique Shape Contexts |

**SUMÁRIO**

# 1 INTRODUÇÃO

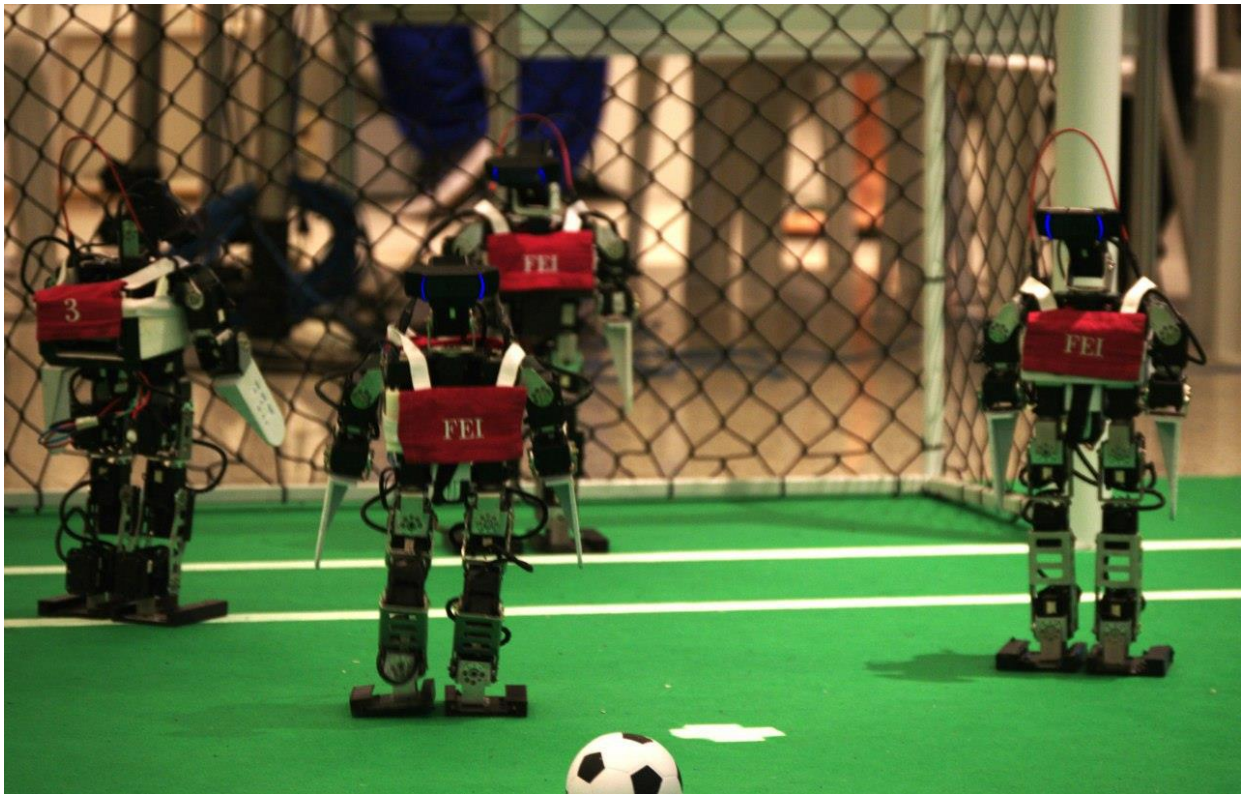
O primeiro robô humanoide articulado na história da civilização ocidental surgiu em 1495, quando o cientista, matemático, engenheiro, inventor, anatomista, pintor, escultor, arquiteto, botânico, poeta e músico Leonardo di Ser Piero da Vinci, ou simplesmente Leonardo da Vinci (1452-1519), apresentou em Milão, Itália, o projeto de um cavaleiro com a armadura utilizada pelo exército alemão-italiano. O robô foi projetado para se sentar, agitar seus braços e mover sua cabeça através de um pescoço flexível (ROSHEIM, 2006).

Após alguns séculos, em 1967, surgiu-se a primeira ocorrência de um robô bípede autônomo, quando Vukobratovic et al. apresentou o primeiro experimento com *dermato-skeletons*. O primeiro robô bípede controlado foi o WL-5, desenvolvido na Universidade Waseda em Tokyo, Japão, em 1972 (GARCIA, et al., 2007).

Em 1997, ocorreu o primeiro evento oficial de uma iniciativa cientifica internacional denominada RoboCup, cujo principal objetivo consiste em expandir o estado da arte da inteligência dos robôs. Desde o seu início, o desafio principal é apresentar um time de robôs capaz de vencer a seleção humana de futebol campeã do mundo até 2050. A competição é realizada anualmente em um diferente país escolhido para sediar o evento e, a cada nova edição, um conjunto com novas regras são divulgadas a fim de tornar a competição mais desafiadora. Uma das diversas categorias que compõem a competição RoboCup é a sub-liga de futebol de robôs humanoides *KidSize*, cujo ambiente consiste em um campo representado por uma superfície de grama artificial, com dimensões de 9 metros de comprimento por 6 metros de largura, em uma bola seguindo o padrão da FIFA tamanho 1 e robôs que seguem a semelhança física de um humano, ou seja, composto por duas pernas, dois braços e uma cabeça anexada ao tronco, onde a altura é restrita entre 40 cm e 90 cm. Os robôs devem utilizar apenas sensores que representam um sentido humano, ou seja, não podem adquirir informações provenientes de uma fonte que um ser humano não possui, como por exemplo, um sensor *Kinect* ou *LIDAR*, amplamente utilizados no segmento da visão computacional, por fazerem uso de tecnologias infravermelho e laser, respectivamente (RoboCup Federation, 2017).

O Centro Universitário FEI participa das competições da RoboCup na liga humanoides, categoria *KidSize*. A figura 1 ilustra o time RoboFEI-HT composto por seus 4 jogadores que disputaram a competição de 2016 e 2017.

Figura 1 – Time RoboFEI-HT da sub-liga *KidSize* da RoboCup




Fonte: Time RoboFEI-HT, 2016 e 2017.

## 1.1 MOTIVAÇÃO

Diante da regra que os robôs humanoides que atuam na sub-liga *KidSize* da RoboCup não devem utilizar sensores que não representam um sentido humano, o sistema de visão dos robôs, representados por uma ou duas câmeras, torna-se um dos sensores mais importantes, pois informações como sua respectiva localização em campo, reconhecimento de robôs adversários ou do mesmo time, estimativas de distâncias de objetos, e muitas outras informações são obtidas através dele. Até a competição de 2017, todas as equipes da sub-liga

*KidSize* utilizaram em seus robôs um sistema de visão representado por uma câmera, porém nenhuma equipe apresentou um sistema de visão estéreo.

O principal benefício de uma visão estéreo está na possibilidade da reconstrução tridimensional do cenário, permitindo o reconhecimento e a detecção da distância de objetos através de seu formato 3D, criando-se a independência de suas características particulares, como por exemplo, a textura da bola, que normalmente só é divulgada no início da competição.

## 1.2 OBJETIVO

O objetivo deste trabalho é comparar o desempenho de três algoritmos descritores 3D aplicados para o reconhecimento de bolas de futebol, com diferentes texturas, utilizadas nas competições da RoboCup, através de um sistema de visão estéreo adequado à um robô móvel humanoide atuando na sub-liga *KidSize* da competição *RoboCup*.

## 1.3 ORGANIZAÇÃO DO TRABALHO

Este primeiro capítulo apresentou uma introdução sobre a evolução da robótica, uma breve descrição da competição *RoboCup*, a definição do cenário real caracterizado pelo ambiente de uma competição da sub-liga *KidSize*, a motivação e o objetivo deste trabalho.

A seguir, este trabalho apresentará mais 5 capítulos, sendo o capítulo 2 uma abordagem sobre os conceitos fundamentais aplicados neste trabalho, o capítulo 3 é composto por estudo sobre os trabalhos correlacionados com o tema abordado, o capítulo 4 apresenta os detalhes do sistema de visão estéreo implementado, o capítulo 5 contém os experimentos e resultados obtidos, e, por fim, o capítulo 6 contempla a conclusão e as propostas futuras para este trabalho.

# 2 CONCEITOS FUNDAMENTAIS

Neste capítulo são abordados alguns conceitos teóricos utilizados neste trabalho. Os conceitos foram organizados em uma sequência progressiva para obter-se o reconhecimento de um objeto 3D, abordando o funcionamento das câmeras digitais, o funcionamento de um sistema de visão estéreo, a geometria epipolar, a calibração de um sistema estéreo, o processo de retificação, o processo de correspondência estéreo, o cálculo da triangulação e alguns métodos para reconhecimento de objetos 3D.

## 2.1 CÂMERAS DIGITAIS

Em um sistema de visão computadorizado, a aquisição da imagem é um passo fundamental para iniciar o processamento de imagens digitais. Para isso, é necessário um sistema capaz de capturar e digitalizar o sinal produzido por um sensor de imagem. Esse sistema é categorizado como aquisitor de imagens. Dois elementos são necessários para a aquisição de imagens digitais: um dispositivo físico sensível a uma banda do espectro de energia eletromagnética e que produza um sinal elétrico de saída proporcional a um nível de energia percebida (sensor); e um dispositivo para a conversão da saída elétrica do sensor para a forma digital (digitalizador) (GONZALES; WOODS, 2013).

O sensor de imagem captura a luminosidade através de uma área de detecção ativa e converte-a em sinais elétricos, permitindo que o digitalizador recrie a cena no formato de imagem que conhecemos. Existem duas principais tecnologias para os sensores de imagem: o *Charge-Couple Device* (CCD) e o *Complementary Metal Oxide on Silicon* (CMOS), porém os sensores CMOS são mais utilizados nas câmeras digitais do que os sensores CCD, devido ao baixo consumo de energia e o baixo custo de desenvolvimento (SZELISKI, 2010).

### 2.1.1 Geometria em Imageamento

Todo sistema de aquisição de imagem realiza algum tipo de transformação do espaço tridimensional real para o espaço bidimensional local. Encontrar estes parâmetros de

transformação é fundamental para descrever o sistema de aquisição (CYGANEK; SIEBERT, 2009).

Neste trabalho serão abordadas as transformações de translação, mudança de escala e rotação.

#### 2.1.1.1 Transformação de Translação

A transformação de **translação** consiste na mudança de um ponto de coordenadas (X, Y, Z) para uma nova posição, através de deslocamentos ($X_0$, $Y_0$, $Z_0$). Esta operação pode ser feita através de uma simples soma (1) (GONZALES; WOODS, 2013).

$$\begin{aligned} X' &= X + X_0 \\ Y' &= Y + Y_0 \\ Z' &= Z + Z_0 \end{aligned} \quad \textbf{(1)}$$

É comum a aplicação de várias transformações consecutivamente. Para isso, é possível representar as transformações no formato de matrizes, onde pode-se concatenar diversas transformações de uma só vez, aplicando-se uma multiplicação de matrizes. Desta forma, a transformada de translação pode ser representada por (GONZALES; WOODS, 2013):

$$p' = Tp \ , \quad \textbf{(2)}$$

onde, p' é a matriz que contém as coordenadas do ponto resultante, p é a matriz que contém as coordenadas do ponto original e T é a matriz de translação (3) (GONZALES; WOODS, 2013):

$$T = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \textbf{(3)}$$

onde, $X_0$, $Y_0$ e $Z_0$ representam o deslocamento em suas respectivas coordenadas.

2.1.1.2 Transformação de mudança de escala

Assim como na transformação de translação, a transformação de **mudança de escala** pode ser aplicada através de uma multiplicação de matrizes. A matriz de mudança de escala é definida por (GONZALES; WOODS, 2013):

$$S = \begin{bmatrix} S_X & 0 & 0 & 0 \\ 0 & S_Y & 0 & 0 \\ 0 & 0 & S_Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \textbf{(4)}$$

onde, $S_X$, $S_Y$ e $S_Z$ representam o fator de escala em suas respectivas coordenadas.

2.1.1.3 Transformação de Rotação

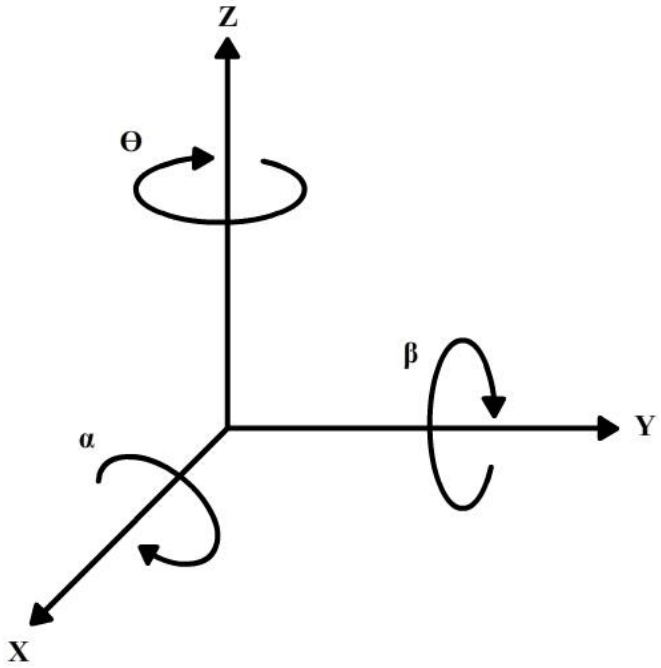
A transformação de **rotação** possui uma representação para cada eixo onde será aplicada a rotação. Desta forma, a rotação em torno do eixo de coordenadas Z de um ângulo Θ é feita através da matriz apresentada na equação (5). Da mesma forma, a rotação em torno do eixo de coordenadas X de um ângulo α é feita através da matriz apresentada na equação (6) e a rotação em torno do eixo de coordenadas Y de um ângulo β é feita através da matriz apresentada na equação (7). A figura 2 ilustra estas rotações em seus respectivos eixos (GONZALES; WOODS, 2013).

$$R(\Theta) = \begin{bmatrix} \cos(\Theta) & sen(\Theta) & 0 & 0 \\ -sen(\Theta) & \cos(\Theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(5)}$$

$$R(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & sen(\alpha) & 0 \\ 0 & -sen(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(6)}$$

$$R(\beta) = \begin{bmatrix} \cos(\beta) & 0 & -sen(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ sen(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(7)}$$

Figura 2 - Rotação de um ponto em torno dos eixos X, Y e Z



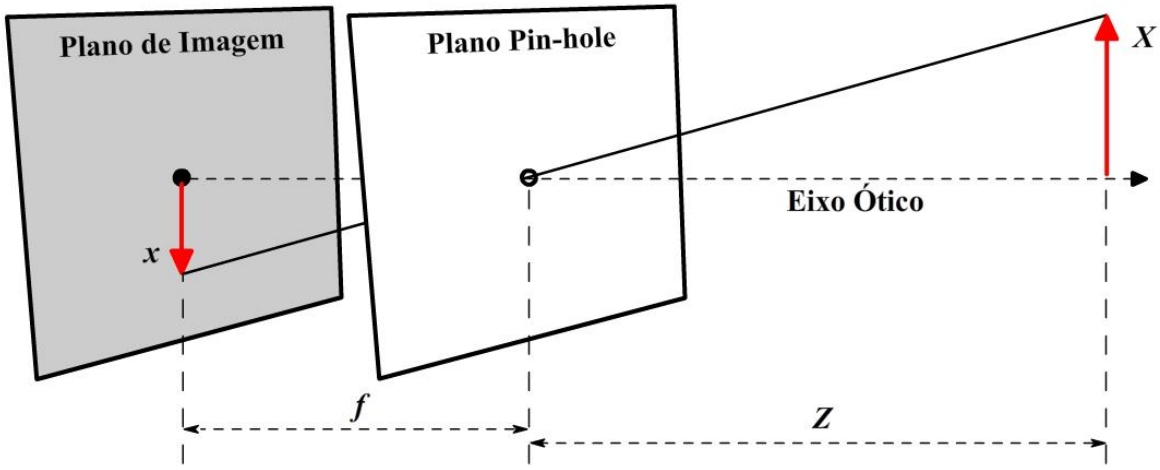

Fonte: Gonzales; Woods, 2013

### 2.1.2 **Modelo de Câmera *Pin-hole***

Neste trabalho será abordado o sistema de aquisição de imagem através de uma câmera *pin-hole*, cuja característica é plotar uma cena do espaço tridimensional em um plano de imagem utilizando um orifício de maneira que qualquer raio de luz emitido ou refletido da cena, antes de chegar no plano de imagem, atravesse este orifício. O tamanho da imagem plotada no plano de imagem depende da distância entre o orifício (*pin-hole*) e o próprio plano de imagem. Esta distância é conhecida como distância focal. A figura 3 ilustra este modelo,

onde *f* é a distância focal da câmera, *Z* é a distância entre a câmera e o objeto, *X* é o tamanho do objeto e *x* é o objeto representado no plano de imagem (BRADSKI; KAEHLER, 2008).

Figura 3 - Diagrama de câmera *pin-hole*




Fonte: Bradski; Kaehler, 2008

Por semelhança de triângulos, pode-se deduzir a equação (8):

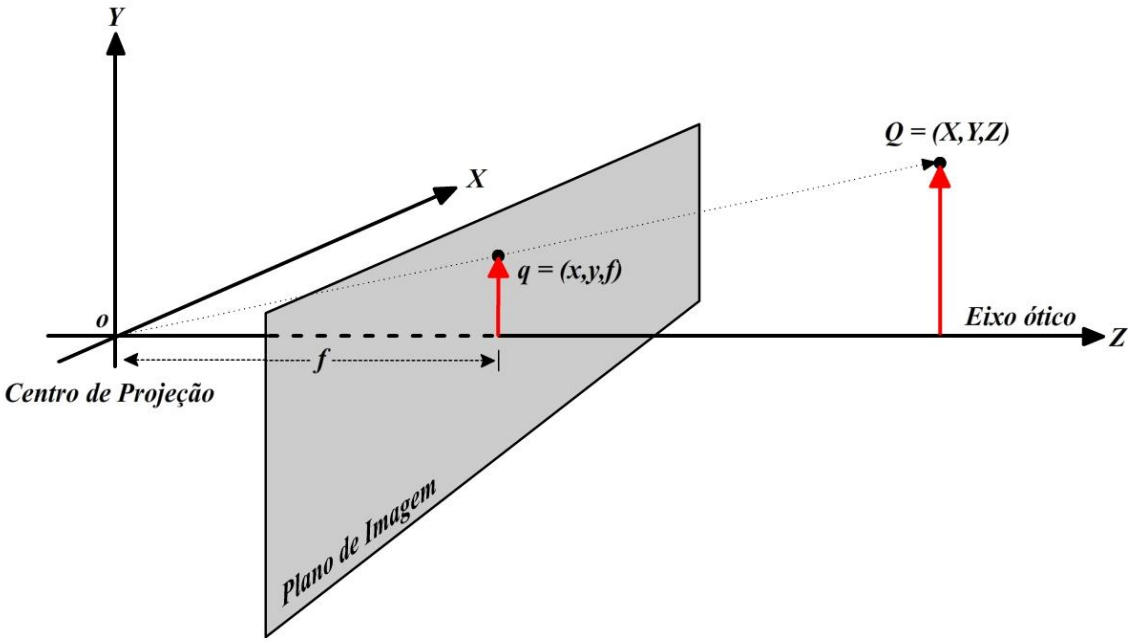
$$-\frac{x}{f} = \frac{X}{Z}$$

$$-x = f\frac{X}{Z} \qquad \textbf{(8)}$$

A fim de facilitar as operações matemáticas, o modelo *pin-hole* pode ser representado de uma outra forma equivalente. A principal diferença é o deslocamento do plano *pin-hole*, de maneira que seu orifício se torna o centro de projeção; e o deslocamento do plano de imagem, ficando à frente do centro de projeção na distância *f*, de maneira que o objeto plotado permaneça com o mesmo tamanho, porém não mais invertido. A figura 4 ilustra este novo arranjo do modelo *pin-hole*. Aplicando novamente a semelhança de triângulos, obtém-se a equação (9), muito semelhante à equação (8), porém sem o sinal negativo, pois a representação do objeto deixou de ser invertida (BRADSKI; KAEHLER, 2008).

$$x = f\frac{X}{Z} \qquad \textbf{(9)}$$

Figura 4 - Segundo diagrama de câmera *pin-hole* para simplificações matemáticas




Fonte: Bradski; Kaehler, 2008

O ponto *Q* é projetado no plano de imagem por um feixe passando pelo Centro de Projeção. O ponto *q* é a intersecção deste feixe com o plano de imagem, representando o objeto com as mesmas características da figura 3, porém na frente do *pin-hole*, facilitando assim as operações matemáticas.

Para relacionar os pontos do mundo tridimensional com os pontos representados no plano de imagem bidimensional, são necessários dois sistemas de coordenadas: o sistema de coordenadas externas, representado por *W* e o sistema de coordenadas da câmera representado por *C*. Estes dois sistemas de coordenadas são relacionados através de transformação de translação (matriz *T* descrita na seção 2.1.1.1) e rotação (matrizes *R* descritas na seção

2.1.1.3), onde o sistema de coordenadas do mundo é independente do posicionamento e dos parâmetros da câmera (CYGANEK; SIEBERT, 2009).

Utilizando estes dois sistemas de coordenadas na mesma cena, conforme ilustrado na figura 5, tem-se o ponto do centro de projeção ($O_C$) da câmera, representando a origem do sistema de coordenadas da câmera, composto pelos eixos ($X_C$, $Y_C$, $Z_C$) e o ponto $O_W$, representando a origem do sistema de coordenadas do mundo, composto pelos eixos ($X_W$, $Y_W$, $Z_W$). O plano de imagem, denominado **Π**, foi fragmentado em diversos elementos retangulares chamados de *pixel* que, dentro de uma implementação de uma câmera eletrônica, estes *pixels* representam locais discretos sensitivos à raios luminosos e são endereçados através de um par de coordenadas (CYGANEK; SIEBERT, 2009).

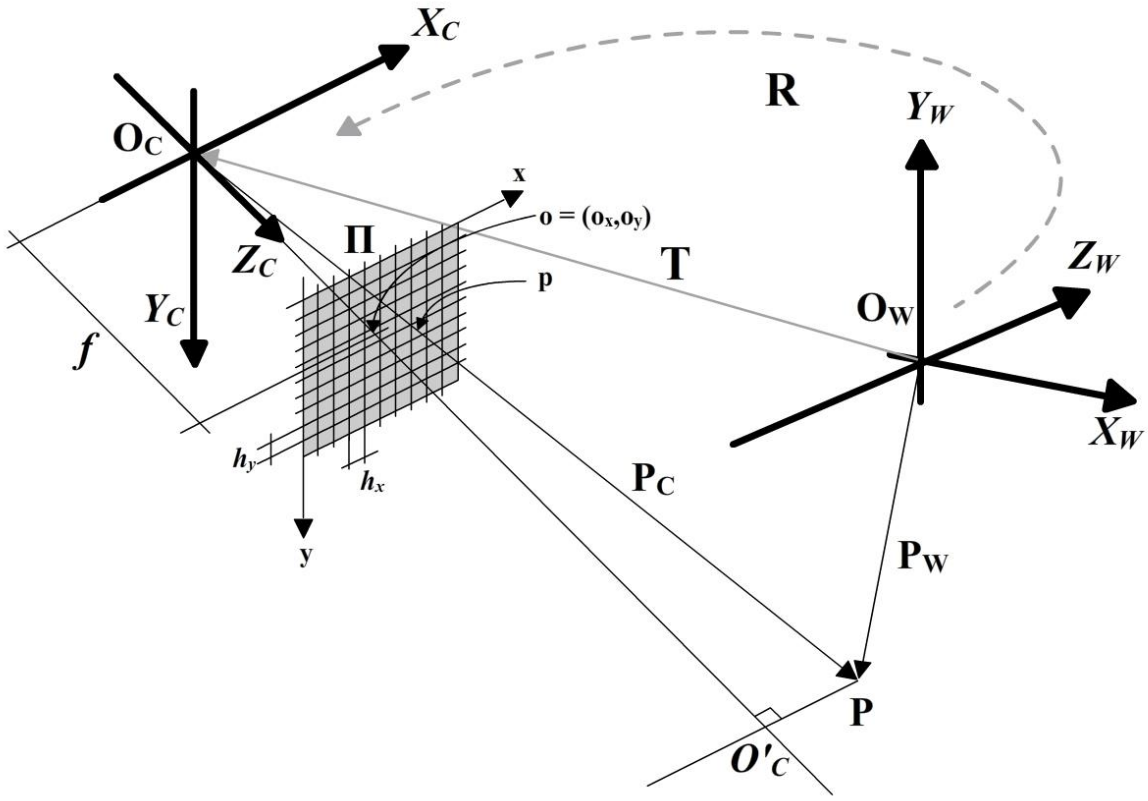
A projeção do ponto $O_C$ no plano **Π** na direção $Z_C$ determina o ponto principal da coordenada local ($o_x$, $o_y$). O eixo principal é a reta entre $O_C$ e $O'_C$. A distância entre o ponto principal $p$, no plano **Π**, e a origem do sistema de coordenadas da câmera $O_C$ é a distância focal, representada por $f$. Por fim, o tamanho de cada *pixel* é representado pelas dimensões $h_x$ e $h_y$, sendo que a relação entre a sua altura e a sua largura ($h_y/h_x$) é chamada de *aspect ratio* (proporção da tela, em tradução livre do inglês) (CYGANEK; SIEBERT, 2009).

A localização do ponto $P$ no espaço tridimensional depende de qual sistema de coordenada que está sendo observada. No sistema de coordenada da câmera, o ponto $P$ é representado por $P_C$. Já no sistema de coordenada do mundo, sua representação é dada por $P_W$. O ponto $p$ é uma projeção do ponto $P$ no plano **Π**. As coordenadas do ponto $P$ e $p$ no sistema de coordenadas da câmera, são representadas pelas equações (10) e (11) (CYGANEK; SIEBERT, 2009).

$$P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \textbf{(10)}$$

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \textbf{(11)}$$

Figura 5 - Modelo *pin-hole* com representação em sistemas de coordenadas *W* e *C*




Fonte: Cyganek; Siebert, 2009

Com o eixo ótico perpendicular ao plano de imagem, obtêm-se através de semelhança de triângulos as equações (12) e (13). Como o plano de imagem está posicionado na distância focal *f* no eixo ótico, pode-se deduzir a equação (14). Estas três equações compõem a base do modelo de câmera *pin-hole* (CYGANEK; SIEBERT, 2009).

$$x = f\frac{X}{Z} \tag{12}$$

$$y = f\frac{Y}{Z} \tag{13}$$

$$z = f \tag{14}$$

O modelo de câmera *pin-hole* pode ser definido através de dois conjuntos de parâmetros: parâmetros extrínsecos e parâmetros intrínsecos.

#### 2.1.2.1 Parâmetros Extrínsecos

Os parâmetros extrínsecos de uma câmera consistem nos parâmetros geométricos que permitem a mudança do sistema de coordenadas da câmera para o sistema de coordenadas do mundo e vice-versa (CYGANEK; SIEBERT, 2009).

A conversão do sistema de coordenadas da câmera *C* para o sistema de coordenadas do mundo *W* pode ser realizada através de uma transformação de translação **T** e de uma transformação de rotação **R**. O vetor de translação **T** descreve uma mudança na posição dos centros de coordenadas $O_C$ e $O_W$. A rotação é realizada em torno dos eixos de cada sistema de coordenadas. Sendo assim, para um ponto *P* no espaço tridimensional, os sistemas de coordenadas *C* e *W* são conectados através da equação (15) (CYGANEK; SIEBERT, 2009):

$$P_C = R(P_W - T) \ , \tag{15}$$

onde, $P_C$ representa o ponto *P* no sistema de coordenadas *C*, $P_W$ representa o mesmo ponto no sistema de coordenadas *W*, *R* representa a matriz de transformação de rotação nas três coordenadas e *T* representa a matriz de translação entre as origens dos dois sistemas de coordenadas.

#### 2.1.2.2 Parâmetros Intrínsecos

Os parâmetros intrínsecos de uma câmera *pin-hole* consistem na distância entre o plano da câmera e o centro do sistema de coordenadas da câmera, isto é, a distância focal dada por *f*; na correlação entre o sistema de coordenadas da câmera e o sistema de coordenadas da imagem, levando em consideração as dimensões físicas dos *pixels*; e nas distorções geométricas que surgem devido aos parâmetros físicos dos elementos óticos da câmera (CYGANEK; SIEBERT, 2009).

Considerando que a origem da imagem constitui um ponto central *o*, composto por ($o_x$,$o_y$), e as dimensões físicas de cada *pixel* é dada por $h_x$ e $h_y$, uma relação entre as coordenadas da imagem ($x_u$,$y_u$) e as coordenadas da câmera ($x$,$y$) pode ser estabelecida conforme as equações (16) e (17) (CYGANEK; SIEBERT, 2009):

$$x = (x_u - o_x)h_x \ , \tag{16}$$

$$y = (y_u - o_y)h_y \ , \tag{17}$$

onde, um ponto ($x$,$y$) está relacionado ao sistema de coordenadas da câmera *C*, enquanto ($x_u$,$y_u$) e ($o_x$,$o_y$) estão relacionados ao sistema de um plano de câmera local. Desta forma, a origem deste plano de câmera ($x_u$,$y_u$) = (*0,0*) é transformada para o ponto (-$o_x h_x$,-$o_x h_x$) do sistema de coordenadas *C*. Portanto, a matriz dos parâmetros intrínsecos pode ser representada conforme a equação (18) (CYGANEK; SIEBERT, 2009):

$$M = \begin{bmatrix} \frac{f}{h_x} & 0 & o_x \\ 0 & \frac{f}{h_y} & o_y \\ 0 & 0 & 1 \end{bmatrix} . \tag{18}$$

As distorções geométricas encontradas em sistemas óticos surgem principalmente por dois fatores: a ausência de linearidade dos elementos óticos presentes na câmera; e a dependência dos parâmetros ópticos com o comprimento de onda da luz incidente. O primeiro fator está relacionado às aberrações esférica, coma, astigmatismo e curvatura de um campo de visão e distorções. O segundo fator está relacionado com a aberração cromática. Na maioria das situações práticas, pode-se modelar estes fenômenos como distorções radiais, onde a intensidade é mais elevada para os pontos mais distantes do centro da imagem (CYGANEK; SIEBERT, 2009).

Desta forma, a distorção radial pode ser representada pelos primeiros termos de uma expansão da série de Taylor arredor de $r = 0$. Geralmente são utilizados os dois primeiros termos, conhecidos como $k_1$ e $k_2$. Para câmeras com distorções elevadas, tais como lentes

olho de peixe, utiliza-se o terceiro termo $k_3$. Esta correção pode ser descrita pelas equações polinomiais (19) e (20) (BRADSKI; KAEHLER, 2008):

$$x_{corrigido} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \ , \quad \textbf{(19)}$$

$$y_{corrigido} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \ . \quad \textbf{(20)}$$

## 2.2 SISTEMA DE VISÃO COMPUTACIONAL ESTEREOSCÓPICA

Em 1838, o filósofo Charles Wheatstone publicou um artigo na "*Philosophical Transactions*" da *Royal Society* de Londres (WHEATSTONE, 1838), onde descreveu seu estudo sobre o fenômeno da Visão Binocular. Neste experimento, Wheatstone observou que quando focava em um objeto à uma distância suficiente para que os eixos óticos de ambos os olhos fossem sensivelmente paralelos, não haviam diferenças entre as perspectivas geradas por cada olho (direito e esquerdo), ou seja, as perspectivas geradas por ambos os olhos eram semelhantes. Porém, quando se aproximava este objeto a uma distância que obrigatoriamente os eixos óticos se convergiam, notava-se uma diferença entre as perspectivas geradas por cada olho. Esta observação foi de grande importância para a percepção de profundidade de um ponto na cena, ocasionando um aumento no interesse de novas pesquisas nos métodos tridimensionais que, no século XX, gerou-se uma nova linha de estudo chamada Visão Computacional Estereoscópica.

Sabe-se que uma única imagem oferece muitas informações a serem exploradas, como formatos geométricos, bordas, texturas, luminosidade e cores. Porém, a informação de profundidade de um ponto na cena não é possível ser obtida, sendo necessário ao menos duas imagens do mesmo cenário para que, através de cálculos trigonométricos, torna-se possível encontrar a profundidade. Esta é uma das razões no qual muitos animais possuem ao menos dois olhos, permitindo desviar-se de obstáculos e também facilitando a estratégia de caça ou de fuga. Animais que possuem apenas um olho focando a mesma cena, como por exemplo algumas aves e insetos, conseguem obter a profundidade de um ponto através de um deslocamento com a cabeça, gerando ao menos duas imagens do mesmo cenário, de diferentes ângulos (FORSYTH; PONCE, 2003).

Neste trabalho, será abordado um sistema de visão estéreo binocular, ou seja, as duas imagens referentes à mesma cena serão geradas simultaneamente através de duas câmeras.

### 2.2.1 **Geometria Epipolar**

A geometria epipolar consiste na correlação geométrica de duas imagens de uma mesma cena geradas simultaneamente por duas câmeras (uma para cada imagem) (BRADSKI; KAEHLER, 2008). As câmeras abordadas neste trabalho são do modelo *pin-hole* (tópico 2.1.2).

Para cada câmera, existe um centro de projeção, $O_l$ e $O_r$, e um par de planos de imagens correspondente, $\Pi_l$ e $\Pi_r$. O ponto $P$ no mundo tridimensional possui uma projeção dentro dos planos gerados pelas duas câmeras, sendo $p_l$ para câmera esquerda e $p_r$ para a câmera direita. Na correlação entre os dois planos de projeção, surgem novos pontos de interesse chamados de epipolos (BRADSKI; KAEHLER, 2008).

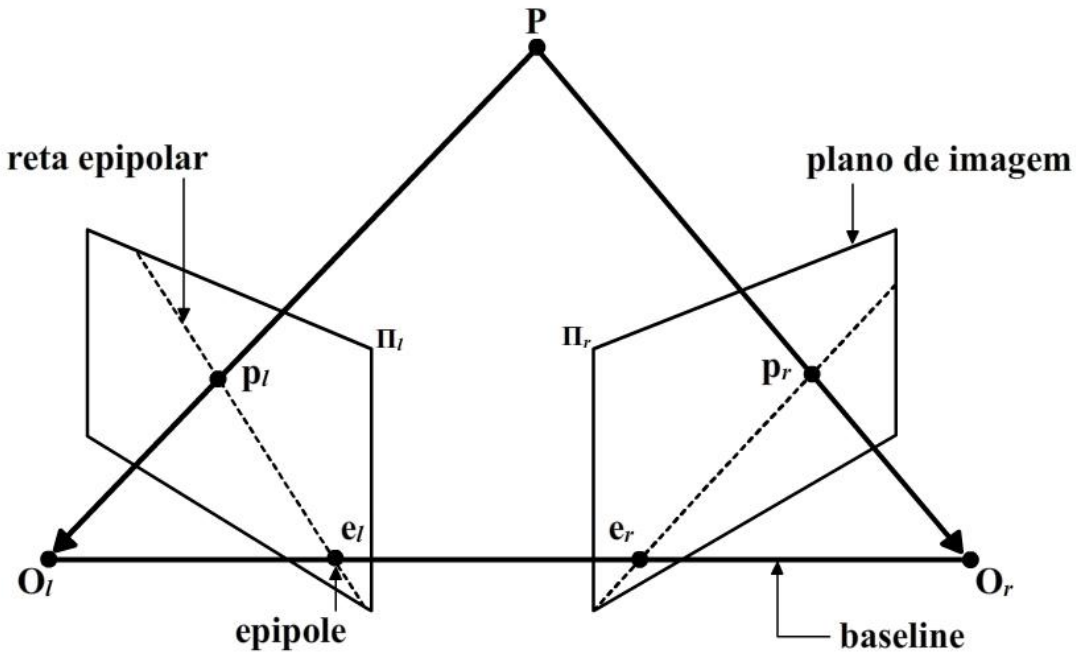
O epipolo é o ponto de intersecção da reta que une os centros de projeção das câmeras (a *baseline*) com o plano de imagem (HARTLEY; ZISSERMAN, 2003). Um epipolo $e_l$ no plano de imagem $\Pi_l$ é definido como a imagem do centro de projeção da outra câmera $O_r$, assim como um epipolo $e_r$ no plano de imagem $\Pi_r$ é definido como a imagem do centro de projeção da câmera $O_l$. O plano no espaço formado pelo ponto $P$ e pelos centros de projeções $O_l$ e $O_r$, passando pelos dois epipolos $e_l$ e $e_r$, é chamado de plano epipolar. As retas que unem $p_le_l$ e $p_re_r$ são chamadas de retas epipolares (BRADSKI; KAEHLER, 2008). A figura 6 ilustra a representação deste cenário.

Portanto, todo ponto do espaço tridimensional projetado em uma câmera possui um plano epipolar que intersecciona cada imagem em uma reta epipolar, sendo que este ponto projetado em uma imagem terá seu ponto correspondente na outra imagem somente ao longo da reta epipolar correspondente. Esta característica é chamada de restrição epipolar (BRADSKI; KAEHLER, 2008).

A restrição epipolar indica que a possível busca bidimensional para correspondência em duas imagens torna-se uma busca unidimensional ao longo das linhas epipolares uma vez que sabemos a geometria epipolar do sistema estéreo. Isto não é apenas uma grande simplificação computacional, mas também permite descartar diversos pontos que poderiam levar a falsas correspondências (BRADSKI; KAEHLER, 2008).

Se a reta epipolar pode ser encontrada para a maioria dos pontos de um plano de projeção, aplicando-se algumas transformações, é possível relacionar o plano de imagem da câmera da direita com o plano de imagem da câmera da esquerda. Estas transformações podem ser representadas através de duas importantes matrizes, conhecidas como matriz essencial e matriz fundamental.

Figura 6 - Representação básica da geometria epipolar



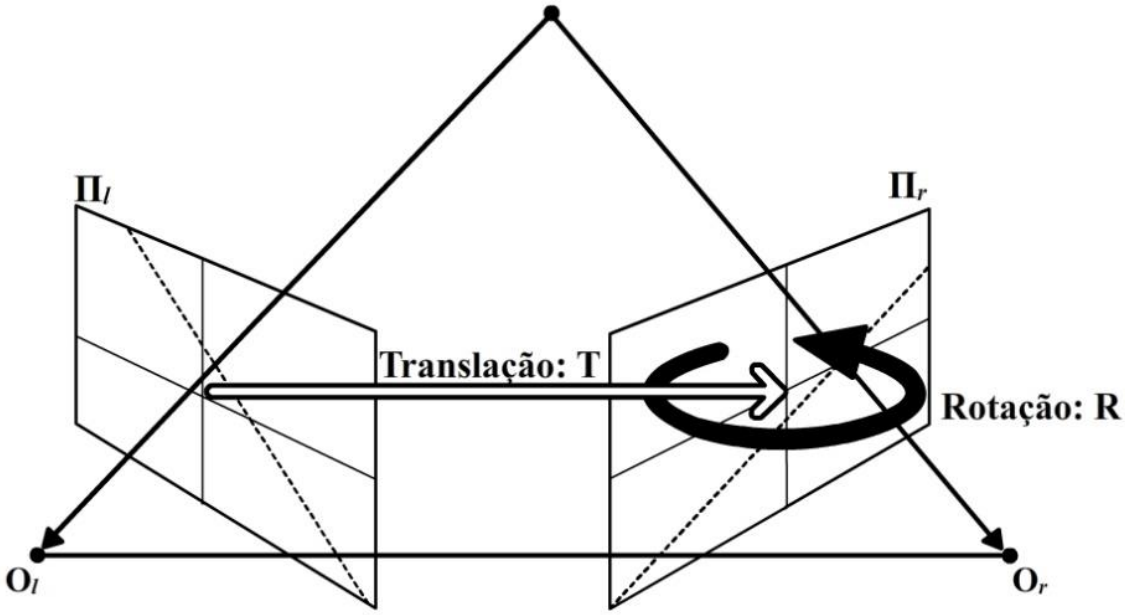

Fonte: Bradski; Kaehler, 2008

2.2.1.1 Matriz Essencial

A matriz essencial determina a relação entre os planos de imagem gerados pelas câmeras do sistema estéreo. Dado um ponto *P* no espaço tridimensional e adotando a origem do sistema de coordenadas como $O_l$, é possível através do ponto $P_l$ (representação do ponto *P* na origem da câmera esquerda) encontrar o $P_r$ (representação do ponto *P* na câmera da direita), aplicando-se as transformações de translação e rotação. A figura 7 ilustra estas transformações,

lembrando que a transformação de translação também pode ser representada por um vetor de descolamento contendo $X_0$, $Y_0$ e $Z_0$ (ver tópico 2.1.1.1) (BRADSKI; KAEHLER, 2008).

Figura 7 - Representação geométrica da matriz essencial




Fonte: Bradski; Kaehler, 2008

Onde, **R** representa a matriz de rotação e **T** representa o vetor de deslocamento de $O_l$ a $O_r$, permitindo-se obter $P_r$ através da equação (21).

$$P_r = R(P_l - T) \quad \textbf{(21)}$$

Sabendo-se que o plano epipolar contém os vetores $P_l$ e **T**, e que o vetor resultante do produto vetorial $(P_l \times T)$ é perpendicular entre eles, pode-se substituir estes valores na equação geral do plano (22), obtendo a equação (23) (BRADSKI; KAEHLER, 2008):

$$(x - a).n = 0 \ , \quad \textbf{(22)}$$

$$(P_l - T)^T.(T \times P_l) = 0 \ . \quad \textbf{(23)}$$

Pode-se reescrever a equação (21) na seguinte forma:

$$R^T P_r = (P_l - T) \ . \tag{24}$$

Substituindo a equação (24) em (23), tem-se a equação (25):

$$(R^T P_r)^T . (T \times P_l) = 0 \ . \tag{25}$$

É possível reescrever o produto vetorial $(T \times P_l)$ como uma multiplicação de matriz, conforme representado pela equação (26):

$$T \times P_l = SP_l \ \rightarrow \ S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} . \tag{26}$$

Substituindo a equação (26) em (25), tem-se a equação (27):

$$P_r{}^T \boldsymbol{RS} P_l = 0 \ , \tag{27}$$

onde, o produto $\boldsymbol{RS}$ representa a definição da matriz essencial **E**. Dessa forma, pode-se reescrever a equação (27) na forma compacta:

$$P_r{}^T \boldsymbol{E} P_l = 0 \ , \tag{28}$$

entretanto, $P_r$ e $P_l$ estão representados como pontos no plano de coordenadas da câmera. Para transformá-los em pontos projetados nos planos de imagens ($p_r$ e $p_l$), é necessário aplicar as

equações (12), (13) e (14) apresentadas no tópico 2.1.2. Com os pontos $P_r$ e $P_l$ convertidos em pontos nos planos de imagens, surge-se a equação (29) (CYGANEK; SIEBERT, 2009):

$$p_r{}^T \boldsymbol{E} p_l = 0 \ . \tag{29}$$

2.2.1.2 Matriz Fundamental

A matriz fundamental **F** possui características semelhantes à matriz essencial **E**, porém, relaciona pontos no plano de imagem de uma câmera em coordenadas de imagem (*pixels*) com pontos no plano de imagem da outra câmera, também em coordenadas de imagem. Para isso, são considerados os parâmetros intrínsecos das câmeras (BRADSKI; KAEHLER, 2008).

Um ponto $p$ nas coordenadas de imagem (*pixels*), pode ser representado como $q$ ao acrescentar os parâmetros intrínsecos da câmera, conforme a equação (30):

$$q = Mp \ , \tag{30}$$

onde **M** representa a matriz dos parâmetros intrínsecos da câmera definida na equação (18).

A equação (30) pode ser reescrita em função de $p$:

$$p = M^{-1} q \ . \tag{31}$$

Substituindo a equação (31) na equação da matriz essencial (29), tem-se a equação (32):

$$q_r{}^T (M_r{}^{-1})^T \boldsymbol{E} M_l{}^{-1} q_l = 0 \ . \tag{32}$$

Desta forma, pode-se definir a matriz fundamental **F** conforme a equação (33):

$$\boldsymbol{F} = (M_r{}^{-1})^T \boldsymbol{E} M_l{}^{-1} \ . \quad \textbf{(33)}$$

Substituindo a equação (33) em (32), tem-se a equação (34):

$$q_r{}^T \boldsymbol{F} q_l = 0 \ . \quad \textbf{(34)}$$

### 2.2.2 **Calibração Estéreo**

A calibração estéreo consiste no relacionamento geométrico entre duas câmeras no espaço. Para isso, é necessário encontrar a matriz de rotação **R** que descreve a rotação relativa entre os sistemas de coordenadas das duas câmeras, e o vetor de translação **T** que descreve a translação entre os centros das duas câmeras (CYGANEK; SIEBERT, 2009).

Após encontrado os parâmetros extrínsecos (ver tópico 2.1.2.1) das duas câmeras individualmente, tem-se: $R_l$ e $T_l$ para a câmera da esquerda, e $R_r$ e $T_r$ para a câmera da direita. Utilizando a equação (15), cuja finalidade é relacionar o sistema de coordenadas do mundo com o sistema de coordenadas da câmera, obtém-se as equações (35) e (36):

$$P_l = R_l(P_W - T_l) \ , \quad \textbf{(35)}$$

$$P_r = R_r(P_W - T_r) \ , \quad \textbf{(36)}$$

onde $P_W$ representa um ponto no espaço tridimensional (sistema de coordenadas do mundo), $P_l$ e $P_r$ são coordenadas deste ponto no sistema de coordenadas das câmeras esquerda e direita, respectivamente, $R_l$ e $R_r$ são as matrizes de rotações e $T_l$ e $T_r$ são os vetores de translação entre um sistema de coordenadas do mundo e os sistemas de coordenadas das respectivas câmeras (CYGANEK; SIEBERT, 2009).

Os pontos $P_l$ e $P_r$ também estão deduzidos no sistema estéreo através da equação (21), onde é possível relacionar as equações (35) e (36), obtendo-se a equação (37):

**(37)**

$$P_r = R_r R_l^T [P_l - R_l(T_r - T_l)] \ .$$

Comparando a equação (21) com a equação (37), é possível deduzir as equações (38) e (39):

$$\boldsymbol{R} = R_r R_l^T \ , \tag{38}$$

$$\boldsymbol{T} = R_l(T_r - T_l) \ , \tag{39}$$

onde **R** e **T** são as matrizes de calibração de um sistema estéreo.

### 2.2.3 **Retificação Estéreo**

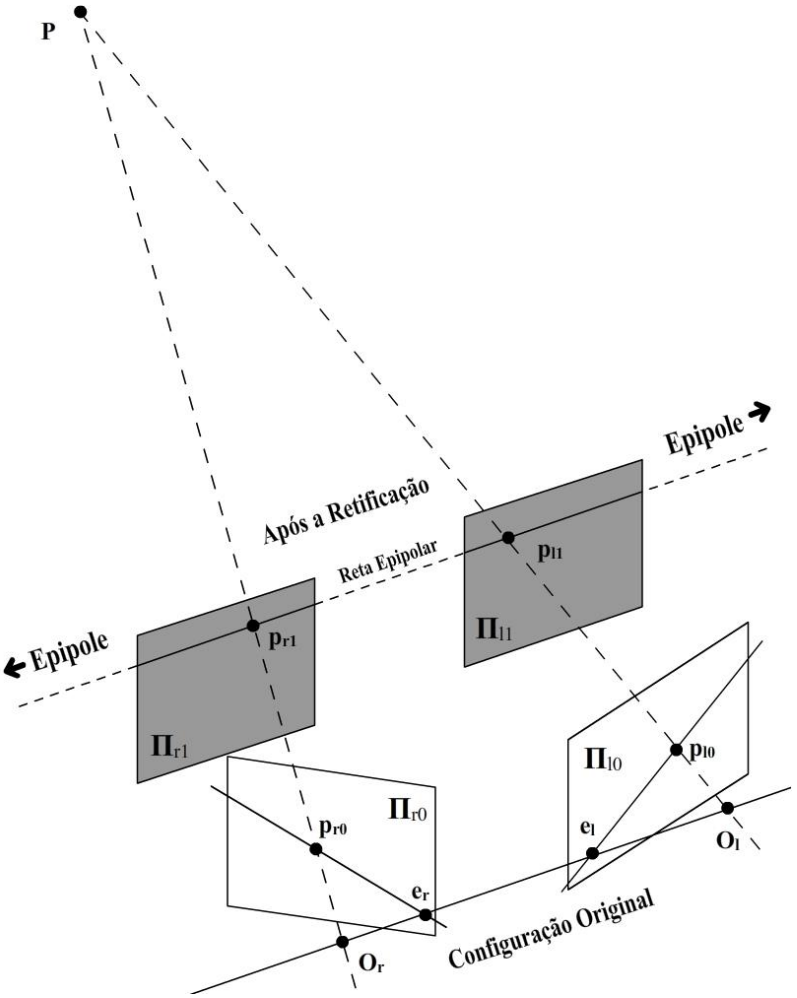
Com a geometria do sistema de câmeras calculada, é possível utilizar a restrição epipolar abordada no tópico 2.2.1, cuja finalidade é restringir a busca da correspondência de um determinado ponto entre os planos de imagens geradas pelas câmeras, somente ao longo da reta epipolar.

Para simplificar ainda mais o custo computacional, é possível realizar um alinhamento horizontal das retas epipolares de ambos planos de imagens tornando a busca pela correspondência ainda mais simples, pois os pontos correspondentes de ambos planos de imagens estarão sempre na mesma posição vertical. Este processo é chamado de Retificação (SZELISKI, 2010).

O processo de retificação consiste em realizar a mudança dos epipolos tendendo ao infinito, gerando-se a transformação dos planos $\boldsymbol{\Pi}_{l0}$ e $\boldsymbol{\Pi}_{r0}$ para os planos $\boldsymbol{\Pi}_{l1}$ e $\boldsymbol{\Pi}_{r1}$, conforme ilustrado na figura 8. Esta transformação pode ser descrita como uma composição da transformação de rotação dos planos das câmeras esquerda e direita, de maneira que os epipolos tendem-se ao infinito, e, consequentemente, as retas epipolares tornam-se paralelas; e pela rotação da câmera da direita de acordo com a transformação descrita pela matriz **R** (ver tópico 2.1.1.3) (CYGANEK; SIEBERT, 2009).

A figura 9 ilustra a aplicação do processo de retificação em uma imagem real, sendo que (a) representa a imagem original com as retas epipolares definidas e (b) a imagem retificada, ou seja, com as retas epipolares paralelas alinhadas horizontalmente.

Figura 8 - Retificação Estéreo




Fonte: Cyganek; Siebert, 2009

Figura 9 - Processo de retificação. (a) imagem original. (b) imagem retificada

(a) (b)

Fonte: Loop; Zhang, 1999

### 2.2.4 **Correspondência Estéreo**

O processo de correspondência estéreo é fundamental para recuperar a informação de um ambiente tridimensional através de um par de imagens coletadas por um sistema de visão estereoscópica. O processo consiste em mensurar a disparidade entre as informações de uma das imagens do par e as respectivas informações correspondentes na outra imagem. Os métodos de correspondência estéreo podem ser classificados em métodos globais ou em métodos locais. Os métodos globais consistem em algoritmos que analisam a imagem como um todo, ou grande parte dela, tornando-os mais robustos contra falsas correspondências, porém com alto custo computacional. Os métodos locais consistem em algoritmos que analisam um ou pequenos grupos de *pixels* por vez, gerando-se um mapa da disparidade existente entre cada correspondência, com um custo computacional otimizado, porém com maior imprecisão (CYGANEK; SIEBERT, 2009).

Em 2005, Heiko Hirschmüller propôs um método semi-global chamado *Semi-Global Matching* (SGM) (HIRSCHMÜLLER, 2005), onde são combinados os conceitos dos métodos globais e dos métodos locais, a fim de obter uma correspondência mais precisa com um menor custo computacional.

O método SGM consiste em uma correspondência de *pixel* (método local) baseado na teoria de probabilidade Informação Mútua (*Mutual Information,* MI), já utilizado como um método de correspondência estéreo por (EGNAL, 2000), e na utilização de um conceito para uma restrição de suavidade 2D, combinando diversas restrições 1D (método global). Dado um par de imagens retificado, sendo $I_b$ a imagem base (origem) localizada à esquerda, e $I_m$ a imagem de correspondência (onde é realizada a busca) localizada à direita, a correspondência de *pixel* (método local) pode ser definida conforme a equação (40) (HIRSCHMÜLLER, 2005):

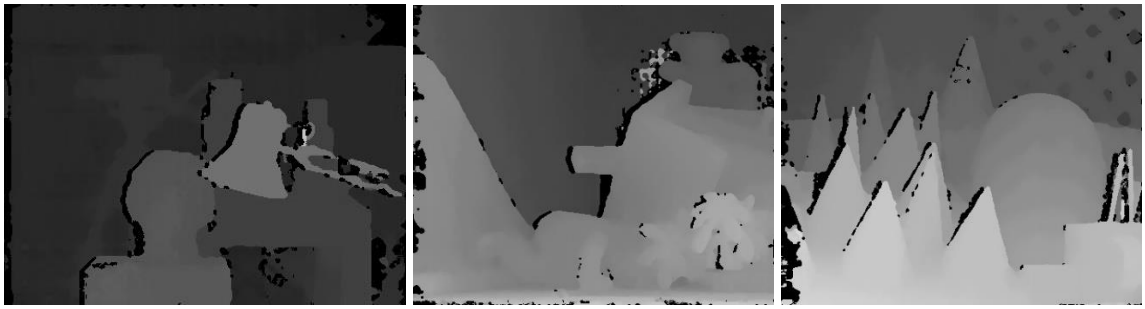
$$\mathbf{q} = \left[p_x - d, p_y\right]^T \quad , \qquad \textbf{(40)}$$

onde $p_x$ e $p_y$ representam as coordenadas do *pixel* na imagem $I_b$, **q** representa o pixel correspondente na imagem $I_m$ e $d$ representa a disparidade.

A figura 10 apresenta as imagens bases ($I_b$) de Tsukuba (384 x 288 *px*), Teddy (450 x 375 *px*) e Cones (450 x 375 *px*) originais (a) e com os respectivos resultados das correspondências realizadas pelo método SGM (b).

Figura 10 – Resultado do método de correspondência SGM

(a)



(b)

Fonte: Hirschmüller, 2005

Devido ao desempenho apresentado nos experimentos realizados em (HIRSCHMÜLLER, 2005), utilizou-se o método SGM para realizar a correspondência do sistema de visão estéreo deste trabalho.

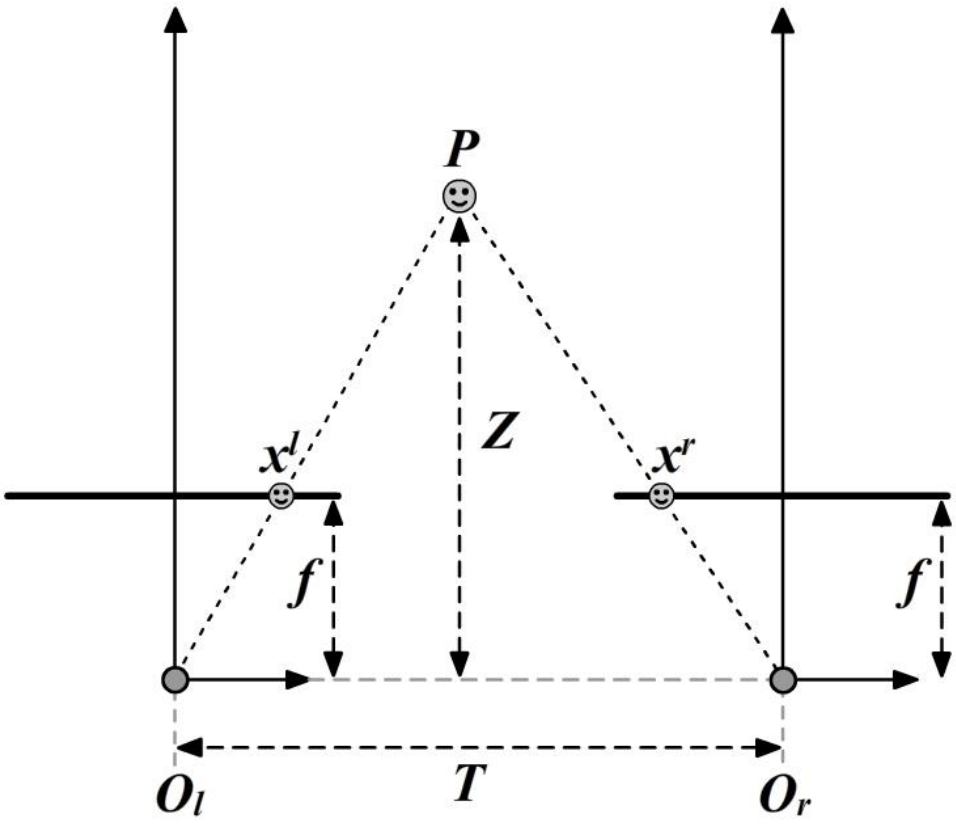
### 2.2.5 **Triangulação**

Com um sistema de câmeras estéreo calibrado, retificado, com as correspondências encontradas e com a disparidade calculada conforme a equação

$$d = x^l - x^r \ , \qquad \textbf{(41)}$$

é possível encontrar a profundidade Z através da semelhança de triângulos, como ilustrado na figura 11 (BRADSKI; KAEHLER, 2008).

Figura 11 - Triangulação




Fonte: Bradski; Kaehler, 2008

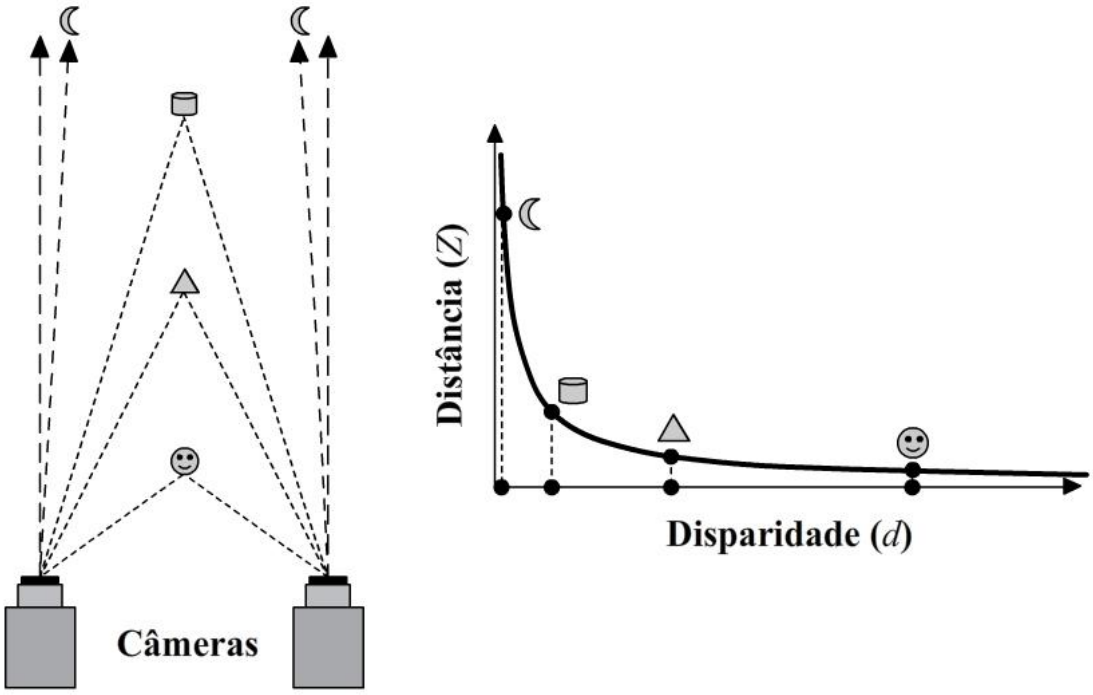
A equação (42) apresenta o cálculo de Z através da semelhança de triângulos:

$$\frac{T-d}{Z-f}=\frac{T}{Z} \quad \Rightarrow \quad Z=\frac{fT}{d}\ , \tag{42}$$

onde, $T$ é a distância entre os centros de projeções $O_l$ e $O_r$, $f$ é a distância focal das câmeras, e $d$ é a disparidade em um ponto ou um bloco, de acordo com resultado do mapa de disparidade gerado no processo de correspondência.

Como a profundidade $Z$ é inversamente proporcional à disparidade $d$, a relação entre estes termos não é linear. Quando a disparidade é próxima de 0, uma pequena diferença em seu valor causa uma diferença muito grande no valor de profundidade. Da mesma forma, quando a disparidade é um valor muito grande, uma pequena diferença em seu valor não altera muito o valor da profundidade. Portanto, conclui-se que em um sistema de visão estéreo, têm-se precisão de profundidade apenas para objetos relativamente próximos das câmeras, conforme ilustrado na figura 12 (BRADSKI; KAEHLER, 2008).

Figura 12 - Relação de distância e disparidade



Fonte: Bradski; Kaehler, 2008

## 2.3 RECONHECIMENTO DE OBJETOS 3D

A representação e a correspondência de objetos 3D têm sido um importante segmento de pesquisa na visão computacional, com aplicações em modelagem, visualização, reconhecimento, classificação e entendimento de uma cena. Descritores baseados em algoritmos de reconhecimento 3D tiveram um forte crescimento e passaram a ser uma solução promissora para aplicações em mundo real 3D, devido à sua robustez contra ruídos (ZHONG, 2009).

Dado uma imagem com a informação de profundidade, resultado da triangulação aplicado em um mapa de disparidade (ver tópicos 2.2.4 e 2.2.5), é possível realizar o reconhecimento de objetos 3D extraindo características da imagem, através de métodos descritores, e comparando essas informações com uma base de dados de objetos já conhecidos.

O custo computacional dos descritores é geralmente alto, inviabilizando sua aplicação em todos os pontos de uma imagem. Portanto, os detectores de pontos de interesse (*keypoints*), são importantes para reduzir a densidade de pontos em uma imagem viabilizando a aplicação dos métodos descritores (FILIPE; ALEXANDRE, 2014).

Segundo (FILIPE; ALEXANDRE, 2014), o método descritor ISS (*Intrinsic Shape Signatures*), apresentado por (ZHONG, 2009), operando como simples detector de *keypoints*, apresentou melhor desempenho na repetitividade dos mesmos conjuntos de *keypoints* em diferentes situações de cenário, considerando ruído, ponto de vista, escala, oclusão ou a combinação destes itens. Neste estudo, foi comparado o desempenho dos algoritmos Harris3D (HARRIS; STEPHENS, 1988), SIFT3D (FLINT, et al., 2007), SUSAN (SMITH; BRADY, 1997) e o ISS, utilizando a base de imagens *RGB-D Object*.

Existem muitos métodos descritores 3D, como por exemplo, SI (JOHNSON; HEBERT, 1999), 3DSC (FROME, et al., 2004), LSP (CHEN; BHANU, 2007), PFH (RUSU, et al., 2008), FPFH (RUSU, et al., 2009), SHOT (TOMBARI, et al., 2010b), USC (TOMBARI, et al., 2010a), RoPS (GUO, et al., 2013a) e o TriSI (GUO, et al., 2013b). Segundo o comparativo de desempenho entre estes descritores realizado por (GUO, et al., 2015), o descritor *"Fast Point Feature Histograms"* (FPFH), apresentado por (RUSU, et al., 2009), e o descritor "*Signature of Histogram of Orientations*" (SHOT), apresentado por

(TOMBARI, et al., 2010a), obtiveram melhores desempenhos na relação de precisão de correspondência no reconhecimento de objetos com eficiência computacional, sendo que o FPFH obteve melhor desempenho na utilização de memória. O descritor "*3D Shape Context*s" (3DSC), apresentado por (FROME, et al., 2004), obteve boa escalabilidade no conjunto de dados testado, porém apresentou alto custo computacional.

Considerando o desempenho e as características de cada método descritor apontado nas pesquisas de (FILIPE; ALEXANDRE, 2014) e (GUO, et al., 2015), foi escolhido o método descritor ISS para ser utilizado como o detector de *keypoints*, e os métodos SHOT, 3DSC e o FPFH, para serem testados no processo de reconhecimento de bola de futebol em 3D, proposto neste trabalho.

Para extrair as características de uma cena e realizar o reconhecimento de objetos 3D, os métodos descritores utilizam um conceito chamado de normal de superfície. A seguir, estão apresentados os conceitos das normais de superfície e o funcionamento de cada um dos métodos descritores utilizados neste trabalho.

### 2.3.1 **Normais de Superfície**

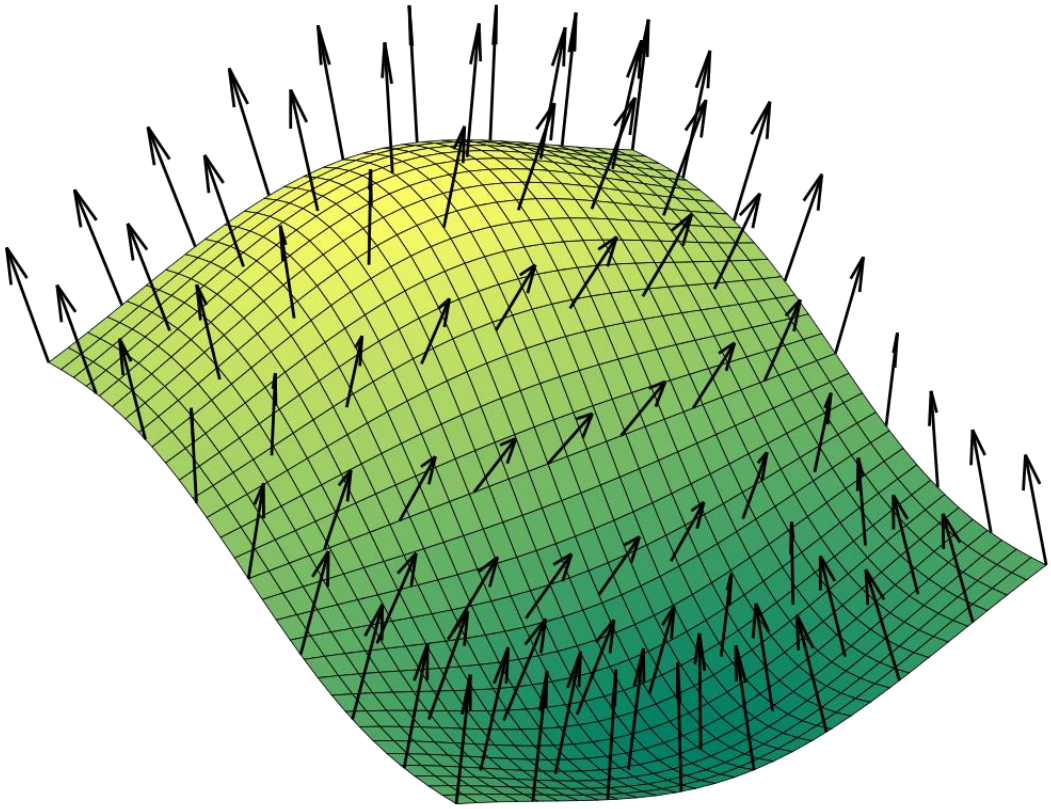
As normais de superfícies são geradas para cada ponto de uma região 3D, através de uma análise dos autovetores e autovalores, também conhecido como o método PCA (*Principal Component Analysis*), de uma matriz de covariância criada a partir dos pontos contidos no arredor do ponto de referência. A definição da matriz de covariância está representada pela equação (43) (RUSU, 2011):

$$C = \frac{1}{k}\sum_{i=1}^{k} \cdot (p_i - \bar{p}) \cdot (p_i - \bar{p})^T \ , \tag{43}$$

onde, $k$ é o número de pontos considerados em uma região 3D em torno do ponto de referência $p_i$ e $\bar{p}$ representa o centroide 3D dos vizinhos mais próximos.

A figura 13 apresenta a orientação dos vetores normais de uma superfície 3D.

Figura 13 – Vetores normais de uma superfície 3D



Fonte: Nicoguaro, 2016

### 2.3.2 **Método descritor ISS (*Intrinsic Shape Signatures*)**

O método descritor "*Intrinsic Shape Signatures*" (ISS) (ZHONG, 2009) consiste em um quadro de referência intrínseco e um vetor de características altamente discriminantes que codifica as características de um modelo 3D.

O quadro de referência intrínseco é uma generalização da clássica referência da normal de superfície para a extração de características em uma região 3D independente do ponto de vista. O vetor de características altamente discriminantes consiste na codificação das características extraídas de uma região 3D, gerado através de um histograma tridimensional utilizando uma esfera centralizada no ponto de referência, representado por $p_i$ (ZHONG, 2009).

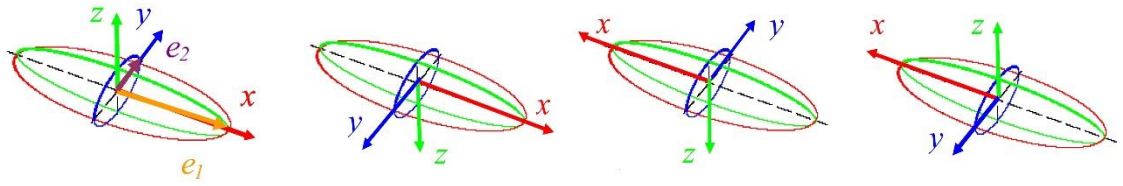
Desta forma, o ISS pode ser representado por (ZHONG, 2009):

$$S_i = \{F_i, f_i\} \ , \qquad \textbf{(44)}$$

onde, $F_i$ representa o quadro de referência intrínseco de uma região 3D em torno de $p_i$ e $f_i$ representa o vetor de características codificadas da mesma região 3D em torno de $p_i$.

O quadro de referência intrínseco $F_i$ é composto pelo ponto de referência $p_i$ e pelo conjunto dos autovetores $\{e_i^1, e_i^2, e_i^3\}$ gerados por uma matriz de dispersão de acordo com os valores das normais de superfície contidas na região 3D. Desta forma, utiliza-se $p_i$ como origem e os vetores $e_i^1$, $e_i^2$ e $e_i^3$, sendo que $e_i^3$ é a resultante do produto vetorial entre $e_i^1$ e $e_i^2$, como os eixos $x$, $y$, $z$, respectivamente, para definir o sistema de coordenadas 3D de $F_i$ para o ponto $p_i$, conforme ilustrado na figura 14.

Figura 14 – Sistema de coordenadas 3D de $F_i$ para o ponto $p_i$




(a) (b) (c) (d)

Fonte: Zhong, 2009

Se um determinado ponto $p_i$ conter amplitudes somente nas direções principais dos eixos, não permitindo a independência do ponto de vista, este ponto é descartado. Este é o princípio de funcionamento deste método como detector de *keypoints*, onde um *keypoint* necessariamente será um ponto $p_i$ independente do ponto de vista.

### 2.3.3 **Método descritor SHOT (*Signature of Histograms of OrienTations*)**

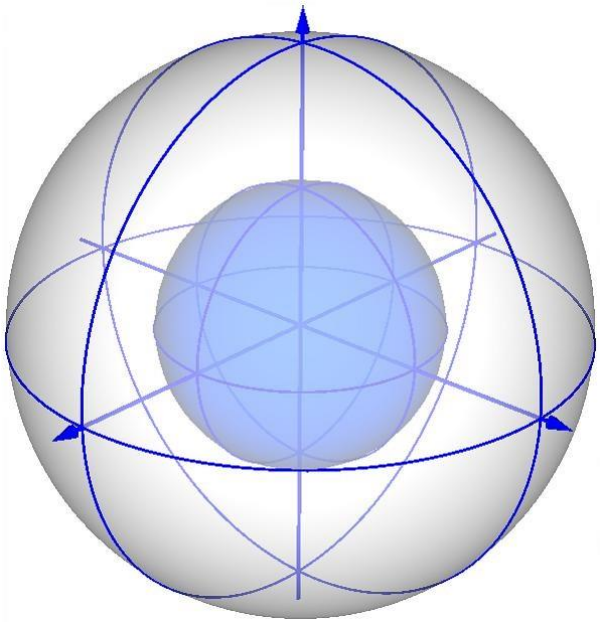
O método descritor SHOT (TOMBARI, et al., 2010b) consiste em uma assinatura gerada através da justaposição dos histogramas locais normalizados, criados para cada ponto de interesse, utilizando-se uma estrutura esférica 3D dividida em 32 setores, sendo 8 divisões de azimutes, 2 divisões de elevação e 2 divisões de radial. Cada histograma local é composto pela contagem acumulada dos pontos de acordo com a função do ângulo $\theta_i$, formado entre a normal do ponto de interesse e a normal de cada ponto contido no setor correspondente da esfera. A função do ângulo $\theta_i$ é representada por (TOMBARI, et al., 2010b):

$$\cos\theta_i = n_u \cdot n_{v_i} \ , \qquad \textbf{(45)}$$

onde, $\theta_i$ representa o ângulo entre a normal do ponto de interesse, representado por $n_u$, e a normal de cada ponto contido no setor correspondente da esfera, representado por $n_{v_i}$.

A figura 15 apresenta o modelo da estrutura esférica utilizada por este método, porém, a fim de proporcionar maior clareza, é apresentado somente 4 azimutes.

Figura 15 – Estrutura esférica 3D utilizada pelo método SHOT

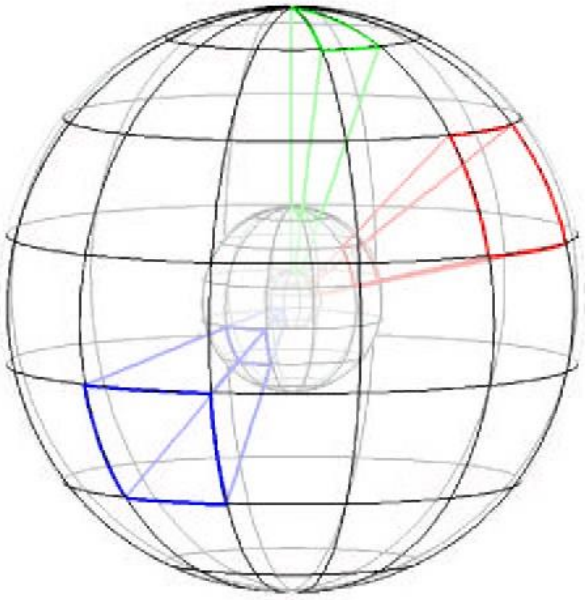


Fonte: Tombari, et al., 2010b

### 2.3.4 **Método descritor 3DSC (*3D Shape Contexts*)**

O método descritor 3DSC (FROME, et al., 2004) é uma adaptação do método 2D *Shape Contexts*, apresentado por Belongie et al. (BELONGIE, et al., 2002), para o ambiente tridimensional, utilizando uma estrutura esférica 3D centralizada no ponto de referência $p_i$ e com a orientação norte alinhada com a normal de superfície de $p_i$. Esta esfera é dividida em setores com espaçamentos iguais de azimute e de elevação, e espaçamentos logarítmicos de radial, conforme ilustrado na figura 16. As divisões de radial, elevação e azimute da estrutura esférica são representadas pelos conjuntos $\{R_0 \ldots R_J\}$, $\{\Theta_0 \ldots \Theta_K\}$ e $\{\Phi_0 \ldots \Phi_L\}$, respectivamente, de maneira que cada setor da esfera corresponde a um elemento no vetor de características $J \times K \times L$. A primeira divisão radial, $R_0$, representa o raio mínimo e a última divisão, $R_J$, representa o raio máximo. Os setores próximos ao centro são pequenos e, para evitar que o método seja sensível à pequenas mudanças, é especificado um raio mínimo maior do que 0 (FROME, et al., 2004).

Figura 16 – Estrutura esférica 3D utilizada pelo método 3DSC



Fonte: Frome, et al., 2004

Cada setor da esfera acumula um valor de peso ω($p_i$) para cada ponto $p_i$ cuja a coordenada esférica relativa fica entre o intervalo de raio, azimute e elevação representados por [$R_j$,$R_{j+1}$], [$\Phi_k$,$\Phi_{k+1}$] e [$\Theta_l$,$\Theta_{l+1}$], respectivamente. O valor acumulado em um setor para cada ponto $p_i$ é descrito através da equação (46) (FROME, et al., 2004):

$$\omega(p_i) = \frac{1}{p_i \sqrt[3]{V(j,k,l)}} , \quad \textbf{(46)}$$

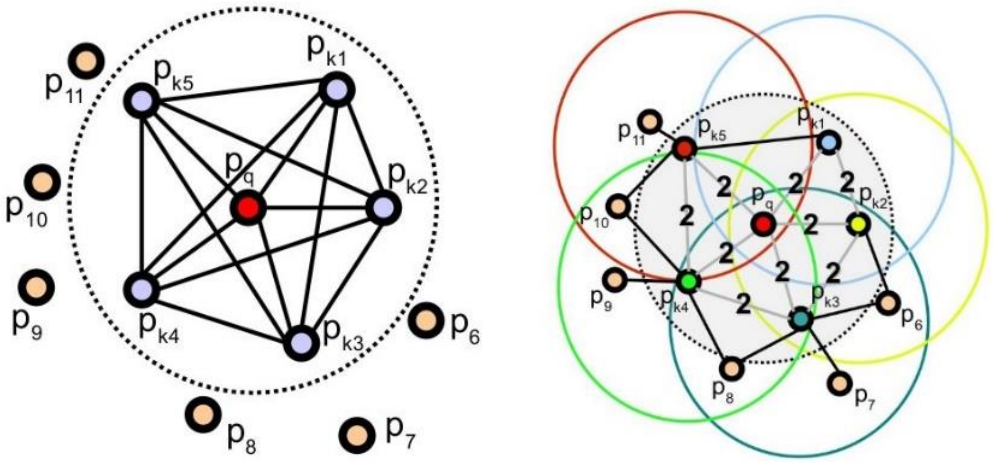
onde, $V(j,k,l)$ representa o volume de um setor e $p_i$ representa o ponto de referência analisado.

### 2.3.5 **Método descritor FPFH (*Fast Point Feature Histograms*)**

O método descritor FPFH (RUSU, et al., 2009) é uma evolução do descritor PFH (RUSU, et al., 2008), com a proposta de reduzir a complexidade computacional limitando as conexões do ponto de interesse com seus respectivos vizinhos. A figura 17 ilustra a diferença do comportamento do algoritmo do PFH e do FPFH. A figura 17 (a) apresenta o diagrama da região influenciada pelo PFH, onde o ponto analisado (vermelho) e os pontos vizinhos (azuis) estão totalmente interconectados em uma malha. A figura 17 (b) apresenta o diagrama da região influenciada pelo FPFH, onde cada ponto analisado (vermelho) é conectado apenas com os vizinhos diretos (contidos no círculo cinza). Cada vizinho direto tem seus próprios vizinhos e os histogramas resultantes são ponderados juntamente com o histograma do ponto analisado para formar o FPFH (RUSU, et al., 2009).

O FPFH utiliza um histograma de pontos simplificados ("*Simplified Point Feature Histogram"* – SPFH) para cada ponto, calculando as relações entre o ponto e seus vizinhos. Com o SPFH calculado, o descritor FPFH é construído através da soma ponderada do SPFH do ponto de interesse e dos SPFHs dos pontos na região de suporte, conforme a equação (47) (RUSU, et al., 2009):

Figura 17 – Comparativo de comportamento do algoritmo do PFH (a) e do FPFH (b)




(a) (b)

Fonte: Zhong, 2009

$$FPFH(p) = SPFH(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_k} \cdot SPFH(p_k) \ , \qquad \textbf{(47)}$$

onde, $w_k$ representa o peso da distância entre o ponto analisado ($p$) e o ponto vizinho ($p_k$) em um dado espaço.

## 2.4 RESUMO DO CAPÍTULO

Este capítulo apresentou os conceitos fundamentais sobre a visão computacional, abordando as câmeras digitais, visão estéreo com geometria epipolar, calibração, retificação, correspondência, triangulação para obter a distância da profundidade e, por fim, foram explorados alguns métodos para reconhecimento de objeto 3D. A seguir, serão apresentados alguns trabalhos que utilizam a visão estéreo para reconhecimento de objetos 3D.

# 3 TRABALHOS CORRELATOS

As pesquisas em visão computacional estéreo aplicada à robôs tiveram uma considerável evolução do final do século XX até os dias de hoje, explorando os benefícios de um sistema estéreo em diversas aplicações.

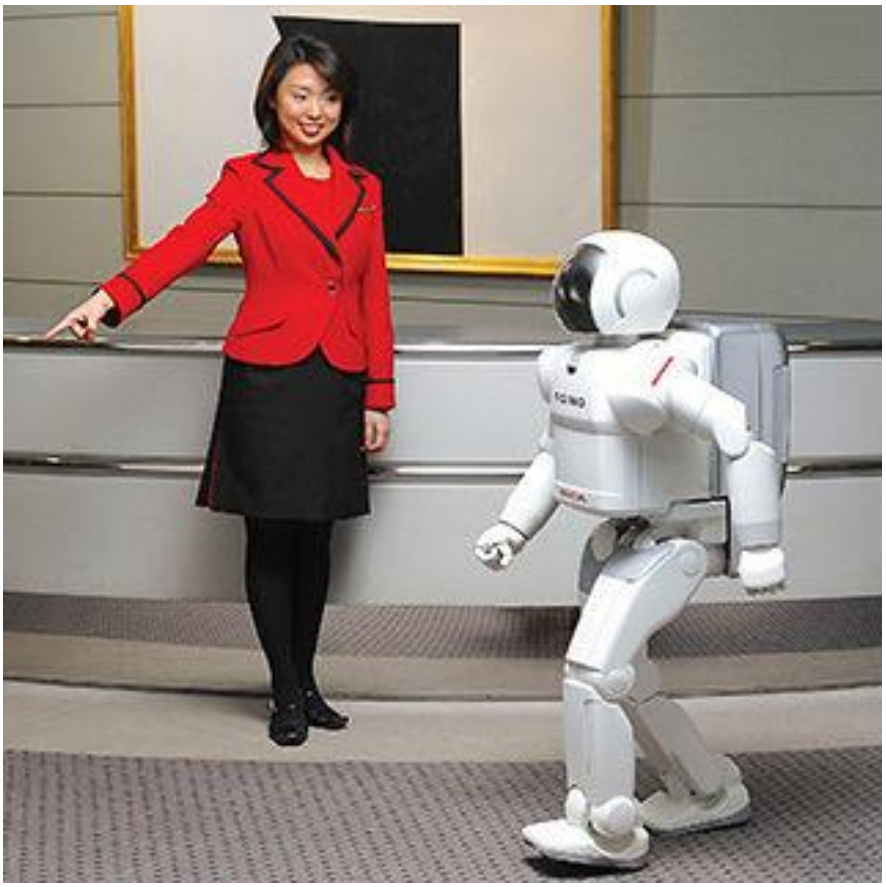
Neste capítulo, são abordados alguns trabalhos que aplicam um sistema de visão estéreo composto somente por duas câmeras *pin-hole* em robôs móveis humanoides, apresentados no tópico 3.1, e alguns trabalhos que aplicam um sistema de visão estéreo para o reconhecimento de objetos 3D, apresentados no tópico 3.2. Por fim, o tópico 3.3 apresenta uma conclusão deste capítulo.

## 3.1 VISÃO ESTÉREO EM ROBÔS MÓVEIS HUMANOIDES

A finalidade de um sistema de visão estéreo em um robô móvel humanoide pode variar de uma simples detecção de obstáculos, até aplicações mais complexas como a de interações com seres humanos.

A Honda R&D desenvolveu uma linha de pesquisa em um robô humanoide chamado ASIMO, ilustrado na figura 18, cujo principal foco é a interação com seres humanos e a realização de tarefas simples. O trabalho Honda R&D (2002) descreve o aperfeiçoamento deste robô com a integração de um sistema de visão computacional estéreo para reconhecimento de humanos e seus gestos, além da detecção de obstáculos para navegação. O sistema estéreo deste robô é composto por duas câmeras coloridas, distanciadas por 74 mm (*baseline*), distância focal de 4mm e uma taxa de atualização de 20 fps na resolução de 768 x 480 *pixels*. O software de visão é composto pelos processos de correção das distorções das lentes, retificação e correspondência estéreo baseado em método local. A detecção de obstáculos é realizada através do mapa de disparidade gerado pela correspondência. Para o reconhecimento de humanos e seus gestos são utilizados algoritmos baseados em *Optical Flow*, para extração do fundo da imagem, algoritmo *Snake* para extração de contornos e formatos geométricos e algoritmo baseado em autovetores para o reconhecimento de faces. A CPU utilizada para realizar o processamento de software foi um Mobile Pentium III – M 1.2 GHz.

Figura 18 - Robô ASIMO, em 2002



Fonte: Honda, 2002

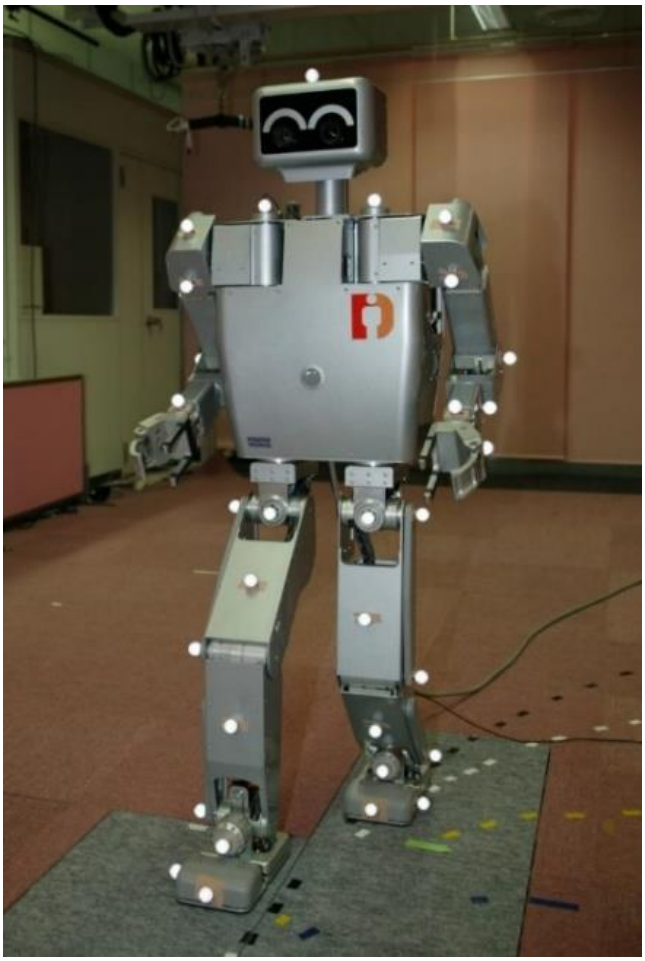
O trabalho Gutmann, et al. (2004), apresenta uma solução para auxiliar o robô humanoide QRIO, desenvolvido pela *Sony Corporation*, apresentado na figura 19, a subir e descer escadas e outros obstáculos cuja superfície seja suficientemente larga e plana para que caiba os pés do robô. Este trabalho faz uso do sistema de visão estéreo apresentado por Sabe, et al. (2004). O sistema de visão estéreo do robô QRIO é composto por duas câmeras coloridas distanciadas por uma *baseline* de 5 cm e resolução de 352 x 288 *pixels*. O software de visão é composto pelos processos de calibração das câmeras, correção das distorções das lentes, correspondência estéreo utilizando um método local e extração de plano do piso utilizando a transformada de *Hough*. A CPU deste robô é composta por um processador MPIS R5000 400MHz.

Figura 19 - Robô QRIO, em 2004

Fonte: Qrio, 2004

Em Thompson; Kagami (2005), é proposto um sistema de visão estéreo para que o robô humanoide H7, desenvolvido pela *Kawada Industries*, ilustrado na figura 20, seja capaz de estimar sua própria localização em um ambiente conhecido e também computar sua odometria, através de um método de correspondência local. O robô H7 possui uma CPU dual Xeon 3.0GHz e um sistema de visão estéreo composto por um sistema SVS da *Videre* montado em sua cabeça, onde captura imagens com resolução de 320 x 240 pixels. Para a geração da segmentação do plano, é utilizado o mapa de profundidade combinado com a transformada de *Hough*. O sistema produz resultados plausíveis em uma faixa de 1 a 3 metros de distância.

Figura 20 - Robô H7



Fonte: Thompson; Kagami, 2005

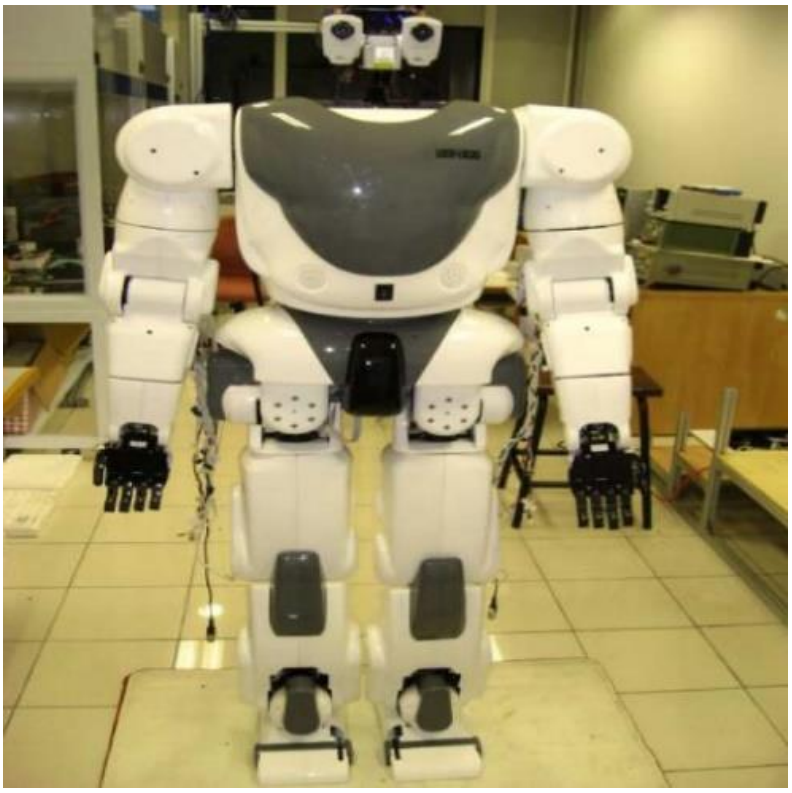
Em Mahani, et al. (2013), aplicou-se o conceito de visão estéreo para auxiliar no controle de um robô humanoide, apresentado na figura 21, quando submetido a um empurrão (problema do *push recovery*). O sistema de visão estéreo auxilia o sistema de recuperação fornecendo a distância de alguns pontos chaves, obtidos através do algoritmo *Shi-tomasi*, uma versão melhorada do algoritmo de detecção de cantos *Harris*. Neste trabalho foi utilizado um kit de um robô humanoide fornecido pela ©Robo Builder, O software de processamento da visão operou com 15-30 pontos chaves em um *laptop* com uma CPU 2.0GHz Core2Duo e 3GB DDR III de RAM. O tempo de resposta para a estimação da distância destes pontos ficou em aproximadamente 30ms.

Figura 21 - Robô humanoide by ©Robo Builder com visão estéreo

Fonte: Mahani, et al., 2013

A pesquisa Zhang, et al. (2014) apresenta um sistema de visão estéreo aplicado para auxiliar no plano de caminhada de um robô humanoide, apresentado na figura 22, onde são detectados obstáculos em sua trajetória. A detecção de obstáculos é realizada através da análise de descontinuidade em seu mapa de disparidade utilizando o método de *Hough*. O sistema de visão deste robô é composto por duas câmeras IP coloridas, com campo de visão de 55 graus na vertical, utilizando uma resolução de 480 x 640 *pixels*.

Figura 22 - Robô humanoide apresentado por Zhang et al., em 2014.



Fonte: Zhang, et al., 2014

## 3.2 VISÃO ESTÉREO NO RECONHECIMENTO DE OBJETOS 3D

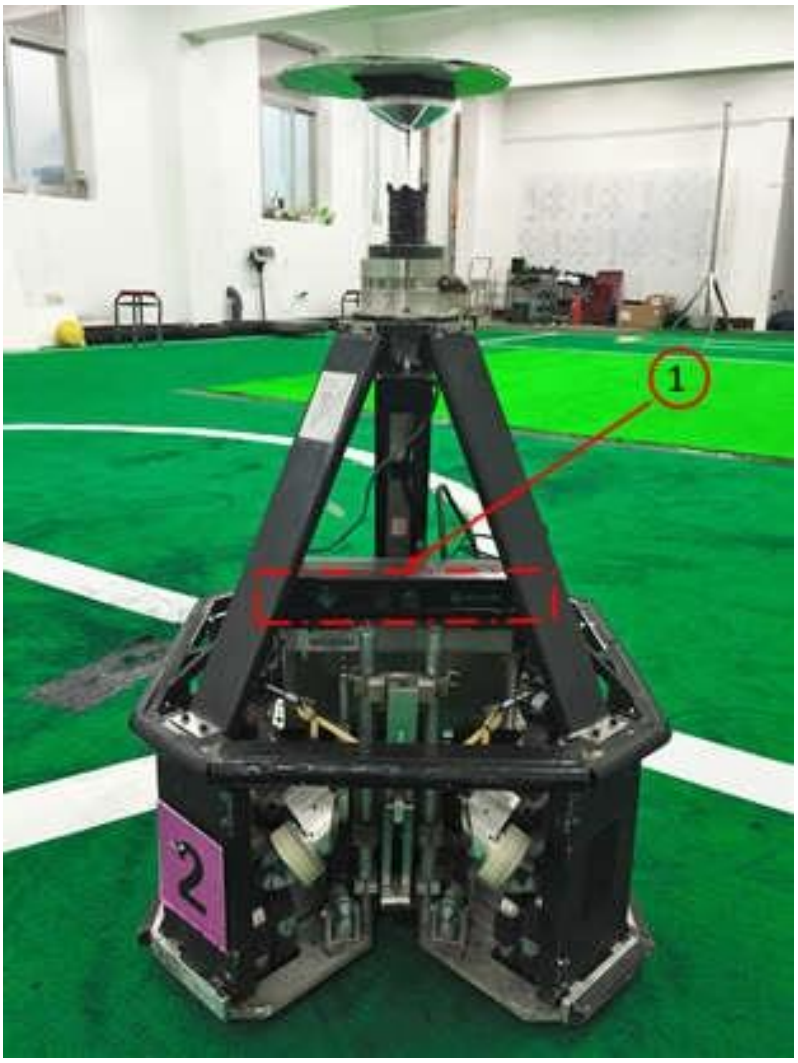
As pesquisas relacionadas com o reconhecimento de objetos 3D normalmente utilizam um sistema de visão estéreo composto por câmeras 3D, isto é, câmeras que capturam a informação de profundidade através de sensores infravermelhos ou *laser*.

A pesquisa Gomes, et al. (2013) aplica um sistema de visão estéreo *Kinect* operando com a biblioteca *Point Cloud Library* para o reconhecimento de objetos utilizando o método descritor SHOT.

A pesquisa Cheng, et al. (2016) consiste em um sistema de visão estéreo aplicado em robôs móveis que atuam na categoria *Middle Size League* (MSL) da RoboCup. O sistema é baseado em câmeras RGB-D, representado por “1” na figura 23. A proposta deste trabalho é a

segmentação de objetos em um ambiente de futebol aplicando o algoritmo RANSAC para extração do plano do solo.

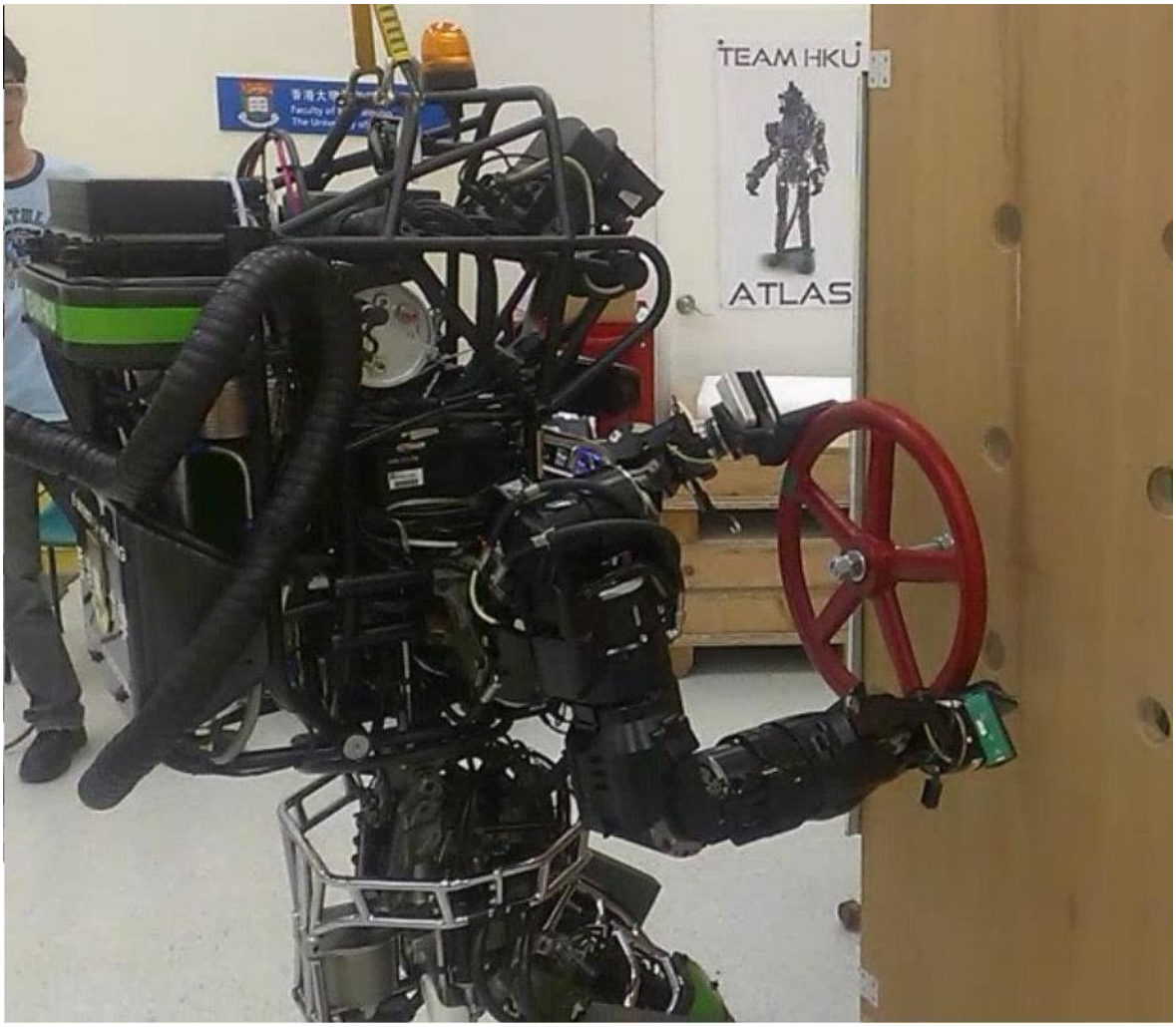
Figura 23 - Robô na liga *Middle Size* da RoboCup, em 2016.




Fonte: Cheng, et al., 2016

O trabalho Newman, et al. (2014), descreve um método para acionamento autônomo de válvula utilizando o robô Atlas, desenvolvido pela *Boston Dynamics Inc.*, apresentado na figura 24. Para realizar o reconhecimento da válvula a ser acionada, o robô utiliza um sistema de visão estéreo operando em conjunto com sensores laser (LIDAR), o sistema operacional ROS e aplica a *Point Cloud Library* na utilização do algoritmo ICP (*Iterative Closest Point*).

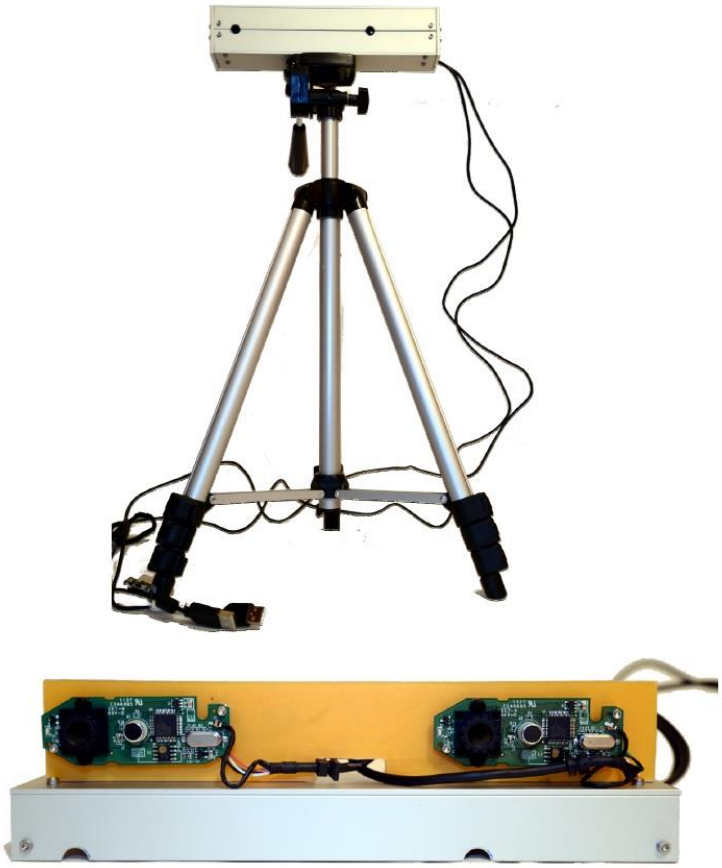
Figura 24 - Robô humanoide Atlas, em 2014.




Fonte: Newman, et al., 2014

A pesquisa Oleari, et al. (2013) apresentou um sistema de visão estéreo composto somente por um par de câmeras Logitech C270, conforme apresentado na figura 25, operando na resolução de 800 x 600 *pixels* e interagindo com o sistema ROS (*Robot Operation System*) para aquisição do mapa de disparidade e reconstrução 3D da cena através da geração de uma *Point Cloud*. Para efetuar o reconhecimento de objetos 3D, foi utilizado o método descritor FPFH.

Figura 25 – Sistema Estéreo apresentado por Fabio Oleari at al., em 2013.



Fonte: Oleari, et al., 2013

## 3.3 RESUMO E CONCLUSÃO DO CAPÍTULO

Este capítulo apresentou algumas pesquisas que utilizaram a visão estéreo para diversas aplicações. Observa-se que os trabalhos relacionados à robôs humanoides que utilizam a visão estéreo sem os recursos de *lasers* ou câmeras 3D, consistem em aplicações como detecção de obstáculos, no problema do *push recovery*, na estimação de odometria, entre outras. Porém, para as aplicações que exigem o reconhecimento de objetos 3D, recorrem-se à utilização de sistemas de visão estéreo com *lasers* ou câmeras 3D. Com isso, nota-se a escassez de trabalhos que utilizam a visão estéreo composta por um simples par de câmeras em robôs humanoides. A seguir, será apresentado o sistema de visão estéreo implementado neste trabalho.

# 4 O SISTEMA DE VISÃO ESTÉREO IMPLEMENTADO

O sistema implementado consiste em um sistema de visão estereoscópica aplicado à um robô móvel humanoide, capaz de identificar bolas de futebol com diferentes texturas utilizadas nas edições passadas da competição RoboCup. Além disso, o sistema também é capaz de estimar a distância que a bola está em relação ao robô.

O sistema é composto por dois segmentos: hardware e software. O tópico 4.1 aborda os detalhes do hardware utilizado para a implementação do sistema. O tópico 4.2 apresenta as etapas do software, tanto para realizar a análise das imagens estéreo como para efetuar o reconhecimento das bolas e estimar suas respectivas distâncias.

## 4.1 HARDWARE

O hardware do sistema de visão estéreo utilizado é composto por duas câmeras Logitech HD Pro Webcam C920, cujas especificações técnicas estão descritas na tabela 1. As câmeras foram montadas em uma estrutura metálica, conforme ilustrado na figura 26, para eliminar possíveis deformações que comprometam a calibração do sistema. As especificações desta estrutura estão descritas na tabela 2.

Tabela 1 – Especificações da Câmera HD Pro Webcam C920.

| | |
|---|---|
| Tipo de conexão | USB 2.0 |
| Tipo de lentes e sensor | Vidro |
| Resolução óptica | Real: 3 MP SW: 15 MP |
| Campo de visão diagonal (DFOV) | 78° |
| Comprimento focal | 3,67 mm |
| Captura da imagem (16:9 W) | 2 MP, 3 MP, 6 MP, 15 MP |
| | 360 p, 480 p, 720 p, 1080 p |
| Taxa de quadros (máx.) | 1080 p a 30 fps |

Fonte: Logitech, 2016

Figura 26 - Estrutura utilizada para o sistema de visão estéreo


Fonte: Autor

Tabela 2 – Especificações da estrutura utilizada para o sistema de visão estéreo.

| | |
|---|---|
| Altura | 60 mm |
| Largura | 210 mm |
| Espessura | 90 mm |
| *Baseline* (ver tópico 2.2.1) | 100 mm |

Fonte: Autor

A estrutura apresentada na figura 26 foi projetada pelo Sr. Diego Rodrigues, responsável pelo desenvolvimento mecânico da empresa Alfatest S.A., e confeccionada gratuitamente pela Indústria Venus. Esta estrutura foi projetada proporcionando uma rigidez suficiente para não permitir movimentos relativos entre as câmeras, a fim de manter a calibração do sistema. Futuramente, esta estrutura poderá ser aperfeiçoada para representar a cabeça de um robô humanoide, porém será necessário manter a rigidez do sistema e a distância da *Baseline* (10 cm), que já está dimensionada no padrão da RoboCup. As demais especificações da estrutura como altura, largura e espessura, permitem alterações de seus valores, pois não ocasionará divergências com os resultados apresentados neste trabalho, uma vez que um sistema de visão estéreo considera apenas as matrizes de rotação e translação entre as câmeras (ver tópico 2.2.1).

Para o desenvolvimento deste trabalho, foi utilizado um tripé profissional para acomodar a estrutura do sistema, conforme ilustrado na figura 27.

Figura 27 - Estrutura do sistema de visão estéreo montada em um tripé

Fonte: Autor

O motivo da utilização deste tripé foi a liberdade de posicionar o sistema em uma altura específica, simulando a altura da cabeça do próximo robô humanoide que está sendo desenvolvido pelo Centro Universitário FEI. Desta forma, todos os pares de imagem capturados pelo sistema foram obtidos com este *offset* na altura, proporcionando uma simulação da visão do próximo robô.

## 4.2 SOFTWARE

O sistema operacional escolhido para a implementação e aplicação do software foi o Linux, distribuição Ubuntu, versão 14.04 LTS. Esta escolha foi devido à compatibilidade com o sistema operacional adotado nos robôs humanoides da FEI.

A linguagem adotada para a implementação do software do sistema de visão estéreo deste trabalho foi o C++, devido à compatibilidade com as bibliotecas gráficas utilizadas neste trabalho.

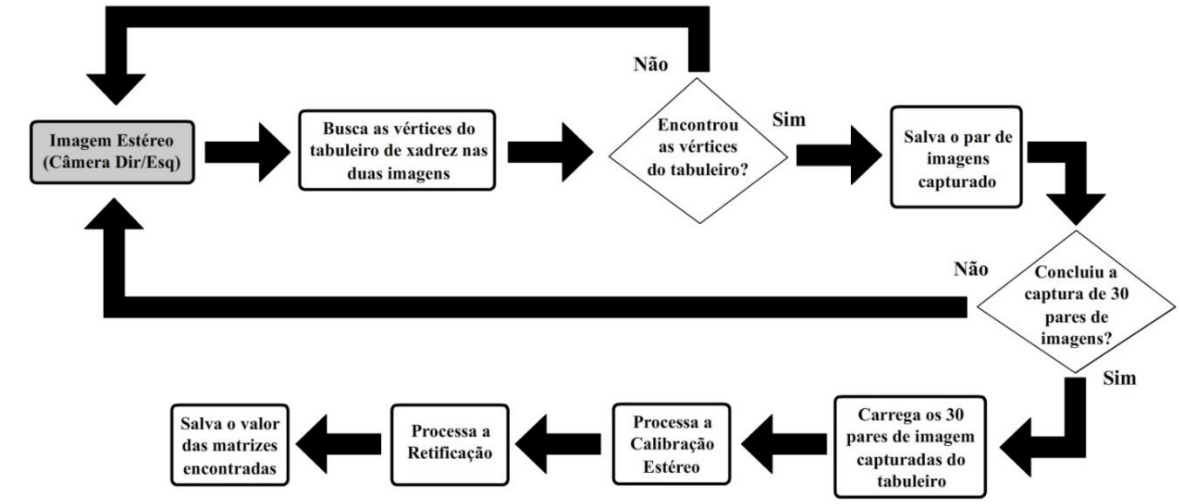
As bibliotecas gráficas escolhidas para o desenvolvimento foram a OpenCV, versão 2.4.9 e a *Point Cloud Library* (PCL), versão 1.8.1. A biblioteca OpenCV foi escolhida devido à compatibilidade com a linguagem de programação C++ e também por ser amplamente utilizada em desenvolvimentos gráficos. A biblioteca PCL foi escolhida para viabilizar o reconhecimento de uma bola através da reconstrução 3D de um par de imagens estéreo.

Após a definição do sistema operacional, da linguagem de programação e das bibliotecas gráficas, o software do sistema de visão estéreo foi implementado para operar em dois modos, sendo o modo *off-line* e o modo *online*.

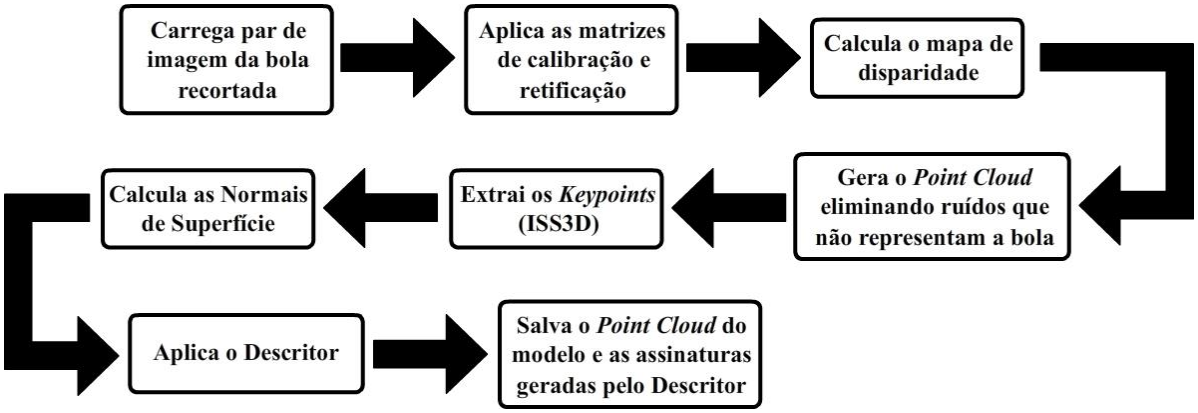
O modo *off-line*, consiste no modo de configuração e captura do sistema, onde são obtidos os pares de imagens responsáveis pela calibração do sistema, é efetuado a calibração do sistema obtendo as matrizes de calibração e retificação, são capturados os pares de imagens e vídeos para testes do sistema, é construído o modelo da bola para o reconhecimento 3D e, por fim, são extraídos pares de imagens dos vídeos capturados para o teste do sistema. A figura 28 ilustra o diagrama do software no modo *off-line* contendo as duas principais funções, que são a calibração do sistema (a) e a construção do modelo da bola (b).

O modo *online* consiste no modo de operação do sistema. Dado um par de imagens do cenário real, seja um par já capturado anteriormente (no modo *off-line*) ou um par capturado no mesmo instante, aplicam-se as matrizes de calibração e retificação obtidas pelo modo *off-line*, realiza-se o processo de correspondência (mapa de disparidade), a reconstrução do cenário em 3D através de uma *Point Cloud*, o reconhecimento 3D da bola e a estimativa da distância da bola em relação ao sistema. A figura 29 ilustra o diagrama do software no modo *online*.

Figura 28 – Diagrama do software operando no modo *off-line*. (a) Calibração estéreo. (b) Construção do modelo da bola.



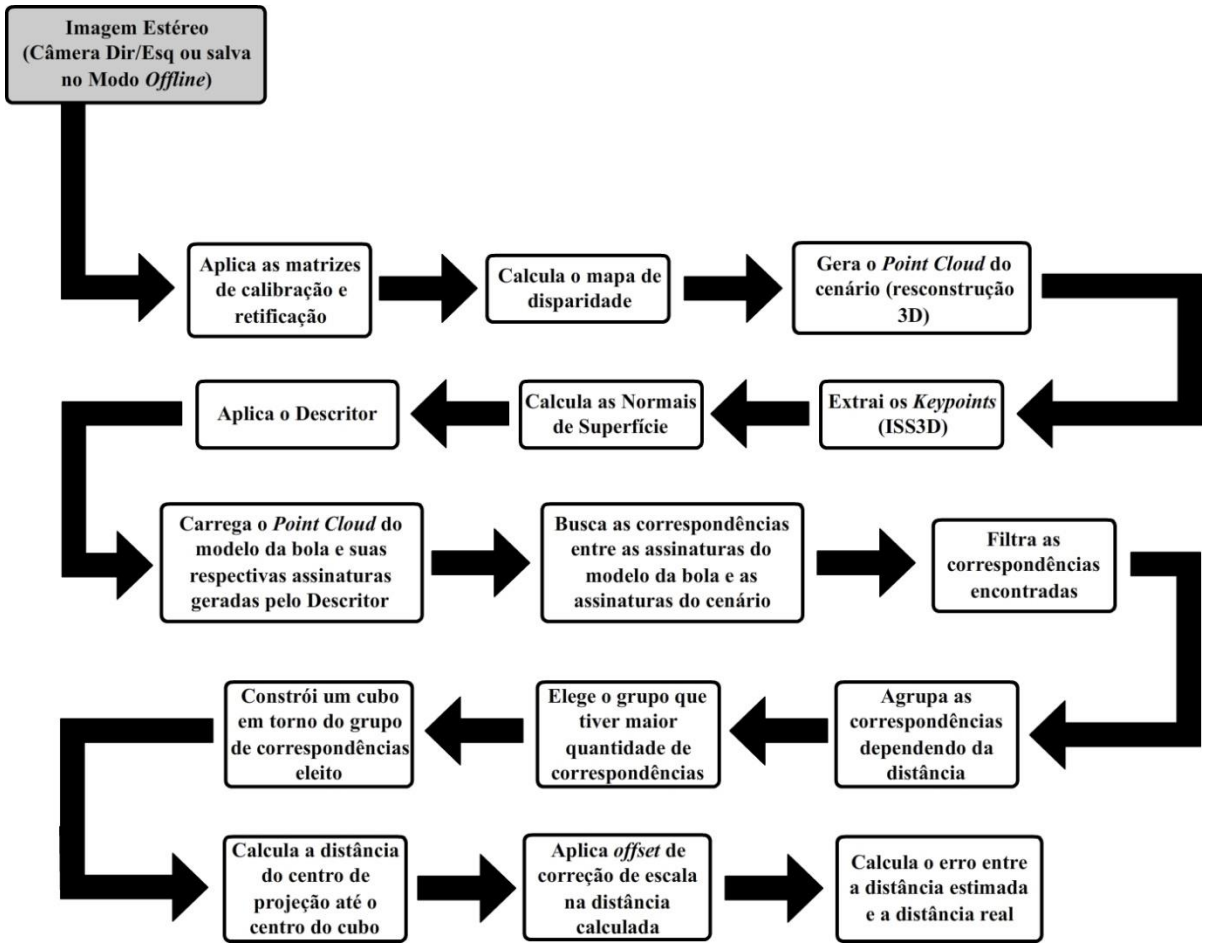

(a)




(b)

Fonte: Autor

Figura 29 – Diagrama do software operando no modo *online*




Fonte: Autor

### 4.2.1 **Calibração Estéreo**

O passo inicial do sistema estéreo é a calibração das câmeras. Existem diversas técnicas para realizar este procedimento. Uma técnica que é amplamente utilizada é a do tabuleiro de xadrez, onde o sistema estéreo captura diversos pares de imagens de um tabuleiro de xadrez e realiza o reconhecimento dos pontos onde existe a mudança de cor das casas internas, ou seja, os vértices internos do tabuleiro, para ambas imagens do par. Através destes pontos, é realizada a calibração do sistema estéreo, que consiste na geração de matrizes que modelam a transição de um vértice de uma imagem do par no mesmo vértice da outra imagem.

A rotina de calibração estéreo do sistema implementado, representada pelo Algoritmo 1, foi baseado no algoritmo disponibilizado por Gary Bradski et al. (BRADSKI; KAEHLER, 2008), e consiste em gerar as matrizes dos parâmetros intrínsecos e os coeficientes de distorções de cada câmera (ver tópico 2.1.2.2), a matriz essencial (ver tópico 2.2.1.1), a matriz fundamental (ver tópico 2.2.1.2), a matriz de rotação e a matriz de translação (ver tópico 2.2.2), utilizando as funções "*findChessboardCorners*", "*cornerSubPix*", "*drawChessboardCorners*" e "*stereoCalibrate*", disponíveis na biblioteca OpenCV. A função "*findChessboardCorners*" é responsável por localizar todos os vértices do tabuleiro nas duas imagens do par, a função "*cornerSubPix*" é responsável por refinar as posições dos vértices encontrados, a função "*drawChessboardCorners*" oferece uma representação gráfica do reconhecimento dos vértices nas próprias imagens e a função "*stereoCalibrate*" calcula as matrizes de calibração do sistema.

### 4.2.2 **Retificação**

Com o sistema estéreo calibrado (ver tópico 4.2.1), é realizado o processo de retificação para o alinhamento das retas epipolares do par de imagens. Para isso, o OpenCV oferece as funções "*stereoRectify*", "*initUndistortRectifyMap*" e "*remap*". A função "*stereoRectify*" recebe como parâmetro a matriz dos parâmetros intrínsecos das duas câmeras, assim como os dois respectivos vetores de coeficientes de distorções de cada câmera, o tamanho da imagem a ser retificada, a matriz de rotação **R** e a matriz de translação **T**. Após receber essas informações, que podem ser obtidas através da função "*stereoCalibrate*" (ver tópico 4.2.1), a rotina calcula as matrizes de rotação de cada câmera e une os dois planos de imagem em um único plano. Consequentemente, todas as linhas epipolares tornam-se paralelas, simplificando o processo da correspondência (ver tópico 2.2.3). Esta rotina, representada no Algoritmo 2, é realizada no modo *off-line*, sendo que as matrizes geradas pela função "*stereoRectify*" são armazenas em um arquivo para que o modo *online* não necessite recalculá-las.

**Algoritmo 1:** Calibração estéreo do sistema

**Entrada:** *imagelist*: lista de pares de imagem capturados,
*boardSize*: quantidade de vértices do tabuleiro (horizontal e vertical),
*squareSize*: dimensões de cada casa do tabuleiro,
*displayCorners*: variável que indica se deseja deve exibir os vértices encontrados na imagem ou não.

**Resultado:** *cameraMatrix[2]*: Matrizes dos parâmetros intrínsecos das duas câmeras,
*distCoeffs[2]:* Matrizes dos coeficientes de distorção das duas câmeras,
*R, T, E, F:* Matrizes de rotação, translação, essencial e fundamental, respectivamente,
*rms_error:* Erro de reprojeção da calibração estéreo.

```
1   for (i←0, j←0; i<30; i++)        //Carrega os vértices dos 30 pares de imagens capturados
2   |  timg ← imagelist[i];
3   |  found ← false;
4   |  for (k←0;k<2;k++)
5   |  |  &corners ← imagePoints[k][j];
6   |  |  found ← findChessboardCorners (timg[k],boardSize,corners,
                                          CALIB_CB_ADAPTIVE_THRESH |
                                          CALIB_CB_NORMALIZE_IMAGE );
7   |  |  if (displayCorners)
8   |  |  |  cimg ← timg[k];
9   |  |  |  drawChessboardCorners (timg[k], boardSize, corners, found);
10  |  |  if (found)
11  |  |  |  cornerSubPix (img, corners, Size(11,11), Size(-1,-1),
12                          TermCriteria(TermCriteria::COUNT + TermCriteria::EPS, 30, 0.01) );
13  for (i←0; i<30; i++)     //Calcula a posição dos vértices de acordo com o tamanho do tabuleiro
14  |  for (j ← 0; j < boardSize.height; j++)
15  |  |  for (k ← 0; k < boardSize.width; k++)
16  |  |  |  objectPoints[i].push_back(Point3f(k*4.9, j*4.6, 0))   // tamanho de das casas do tabuleiro

    //Calcula a calibração estéreo considerando os vértices das imagens capturadas e as posições dos
    vértices calculados de acordo com o tamanho das casas do tabuleiro (4.9 cm x 4.6 cm)
17  rms_error ← stereoCalibrate (objectPoints, imagePoints[0], imagePoints[1],
                                 cameraMatrix[0], distCoeffs[0],
                                 cameraMatrix[1], distCoeffs[1],
                                 imageSize, R, T, E, F,
                                 TermCriteria (TermCriteria::COUNT+TermCriteria::EPS, 100,
                        0.00001),
                                                  CALIB_FIX_ASPECT_RATIO +
                                                  CALIB_ZERO_TANGENT_DIST +
                                                  CALIB_USE_INTRINSIC_GUESS +
                                                  CALIB_SAME_FOCAL_LENGTH +
                                                  CALIB_RATIONAL_MODEL +
                                                  CALIB_FIX_K3 + CALIB_FIX_K4 +
                                                  CALIB_FIX_K5);
```

**Algoritmo 2:** Retificação Estéreo – Modo *off-line*

**Entrada:** *cameraMatrix[2]*: Matrizes dos parâmetros intrínsecos das duas câmeras,
*distCoeffs[2]:* Matrizes dos coeficientes de distorção das duas câmeras,
*imageSize:* Tamanho das imagens do par,
*R, T:* Matrizes de rotação, translação do sistema estéreo.

**Resultado:** *R1, R2:* Matrizes de retificação das câmeras,
*P1, P2:* Matrizes de projeção das câmeras,
*Q:* Matriz da transformação de perspectiva.

```
1   stereoRectify (cameraMatrix[0], distCoeffs[0],
                   cameraMatrix[1], distCoeffs[1],
                   imageSize, R, T, R1, R2, P1, P2, Q,
                   CALIB_ZERO_DISPARITY, 0, imageSize, &validRoi[0], &validRoi[1]);
```

**Algoritmo 3:** Retificação Estéreo – Modo *online*

**Entrada:** *cameraMatrix[2]*: Matrizes dos parâmetros intrínsecos das duas câmeras,
*distCoeffs[2]:* Matrizes dos coeficientes de distorção das duas câmeras,
*imageSize:* Tamanho das imagens do par,
*R1, R2:* Matrizes de retificação das câmeras,
*P1, P2:* Matrizes de projeção das câmeras,
*Img_l, Img_r:* Par de imagens originais (esquerda e direita, respectivamente).

**Resultado:** *rmap[2][2]:* Mapas de re-projeção,
*rimg_l, rimg_r:* Par de imagens retificadas (esquerda e direita, respectivamente).

```
1   initUndistortRectifyMap (cameraMatrix[0], distCoeffs[0],
                             R1, P1, imageSize, CV_16SC2,
                             rmap[0][0], rmap[0][1]);
2   initUndistortRectifyMap (cameraMatrix[1], distCoeffs[1],
                             R2, P2, imageSize, CV_16SC2,
                             rmap[1][0], rmap[1][1]);

3   remap (img_l, rimg_l, rmap[0][0], rmap[0][1], INTER_LINEAR);
4   remap (img_r, rimg_r, rmap[1][0], rmap[1][1], INTER_LINEAR);
```

No modo *online*, após obter um par de imagens do cenário, é utilizada a função "*initUndistortRectifyMap*", responsável pela geração dos mapas das distorções e da retificação de cada câmera, onde são passados para a função "*remap*", responsável por aplicar esses mapas gerados no par de imagens original, gerando-se um novo par de imagens retificado. A rotina que realiza este procedimento está representa no Algoritmo 3.

### 4.2.3 **Aplicação da calibração e retificação no cenário real**

Neste processo, as matrizes de calibração e de retificação do sistema estéreo, obtidas anteriormente (ver tópico 4.2.1 e tópico 4.2.2), são aplicadas nas funções "*initUndistortRectifyMap*" e "*remap*" do OpenCV para realizar a retificação das novas imagens capturadas, sem a necessidade de recalcular as matrizes. Este processo é apresentado no Algoritmo 3, no tópico 4.2.2.

### 4.2.4 **Mapa de Disparidade**

Com o par de imagens retificado (ver tópico 4.2.3), é gerado o mapa de disparidade utilizando o algoritmo *Semi Global Block Matching* desenvolvido por Heiko Hirschmüller (ver tópico 2.2.4) e disponível na biblioteca OpenCV. O Algoritmo 4 apresenta a rotina para geração do mapa de disparidade implementada neste sistema.

**Algoritmo 4:** Mapa de Disparidade

**Entrada:** *img_L_rect, img_R_rect:* imagens retificadas.

**Resultado:** *disp:* Mapa de disparidade

```
1   StereoSGBM sgbm;

    //Atribuição dos valores de configuração do método Semi Global Block Matching
2   sgbm.preFilterCap ← 10;
3   sgbm.SADWindowSize ← 1;
4   sgbm.P1 ← 24;
5   sgbm.P2 ← 96;
6   sgbm.numberOfDisparities ← 96;
7   sgbm.uniquenessRatio ← 1;
8   sgbm.fullDP ← true;

    // Chamada para a função que constrói o mapa de disparidade
9   sgbm (img_L_rect, img_R_rect, *disp);
```

### 4.2.5 **Reconstrução 3D do par de imagens estéreo**

Com o mapa de disparidade gerado (ver tópico 4.2.4), o cálculo da triangulação (ver tópico 2.2.5) é realizado através da rotina "*reprojectImageTo3D*", também disponível no OpenCV. Esta rotina recebe como parâmetro o mapa de disparidade e a matriz de transformação de perspectiva gerada no processo de retificação, Algoritmo 2 (ver tópico 4.2.2), e retorna uma projeção da imagem em 3D, isto é, com a informação de profundidade calculada pela equação (42) do tópico 2.2.5. Com esta projeção, o cenário é reconstruído tridimensionalmente através de uma *Point Cloud*, utilizando a biblioteca PCL, conforme descrito no Algoritmo 5.

**Algoritmo 5:** Recostrução 3D do par de imagens

**Entrada:** *disp:* Mapa de disparidade,
*Q:* Matriz da transformação de perspectiva,

**Resultado:** *pointcloud_xyz:* Point Cloud contendo o cenário 3D reconstruído

```
1   pcl::PointCloud<pcl::PointXYZ>::Ptr pointcloud_xyz(new  pcl::PointCloud<pcl::PointXYZ>());

    //Calcula o Z de cada pixel a partir da disparidade
2   reprojectImageTo3D (disp, xyz, Q, false, CV_32F);

    //Configura o tamanho da pointcloud
3   pointcloud_xyz->width ← static_cast<uint32_t>(disp.cols);
4   pointcloud_xyz->height ← static_cast<uint32_t>(disp.rows);

5   //Informa que é uma pointcloud do tipo não-densa
6   pointcloud_xyz->is_dense ← false;

    //Armazena todos os pontos na pointcloud
7   for (i←0, j←0; i<disp.rows; i++)
8   |   uchar* disp_ptr ← disp.ptr<uchar>(i);
9   |   double* xyz_ptr ← xyz.ptr<double>(i);
10  |   for (int j ← 0; j < disp.cols; ++j)
11  |   |   uchar d ← disp_ptr[j];
12  |   |   Point3f  p ← xyz.at<Point3f>(i, j);
13  |   |   if (d == 0)        //Se a disparidade do ponto for 0 (Z muito longo), descarta o ponto
14  |   |   |   continue;
    |   |   └
15  |   |   point_xyz.x ← p.x;
16  |   |   point_xyz.y ← p.y;
17  |   |   point_xyz.z ← p.z;
18  |   |   pointcloud_xyz->points.push_back(point_xyz);
    |   └
    └
```

### 4.2.6 **Extração dos pontos de interesse (*keypoints*) de uma *Point Cloud***

Com a *Point Cloud* criada (ver tópico 4.2.5), são extraídos os pontos de interesse (*keypoints*), gerando-se uma nova *Point Cloud* menos densa, contendo apenas os *keypoints* (ver tópico 2.3). Para diferenciar-se da *Point Cloud* original, a nova *Point Cloud* será chamada de ***Point Cloud ISS***. O principal objetivo da *Point Cloud ISS* é otimizar o cálculo das normais de superfície e do mapa de descrições gerado pelos métodos descritores. O método utilizado neste trabalho para extrair os *keypoints* foi o *Intrinsic Shape Signatures* (ISS), abordado no tópico 2.3.2, cujo algoritmo está disponível na biblioteca PCL. O processo de extração dos *keypoints* implementado neste sistema é descrito conforme o Algoritmo 6.

### 4.2.7 **Cálculo das normais de superfície**

Após a geração da *Point Cloud ISS* (ver tópico 4.2.6), o sistema calcula as normais de superfície (ver tópico 2.3.1) para cada ponto através da classe "*NormalEstimation*", disponível na biblioteca PCL, considerando um raio de vizinhança pré-definido. A rotina que efetua este processo está descrita no Algoritmo 7.

### 4.2.8 **Cálculo dos descritores**

A *Point Cloud ISS* (ver tópico 4.2.6) e as normais de superfície calculadas para cada ponto (ver tópico 4.2.7) são passadas para um algoritmo descritor, a fim de gerar uma descrição específica para cada ponto (ver tópico 2.3). A estratégia para a geração das descrições varia de acordo com cada método descritor, mas basicamente consiste em um cálculo considerando os vetores das normais de superfície em um raio de vizinhança em torno do ponto que está sendo calculado. Com isso, a descrição de cada ponto sofrerá influência dos pontos vizinhos contidos na região pré-definida. O Algoritmo 8 apresenta a rotina implementada para obter as descrições de cada método descritor.

**Algoritmo 6:** Extração dos Keypoints de uma Point Cloud

**Entrada:** *pointcloud_xyz:* Point Cloud original.

**Resultado:** *pointcloud_iss:* Point Cloud contendo apenas os Keypoints.

```
    // Declaração das variáveis utilizadas
1   pcl::search::KdTree<pcl::PointXYZ>::Ptr tree (new pcl::search::KdTree<pcl::PointXYZ> ());
2   pcl::ISSKeypoint3D<pcl::PointXYZ, pcl::PointXYZ> iss_detector;
3   pcl::PointCloud<pcl::PointXYZ>::Ptr pointcloud_iss (new pcl::PointCloud<pcl::PointXYZ> ());

    //Atribuição dos valores de configuração do detector de keypoints ISS
4   iss_detector.setSearchMethod (tree);
5   iss_detector.setSalientRadius (0.05);
6   iss_detector.setNonMaxRadius (0.03);
7   iss_detector.setThreshold21 (0.975);
8   iss_detector.setThreshold32 (0.975);
9   iss_detector.setMinNeighbors (1);
10  iss_detector.setNumberOfThreads (4);
11  iss_detector.setInputCloud (pointcloud_xyz);

    //Chamada para a função que constrói a pointcloud contendo os ISS Keypoints
12  iss_detector.compute (*pointcloud_iss);
```

**Algoritmo 7:** Cálculo das normais de superfície

**Entrada:** *pointcloud_iss:* Point Cloud contendo apenas os Keypoints.

**Resultado:** *normals:* Point Cloud contendo as normais de superfície.

```
    // Declaração das variáveis utilizadas
1   pcl::NormalEstimation<pcl::PointXYZ, pcl::Normal> ne;
2   pcl::search::KdTree<pcl::PointXYZ>::Ptr tree_2 (new pcl::search::KdTree<pcl::PointXYZ> ());
3   pcl::PointCloud<pcl::Normal>::Ptr normals (new pcl::PointCloud<pcl::Normal>);

    // Atribuição dos valores de configuração para o cálculo das normais de superfície
4   ne.setInputCloud (pointcloud_iss);
5   ne.setSearchMethod (tree_2);
6   ne.setRadiusSearch (0.40);

    //Chamada para a função que calcula as normais de superfície
7   ne.compute (*normals);
```

**Algoritmo 8:** Métodos descritores

**Entrada:** *pointcloud_iss:* Point Cloud contendo apenas os Keypoints,
*normals:* Point Cloud contendo as normais de superfície,
*descriptor_type:* Variável que indica qual método deve-se utilizar.

**Resultado:** *descriptors:* Mapa de descrições gerado pelo método descritor.

```
1   pcl::search::KdTree<pcl::PointXYZ>::Ptr tree (new pcl::search::KdTree<pcl::PointXYZ> ());

2   if (descriptor_type == FPFH_DESCRIPTOR)
3   |   pcl::FPFHEstimation<pcl::PointXYZ, pcl::Normal, pcl::FPFHSignature33> fpfh;
4   |   fpfh.setInputCloud(pointcloud_iss);
5   |   fpfh.setInputNormals(normals);
6   |   fpfh.setSearchMethod(tree);
7   |   fpfh.setRadiusSearch(0.45);
8   |   fpfh.compute(*descriptors);
9   else if (descriptor_type == SHOT_DESCRIPTOR)
10  |   pcl::SHOTEstimation<pcl::PointXYZ, pcl::Normal, pcl::SHOT352> shot;
11  |   shot.setInputCloud(pointcloud_iss);
12  |   shot.setInputNormals(normals);
13  |   shot.setSearchMethod(tree);
14  |   shot.setRadiusSearch(0.45);
15  |   shot.compute(*descriptors);
16  else if (descriptor_type == SC3D_DESCRIPTOR)
17  |   pcl::ShapeContext3DEstimation<pcl::PointXYZ, pcl::Normal, pcl::ShapeContext1980> sc3d;
18  |   sc3d.setInputCloud(pointcloud_iss);
19  |   sc3d.setInputNormals(normals);
20  |   sc3d.setSearchMethod(tree);
21  |   sc3d.setRadiusSearch(0.675);
22  |   sc3d. setMinimalRadius (0.0675);
23  |   sc3d. setPointDensityRadius (0.135);
24  |   sc3d.compute(*descriptors);
```

### 4.2.9 **Criação do modelo da bola**

Para realizar o reconhecimento da bola em uma cena, é necessário a criação prévia de um modelo da bola para que o sistema a tenha como referência de correspondência. Este modelo consiste em uma *Point Cloud ISS* (ver tópico 4.2.6) e um mapa de descrições gerado por um método descritor (ver tópico 4.2.8). A *Point Cloud ISS* e o mapa de descrições não

possuem a informação de cor dos pontos. Portanto, a cor e a textura da bola são irrelevantes para a criação do modelo.

### 4.2.10 **Reconhecimento da Bola**

O reconhecimento da bola em uma cena consiste na busca de correspondências entre o mapa de descrições da cena e o mapa de descrições do modelo da bola (ver tópico 2.3). As correspondências são realizadas através da classe "*CorrespondenceEstimation*" e são filtradas através dos métodos "*CorrespondenceRejectorOneToOne*" e "*CorrespondenceRejectorMedianDistance*", todos disponíveis na biblioteca PCL. O filtro "*CorrespondenceRejectorOneToOne*" elimina uma dupla correspondência para o mesmo ponto, e o filtro "*CorrespondenceRejectorMedianDistance*" rejeita as correspondências que possui uma distância, em relação à sua correspondência mais próxima, superior à distância média entre todas as correspondências.

Após identificar todas as correspondências, as mesmas são agrupadas de acordo com um limite máximo de distância estabelecido entre uma correspondência e outra. O grupo que possuir maior número de correspondência, isto é, a região que tiver a maior concentração de correspondências, e este número for maior do que um limite mínimo estabelecido, é eleito como sendo a localização da bola na cena. Para sinalizar a posição na cena onde o sistema identificou a bola, é desenhado um cubo em torno do grupo de maior correspondência.

A rotina que define o processo de reconhecimento da bola está descrita no Algoritmo 9.

### 4.2.11 **Cálculo da distância da bola**

O cálculo da distância estimada depende do resultado do reconhecimento da bola (ver tópico 4.2.10). Caso a bola seja reconhecida, o cálculo de sua respectiva distância, em centímetros, é baseado nos valores das coordenadas da *Point Cloud* referentes ao centro do cubo criado como referência, aplicando-se um fator métrico para correção de escala. O Algoritmo 10 apresenta a rotina que efetua este cálculo.

**Algoritmo 9:** Reconhecimento 3D da bola

**Entrada:** *pointcloud_iss:* Point Cloud da cena contendo apenas os Keypoints,
*normals:* Point Cloud da cena contendo as normais de superfície,
*descriptors:* Mapa de descrições da cena gerado pelo método descritor,
*pointcloud_model:* Point Cloud do modelo contendo apenas os Keypoints,
*normals_model:* Point Cloud do modelo contendo as normais de superfície,
*descriptors_model:* Mapa de descrições do modelo gerado pelo método descritor,
*descriptor_type:* Variável que indica qual método deve-se utilizar.

**Resultado:** *x_corr_max:* Posição na coordenada x do vértice referente ao limite superior do cubo,
*x_corr_min:* Posição na coordenada x do vértice referente ao limite inferior do cubo,
*y_corr_max:* Posição na coordenada y do vértice referente ao limite superior do cubo,
*y_corr_min:* Posição na coordenada y do vértice referente ao limite inferior do cubo,
*z_corr_max:* Posição na coordenada z do vértice referente ao limite superior do cubo,
*z_corr_min:* Posição na coordenada z do vértice referente ao limite inferior do cubo.

```
    //Obtêm as Correspondências
1   pcl::CorrespondencesPtr correspondences(new pcl::Correspondences());

2   if (descriptor_type == FPFH_DESCRIPTOR)
3   |   pcl::registration::CorrespondenceEstimation<pcl::FPFHSignature33, pcl::FPFHSignature33>
    est;
    L
4   else if (descriptor_type == SHOT_DESCRIPTOR)
5   |   pcl::registration::CorrespondenceEstimation<pcl::SHOT352, pcl::SHOT352> est;
    L
6   else if (descriptor_type == SC3D_DESCRIPTOR)
7   |   pcl::registration::CorrespondenceEstimation<pcl::ShapeContext1980, pcl::ShapeContext1980>
    est;
    L
8   est.setInputSource (descriptors_model);
9   est.setInputTarget (descriptors);
10  est.determineCorrespondences (*correspondences);

    //Filtra as Correspondências
11  pcl::CorrespondencesPtr correspondences_result_rej_one_to_one(new pcl::Correspondences());
12  pcl::CorrespondencesPtr correspondences_result_rej_median_dist(new pcl::Correspondences());
13  pcl::registration::CorrespondenceRejectorOneToOne corr_rej_one_to_one;
14  pcl::registration::CorrespondenceRejectorMedianDistance corr_rej_med_dist;

15  corr_rej_one_to_one.setInputCorrespondences (correspondences);
16  corr_rej_one_to_one.getCorrespondences (*correspondences_result_rej_one_to_one);
17  corr_rej_med_dist.setInputCorrespondences (correspondences_result_rej_one_to_one);
18  corr_rej_med_dist.getCorrespondences (*correspondences_result_rej_median_dist);

    //Calcula o valor médio das distâncias das Correspondências filtradas
19  num_corr = correspondences_result_rej_median_dist->size ();
20  for (int i ← 0; i < num_corr; i++)
21  |   values[i][0] ← (*correspondences_result_rej_median_dist)[i].index_match;
22  |   values[i][1] ← sqrt(pow((model_keypoints->points[(* correspondences_result_rej_median_dist)
    [i].index_match].x),2)+pow((model_keypoints->points[(*correspondences_result_rej_median_dist)
    [i].index_match].y),2)+pow((model_keypoints->points[(*correspondences_result_rej_median_dist)
    [i].index_match].z),2));
    L
```

```
    //Chama-se o algoritmo quickSort para ordenar as Correspondências
23  quickSort(values,0,num_corr-1);

24  for (int i ← 0; i < num_corr; i++)
25  |   if (i>0)
26  |   |   values_dif[i][0] ← values[i][0] - values[i-1][0];
27  |   |   values_dif[i][1] ← sqrt(pow((values[i][1] - values[i-1][1]),2));
28  |   |   id_median ← id_median + values_dif[i][0];
29  |   |   dist_median ← dist_median + values_dif[i][1];
30  |   └
31  |   else
32  |   |   values_dif[i][0] ← 0;
33  |   |   values_dif[i][1] ← 0;
34  |   └
35  └
36  id_median ← (id_median/num_corr);
37  dist_median ← (dist_median/num_corr);

    //Agrupa as Correspondências de acordo com a distância entre elas e identifica o grupo que possui
    mais Correspondências
38  for (int i ← 0; i < num_corr; i++)
39  |   if ( (i==0) ||
            ((values_dif[i][0] > id_median) && ((values_dif[i][1] > dist_median))) ||
            (values_dif[i][0] > MAX_ID_TOLERANCE) || (values_dif[i][1] > MAX_DIST_TOLERANCE)
            ||
            (i == num_corr) )
40  |   |   if (i != num_corr)
41  |   |   |   vector<int> elements;
42  |   |   |   result.push_back(elements);
43  |   |   └
44  |   |   if (count_por_grupo)
45  |   |   |   if (count_por_grupo > max_count_por_grupo)
46  |   |   |   |   max_count_por_grupo ← count_por_grupo;
47  |   |   |   |   grupo_max_count ← grupo;
48  |   |   |   └
49  |   |   └
50  |   |   if (i != num_corr)
51  |   |   |   grupo++;
52  |   |   |   count_por_grupo ← 0;
53  |   |   |   count_por_grupo++;
54  |   |   |   result[grupo-1].push_back(values[i][0]);
55  |   |   └
56  |   └
57  |   else
58  |   |   result[grupo-1].push_back(values[i][0]);
59  |   |   count_por_grupo++;
60  |   └
61  └

    //Calcula a porcentagem que representa as Correspondências do maior grupo em relação a todas
    Correspondências encontradas e verifica os requisitos de porcentagem e número de pontos
    mínimos. Caso atenda, calcula os limites do cubo envolvendo todas as Correspondências do maior
    grupo.
62  per_cent_corr ← ((result[grupo_max_count-1].size()*100.0) / num_corr);
63  ball_wasFound ← true;
```

```
64      if (per_cent_corr < CORR_MIN_PER_CENT)
65      |   ball_wasFound ← false;
66      |
67      else if (correspondences_result_rej_median_dist->size () < NUM_MIN_POINTS)
68      |   ball_wasFound ← false;
69      |
70      else
71      |   for (int i←0; i< result[grupo_max_count-1].size();i++)
72      |   |   if (model_keypoints->points[result[grupo_max_count-1][i]].x > x_corr_max)
73      |   |   |   x_corr_max ← model_keypoints->points[result[grupo_max_count-1][i]].x+0.1;
74      |   |   |
75      |   |   if (model_keypoints->points[result[grupo_max_count-1][i]].x < x_corr_min)
76      |   |   |   x_corr_min ← model_keypoints->points[result[grupo_max_count-1][i]].x-0.1;
77      |   |   |
78      |   |   if (model_keypoints->points[result[grupo_max_count-1][i]].y > y_corr_max)
79      |   |   |   y_corr_max ← model_keypoints->points[result[grupo_max_count-1][i]].y+0.1;
80      |   |   |
81      |   |   if (model_keypoints->points[result[grupo_max_count-1][i]].y < x_corr_min)
82      |   |   |   y_corr_min ← model_keypoints->points[result[grupo_max_count-1][i]].y-0.1;
83      |   |   |
84      |   |   if (model_keypoints->points[result[grupo_max_count-1][i]].z > z_corr_max)
85      |   |   |   z_corr_max ← model_keypoints->points[result[grupo_max_count-1][i]].z+0.1;
86      |   |   |
87      |   |   if (model_keypoints->points[result[grupo_max_count-1][i]].z < z_corr_min)
88      |   |   |   z_corr_min ← model_keypoints->points[result[grupo_max_count-1][i]].z-0.1;
89      |   |   |
90      |   |
91      |
```

**Algoritmo 10:** Cálculo da distância estimada da bola

**Entrada:** *x_corr_max:* Posição na coordenada x do vértice referente ao limite superior do cubo,
*x_corr_min:* Posição na coordenada x do vértice referente ao limite inferior do cubo,
*y_corr_max:* Posição na coordenada y do vértice referente ao limite superior do cubo,
*y_corr_min:* Posição na coordenada y do vértice referente ao limite inferior do cubo,
*z_corr_max:* Posição na coordenada z do vértice referente ao limite superior do cubo,
*z_corr_min:* Posição na coordenada z do vértice referente ao limite inferior do cubo.

**Resultado:** *dist_estimada_bola:* Distância estimada da bola em cm.

```
1   dist_estimada_bola ← (sqrt (pow (sqrt (pow ( ((z_corr_max+z_corr_min)/2),2) +
                          pow(((x_corr_max+x_corr_min)/2),2)),2) +
                          pow(((y_corr_max+y_corr_min)/2),2))
                          *16.9)); //O valor 16.9 é o fator métrico para compatibilização de escala (ver
                          5.5.4)
```

## 4.3 RESUMO DO CAPÍTULO

Este capítulo apresentou o sistema implementado neste trabalho que consiste em um sistema de visão estéreo composto por hardware e software, para aplicação em robôs móveis humanoides que atuam na categoria de futebol da competição RoboCup. A seguir, serão apresentados os experimentos e resultados referentes aos processos abordados neste capítulo.

# 5 EXPERIMENTOS E RESULTADOS

Os experimentos realizados foram divididos nas seguintes fases: calibração e retificação do sistema estéreo, captura das imagens de um cenário real, processamento das imagens coletadas para uma reconstrução 3D através da biblioteca *Point Cloud Library*, criação do modelo da bola e reconhecimento da bola com sua respectiva distância estimada em relação ao sistema utilizando três diferentes métodos descritores.

O hardware utilizado para realizar os experimentos foi um *laptop* Dell Inspiron 7520 15R SE, com processador i7-3632QM 2.2GHz (3ª geração) e 8 GB de RAM DDR3 1600 MHz. Este hardware é semelhante ao hardware utilizado na próxima geração de robôs humanoides do Centro Universitário FEI, que será composto por um Intel NUC i7 com 8 GB de RAM DDR3.

A seguir, são abordados os detalhes de cada uma das fases dos experimentos e seus respectivos resultados.

## 5.1 CALIBRAÇÃO E RETIFICAÇÃO DO SISTEMA ESTÉREO

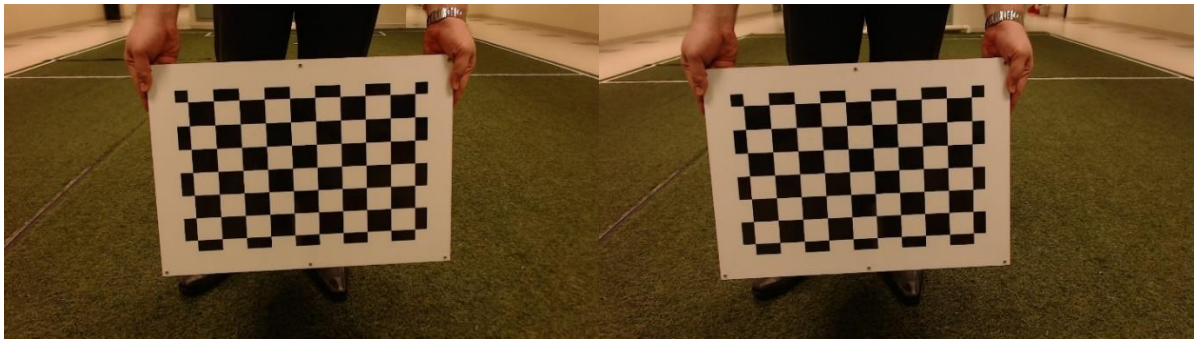
Para realizar a calibração do sistema estéreo, foi adotada a técnica do tabuleiro de xadrez, conforme mencionada no tópico 4.2.1. Para isso, foi impresso, em alta definição, um tabuleiro de xadrez utilizando o material poliestireno com dimensões de 59,3 cm de largura por 42,1 cm de altura, margens de 5 cm, 10 vértices internas na horizontal e 7 vértices internas na vertical, onde cada casa possui dimensões de 4,9 cm de largura por 4,6 cm de altura. Este tabuleiro foi fixado à uma tábula de madeira rígida com a mesma dimensão, para garantir que o tabuleiro não sofresse nenhum tipo de deformação durante a captura das imagens.

Na sequência, foram obtidos 30 pares de imagens deste tabuleiro em diferentes posições e orientações. Cada par de imagens possui 70 vértices (10 vértices na horizontal e 7 vértices na vertical), sendo que para cada vértice é efetuado uma correspondência entre as imagens do par. Desta forma, considerando 70 pontos de correspondência entre as imagens do par (imagem direita e imagem esquerda), e considerando também que foram utilizados 30

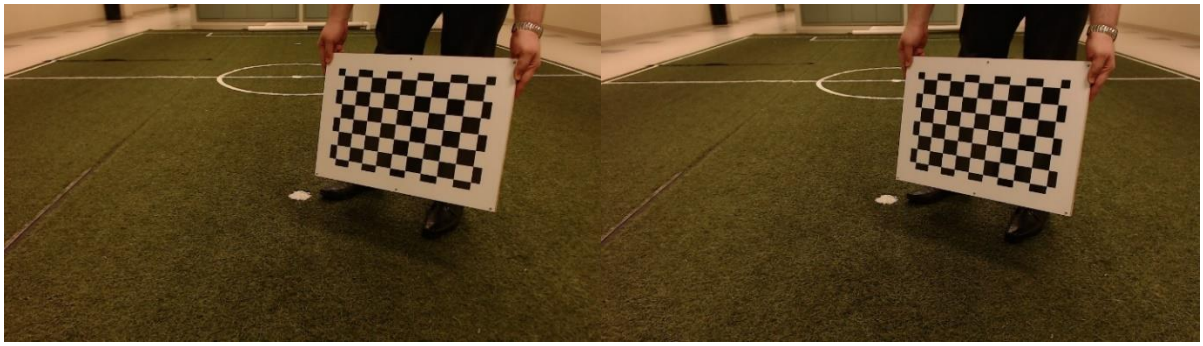
pares de imagens, podemos concluir que a calibração do sistema foi efetuada com o total de 2.100 pontos de correspondência.

A figura 30 apresenta dois pares de imagens do tabuleiro de xadrez utilizado para a calibração do sistema.

Figura 30 - Pares de imagens durante a calibração estéreo



(a)



(b)

Fonte: Autor

A tabela 3 apresenta valores do resultado do processo de calibração do sistema estéreo.

Tabela 3 – Resultados da Calibração Estéreo

| **Descrição** | **Valores** |
|---|---|
| Parâmetros Intrínsecos da Câmera Esq. (ver tópico 2.1.2.2) | $\begin{bmatrix} 969{,}9328 & 0 & 641{,}0605 \\ 0 & 961{,}8126 & 349{,}6397 \\ 0 & 0 & 1 \end{bmatrix}$ |
| Parâmetros Intrínsecos da Câmera Dir. (ver tópico 2.1.2.2) | $\begin{bmatrix} 969{,}9328 & 0 & 654{,}2187 \\ 0 & 961{,}8126 & 357{,}4234 \\ 0 & 0 & 1 \end{bmatrix}$ |
| Coeficientes de Distorções da Câmera Esq. (ver tópico 2.1.2.2) | K1 = 0,04782449<br>K2 = -0,23720237<br>K3 = -0,22979700 |
| Coeficientes de Distorções da Câmera Dir. (ver tópico 2.1.2.2) | K1 = 0,05210654<br>K2 = -0,27168099<br>K3 = -0,30571211 |
| Matriz de Rotação (**R**) (descrita nos tópicos 2.2.1 e 2.2.2) | $\begin{bmatrix} 0{,}9999 & -0{,}0002 & 0{,}0136 \\ 0{,}0002 & 0{,}9999 & 0{,}0004 \\ -0{,}0136 & -0{,}0004 & 0{,}9999 \end{bmatrix}$ |
| Vetor de Translação (**T**) (descrita nos tópicos 2.2.1 e 2.2.2) | $\begin{bmatrix} -9{,}9757 \\ -0.0317 \\ -0.0884 \end{bmatrix}$ |
| Matriz Essencial (**E**) (descrita no tópico 2.2.1.1) | $\begin{bmatrix} 4{,}5197e-04 & 8{,}8430e-02 & -3{,}1731e-02 \\ -2{,}2449e-01 & -4{,}7860e-03 & 9{,}9736 \\ 2{,}9690e-02 & -9{,}9757 & -4{,}4004e-03 \end{bmatrix}$ |
| Matriz Fundamental (**F**) (descrita no tópico 2.2.1.2) | $\begin{bmatrix} -4.1253e-09 & -8{,}1394e-07 & 5{,}6814e-04 \\ 2{,}0663e-06 & -4{,}4424e-08 & -9{,}0381e-02 \\ -9{,}9871e-04 & 8{,}9576e-02 & 1 \end{bmatrix}$ |
| Erro de reprojeção | 0,481268 |

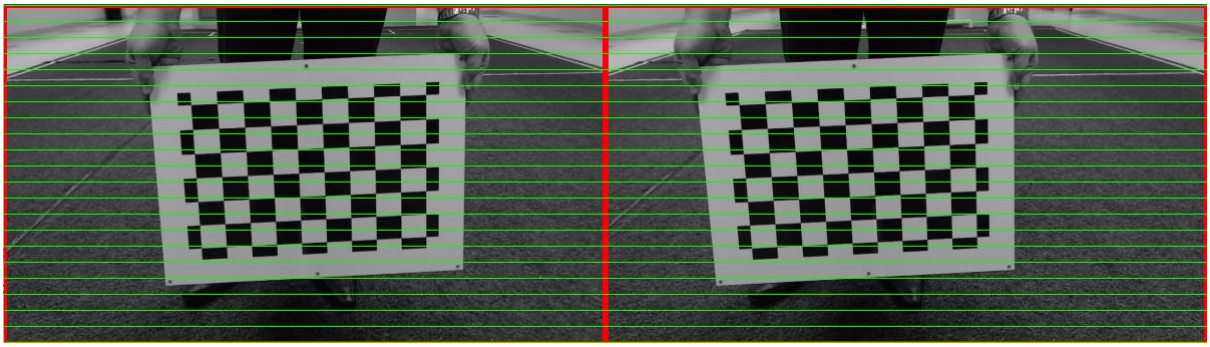
Fonte: Autor

Analisando o vetor de Translação (**T**), são notáveis os valores de deslocamento das câmeras do sistema, em centímetros. O valor -9,9757 cm representa o deslocamento do sistema na coordenada X (*Baseline*), cujo valor teórico é -10 cm. Os demais valores representam o pequeno deslocamento entre câmeras tanto na coordenada Y (-0.0317 cm) como na coordenada Z (-0.0884 cm) do sistema estéreo.

Analisando o valor do erro de reprojeção, que representa o erro RMS entre os vértices identificados do tabuleiro de xadrez e os vértices reprojetados através das matrizes dos parâmetros intrínsecos e extrínsecos das câmeras, observamos um valor de 0,48 px. Considerando que a resolução utilizada é HD (1280x720 px), um erro abaixo de 0,5 px é satisfatório.

A figura 31 apresenta o resultado do processo de retificação abordado no tópico 4.2.2.

Figura 31 - Par de imagens retificado



Fonte: Autor

## 5.2 CAPTURA DE IMAGENS DE UM CENÁRIO REAL

A segunda fase do experimento foi a criação de um banco de imagens de um cenário real, isto é, cenas contendo um campo de futebol semelhante ao utilizado na competição da RoboCup. A importância do cenário real é a análise da influência da textura da grama do campo na geração do mapa de disparidade. A figura 32 apresenta o ambiente do cenário real localizado no Centro Universitário FEI que foi utilizado para os experimentos deste trabalho.

O tripé contendo o sistema estéreo, conforme ilustrado na figura 27, foi posicionado no centro da linha de fundo do campo. A altura do sistema estéreo foi fixada em 85 cm, pois é a aproximadamente a altura que será utilizada na próxima geração de robôs *KidSize* do time RoboFEI. O sistema foi levemente inclinado ao solo a fim de obter a maior visibilidade possível do campo. A figura 33 apresenta o posicionamento do sistema estéreo no campo e a figura 34 apresenta um par de imagem capturado pelo sistema estéreo nesta posição.

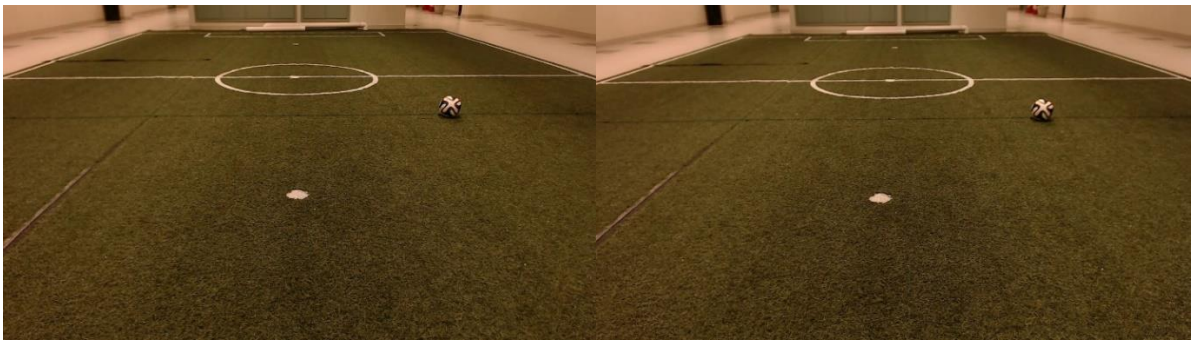
Figura 32 – Ambiente de um cenário real

Fonte: Autor

Figura 33 – Posicionamento do sistema estéreo no campo

Fonte: Autor

Figura 34 – Par de imagens capturado pelo sistema estéreo



Fonte: Autor

Os parâmetros de cada uma das câmeras do sistema estéreo foram configurados individualmente com os mesmos valores, evitando possíveis diferenças entre ambas imagens do par e, consequentemente, evitando possíveis ruídos no mapa de disparidade. A tabela 4 apresenta os parâmetros configurados nas câmeras e seus respectivos valores.

Nos primeiros testes, ocorreram sérias divergências no mapa de disparidade pelo motivo das câmeras estarem com os parâmetros configurados com diferentes valores. Estes parâmetros foram configurados através da biblioteca *v4l2-ctl*, disponível para o sistema operacional Linux. Posteriormente foram acrescentadas as atribuições de valores dos parâmetros no próprio código do software (online e off-line) de maneira que toda vez que for realizar uma captura, as câmeras estarão obrigatoriamente com os mesmos valores apresentados na tabela 4.

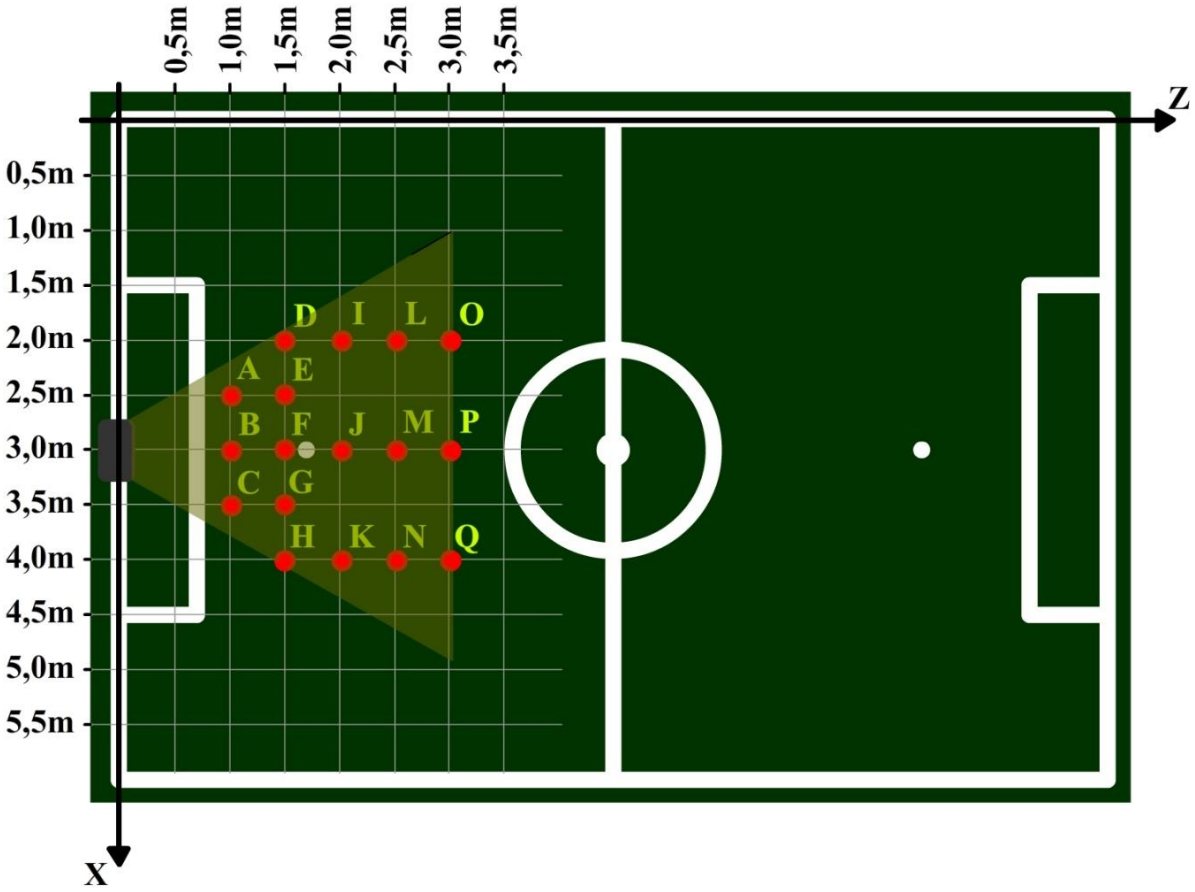
Com os parâmetros das câmeras configurados com os mesmos valores, iniciou-se a captura da base de imagens. Para isso, foram eleitos 17 pontos no campo, sendo os pontos A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P e Q, conforme ilustrados na figura 35 e descritos na tabela 5. Para cada um destes pontos, foram capturadas 30 pares de imagens, sendo 10 pares para cada uma das três diferentes bolas utilizadas neste trabalho. Para cada um dos 10 pares de imagens da mesma bola na mesma posição no campo, foi alterada a posição da bola para que o sistema capturasse uma visão diferente de sua textura.

Tabela 4 – Configuração dos parâmetros das câmeras do sistema estéreo

| **Parâmetro** | **Valor atribuído** |
|---|---|
| Brightness | 128 |
| Contrast | 128 |
| Saturation | 128 |
| White Balance Temperature Auto | 0 |
| Gain | 0 |
| Power Line Frequency | 0 |
| White Balance Temperature | 6500 |
| Sharpness | 128 |
| Backlight Compensation | 0 |
| Exposure Auto | 1 |
| Pan Absolute | 0 |
| Tilt Absolute | 0 |
| Focus Auto | 0 |
| Focus Absolute | 1 |
| Zoom Absolute | 100 |

Fonte: Autor

Figura 35 – Pontos para captura de imagens no campo




Fonte: Autor

A distância total representada na tabela 5 indica a distância entre o sistema estéreo e a bola. Para realizar este cálculo, foi utilizado

$$Dist_{Total} = \sqrt{(300 - POS_X)^2 + {POS_Z}^2 + {POS_Y}^2} \quad , \qquad \textbf{(48)}$$

onde a diferença (300 – $POS_X$) representa a distância entre a câmera posicionada no centro do campo e a bola, no eixo X.

Tabela 5 – Pontos para captura de imagens no campo

| Ponto | Posição em Z(cm) | Posição em X(cm) | Posição em Y(cm) | Distância total(cm) |
|---|---|---|---|---|
| A | 100 | 250 | 85 | 140,4 |
| B | 100 | 300 | 85 | 131,2 |
| C | 100 | 350 | 85 | 140,4 |
| D | 150 | 200 | 85 | 199,3 |
| E | 150 | 250 | 85 | 179,5 |
| F | 150 | 300 | 85 | 172,4 |
| G | 150 | 350 | 85 | 179,5 |
| H | 150 | 400 | 85 | 199,3 |
| I | 200 | 200 | 85 | 239,2 |
| J | 200 | 300 | 85 | 217,3 |
| K | 200 | 400 | 85 | 239,2 |
| L | 250 | 200 | 85 | 282,4 |
| M | 250 | 300 | 85 | 264,1 |
| N | 250 | 400 | 85 | 282,4 |
| O | 300 | 200 | 85 | 327,5 |
| P | 300 | 300 | 85 | 311,8 |
| Q | 300 | 400 | 85 | 327,5 |

Fonte: Autor

A figura 36 apresenta os três modelos de bolas utilizados neste trabalho.

Figura 36 - (a) Bola A. (b) Bola B. (c) Bola C.

(a) (b) (c)

Fonte: Autor

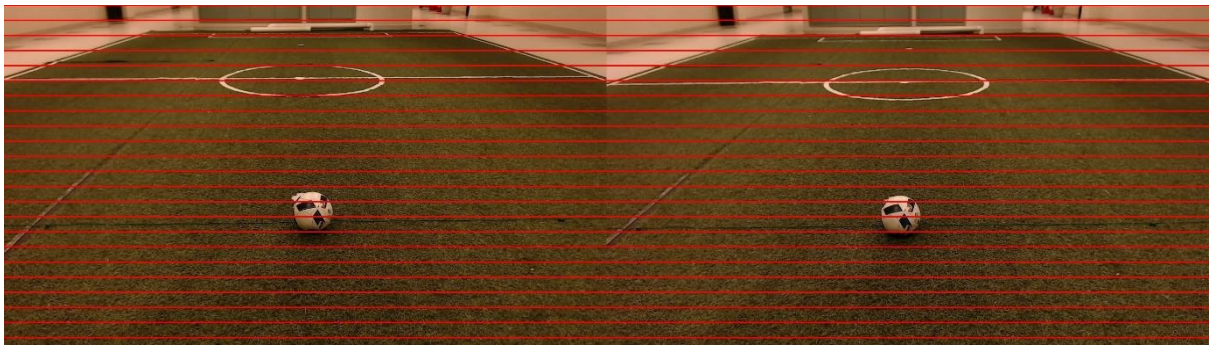
## 5.3 RECONSTRUÇÃO 3D DE UMA CENA

Para realizar a reconstrução 3D de uma cena, são necessários os seguintes passos: aplicação dos valores das matrizes de calibração e retificação, geração do mapa de disparidade, cálculo da triangulação e criação da *Point Cloud*. A seguir, são apresentados estes passos com seus respectivos resultados.

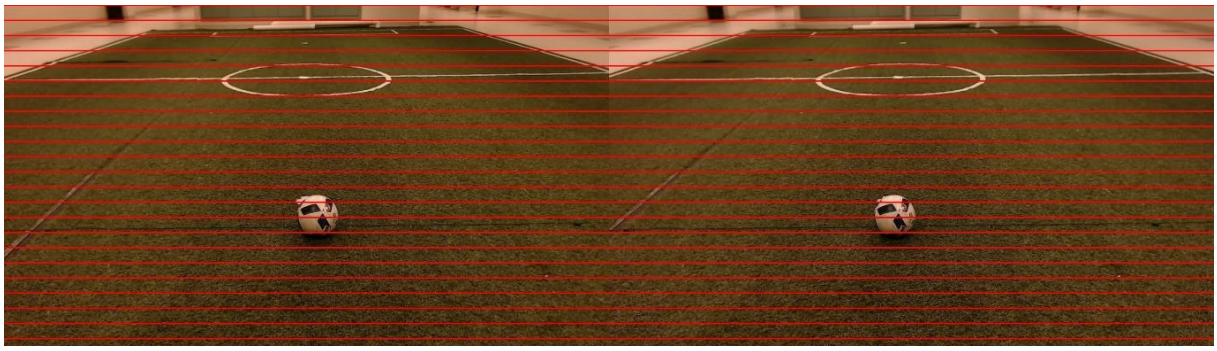
### 5.3.1 **Aplicação das matrizes de calibração e retificação em um cenário real**

Este processo consiste em ler os valores das matrizes salvas em um arquivo previamente pelo modo *off-line*, alocar estes valores em variáveis e passá-las como parâmetro nas funções responsáveis pela retificação de imagens do OpenCV (ver tópico 4.2.3). A figura 37 ilustra a comparação entre um par de imagens sem retificação e um par de imagens retificado, sendo ambos pares de um cenário real. O tempo de processamento para executar esta tarefa é de aproximadamente 0,5 ms.

Figura 37 - (a) Par de imagens sem retificação. (b) Par de imagens retificado



(a)



(b)

Fonte: Autor

### 5.3.2 **Geração do mapa de disparidade**

Com o par de imagem retificado, o próximo passo é a geração do mapa de disparidade descrito no tópico 2.2.4. O mapa de disparidade é gerado utilizando o algoritmo *Semi Global Stereo Matching* (SGBM) do OpenCV (ver tópico 4.2.4), com os valores dos parâmetros de configuração apresentados na tabela 6. Estes valores foram obtidos experimentalmente, proporcionando mapas de disparidade com o menor índice de ruídos. A figura 38 apresenta o resultado obtido com o par de imagens apresentado na figura 37 (b). O tempo de processamento para gerar o resultado da figura 38 foi de aproximadamente 5.200 ms, porém este tempo varia de acordo com o tamanho do par de imagens e com os valores dos parâmetros do SGBM.

Tabela 6 – Valores utilizados nos parâmetros do SGBM

| **Parâmetro** | **Valor atribuído** |
|---|---|
| preFilterCap | 10 |
| SADWindowSize | 1 |
| P1 | 24 |
| P2 | 96 |
| numberOfDisparities | 96 |
| uniquenessRatio | 1 |

Fonte: Autor

Figura 38 – Resultado do mapa de disparidade da figura 37 (b).

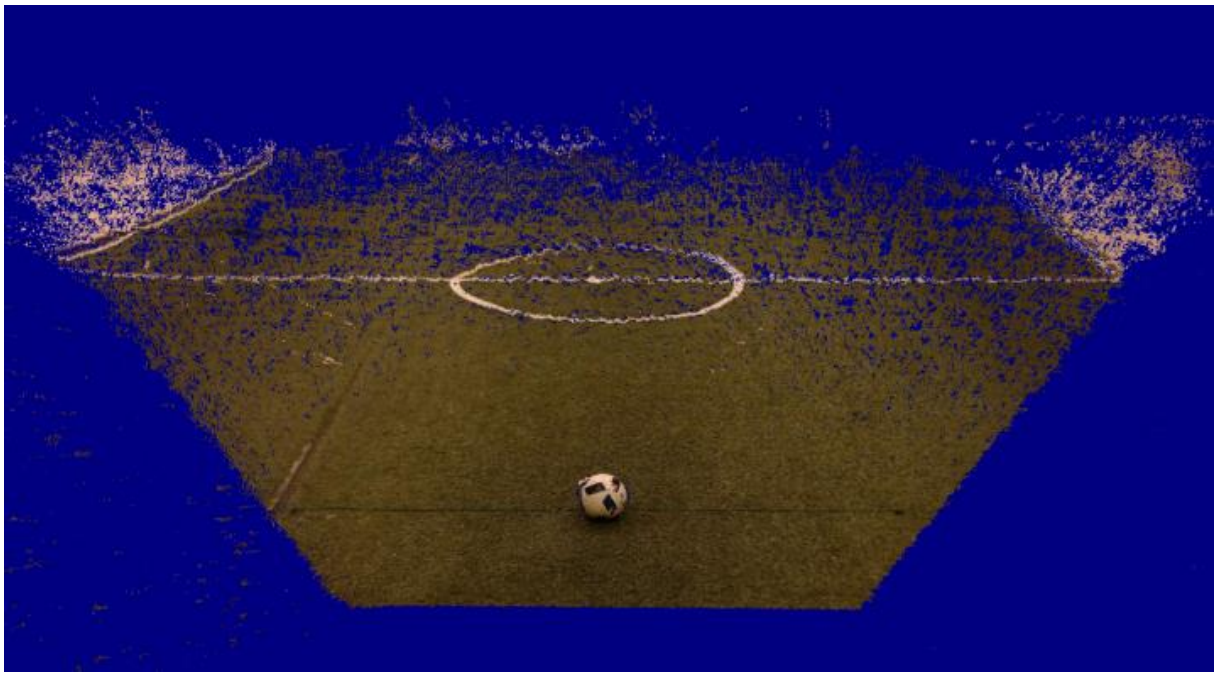
Fonte: Autor

### 5.3.3 **Cálculo da triangulação e criação da *Point Cloud***

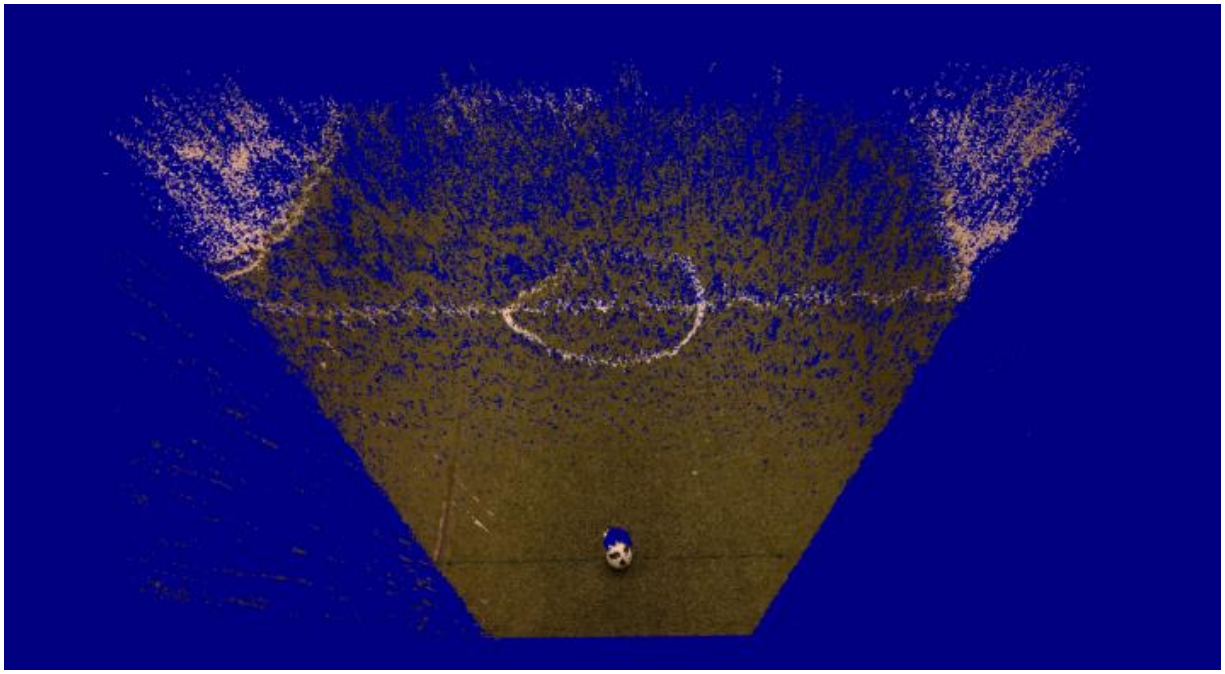
Com o mapa de disparidade gerado, é realizado o cálculo da triangulação conforme descrito no tópico 4.2.5. Após obter os valores de profundidade do par de imagens, é criado, através da biblioteca PCL, um objeto *PointCloud* com as mesmas dimensões da imagem estéreo. Os pontos deste objeto recebem os valores das posições X e Y da imagem esquerda do par de imagens estéreo e os valores da posição Z calculados pela triangulação.

Um objeto *PointCloud* pode ser do tipo que contém apenas a informação da posição XYZ de cada ponto ou do tipo que, além da posição, contém a informação sobre sua respectiva cor RGB. Neste trabalho, criamos dois objetos *PointCloud*, sendo um com informação de cor, utilizado apenas para exibição da cena, e outro sem a informação de cor, utilizado no processamento do reconhecimento da bola. A figura 39 apresenta diversas vistas da *Point Cloud* gerada com base na cena apresentada na figura 37 (b).

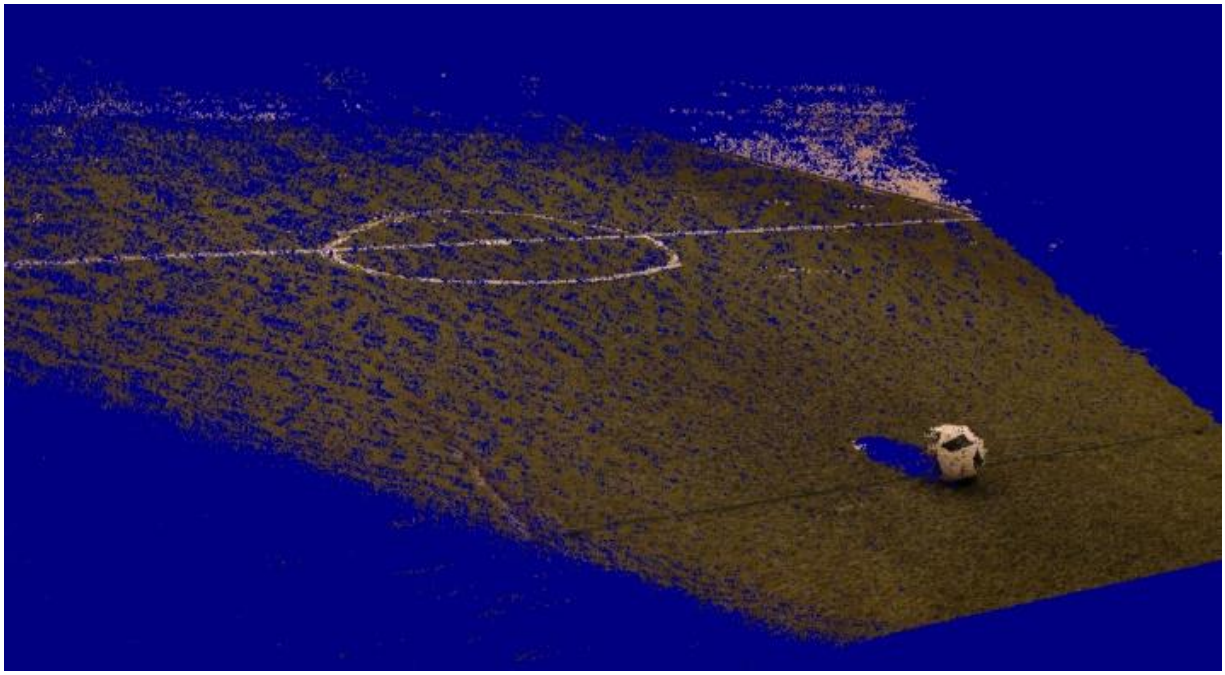
Figura 39 –*Point Cloud* da cena ilustrada na figura 37 (b).



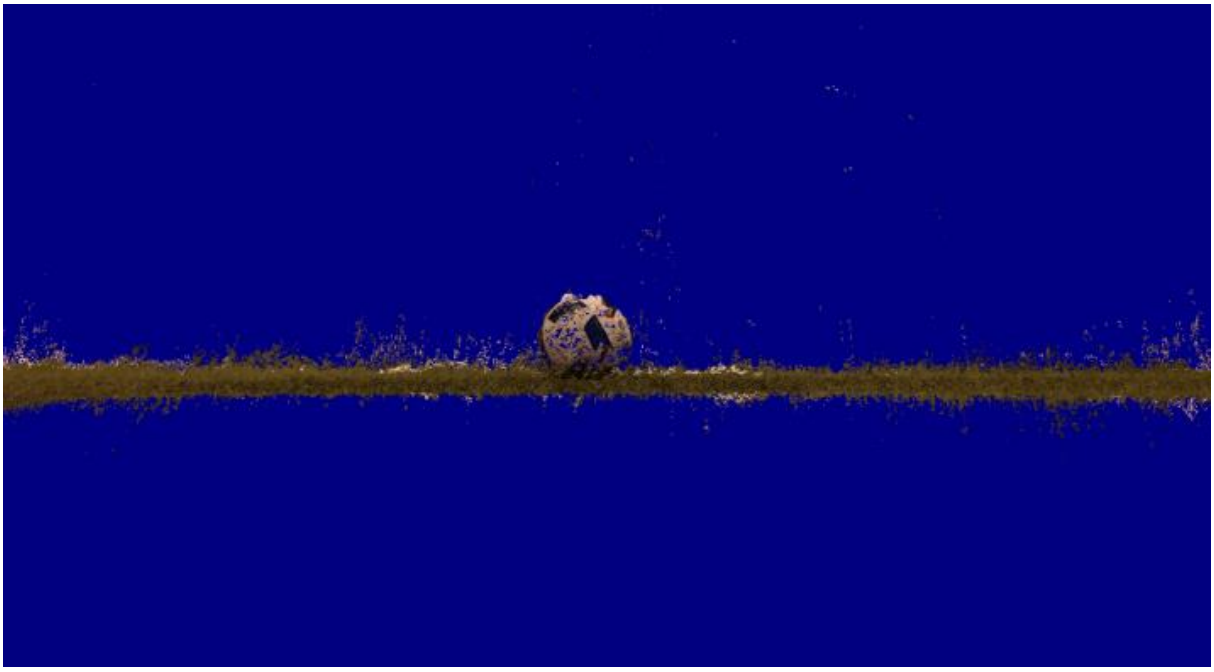
(a)



(b)



(c)



(d)

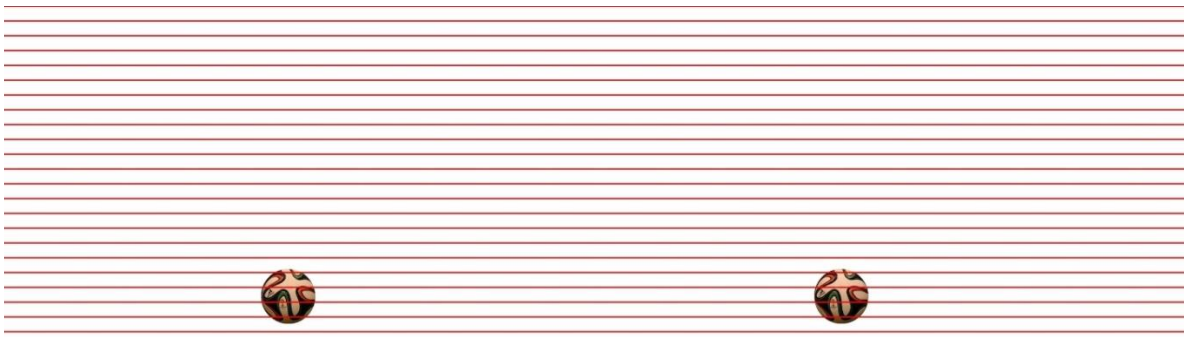
Fonte: Autor

## 5.4 CRIAÇÃO DO MODELO DA BOLA

Para criar o modelo da bola, o primeiro passo foi capturar um par de imagens contendo a bola. Este par de imagem foi editado para que apresentasse somente a bola, desconsiderando o resto da cena, mas mantendo o tamanho original e o posicionamento da bola em ambas as imagens. Com o par de imagens editado, foi realizado o procedimento descrito no tópico 5.3.1 (Aplicação das matrizes de calibração e retificação), no tópico 5.3.2 (Geração do mapa de disparidade) e no tópico 5.3.3 (Cálculo da triangulação e criação da *Point Cloud*). A figura 40 apresenta o par de imagens original (a), o par editado e retificado (b), e a *Point Cloud* gerada para o modelo da bola com informação de cor (c) e sem informação de cor (d).

Figura 40 – (a) Par de imagens original utilizado para extrair o modelo da bola. (b) Par de imagens editado e retificado. (c) *Point Cloud* colorida do modelo da bola. (d) *Point Cloud* do modelo da bola sem informação de cor.

(a)



(b)

(c) (d)

Fonte: Autor

### 5.4.1 **Extração dos pontos de interesse (*keypoints*) para o modelo da bola**

Após a geração da *Point Cloud* do modelo da bola, o próximo passo foi obter a *Point Cloud ISS*, nome utilizado para identificar a *Point Cloud* que contém apenas os *keypoints*, conforme descrito no tópico 4.2.6. Para extrair os *keypoints* da *Point Cloud* original, os parâmetros do algoritmo *ISS* foram configurados conforme a tabela 7.

Tabela 7 – Valores utilizados nos parâmetros do *ISS*

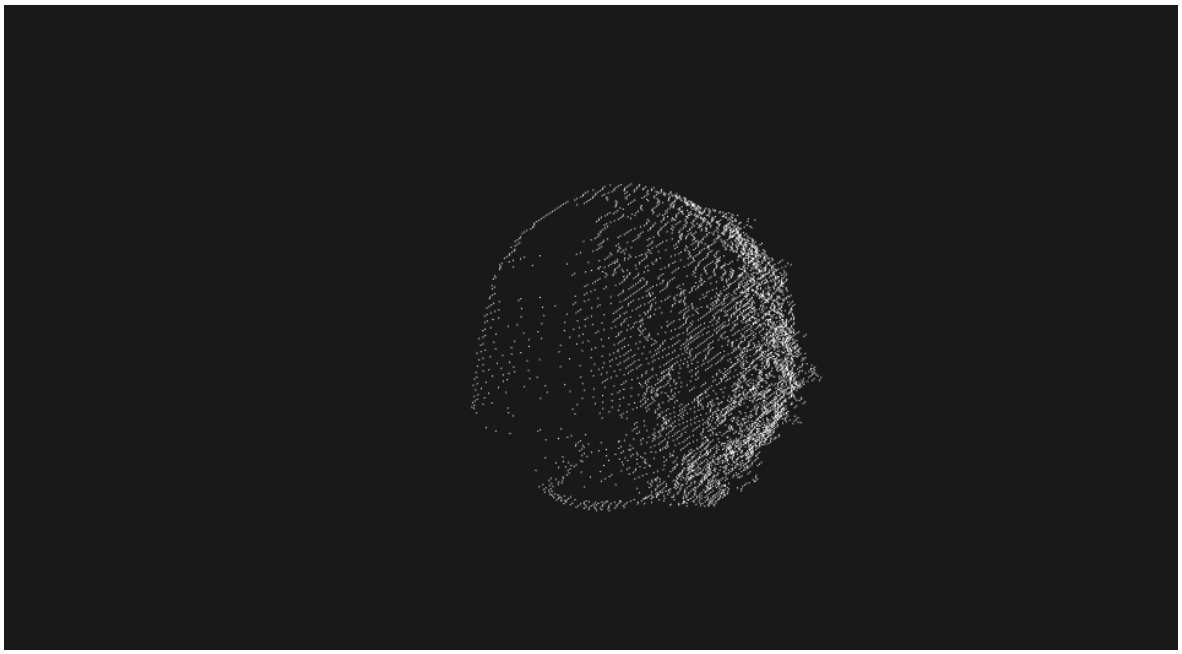
| **Parâmetro** | **Valor atribuído** |
|---|---|
| setSearchMethod | KdTree |
| setSalientRadius | 0.05 |
| setNonMaxRadius | 0.03 |
| setThreshold21 | 0.975 |
| setThreshold32 | 0.975 |
| setMinNeighbors | 1 |
| setNumberOfThreads | 4 |

Fonte: Autor

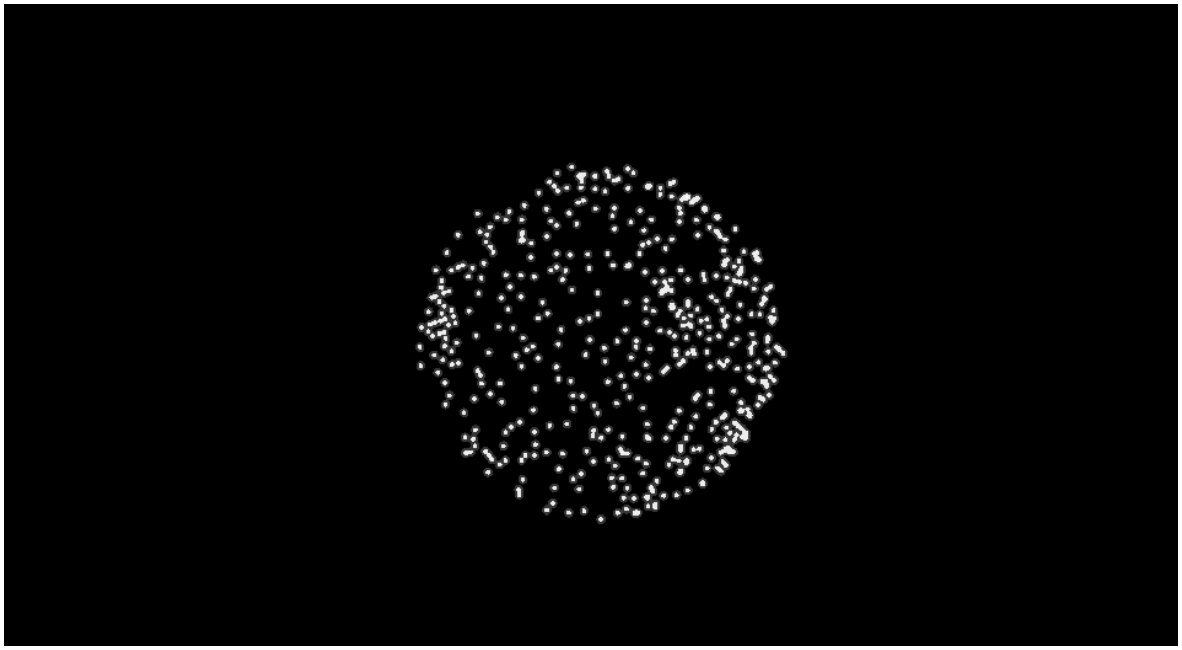
Os valores da tabela 7 foram obtidos experimentalmente, buscando a eliminação dos *outliers* e mantendo uma quantidade de *keypoints* suficiente para manter as características da *Point Cloud* original. A *Point Cloud* do modelo da bola contém 5.489 pontos e a respectiva *Point Cloud ISS* possui apenas 482 pontos, ou seja, houve uma redução de aproximadamente 91,22% na quantidade de pontos e, consequentemente, uma considerável redução nos processamentos das próximas operações realizadas com a *Point Cloud ISS*. A figura 41 apresenta a *Point Cloud* do modelo da bola com todos os pontos (a) e a *Point Cloud ISS* (b).

A fim de manter a repetitividade na extração dos pontos, foram utilizados os mesmos valores de configuração descritos na tabela 7, tanto para a *Point Cloud* do modelo da bola como para a *Point Cloud* da cena utilizada no reconhecimento da bola.

Figura 41 – (a) *Point Cloud* com todos os pontos. (b) *Point Cloud ISS*.



(a)



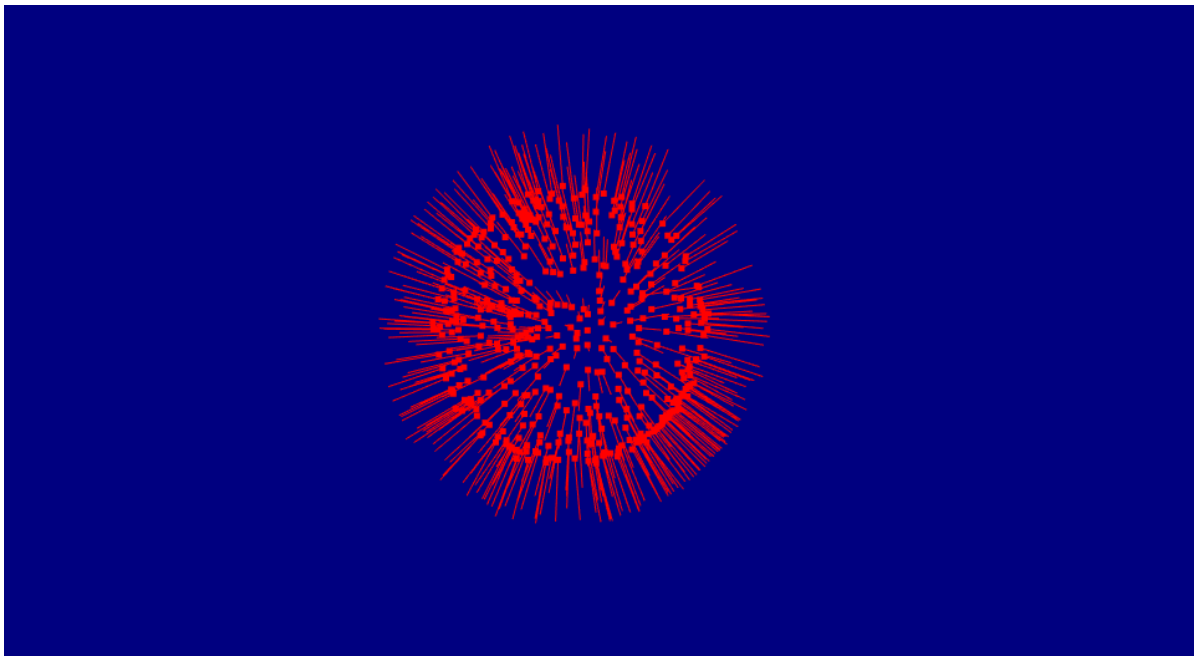
(b)

Fonte: Autor

### 5.4.2 **Cálculo das normais de superfície para o modelo da bola**

Com a geração da *Point Cloud ISS* do modelo da bola realizada, são calculadas as normais de superfície para cada ponto, considerando o raio de vizinhança abordado no tópico 4.2.7 com o valor de 0,40. Este valor foi obtido experimentalmente e também é utilizado no processo para o reconhecimento da bola. A figura 42 apresenta a *Point Cloud ISS* do modelo da bola com o resultado do cálculo das normais de superfície.

Figura 42 – *Point Cloud ISS* do modelo da bola com as normais de superfície.



Fonte: Autor

### 5.4.3 **Cálculo dos descritores para o modelo da bola**

Com as normais de superfície calculadas, o último passo para a criação do modelo da bola é o cálculo dos descritores, abordado no tópico 4.2.8.

Neste trabalho foram utilizados três diferentes métodos descritores: o FPFH, o SHOT e o 3DSC. Portanto, foram gerados três mapas de descrição para os pontos do modelo da bola, sendo um mapa para cada método descritor.

Com o objetivo de gerar uma comparação de eficiência entre os métodos, foi adotado o mesmo raio de vizinhança para os métodos FPFH e SHOT, com o valor de 0,45. Para o método 3DSC operar, foi necessário ajustar o valor do raio de vizinhança para 0,675 e configurar os parâmetros "*setMinimalRadius*" e "*setPointDensityRadius*" com os valores 0,0675 e 0,135, respectivamente. Tanto o valor de raio de vizinhança aplicado nos métodos FPFH e SHOT, como os valores dos parâmetros aplicados no método 3DSC, foram obtidos experimentalmente e também foram utilizados no processo de reconhecimento da bola.

Os três mapas de descrição do modelo da bola foram salvos em arquivos diferentes e, durante o processo de reconhecimento da bola, é carregado o mapa de descrições correspondente ao método que está sendo utilizado.

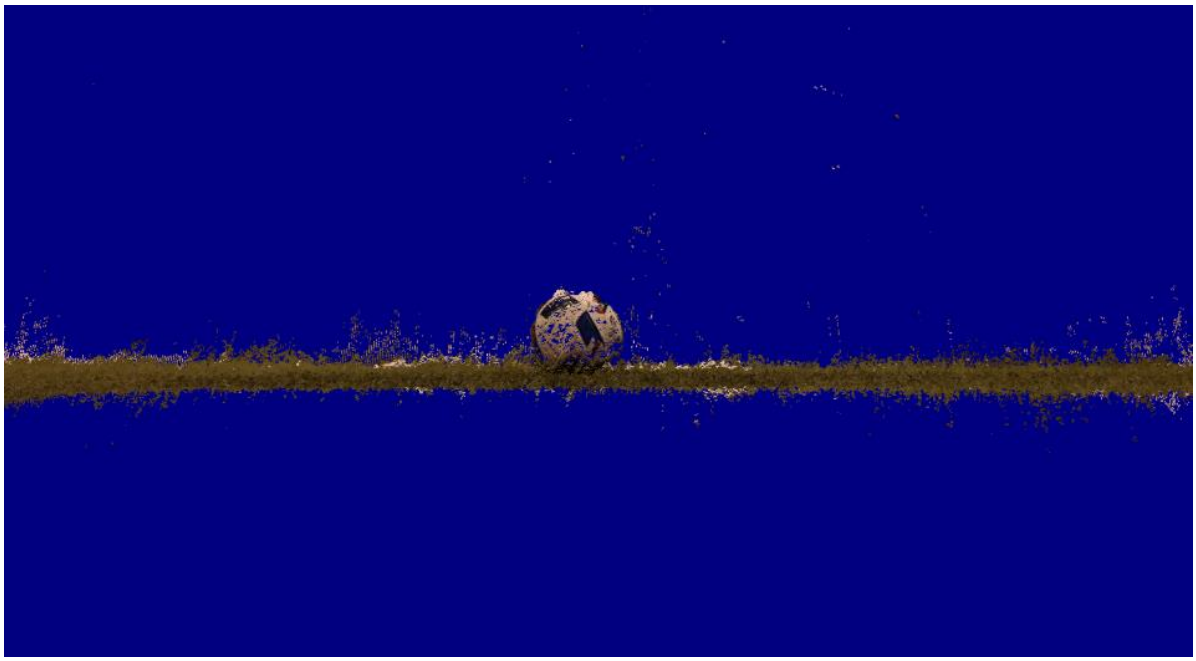
## 5.5 RECONHECIMENTO DA BOLA E ESTIMATIVA DA DISTÂNCIA

Com o modelo da bola criado, o sistema é capaz de realizar o reconhecimento da bola e estimar sua respectiva distância seguindo o mesmo procedimento utilizado para a criação do modelo da bola, ou seja, dado um par de imagem de um cenário real contendo a bola, são aplicadas as matrizes de calibração e retificação (ver tópico 5.3.1), é gerado o mapa de disparidade (ver tópico 5.3.2) e é realizado a reconstrução 3D da cena (ver tópico 5.3.3). A figura 43 apresenta o resultado destes processos em uma *Point Cloud* colorida (a) e uma *Point Cloud* sem informação de cor (b) da mesma cena representada pela *Point Cloud* da figura 39.

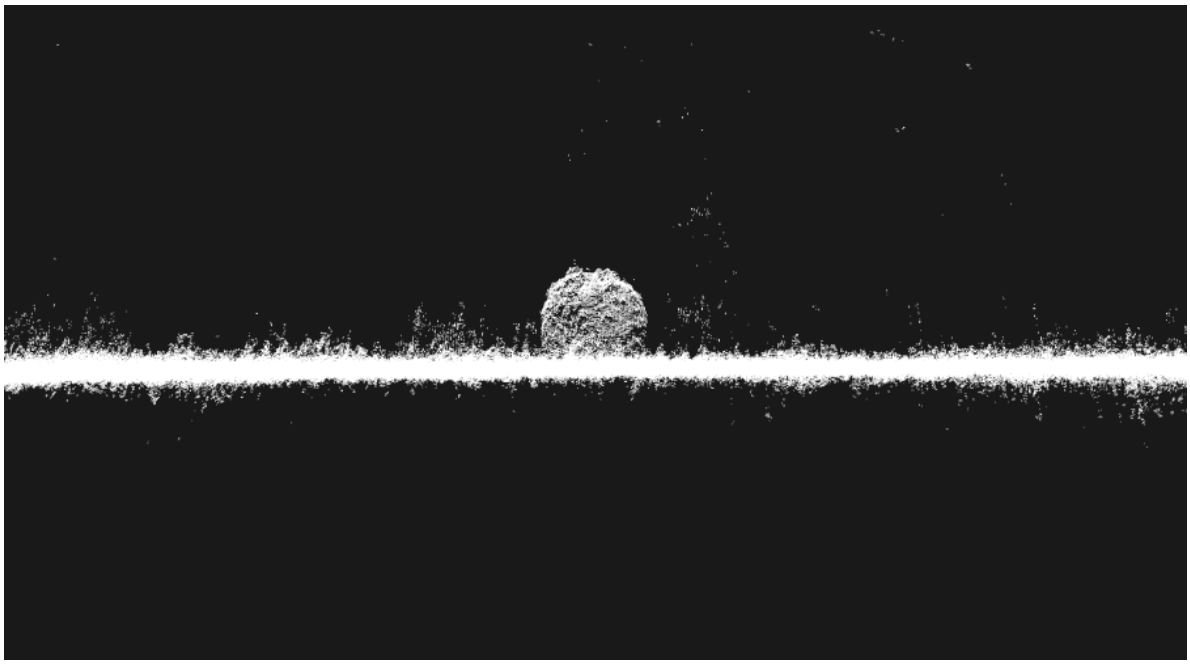
### 5.5.1 **Extração dos pontos de interesse (*keypoints*) para o reconhecimento da bola**

A seguir, é gerada a *Point Cloud ISS*, descrita no tópico 4.2.6, a partir da *Point Cloud* da cena apresentada na figura 43. Neste procedimento foram utilizados os mesmos parâmetros de configuração do método ISS utilizados para a criação do modelo da bola, mencionados no tópico 5.4.1.

Figura 43 – *Point Cloud* de um cenário real contendo a bola. (a) *Point Cloud* colorido. (b) *Point Clout* sem informação de cor.



(a)



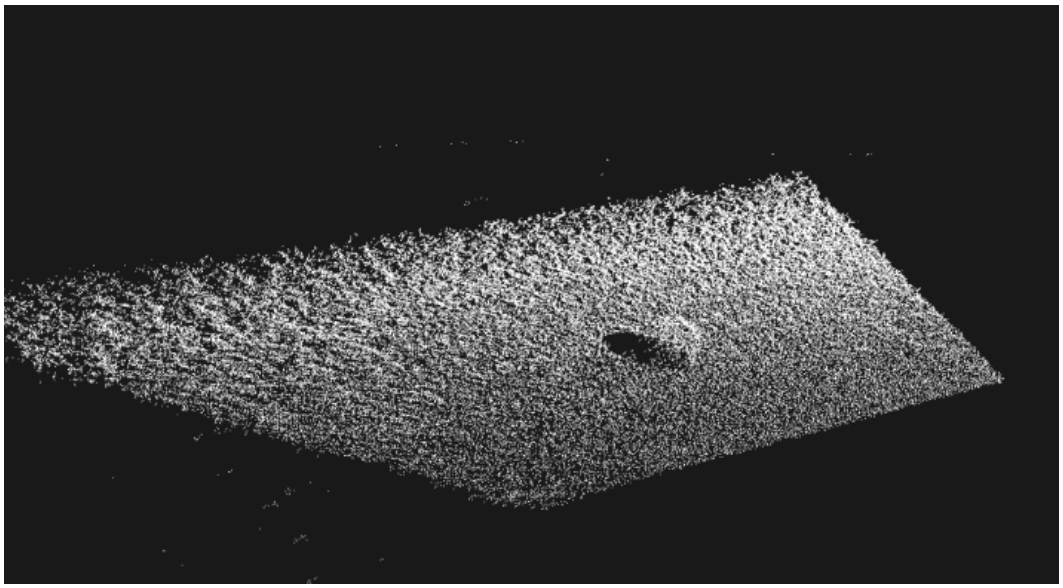
(b)

Fonte: Autor

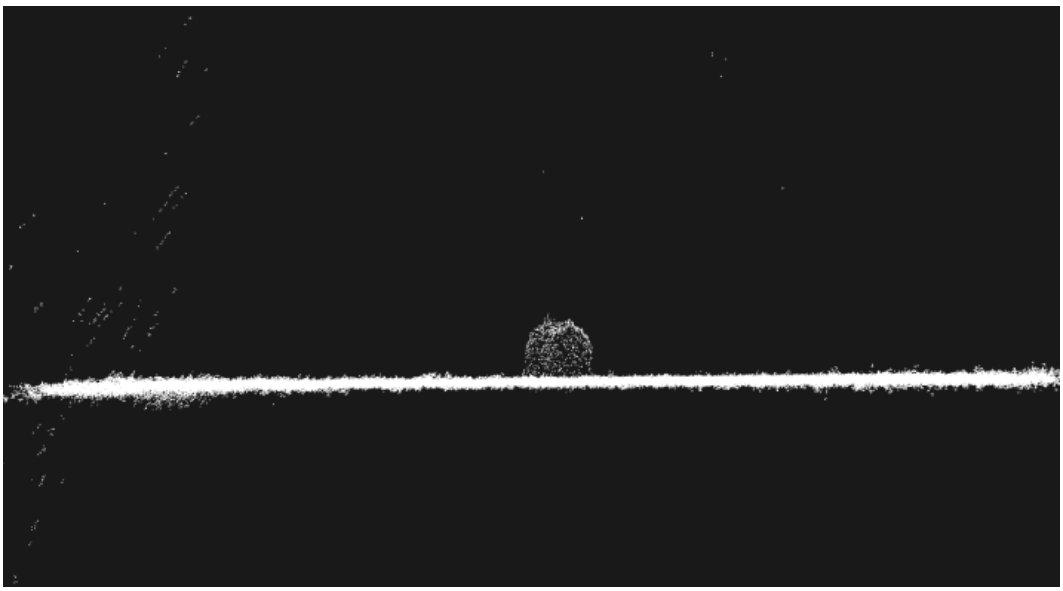
Para realizar este processo, leva-se aproximadamente 9,5 s, porém, a *Point Cloud* original da cena, representada na figura 43, que contém 770.626 pontos, é otimizada para um valor muito menor na *Point Cloud ISS*, que possui um total de 84.347 pontos.

A figura 44 apresenta duas diferentes vistas da *Point Cloud ISS* para o reconhecimento da bola.

Figura 44 – *Point Cloud ISS* para reconhecimento da bola.



(a)



(b)

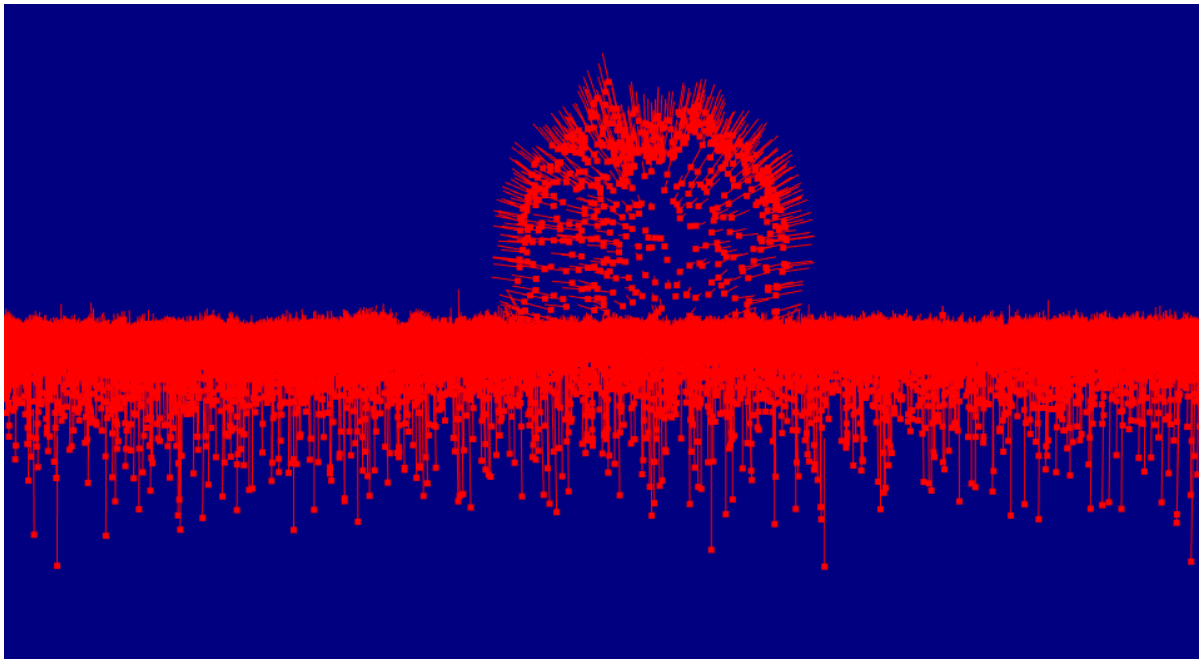
Fonte: Autor

### 5.5.2 **Cálculo das normais de superfície para o reconhecimento da bola.**

Com a *Point Cloud ISS* da cena gerada, é realizado o cálculo das normais de superfície utilizando o mesmo procedimento e valores de configuração abordados no tópico 5.4.2. O tempo aproximado para a execução deste processo é em torno de 1,5 s.

A figura 45 apresenta os pontos da *Point Cloud ISS* com os vetores das normais de superfície calculados.

Figura 45 – *Point Cloud ISS* com as normais de superfície.



Fonte: Autor

### 5.5.3 **Cálculo dos descritores e busca de correspondências com o modelo da bola.**

Após o cálculo das normais de superfície para a *Point Cloud ISS* da cena, é gerado o mapa de descrições utilizando os mesmos métodos descritores e os mesmos parâmetros de configuração utilizados para a geração dos modelos da bola (ver tópico 5.4.3). Com o mapa de descrições gerado, é carregado o modelo da bola correspondente ao método descritor utilizado, conforme mencionado no tópico 5.4.3. O tempo gasto neste processo varia de acordo com o método descritor utilizado.

A figura 46 apresenta a *Point Cloud* da cena junto com a *Point Cloud* do modelo da bola.

Figura 46 – *Point Cloud* da cena junto com o *Point Cloud* do modelo da bola.

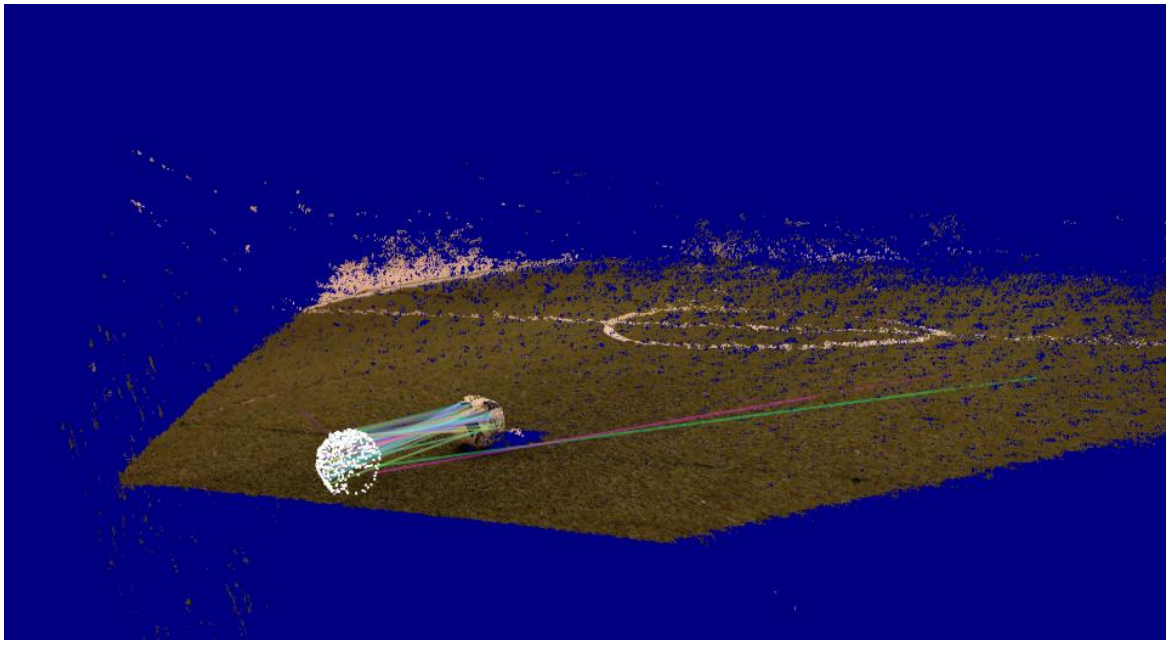
Fonte: Autor

Com os mapas de descrições da cena e do modelo da bola, é realizada a busca de correspondência entre os pontos da cena e do modelo, através dos métodos mencionados no tópico 4.2.10. Como os mapas de descrições são sensíveis às normais de superfície e estas são sensíveis aos *keypoints* extraídos pelo método ISS, é fundamental que o modelo e a cena possuam os mesmos parâmetros de configuração em todo o processo, pois de outra forma, o mapa de descrições do modelo e o mapa de descrições da cena ficarão divergentes, impossibilitando o sucesso das correspondências no processo de busca.

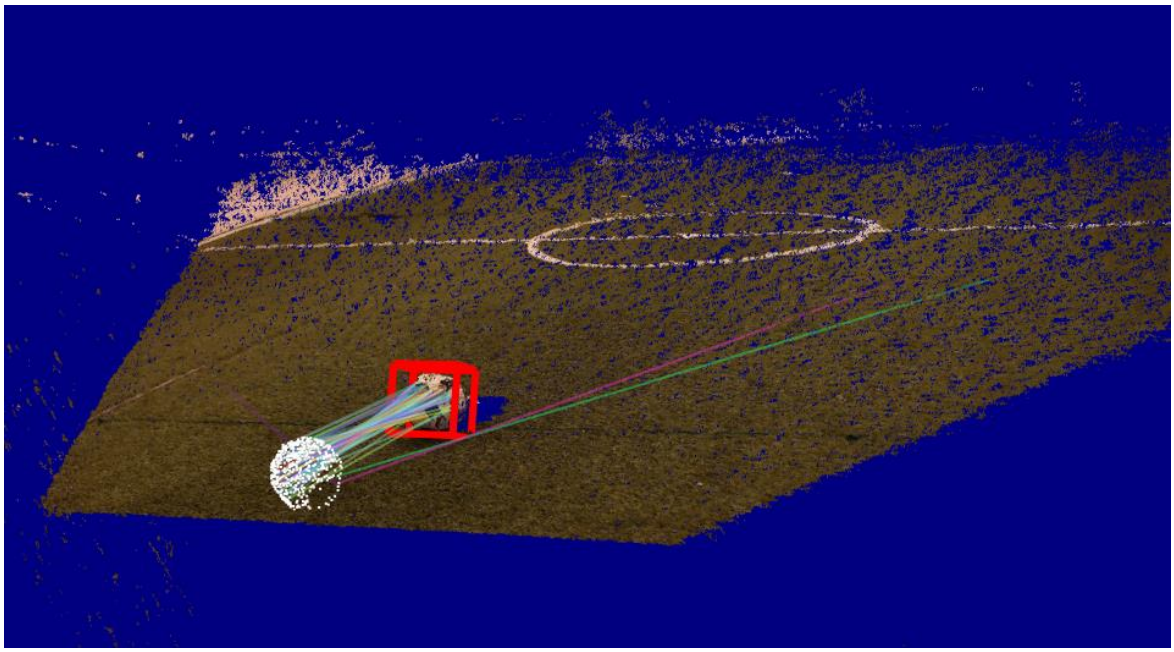
Após identificar todas as correspondências, as mesmas são divididas em diferentes grupos caso ultrapassem a distância limite estipulada em 1.0 (valor na escala de coordenadas da *Point Cloud*). O grupo que contiver o maior número de correspondência, e este número for maior do que 10, é eleito como a localização da bola na cena e um cubo é desenhado em torno deste grupo. A figura 47 apresenta o resultado da busca de correspondência entre o modelo da

bola e a cena (a) e o resultado do grupo eleito com seu respectivo cubo em torno de suas correspondências (b).

Figura 47 – (a) Resultado da correspondência do modelo da bola e a cena. (b) Representação do cubo em torno do grupo com maior número de correspondências.



(a)



(b)

Fonte: Autor

### 5.5.4 **Cálculo da distância estimada da bola.**

Com a bola reconhecida, a distância estimada da bola é calculada com base no centro do cubo, conforme abordado no tópico 4.2.11. Para ajustar a escala das coordenadas X, Y e Z da *Point Cloud*, o valor da distância encontrada entre o centro de projeção da câmera esquerda (utilizada na reconstrução 3D da cena, ver tópico 5.3.3) até o centro do cubo representando a localização da bola encontrada é multiplicado por um fator métrico de valor 16.9 (valor obtido experimentalmente). Conhecendo a distância real da bola, com base na tabela 5 (tópico 5.2), o sistema estima o erro entre a distância real e a distância estimada, gerando o valor de precisão do sistema.

A figura 48 apresenta o resultado do cálculo da distância estimada e a precisão do sistema para uma imagem onde a bola está posicionada no ponto F descrito na tabela 5, ou seja, com distância real de 172,4 cm.

Figura 48 – Resultado do cálculo da distância estimada para uma bola posicionada a uma distância real de 172,4 cm.


Fonte: Autor

## 5.6 COMPORTAMENTO DO SISTEMA NO RECONHECIMENTO DE BOLAS E NA ESTIMATIVA DE SUAS RESPECTIVAS DISTÂNCIAS

A fim de analisar o comportamento do sistema implementado no reconhecimento de três modelos de bolas (figura 36) em diferentes posições, diferentes pontos do campo e com diferentes métodos descritores, foi gerado um banco de imagem com 510 pares de imagens do campo com bola, conforme mencionado no tópico 5.2, e mais 510 pares de imagens do campo sem a bola. Cada par de imagens foi submetido aos processos descritos no tópico 5.5.

Para os resultados que envolvem os 510 pares de imagens do campo com bola, foi considerado o valor médio e o desvio padrão para cada ponto do campo composto por 30 pares de imagens, sendo 10 diferentes pares de imagens para cada um dos três tipos de bola utilizado (ver tópico 5.2). Para os resultados referentes aos pares de imagens do campo sem a bola, foi considerado o valor médio de todo o conjunto (510 pares de imagens) e o respectivo desvio padrão.

### 5.6.1 **Tempo de processamento para o reconhecimento da bola.**

O comparativo do tempo de processamento para o reconhecimento da bola para cada método está representado na tabela 8, para pares de imagens sem bola e na tabela 9, para os pares de imagens com bola e seus respectivos pontos.

Tabela 8 – Tempo de processamento dos pares de imagens sem bola.

| **Descritor FPFH** | | **Descritor SHOT** | | **Descritor 3DSC** | |
|---|---|---|---|---|---|
| **Tempo Médio (s)** | **σ (s)** | **Tempo Médio (s)** | **σ (s)** | **Tempo Médio (s)** | **σ (s)** |
| 24,3 | 0,5 | 35,4 | 0,7 | *625,2 | *34,3 |

* Devido ao elevado tempo, foram processados apenas 60 pares dos 510 disponíveis.

Fonte: Autor

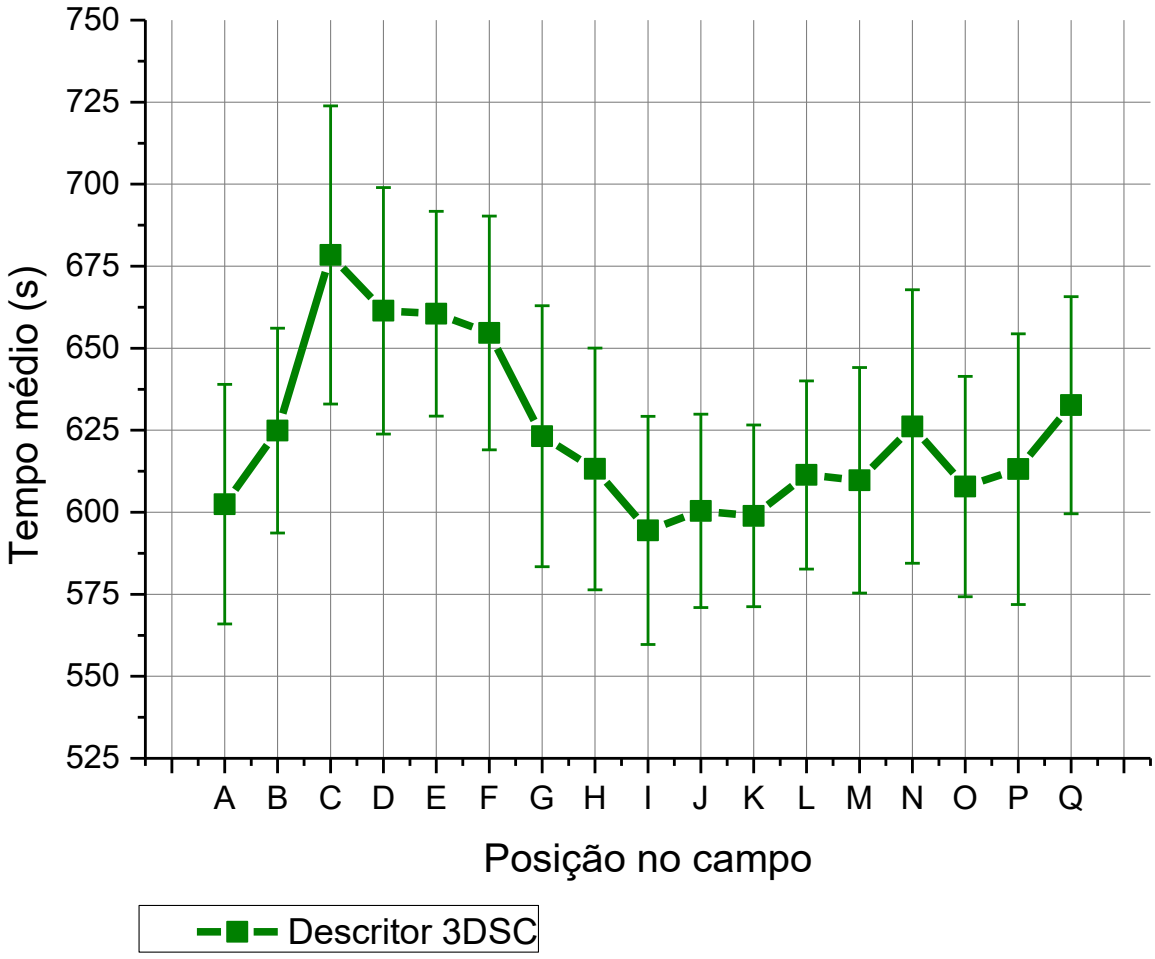
Tabela 9 – Tempo de processamento para pares de imagens com bola.

| Ponto | Descritor FPFH | | Descritor SHOT | | Descritor 3DSC | |
|---|---|---|---|---|---|---|
| | Tempo Médio (s) | σ (s) | Tempo Médio (s) | σ (s) | Tempo Médio (s) | σ (s) |
| A | 24,3 | 0,3 | 27,8 | 0,6 | 602,5 | 36,5 |
| B | 24,4 | 0,4 | 27,4 | 0,8 | 624,9 | 31,2 |
| C | 24,8 | 0,3 | 29,2 | 1,7 | 678,4 | 45,4 |
| D | 24,6 | 0,4 | 29,0 | 1,3 | 661,4 | 37,6 |
| E | 24,5 | 0,3 | 28,1 | 0,8 | 660,5 | 31,2 |
| F | 24,7 | 0,3 | 27,7 | 0,9 | 654,7 | 35,6 |
| G | 24,6 | 0,3 | 28,2 | 1,0 | 623,2 | 39,8 |
| H | 24,7 | 0,3 | 28,7 | 1,0 | 613,2 | 36,8 |
| I | 24,5 | 0,4 | 29,0 | 0,9 | 594,5 | 34,8 |
| J | 24,5 | 0,4 | 29,2 | 1,2 | 600,4 | 29,5 |
| K | 24,6 | 0,4 | 30,2 | 1,0 | 598,9 | 27,7 |
| L | 24,9 | 0,4 | 31,0 | 2,2 | 611,4 | 28,7 |
| M | 24,6 | 0,3 | 30,3 | 1,7 | 609,7 | 34,4 |
| N | 24,9 | 0,4 | 32,2 | 1,4 | 626,1 | 41,7 |
| O | 24,5 | 0,5 | 31,7 | 1,6 | 607,9 | 33,6 |
| P | 24,9 | 0,6 | 31,9 | 1,3 | 613,1 | 41,2 |
| Q | 25,4 | 0,7 | 34,5 | 1,0 | 632,6 | 33,1 |
| **Média Total** | **24,7** | **0,4** | **29,8** | **1,2** | **624,3** | **35,2** |

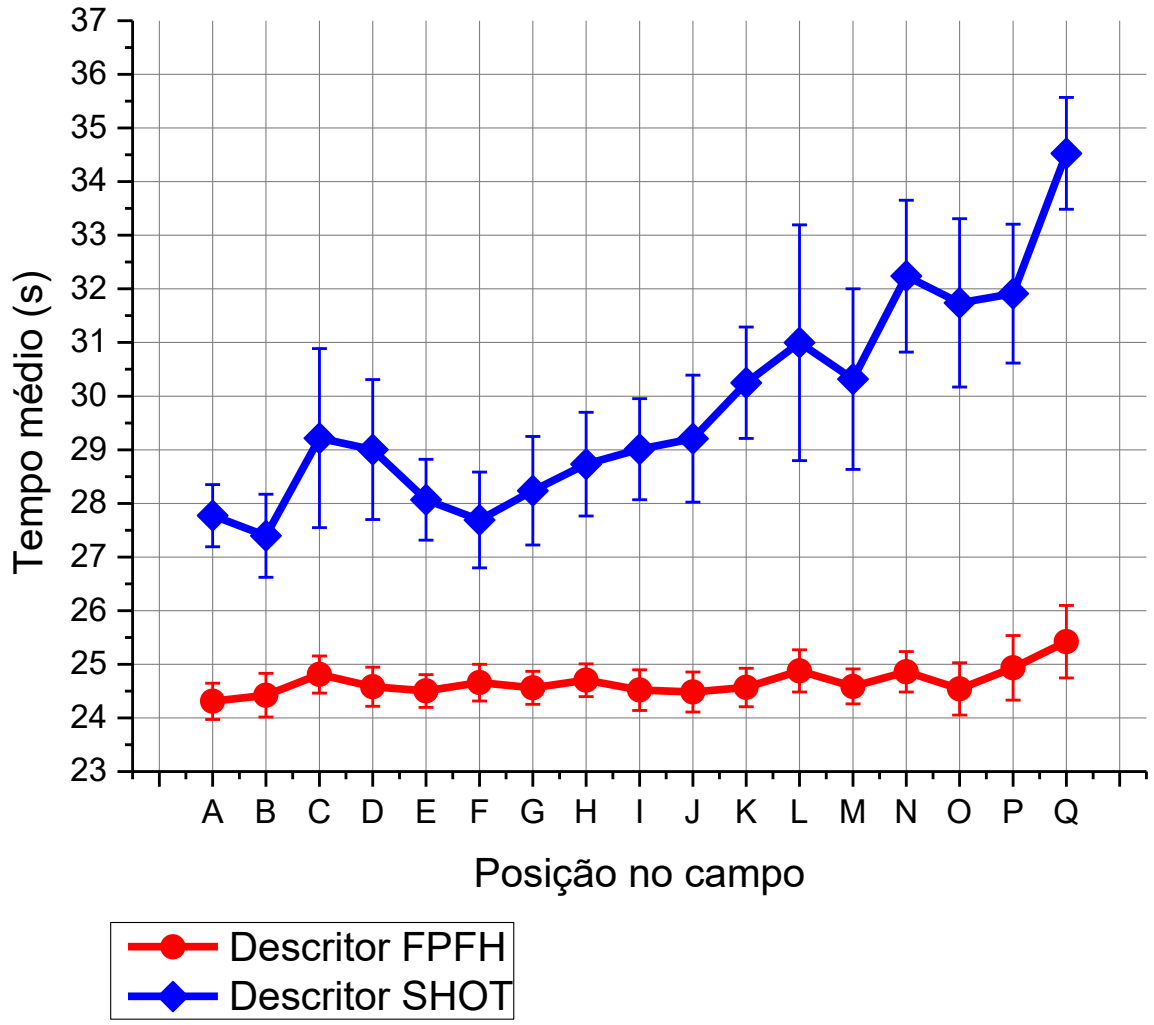
Fonte: Autor

A figura 49 apresenta o gráfico referente aos tempos apresentados na tabela 9 para o descritor 3DSC (a) e para os descritores FPFH e SHOT (b).

Figura 49 – Gráfico do tempo de processamento para pares de imagens com bola. (a) 3DSC. (b) FPFH e SHOT.




(a)




(b)

Fonte: Autor

### 5.6.2 **Distância estimada da bola**

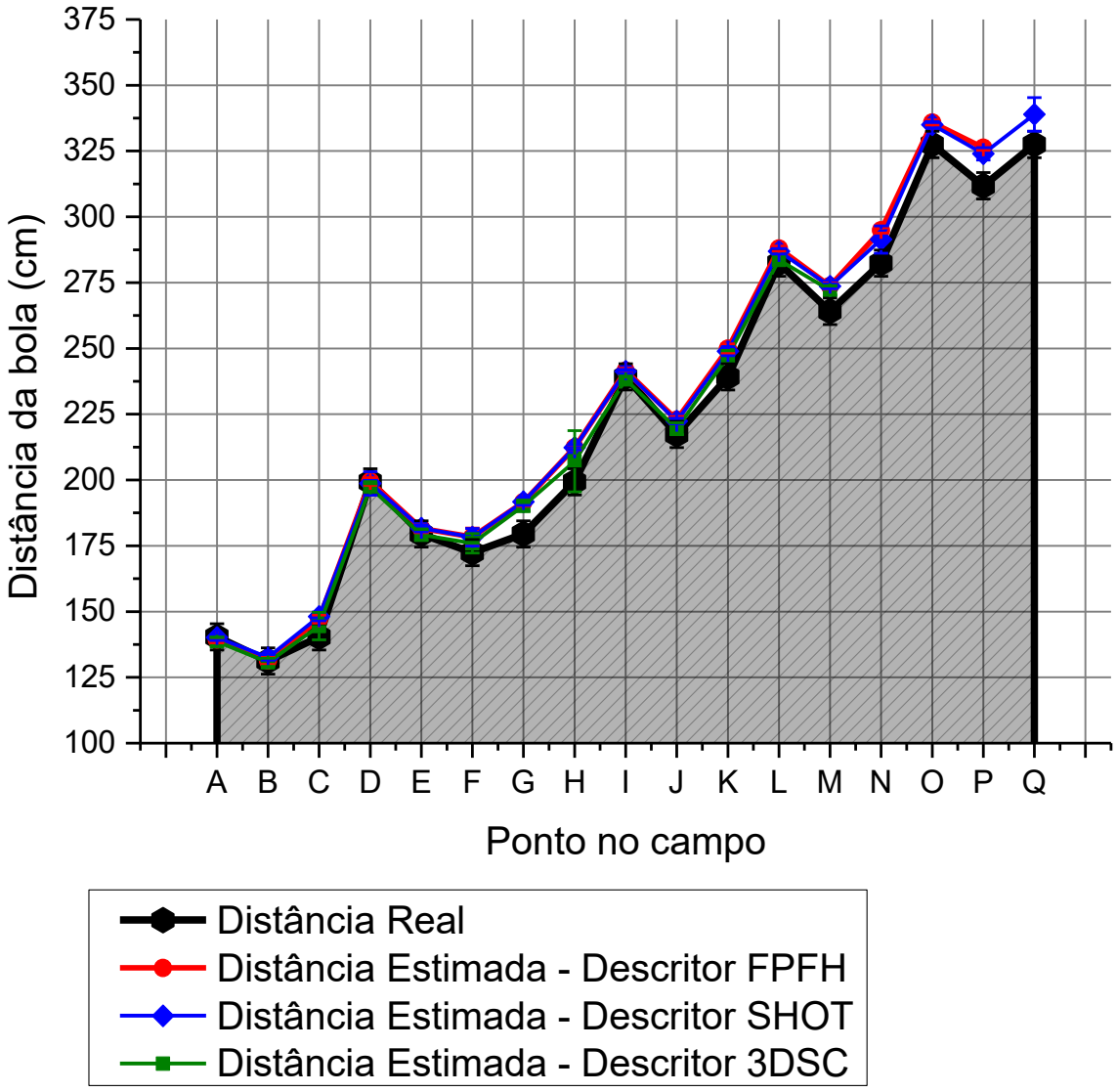
A tabela 10 apresenta o comparativo entre a distância real e a distância estimada da bola para cada método descritor. Devido às imprecisões na medição da distância real, foi considerado um desvio padrão de 5 cm na distância real para cada ponto do campo. A figura 50 apresenta o gráfico referente à tabela 10.

Tabela 10 – Comparativo entre a distância real e a distância estimada da bola para cada método descritor.

| Ponto | Distância Real | | Descritor FPFH | | Descritor SHOT | | Descritor 3DSC | |
|---|---|---|---|---|---|---|---|---|
| | Distância (cm) | σ (cm) | Distância (cm) | σ (cm) | Distância (cm) | σ (cm) | Distância (cm) | σ (cm) |
| A | 140,4 | 5 | 139,2 | 1,1 | 140,1 | 1,1 | 138,6 | 1,7 |
| B | 131,2 | 5 | 131,4 | 1,4 | 132,6 | 1,0 | 130,8 | 1,5 |
| C | 140,4 | 5 | 147,2 | 1,5 | 148,1 | 1,7 | 144,5 | 5,2 |
| D | 199,3 | 5 | 199,5 | 1,5 | 198,7 | 4,5 | 197,1 | 1,4 |
| E | 179,5 | 5 | 181,6 | 1,6 | 181,7 | 1,3 | 179,2 | 2,0 |
| F | 172,4 | 5 | 178,3 | 3,4 | 178,3 | 3,3 | 176,0 | 3,7 |
| G | 179,5 | 5 | 191,6 | 1,5 | 191,7 | 1,4 | 190,2 | 2,1 |
| H | 199,3 | 5 | 212,3 | 0,9 | 212,3 | 0,9 | 207,1 | 11,6 |
| I | 239,2 | 5 | 241,5 | 0,9 | 241,3 | 0,7 | 238,2 | 0,8 |
| J | 217,3 | 5 | 222,9 | 1,3 | 222,4 | 1,1 | 219,5 | 2,2 |
| K | 239,2 | 5 | 249,9 | 1,9 | 248,9 | 1,7 | 247,0 | -- |
| L | 282,4 | 5 | 287,9 | 1,0 | 286,9 | 1,0 | 283,6 | -- |
| M | 264,1 | 5 | 273,7 | 1,2 | 273,6 | 1,0 | 272,1 | 1,7 |
| N | 282,4 | 5 | 294,8 | 1,0 | 291,4 | 5,2 | -- | -- |
| O | 327,5 | 5 | 335,8 | 1,0 | 335,0 | 1,2 | -- | -- |
| P | 311,8 | 5 | 326,2 | 1,0 | 324,0 | 2,3 | -- | -- |
| Q | 327,5 | 5 | -- | -- | 338,9 | 6,4 | -- | -- |

Fonte: Autor

Figura 50 – Gráfico do comparativo entre a distância real e a distância estimada da bola para cada método descritor.




Fonte: Autor

### 5.6.3 **Desempenho do sistema**

Para identificar o desempenho do sistema no reconhecimento de bola, foi utilizado o método Precisão e Revocação. Para isso, foram analisados os resultados dos 1020 pares de

imagem (510 pares de imagem com bola e 510 pares de imagem sem bola) submetidos ao processo de reconhecimento da bola nos três métodos descritores. A tabela 11 apresenta a classificação de cada par como: Falso Positivo (FP), para os casos onde o sistema identificou bola onde não havia; Verdadeiro Positivo (VP), para os casos onde o sistema identificou a bola na posição correta; Falso Negativo (FN), para os casos onde o sistema não encontrou a bola, sendo que a bola estava presente na cena; e Verdadeiro Negativo (VN), para os casos onde o sistema não identificou a bola e realmente ela não estava presente na cena.

Tabela 11 – Classificação dos pares de imagem pelo sistema.

| **Classificação** | **Descritor FPFH** | | **Descritor SHOT** | | **Descritor 3DSC** | |
|---|---|---|---|---|---|---|
| | **Com bola** | **Sem bola** | **Com bola** | **Sem bola** | **Com bola** | ***Sem bola** |
| **Falso Positivo** | 0 | 0 | 12 | 172 | 13 | 0 |
| **Verdadeiro Positivo** | 399 | 0 | 473 | 0 | 187 | 0 |
| **Falso Negativo** | 111 | 0 | 25 | 0 | 310 | 0 |
| **Verdadeiro Negativo** | 0 | 510 | 0 | 338 | 0 | 60 |
| **Total** | 510 | 510 | 510 | 510 | 510 | 60 |

* Devido ao alto tempo de processamento, foram analisados apenas 60 pares dos 510 disponíveis.

Fonte: Autor

É possível obter o valor da precisão, da revocação (*recall*) e da acurácia do sistema, aplicando as equações (49), (50) e (61), respectivamente (OLSON; DELEN, 2008):

$$Precisão = \frac{VP}{VP + FP} \text{ ,} \tag{49}$$

$$Revocação = \frac{VP}{VP + FN} \text{ ,} \tag{50}$$

$$Acurária = \frac{VP + VN}{VP + VN + FP + FN} \text{ .} \tag{51}$$

A tabela 12 apresenta os resultados das equações (49), (50) e (51), aplicando os valores correspondentes de cada método descritor.

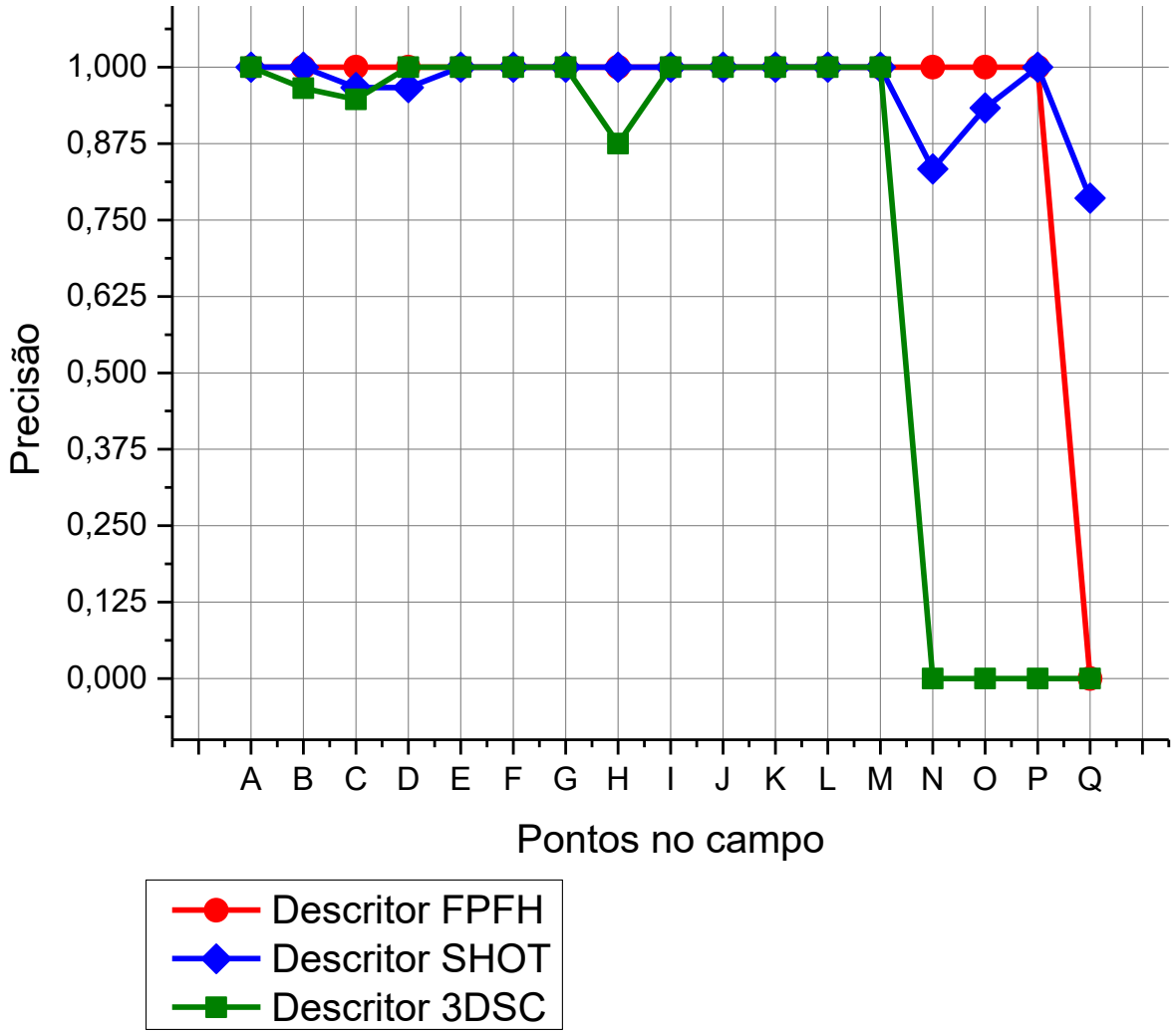
Tabela 12 – Desempenho do sistema no reconhecimento de bola.

| | **Descritor FPFH** | **Descritor SHOT** | **Descritor 3DSC** |
|---|---|---|---|
| **Precisão** | 1,0 | 0,71 | 0,93 |
| **Revocação** | 0,78 | 0,94 | 0,37 |
| **Acurácia** | 0,89 | 0,79 | 0,43 |

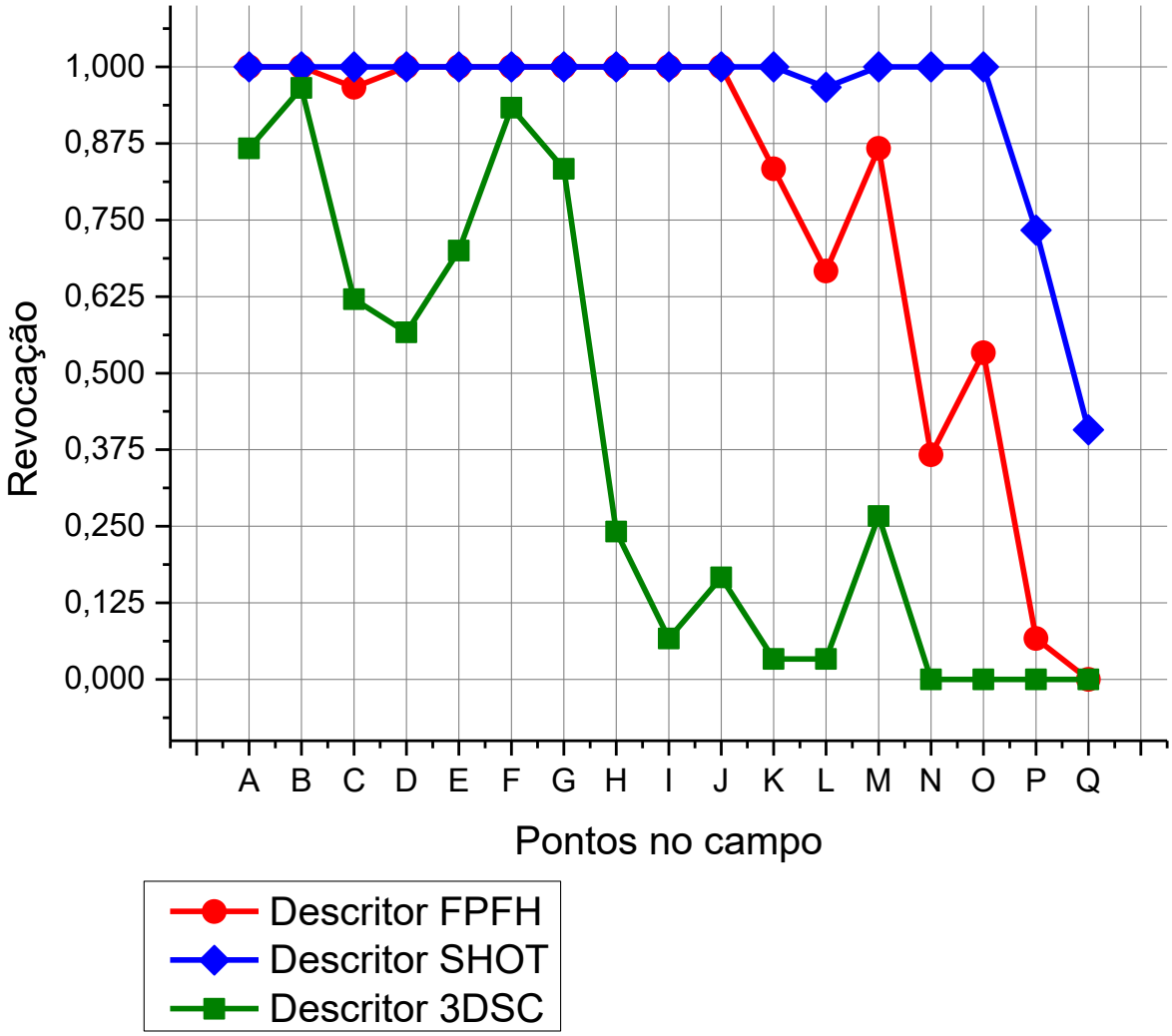
Fonte: Autor

A figura 51 apresenta os gráficos de cada método descritor para cada ponto no campo em função da precisão (a), revocação (b) e acurácia (c).

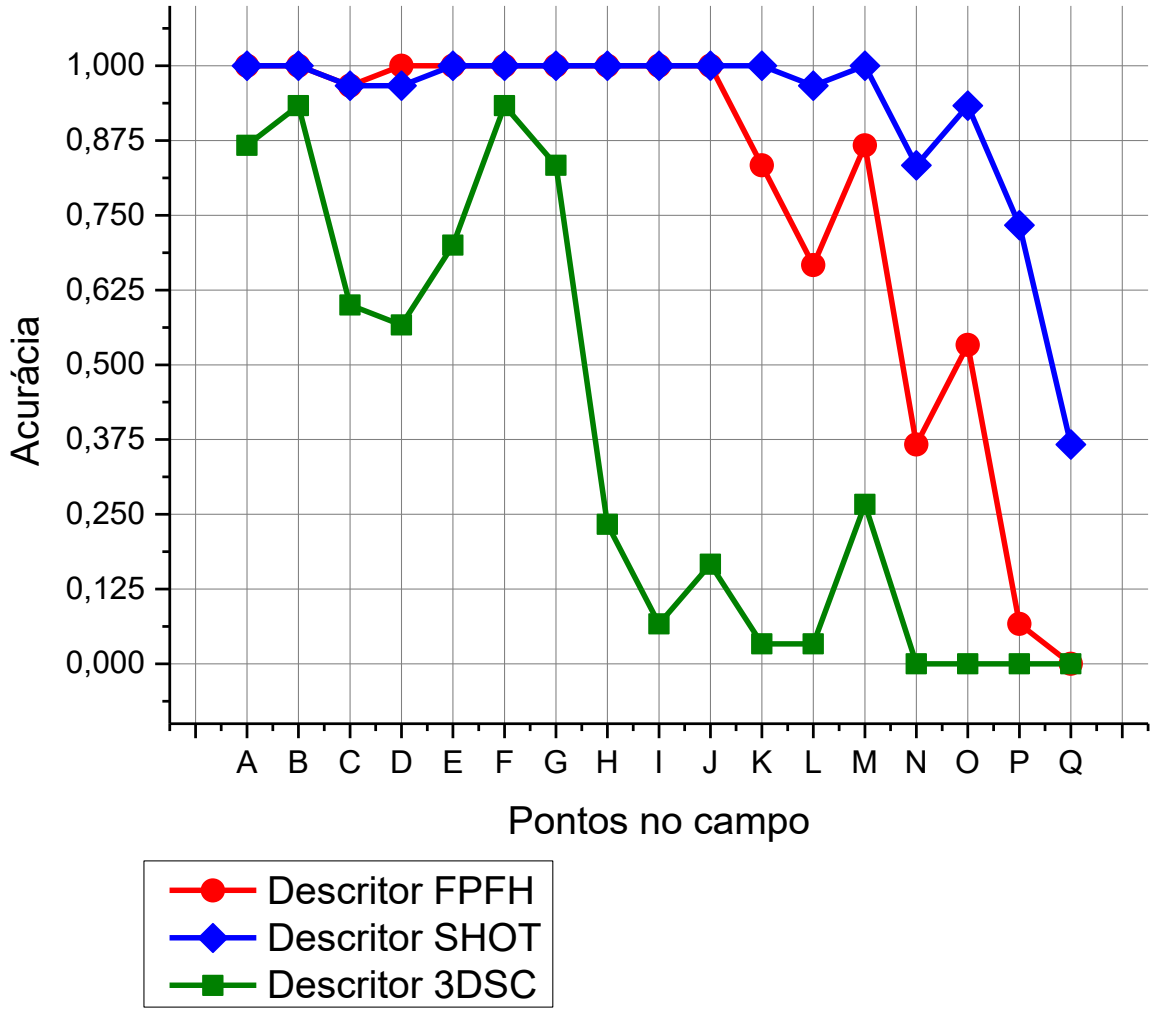
Figura 51 – Gráfico do desempenho do sistema em precisão (a), revocação (b) e acurácia (c).




(a)


Revocação
1,000
0,875
0,750
0,625
0,500
0,375
0,250
0,125
0,000
A B C D E F G H I J K L M N O P Q
Pontos no campo
Descritor FPFH
Descritor SHOT
Descritor 3DSC

(b)




(c)

## 5.7 RESUMO E CONCLUSÃO DO CAPÍTULO

Este capítulo apresentou os experimentos realizados e os resultados obtidos com o sistema de visão estéreo implementado. Pode-se observar que a calibração do sistema estéreo foi realizada com sucesso e apresentou resultados satisfatórios, com erro inferior a 0,5 px em uma resolução HD (1280x720 px). Com este resultado, foi possível obter bons resultados na geração do mapa de disparidade e, consequentemente, obteve-se bons resultados na reconstrução 3D dos pares de imagens capturados. Foi criado um modelo de bola para cada método descritor para viabilizar o reconhecimento da bola e a estimativa de sua respectiva

distância. O reconhecimento da bola foi realizado para os três métodos descritores proposto e a estimativa da distância foi satisfatória, apresentando erros insignificantes para o ambiente de futebol onde o sistema será aplicado.

O tempo médio mais rápido para o reconhecimento da bola neste trabalho foi 24,3 segundos, utilizando o método FPFH. Considerando que este trabalho não teve como objetivo o desempenho do tempo, foram utilizados recursos e configurações de alta definição que podem ser otimizados a fim de reduzir o tempo de processamento.

No capítulo seguinte será apresentada a conclusão do trabalho e as sugestões para os trabalhos futuros.

# 6 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho comparou o desempenho de três algoritmos descritores 3D aplicados para o reconhecimento de bolas de futebol com diferentes texturas utilizadas nas competições da RoboCup, através de um sistema de visão estéreo adequado à um robô móvel humanoide atuando na sub-liga *KidSize* da competição *RoboCup*, cumprindo o objetivo proposto.

A principal contribuição deste trabalho foi o reconhecimento de um objeto 3D e a estimativa de sua distância utilizando um sistema de visão estéreo composto apenas por duas câmeras, sem a utilização de *lasers* ou câmeras 3D.

Outra contribuição deste trabalho foi combinar o funcionamento dos algoritmos de calibração, retificação e correspondência estéreo, disponibilizados pela biblioteca OpenCV, com as técnicas de reconhecimento de objeto 3D, disponíveis na biblioteca PCL, ajustando os valores de configuração desses métodos experimentalmente, buscando o melhor resultado para o cenário de um ambiente de futebol.

Por fim, este trabalho contribuiu com o resultado da comparação dos três métodos descritores para reconhecimento 3D de diferentes bolas em diferentes posições, sendo eles o FPFH, o SHOT e o 3DSC, onde o método FPFH apresentou melhor desempenho atingindo precisão de 100%, revocação de 78% e acurácia de 89%, diante de 71% de precisão, 94% de revocação e 79% de acurácia do método SHOT e 93% de precisão, 37% de revocação e 43% de acurácia do método 3DSC. Outra conclusão foi o custo computacional, onde o método FPFH apresentou melhor desempenho no tempo de resposta do sistema em todo o processo de localização da bola e cálculo da distância (média de 24,7s) em relação ao método SHOT (média de 29,8s) e ao método 3DSC (média de 624,3s).

Durante a implementação do sistema, surgiram-se alguns problemas relacionados aos processos de calibração do sistema, mapa de disparidade e cálculo das descrições pelo método FPFH, sendo que todos foram corrigidos com sucesso.

O problema apresentado na calibração do sistema foi devido à uma câmera do sistema estar com defeito no mecanismo de foco, não respondendo aos comandos de configuração e apresentando duas imagens divergentes do tabuleiro de xadrez. A solução para este problema foi adquirir um novo par de câmeras e efetuar a mesma configuração de foco nas duas câmeras.

O problema ocorrido na geração do mapa de disparidade foi decorrente à diferença de brilho e contraste entre as imagens do par, ocasionando divergências de cor entre as imagens e, consequentemente, grandes ruídos no mapa de disparidade. A solução consistiu em uma rotina que sempre configura os parâmetros das câmeras ao iniciar o sistema.

O problema enfrentado no descritor FPFH ocorreu devido à uma falha no gerenciamento de memória no processamento do método descritor FPFH da biblioteca PCL 1.7.2, ocasionando falha de segmentação no software Modo *Online*. A solução para este problema foi atualizar a biblioteca PCL da versão 1.7.2 para a versão 1.8.1.

Entre as sugestões para os trabalhos futuros estão:

a) implementar a estrutura das câmeras em material rígido, porém mais apropriado para a aplicação nos robôs;
b) implementar este sistema nos futuros robôs do time RoboFEI;
c) testar o funcionamento deste sistema com o robô andando;
d) expandir o reconhecimento 3D para outros objetos, tais como robôs, gol e linhas do campo;
e) implementar o reconhecimento de objetos 3D utilizando conceitos de *Deep Learning*.

# REFERÊNCIAS

BELONGIE, S., MALIK, J.; PUZICHA, J. Shape matching and object recognition using shape contexts. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, p. 509-522, 07 ago., 2002.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV**. 1. ed. Sebastopol: O´Reilly Media, 2008.

CHENG, Q., YU, S., YU, Q.; XIAO, J. Real-time Object Segmentation for Soccer Robots Based on Depth Images. **Internation Conference on Information and Automation - IEEE**, p. 1532-1537, Ningbo, China, 2016.

CHEN, H.; BHANU, B. 3D free-form object recognition in range images using local surface patches. **Pattern Recognition Letters 28**, 28 Fev., 2007.

CYGANEK, B.; SIEBERT, J. P. **An Introduction to 3D Computer Vision** - Techniques and Algorithms. 1. ed. Chichester: Wiley, 2009.

EGNAL, G. **Mutual Information as a Stereo Correspondence Measure**, Pennsylvania: University of Pennsylvania, 2000.

FILIPE, S.; ALEXANDRE, L. A. A Comparative Evaluation of 3D Keypoint Detectors in a RGB-D Object Dataset. **Computer Vision Theory and Applications (VISAPP), 2014 International Conference on**, 5-8 Jan., 2014.

FLINT, A., DICK, A.; HENGEL, A. V. d. Thtift: Local 3D Structure Recognition. **Digital Image Computing Techniques and Applications**, p. 182-188, 2007.

FORSYTH, D. A.; PONCE, J. **Computer Vision:** A Modern Approach. New Jersey: Prentice Hall, 2003.

FROME, A. et al. Recognizing Objects in Range Data Using Regional Point Descriptors. **8th European Conference on Computer Vision**, p. 224-237, 11-14 maio, 2004.

GARCIA, E., JIMENEZ, M. A., DE SANTOS, P. G.; ARMADA, M. The Evolution of Robotics Research. **IEEE Robotics & Automation Magazine***,* 14(1), p. 90-103, 2007.

GOMES, R. B. et al. Efficient 3D object recognition using foveated point clouds. **Computers & Graphics***,* 37(5), p. 496-508, 2013.

GONZALES, R. C.; WOODS, R. E. **Processamento de Imagens Digitais**. s.l.:Edgard Blüncher LTDA, 2013.

GUO, Y. et al. A Comprehensive Performance Evaluation of 3D Local Feature Descriptors. **International Journal of Computes Vision**, p. 66-89, 16 abril, 2015.

GUO, Y. et al. Rotational Projection Statistics for 3D Local Surface Description and Object Recognition. **International Journal of Computer Vision**, p. 63-86, 24 abril, 2013a.

GUO, Y. et al.TriSI: A Distinctive Local Surface Descriptor for 3D Modeling and Object Recognition. **8th International Conference on Computer Graphics Theory and Applications**, p. 86-93, 2013b.

GUTMANN, J. S., FUKUCHI, M.; FUJITA, M. Stair Climbing for Humanoid Robot Using Stereo Vision. **International Conference on Intelligent Robots and Systems**, p. 1407-1413, 2 out., 2004.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. **Alvey vision conference**, 15(50), p. 147-152, 1988.

HARTLEY, R.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. New York: Cambridge University Press, 2003.

HIRSCHMÜLLER, H. **Accurate and efficiente stereo processing by semi-global matching and mutual information**. San Diego, IEEE, 2005.

Honda R&D. The intelligent ASIMO: System overview and integration. **Intl. Conference on Intelligent Robots and Systems**, p. 2478-2483, out., 2002.

Honda, 2002. **Robô Humanoide Asimo**.
Disponível em: <http://world.honda.com/ASIMO/technology/2002/index.html>. Acesso em: 19 Janeiro 2017.

JOHNSON, A. E.; HEBERT, M.. Using spin images for efficient object recognition in cluttered 3D scenes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, p. 433-449, maio, 1999.

Logitech, 2016. **C920 Specifications**.
Disponível em: <http://support.logitech.com/en_us/product/hd-pro-webcam-c920>. Acesso em 20 dezembro 2016.

LOOP, C.; ZHANG, Z. Computing rectifying homographies for stereo vision. **IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 125-131, 1999.

MAHANI, M.-A. N., JAFARI, S.; RAHMATKHAH, H. A novel approach for humanoid push recovery using stereopsis. **Cambridge University Press**, Volume 32, p. 413-431, 2013.

NEWMAN, W. et al. **Autonomous Valve Turning with an Atlas Humanoid Robot**. Madrid, Spain, IEEE-RAS, 2014.

Nicoguaro, 2016. **Wikimedia**.
Disponível em:<https://commons.wikimedia.org/wiki/File:Surface_normals.svg>. Acesso em 28 novembro 2017.

OLEARI, F., RIZZINI, D. L.; CASELLI, S. **A low-cost stereo system for 3D object recognition**. Cluj-Napoca, IEEE, 2013.

OLSON, D. L.; DELEN, D. **Advanced Data Mining Techniques.** 1. ed. Berlin: Springer, 2008.

OpenCV - Download. **Biblioteca Gráfica OpenCV 2.4.9.**
Disponível em: <http://opencv.org/downloads.html>. Acesso em 13 dezembro 2016.

OpenCV - Installation. **OpenCV - Installation in Linux.**
Disponível em: <http://docs.opencv.org/3.0-last-rst/doc/tutorials/introduction/linux_install/linux_install.html>. Acesso em 13 dezembro 2016.

QRIO, W. **QRIO.**
Disponível em: <https://en.wikipedia.org/wiki/QRIO>. Acesso em 19 janeiro 2017.

RoboCup Federation. **RoboCup.**
Disponível em: <http://www.robocup.org/>. Acesso em 28 novembro 2017.

ROSHEIM, M. E. **Leonardo's Lost Robots.** 1. ed. Wurtzburgo, Alemanha: Springer, 2006.

RUSU, R. B. **Estimating Surface Normals in a PointCloud.**
Disponível em:
<http://pointclouds.org/documentation/tutorials/normal_estimation.php#normal-estimation>. Acesso em 25 novembro 2017.

RUSU, R. B., BLODOW, N.; BEETZ, M. Fast Point Feature Histograms (FPFH) for 3D Registration. **International Conference on Robotics and Automation,** p. 3212-3217, 12 maio, 2009.

RUSU, R. B., BLODOW, N., MARTON, Z. C.; BEETZ, M.. Aligning Point Cloud Views using Persistent Feature Histograms. **IEEE/RSJ International Conference on Intelligent Robots and Systems**, p. 3384-3391, 22-26 set., 2008.

SABE, K. et al. Obstacle Avoidance and Path Planning for Humanoid Robots using Stereo Vision. **International Conference on Robotics & Automation,** p. 592-597, abril, 2004.

SMITH, S. M.; BRADY, J. M. SUSAN - A new approach to low level image processing. **International Journal of Computer Vision***,* 23(1), p. 45-78, 1997.

SZELISKI, R. **Computer Vision:** Algorithms and Applications. s.l.:Springer, 2010.

THOMPSON, S.; KAGAMI, S. Humanoid Robot Localisation using Stereo Vision. **5th IEEE-RAS International Conference on Humanoid Robots**, p. 19-25, 2005.

TOMBARI, F., SALTI, S.; STEFANO, L. D. Unique Shape Context for 3D Data Description. **ACM Workshop on 3D Object Retrieval**, p. 57-62, out., 2010a.

TOMBARI, F., SALTI, S.; STEFANO, L. D. Unique Signatures of Histograms for Local Surface Description. **European Conference on Computer Vision**, p. 356-369, 2010b.

WHEATSTONE, C. Contributions to the Physiology of Vision. **"Philosophical Transactions" of the Royal Society of London, Vol. 128**, p. 371-394, 21 jun., 1838.

ZHANG, Z., SONG, Y., ZHANG, W.; YIN, S. Research Humanoid Robot Walking Based on Vision-Guided. **Applied Mechanics and Materials**, Volume 496-500, p. 1426-1429, 2014.

ZHONG, Y. Intrinsic Shape Signatures: A Shape Descriptor for 3D Object Recognition. **IEEE 12th Internation Conference on Computer Vision Workshops**, pp. 689-696, 2009.