Ximena Mariel Zeballos Vasquez

POSICIONAMENTO DE MÚLTIPLOS OBJETOS A PARTIR DE VISÃO ESTÉREO

Porto Alegre

2015

Ximena Mariel Zeballos Vasquez

POSICIONAMENTO DE MÚLTIPLOS OBJETOS A PARTIR DE VISÃO ESTÉREO

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Pontifícia Universidade do Rio Grande do Sul, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Sinais, Sistemas e Tecnologia da Informação.
Linha de pesquisa: Automação e Sistemas.

Pontifícia Universidade do Rio Grande do Sul – PUCRS

Faculdade de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

Orientador: Aurélio Tergolina Salton

Porto Alegre

2015

POSICIONAMENTO DE MÚLTIPLOS OBJETOS A PARTIR DE VISÃO ESTÉREO

CANDIDATA: XIMENA MARIEL ZEBALLOS VASQUEZ

Esta Dissertação de Mestrado foi julgada para obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Pontifícia Universidade Católica do Rio Grande do Sul.

DR. AURELIO TERGOLINA SALTON - ORIENTADOR

BANCA EXAMINADORA

DR. JEFERSON VIEIRA FLORES - DEPARTAMENTO DE ENGENHARIA ELÉTRICA - UFRGS

DR. DARIO F. GUIMARÃES DE AZEVEDO - DO PPGEE/FENG - PUCRS

PUCRS
Campus Central
Av. Ipiranga, 6681 - Prédio 30 - Sala 103 - CEP: 90619-900
Telefone: (51) 3320.3540 - Fax: (51) 3320.3625
E-mail: engenharia.pg.eletrica@pucrs.br
www.pucrs.br/feng

A mi mamá, por todo su amor y apoyo incondicional.
A mi sobrina Warita, por alegrar mis días en esta etapa de mi vida brindandome su cariño y su sonrisa.

Agradecimentos

Ao meu orientador, Prof. Aurélio Tergolina Salton, por seu apoio, sua disposição, seu conhecimento, sua compreensão, sua paciência e seus grandes conselhos em todos os momentos na realização deste projeto e ao longo do mestrado.

À Diretora Profa. Letícia B. Poehls, pela confiança desde o início do mestrado, e por fazer que um sonho se torne realidade. Também agradeço à comissão coordenadora e à equipe administrativa do PPGEE, pela ajuda e apoio.

Aos bolsistas e mestrandos do GACS, a Guilherme F. Fernandez, por seu apoio e seu bom trabalho na construção do gimbal. A Rafael da S. Castro, por seus conselhos e sua ajuda ao longo do projeto. E aos colegas do laboratório do GACS, por terem me acolhido no laboratório e oferecido sua amizade.

À minha mãe, por me mostrar o valor e coragem à vida, pela ajuda e amor incondicional.

À minha irmã, meu cunhado e minhas sobrinhas: muito obrigado por tudo. Sem seu apoio, eu não teria tornado este sonho uma realidade.

A meu irmão, sua esposa e meus sobrinhos, por terem me apoiado na conclusão deste projeto.

Ao meu parceiro Eddy, pelo carinho, pela compreensão durante todo este tempo, ajudando-me incondicionalmente com seus conselhos de determinação e atitude na vida.

A meu bom amigo Pablo, pela amizade, pelo apoio e pelos conselhos.

E, finalmente, um agradecimento muito especial à PUCRS, por ter me acolhido na universidade e ter-me mostrado uma realidade diferente. Ao mesmo tempo, outro agradecimento especial para este país tão grande que é Brasil, às pessoas bondosas que estiveram sempre prestes a me ajudar, mostrando-me a alegria que levam nos seus corações e a ordem que as coisas deveriam ter, para que uma nação seja “Grande”.

"Para a perseverança o fracasso não é uma opção. A pereseverança é a mãe do triunfo."
(X.Z.V.)

Resumo

Este trabalho trata da estimação da posição tridimensional de múltiplos objetos a partir de imagens capturadas em um ambiente de visão estéreo. O formalismo matemático utilizado no projeto do sistema de visão estéreo inicia-se com a representação de cenário, levando todo o ambiente para um espaço euclidiano, também chamado de representação em três dimensões. Depois de representar o espaço euclidiano no sistema de visão estéreo, leva-se esse formalismo às projeções das imagens numa perspectiva da câmera ideal em coordenadas homogêneas, das quais se obtém a câmera com parâmetros intrínsecos e extrínsecos. A partir disso, utilizam-se os conceitos de visão estéreo e triangularização entre as câmeras para calcular a profundidade dos diversos objetos presentes no cenário. Assim, a reconstrução tridimensional da imagem é obtida.

A validação dos algoritmos propostos é feita através de um experimento construído especialmente para o sistema de visão estéreo, em que foram colocadas duas câmeras com vistas semiparalelas em um ambiente, sendo também adicionada uma plataforma do tipo gimbal movimentando-se em três dimensões. Acima dessa, é colocado um plano com três pontos (marcadores), os quais simulam o movimento de três objetos. A partir daí, são capturadas duas imagens a serem processadas mediante os algoritmos propostos, resultando na reconstrução dos pontos no espaço tridimensional. O processo de validação se dá através da comparação entre as orientações dos planos fornecidos pelo algoritmo proposto e pelos comandos dos servomotores do gimbal.

Palavras-chaves: **Visão estéreo. Reconstrução a partir de duas vistas. Mapeamento 3D. Gimbal.**

Abstract

The following study is about the estimation of three-dimensional position of multiple objects captured in images in a stereo vision setting. The mathematical formalism begins with the representation of motion in Euclidean space. After that, the projection of the images is performed in an ideal camera perspective using homogeneous coordinates. This results in the intrinsic and extrinsic parameters, which are part of the camera calibration. From that, stereo vision is used to obtain three-dimensional position of objects that are captured in the images from two cameras. Thus, depths of various objects are obtained to reconstruct their positions.

The validation of the proposed algorithms are made through an experiment built for this purpose. This experiment uses two cameras with semi-parallel views to the center. In the center of the environment there is a gimbal platform, which performs a uniform motion. Above, there is a plan with three white dots, simulating the objects position. This motion captures two images, which are processed by the proposed algorithm. The validation consists in making a comparison between the orientation planes of the gimbal servomotors and reconstruction objects.

Key-words: **Stereo vision. Reconstruction of two view. Maping 3D. Gimbal.**

Lista de ilustrações

Lista de abreviaturas e siglas

RA	*Realidade Aumentada*
VE	*Visão Estéreo*
VC	*Visão Computacional*
VANT	*Veículo Aéreo Não Tripulado*
UAV	*Unmanned Aerial Vehicle*
MAV	*Micro Veículos Aéreos*
ARM	*Advanced RISC Machine – Máquina avançada RISC. RISC (Reduced Instruction Set Computer = Computador com Conjunto de instruções reduzidas*

Lista de símbolos

A'	transposta da matriz A.
$rank(A)$	posto da matriz A.
$diag\{A, B\}$	matriz bloco-diagonal formada pelas matrizes A e B, isto é, $\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$.
$\|a\|$	valor absoluto do escalar a.
$\|\|x\|\|$	a norma do vetor x.
$\mathbb{N}$	conjunto de números naturais.
$\mathbb{R}$	conjunto dos números reais.
$\mathbb{R}^n$	espaço euclidiano de ordem n.
$\mathbb{R}^{n \times m}$	espaço das matrizes reais de dimensão $n \times m$.
P	posição do objeto no espaço tridimensional $P = [X_1, Y_1, Z_1]^T \in \mathbb{R}^{3\times 1}$.
$\mathbf{P}$	posição do objeto na projeção da imagem $\mathbf{P} \in \mathbb{R}^{4\times 4}$, usa o movimento completo de um objeto g.
$\boldsymbol{x}$	coordenadas da imagem em pixels, projetada a partir do ponto tridimensional P.
g	movimento completo de um objeto denotado por sua rotação e translação, ou de outra forma $g(R, T) \in \mathbb{R}^{4\times 3}$, também conhecido como parâmetros extrínsecos.
$\overline{g}$	movimento completo de um objeto em coordenadas homogêneas, dado por $\overline{g}(R, T) \in \mathbb{R}^{4\times 4}$.
Π_0	matriz usada na projeção da imagem dado por uma matriz identidade e uma coluna de zeros $[I\ 0] \in \mathbb{R}^{3\times 4}$.
K	matriz de parâmetros intrínsecos, $K \in \mathbb{R}^{3\times 3}$.
$\sim$	indica igualdade até um fator escalar.
Texto itálico	indica que o texto está escrito em inglês.

Sumário

1 Introdução

A recuperação da posição tridimensional a partir de duas imagens é um tópico de pesquisa muito útil na visão computacional (VC), devido a seu amplo potencial em muitas aplicações, como reconstrução de objetos, ambientes, superfícies etc. (LI, 2015). Nos últimos anos, tem havido grande foco na literatura para recuperar o espaço tridimensional das imagens. Diferentes tipos de algoritmos são usados devido à grande variedade de opções, por exemplo, pelo número de câmeras, modos de exibição, calibração da câmera, tipos de recursos e modelos da cena (MCCOUN, 2010).

Dessa forma, a VC pode ser definida como o estudo de processos de reconhecimento, localização, compreensão e detecção de objetos a partir de imagens (TAVARES, 1995). As áreas de estudo típicas da VC são: o reconhecimento, análise do movimento, reconstrução da cena e restauração da imagem (MARTINET, 2013).

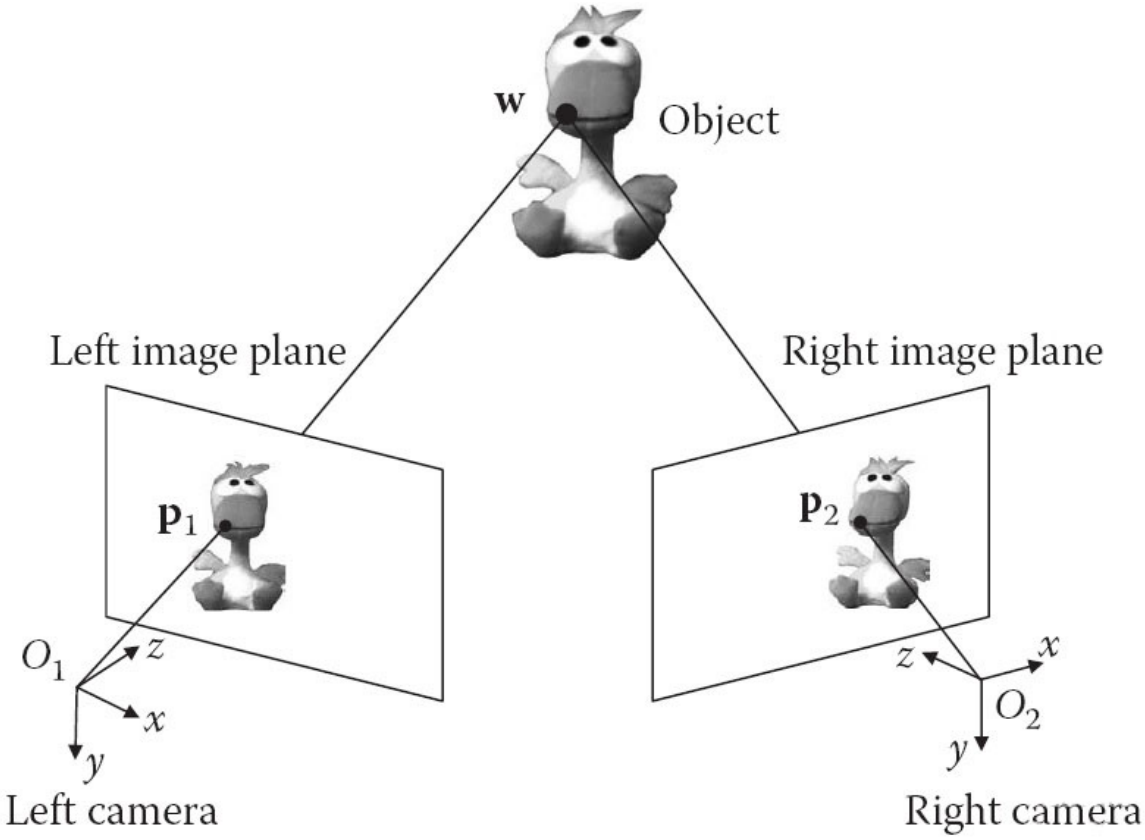
Por outro lado, no início da VC, existiram pesquisas para aplicações industriais, utilizando um sistema de visão com apenas uma câmera. Esses sistemas possuem muitas limitações por terem apenas um ponto de vista da cena (DOMINGUEZ, 2012). Uma limitante importante é a recuperação do espaço tridimensional de uma imagem, a partir de uma vista só. Dado que o processo de projeção de um objeto do mundo tridimensional (3-D) para uma imagem bidimensional (2-D), não é reversível (SUCAR, 2007). Assim, uma vez que a degeneração é introduzida na imagem, nenhuma informação é retida do espaço tridimensional (TARAGLIO, 2011). As informações dessa transformação são perdidas (YI, 2004).

Um avanço importante na VC foi a implementação de sistemas com dois pontos de vista, conhecidos como VE (DOMINGUEZ, 2012). A VE é uma alternativa para recuperar a terceira dimensão. Uma definição dada por (DUFAUX, 2013) diz que a VE proporciona uma sensação de profundidade, apresentando ao espectador duas perspectivas ligeiramente diferentes da mesma cena.

Outras alternativas para recuperar a terceira dimensão podem ser por sombreamento, textura (SUCAR, 2007), entre outros. A VE é motivada a partir da visão humana que pode perceber propriedades de profundidade de uma cena (ZHANG, 2013).

Em geral, uma configuração convencional de um sistema de VE é mostrada na Fig. 1. Duas câmeras de visão idênticos capturam as imagens da esquerda e da direita de um objeto. Um ponto (3-D) w, sobre a superfície de um objeto real, é projetada para os planos das imagens. Dois pontos bidimensionais (2D), p_1 e p_2, são as projeções do ponto, nos planos das imagens (ZHANG, 2013), (HARTLEY, 2004), (GONÇALVES, 2005). O ponto w w é a posição tridimensional a ser recuperada.

Figura 1 – Configuração de um sistema de VE para um ponto

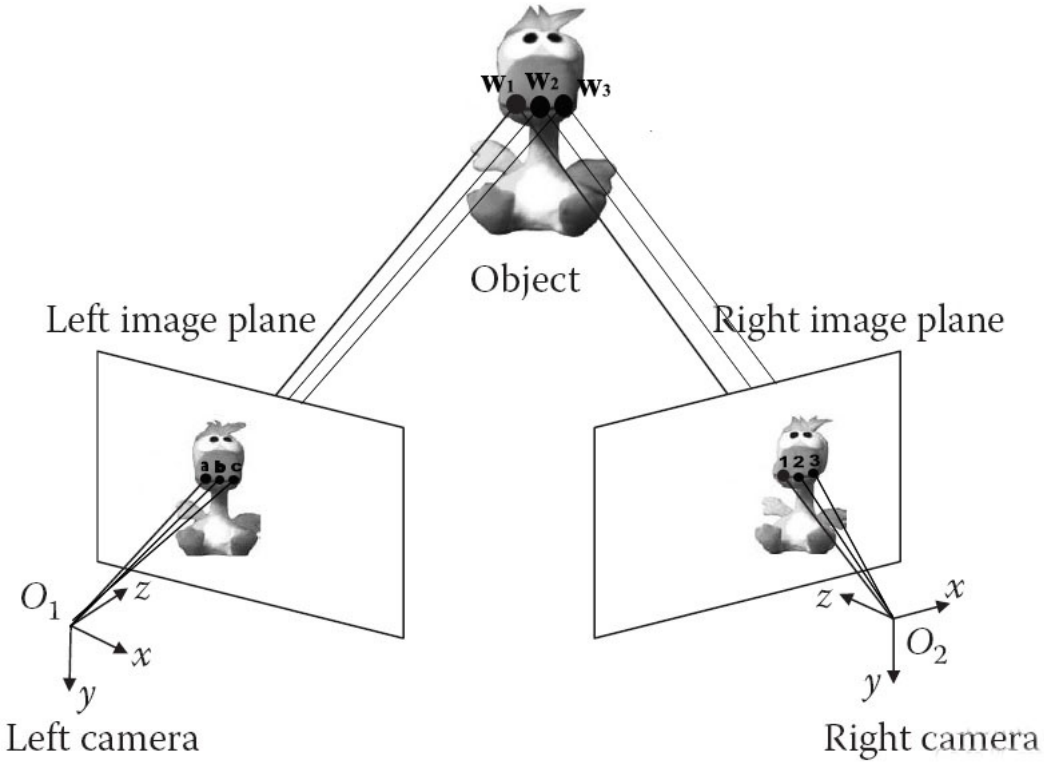


FONTE: (ZHANG, 2013).

Sempre que existir esse tipo de configuração da Fig. 1, relações geométricas podem ser derivadas, as quais podem ser matematicamente descritas sob a suposição de modelo de câmeras *pinhole* e geometria estéreo (TORREAO, 2011). A geometria estéreo permite recuperar a profundidade para um ponto (BóDIS, 2008).

Assim, numa generalização, a geometria estéreo também permite recuperar a profundidade para vários pontos projetados em um par de imagens, como mostra a Fig. 2. Neste caso da Fig. 2, procura-se obter a profundidade dos pontos w_1, w_2, w_3, mas, no decorrer da solução deste problema, aparece um dos problemas inerentes a VE, chamado de correlação estéreo (TORREAO, 2011). O problema de correlação estéreo pode ser definido como, a correspondência de pontos entre duas imagens (ZHANG, 2013). Por exemplo, na Fig. 2, numa relação entre a imagem da esquerda e a imagem da direita, o ponto 'a' corresponde ao ponto '1', o ponto 'b' corresponde ao ponto '2', e o ponto 'c' corresponde ao ponto '3'.

Figura 2 – Configuração de um sistema de VE para vários pontos



FONTE: (ZHANG, 2013).

Uma vez que, para resolver o problema de correlação estéreo, precisa-se de um grande número de cálculos computacionais (ZHANG, 2013), e, a fim de reduzir a complexidade computacional do problema de correlação estéreo, a geometria estéreo deve ser considerada. Além disso, para aplicar a geometria estéreo, é necessário saber a calibração das câmeras, em relação a um sistema de coordenadas de referência.

O método de calibração serve para obter uma relação geométrica entre as duas câmeras, sendo representado por transformações euclidianas 3-D, rotação e translação (ZHANG, 2013). A partir da calibração estéreo, sabe-se que correspondências estéreo são relacionadas por uma projeção geométrica entre duas vistas, chamada geometria epipolar (HARTLEY, 2004).

A geometria epipolar entre as imagens estéreo fornece um restrição muito forte para encontrar correlações estéreo. Essa restrição é chamada como restrição epipolar, que é essencial na correlação estéreo. Assim, a complexidade do cálculo e tempo computacional entre correspondências estéreo pode ser gradualmente reduzida (ZHANG, 2013), (BóDIS, 2008).

Tudo o que foi explicado acima sobre VE, como: geometria epipolar, correlação estéreo, restrição epipolar e calibração de câmeras, será abordado nesta dissertação, com o objetivo de obter a profundidade de vários pontos entre duas imagens a partir de um sistema de VE.

Para validar o algoritmo proposto, será construído o gimbal, o qual terá três marcadores brancos em cima, os quais formam um plano de orientação. A validação surge da comparação entre o plano de orientação da reconstrução dos pontos e um plano criado a partir dos servomotores do gimbal.

Contudo, a obtenção da profundidade tridimensional, a partir de duas imagens, constitui um tópico importante na VE, com diferentes aplicações como robôs móveis, inspeção de peças industriais, reconhecimento de objetos, reconstrução tridimensional da cena e outros (ZHANG, 2013). Em seguida, alguns projetos propostos em diferentes áreas serão mencionados.

Na área da robótica móvel:

- um projeto de (AESCHIMANN, 2015) propõe um algoritmo para detectar trajetórias seguras e robustas de um robô móvel num sistema de VE, usando laser infravermelho;
- a Micro Veículos Aéreos (MAV), proposta por (BEUL, 2015), apresenta uma plataforma de alto desempenho para pesquisa sobre a navegação autônoma em ambientes complexos 3-D. O MAV está equipado com um sensor omnidireccional e três pares de câmeras estéreo, para realizar tarefas em tempo real. Além de outros projetos que incorporam sistemas de VE numa MAV, propostos por (OLEYNIKOVA, 2015) e (FU, 2015).

Na área da saúde:

- uma proposta de (BORGSTADT, 2015) centra-se num algoritmo de localização multimodal, para intervenções cirúrgicas de cateter. Utiliza filtros de partículas mediante um sistema de VE e um sensor eletromagnético de pose[1];
- uma pesquisa de (FINDEISEN, 2015) apresenta uma proposta de um sistema óptico empregando uma configuração omnidireccional de câmeras estéreo, para análise do comportamento humano em ambientes vivos complexos. O objetivo dessa pesquisa é detectar um objeto, estimar a sua pose e fazer uma análise do seu comportamento;

[1] *Pose*: posição e orientação de um objeto específico em relação à câmera

- outras pesquisas baseadas na VE, tais como sistemas de navegação de imagem e robótica, estão levando tecnologias para melhorar a precisão da cirurgia dental. Ou sistemas de VE para visualização 3-D, em cirurgias com endoscópio (YU, 2015).

Na área da reconstrução tridimensional (3-D):

- um design de interação 3-D proposto por (FETZER, 2015) apresenta uma abordagem baseada na interação da mão, para agarrar e manipular objetos virtuais dentro de um ambiente virtual 3-D. Esse estudo é comparado por dois tipos de análises de visão, monoscópico e VE.

- consequentemente, existem pesquisas de reconstrução de objetos, apresentadas por (HUNG, 2015), baseadas na fotometria estéreo, nas quais se aproveita a luz ambiente, capturando as imagens ao ar livre;

- além de de outros projetos inovadores, destinados a deficientes visuais, que detecta obstáculos na cena 3-D e obtém medidas reais, com smartphone e em tempo real (SAEZ, 2014).

Na área automotiva:

- existem algoritmos de correspondência estéreo para assistência ao condutor, baseados em informações de cores e gradientes das imagens (KOO, 2015), aplicações de anticolisão de veículos (LAI, 2015), além de criação de trinóculos para obter a posição tridimensional do trajeto de um veículo (KWON, 2015)

No entanto, uma parte importante do projeto é contribuir na resolução do problema da **correlação estéreo** entre imagens. Esse problema foi resolvido por diferentes abordagens. Algumas pesquisas na resolução desse problema são, baseadas em filtros e minimização da energia das imagens (MOZEROV, 2015), em variações de iluminação (MOUATS, 2015), canais R G B (LIU, 2015), utilização do algoritmo *Optimal Local Adaptive Radiometric Compensation* (XU, 2015) e segmentação entre imagens a partir de mapas de disparidade (KAREKAR, 2015).

Entretanto, outra parte fundamental do projeto é a calibração das câmeras, a qual tem avanços na calibragem através de ponteiros laser (ZOU, 2014) e métodos não calibrados para obter a pose da câmera (JEONG, 2015).

A fim de apresentar todas as técnicas utilizadas no projeto, existem pesquisas que utilizam marcadores em objetos através da realidade aumentada (SIMõES, 2011) e estimação da orientação do mergulhador abaixo da água através de marcadores no corpo (RENDULIC, 2015).

1.1 Motivação

Os robôs manipulares industriais são amplamente utilizados em indústrias de automóvel, mecânicos, semicondutores, eletrônicos, alimentos e bebidas, tendo substituído gradualmente a força de trabalho. Os robôs industriais estão classificados em robôs em série e paralelo (BROGARDH, 2009).

Os robôs em série são constituídos por links ligados sequencialmente para formar uma cadeia aberta . Têm alta flexibilidade e são capazes de operar em um âmbito maior. No entanto, possuem algumas desvantagens, tais como: a precisão de cada articulação afetado por um erro, resposta mais lenta devido ao mecanismo de série e pobreza na rigidez para lidar com cargas mais pesadas (CHIANG, 2011).

Pelo contrário, os robôs paralelos têm altos índices de rigidez ao peso, elevada firmeza, alta precisão, elevada resposta e boa capacidade de suportar cargas mais pesadas. As principais desvantagens dos robôs paralelos são no espaço de trabalho menor, devido ao seu design mecanismo, análise cinemática complexa e mecanismo fechado (WANG, 2003); (LI; WU, 2004); (LARIBI, 2007); (SAMANTARAY, 2010).

Nos últimos anos, alta resposta, alta precisão e boa rigidez estão na demanda por robôs em muitas aplicações. Os robôs paralelos se tornaram mais populares em automação industrial, devido às suas vantagens do robô tipo serial (ROBERTZ; KELKAR, 2010); (UCHIYAMA, 2010).

Assim, os robôs paralelo são capazes de fazer movimentos no sistema de coordenadas tridimensional, levando o efetuador do robô para uma posição final. O posicionamento do efetuador do robô é conseguido através da cinemática direta (LASEVITCH1, 2012), ou da cinemática inversa (RAMAZINI, 2011).

No entanto, a posição 3-D calculada do efetuador não considera a tolerância de fabricação e montagem das articulações do mecanismo paralelo. De modo que existem erros entre a pose desejada e a pose 3-D calculada do efetuador.

A fim de melhorar a precisão da pose 3-D calculada do efetuador final do robô, surge a motivação principal do projeto, que é obter a pose do efetuador a partir de um sistema de VE. O sistema de VE desenvolvido permitirá medir a pose desejada do efetuador final, corrigindo os erros entre a diferença da pose calculada do robô e da pose medida pelo sistema de VE. Ao combinar os sensores de visão com outros sensores de um robô, são obtidas melhores estimativas de pose de um robô, durante um movimento (DUC; KANG, 2013).

Essa principal motivação surge devido à necessidade existente do robô delta construído no laboratório GACS – PUCRS. Esse robô tem alguns erros de posicionamento no efetuador final, além de certos desvios no posicionamento do efetuador. A partir des-

ses erros, surge a motivação do projeto, que é obter a posição tridimensional de pontos, mediante um sistema VE, a fim de calcular a pose final desejada do efetuador do robô paralelo Delta.

O sistema de VE proposto neste projeto fornecerá um algoritmo, a ser aplicado no robô paralelo Delta, uma vez que a integração entre o robô Delta e o sistema de VE não será efetuado neste projeto. Esta tarefa de integração permanecerá para trabalhos futuros. O algoritmo proposto estará em aberto, para livre uso de outros pesquisadores.

Por outro lado, existem alguns trabalhos desenvolvidos, para posicionar o robô paralelo Delta, apresentados por (CHIANG, 2011), (HAO, 2008) e (LAZZARI, 2008), os quais propõem controlar a posição final do efetuador do robô, através de um ponto reconstruido num sistema de VE.

Assim, também existem outros trabalhos desenvolvidos para robôs em série, que utilizam a VE em conjunto com a servovisão. Esses trabalhos propõem algoritmos para estimar a posição final do robô por meio da servovisão com a configuração das câmeras *eye to hand*, arquitetura IBVS (*Image Based Visual Servoing*) (MOHEBBI, 2014), (PARI, 2009), (NAMMOTO, 2013), arquitetura PBVS (*Position Based Visual Servoing*) (SHA, 2014), (ZIRAKNEJAD, 2007), (GE; JIE, 2007) e câmeras não calibradas (CAI, 2014), (CAI, 2013).

Contudo, a principal motivação deste trabalho é obter o plano de orientação do efetuador do robô paralelo delta, através de um sistema de VE. A maioria dos projetos mencionados acima calcula apenas uma posição do efetuador através de sistemas de VE, mas nenhum deles funciona com o plano de orientação do efetuador. Essa adição dá uma riqueza para o projeto, uma vez que se pode saber com maior precisão o erro de alguma articulação do robô, e, em seguida, corrigir apenas aquele erro.

Sendo assim, determinar a posição e orientação final do efetuador para robôs manipuladores, é um ponto-chave importante em aplicações industriais. Vários métodos e soluções têm sido propostos e aplicados na área da robótica, sendo alguns deles fazer medições através de sensores, tais como odômetros, medição inercial (IMU), entre outros. Contudo, a maioria desses métodos tem diferentes desvantagens na precisão do cálculo da posição. Portanto, para compensar essas desvantagens, os sensores de visão são os que fornecem a medição da posição absoluta, sendo extremamente importantes para determinar a posição de um robô manipulador (HABIB, 2013).

Enquanto a outras áreas, o projeto proposto também pode contribuir, tais como, na medicina, com projetos que determinem a pose do paciente por meio de marcadores, na área da robótica móvel (VANTS [2] ou UAV [3]) que posicionam o robô através do rastreio de marcadores, na área da reconstrução de ambientes 3-D para calcular a posição

[2] VANTs: Veículos Aéreos Não Tripulados
[3] *UAVs: Unmanaged aerial veicles*

tridimensional de vários pontos numa cena através da VE.

Dessa forma, a ideia principal deste projeto é usar a abordagem da VE, para cálculo da profundidade de vários pontos, os quais, permitirão determinar a posição e orientação de um plano, com o foco principal na implementação em robôs manipuladores, além, que o projeto também pode contribuir em diferentes áreas de pesquisa.

1.2 Objetivos

1.2.1 Geral

O objetivo do projeto é desenvolver um algoritmo para reconstruir o espaço tridimensional de objetos, a partir de um sistema de VE, que simula o movimento de objetos em um ambiente dinâmico.

1.2.2 Específicos

a) Realizar a modelagem matemática utilizando VE para a reconstrução de um objeto;

b) realizar a modelagem matemática utilizando VE para a reconstrução de vários objetos;

c) implementar os modelos matemáticos propostos no Matlab para testar os algoritmos;

d) construir um ambiente de experimentação para testar os algoritmos propostos;

e) testar os algoritmos propostos no ambiente de experimentação.

1.3 Organização da dissertação

O trabalho está organizado em seis capítulos:

O Capítulo 1 é a introdução, que dá uma compreensão básica do desenvolvimento ao longo do projeto. O Capítulo 2 traz conceitos básicos incluídos no projeto para o seu desenvolvimento, abrangendo conceitos de espaços euclidianos, calibração de câmeras, parâmetros intrínsecos, parâmetros extrínsecos, projeção da imagem, VE e processamento de imagens, os quais serão usados ao longo do projeto. Já o Capítulo 3 apresenta a modelagem matemática da visão estéreo para reconstruir um objeto único, que abrange processos como a calibração das câmeras e, em seguida, apresenta a reconstrução de um objeto.

O Capítulo 4 apresenta a generalização do modelo matemático para vários objetos a partir do modelo de um objeto. Também são apresentadas as correspondências entre vários objetos, filtros usados para remover o ruído em imagens e um algoritmo que representa

a sequência dos passos para a reconstrução de vários objetos. E o Capítulo 5 analisa os resultados práticos da reconstrução de um objeto, além de apresentar os resultados da reconstrução de vários objetos, incluindo modelagem do plano criado para a reconstrução de diversos objetos, usado para colocar sobre o gimbal.

Por fim, o Capítulo 6 traz as conclusões dos resultados do trabalho, além de apresentar discussões e sugestões para trabalhos futuros.

2 Conceitos Básicos

Neste capítulo, será feita uma introdução aos conceitos utilizados no projeto, tais como: espaços tridimensionais euclidianos, representação homogênea, formação de imagens, visão estéreo e, finalmente, conceitos de processamento de imagens.

Para atingir o objetivo do projeto, é apresentada, na Fig. 3, uma sequência de passos. Essa figura ajuda a identificar e acompanhar mais claramente que conceitos básicos foram utilizados ao longo do projeto.

Figura 3 – Passos para atingir o objetivo do projeto

Fonte: A autora.

2.1 Representação do espaço euclidiano

Um requisito fundamental em robótica e visão computacional é representar a posição e a orientação de objetos em uma cena. Tais objetos incluem robôs, câmeras, peças, obstáculos e trajetórias (CORKE, 2011).

O espaço euclidiano foi descoberto por Chasles e Poisot no início de 1800, sendo amplamente pesquisado e adotado até hoje (YI, 2004). Além disso, permite medir os ângulos, posições e distâncias entre objetos. Em geral, um espaço euclidiano é um conjunto de elementos que satisfazem os axiomas de Euclides.

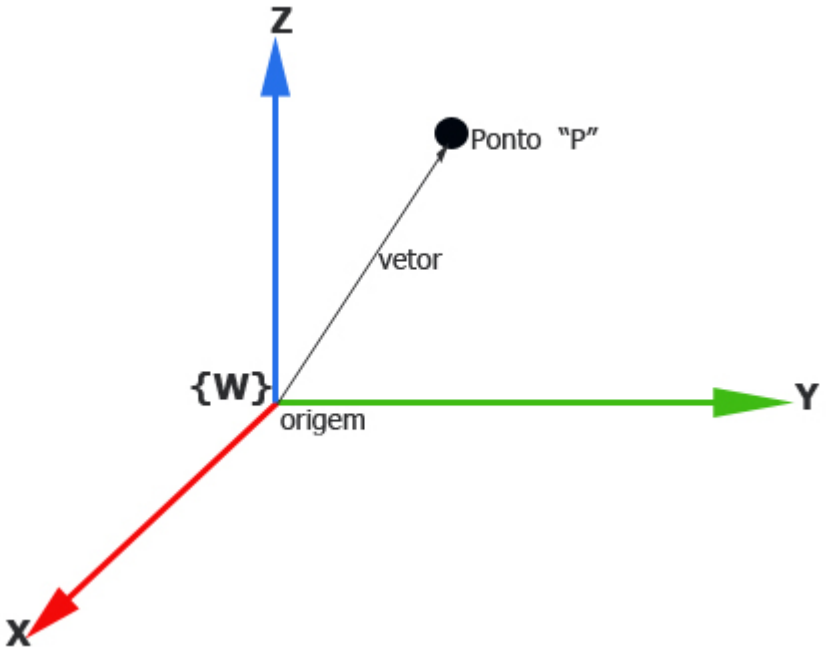
Analiticamente, um espaço tridimensional euclidiano é denotado por $\mathbb{E}^3$. Pode ser representado por um sistema de coordenadas cartesianas, assim como por um ponto P, que pertence ao espaço euclidiano $\mathbb{E}^3$ ($P \in \mathbb{E}^3$). Dessa forma , no sistema de coordenadas cartesianas de números reais, é representado por $P \in \mathbb{R}^3$ (KANATANI, 1989), como mostra a relação a seguir.

$$P = [X_1, Y_1, Z_1]^T = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} \in \mathbb{R}^3 \tag{1}$$

A posição do ponto P, está em relação a um sistema de coordenadas, em que as correspondências entre $\mathbb{E}^3$ e $\mathbb{R}^3$ são essencialmente a mesma coisa.

Contudo, essa representação do ponto P em relação ao sistema de coordenadas cartesianas pode ser visto na Fig. 4, sendo que um sistema de coordenadas cartesianas é uma configuração de eixos ortogonais que se cruzam num ponto de origem.

Figura 4 – Ponto descrito no sistema de coordenadas do mundo {W}



Fonte: A autora.

A figura acima pode ser entendida como a representação do ponto P em relação ao sistema de coordenadas global do mundo {W}.

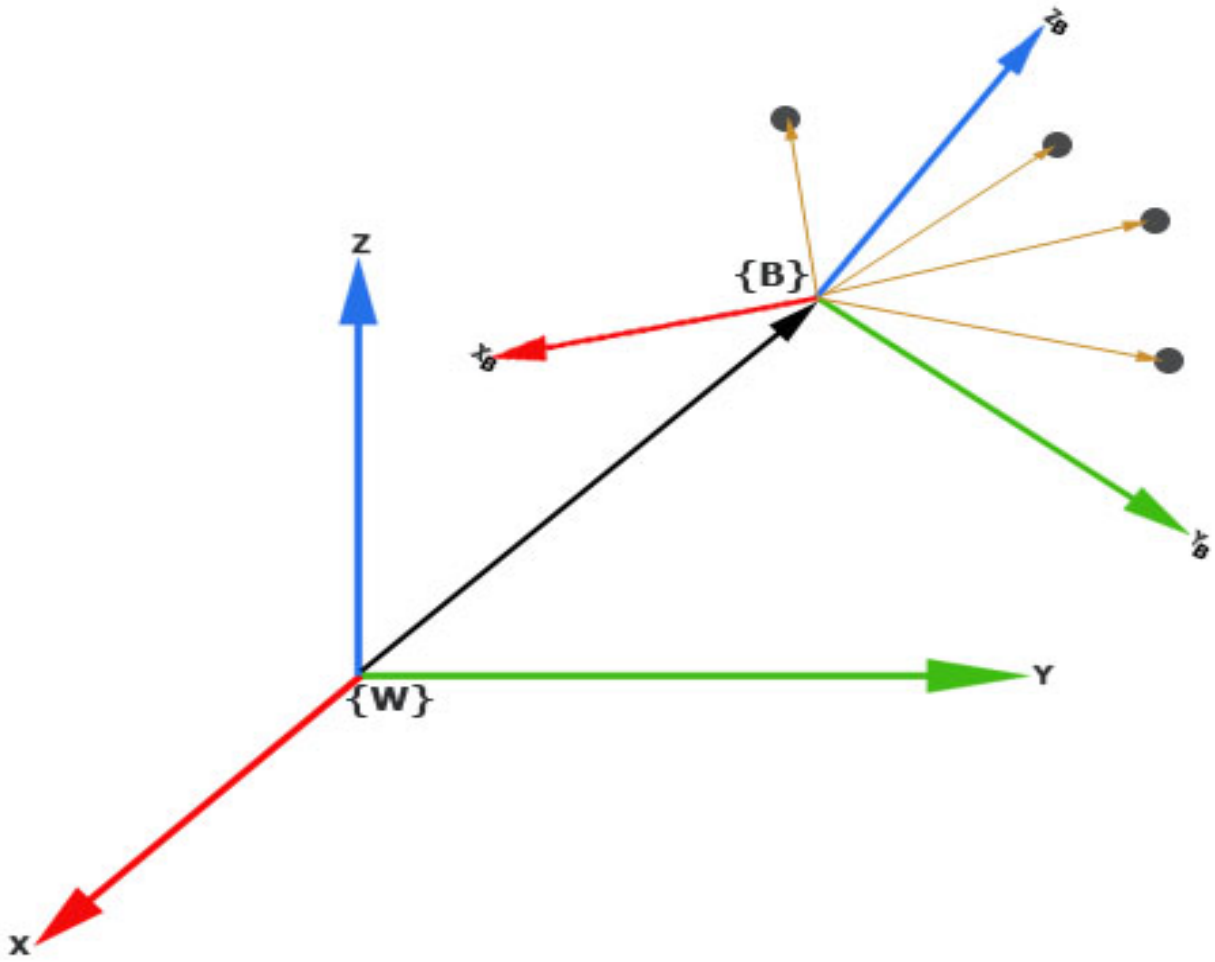
Pela regra da mão direita, pode ser verificado o sistema de coordenadas. Dado que os eixos de coordenadas podem ser qualquer dos vetores $(\vec{e_1}, \vec{e_2}, \vec{e_3})$, com $\vec{e_1} = [1, 0, 0]^T$,

$\vec{e_2} = [0,1,0]^T \in \mathbb{R}^3$ e $e_1 \times e_2$ [1] $= [0,0,1]^T = e_3$. Segundo o sistema de coordenadas cartesianas, o produto cruzado de dois principais eixos X e Y dá outro principal eixo Z.

Além disso, o sistema de coordenadas é ortogonal e deve cumprir o produto interno[2] $\langle u, v \rangle = 0$ e o produto cruzado $\langle x \times y, z \rangle = 0$, entre seus eixos de coordenadas, igual a zero, em que ordem dos fatores define a orientação. Sendo que u e v, são qualquer dos vetores X, Y, Z, do sistema de coordenadas principal, para seus vetores respectivos $\vec{e_1} = [1,0,0]^T$, $\vec{e_2} = [0,1,0]^T, \vec{e_3} = [0,0,1]^T \in \mathbb{R}^3$.

Por outro lado, muitas vezes, é necessário considerar mais do que um único ponto, podendo ser necessário considerar vários pontos, como apresenta-se na Fig. 5. Presume-se que existem várias posições de objetos, representados por pontos, os quais estão em relação ao sistema de coordenadas {B}, em que seus componentes são X_B, Y_B, Z_B.

Figura 5 – Vários pontos em relação ao sistema de coordenadas {B}



FONTE: A autora.

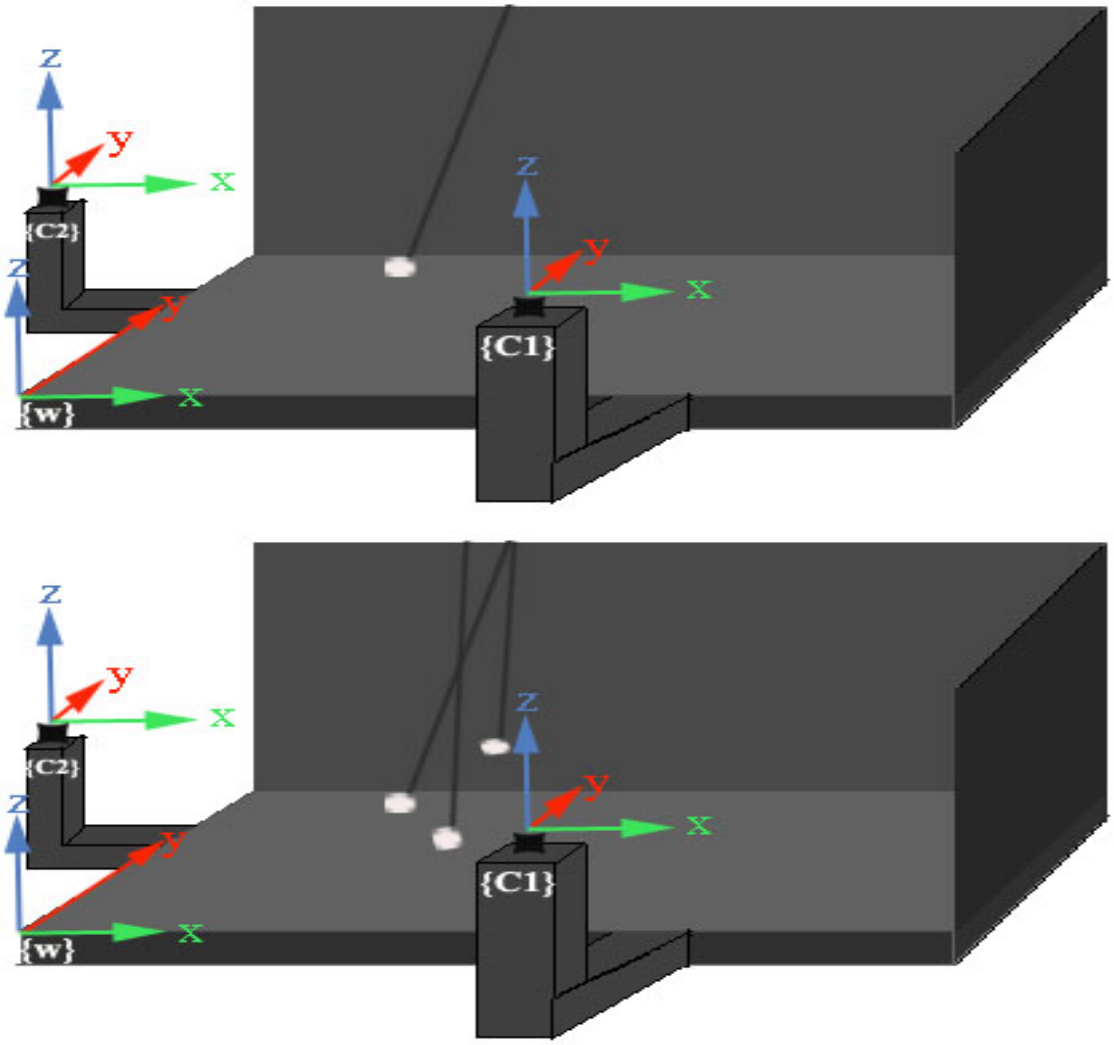
[1] $e_1 \times e_2$: é o produto cruzado entre os dois vetores e_1, e_2, isto será explicado no Capitulo 5, seção 5.3.2

[2] $\langle ., . \rangle$: é o produto interno entre dois vetores qualquer $\vec{u} = (a_1; a_2; a_3)$, $\vec{v} = (b_1; b_2; b_3)$, dado pela relação $\langle \vec{u}, \vec{v} \rangle = a_1\dot{b}_1 + a_2\dot{b}_2 + a_3\dot{b}_3$

A notação da Fig. 5 é lida, como os pontos que estão em relação ao sistema de coordenadas {B}, o qual está em relação ao sistema de coordenadas global {W}.

Relacionando o que foi explicado em tópicos anteriores com o ambiente do projeto, apresentam-se na Fig. 6 os pontos que estão dentro do ambiente.

Figura 6 – Espaço euclidiano para um ponto (figura acima). Espaço euclidiano para vários pontos (figura abaixo)



FONTE: A autora

No caso da Fig. 6 existem três sistemas de coordenadas {C1}, {C2} e {W}, um sistema de coordenadas para a câmera 1, um sistema de coordenadas para a câmera 2 e outro sistema de coordenadas para o mundo, respectivamente. Os pontos dentro do ambiente estão em relação ao sistema de coordenadas {C1}, o qual fica em relação ao sistema de coordenadas {C2}. Além disso, o segundo sistema de coordenadas {C2} fica

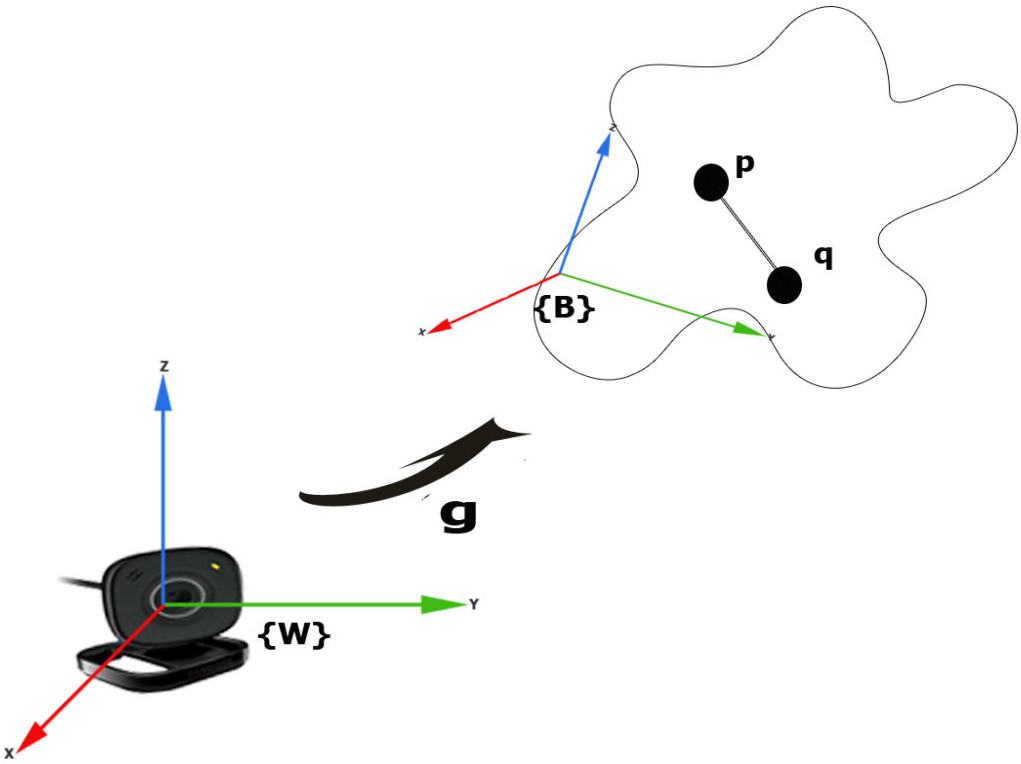
em relação ao sistema de coordenadas do mundo {W}.

2.1.1 Representação do movimento no espaço euclidiano

Até agora, focou-se na representação de objetos estáticos, sem movimento, mas, na maior parte dos casos, os objetos estão em movimento. Portanto, nesta seção será analisada a representação de objetos em movimento.

Para entender melhor o movimento de um objeto, considere a Fig. 7, na qual o objeto está posicionado no sistema de coordenadas {B}. Contudo, existe também outro sistema de coordenadas {W} sobre o qual está posicionada uma câmera que captura os movimentos do objeto. A existência de pontos internos dentro do objeto significa que, apesar do fato de existirem "n" pontos dentro de um objeto, a distância entre cada ponto não varia ao longo do tempo. Outra regra importante é que os pontos dentro de um objeto sempre preservam a sua orientação. Em outras palavras, preservam a distância entre pontos (norma entre vetores) e a orientação (produto cruzado) (TWEED, 2004).

Figura 7 – Movimento de um objeto em relação a uma câmera



FONTE: A autora

Para formalizar o movimento de um objeto, a seguinte definição mostra a transformação de pontos no espaço de um mapa g.

Definição: 2.1 *Movimento de um objeto*

Um mapa de transformação $g : \mathbb{R}^3 \to \mathbb{R}^3$ é conhecido como movimento do objeto ou transformação especial euclidiana (*special euclidian - SE(3)*). Esse mapa deve preservar a norma, o produto cruzado e o produto interno entre seus vetores. Os vetores podem ser qualquer dos eixos de coordenadas X, Y, Z, pertencentes ao sistema de coordenadas:

1. *norma:* $\| g_*(v) \| = \| v \|, \forall v \in \mathbb{R}^3$,
2. *produto cruzado:* $g_*(u) \times g_*(v) = g_*(u \times v), \forall u, v \in \mathbb{R}^3$.
3. *produto interno:* $\langle u, v \rangle = \langle g_*(u), g_*(v) \rangle, \forall u, v \in \mathbb{R}^3$

Por outro lado, no movimento do objeto, deve-se considerar um sistema de coordenadas com seus eixos principais ortonormais, compostos por três vetores $e_1, e_2, e_3 \in \mathbb{R}^3$, do seguinte modo:

$$e_i^T e_j = \delta_{ij} \doteq \begin{cases} 1 & \text{for} \quad i = j, \\ 0 & \text{for} \quad i \neq j. \end{cases}$$

Os vetores são dispostos de modo que eles respeitam a regra da mão direita: $e_1 \times e_2 = e_3$.

Finalmente, o movimento de um objeto é representado num sistema de coordenadas orthonormal, a partir de um mapeamento g. Esse mapeamento g está composto pelos movimentos de translação e rotação do objeto. Além disso, o movimento do objeto pode ser definido no seu sistema de coordenadas quando preserva o produto cruzado e o produto interno entre seus eixos componentes (KANATANI, 1989).

2.1.2 Representação do movimento: rotação e translação

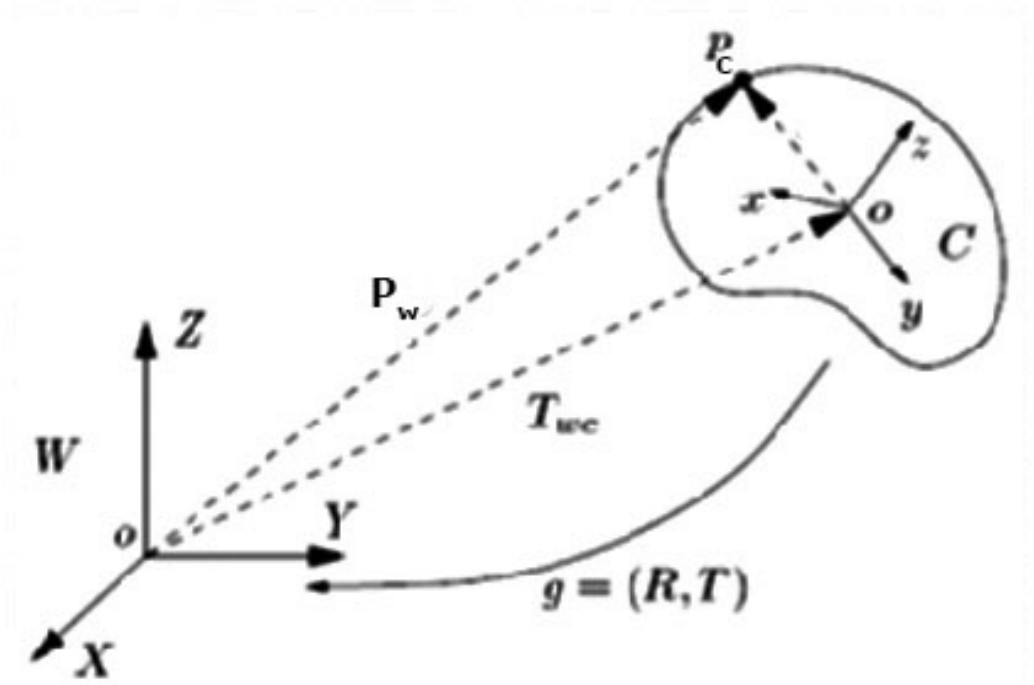
O movimento de um objeto consiste basicamente em dois componentes, rotação e translação. O movimento de translação pode ser entendido como a orientação inalterada, isto é, manter a forma e o tamanho dos dados ou objetos movidos, de acordo com o deslizamento do vetor. A rotação é o movimento de mudança em orientação de um corpo ou de seu sistema de coordenadas (LIN, 2011).

Para ilustrar esses conceitos, é utilizada a Fig. 8. A posição de um objeto é nomeada como P_C. Uma câmera fixa está localizada sobre o sistema de coordenadas {C}, o qual está em relação ao sistema de coordenadas do mundo {W}.

Uma vez que:

a) $P_C \in \mathbb{R}^3$: é o objeto em relação ao sistema de coordenadas {C} representado por um vetor na mesma posição;

Figura 8 – Posição do objeto P_C em relação ao sistema de coordenadas da câmera {C}, a qual está em relação ao sistema de coordenadas do mundo {W}



FONTE: (YI, 2004)

b) $T_{wc} \in \mathbb{R}^3$: é o vetor de translação entre a origem do sistema de coordenadas {W} e o sistema de coordenadas da câmera {C}.

c) $R_{wc}^{3\times3} \in \mathbb{R} \in SO(3)$: é a matriz de rotação do sistema de coordenadas da câmera {C}.

d) $P_W \in \mathbb{R}^3$: é a posição do objeto P_C, mas em relação ao sistema de coordenadas do mundo {W}.

Entretanto, a posição do objeto PC em relação ao sistema de coordenadas da câmera {C} é conhecida, e busca-se conhecer a posição do objeto P_W, em relação ao sistema de coordenadas {W}. Portanto, é apresentada a seguinte relação:

$$P_W = R_{wc} \cdot P_C + T_{wc} \tag{2}$$

Entretanto, esta relação da equação (2) pode ser representada pelo movimento de um objeto g.

$$g = (R_{wc}, T_{wc}) \tag{3}$$

Substituindo a equação (3) na equação (3):

$$P_W = g_{wc}(P_C)$$

Portanto, a configuração geral do movimento de um objeto, é representado por:

$$SE(3) = \{g_w c = (R_w c, T_w c) | R_w c \in SO(3), T_w c \in \mathbb{R}^{3\times3}\} \tag{4}$$

Por outro lado, para voltar para trás, e saber a posição do objeto P_C em relação ao sistema de coordenadas {C}, deve-se multiplicar a matriz de rotação inversa ou a transposta, pela posição do objeto em relação ao sistema de coordenadas do mundo {W}.

$$P_C = R_{wc}^{-1} P_W = R_{wc}^T P_W \tag{5}$$

Contudo, para terminar de definir todas as componentes das configurações do movimento, a matriz de rotação R_{wc}, deve ser definida a seguir.

Matriz ortogonal de rotação

A matriz de rotação $\mathbb{R}^{3\times3}$ é definida da seguinte maneira (YI, 2004).

$$R_{WC} = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix} \in \mathbb{R}^{3\times3}$$

Desde que r_1, r_2, r_3 são os vetores dos eixos de coordenadas "X, Y, Z", os quais, formam um quadro ortonormal, da seguinte maneira:

$$r_i^T r_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Isso pode ser escrito na forma matricial como:

$$R_{WC}^T R_{WC} = R_{WC} R_{WC}^T = I$$

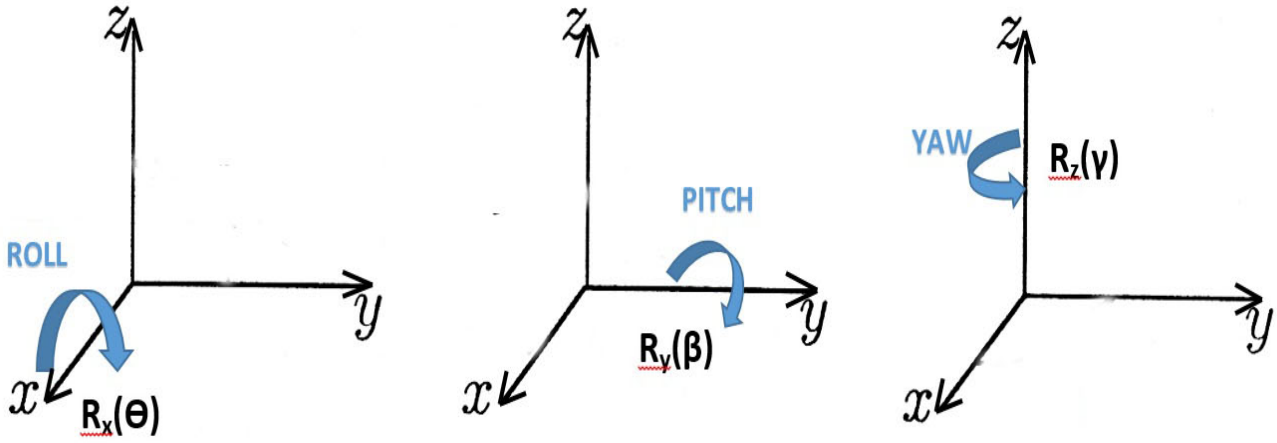
Qualquer matriz que satisfaz a identidade acima é chamada uma matriz ortogonal. Resulta da definição acima que o inverso de uma matriz ortogonal é simplesmente sua transposta: $R_{wc}^{-1} = R_{wc}^T$. Além disso, a determinante de R_{WC} deve ser igual a 1.

Assim, de modo geral, a matriz de rotação pode ser definido pela seguinte relação.

$$SO(3) \doteq \{R_{wc} \in \mathbb{R}^{3\times3} | R_{wc}^T R_{wc} = I, det(R_{wc}) = 1\} \tag{6}$$

Uma vez que a matriz de rotação R_{WC} é composta pelo produto entre as matrizes ortogonais de rotação dos diferentes eixos X, Y, Z (Fig. 9)), do sistema de coordenadas, em que a ordem de multiplicação das matrizes de rotação dos eixos determina a orientação (LIN, 2011).

Figura 9 – Rotação nos três eixos X, Y, Z



FONTE: (LIN, 2011)

Portanto, a rotação completa de um objeto está dada por:

$$R_{wc} = R_x(\theta) * R_y(\beta) * R_z(\gamma) \tag{7}$$

Sendo $R_x(\theta), R_y(\beta), R_z(\gamma)$ as matrizes de rotação em relação aos eixos de coordenadas X, Y, Z:

Em relação ao eixo X

$$R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix}$$

Em relação ao eixo Y

$$R_Y(\beta) = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix}$$

Em relação ao eixo Z

$$R_Z(\gamma) = \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.1.3 Representação homogênea

As relações da representação do movimento de um objeto, descrito no tópico anterior, ainda não são lineares, mas sim são afins. De modo que, para converter uma transformação afim para uma representação linear, é necessário utilizar coordenadas homogêneas, que é uma extensão do espaço euclidiano $\mathbb{R}^3$ ao hiperplano $\mathbb{R}^4$ (YI, 2004).

Entretanto, dado o ponto $P = [X, Y, Z] \in R^3$ da equação (1), pode ser convertido de uma representação afim para uma representação linear usando coordenadas homogêneas, representado por $\overline{P}$ (NAKAMURA, 2009), da seguinte maneira:

$$\overline{P} = \begin{bmatrix} P \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \in \mathbb{R}^4 \tag{8}$$

Dessa forma, continuando com o exemplo apresentado na Fig. 8 do tópico anterior, aplica-se a definição de coordenadas homogêneas na equação (2).

$$\overline{P}_w = \begin{bmatrix} R_{wc} & T_{wc} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} P_C \\ 1 \end{bmatrix} \tag{9}$$

Sendo $\overline{g_{wc}}$ e $\overline{P_c}$:

$$\overline{g_{wc}} = \begin{bmatrix} R_{wc} & T_{wc} \\ 0 & 1 \end{bmatrix} \tag{10}$$

$$\overline{P_c} = \begin{bmatrix} P_C \\ 1 \end{bmatrix} \tag{11}$$

De uma forma mais geral, a matriz $\overline{g} \in \mathbb{R}^{4\times4}$ é:

$$\overline{g} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tag{12}$$

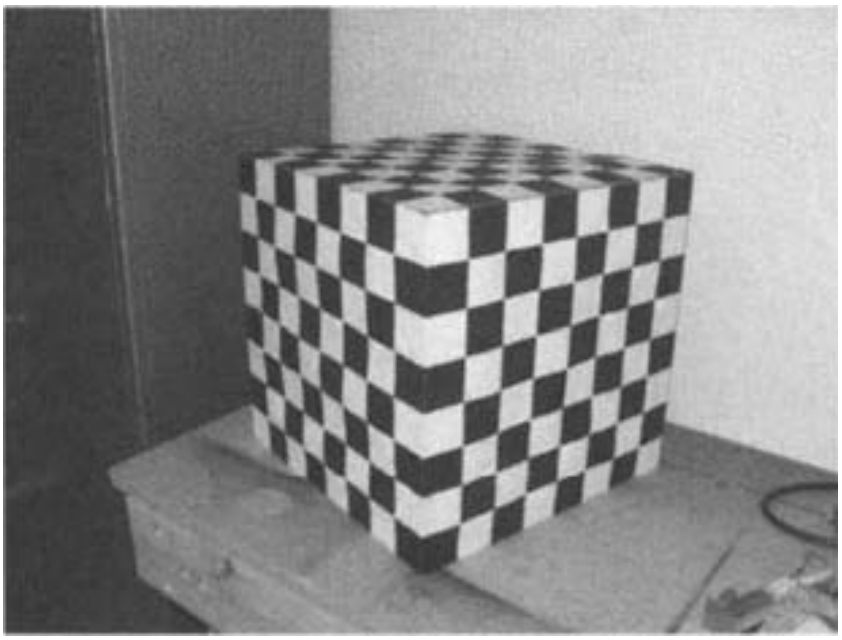
Generalizando a representação homogênea do movimento de um objeto:

$$SE(3) = \left\{ \overline{g} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} | R \in SO(3), T \in \mathbb{R}^{3\times 1} \right\} \subset \mathbb{R}^{4x4} \tag{13}$$

2.2 Calibração de câmeras

A calibração de câmeras é o processo de determinar os parâmetros intrínsecos e extrínsecos em relação a um sistema de coordenadas do mundo (MONACO, 2007). Existem técnicas de calibração, como as propostas de (BOUGUET, 2013), que requerem uma série de imagens do ambiente, podendo ser realizadas a partir do plano do tabuleiro de xadrez. Algumas dessas técnicas são: calibração através de um equipamento (Fig. 10) e calibração com um padrão plano (Fig. 11) (YI, 2004).

Figura 10 – Calibração da câmera com um equipamento



FONTE: (YI, 2004).

Entretanto, a calibração com um equipamento é um método que serve para calibrar a câmera, quando se tem acesso à câmera e se pode colocar um objeto conhecido na cena.

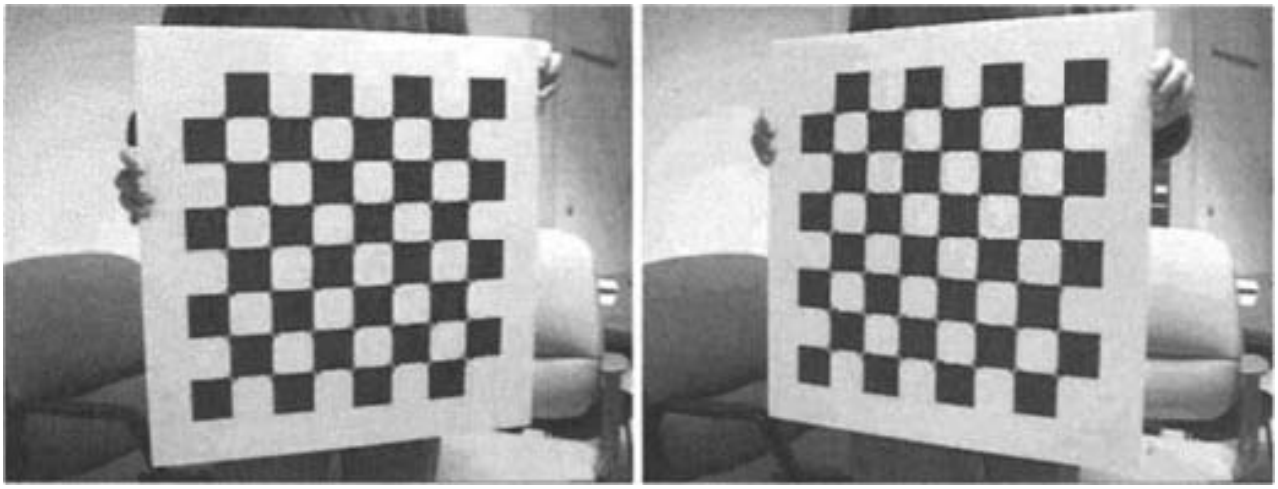
Sob essas condições, pode-se utilizar um objeto como um equipamento de calibração, em relação a algum sistema de coordenadas de referência. Observe, na Fig. 10, que o equipamento de calibração poderia ser um objeto real fabricado principalmente com a finalidade de calibração da câmera, ou simplesmente um objeto na cena com geometria

conhecida. Por exemplo, uma bola de golfe com pontos pintados sobre ela, ou a borda de uma roda de carro cujo alinhamento precisa ser calculado a partir de câmeras.

Embora a abordagem descrita acima requeira apenas uma imagem de um objeto conhecido, ele não retornará uma estimativa única e bem condicionada da calibração, se o objeto fosse plano. Além disso, as plataformas de calibração não planas não são fáceis de fabricar.

Deste modo, aparece a calibração com um padrão plano, que é uma abordagem commummente adaptada, a qual, consiste em capturar várias imagens de um plano conhecido, tal como é um tabuleiro de xadrez, como mostra a Fig. 11, uma vez que nessa calibração existe a liberdade de escolher o sistema de coordenadas de referência e pode-se escolher alinhar o tabuleiro de xadrez com o sistema de coordenadas.

Figura 11 – Calibração da câmera com um padrão plano



FONTE: (YI, 2004).

Contudo, se existe o acesso à câmera e o objeto é conhecido e disponível, esse procedimento é conceitualmente simples, e várias distribuições de software de calibração estão disponíveis (YI, 2004).

Por outro lado, do processo de calibração de câmeras, obtém-se os parâmetros extrínsecos e parâmetros intrínsecos, os quais serão explicados a seguir.

a) ***Parâmetros extrínsecos:*** os parâmetros extrínsecos relacionam o sistema de coordenadas da câmera com o sistema de coordenadas do mundo, descrevendo a posição e orientação da câmera no mundo 3-D. Assim, existem dois parâmetros extrínsecos: a matriz de rotação $\mathbf{Rc} \in \mathbb{R}^{3\times3}$ e o vetor de translação $\mathbf{T}_c \in \mathbb{R}^{3\times1}$. A matriz de ro-

tação pode ser representada pela fórmula de *Rodrigues* (SHIMIZU, 2008), (ZHOU, 2012).

b) ***Parâmetros intrínsecos:*** utilizados para transformar as coordenadas do objeto físico projetado ao plano da imagem. Existem quatro parâmetros intrínsecos (SHIMIZU, 2008), (URFALIOGLU, 2004):

- **Distância focal:** representa a distância, entre o centro ótico da lente da câmera e o foco da câmera;
- **centro ótico da câmera:** representa o centro ótico da câmera da lente. É o centro em que passa a interseção da luz *pinhole* e o sistema de coordenadas principal da imagem x do plano ótico;
- **coeficiente de desfasamento:** representa o ângulo de desfasamento entre os eixos x e y das coordenadas da imagem;
- **distorções:** representa os coeficientes de distorção da imagem (radial e tangencial).

Os parâmetros intrínsecos e extrínsecos apresentados nesta seção da calibração de câmeras serão entendidos melhor através de conceitos tais como: imagens através do *pinhole*, modelo geométrico de formação de imagens e perspectiva de câmera ideal, os quais serão explicados nos seguintes tópicos.

2.2.1 Formação da Imagem

A visão computacional em sentido amplo é o problema inverso da formação de imagens. Portanto, para projetar algoritmos de visão, é necessário desenvolver um modelo de formação de imagens adequado. De modo que o nível de abstração e complexidade do modelo da formação de imagens, fora das restrições físicas e matemáticas, deve ter um modelo gerenciável com esforço razoável (DHAWAN, 2011).

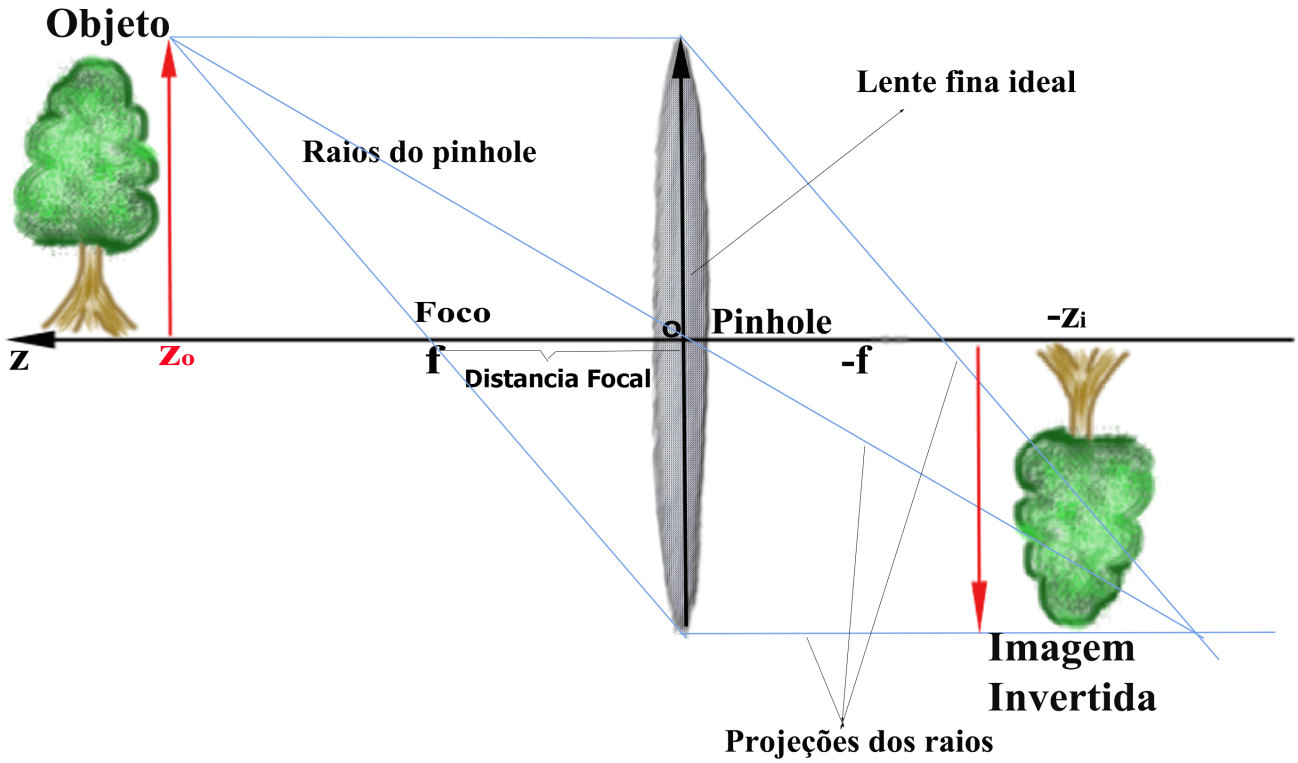
O estudo das imagens foi durante séculos o domínio da produção e composição artística, mais do que a matemática e a engenharia. A compreensão da geometria de formação da imagem inclui vários modelos para projetar o mundo tridimensional num plano. Essa projeção faz parte de uma formulação geométrica, que foi pesquisada desde a obra de Euclides, no século IV AC, até hoje, que esta sendo pesquisado com grande interesse (LIANG, 2011).

Nos seguintes tópicos serão apresentados: imagens através das lentes, imagens através do *pinhole*; e o processo de formação de imagens segundo o modelo geométrico, os quais irão ajudar a compreender melhor a formulação do projeto.

2.2.1.1 Imagens através das lentes

Para estudar a formação de imagens através das lentes, é necessário mencionar algumas das características que descrevem o processo dos raios de luz. Essas características podem ser observadas através da Fig. 12.

Figura 12 – Geometria da formação da imagem para uma lente fina



FONTE: A autora.

Assim:

a) *Plano ótico*: é o plano central da lente;

b) *centro ótico*: é o centro geométrico da lente 'o'. Tem a propriedade que cada raio que passa por ele não sofre qualquer desvio. A interseção é conhecida como *pinhole*, que produz uma imagem tênue;

c) *eixo principal 'z'*: é a linha que passa através do centro ótico e é perpendicular ao plano ótico;

d) *principais focos f e -f'*(foco do objeto e foco da imagem, respetivamente): eles são um par de pontos, correspondendo a cada uma superfície onde os raios (ou suas extensões) incidentes na lente paralela ao sistema de coordenadas principal se intersectam;

e) *distância focal 'f'*: é a distância entre o centro ótico 'o' e o foco 'f'. É a metade entre o objeto(z_0) e o centro ótico 'o';

f) *imagem invertida*: é o resultado da formação da imagem através do centro ótico, tornando-se geralmente em pixels;

g) *raios*: são raios de luz que vêm do objeto para o centro ótico e são projetadas para a imagem formada.

Todo esse estudo é conhecido como formação da imagem através de uma lente fina (KIM, 2007). Esses dados fazem parte da geometria da formação da imagem, chamada lei de Gauss ou lei da lente, obtida por semelhança de triângulos (DHAWAN, 2011),é apresentada a seguir.

$$\frac{1}{f} = \frac{1}{z_o} + \frac{1}{z_i} \tag{14}$$

Sendo:

z_0: a distância entre o ponto 'o' da lente até o objeto.

z_i: a distância entre o ponto 'o' da lente até a imagem.

2.2.1.2 Vergência das lentes

A vergência é uma característica das lentes, entendida como uma medida da capacidade da lente de desviar a luz (ANTONIAZI, 2013).

A vergência D ou convergência de uma lente é uma grandeza que corresponde ao inverso de sua distância focal f (ANTONIAZI, 2013), (DESOUZA, 2012), a qual está dada pela seguinte relação:

$$D = \frac{1}{f}$$

Onde:

- Lente convergente: $D > 0$

- Lente divergente: $D < 0$

A unidade de medida usual da vergência é a dioptria (di), que corresponde ao inverso do metro ($m^{-}1$) (DESOUZA, 2012).

Quando a lente é divergente, a distância focal é negativa, portanto a vergência também será negativa. Quando a lente for convergente, a vergência será positiva.

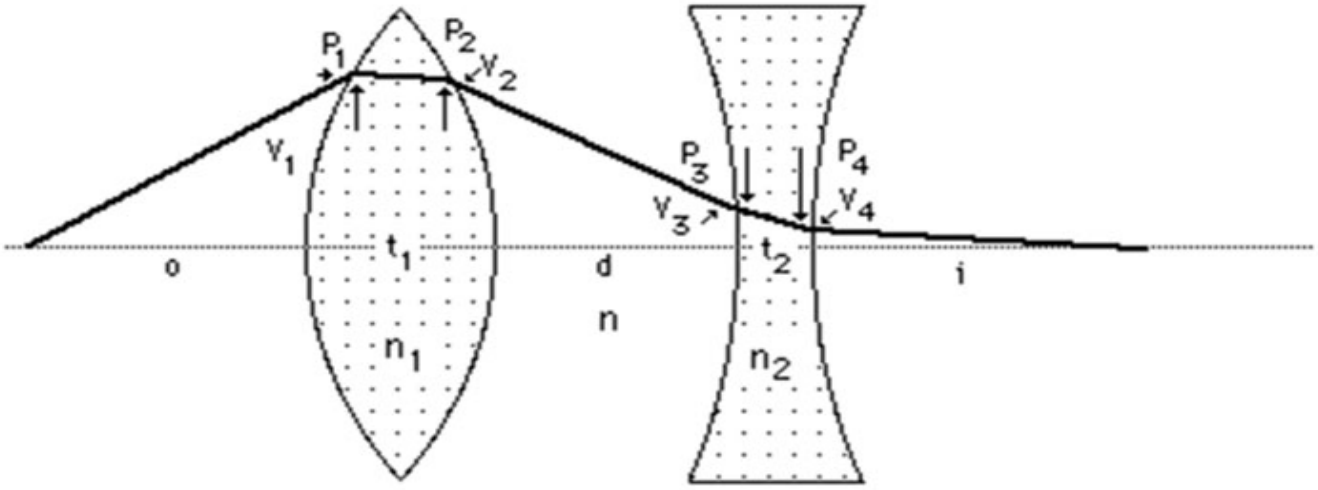
Uma vergência de $5[di]$ significa que a lente a ser usada é uma lente convergente, com uma distância focal 0.2 m ou 20 cm. Por outro lado, uma vergência de $-5[di]$ significa

que a lente a ser usada é uma lente divergente, com uma distância focal de 0.2 m ou 20 cm (COURROL, 2006).

Ao ver uma receita de óculos, esta terá as medidas, por exemplo, $5[di]$ ou $-5[di]$], estas medidas indicam as vergências das lentes (ANTONIAZI, 2013).

O exemplo a seguir da Fig. 13 mostra a lente da esquerda que é convergente e a lente da direita que é divergente, chamado de lente positiva e lente negativa, com suas potências respetivas. Essas potências podem ter qualquer valor positivo ou negativo. Em cada superfície, aplica-se a fórmula $V = V + Ps'$, na qual é calculada a mudança da vergência entre as superfícies (ROD, 2010).

Figura 13 – Exemplo da vergência da lente



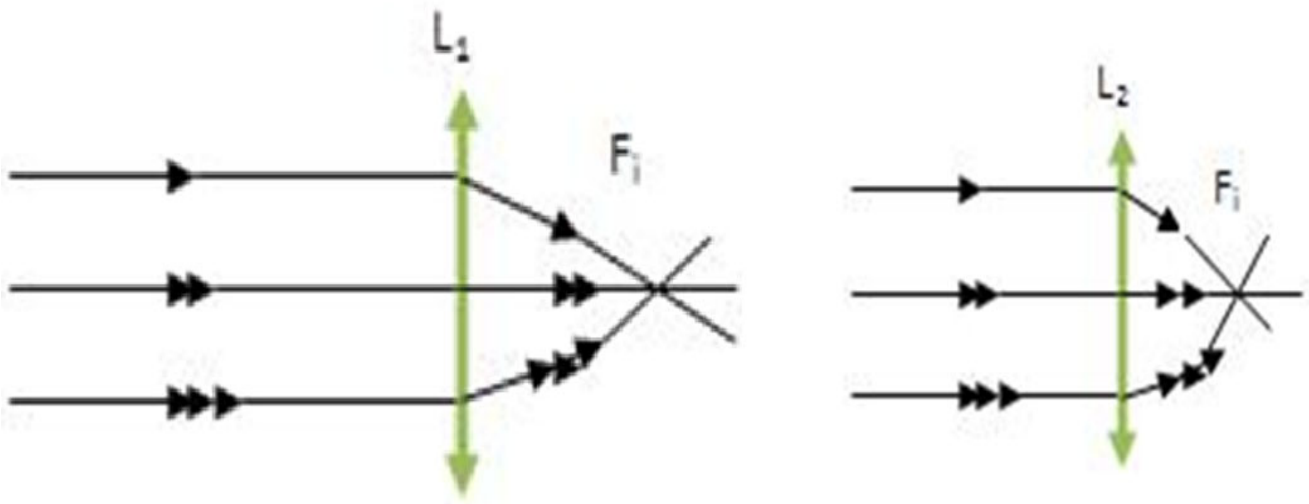
FONTE: (ROD, 2010)

Sendo, P_s a potência da superfície e V a vergência. Alem disso, as vergências V_1, V_2 são maiores na lente convergente do que as vergências V_3, V_4, que são menores para a lente divergente. Como também, simultaneamente, as potências P_1, P_2 são maiores do que as potências P_3, P_4, para as lentes convergentes e divergentes, respetivamente.

Por outro lado, para medir o poder de uma lente divergir ou convergir entre os raios de luz, é definida uma grandeza chamada **Vergência** ou **Convergência** da lente. É importante saber que, quanto menor for a distância focal de uma lente, maior será o poder de convergir dos raios de luz (COURROL, 2006), como pode ser visto na Fig. 14.

Nos esquemas da Fig. 14, a lente L_2 é mais convergente do que a lente L_1. Isso se explica pelo fato de a lente L_2 ter uma menor distância focal, convergindo mais repentinamente os raios de luz.

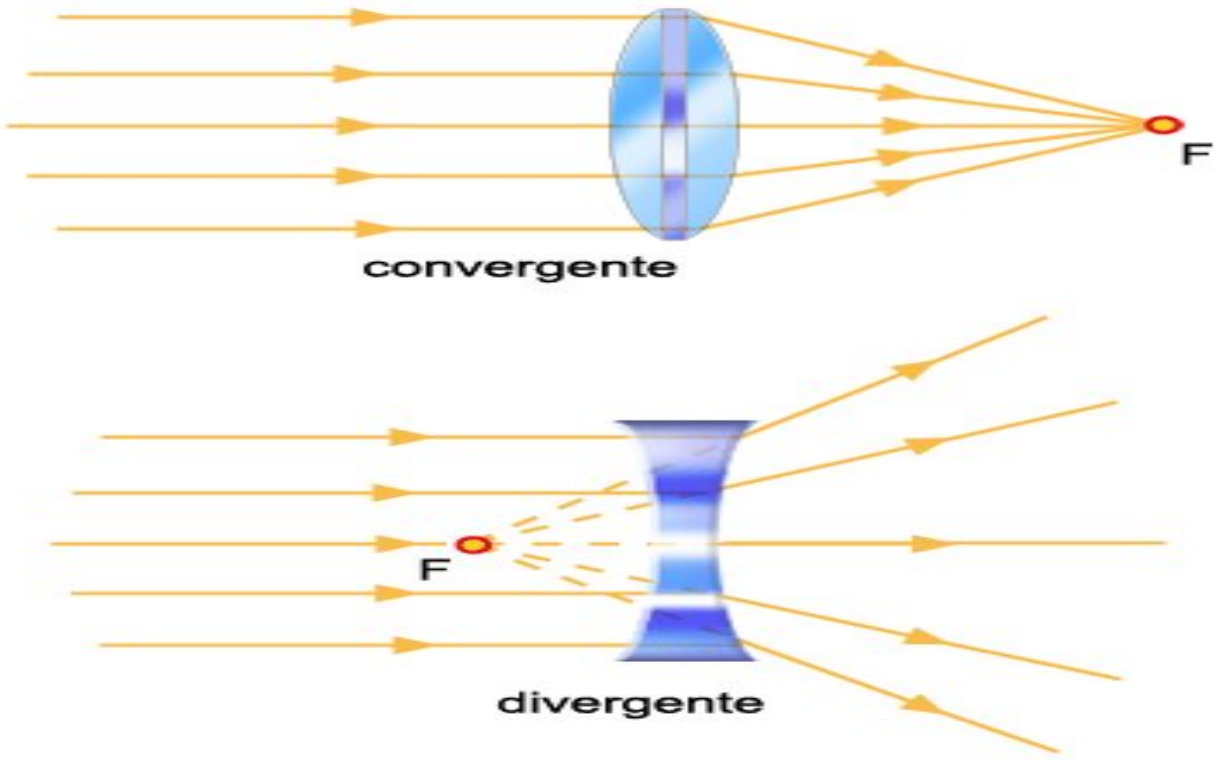
Figura 14 – Convergência da lente focal



FONTE: (COURROL, 2006)

Mas como é que funciona a convergência ou divergência do foco da lente?, a seguir, na Fig. 15 apresentam-se alguns detalhes sobre esse tema.

Figura 15 – Convergência e divergência da lente



FONTE: (LIMA, 2010)

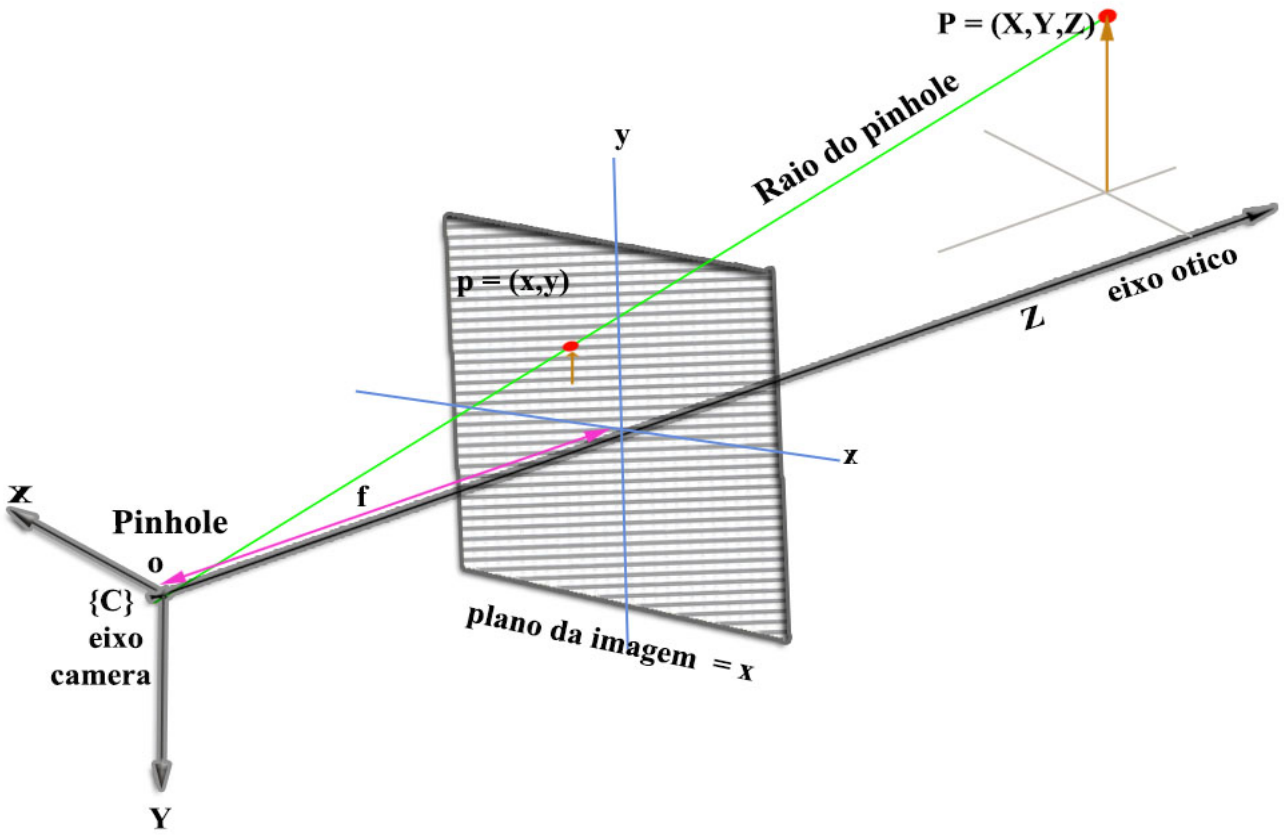
Na lente convergente, os raios de luz refratados convergem para um ponto em comum, no entanto os prolongamentos dos feixes são refratados inversamente numa lente divergente.

É esse ponto de convergência que se denomina foco: ponto para o qual todos os feixes refratados, ou os seus prolongamentos, convergem e formam a imagem de um objeto. Sendo que, numa lente convergente, o foco encontra-se após a lente (sistematicamente, a posição da imagem real assume sinal negativo), e, numa lente divergente, o foco localiza-se antes da lente (posição da imagem é positiva, assim é considerada uma imagem virtual) (LIMA, 2010).

2.2.1.3 Imagens através de um pinhole

Se a abertura da lente fina diminui para zero, todos os raios de luz passam através do centro ótico 'o', sem serem desviados; esse modelo é conhecido como um modelo de câmera ideal *pinhole*, conforme é mo Fig. 16.

Figura 16 – Modelo da imagem através do *pinhole*



FONTE: A autora

A partir do que foi explicado na seção 2.1.1.1, sobre a formação de imagens através das lentes, nesta seção será abordada a formação de imagens numa perspectiva ideal.

Para isso, é utilizada a lei da lente da equação (14) e a Fig. 16, na qual é considerado considerado um ponto $P = [X, Y, Z]^T \in \mathbb{R}^3$ qualquer, de algum objeto do mundo real[3], um plano da imagem "**x**" com coordenadas $x, y \in \mathbb{N}^2$, o ponto $p = [x, y]^T \in \mathbb{N}^2$ projetado na imagem e o sistema de coordenadas da câmera {C}.

Assim da Fig. 16, o processo de projeção em perspetiva ideal, considera um ponto P que passa pelo plano da imagem "**x**", projetando-se ao ponto $p = [x, y]^T$ em coordenadas da imagem; além disso, um raio de *pinhole*, vã até o centro ótico "o" da lente da câmera; também existe, uma distância focal "f" que passa pelo eixo ótico "Z" do plano da lente, entre o centro otico "o" e o centro do plano da imagem **x** (STURM, 2005).

Contudo, a partir da lei da lente da equação (14), são formadas as equações da projeção da imagem em perspectiva ideal, que segue um mapa $\pi = \mathbb{R}^3 \rightarrow \mathbb{R}^2$, denotado por $X \rightarrow x$, apresentado a seguir:

$$x = f\frac{X}{Z} \tag{15}$$

$$y = f\frac{Y}{Z} \tag{16}$$

Onde:

X, Y, Z são as coordenadas do ponto tridimensional P.

x,y são os pixeis do ponto projetado p, no plano da imagem.

Assim, o ponto projetado na imagem na representação homogênea é:

$$\left[f\frac{X}{Z}, f\frac{Y}{Z}, 1\right] \in \mathbb{R}^3 \tag{17}$$

2.2.1.4 Modelo geométrico da formação de imagens – perspectiva da câmera ideal

A projeção da imagem em perspectiva ideal através do *pinhole*, apresentada no tópico anterior, deu uma abordagem geral sobre a projeção de um ponto tridimensional ao espaço bidimensional de uma imagem. No entanto, para estabelecer uma exata correspondência entre o espaço 3-D do ponto P e a projeção do ponto p no plano da imagem 2-D, a seguir, apresenta-se um modelo matemático mais ajustado à representação da câmera ideal, além de introduzir à representação do movimento de um objeto.

[3] Na Fig. 1, apresenta-se a configuração de um sistema de VE para um ponto qualquer de uma imagem

Entretanto, leva-se a equação (15) à forma matricial da seguinte forma:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{18}$$

Isolam-se os pixels da imagem e o eixo Z do ponto P na forma homogênea.

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{19}$$

Realiza-se a descomposição da matriz.

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{20}$$

Onde as matrizes K_f e Π_0, são:

$$K_f = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3x3} \tag{21}$$

$$\Pi_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{3x4} \tag{22}$$

Nesse passo, apresenta-se o movimento de um objeto na forma homogênea, visto na seção 2.1.2, equação(9).

$$\overline{\mathbf{P}}_w = \begin{bmatrix} P_w \\ 1 \end{bmatrix} \begin{bmatrix} R_{wc} & T_{wc} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} P_C \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{23}$$

Dessa forma, são combinadas as duas equações (20) e (23), do modelo geométrico da câmera ideal e da relação do movimento de um objeto.

$$Z\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{24}$$

Fazendo a mudança de variável de $Z = \lambda$, onde λ representa a profundidade da imagem, que é o valor desconhecido a ser calculado. Então, o modelo geral da projeção da imagem é:

$$\lambda \mathbf{x} = K_f \Pi_0 \overline{\mathbf{g}} \mathbf{X}_0 \tag{25}$$

2.2.1.5 Câmera com parâmetros intrínsecos

Até agora, apresentou-se a correspondência entre o plano tridimensional e o plano bidimensional, do ponto de vista puramente físico, independentemente de a imagem ser capturada. Esta seção fará com que o modelo da equação (25) seja utilizável, ao fazer uma correspondência entre o plano da retina da câmera e a matriz de pixels do plano da imagem, como pode ser visto na Fig. 17.

Entretanto, na Fig. 17, existe um ponto (o_x, o_y) que é o ponto central do plano da imagem; (s_x, s_y) são os valores de escalonamento da imagem no eixo x, y; (x',y') são os novos pontos a calcular em pixels; o ponto $p = (x, y)$ é a projeção em pixels do ponto no espaço tridimensional $P = (X, Y, Z)$ e o eixo Z é o eixo ótico que passa através do centro da câmera do plano da imagem.

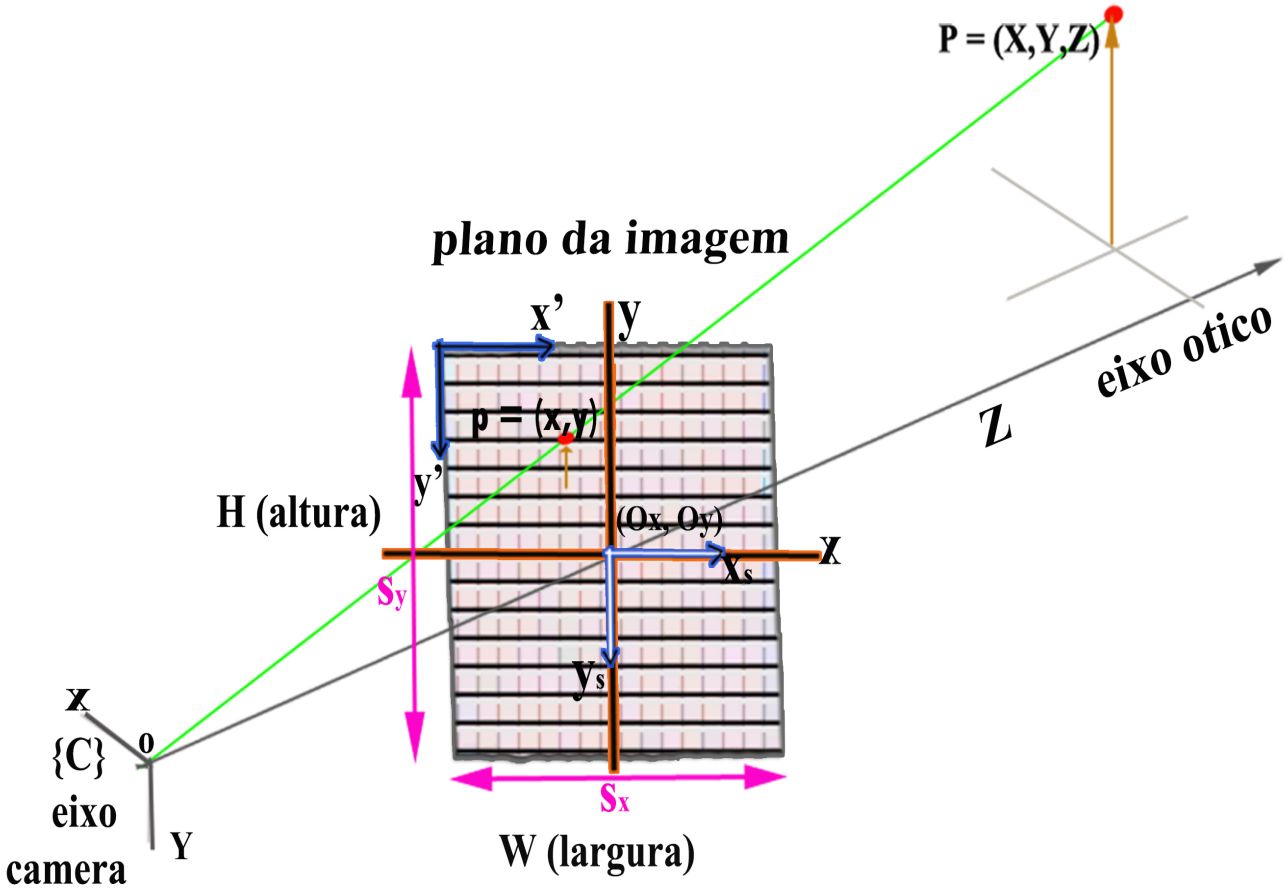
Então, o primeiro passo consiste em especificar as unidades entre os eixos x e y do plano da imagem, a partir de x_s, y_s que são os valores escalonados:

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{26}$$

Deste modo, utilizando os valores escalonados, leva-se a origem do sistema de coordenadas de referência ao canto do plano da imagem

$$x' = x_s + o_x$$
$$y' = y_s + o_y$$

Figura 17 – Correspondência entre o plano da retina da câmera e a matriz de pixels do plano de imagem



FONTE: A autora

Entretanto, as novas coordenadas em pixels da imagem na representação homogênea são:

$$\mathbf{x'} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{27}$$

Assim, a equação (27) pode ser reescrito como a matriz K_s com um "*fator inclinado*" s_θ:

$$Ks = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3x3} \tag{28}$$

Onde o *fator inclinado* é proporcional à $\cot(\theta)$, sendo que θ é o ângulo entre os eixos da imagem x_s e y_s. Na maioria das aplicações é comum presumir que $s_\theta = 0$

Contudo, a seguir, relaciona-se, à equação (24) do modelo geométrico da câmera ideal com a matriz K_s da equação (28)

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{29}$$

Desta forma, obtém-se a matriz K, conhecida como parâmetros intrínsecos da câmera, a partir das matrizes K_s da equação (28) e K_f da equação (21).

$$K = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{30}$$

Portanto, o modelo geral com parâmetros intrínsecos e extrínsecos, que realiza uma correspondência entre o plano da retina da câmera e a matriz de pixels do plano da imagem, é apresentado a seguir:

$$\lambda \mathbf{x}' = K\Pi_0 X = K\Pi_0 \overline{\mathbf{g}} X = \underbrace{\begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Parámetros intrínsecos}} \underbrace{\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}}_{\text{Parámetros extrínsecos}} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{31}$$

Consequentemente, o modelo geométrico da equação (31) representa a projeção de um ponto P em 3-D ao ponto p do plano da imagem 2D em pixeis, onde, $\mathbf{x}$' são os pixels em coordenadas homogêneas do ponto p e λ é a profundidade da imagem, que foi perdida no processo de projeção da imagem.

Assim, de maneira geral, obtêm-se os parâmetros extrínsecos (translação e rotação), que descrevem a pose da câmera, e os parâmetros intrínsecos, que descrevem os parâmetros da câmera. Sendo $\mathbf{K}$ a matriz dos parâmetros intrínsecos e $\overline{\mathbf{g}}$ a matriz de parâmetros extrínsecos na forma homogênea.

2.3 Visão estéreo

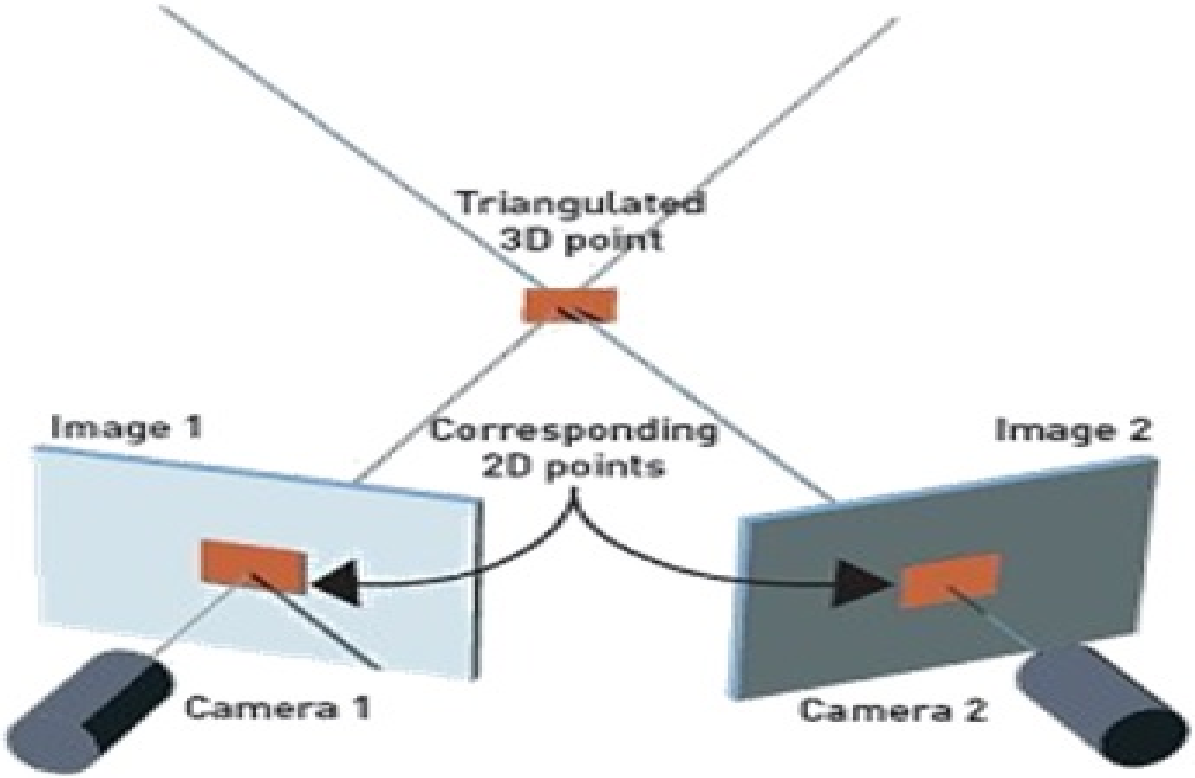
Ao longo da última década, a tecnologia de sensoriamento 3-D baseado em visão tem sido cada vez mais aplicada nas indústrias. A forma 3-D de um objeto, pode ser representado utilizando uma nuvem de pontos, é geralmente necessária para satisfazer dois objetivos principais: engenharia reversa ou controle dimensional. Por outro lado,

técnicas de sensoriamento 3-D baseadas na visão podem ser divididas em duas categorias: visão estéreo passiva e visão estéreo (VE) ativa (TORREAO, 2011).

A VE baseada sem nenhum dispositivo adicional além das câmeras é conhecida como visão estéreo passiva (Fig. 18), que funciona de uma forma semelhante aos olhos humanos. Nesse caso, a VE pode ser muito compacta e de baixo custo, sem nenhum componente adicional. A extensa aplicação da visão passiva beneficia a geometria epipolar, que foi introduzida pela primeira vez pelo britânico Longuet-Higgins em 1981 (YI, 2004). Essa geometria fornece as restrições geométricas entre os pontos da imagem 2D das duas câmeras, em relação aos mesmos pontos em 3-D, com a suposição de que as câmeras são apresentadas utilizando o modelo *pinhole* e calibração da câmera.

No entanto, a VE ainda apresenta alguns inconvenientes para inspeção industrial. A primeira dificuldade é o problema de correspondência. Outro problema é a resolução escassa da reconstrução, geralmente com um pequeno número de pontos. Além disso, a luz ambiente inadequada também iria levar a VE ao fracasso (TORREAO, 2011), (ZHANG, 2013).

Figura 18 – Configuração da visão estéreo



FONTE: (VISIONRT, 2009).

Entretanto, a partir da Fig. 18, considerem-se duas imagens capturadas da mesma cena, entre dois pontos de vista diferentes, em que existe um ponto 3-D, que é projetado,

num ponto 2D nas duas imagens. O ponto 3-D é a posição reconstruída a partir das projeções das imagens (GOESELE, 2006).

Por outro lado, para representar o sistema de VE, é utilizado o movimento do objeto no espaço euclidiano, além disso, é considerado um ponto P no espaço tridimensional, onde X_2 é o ponto da câmera 2, que é o sistema de coordenadas global, enquanto o ponto X_1 da câmera 1 está em relação à câmera 2. Desse modo, o ponto X_1 da câmera 1 está em relação ao ponto X_2 da câmera 2, sendo $X_1 \in \mathbb{R}^3$ e $X_2 \in \mathbb{R}^3$. Tudo que foi explicado está definido na seguinte relação:

$$X_2 = RX_1 + T \tag{32}$$

Agora assume-se que $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{N}^3$, são as coordenadas homogêneas da projeção do mesmo ponto p mas no plano entre as duas imagens, desde que $X_i = \lambda_i \boldsymbol{x}_i, \forall i = 1, 2$. Portanto, a equação pode ser escrita em termos de coordenadas da imagem $\boldsymbol{x}_i$ e profundidades λ_i, conforme é descrito a seguir:

$$\lambda_2 \boldsymbol{x}_2 = R\lambda_1 \boldsymbol{x}_1 + T \tag{33}$$

Para remover a profundidade λ_i da equação acima, multiplicam-se ambos os lados pelo operador chapéu $\hat{T}$:

$$\lambda_2 \hat{T}\boldsymbol{x}_2 = \hat{T}R\lambda_1 \boldsymbol{x}_1 \tag{34}$$

Uma vez que o ***operador chapéu*** do vetor de translação $T \in \mathbb{R}^3$ é definido como $\hat{T}$ que é uma matriz $\in \mathbb{R}^{3\times3}$ semisssimétrica, com determinante igual a um, esse está associado ao vetor $T \in \mathbb{R}^3$, representado por:

$$\hat{T} = \begin{bmatrix} 0 & -T_3 & T_2 \\ T_3 & 0 & -T_1 \\ -T_2 & T_1 & 0 \end{bmatrix} \tag{35}$$

Esse operador será usado ao longo do projeto, uma vez que, em diferentes bibliografias, está associado como o operador chapéu $\hat{u} \in \mathbb{R}^{3\times3}$ do vetor $\vec{u} \in \mathbb{R}^3$ (YI, 2004).

A partir disso, o vetor $\widehat{T}\boldsymbol{x}_2 = T \times \boldsymbol{x}_2$ é perpendicular ao vetor $\boldsymbol{x}_2$, e o produto interno $\langle \boldsymbol{x}_2, \widehat{T}\boldsymbol{x}_2 \rangle = \boldsymbol{x}_2^T \widehat{T}\boldsymbol{x}_2$ é zero. Multiplicando a equação (34) por $\boldsymbol{x}_2^T$ pela esquerda, obtêm-se a relação:

$$\boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 = 0$$

Essa relação é conhecida como “Restrição epipolar”.

2.3.1 Restrição epipolar

Considere duas imagens $\boldsymbol{x}_1, \boldsymbol{x}_2$ do mesmo ponto p, com diferentes poses relativas das câmeras. Além disso, uma das imagens está em relação à outra, pelos movimentos (R, T), onde $R \in \mathbf{SO(3)}$ é a matriz de rotação e $T \in \mathbb{R}^3$ é o vetor de translação. Por conseguinte, a restrição epipolar é dada pela seguinte equação (36)

$$\boldsymbol{x}_2{}^T \hat{T} R \boldsymbol{x}_1 = 0 \tag{36}$$

A restrição epipolar é essencial na correlação estéreo, sendo também conhecida como *restrição essencial*. Portanto, a partir da equação (36) é formada a matriz essencial E.

$$E = \hat{T} R \in \mathbb{R}^{3\times 3} \tag{37}$$

2.3.2 Entidades geométricas epipolares

A geometria epipolar, vista na Fig. 19, entre as imagens, tem o propósito de fornecer um restrição epipolar muito forte para encontrar correlações estéreo (ZHANG, 2013).

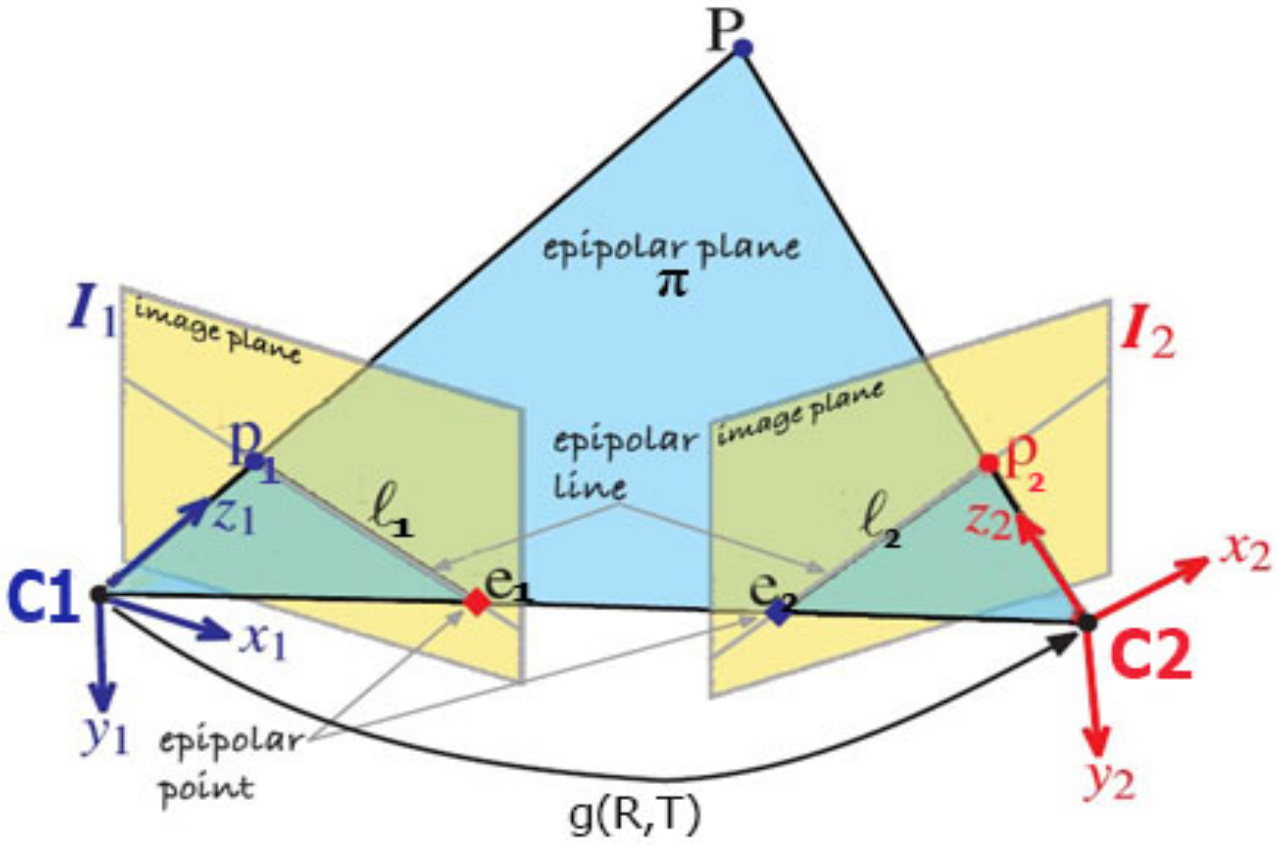
Desta forma, a partir dessa Fig. 19, a geometria epipolar está composta por dois pontos de vista diferentes associados a duas câmeras, $C1$ e $C2$, e um ponto P em 3-D. Tudo isto define o plano epipolar Π. Da intersecção da projeção dos pontos em 2D até a base do plano epipolar Π, surgem as linhas epipolares l_1 e l_2. Os *epipolos* e_1 e e_2 são as projeções do plano da imagem, passando pela las linhas epipolares até a base do plano epipolar Π. Entretanto, p_1 e p_2 são a projeção do ponto P em 3-D até o plano da imagem em 2D (DEFARIAS, 2012).

Assim, a seguir, são apresentados os conceitos básicos para cada entidade geométrica epipolar.

1. *Plano epipolar (o_1, o_2, P):* é determinado entre os dois centros de projeções o_1, o_2 e o ponto P, conhecido como plano epipolar associado com as configurações das câmeras e o ponto P. Existe um único plano epipolar para cada ponto (YI, 2004), (WEXLER, 2003). O plano epipolar da Fig. 19 está determinado pelo triângulo de cor azul, chamado de *epipolar plane*.

2. *Epipolos:* a projeção e_1 ou e_2 do centro de uma câmera, até a projeção do ponto da imagem é conhecido como *epipolo*. Esta projeção poderia acontecer fora dos limites físicos dos sensores da imagem (YI, 2004), (WEXLER, 2003). Os epipolos podem ser mostrados na Fig. 19, como *epipolar point*.

Figura 19 – Geometria epipolar e entidades geométricas epipolares



FONTE: (CORKE, 2011).

3. *Linhas epipolares:* a interseção da projeção do ponto p da imagem, até os epipolos, é conhecida como linhas epipolares. A linha l_2 é a linha epipolar associada com o ponto p_2, e passa através do ponto e_2, em que a linha base entre os centros óticos das câmeras são $C1$ e $C2$ (YI, 2004), (WEXLER, 2003). As linhas epipolares $\boldsymbol{l}_1$ ou $\boldsymbol{l}_2$ da Fig. 19, são apresentadas como *epipolar line*.

2.3.3 Propriedades dos epipolos e linhas epipolares

A partir das definições anteriores, a seguir são apresentadas as propriedades dos epipolos, linhas epipolares e pontos projetados das imagens.

Assim, a partir de uma matriz essencial $E = \hat{T}R$ que define a relação entre as duas imagens $\boldsymbol{x}_1, \boldsymbol{x}_2$.

1. *Epipolos:* os epipolos $e_1, e_2 \in \mathbb{R}^3$, devem satisfazer as seguintes condições.

$$e_2^T E = 0 \tag{38}$$

$$Ee_1 = 0 \tag{39}$$

Onde: $e_2 = T$ e $e_1 = R^T T$

2. *Linhas epipolares:* as linhas epipolares ou co-imagens, $l_1, l_2 \in \mathbb{R}^3$, estão associadas às projeções das imagens $\boldsymbol{x}_1, \boldsymbol{x}_2$, apresentados a seguir:

$$l_2 \sim {}^4 E\boldsymbol{x}_1, l_1 \sim E^T \boldsymbol{x}_2 \in \mathbb{R}^3 \tag{40}$$

Onde l_1, l_2 são os vetores normais do plano epipolar expressado em relação aos dois sistemas de coordenadas das câmeras.

3. *Epipolos e linhas epipolares:* as equações entre linhas epipolares e epipolos estão dadas a seguir:

$$l_i^T e_i = 0, l_i^T x_i = 0, para : i = 1, 2. \tag{41}$$

2.4 Processamento de imagens

Processamento de imagens é qualquer forma de processamento de dados no qual a entrada e saída são imagens, tais como fotografias ou quadros de vídeo. Ao contrário do tratamento de imagens, que se preocupa somente com a manipulação de figuras para sua representação final, o processamento de imagens é um estágio para novos processamentos de dados, como, por exemplo, aprendizagem de máquina ou reconhecimento de padrões. A maioria das técnicas envolve o tratamento da imagem como um sinal bidimensional, no qual são aplicados padrões de processamento de sinal (NIXON, 2002).

Existem técnicas para realçar ou suprimir seletivamente as informações contidas em uma imagem em diferentes escalas espaciais, para destacar alguns elementos da imagem, ou mesmo para ocultar valores discrepantes. Uma dessas técnicas é a filtragem de imagens, na qual ocorrem transformações da imagem "pixel" a "pixel", que dependem do nível de cinza de um determinado "pixel" e dos seus "pixels" vizinhos da imagem (NIXON, 2002).

2.4.1 Filtro de convolução

O processo de filtragem envolve utilizar filtros de convolução, que são usados para criar efeitos em imagens, como, por exemplo, borramento e aguçamento. Os filtros de convolução podem ser calculados de forma espacial ou no domínio da frequência. Filtros

[4] Sendo que $\sim$ indica igualdade até um fator escalar.

no domínio espacial têm a vantagem de não requerirem nenhuma transformação e operam diretamente sobre os pixels da imagem. Os filtros espaciais serão abordados neste trabalho. Para facilitar a compreensão de filtros de convolução espaciais, primeiro lida-se com a aplicação deles em imagens em escala de cinza. O cálculo de filtros de convolução em imagens é feito usando uma máscara ou kernel, que é multiplicado pelos valores de intensidade da imagem. A aplicação dessa máscara na imagem atual gera uma imagem nova de mesmas dimensões (SARRíA, 2005,2006)

A operação do filtro de convolução pode ser matematicamente representada a seguir:

$$N(x,y) = \sum_{j=-1}^{1} \sum_{k=-1}^{1} K(j,k)p(x-j, y-k) \tag{42}$$

Alguns detalhes sobre o cálculo:

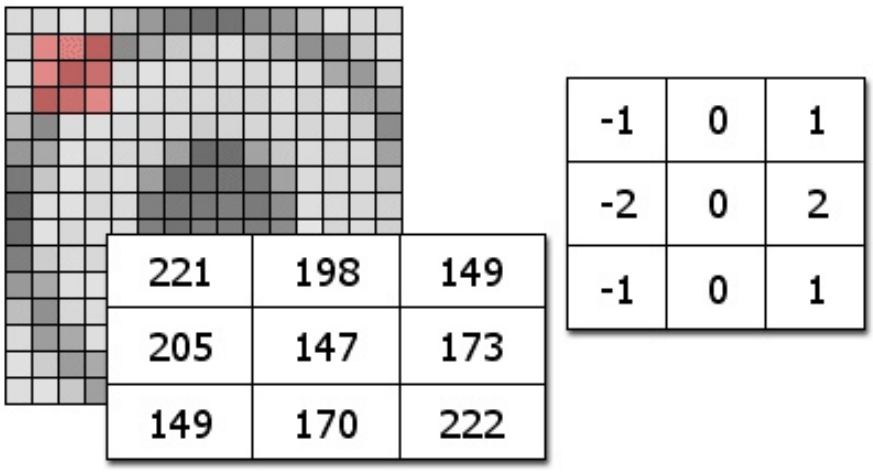
a) a máscara tem tamanho $N \times N$, com N um número ímpar;

b) a máscara é colocada sob uma determinada região da imagem. Dessa forma, a máscara encobre uma região da imagem, sendo que cada valor da máscara encobre exatamente um pixel da imagem;

c) o pixel encoberto pela região central é aquele que está sendo alterado.

Segue um exemplo da alteração de um pixel da imagem (Fig. 20) através de convolução espacial:

Figura 20 – Exemplo do filtro de convolução



FONTE: (MATTHEWS, 2002).

Aqui, o pixel com valor 147 é alterado. O cálculo do novo valor é:

$221 \cdot (-1) + 198 \cdot (0) + 149 \cdot (1) + 205 \cdot (-2) + 147 \cdot (0) + 173 \cdot (2) + 149 \cdot (-1) = 170 \cdot (0) + 222 \cdot (1) = -63$

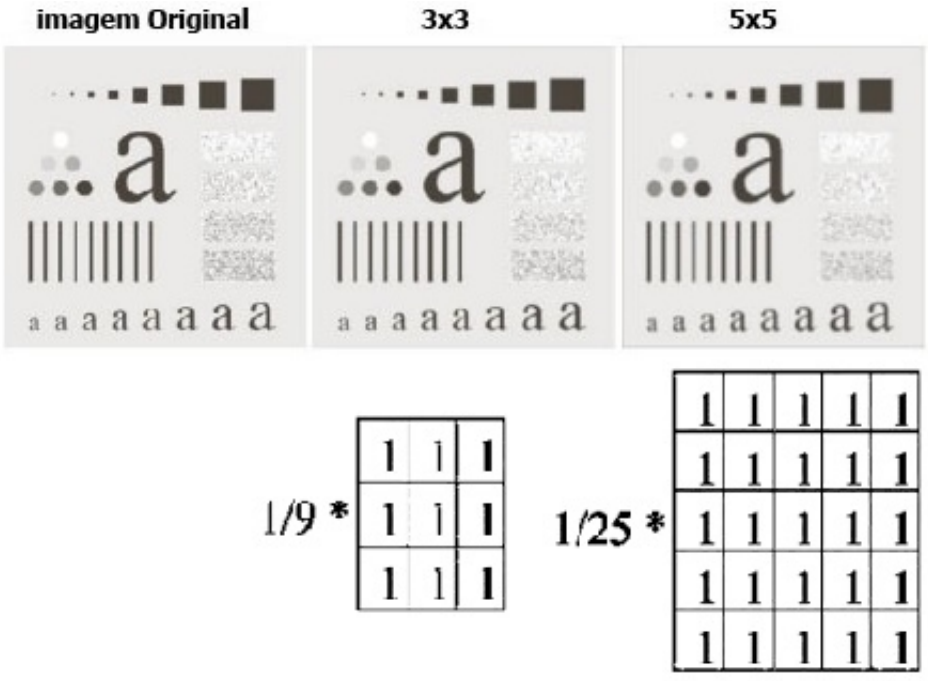
Nesse caso, como o valor de imagens em escala de cinza está no alcance de [0, 255], o resultado -63 é fixado ao valor mais próximo representável em escala de cinza, isto é, zero. Se fosse maior do que zero, o valor seria aquele que foi obtido.

2.4.2 Filtro da Média

O filtro da média é o filtro mais simples e fácil de implementar para suavizar imagens, ou seja, reduzir a quantidade de variações de intensidade entre os pixels vizinhos (JURADO, 2012,2013).

O filtro funciona com cada pixel da imagem que se visita e passa a ter a média de pixels vizinhos. Ele pode operar com uma máscara de convolução particular. O resultado dessa aplicação de filtros é atribuído ao núcleo do valor de pixel filtrada correspondente ao valor da média aritmética de cinza determinado de $N \times N$ pixels da imagem correspondente submatriz (a máscara para 8 vizinhos tem um filtro de 3×3; para 24 vizinhos, o filtro é de 5×5) (PERTUSA, 2011). O resultado da aplicação do filtro é dado na Fig. 21:

Figura 21 – Resultado do uso de máscaras com o filtro da média (figura acima). Máscaras do filtro da média 3×3 e 5×5 (figura abaixo)



FONTE: (THOME, 2010), (DOUGHERTY, 1994).

Nesse projeto, o filtro de média com uma máscara de 3×3 será utilizado, ajudando a melhorar o ruído proveniente das imagens capturadas.

2.4.3 Esforço computacional

O esforço computacional tem a ver com a quantidade de recursos computacionais requeridos para a solução de um problema (HEIDEMAN, 1988). Além disso, fornece uma ideia dos limites práticos do que pode ser feito. O esforço computacional varia de acordo com o problema a se tratar, os recursos físicos disponíveis, a memória disponível, o algoritmo de execução do problema, a complexidade aritmética, entre outros (BRIGGS, 1987).

Se os recursos físicos são fixos, o esforço computacional pode ser aproximado através da complexidade aritmética, dada pelo número de adições, deslocamentos de bits e multiplicações. Ao reduzir a complexidade aritmética, há uma redução no esforço computacional e, provavelmente, no tempo de cômputo.

O esforço computacional no uso do filtro de convolução, e, portanto, do filtro da média, é alto. Sendo que, numa imagem de tamanho $M \times M$ e mascara de $N \times N$, o número de multiplicações é de M^2N^2 (BACKES, 2014).

Um exemplo do esforço computacional é apresentado na tabela a seguir, que mostra a quantidade de multiplicações que se deve realizar para diferentes tamanhos de máscaras, com um tamanho da imagem de 512×512 e o tamanho das máscaras de $N \times N$.

Tabela 1 – Esforço computacional do filtro de convolução

Mascara $N \times N$	$M^2 \cdot N^2$z - Quantidade de multiplicações
3×3	2.359 269
5×5	6.553 600
7×7	12.845 056
16×16	67.108 864

FONTE: (BACKES, 2014).

No caso da tabela acima, se a quantidade de multiplicações realizada no filtro é cada vez mais alta, o esforço computacional é maior, o qual envolve uma maior utilização de recursos físicos, tais como mais memória ou melhoramento dos algoritmos de execução. Uma alternativa para melhorar o esforço computacional é realizar a análise das imagens no domínio da frequência (Fourier), sendo que o custo computacional da transformada de Fourier pode ser menor (BACKES, 2014).

3 Geometria epipolar para um ponto e calibração de câmeras

Neste capítulo, serão aplicados os conceitos explicados no capítulo anterior, concentrando-se em conceitos como a calibração de câmeras, parâmetros intrínsecos, extrínsecos e aplicação da teoria da visão estéreo. Todas essas abordagens serão aplicadas na reconstrução tridimensional a partir do espaço bidimensional para um objeto.

3.1 Preparação do ambiente de experimentação e posição das câmeras

O ambiente de experimentação visto na Fig. 22, foi realizado sob a forma de um cubo com duas paredes.

Figura 22 – Ambiente de experimentação do projeto

FONTE: A autora.

Todo esse ambiente está forrado com fundo preto sem brilho. Nos outros dois lados do ambiente onde não há paredes, são incluídos dois suportes, os quais são utilizados para colocar as duas câmeras. Essas duas câmeras são parte do sistema de visão estéreo, que servem para obter o movimento dos objetos.

Ao posicionar as duas câmeras no ambiente de experimentação, deve-se verificar os limites da posição das câmeras. Esses limites são definidos de acordo com o campo de visão dos objetos, tendo em conta a velocidade e movimento, ou seja, as duas câmeras devem alcançar a visão dos objetos em todos os intervalos de tempo no movimento (esse movimento é dado pelo gimbal, que será explicado no capítulo 5).

Em relação à posição das câmeras, estas estão de acordo com a orientação do sistema de coordenadas do ambiente. Deste modo, a rotação de cada câmera $\{C_1\}$, $\{C_2\}$, está em relação ao sistema de coordenadas do ambiente $\{\mathrm{W}\}$. Os ângulos aproximados utilizados no ambiente para cada câmera são:

- Câmera 1: $z = 90°$, $x = 95°$, $y = 10°$, nessa ordem de rotação.
- Câmera 2: $z = 130°$, $x = 100°$, $y = 10°$, nessa ordem de rotação.

Deve notar-se que essa posição das câmeras terá uma pequena variação, de acordo com a necessidade do campo de visão que seja necessário para calibrar as câmeras. Esse processo de calibragem de câmeras será explicado no tópico seguinte. Outro ponto a considerar é que, quando são posicionadas as câmeras numa determinada posição, elas devem ficar nessa posição até o fim de todo o processo de teste, pois, se existe uma diferença na posição, os testes da reconstrução dos objetos estarão errados.

3.2 Calibração de câmeras

A calibração das câmeras é um passo muito importante no projeto, pois fornece os parâmetros intrínsecos (foco de f, matriz K) e extrínsecos (rotação e translação). A calibração foi feita sob um sistema de visão estéreo, com duas câmeras (Microsoft Vx 800). As atividades para calibrar as câmeras no projeto foram: a delimitação do ambiente, a criação de um tabuleiro em forma de xadrez, a posição do tabuleiro no ambiente (Fig. 23), o uso do software de calibração e a obtenção de parâmetros intrínsecos e extrínsecos.

Toda a calibração serve para conhecer as posições exatas no ambiente.

Figura 23 – Inicio de calibração das câmeras no projeto.

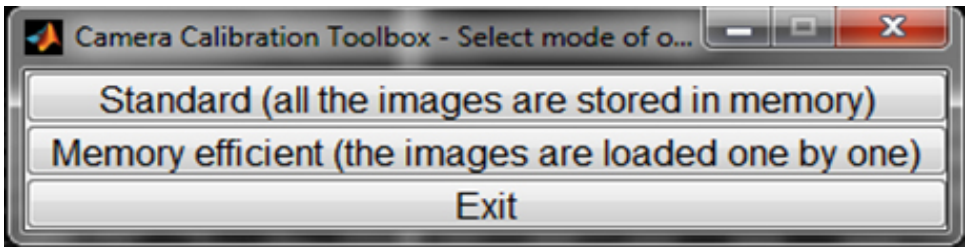
FONTE: A autora.

Para calibrar as câmeras no ambiente de VE., foi utilizado o *"Tolbox "* de calibração de câmeras para Matlab, desenvolvido pela Universidade Caltech (BOUGUET, 2013).

A seguir, os passos para calibrar as câmeras:

1. *Inicio da calibração:* para iniciar a calibração das câmeras, deve-se entrar na janela principal da Fig. 24 e escolher a opção *"Standard"*.

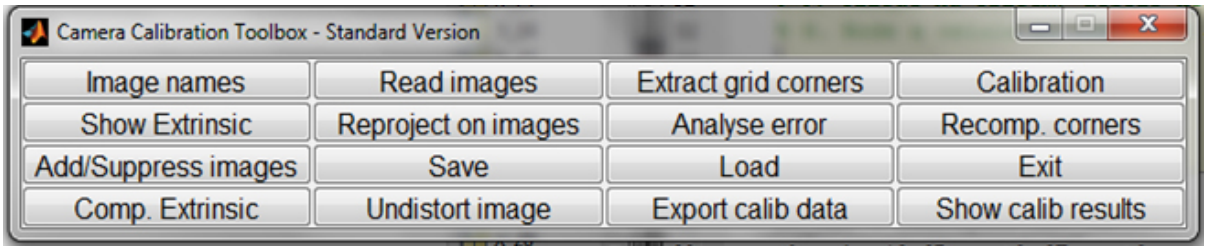
Figura 24 – Janela do inicio de calibração



FONTE: A autora.

2. *Obtenção dos parâmetros intrínsecos:* para obter os parâmetros intrínsecos, deve-se escolher a opção *"calibration"* da Fig. 25.

Figura 25 – Início de calibração e parâmetros intrínsecos



FONTE: A autora.

Depois, o software dá os parâmetros intrínsecos das duas câmeras, tais como: matriz K (equação (31)), lente focal f da câmera, , centro ótico, coeficientes de inclinação entre pixels X, Y e matriz distorção da imagem, os quais serão apresentados a seguir.

Parâmetros intrínsecos da câmera 1

- Lente focal f

$$f_1 = \begin{bmatrix} 669.73921 \\ 669.73921 \end{bmatrix}$$

- Fator inclinado s_θ

$$s_\theta = 0$$

- Origem da imagem o_x e o_y

$$\begin{bmatrix} o_x \\ o_y \end{bmatrix} = \begin{bmatrix} 333.4842 \\ 233.4144 \end{bmatrix}$$

- Matriz K_1

$$K_1 = \begin{bmatrix} 684.6224 & 0 & 333.4842 \\ 0 & 686.8438 & 233.4144 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

Parâmetros intrínsecos da câmera 2

- Lente focal f

$$f_2 = \begin{bmatrix} 684.62241 \\ 686.84379 \end{bmatrix}$$

- Fator inclinado s_θ

$$s_\theta = 0$$

- Origem da imagem o_x e o_y

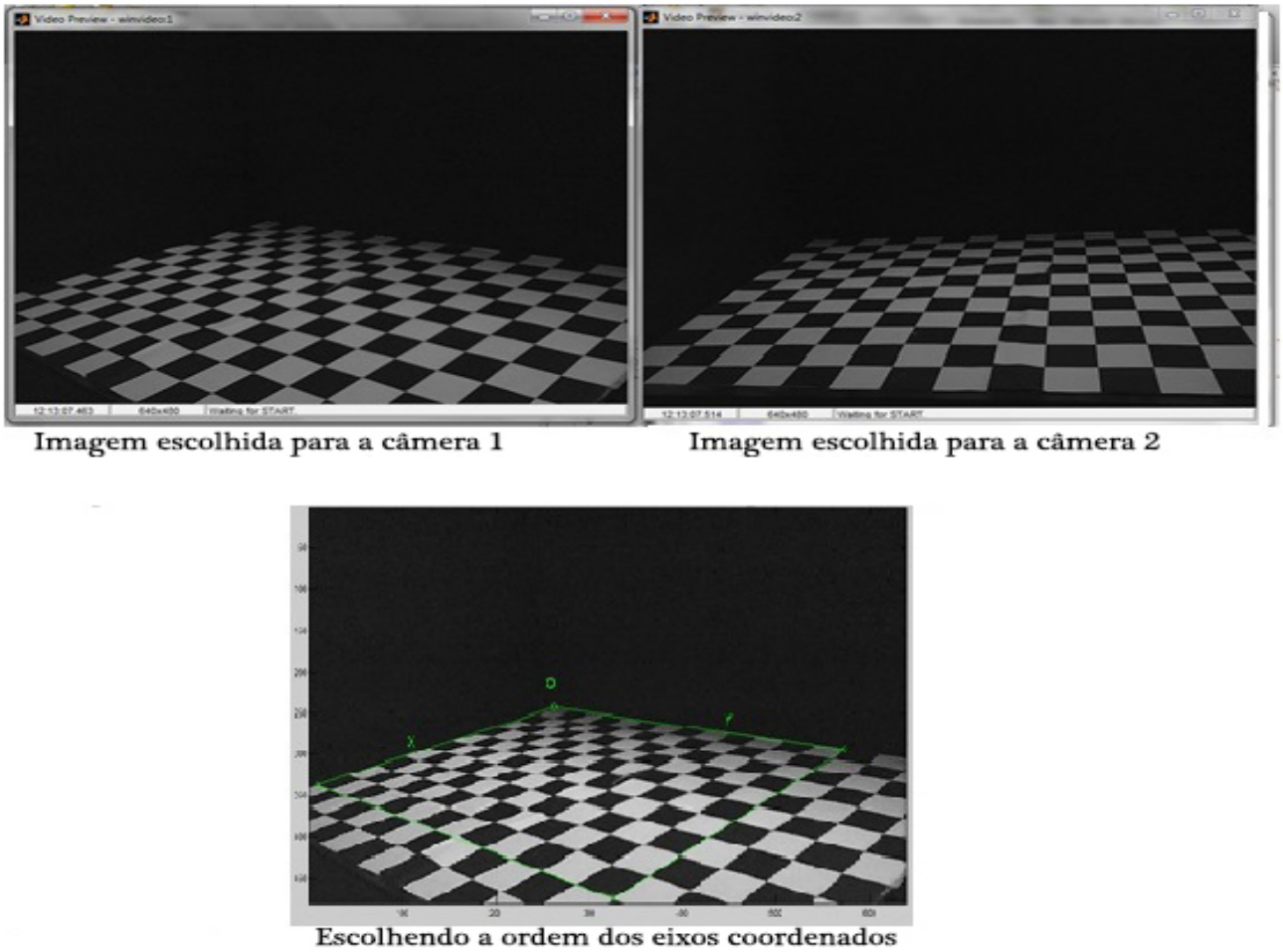
$$\begin{bmatrix} o_x \\ o_y \end{bmatrix} = \begin{bmatrix} 325.6238 \\ 240.6365 \end{bmatrix}$$

- Matriz K_2

$$K_2 = \begin{bmatrix} 689.3351 & 0 & 325.6238 \\ 0 & 690.4951 & 240.6365 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

3. *Escolhendo as imagens de calibração e a ordem dos eixos:* nesse passo, as duas imagens são obtidas. Deve-se escolher a ordem dos eixos de coordenadas no ambiente, na seguinte ordem: "Y - YX - X". Depois, deve-se colocar o número de quadrados que serão consideradas para cada um dos eixos de coordenadas, em milímetros. Tudo isso pode ser visto na Fig. 26.

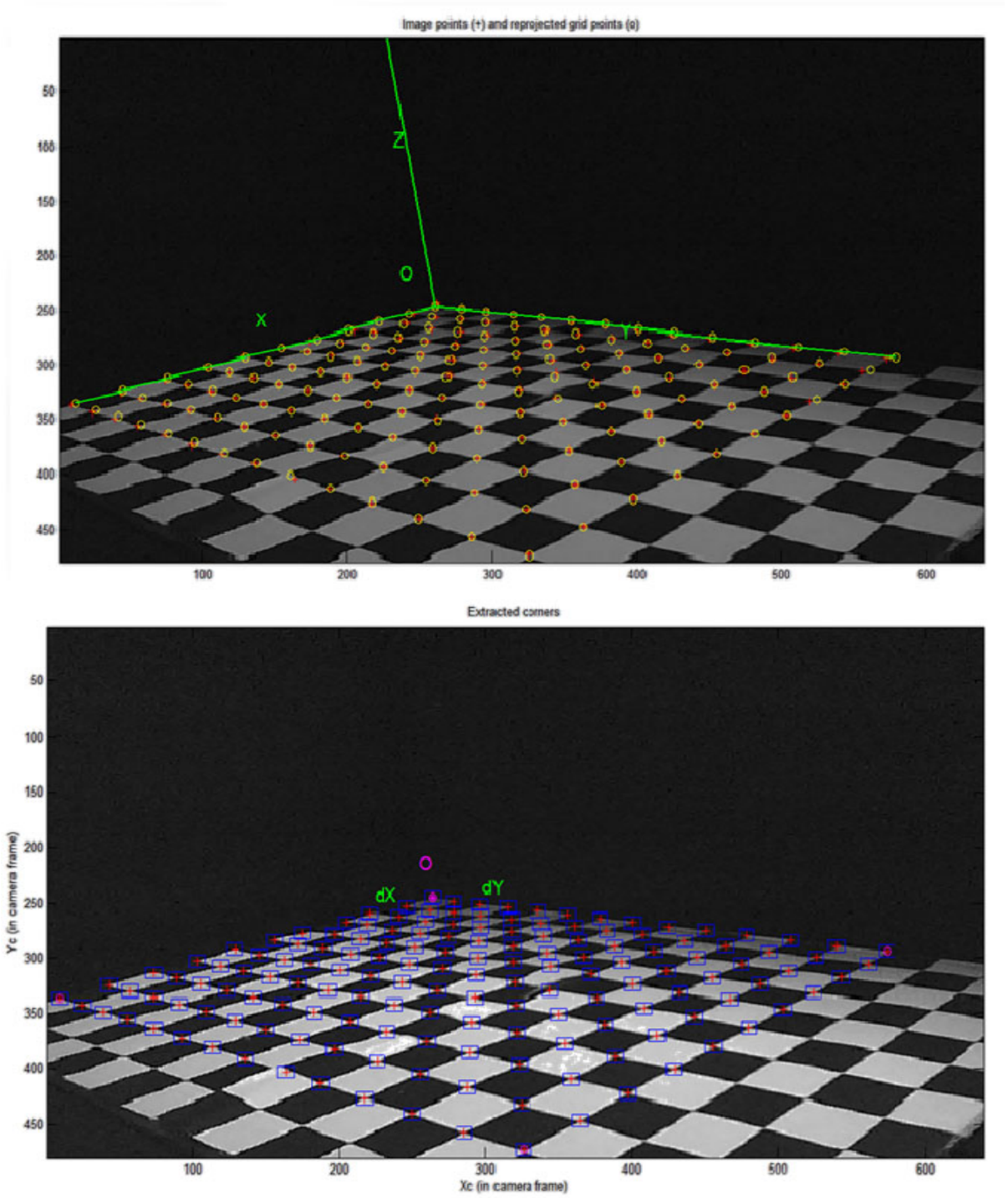
Figura 26 – Escolha de imagens e a ordem dos eixos de coordenadas



FONTE: A autora.

4. *Resultados do sistema de coordenadas e pontos calibrados:* a partir da seleção de dados do item anterior, o software projeta o sistema de coordenadas extrai os pontos de cada canto das figuras e as derivadas de cada ponto (Fig. 27).

Figura 27 – Re-projeção dos pontos e extração de cantos



FONTE: A autora.

5. *Parâmetros extrínsecos :* depois de ter câmeras calibradas e os resultados obtidos em termos dos parâmetros extrínsecos (como explicado no capítulo anterior), obtêm-se a matriz de rotação e o vetor de translação de ambas câmeras, denotado como matriz $g(R, T)$. Todos esses dados ficam em relação ao sistema de coordenadas do mundo, e os resultados serão apresentados a seguir.

   Parâmetros extrínsecos da câmera 1

   - Matriz g_1

$$g_1 = \begin{bmatrix} -0.7137 & 0.1917 & -0.6737 & 512.6399 \\ 0.6976 & 0.1071 & -0.7085 & 612.1398 \\ -0.0636 & -0.9756 & -0.2101 & 200.0854 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

   Parâmetros extrínsecos da câmera 2

   - Matriz g_2

$$g_2 = \begin{bmatrix} -0.1230 & 0.1853 & -0.9750 & 665.0527 \\ 0.9905 & -0.0372 & -0.1320 & 286.4502 \\ -0.0607 & -0.9820 & -0.1790 & 192.3744 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

3.3 Reconstruir o espaço tridimensional para um ponto

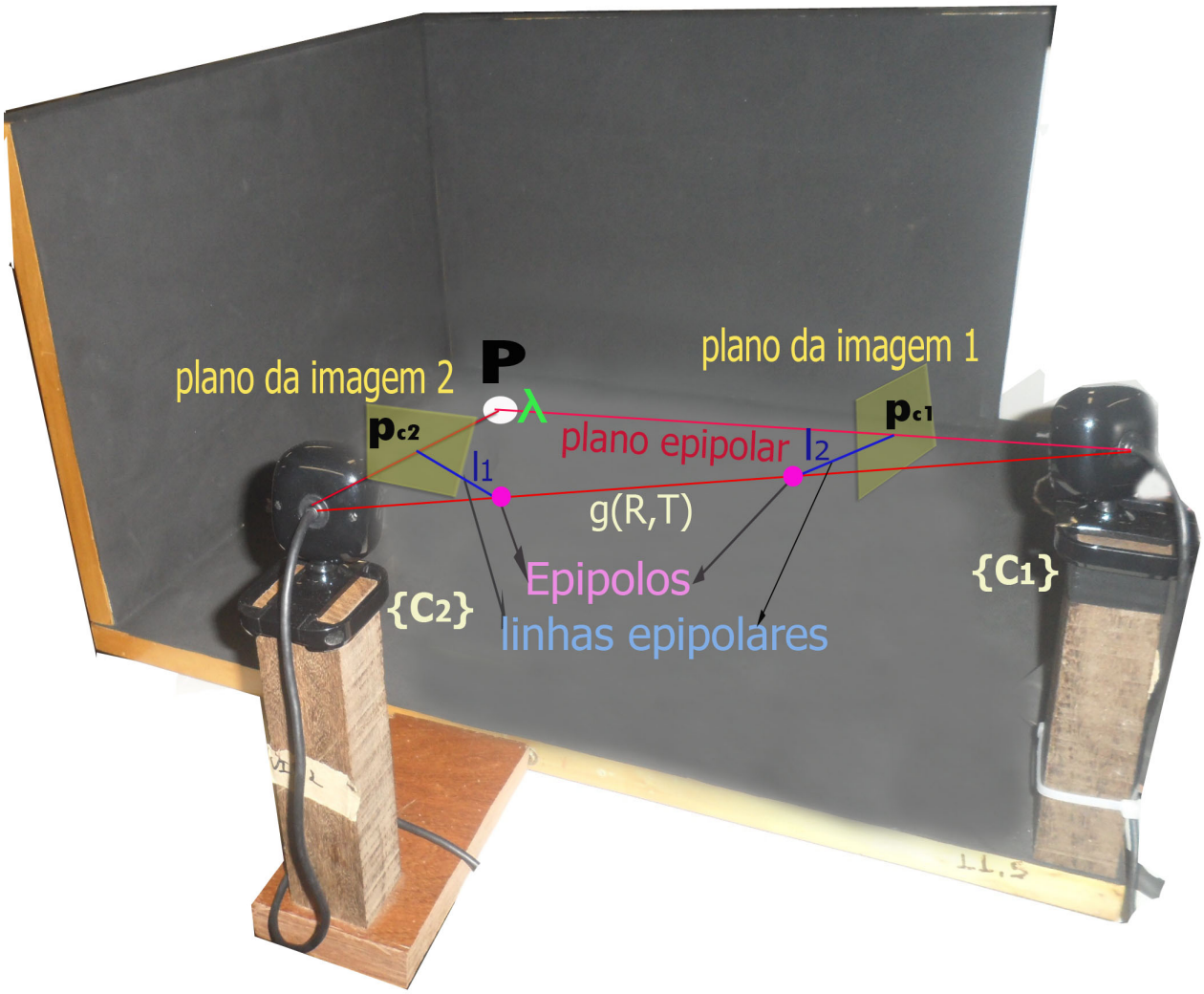
Para reconstruir o espaço tridimensional de uma imagem a partir da VE, é iniciado o caso mais básico, que é obter a posição tridimensional de um único ponto. Para a posição tridimensional de um ponto, deve-se obter um modelo matemático. Depois, esse modelo será generalizado para obter a posição tridimensional de vários pontos.

O ambiente de experimentação utilizado é apresentado na Fig. 28, que serve para reconstruir o espaço tridimensional para um ponto. O processo de reconstrução inicia-se com a configuração do ambiente, em que existem um sistema de coordenadas global {W}, dois sistemas de coordenadas para cada câmera e a configuração da câmera 1 que fica em relação à câmera 2.

Entretanto, relaciona-se a equação (32) do sistema de VE, do capítulo anterior, onde existe um ponto da câmera 1 representado por $P_{c_1} \in \mathbb{R}^3$ e um ponto da câmera 2, representado por $P_{c_2} \in \mathbb{R}^3$. No entanto, para representar a relação entre os dois pontos, diz-se que o ponto da câmera 1 fica em relação ao ponto da câmera 2, apresentado a seguir:

$$P_{c_2} = RP_{c_1} + T$$

Figura 28 – Reconstrução do espaço tridimensional para um ponto



FONTE: A autora.

Levando a equação (33) da representação homogênea em coordenadas de pixels, tem-se a projeção dos pontos no plano das imagens, da seguinte forma:

$$\lambda_1 \boldsymbol{p}_{c_1} = R\lambda_2 \boldsymbol{p}_{c_2} + T \tag{43}$$

Uma vez que:

$$\boldsymbol{p}_i = \lambda_{ci} K_i^{-1} \bar{p}_i' \tag{44}$$

Sendo i = 1,2 o numero de imagens.

Isolando as profundidades das imagens λ_1, λ_2 :

$$\underbrace{\begin{bmatrix}\boldsymbol{p}_{c1} & -R\boldsymbol{p}_{c2}\end{bmatrix}}_{M}\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix} = T \tag{45}$$

Mudando de nomes às variáveis, obtêm-se a matriz M e o vetor λ:

$$M = \begin{bmatrix}\boldsymbol{p}_{c1} & -R\boldsymbol{p}_{c2}\end{bmatrix} \in \mathbb{R}^{3\times 2}$$

$$\lambda = \begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}$$

Então, a equação geral será:

$$M \cdot \lambda = T \tag{46}$$

Para resolver a equação (46), isola-se λ, e multiplica-se por M^T:

$$M^T \cdot M\lambda = M^T \cdot T$$

Multiplicando por $(M^T \cdot M)^{-1}$

$$(M^T \cdot M)^{-1} \cdot \lambda \cdot (M^T \cdot M) = (M^T \cdot M)^{-1} M^T \cdot T$$

Eliminando $(M^T \cdot M) \cdot (M^T \cdot M)^{-1}$

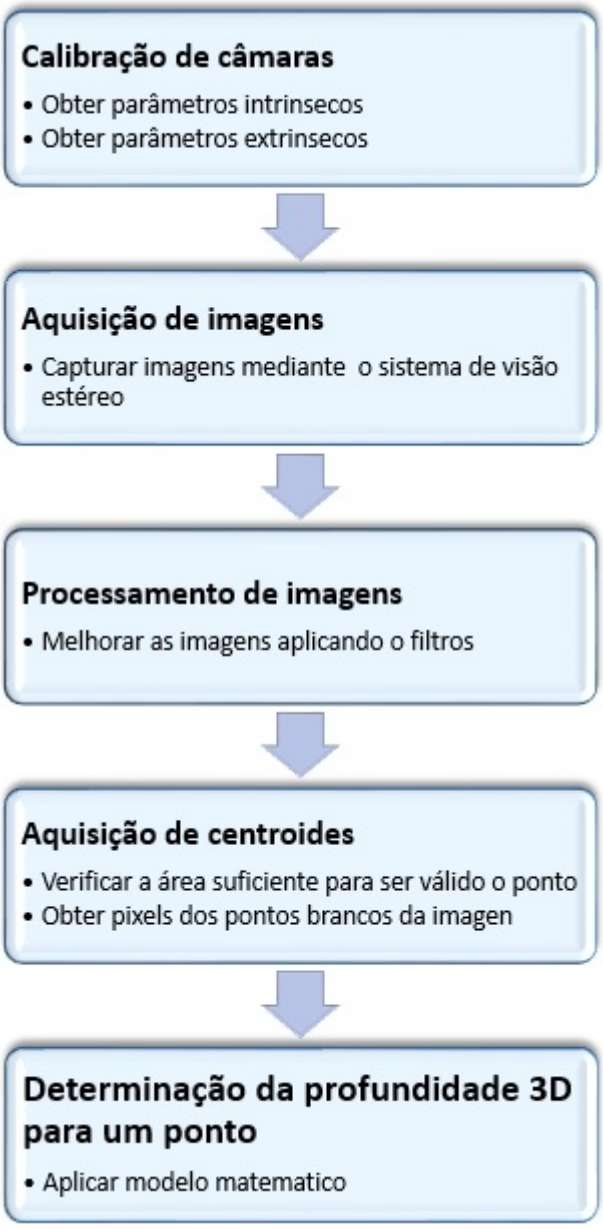
$$\lambda = (M^T \cdot M)^{-1} \cdot M^T \cdot T \tag{47}$$

Portanto, *lambda* λ **é a posição tridimensional para um ponto**, a partir de duas imagens pertencentes a um sistema de VE.

3.4 Processo geral para a obtenção da profundidade de um ponto

Nesta seção, apresenta-se o processo geral dos passos seguidos, para obter a profundidade de um ponto na Fig. 29.

Figura 29 – Processo geral da obtenção da profundidade de um ponto



FONTE: A autora.

Entretanto, a partir da Fig. 29, no primeiro bloco é realizada a calibração das câmeras, obtendo os parâmetros intrínsecos e extrínsecos. No segundo bloco, as imagens das câmeras são capturadas. No terceiro bloco, é realizado o processamento de imagens para serem melhoradas através dos filtros. E, no último bloco, são obtidos os features de cada imagem, como os centroides e as áreas de cada feature. Todos esses blocos resultam na determinação da profundidade de um único objeto, que se aplica ao modelo matemático apresentado na secção anterior.

3.5 Algoritmo de obtenção da profundidade para um ponto

A seguir, no algoritmo 1, serão apresentados os passos mais importantes para a obtenção da profundidade de um ponto.

Algorithm 1 Reconstrução de um único ponto

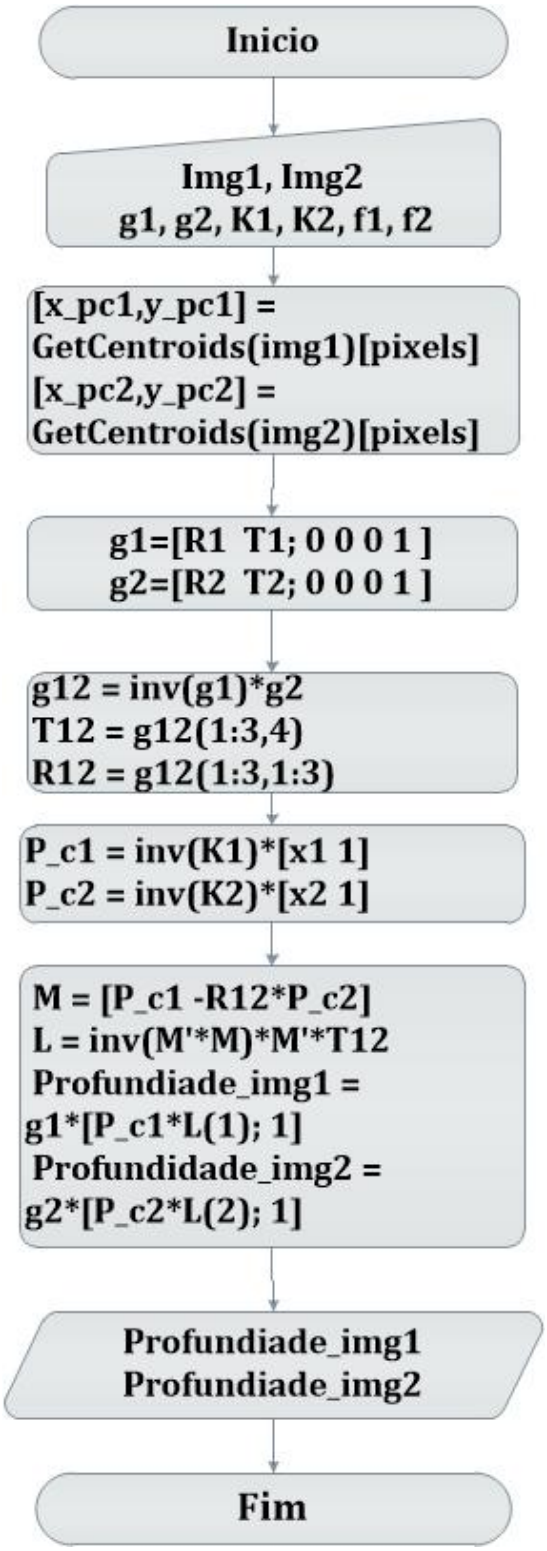
1: **function** PONTOS3D($\bar{p_{c1}}', \bar{p_{c2}}', g_1, g_2, K_1, K_2$)
2: $g_{12} = g_1^{-1} \cdot g_2$
3: $T_{12} = g_{12}(1:3, 4)$
4: $R_{12} = g_{12}(1:3, 1:3)$
5: $M = [K_1^{-1} \cdot \bar{p_{c1}}' \; -R_{12} \cdot K_2^{-1} \cdot \bar{p_{c2}}']$
6: $L = (M^T \cdot M)^{-1} M^T \cdot T_{12}$
7: **return** $p_w = g_1 L_{c1} K_1^{-1} \bar{p_{c1}}'$
8: **end function**

Onde:

- $\bar{p_{c1}}$: são as posições em pixels do centro de um ponto $x, y \in \mathbb{N}^{2\times1}$, da imagem da câmera 1.
- $\bar{p_{c2}}$: são as posições em pixels do centro de um ponto $x, y \in \mathbb{N}^{2\times1}$ da imagem da câmera 2.
- g_1: é a matriz de parâmetros extrínsecos $\in \mathbb{R}^{4\times4}$ na forma homogênea e vem da calibração da câmera 1.
- g_2: é a matriz de parâmetros extrínsecos $\in \mathbb{R}^{4\times4}$ na forma homogênea e vem da calibração da câmera 2.
- K_1: é a matriz de parâmetros intrínsecos $\in \mathbb{R}^{3\times3}$ e vem da calibração da câmera 1.
- K_2: é a matriz de parâmetros intrínsecos $\in \mathbb{R}^{3\times3}$ e vem da calibração da câmera 2.
- g_{12}: é a matriz de parâmetros extrínsecos, calculado entre as duas matrizes g_1 e g_2 $\in \mathbb{R}^{4\times4}$ na forma homogênea.
- T_{12}: é o vetor de translação $\in \mathbb{R}^{3\times1}$, separado da matriz g_{12}.
- R_{12}: é a matriz de rotação $\in \mathbb{R}^{3\times3}$, separada da matriz g_{12}.
- M: é a matriz que foi obtida da equação geral (4.1) do sistema de visão estéreo, que $\in \mathbb{R}^{3\times2}$.
- L: é *lambda* a posição tridimensional obtida para um ponto $\in \mathbb{R}^{3\times1}$ entre duas imagens.

A Fig. 30 apresenta o diagrama de fluxo, para obter a profundidade de um ponto.

Figura 30 – Algoritmo de implementação do modelo matemático para um único ponto



FONTE: A autora.

Da Fig. 30, o algoritmo é iniciado no primeiro bloco com a captura das imagens (img1, img2) e os parâmetros da calibração das câmeras. Os resultados da calibração são os parâmetros extrínsecos: vetores de translação (T_1, T_2) e matrizes de rotação (R_1, R_2) e os parâmetros intrínsecos, tais como os focos de cada câmera (f_1, f_2)). Já no segundo bloco, a imagem é interpretada para que os *features*[1] sejam adquiridos.

No terceiro bloco, as matrizes g_1 e g_2 construídas a partir dos parâmetros extrínsecos das câmeras. No quarto bloco, é formada a matriz g_{12} a partir de g_1 e g_2, formando o vetor de translação T_{12} e a matriz de rotação R_{12}, os quais serão utilizados no resto do algoritmo.

No quinto bloco, é realizada uma correspondência entre o plano da retina da câmera e a matriz de pixels do plano da imagem. As novas coordenadas das imagens são obtidas, usando os parâmetros intrínsecos das câmeras K_1, K_2 na forma homogênea.

No sexto bloco, é obtida a matriz M e lambda, que são as profundidades de cada imagem. Depois, levam-se as profundidades obtidas ao sistema de coordenadas global. E, no último bloco, são enviadas as posições tridimensionais para os algoritmos gráficos apresentados na tela.

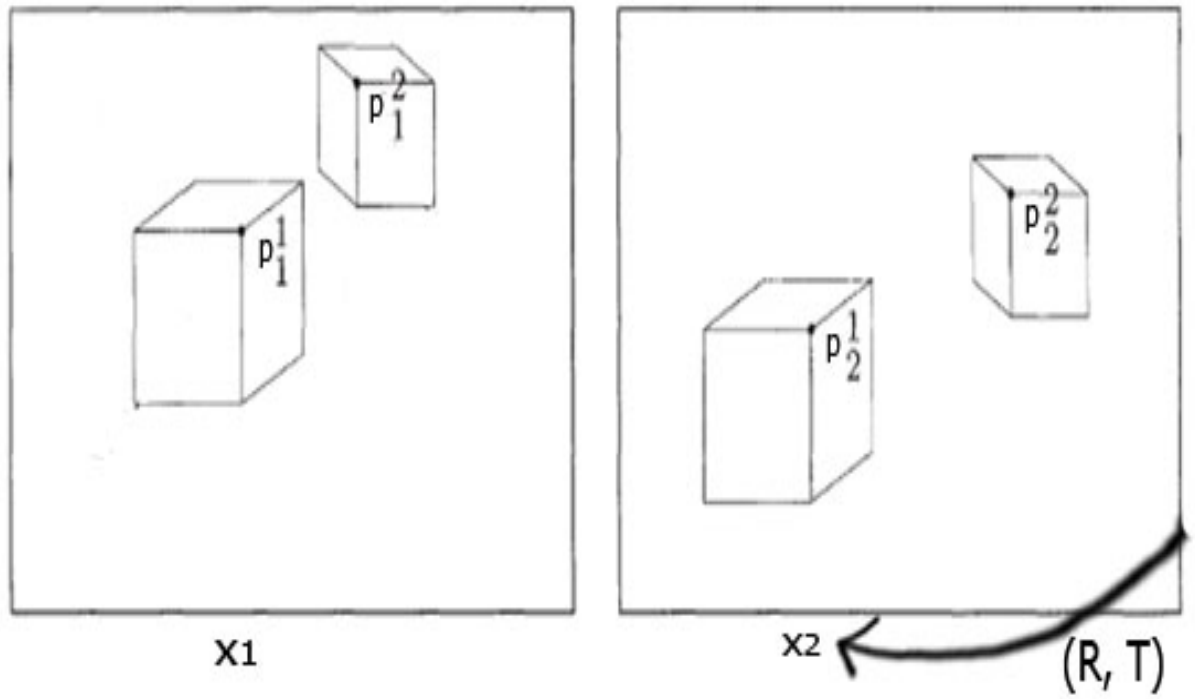
[1] *features*; qualquer interesante característica ou estrutura geométrica extraída, a partir de uma imagem ou uma cena, como pontos, linhas, elipses (ou qualquer outro contorno 2D). No caso do projeto, são os pontos brancos obtidos da imagem

4 Estimação de múltiplos pontos a partir da-geometria epipolar

Até agora, foi levada em conta apenas a reconstrução para um ponto, por meio da VE. Na prática, essa hipótese é bastante restritiva, pois interagir com as cenas do mundo real exige analisar o espaço físico com vários objetos.

Neste capítulo, é generalizado o algoritmo de um ponto para vários pontos, conforme ilustrado na Fig. 31:

Figura 31 – Vários objetos numa cena

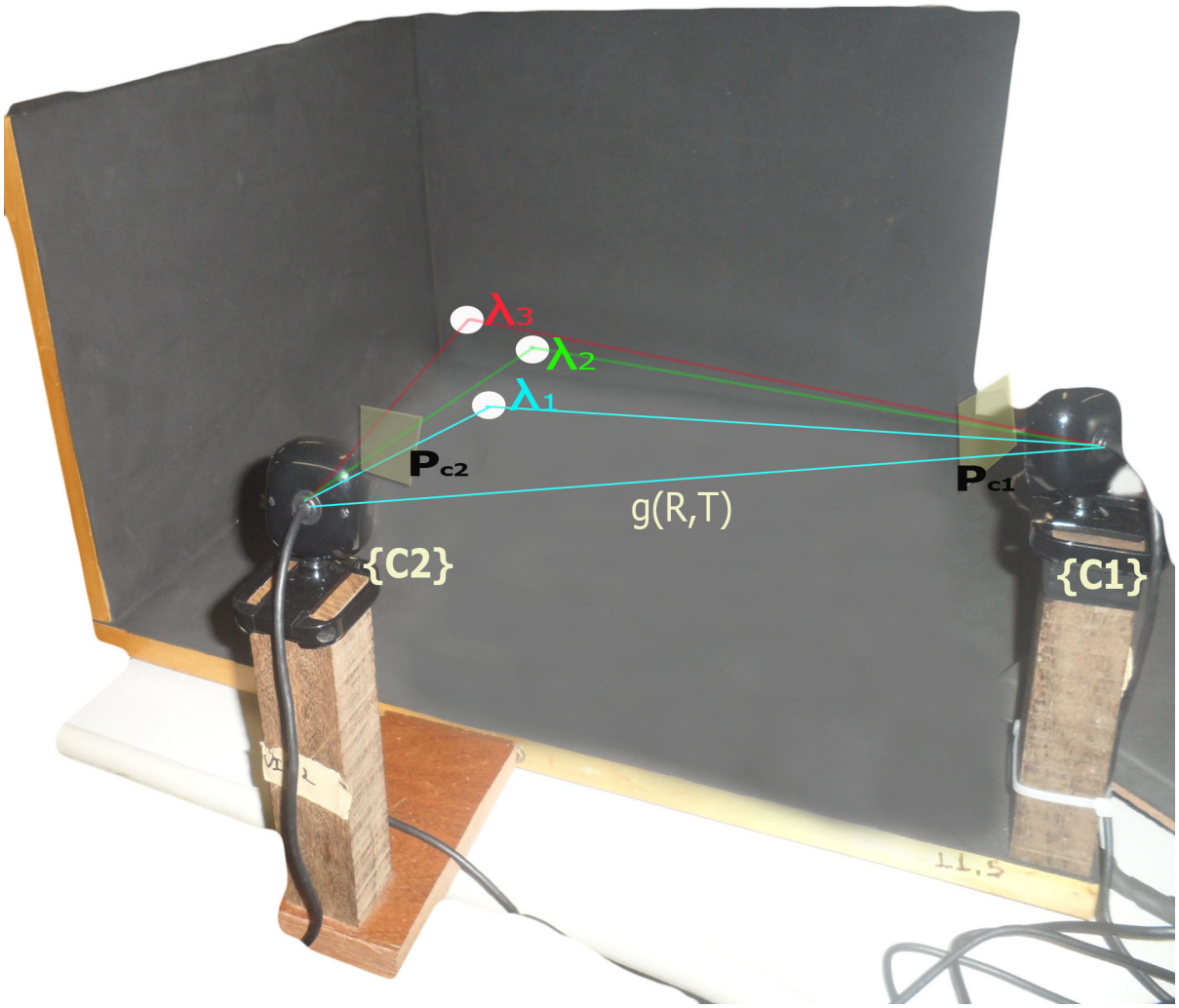


FONTE: Yi.Ma2004

Nessa figura, duas imagens diferentes são apresentadas e nelas são mostrados dois pontos, com dois pontos de vistas. Na Fig. 32 à esquerda, mostra a vista feita pela câmera 1 e a Fig. 32 à direita, mostra a vista feita pela câmera 2. Esse cenário estático está conforme o movimento de translação e rotação dos objetos. Todo esse cenário é visto constantemente no mundo real, o qual será analisado para obter um algoritmo de obtenção tridimensional para vários pontos.

Associando a situação acima com o projeto mostrado na Fig. 32, os três pontos representam a posição de três objetos, em que ficam movimentando-se constantemente. As duas câmeras C_1 e C_2 ficam estáticas capturando os movimentos dos objetos através das imagens x_1 e x_2. A matriz $g(R,T)$ é obtida da triangulação entre as matrizes $g_1(R,T)$ da câmera 1 e $g_2(R,T)$ da câmera 2. Os valores de λ são as profundidades de cada ponto.

Figura 32 – Ambiente do projeto para múltiplos pontos



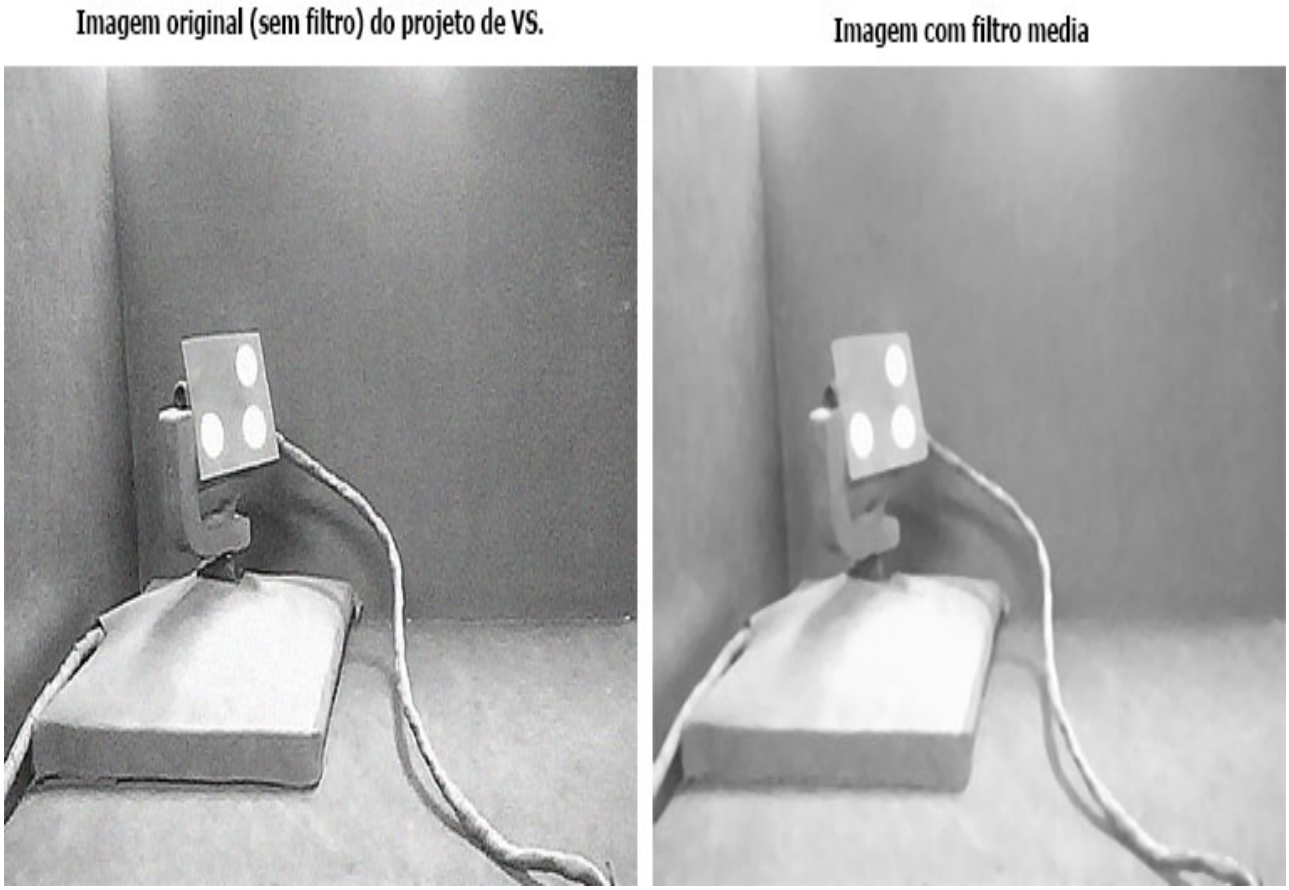
FONTE: A autora.

Para atingir o objetivo deste capítulo, que é obter a profundidade de vários objetos em um ambiente de VE, quatro passos são importantes: melhoramento de imagens, determinação do número de objetos na cena, correlação entre os pontos e, por último, o algoritmo de reconstrução da profundidade de vários objetos.

4.1 Melhoramento de imagens

Ao capturar as imagens no projeto (Fig. 33), percebe-se que estes saíram com algum ruído, causado por imperfeições, por vários motivos, tais como a sujeira na câmera ou no ambiente. Por conseguinte, pensou-se em usar um filtro que pode ajudar a melhorar esses problemas, conhecido como o filtro médio, o qual executa uma operação não linear para reduzir o ruído. Esse filtro ajuda a reduzir o ruído da imagem e preserva, simultaneamente, as bordas.

Figura 33 – Imagens capturadas pelas duas câmeras do ambiente, aplicando o filtro da média.



FONTE: A autora.

A máscara usada no algoritmo é de tamanho $n = 3$, visto da seguinte maneira:

$$1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Para reduzir o trabalho de construção do algoritmo, é aplicado o filtro da média existente no Matlab, conhecido como: ***medfilt2***, da seguinte maneira: *imagem=filt2(imagem)*

4.2 Número de objetos na cena

Contar os pontos brancos em uma imagem com fundo preto pode ser uma tarefa simples, mas, quando se trata de saber quantos pontos existem entre duas imagens diferentes, a tarefa torna-se um pouco mais complexa, pois uma imagem pode ter uma quantidade de pontos diferente a outra imagem. Por conseguinte, neste tópico se normaliza o número de pontos entre as duas imagens.

Os passos para obter o número de objetos entre as duas imagens são:

a) aplicar o filtro da média nas duas imagens;

b) binarizar as imagens com a função do Matlab *im2bw(Img, level)*;

c) encontrar os objetos conectados na imagem binária com a função do Matlab *bwconncomp(binário)*.

d) obter os centroides de acordo ao tamanho das áreas de cada objeto, nas duas imagens com a função Matlab *regionprops(CC, 'Centroid', 'Area')*;

e) obter a quantidade de objetos para as imagens 1 e 2;

f) escolher o menor número de objetos entre as imagens.

Conforme explicado, tudo isso está dado passo a passo no algoritmo 2 e no algoritmo 3.

Algorithm 2 Centroides do objeto

```
1: function GetCentroids(img)
2:     binario = binarizar(img)
3:     props = bwconncomp(bw)
4:     S = regionprops(CC, 'Centroid', 'Area')
5:     centroids = S.Centroid; areas = S.Area ; minArea = 55
6:     if areas < minArea then
7:         Centroid_img= centroids; areas= S.Area
8:     end if
9:     return Centroid_img, areas
10: end function
```

O último passo é escolher o número de objetos a serem considerados entre as duas imagens. Para isso, o algoritmo 3 apresenta os seguintes passos:

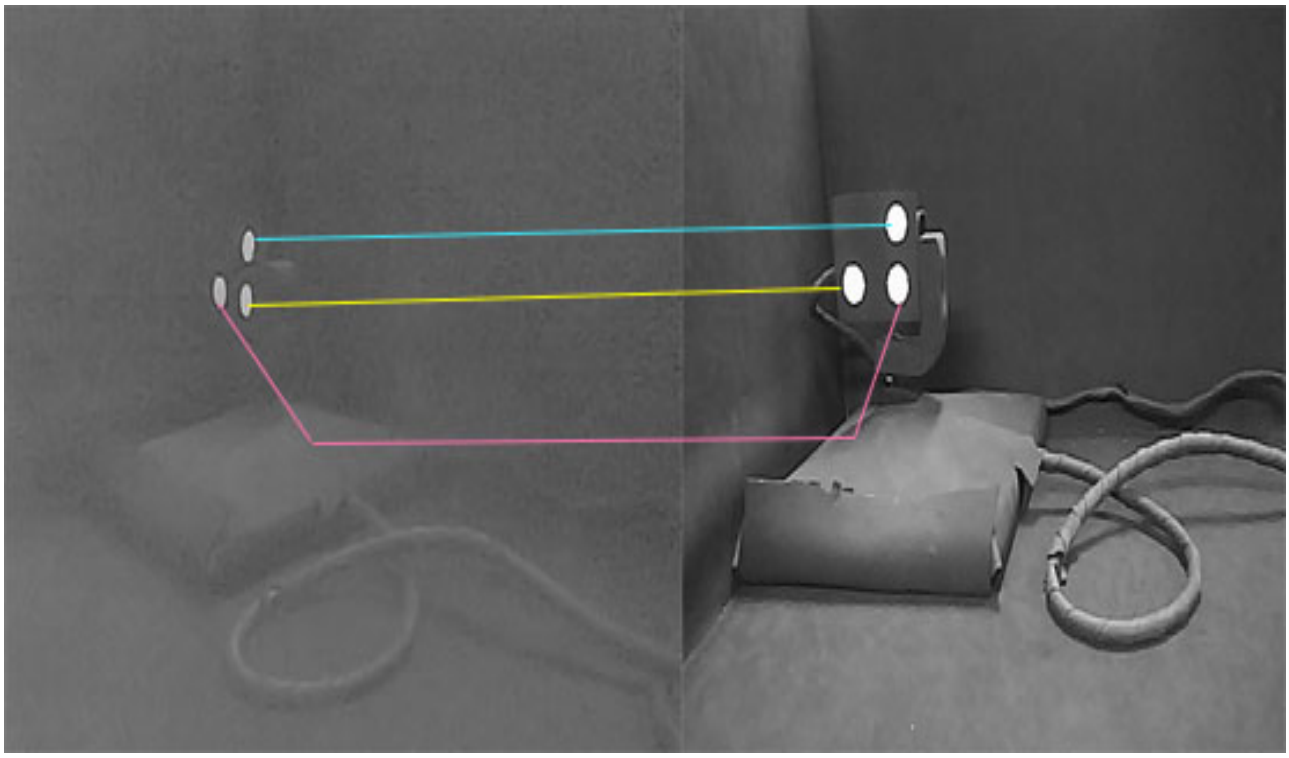
Algorithm 3 Calcular número de objetos

```
1: function ContarObjetos(Centroid_img1, Centroid_img2)
2:     tamanho_img1 = size(Centroid_img1)
3:     tamanho_img2 = size(Centroid_img2)
4:     NroObjetos = min(tamanho_img1,tamanho_img2)
5:     return NroObjetos
6: end function
```

4.3 Correlação estéreo entre duas imagens

A correlação estéreo pode ser definido como a correspondência de pontos entre duas imagens. De outra forma, é a correspondência entre a imagem 1 e a imagem 2 da Fig. 34, a qual mostra duas imagens com diferentes vistas através da VE. O objetivo deste tópico é correlacionar todos os pontos entre as duas imagens.

Figura 34 – Correlação de objetos entre duas imagens



FONTE: A autora.

Para resolver o problema da correlação entre pontos das duas imagens, utilizam-se os conceitos de VE vistos no capítulo 2. Esses conceitos são: a restrição epipolar (equação (36)), que é essencial para definir um sistema de VE, as propriedades da VE, como ser, os epipolos (equações (38) e (39)) e as linhas epipolares (equação (41)). Todas essas propriedades ajudam a resolver o problema de correlação dos sistemas de VE, de modo que é possível saber se o objeto de uma imagem pertence ao objeto da outra imagem.

4.3.1 Restrição epipolar e matriz essencial

Dados os pontos em cada uma das imagens $\boldsymbol{x}_1$, $\boldsymbol{x}_2 \in \mathbb{R}^3$ e uma matriz essencial E, que formam a chamada restrição epipolar, apresentada a seguir:

$$\boldsymbol{x}_2{}^T E \boldsymbol{x}_1 = 0$$

Onde, a matriz essencial está formada pela matriz de rotação e o operador chapéu da matriz de translação, dada por:

$$E = \hat{T}R$$

Assim, o operador chapéu do vetor de translação T $\in \mathbb{R}^3$, foi definido na equação (35)

Deste modo, é generalizando a restrição epipolar entre duas imagens para n pontos, sendo $(\boldsymbol{x}_1^1, \boldsymbol{x}_2^1)$, $(\boldsymbol{x}_1^2, \boldsymbol{x}_2^2)$ até $(\boldsymbol{x}_1^n, \boldsymbol{x}_2^m)$, onde $i = 1, 2, ..., n$, $j = 1, 2, ..., n$ objetos, tem a seguinte relação:

$$(\boldsymbol{x}_2^i)^T E \boldsymbol{x}_1^j = 0; se, i = j \tag{48}$$

$$(\boldsymbol{x}_2^i)^T E \boldsymbol{x}_1^j \neq 0; se, i \neq j \tag{49}$$

Para qualquer combinação de correlações de objetos em que a restrição epipolar é aproximadamente zero, diz-se que a condição é válida. Isso significa que uma imagem correlaciona com o objeto da outra imagem.

4.3.2 Linhas epipolares

Esta restrição ajuda na correlação entre os pontos, conhecidas como linhas epipolares apresentadas no Capitulo 2 (equação (41)), as quais são dadas por:

$$l_1^T e_1 = 0, l_2^T e_2 = 0$$

$$l_1^T \mathbf{x}_1 = 0, l_2^T \mathbf{x}_2 = 0$$

4.4 Reconstrução do espaço tridimensional para vários objetos

No capítulo anterior foi vista a reconstrução do espaço tridimensional para um objeto, através da criação de um algoritmo de VE. Esse algoritmo irá servir como uma maneira de generalizar a obtenção da profundidade para vários objetos. Além disso, neste tópico se dará uma solução ao problema de correlação entre vários pontos, por meio da restrição epipolar e propriedades epipolares

Para entrar mais plenamente na generalização do modelo matemático, a Fig. 32 é usada, a qual tem três objetos cujas profundidades "λ^i" são as incógnitas a serem obtidas; 'i' são os índices de cada objeto, sendo que i = 1,2,3 objetos; o par de imagens é representado por p_{c1} e p_{c2}, com uma pose relativa das câmeras (R, T) em relação ao sistema de coordenadas global {W}, portanto apresenta-se:

$$\lambda_1^i \boldsymbol{p}_{c1}^i = R\lambda_2^i \boldsymbol{p}_{c2}^i + T \tag{50}$$

Isolando as profundidades das imagens λ_1^i, λ_2^i :

$$\begin{bmatrix} \boldsymbol{p}_{c1}^i & -R\boldsymbol{p}_{c2}^i \end{bmatrix} \begin{bmatrix} \lambda_1^i \\ \lambda_2^i \end{bmatrix} = T \tag{51}$$

Mudando de variáveis obtêm-se a matriz M^i e o vetor λ^i:

$$M^i = \begin{bmatrix} \boldsymbol{p}_{c2}^i & -R\boldsymbol{p}_{c2}^i \end{bmatrix} \in \mathbb{R}^{3\times 2} \tag{52}$$

$$\lambda^i = \begin{bmatrix} \lambda_1^i \\ \lambda_2^i \end{bmatrix}$$

Portanto, a equação geral será:

$$M^i \cdot \lambda^i = T \tag{53}$$

Para isolar as profundidades λ^i, muda-se a variável M^i por M_p e multiplica-se a equação (53) por M_p^T :

$$M_p^i \cdot \lambda_p^i = T$$

$$M_p^T \cdot M_p \lambda^i = M_p^T \cdot T$$

Agora, multiplica-se por $(M_p^T \cdot M_p)^{-1}$

$$(M_p^T \cdot M_p)^{-1} \cdot \lambda^i \cdot (M_p^T \cdot M_p) = (M_p^T \cdot M_p)^{-1} M_p^T \cdot T$$

Remove-se $(M_p^T \cdot M_p) \cdot (M_p^T \cdot M_p)^{-1}$

$$\lambda^i = (M_p^T \cdot M_p)^{-1} \cdot M_p^T \cdot T \tag{54}$$

Para n objetos, $i = 1, 2, ...n$

Assim, λ^i são as ***profundidades tridimensionais para n objetos***, a partir do par de imagens ($\boldsymbol{p}_{c1}$ e $\boldsymbol{p}_{c2}$) capturadas pelo sistema de visão estéreo.

4.5 Algoritmo de obtenção da profundidade para múltiplos objetos

Para ilustrar o processo geral de obtenção da profundidade de vários objetos, o algoritmo 4 apresenta os passos principais da reconstrução.

Algorithm 4 Reconstrução de vários pontos

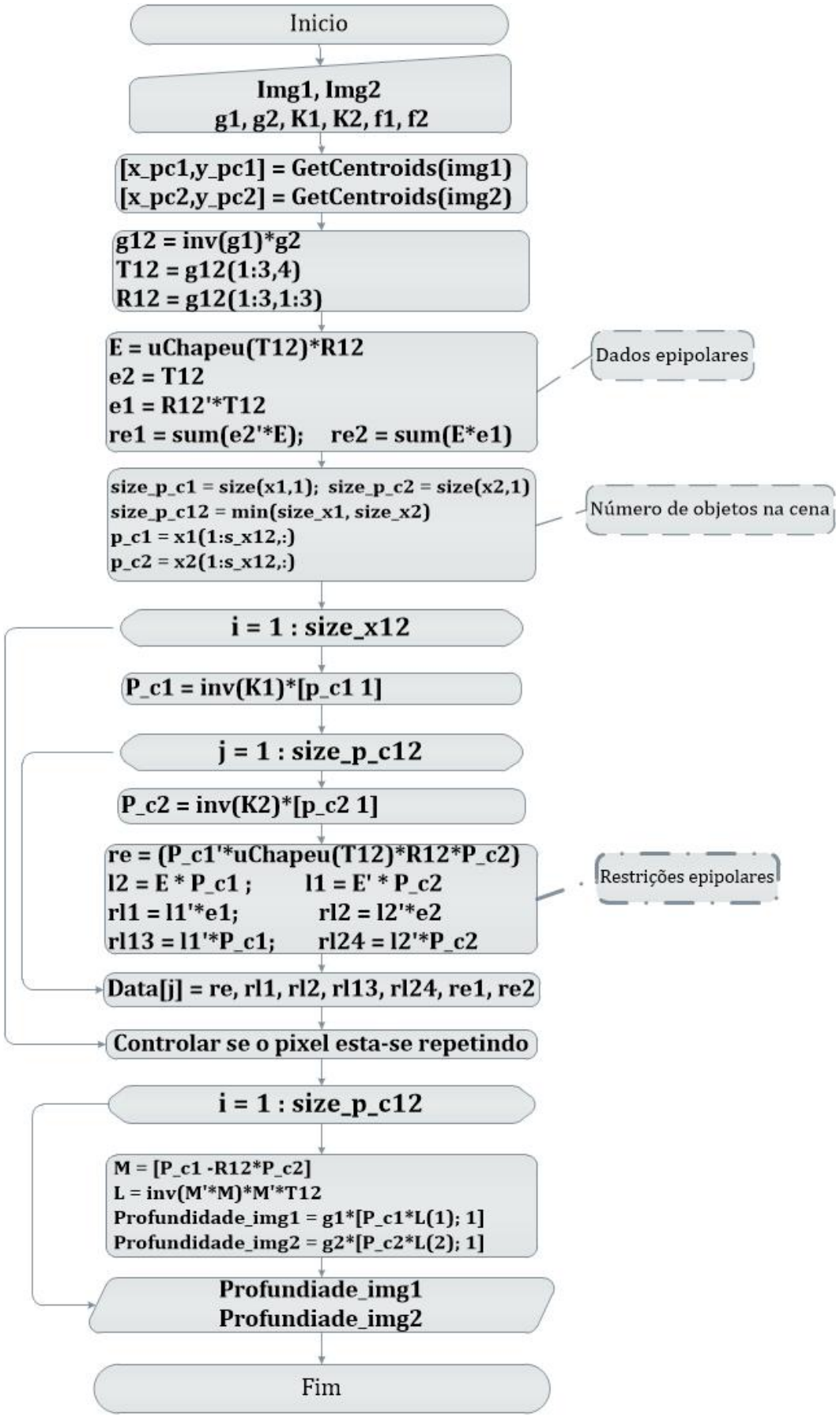
function VARIOSPONTOS3D($\bar{p_{c1}}', \bar{p_{c2}}', g_1, g_2, K_1, K_2$)
  $g_{12} = g_1^{-1} \cdot g_2$
  $T_{12} = g_{12}(1:3, 4)$; $R_{12} = g_{12}(1:3, 1:3)$; $E = uChapeu(T_{12} * R_{12})$
  $size_{pc12} = min(size_{pc1}, size_{pc2})$
  for i=1:$size_{pc12}$ **do**
    $P_{c1} = K_1^{-1} \cdot [\bar{p_{c1}}' 1]$
    for j= 1 : $size_{pc12}$ **do**
      $P_{c2} = K_2^{-1} \cdot [\bar{p_{c2}}' 1]$
      $re = P_{c1}^{-1} \cdot E \cdot P_{c2}$; $data[j] = re, P_{c1}, P_{c2}$
    end for
    $data = sort(re)$
  end for
  for i = 1 : $size_{pc12}$ **do**
    $M = [K_1^{-1} \cdot \bar{p_{c1}}' \ -R_{12} \cdot K_2^{-1} \cdot \bar{p_{c2}}']$
    $L[i] = (M^T \cdot M)^{-1} M^T \cdot T_{12}$
  end for
  return $p_w1 = g_1 L_{c1} K_1^{-1} \bar{p_{c1}}'$, $p_w2 = g_2 L_{c2} K_2^{-1} \bar{p_{c2}}'$
end function

Sendo, as variáveis:

- $\bar{p_{c1}}$: são as posições em pixels, de um dos pontos $x, y \in \mathbb{N}^{2\times1}$, da imagem da câmera 1.
- $\bar{p_{c2}}$: são as posições em pixels, de um dos pontos $x, y \in \mathbb{N}^{2\times1}$, da imagem da câmera 2.
- g_1: é a matriz de parâmetros extrínsecos $\in \mathbb{R}^{4\times4}$ na forma homogênea, que vem da calibração da câmera 1.
- g_2: é a matriz de parâmetros extrínsecos $\in \mathbb{R}^{4\times4}$ na forma homogênea, que vem da calibração da câmera 2.
- K_1: é a matriz de parâmetros intrínsecos $\in \mathbb{R}^{3\times3}$, vem da calibração da câmera 1
- K_2: é a matriz de parâmetros intrínsecos $\in \mathbb{R}^{3\times3}$, vem da calibração da câmera 2
- g_{12}: é a matriz de parâmetros extrínsecos, calculado entre as duas matrizes g_1 e g_2 $\in \mathbb{R}^{4\times4}$ na forma homogênea.
- T_{12}: é o vetor de translação $\in \mathbb{R}^{3\times1}$, separado da matriz g_{12}.
- R_{12}: é a matriz de rotação $\in \mathbb{R}^{3\times3}$, separada da matriz g_{12}.
- P_{c1}: é a projeção dos pontos no plano da imagem 1, na forma homogênea.
- P_{c2}: é a projeção dos pontos no plano da imagem 2, na forma homogênea.
- $size_{pc12}$: é a projeção das imagens.
- re: é a restrição epipolar para cada ponto.
- *data*: é a variável que armazena todos os dados epipolares e os pixeis das imagens em cada *loop*
- M: é a matriz que foi obtida da equação geral (52) do sistema de visão estéreo, que $\in \mathbb{R}^{3\times2}$.
- L: é *lambda* a posição tridimensional obtida para vários pontos da equação (54) $\in \mathbb{R}^{3\times1}$, formado pelo sistema de VE.

Por outro lado, o fluxograma da Fig. 35 apresenta os passos mais específicos da obtenção da profundidade de vários objetos.

Figura 35 – Algoritmo da implementação do modelo matemático – múltiplos pontos



FONTE: A autora.

Assim, em uma explicação mais detalhada da Fig. 35, no primeiro bloco, são introduzidas as imagens que vêm do ambiente de VE das câmeras. No segundo bloco, são introduzidos os parâmetros intrínsecos e extrínsecos, que vêm das câmeras calibradas.

No terceiro bloco obtêm-se os features de cada imagem.

No quarto bloco, são formadas as matrizes g_1, g_2; a partir dos parâmetros extrínsecos. No quinto bloco, é feita a triangulação entre as matrizes g_1 e g_2, formando o vetor de translação e de rotação da triangulação.

No sexto bloco, são calculados os dados epipolares formalizados no sistema de visão estéreo.

No sétimo bloco, é obtido o número de objetos na cena entre as duas imagens. No primeiro e no segundo *loop*, é realizada a combinação dos objetos das duas imagens. No oitavo e nono blocos, é realizada a correspondência entre o plano da retina da câmera e da matriz de pixels de cada imagem.

No décimo bloco, os dados epipolares como a restrição epipolar, as linhas epipolares e propriedades epipolares são calculadas. No décimo primeiro bloco, todos os dados epipolares e os pixels de cada imagem são armazenados. No décimo segundo bloco, são escolhidos os objetos (pixels) de menor restrição epipolar, além de verificar se um pixel está se repetindo. Já no último *loop*, a matriz M é calculada, a profundidade de cada objeto e o cálculo da posição tridimensional para cada objeto a partir da posição e orientação das matrizes g_1 e g_2 das câmeras. No último bloco, as posições tridimensionais obtidas são enviadas à tela para serem apresentadas.

5 Aplicação e resultados práticos

Este capítulo apresenta os resultados após a aplicação dos algoritmos vistos em capítulos anteriores. Os algoritmos foram implementados no Matlab, antes de serem utilizados de forma experimental.

A aplicação e os resultados do projeto estão basicamente divididos em quatro etapas. A primeira etapa é realizar a experimentação do algoritmo da profundidade para um único *feature* percorrendo o ambiente. A segunda etapa é realizar a experimentação do algoritmo da profundidade de vários *features*, formando diferentes figuras. A terceira etapa é realizar a validação do algoritmo da profundidade de vários pontos, utilizando o gimbal, o qual foi feito para esse propósito. Na quarta etapa, é apresentado o custo computacional do algoritmo de reconstrução de vários pontos.

Por outro lado, os *features* ajudaram nas duas primeiras etapas. Eles simulam as posições dos objetos movimentando-se, além de realizar trajetórias e figuras diferentes.

No caso da terceira etapa, foi feita com a ajuda do gimbal, que realiza movimentos constantes no sistema de coordenadas. Acima do gimbal, existem três *features*, que simulam o movimento dos objetos. Consequentemente, a validação é feita a partir da comparação entre os planos da reconstrução dos *features* e o plano formado pelos *servomotores* do gimbal.

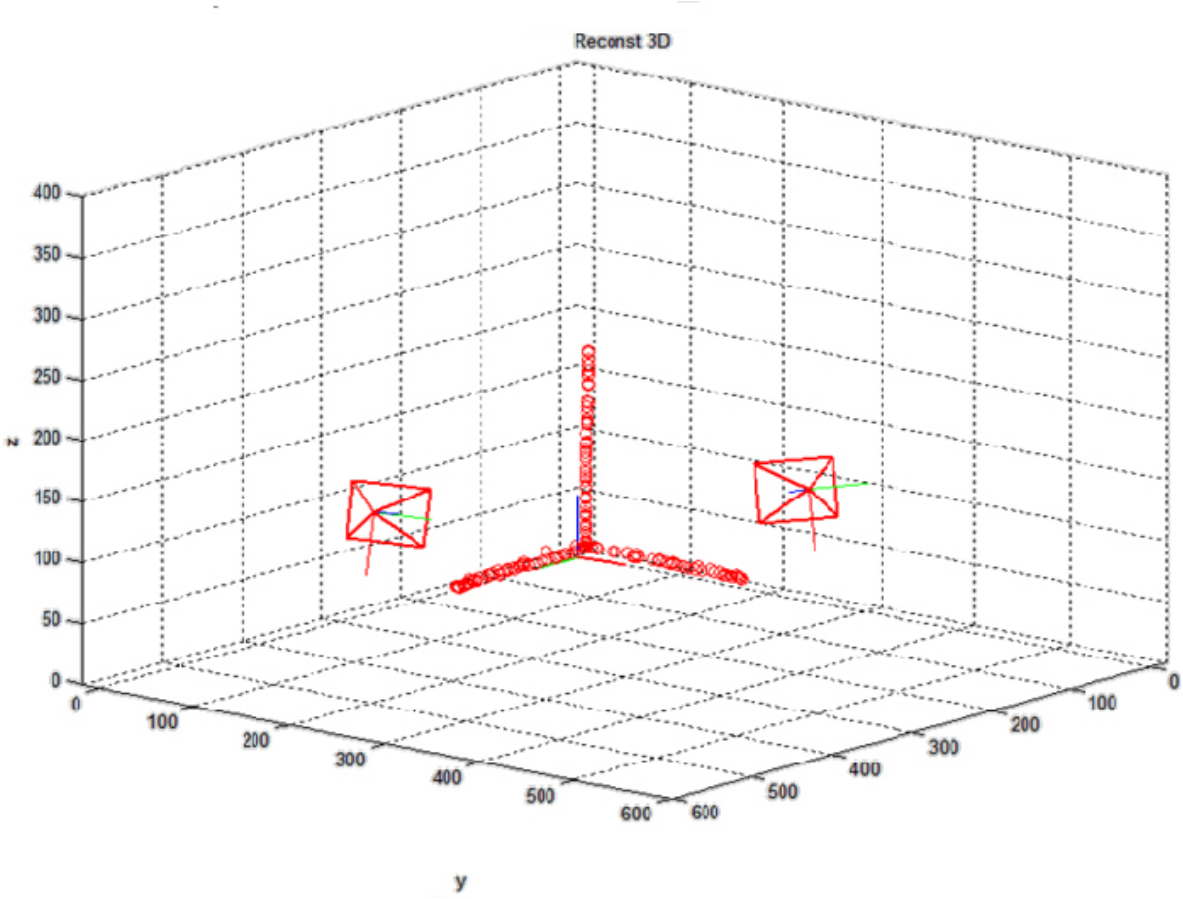
Por conseguinte, os tópicos a seguir serão organizados da seguinte maneira: resultados do algoritmo da profundidade para um ponto, resultados do algoritmo da profundidade para vários pontos, validação dos resultados e custo computacional.

5.1 Resultados da obtenção da profundidade para um ponto

De acordo com o modelo e o algoritmo, obtido no capítulo 3, aplicados no experimento de visão estéreo, conseguiram-se atingir os seguintes resultados, apresentados nas Figs. 36, 37, 38,39 e 40.

Todos esses testes são realizados com a ajuda de uma bola de isopor e diferentes planos geométricos como quadrados ou círculos colocados no ambiente de experimentação. O parâmetro utilizado para determinar se os resultados atingidos tiveram sucesso ou não é por meio da comparação entre a forma colocada no ambiente e a reconstrução.

Figura 36 – Resultados da reconstrução de um ponto – Teste 1 – trajeto do sistema de coordenadas

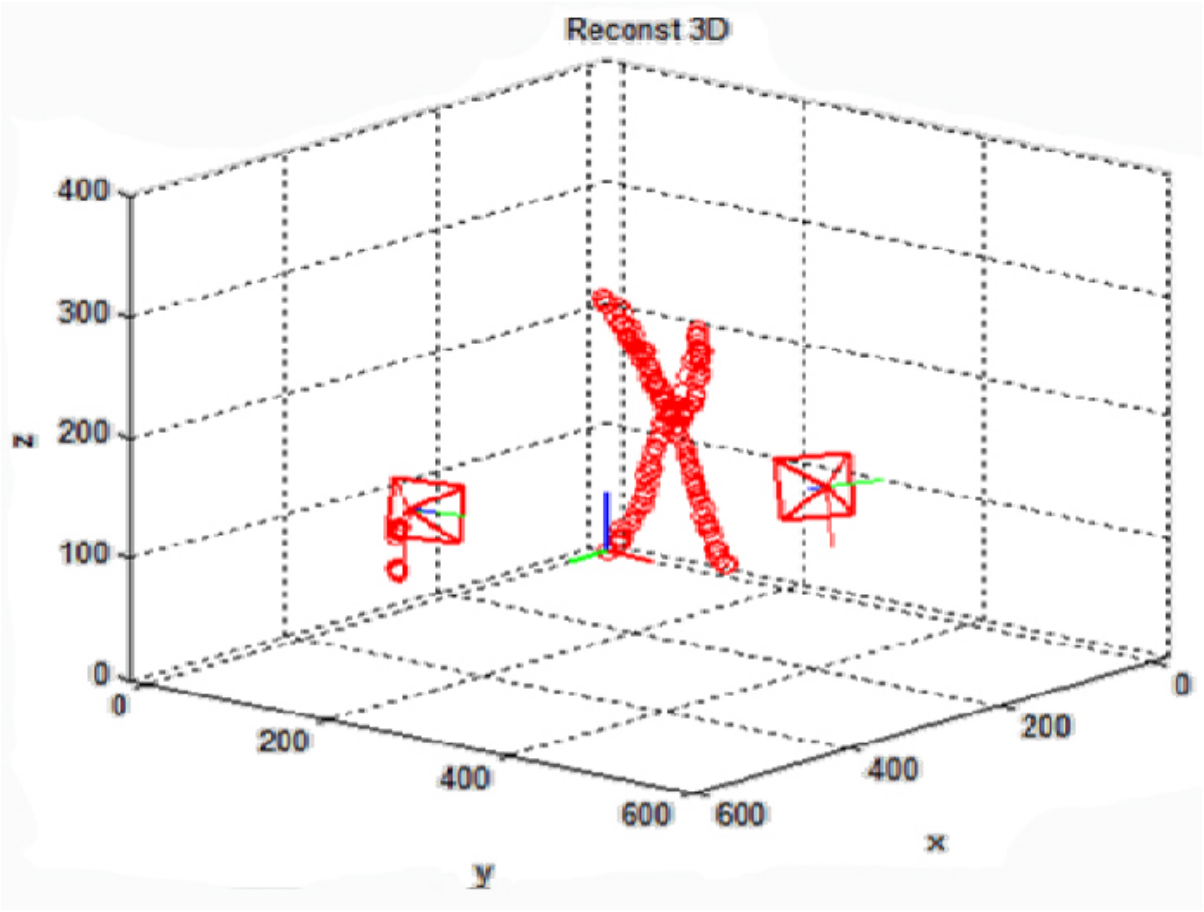


FONTE: A autora

Esse resultado da Fig. 36 foi obtido com uma bola de isopor, percorrendo os eixos de coordenadas X, Y, Z, do plano do ambiente de experimentação. Nesse resultado pode ser visto que, com uma boa calibração das câmeras, a trajetória é mais precisa, assim, o caminho seguido no ambiente de experimentação é igual ao ambiente gráfico do programa.

Algumas coisas a considerar desse teste é que a reconstrução no eixo de coordenadas "z" foi uniforme, isto é, porque a velocidade de percorrido da bola de isopor é constante, enquanto que nos outros dois eixos de coordenadas "x, y", a distância entre cada ponto reconstruído não é muito uniforme, isso é devido ao atrito entre a base do ambiente de experimentação e a bola de isopor. Mas, de maneira geral, pode-se dizer que a obtenção da profundidade da trajetória do sistema de coordenadas é considerada bem-sucedida.

Figura 37 – Resultados da reconstrução de um ponto – Teste 2 – letra “X”

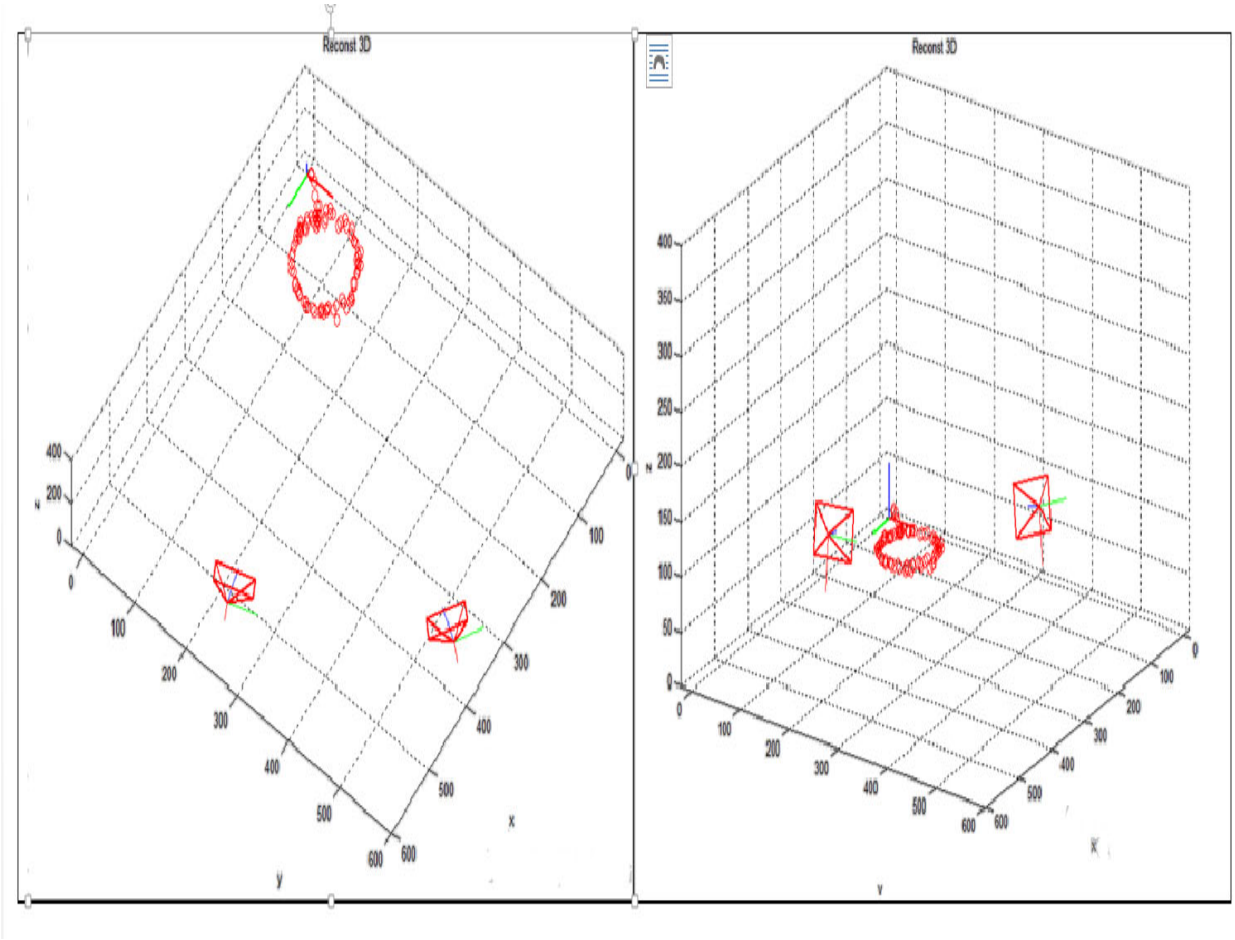


FONTE: A autora

No caso da Fig. 37, esse teste foi feito com uma bola de isopor, que realiza uma trajetória na forma da letra “X”. O trajeto é através dos eixos $\vec{y}, \vec{z}$, podendo ser visto no teste que a figura formada no ambiente gráfico do programa é a mesma da figura do ambiente de experimentação. Mas também se pode ver que existem algumas posições mal reconstruídas próximas à câmera da esquerda, que não pertencem à letra “X”, isto é devido a algum ruído do ambiente capturado pelas câmeras.

Numa análise mais detalhada sobre esse teste, pode-se verificar que a trajetória da forma tem algumas imperfeições entre a distância da reconstrução dos pontos, sendo que do lado direito da letra “X” há uma pequena divisão no caminho, pois se seguiram duas ou mais vezes pelo mesmo trajeto e na reconstrução não foram obtidas as mesmas posições dos pontos, e, além disso, o trajeto foi feito à mão livre sem o auxílio de alguma forma geométrica para seguimento das bordas. Em geral, pode-se ver a forma obtida na reconstrução é muito semelhante à forma buscada no ambiente de experimentação.

Figura 38 – Resultados da reconstrução de um ponto - Teste 3 - círculo

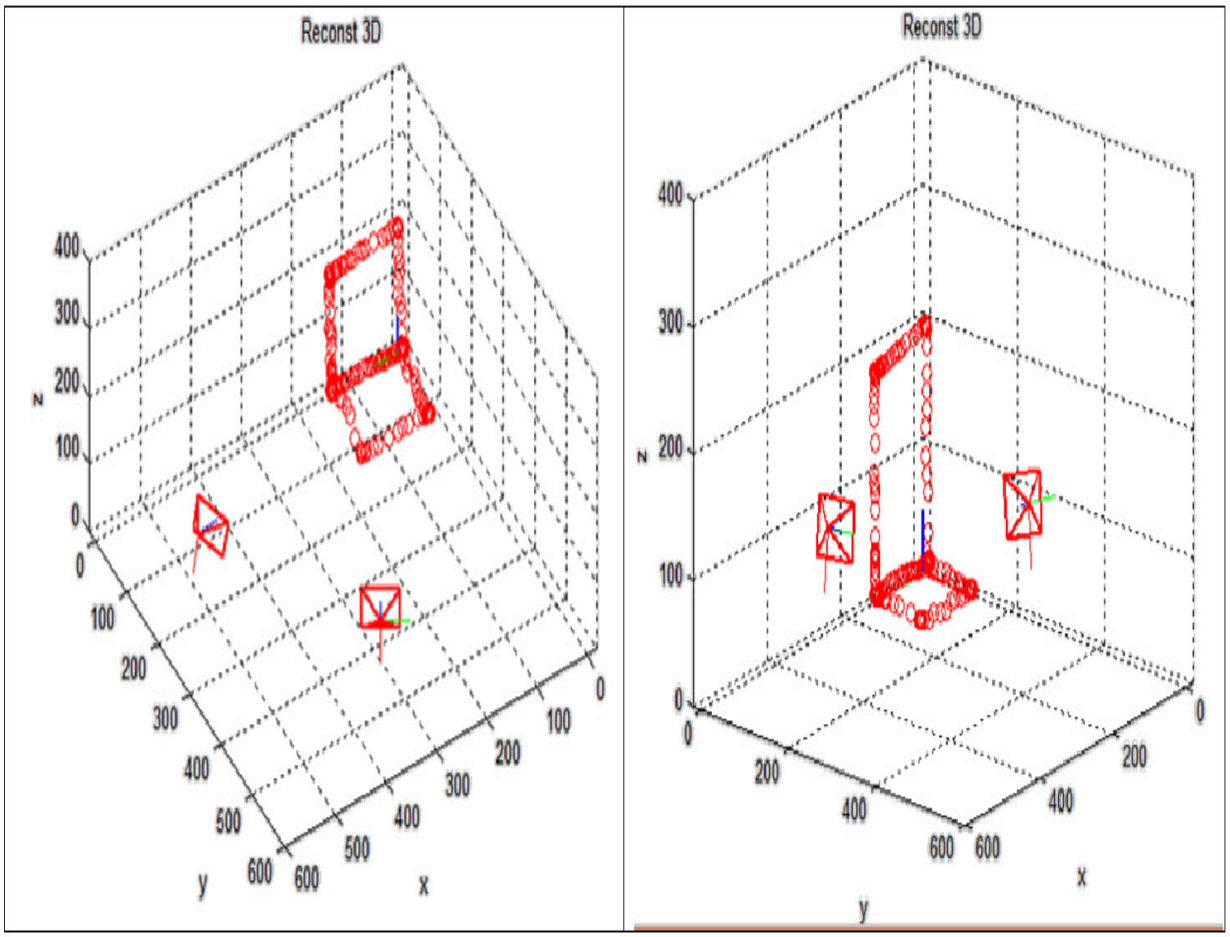


FONTE: A autora

O resultado da Fig. 38 foi feito através da colocação de uma forma geométrica de um círculo no ambiente do experimento; em seguida, com a ajuda de uma bola de isopor, realizou-se a trajetória ao redor da forma. Em relação aos resultados, pode-se observar que a forma colocada no ambiente é exatamente à forma da reconstrução. Na figura da esquerda, pode-se ver que a figura da reconstrução é um círculo e, comparando com uma segunda figura de outra vista, também forma um círculo, e isso afirma que a reconstrução é igual à forma geométrica colocada no ambiente real.

De forma mais detalhada para esse teste, pode ser visto que, apesar de ser utilizada uma forma geométrica como um círculo no ambiente de experimentação, existem algumas irregularidades na forma reconstruída, causadas pela imprecisão ao realizar o trajeto ou por algum ruído no ambiente, e isso pode ser visto na figura da esquerda. Por outro lado, o círculo reconstruído no ambiente gráfico é muito semelhante à forma geométrica do ambiente de experimentação, portanto foi realizada uma boa reconstrução.

Figura 39 – Resultados da reconstrução de um ponto - Teste 4 - forma de dos quadrados

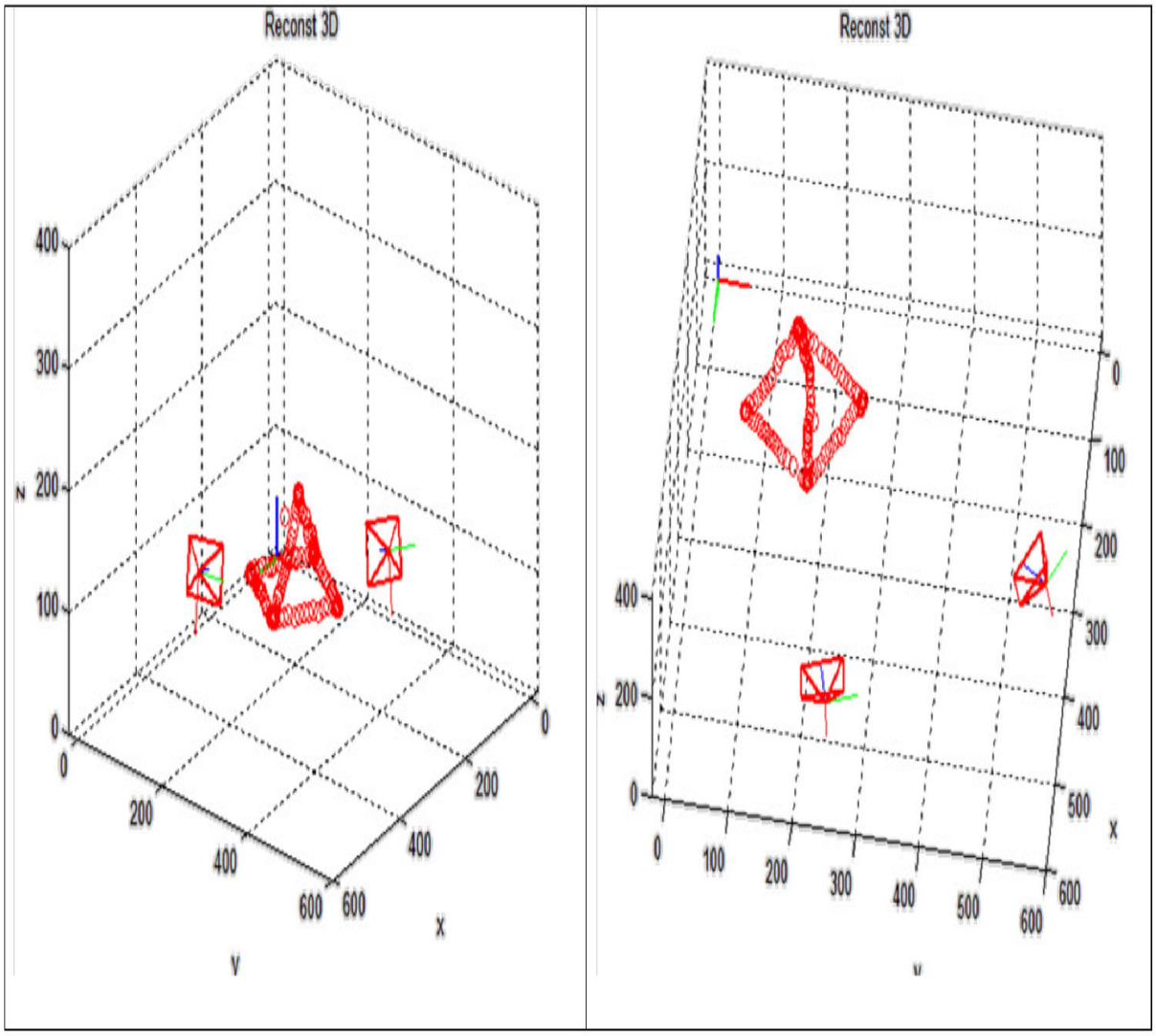


FONTE: A autora.

Nesse teste da Fig. 39, dois planos em forma de quadrados foram colocadas, uma na base do experimento nos eixos de coordenadas X, Y e o outro ao longo dos eixos X, Z, o trajeto foi feito através, de uma bola de isopor percorrendo as bordas dos planos.

Por outro lado, pode-se ver que os dois quadrados têm formas precisas, e na reconstrução não possuem irregularidades, em razão de que os planos são regulares. Outra observação é que no plano entre os eixos de coordenadas x, z, existe uma distância entre alguns pontos reconstruídos, que não é uniforme em alguns setores. Isso é devido à velocidade de deslocamento da bola de isopor no ambiente de experimentação. Portanto, a reconstrução é considerada bem-sucedida, pois a forma buscada no ambiente real é a mesma da reconstrução do ambiente gráfico.

Figura 40 – Resultados da reconstrução de um ponto - Teste 5 - forma parecida ao diamante



FONTE: A autora.

O último teste da Fig. 40 foi realizado com a ideia de fazer uma forma parecida ao diamante, foi utilizado uma forma de um quadrado na base do ambiente, e as arestas foram feitas à mão livre. Esse teste foi realizado por meio da trajetória percorrida entre as bordas com a bola de isopor.

Em relação ao resultado, pode ser visto que o quadrado reconstruído da base do experimento tem uma forma uniforme, com pequenas irregularidades num lado do quadrado. Isso é devido à velocidade da trajetória da bola de isopor e o atrito entre o ambiente e a bola de isopor. Entretanto, as arestas seguiram trajetórias regulares. Assim, os resultados obtidos no ambiente gráfico são iguais à forma desejada no ambiente real.

De acordo com os resultados obtidos, pode-se ver que em cada teste as trajetórias realizadas no ambiente de experimentação correspondem às imagens obtidas no ambiente gráfico. Nesse sentido, a reconstrução das formas geometrias no ambiente gráfico foi satisfatória.

Portanto, pode-se concluir que, de acordo com o modelo matemático construído em conjunto com o algoritmo implementado de reconstrução de um único ponto, cumpriu-se o objetivo proposto de obter o espaço tridimensional para um único ponto num sistema de VE.

A partir desses resultados da obtenção da profundidade para um ponto, o algoritmo será generalizado para obter a posição tridimensional para vários pontos. Os resultados da implementação desse algoritmo serão apresentado no seguinte tópico.

5.2 Resultados da obtenção da profundidade para vários pontos

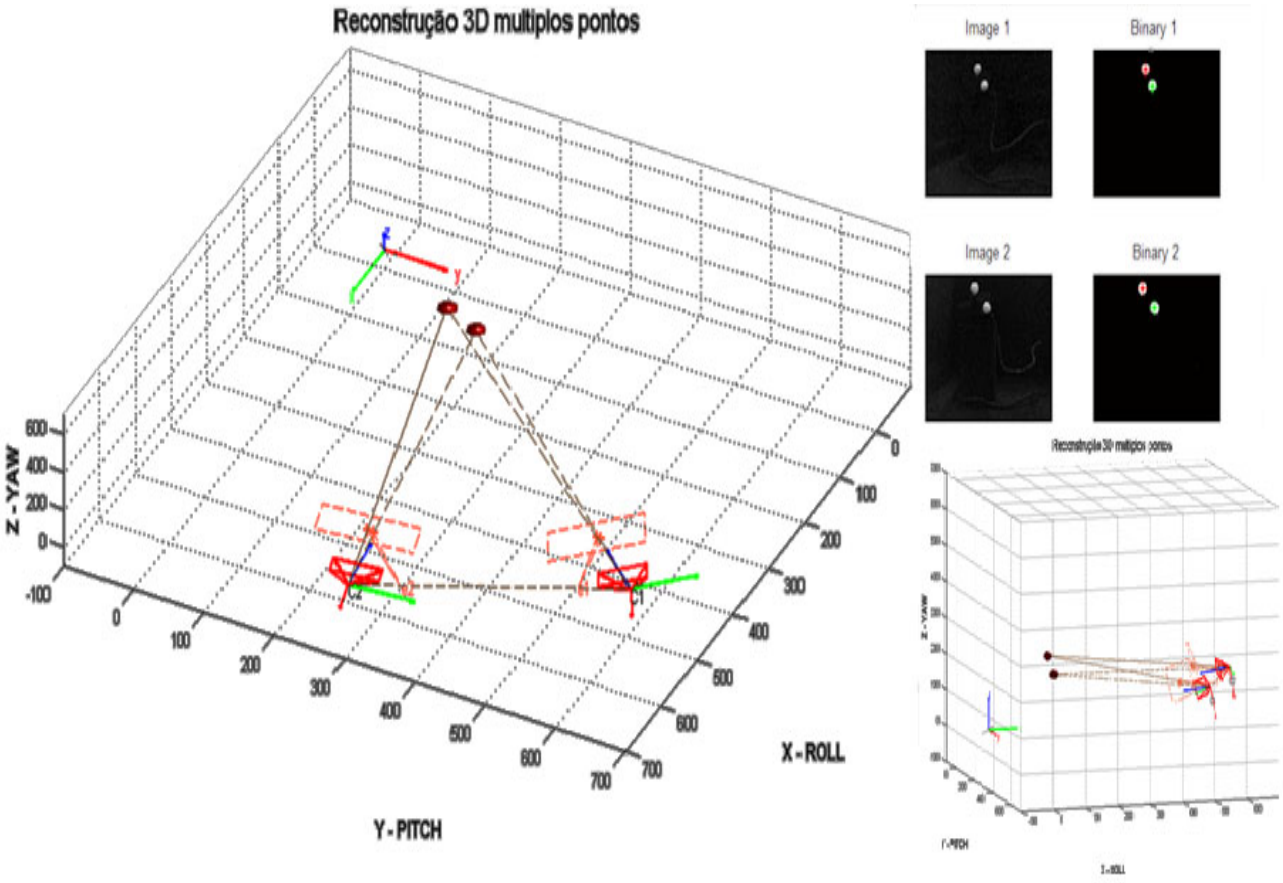
Para testar o algoritmo da obtenção da profundidade de vários pontos, foram utilizados o ambiente de experimentação do sistema de VE e *features*, tais como bolas de isopor e planos pretos com marcadores pintados de branco. Esses *features* servem para simular a posição de vários pontos no ambiente de experimentação. A verificação dos testes é através da comparação entre as figuras formadas com os *features* no ambiente de experimentação e a reconstrução do ambiente gráfico.

Assim, ao realizar os testes, surgiu o problema do alinhamento entre os pontos, sendo que isso acontece quando os pontos estão na mesma altura, nos eixos X ou Y da imagem. O problema do alinhamento foi diminuído através da criação de dois algoritmos: o primeiro por meio do controle das restrições epipolares, e o segundo através do controle de distâncias entre pixels das colunas das imagens.

Entretanto, o algoritmo de reconstrução apresenta os resultados em diferentes vistas, tais como: a vista da reconstrução dos pontos, a vista da reconstrução junto com as linhas epipolares, epipolos e projeções da imagem, a vista da validação entre o plano de orientação da reconstrução e do gimbal e a última vista com as imagens dos pontos capturados pelas câmeras, em que os pontos estão marcados com a correlação entre imagens.

Resultados para dois pontos:

Figura 41 – Resultados da reconstrução de dois pontos



FONTE: A autora.

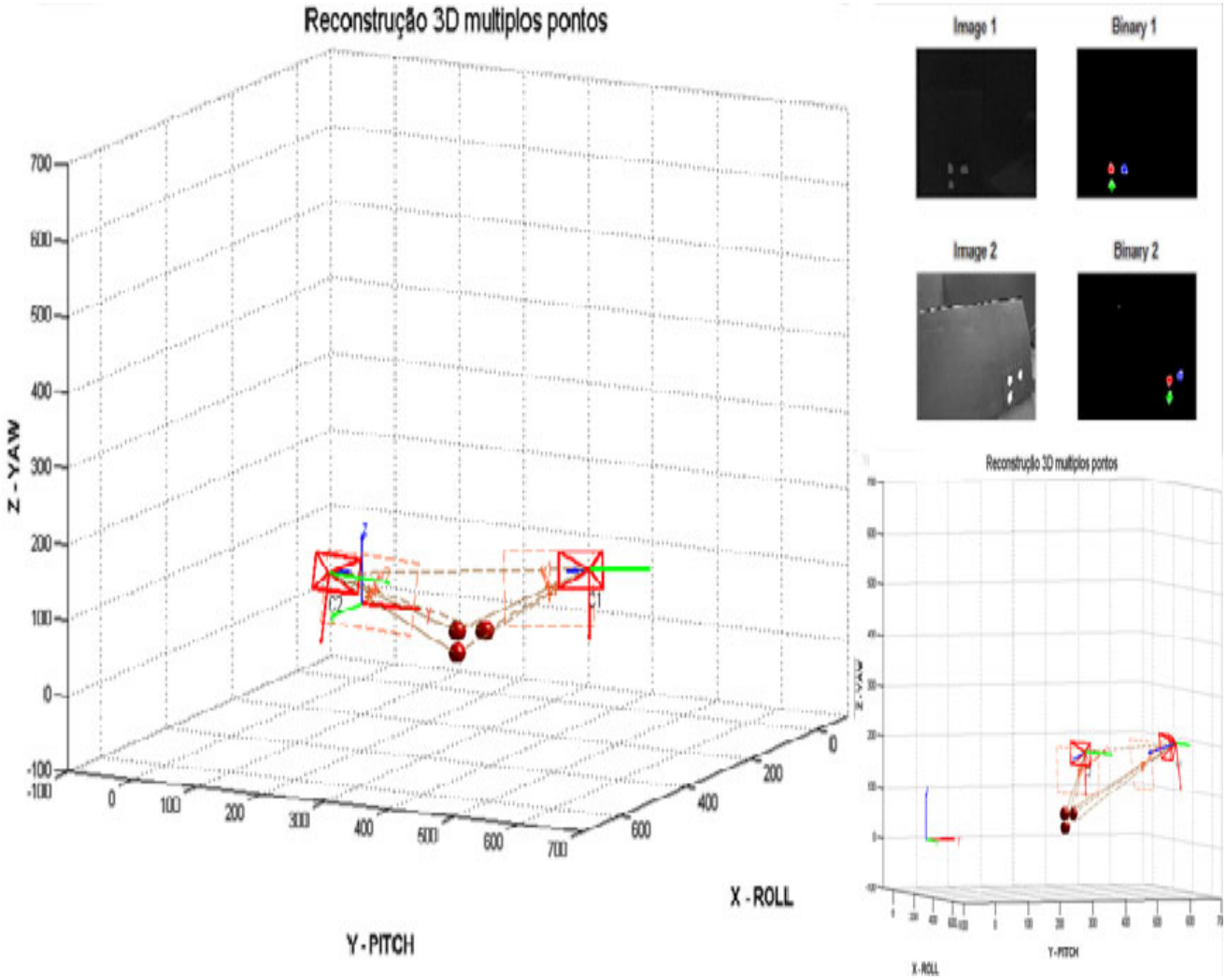
Esse teste da Fig. 41 foi feito com duas bolas de isopor, colocadas em ordem diferente. Na vista maior da esquerda, pode-se ver a reconstrução de dois pontos, os quais contêm seu próprio plano epipolar, ou seja, suas linhas epipolares, epipolos e projeções das imagens para cada ponto. Na vista menor da direita acima, estão as imagens reais e binarizadas capturadas pelas câmeras. Nessa vista, está a correlação entre os pontos, podendo-se ver a correspondência dos pontos entre as imagens classificados por cores acima de cada ponto. Por exemplo, se o asterisco vermelho num ponto da imagem 1 tem seu asterisco vermelho no mesmo ponto da imagem 2, significa que foi feita uma boa correlação entre pontos. Na última vista da direita abaixo, está a mesma reconstrução da primeira vista, só que apresentada em um ângulo diferente.

Em relação aos resultados obtidos dessa reconstrução, pode ser visto que ela foi bem-sucedida, porque a correlação entre os dois pontos das duas imagens foi feita corretamente, o que ajudou a realizar uma triangulação correta com seus respetivos pontos.

Portanto, realizou-se uma boa reconstrução entre de dois pontos.

Resultados para três pontos:

Figura 42 – Resultados da reconstrução para três pontos

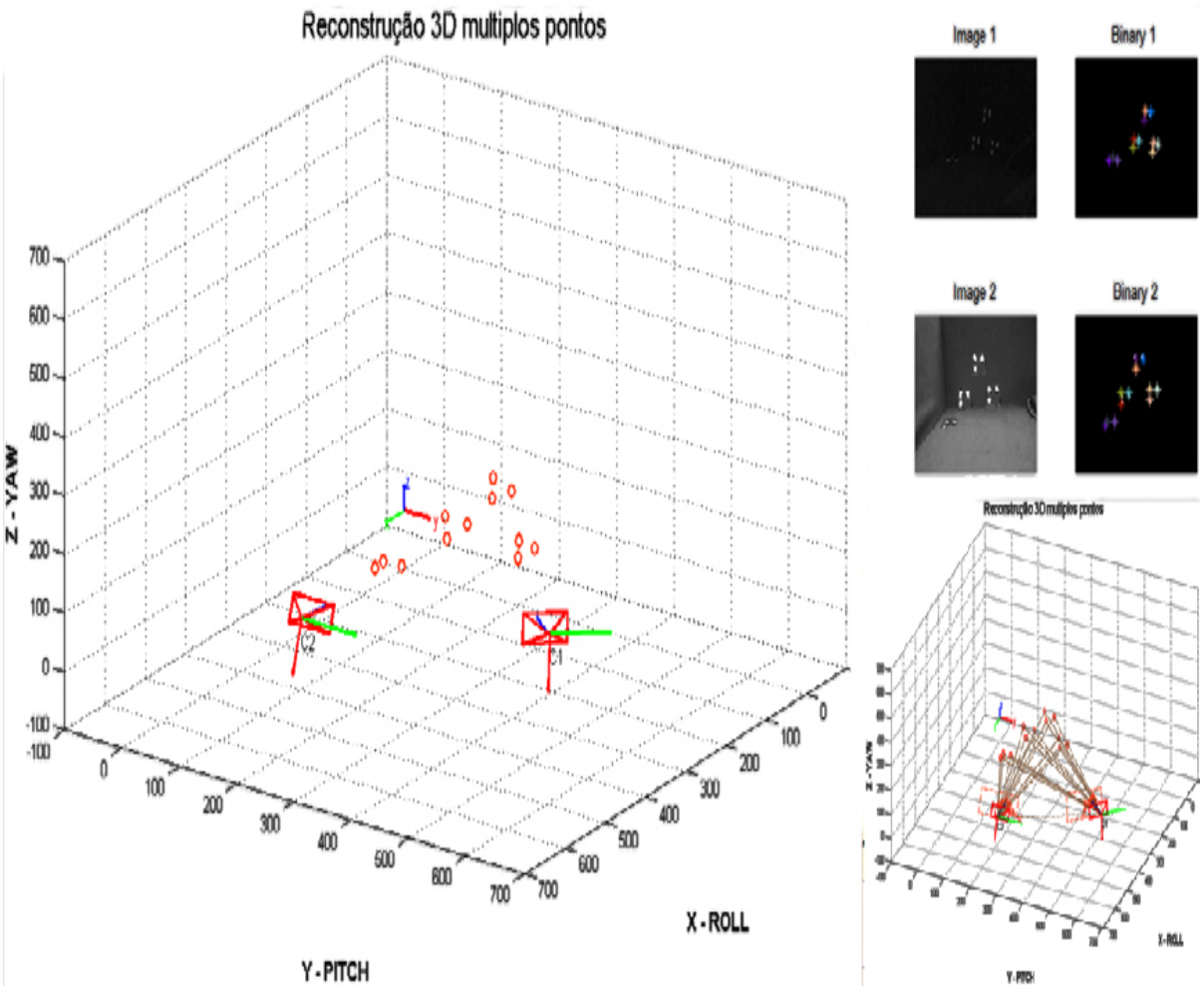


FONTE: A autora.

Esse teste da Fig. 42 foi feito com a ajuda de um plano preto com três marcadores brancos pintados acima desse. Esses marcadores simulam a posição dos objetos. Na vista maior do lado esquerdo, é possível observar que os três pontos reconstruídos do ambiente gráfico são iguais à figura formada no ambiente de experimentação. Nessa vista, é apresentado o plano epipolar para cada ponto, os epipolos e linhas epipolares. Na vista da parte superior direita, pode-se observar que os três pontos foram correlacionados corretamente, já que as cores dos asteriscos de cada ponto da imagem 1 são as mesmas do que as da imagem 2. Por conseguinte, a correlação entre pontos foi feita corretamente, então a reconstrução para três pontos pode ser considerada bem-sucedida.

Resultados para doze pontos:

Figura 43 – Resultados da reconstrução para doze pontos.



FONTE: A autora.

Esse resultado da Fig. 43foi conseguido por meio de quatro planos pretos onde cada plano tem três marcadores pintados em forma de triângulo. Os planos foram colocados no ambiente de experimentação, numa ordem diferente. Ao observar os resultados da primeira vista à esquerda, a disposição dos triângulos são iguais comparando, o ambiente de experimentação e o ambiente gráfico.

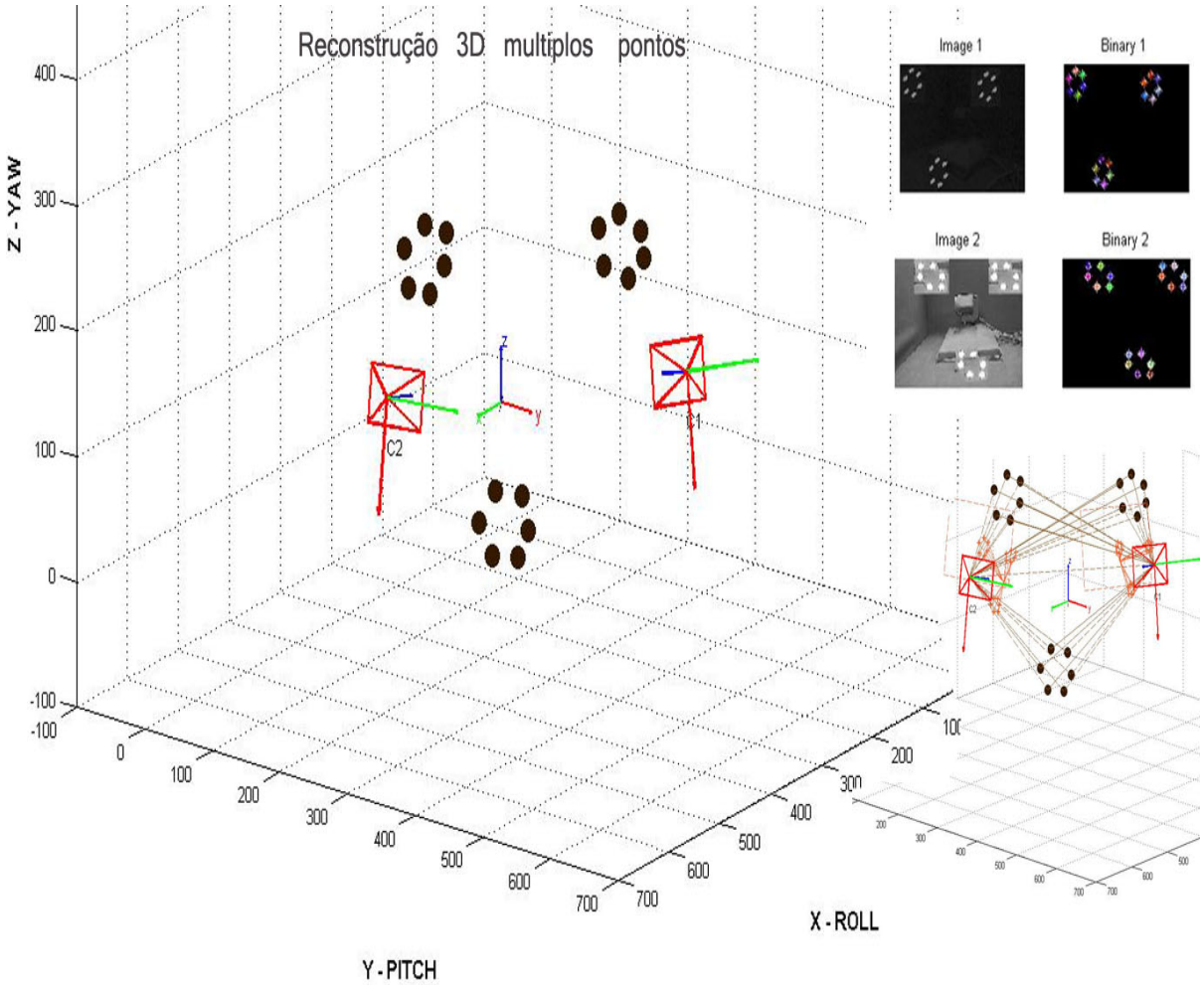
A segunda vista à direita acima mostra a correlação entre os pontos, podendo-se verificar que, dos quatro grupos de triângulos reconstruídos, dois deles estão correlacionados corretamente e dois deles não. Isso é causado pelo problema de alinhamento entre os pontos e devido aos pontos estarem unidos.

A terceira vista à direita abaixo apresenta um plano epipolar para cada ponto, onde

se visualizam as linhas epipolares, epipolos e projeções de imagens, que foram realizadas corretamente. Portanto, numa análise geral dos resultados, apesar dos erros de correlação entre alguns pontos, a reconstrução foi boa, já que comparamos os grupos de triângulos do ambiente gráfico e do ambiente de experimentação.

Resultados para vários pontos:

Figura 44 – Resultados da reconstrução de vários pontos.



FONTE: A autora.

Para esse teste da Fig. 44, foram utilizados três planos pretos, em que cada plano tem seis marcas brancas na forma de círculos. Na primeira vista à esquerda, foi possível ver que as formas do ambiente gráfico são as mesmas que o ambiente de experimentação. Enquanto que, na segunda vista à direita acima, é visto que alguns pontos não são bem correlacionado, mas outros pontos sim . A terceira vista mostra os planos epipolares para cada ponto, as linhas epipolares, os epipolos e projeções das imagens, que foram projetados corretamente. Portanto, fazendo uma avaliação dessa reconstrução, é visto que

a maioria dos pontos foi correlacionada corretamente; além disso, em razão da forma entre os círculos formados do ambiente de experimentação serem iguais às figuras do ambiente gráfico, pode-se dizer que houve uma boa reconstrução.

Em conclusão, para todos os testes realizados pode ser visto que, quanto maior a quantidade dos pontos nas imagens, maior probabilidade que os pontos fiquem alinhados e apareçam mais erros na reconstrução, o que torna a reconstrução mais complexa.

Por outro lado, os erros da correlação entre pontos são causados pelo alinhamento, entre a linha ou coluna dos pixels das imagens. Quando aparece o alinhamento entre pontos, é menos provável que exista uma boa correlação, devido aos pontos alinhados poderem ter a mesma restrição epipolar, então se torna mais difícil a escolha de pontos a serem correlacionados. Para diminuir esse problema, foi criado um algoritmo que controla a escolha de pontos prováveis a serem correlacionados, através do controle de colunas entre os pixels das duas imagens. Com esse algoritmo, diminuiu o problema de alinhamento e melhorou a correlação, contudo ainda existe o problema em menor quantidade, o qual poderia ser resolvido através da criação de uma lei de controle para a reconstrução.

De modo geral, a criação do algoritmo da reconstrução de vários pontos foi satisfatória, quando os pontos estejam dispersos e não alinhados. Contudo, esse algoritmo da reconstrução de vários pontos será validado na seguinte seção.

Contudo, este algoritmo da reconstrução de vários pontos, será validado na seguinte seção.

5.3 Validação do algoritmo de vários pontos

Uma vez feita a reconstrução para vários pontos, o seguinte passo é a validação dos pontos reconstruídos. Essa validação surge da necessidade de saber se a reconstrução foi feita corretamente, sendo que é difícil saber se foram obtidas posições tridimensionais corretas, somente com a comparação das figuras formadas entre o ambiente de experimentação e o ambiente gráfico. Entretanto, os resultados da seção anterior podem ser considerados resultados *a - priori*, antes da validação.

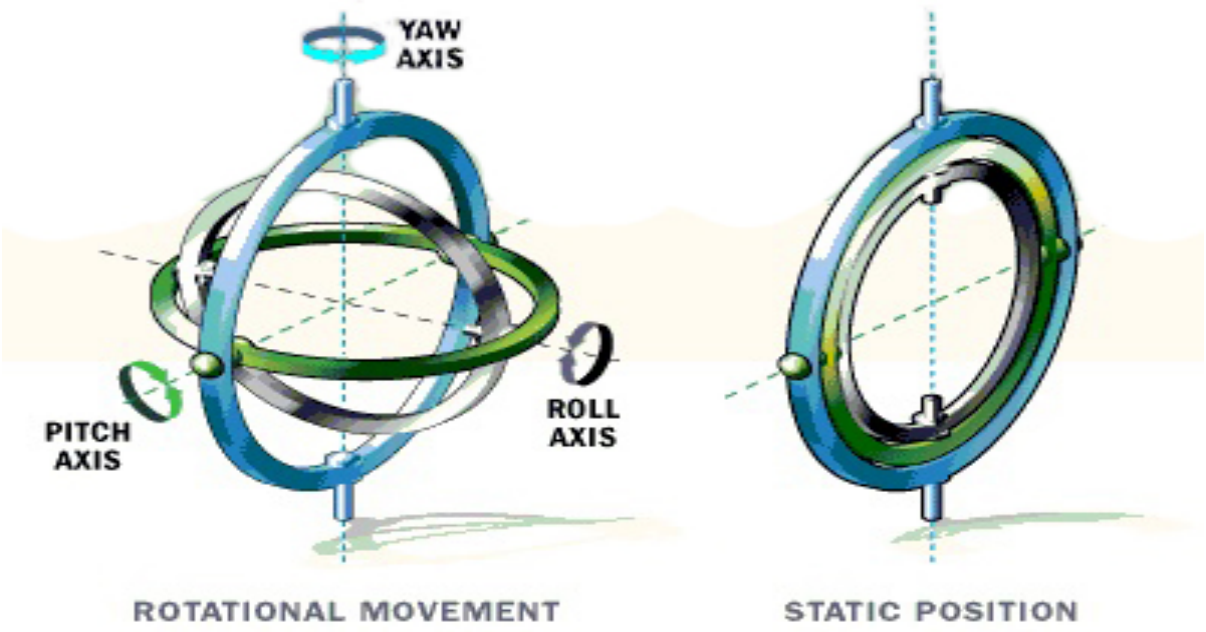
Portanto, para validar o algoritmo de reconstrução de vários pontos, foi construído um gimbal, com o objetivo de fornecer as posições do seu plano de orientação, calculado por sensores, que serve para comparar com o plano formado da reconstrução dos pontos. Por conseguinte, através dessa comparação, os resultados da reconstrução de vários pontos serão validados.

Tudo isto será explicado nas seções seguintes, as quais foram organizadas da seguinte maneira: funcionamento do gimbal, funcionamento do plano da reconstrução de pontos e comparação entre resultados do gimbal e da reconstrução.

5.3.1 Funcionamento do Gimbal

O gimbal apresentado na Fig. 45 é um suporte oscilante que permite a rotação de um objeto sobre o mesmo sistema de coordenadas, tem um conjunto de três suspensões, uma montada sobre a outra. Vários modelos de gimbal têm aparecido com uma série de anéis concêntricos. O anel mais externo é montado em uma superfície maior, o segundo anel se liga ao anel exterior em dois pontos perpendiculares à superfície e o terceiro anel menor é montado no segundo anel em dois pontos perpendiculares. O gimbal pode ser utilizado para permitir que um objeto montado no interior possa manter-se independente da rotação do seu apoio. No entanto, o gimbal tem um problema de bloqueio, que ocorre quando dois eixos de coordenadas ficam alinhados. Quando isso acontece, o movimento do objeto é limitado. Além disso, o gimbal pode ser aplicado em sistemas de navegação inercial, motores de foguete, cronômetros marítimos, vídeo ou fotografia. Por exemplo, atualmente é utilizado nos VANTs, onde as câmeras estão montadas no gimbal (BRUSHLESSGIMBALS, 2013).

Figura 45 – Funcionamento de um gimbal geral

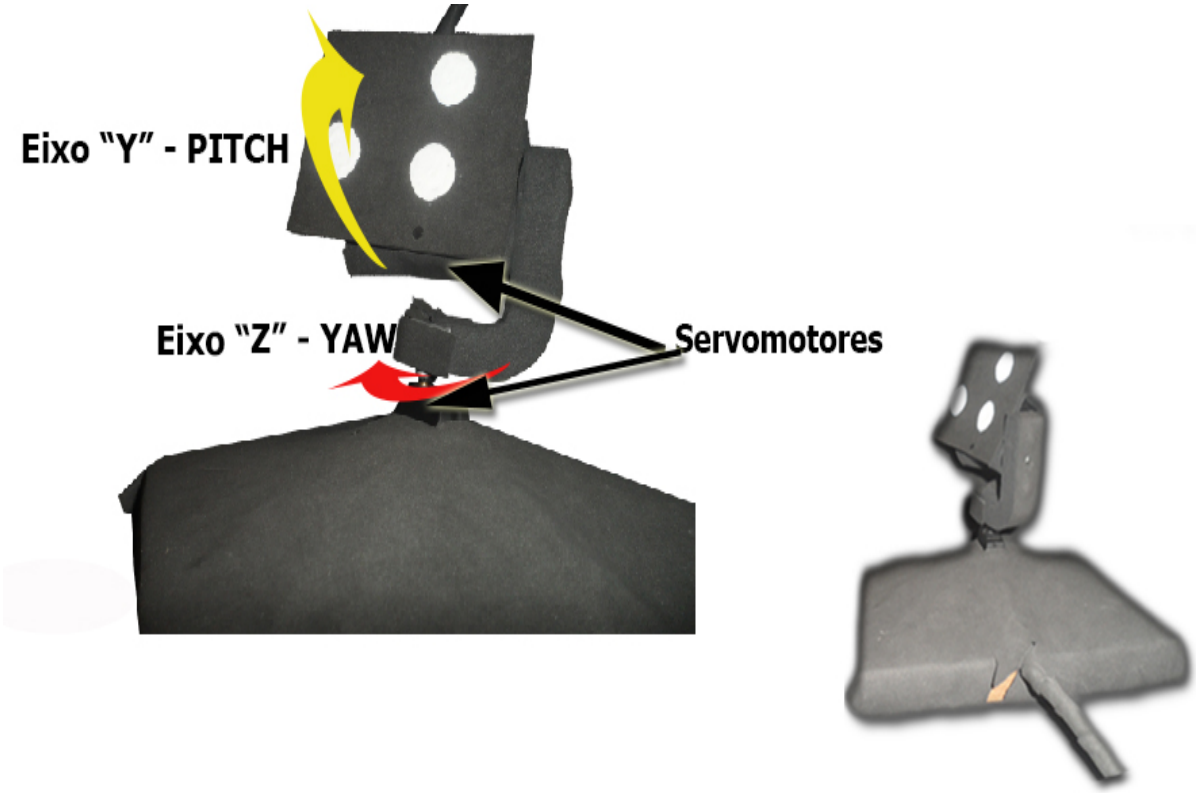


FONTE: (STRICKLAND, 2015)

No projeto, foi construído o gimbal da Fig. 46, com o objetivo de enviar as posições angulares do movimento do plano do gimbal. O gimbal tem um sistema de coordenadas, representados pelos três eixos $X = Roll$, $Y = Pitch$, $Z = Yaw$. O gimbal tem dois servomotores, um para o *Pitch* e o outro para o *Yaw*, sendo que o *Roll* obedece aos movimentos dos sensores anteriores. Os servomotores destinam-se a enviar as posições angulares em cada

movimento que eles fazem, sendo essas posições enviadas através de uma placa "*arduino DUE*", que envia essas posições para o programa principal da reconstrução de pontos feito no *matlab*.

Figura 46 – Funcionamento do gimbal construido



FONTE: A autora.

Os movimentos de cada servomotor estão entre 0° e 180°, com o objetivo de que as câmeras possam alcançar a visão dos pontos nas imagens. Para conseguir a visão nas câmeras, realizou-se a calibragem do gimbal, em que foram reduzidos os intervalos entre as posições angulares e a velocidade dos servomotores. Essa calibração permitiu alcançar a visão em ambas câmeras e permitiu ao sistema de VE proposto, funcionar corretamente.

Depois da calibração do gimbal, foi obtido o plano de orientação do seu movimento, baseado nas posições angulares dos servomotores *Yaw* e *Pitch*. Entretanto, o objetivo do gimbal foi obter o plano de orientação do seu movimento.

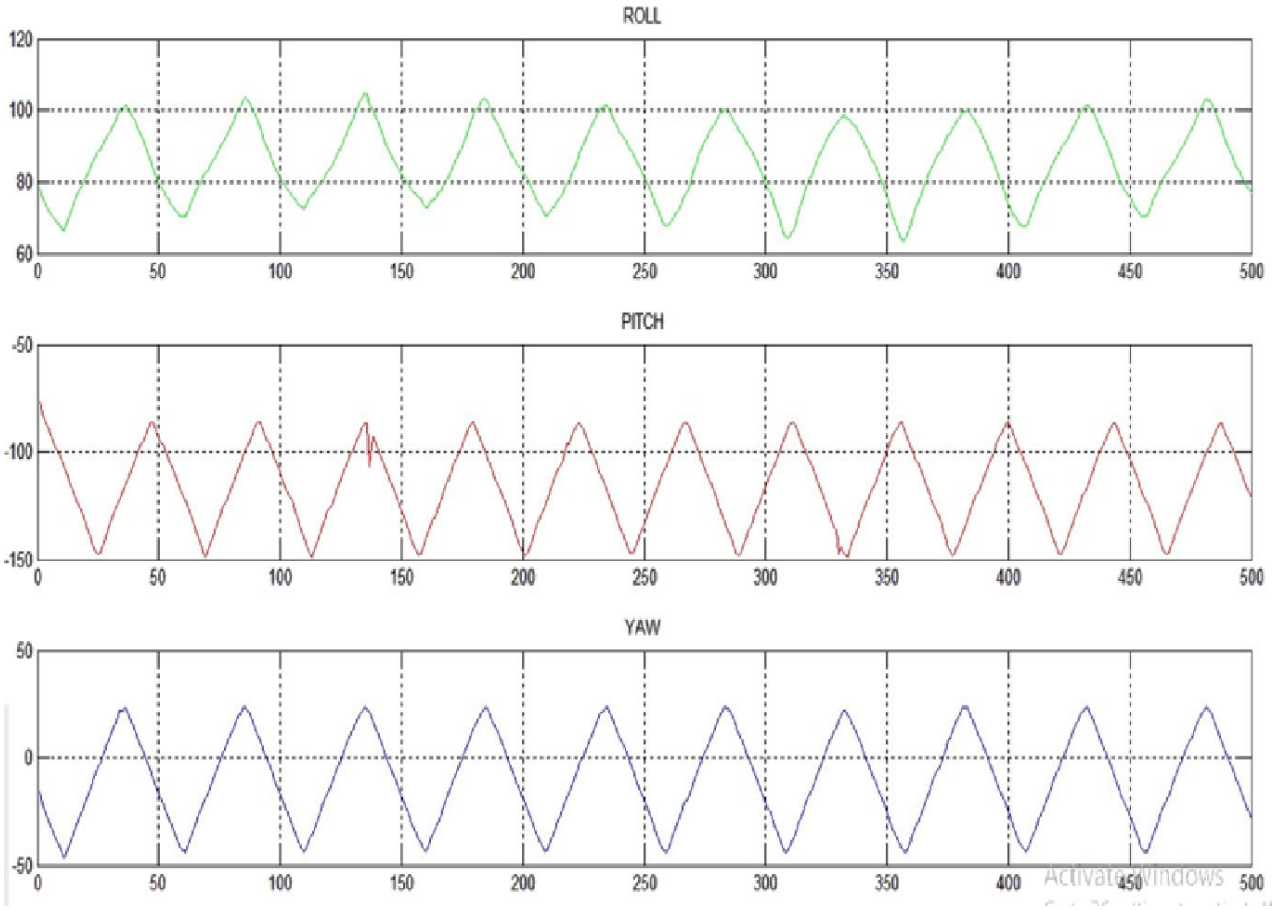
Uma vez explicado o funcionamento do gimbal, o próximo passo é fazer com que ele funcione. Para isso, deve-se conectar o gimbal numa tensão de entrada de 12 volts e conectar a uma porta USB no computador, para receber as posições angulares. Uma vez conectadas todas as portas respetivas, os servomotores começam a girar sobre os eixos *Pitch* e *Yaw*. A partir desse movimento do gimbal, é possível obter seu sistema de

coordenadas; em outras palavras, é possível obter o plano de orientação do movimento do gimbal. Outros detalhes do gimbal podem ser vistos no Apêndice A.

Enquanto, aos resultados do plano de orientação do gimbal serão exibidos nas Figs. 47 e 48 e o período de amostragem do movimento na Fig. 49.

Para este resultado da Fig. 47, são apresentados os ângulos de rotação obtidos em cada eixo de coordenadas *Roll*, *Pitch* e *Yaw*, do plano de orientação do gimbal.

Figura 47 – Resultados dos ângulos de orientação do plano do gimbal



FONTE: A autora.

Entretanto, na Fig. 47 foram apresentados os valores das posições angulares enviados pelos servomotores, que representam a orientação do plano do gimbal, em relação ao sistema de coordenadas do mundo {W}, do ambiente de experimentação.

Por outro lado, na Fig. 47, são apresentadas as três posições angulares dos servomotores (*Roll*, *Pitch* e *Yaw*), mas, como foi explicado no tópico acima, o gimbal envia só duas posições angulares dos servomotores (*Pitch* e *Yaw*). Assim, para obter estes resultados, foram levados ao sistema de coordenadas em relação ao ambiente de experimentação,

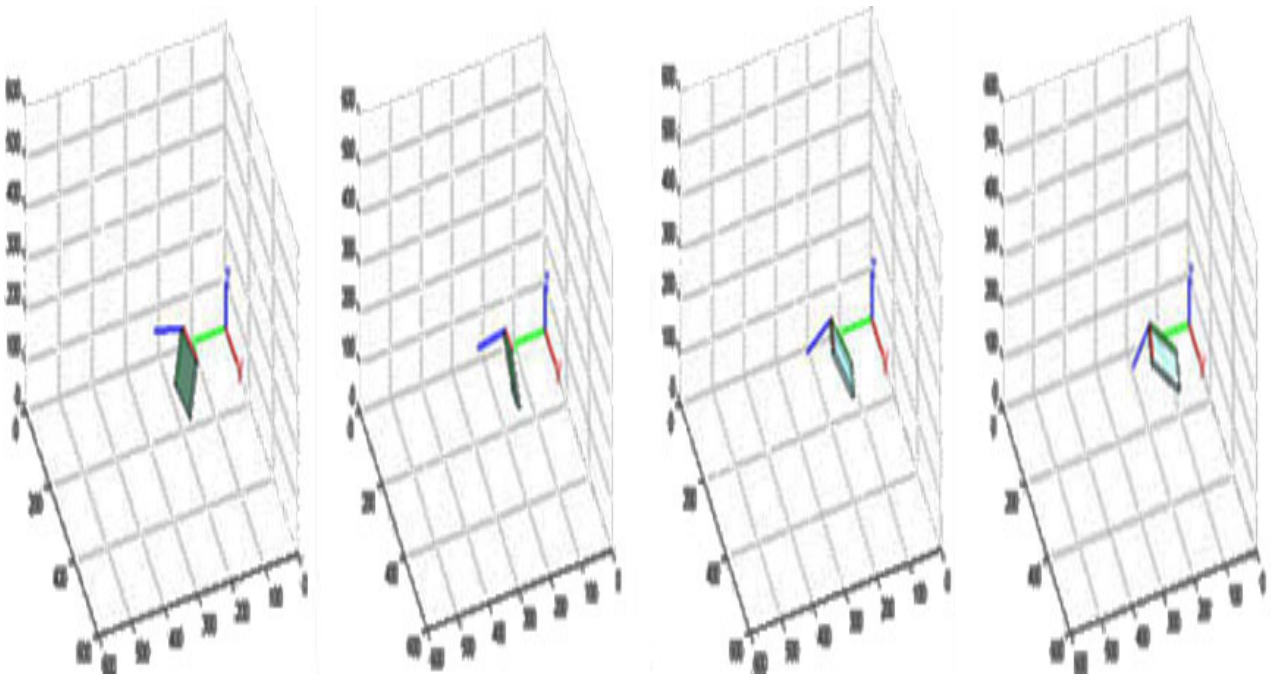
da seguinte maneira: os ângulos *Pitch* e *Yaw* são mantidos e *Roll* utiliza o valor igual a zero. A partir disso, os ângulos foram convertidos à matriz de rotação, utilizando a seguinte função predeterminada do Matlab:

$$R_{sens} = angle2dcm(pitch_{sensors}, roll_{sensors}, yaw_{sensors}, 'YXZ') \tag{55}$$

Esta matriz de rotação apresenta a orientação em relação ao sistema de coordenadas da VE. Para obter a orientação do plano do gimbal em relação ao sistema de coordenadas do mundo {W}, foi utilizada a equação (5), que representa o plano de orientação do gimbal em relação ao ambiente de experimentação {W}. Desta forma, foram obtidos os três ângulos de rotação do plano do gimbal, em relação ao ambiente de experimentação.

Em seguida, na Fig. 48, apresentam-se os resultados do movimento do plano de orientação do gimbal, em diferentes instantes de tempo.

Figura 48 – Resultados dos planos de orientação do gimbal

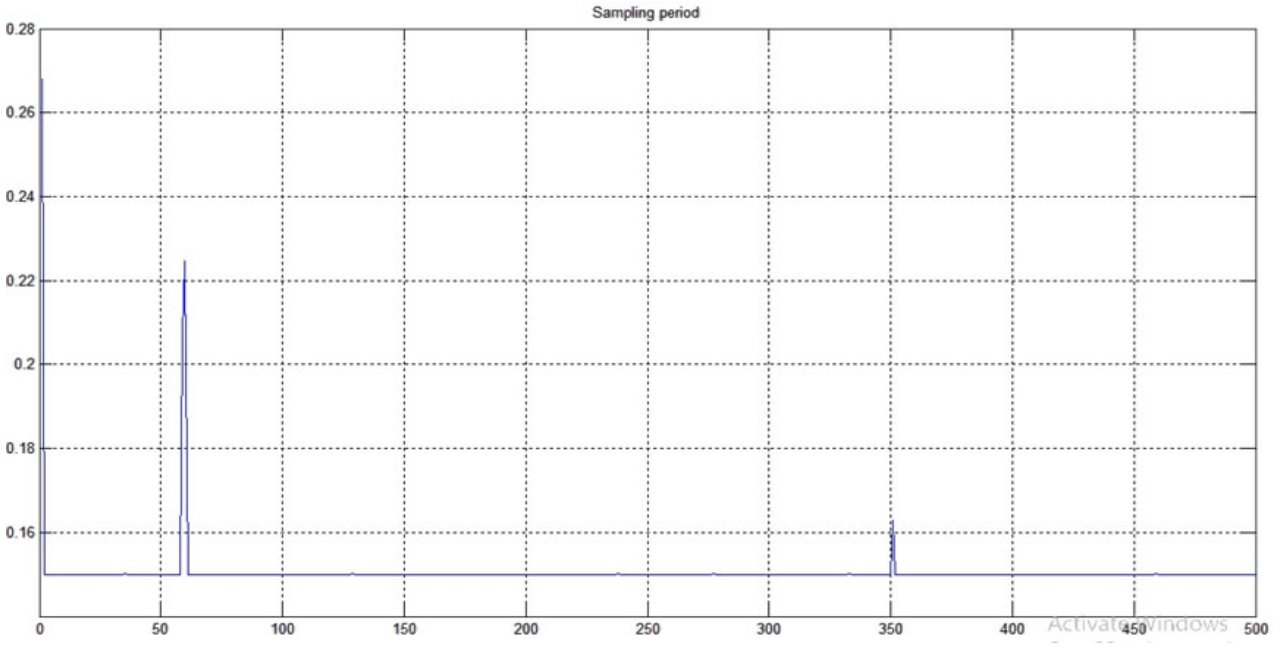


FONTE: A autora.

Nessa Fig. 48, podem ser vistos os diferentes instantes de tempo em que o plano de orientação do gimbal realiza diferentes movimentos. Esses movimentos são apresentados a partir do ambiente gráfico, que mostra os movimentos em tempo real do gimbal, obtidos a partir da matriz de rotação em relação ao ambiente de experimentação {W}.

Assim, na seguinte Fig. 49, é apresentado o período de amostragem, para cada amostra.

Figura 49 – Resultados do período de amostragem do gimbal



FONTE: A autora.

A partir dessa Fig. 49, pode ser visto o período de amostragem obtido para cada amostra, em que uma amostra é o intervalo de tempo que demora em calcular as posições angulares obtidas dos servomotores. Em relação aos resultados, é visto que, para 500 amostras, o período de amostragem é de 0.15 milissegundos.

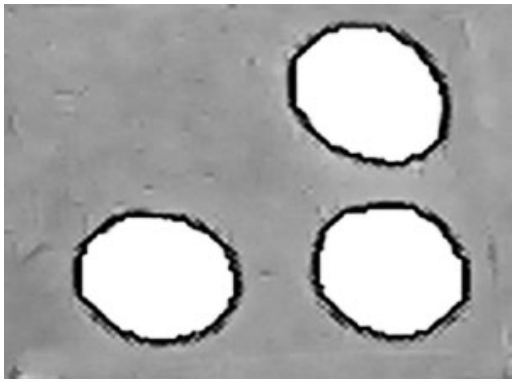
Esses dados do plano de orientação do gimbal servirão para comparar com o plano de orientação da reconstrução tridimensional dos pontos.

5.3.2 Funcionamento do plano da reconstrução dos pontos

Para calcular a orientação do plano da reconstrução dos pontos, apresentado na Fig. 50, utiliza-se o plano colocado acima do gimbal. Esse plano tem três pontos, dispostos na forma de um triângulo, com o objetivo de formar um sistema de coordenadas do plano da reconstrução.

Esse plano da reconstrução dos pontos servirá para ser comparado com o plano de orientação do gimbal calculado na seção anterior.

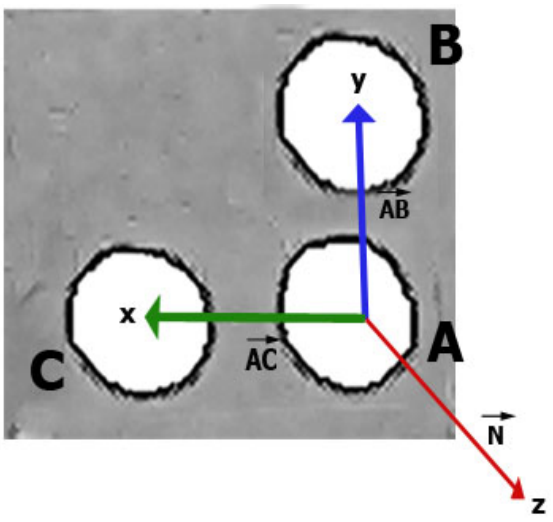
Figura 50 – Plano para a reconstrução dos pontos



FONTE: A autora.

Entretanto, para obter o plano de orientação dos pontos, utiliza-se a Fig. 51, que mostra os sistemas de coordenas formados através dos pontos, na forma de vetores. Esse plano é formado através de um modelo matemático feito por vetores, da seguinte forma: a união dos dois centros dos círculos dos pontos A, C formam o vetor $\vec{AC}$ e a união dos centros dos círculos dos pontos A, B forma o vetor $\vec{AB}$. Esses dois vetores têm seus equivalentes nos eixos de coordenadas $X = \vec{AC}$ e $Y = \vec{AB}$. Enquanto isso, o eixo de coordenadas Z será calculado, normalizando estes dois vetores $\vec{AC}$ e $\vec{AB}$, que será referido como $\vec{N}$. A seguir, apresentam-se os cálculos realizados para obter este modelo do sistema de coordenadas através de vetores:

Figura 51 – Modelo do plano da reconstrução de pontos por vetores



FONTE: A autora.

Segundo a teoria de vetores (ANEXO A), tem-se dois vetores ($\vec{AB}$ e $\vec{AC}$), definidos como:

$$\vec{AC} = A - C$$

$$\vec{AC} = (a_x, a_y, a_z) - (c_x, c_y, c_z) \tag{56}$$

E para o vetor $\vec{AB}$:

$$\vec{AB} = A - B$$

$$\vec{AB} = (a_x, a_y, a_z) - (b_x, b_y, b_z) \tag{57}$$

Sendo que um vetor unitário de acordo com a teoria de vetores está definido por:

$$\vec{A} = \frac{\vec{A_k}}{|\vec{A_k}|}$$

Entretanto,

$$\vec{A_k} = \frac{a_x, a_y, a_z}{\sqrt{a_x^2 + a_y^2 + a_z^2}} \tag{58}$$

Onde $|\vec{A_k}|$ é o modulo do vetor.

Generalizando nos vetores $\vec{AB}$ e $\vec{AC}$, tem-se:

$$\vec{AC} = \frac{ac_x, ac_y, ac_z}{\sqrt{ac_x^2 + ac_y^2 + ac_z^2}} \tag{59}$$

‘

$$\vec{AB} = \frac{ab_x, ab_y, ab_z}{\sqrt{ab_x^2 + ab_y^2 + ab_z^2}} \tag{60}$$

Assim, a partir desses dois vetores forma-se o vetor normal $\vec{N}$, definido pelo vetor cruzado entre seus componentes, sendo que o produto vetorial é sempre perpendicular ao plano $\vec{AC} \cdot \vec{AB}$.

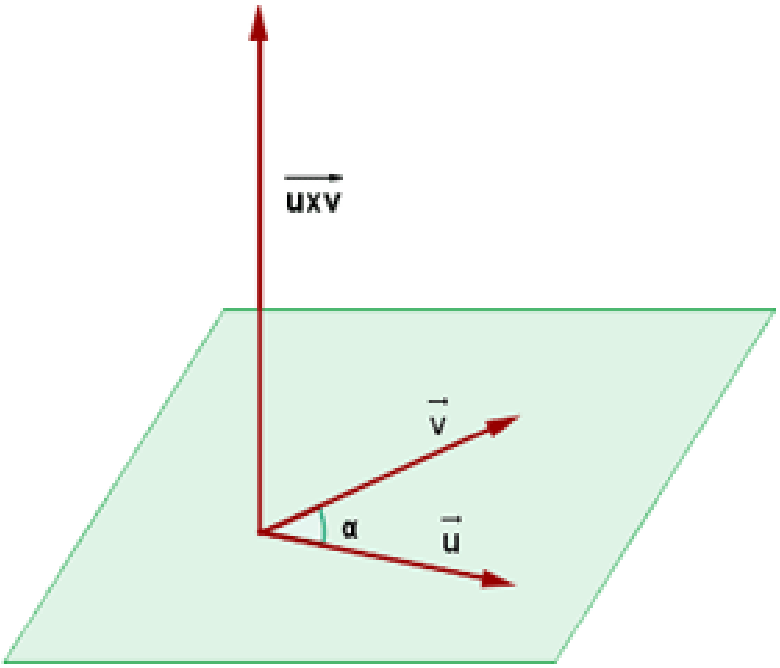
$$\vec{N} = \vec{AC} \times \vec{AB} \tag{61}$$

Formando um ângulo reto entre $\vec{AC}$ e $\vec{N}$, melhora-se o vetor $\vec{AB}$:

$$\vec{AB} = \vec{AC} \times \vec{N} \tag{62}$$

Uma vez que "×" é definido pelo produto vetorial (DECAROLI, 1976), que é representada na figura Fig. 52 junto com a sua definição.

Figura 52 – Produto vetorial



FONTE: (DECAROLI, 1976)

Definição do produto vetorial: chama-se *produto vetorial* ao produto entre os vetores $\vec{u}$ e $\vec{v}$, $\times$ é o operador do produto vetorial. Portanto, o produto vetorial de $\vec{u} = x_1\vec{i} + y_1\vec{j} + z_1\vec{k} \times \vec{v} = x_2\vec{i} + y_2\vec{j} + z_2\vec{k}$ é resolvido da seguinte forma:

$$\vec{u} \times \vec{v} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} = \begin{vmatrix} y_1 & z_1 \\ y_2 & z_2 \end{vmatrix} \vec{i} + \begin{vmatrix} z_1 & x_1 \\ z_2 & x_2 \end{vmatrix} \vec{j} + \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} \vec{z} \tag{63}$$

Por outro lado, mudando nomes às variáveis dos vetores $\vec{AB}$, $\vec{AC}$ e $\vec{N}$, segundo a Fig. 51:

$$\vec{X} = \vec{AC}$$
$$\vec{Y} = \vec{AB}$$
$$\vec{Z} = \vec{N}$$

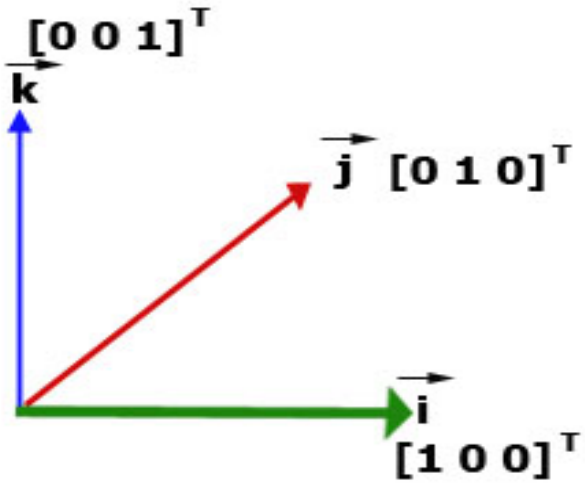
Assim, a partir dos eixos de coordenadas $\vec{X}$, $\vec{Y}$ e $\vec{Z}$ obtidos, é possível obter o sistema de coordenadas do plano de orientação da reconstrução de três pontos. A partir

disso, a matriz de rotação é formada.

$$\mathbf{R} = \begin{bmatrix} \vec{X} & \vec{Y} & \vec{Z} \end{bmatrix}$$

Com a ajuda dos vetores unitários (Ver Fig. 53),

Figura 53 – Vetores unitários



FONTE: A autora.

Associando com os vetores unitários na matriz de rotação **R**:

$$\begin{bmatrix} \vec{X} & \vec{Y} & \vec{Z} \end{bmatrix} = \mathbf{R} \cdot \begin{bmatrix} \vec{i} & \vec{j} & \vec{k} \end{bmatrix} \tag{64}$$

De maneira global:

$$\begin{bmatrix} ac_x & ac_y & ac_z \\ ab_x & ab_y & ab_z \\ n_x & n_y & n_z \end{bmatrix} = \mathbf{R} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{65}$$

$$\mathbf{R} = \begin{bmatrix} ac_x & ac_y & ac_z \\ ab_x & ab_y & ab_z \\ n_x & n_y & n_z \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T$$

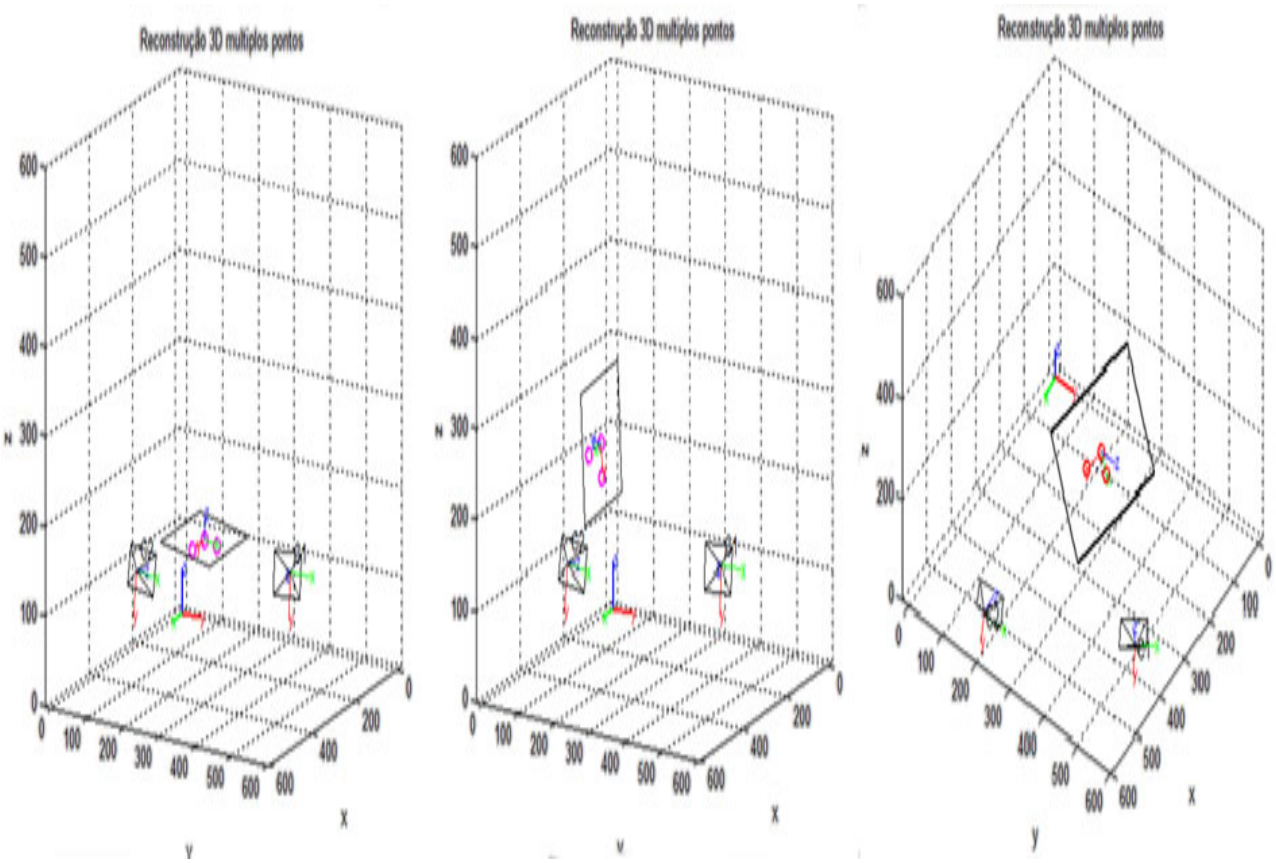
Portanto o plano de orientação da reconstrução de três pontos está dado pela matriz **R**, como mostra a seguir.

$$\mathbf{R} = \begin{bmatrix} \vec{X}\vec{i} & \vec{Y}\vec{j} & \vec{Z}\vec{k} \end{bmatrix} \tag{66}$$

Assim, a partir da matriz de orientação **R** é possível transformar em ângulos de Euler, quaternions ou vice-versa. Todas essas transformações podem ser realizadas através de funções predeterminadas do *matlab*, como, *dcm2angle(R)*, *angle2dcm* ou *dcm2quat*.

A partir da obtenção do plano de orientação dos pontos, a seguir apresentam-se os resultados na Fig. 54.

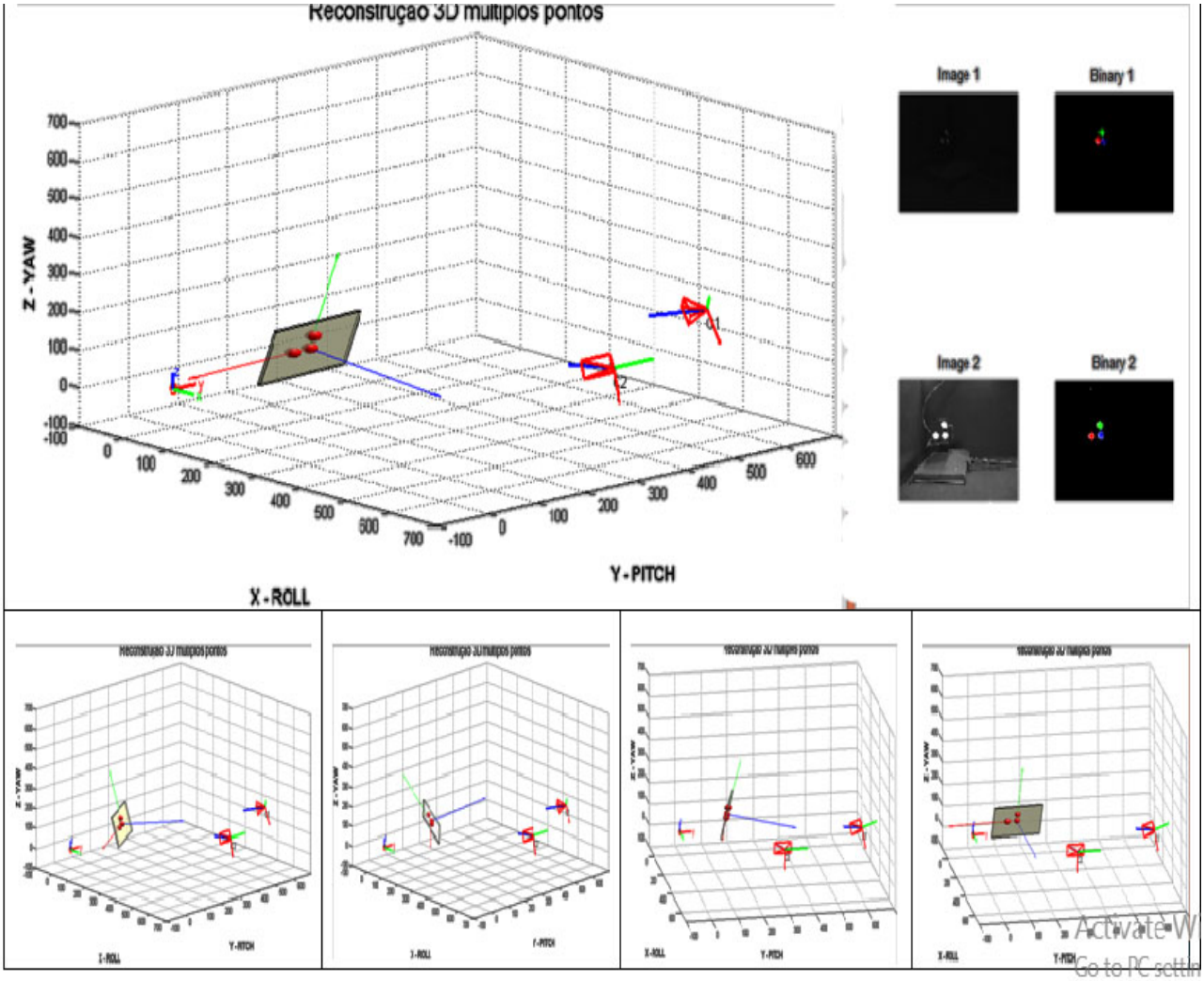
Figura 54 – Resultados do plano de orientação entre três pontos reconstruídos



FONTE: A autora.

Os resultados dessa Fig. 54 apresentam a orientação dos planos dos pontos, baseados no modelo matemático feito através de vetores, em três intervalos de tempo diferentes. A primeira imagem à esquerda mostra a orientação do plano dos pontos. Quando os três ângulos *Roll*, *Pich* e *Yaw* são igual a zero, esse plano está parado na horizontal. A segunda imagem do meio mostra a orientação do plano em que o eixo de coordenadas *Roll* é igual a 90°, e os outros eixos são iguais a zero. Nesse caso, o plano está parado no plano vertical ao eixo de coordenadas *Roll*. Enquanto a terceira imagem da direita apresenta a orientação do plano, em que *Roll* igual a $-30°$, *Pitch* igual a 30° e *Yaw* igual a 0°. Em geral a orientação plano a partir dos pontos reconstruidos é calculado corretamente

Figura 55 – Resultados da orientação do plano a partir de vários pontos - ângulo Yaw: 60°



FONTE: A autora.

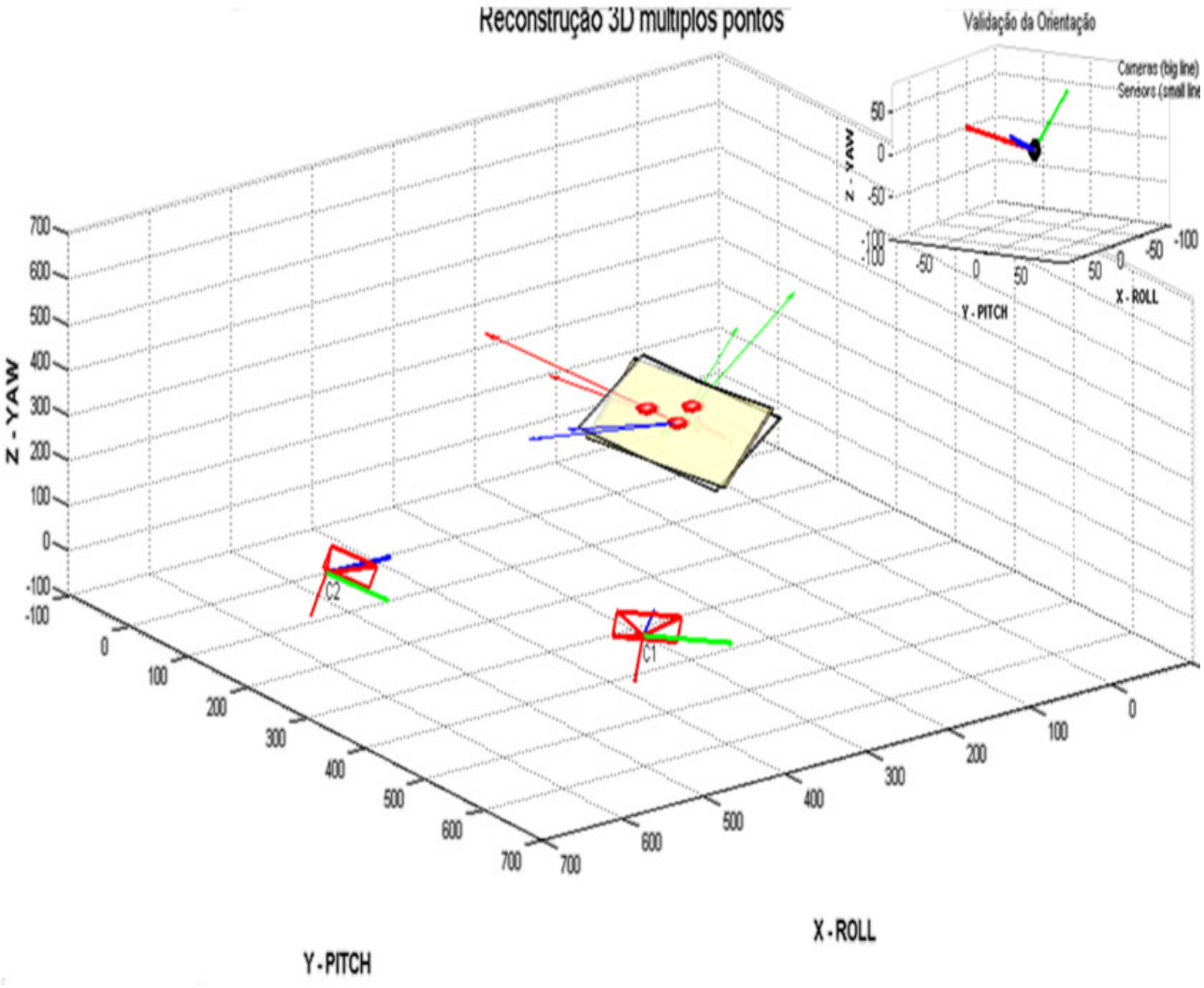
O resultado apresentado nessa Fig. 55mostra a orientação do plano reconstruído a partir de três pontos, utilizando os movimentos do gimbal no programa principal da reconstrução varios pontos. A vista da esquerda acima mostra a orientação do plano da reconstrução, quando o ângulo de *Yaw* é 60°. Na segunda vista à direita acima, observam-se as imagens capturadas pelo sistema de VE, com suas respetivas correlações entre pontos. Entretanto, a imagem de baixo mostra os movimentos do gimbal, com diferentes orientações dos planos, em intervalos de tempo diferentes. Em geral, a orientação do plano a partir da reconstrução dos pontos foi bem-sucedida, por obter uma correlação precisa entre pontos, além de obter o plano de orientação dos pontos, a partir do movimento do gimbal.

5.3.3 Comparação dos resultados entre os planos do gimbal e da reconstrução

Dados os dois resultados das orientações dos planos entre o gimbal e a reconstrução de pontos, foram colocados os dois algoritmos no programa principal da reconstrução, uma vez que o próximo passo é a comparação.

As Figs. 56, 57 e 58 mostram algumas fases da comparação entre os planos, do gimbal e da reconstrução de pontos. Nessas imagens, apresentam-se duas vistas, sendo que na vista maior estão os dois planos, do gimbal e da reconstrução, enquanto na vista pequena acima apresentam-se as orientações dos sistemas de coordenadas entre os dois planos (o sistema de coordenadas maior é da reconstrução, enquanto o sistema de coordenadas menor é do gimbal).

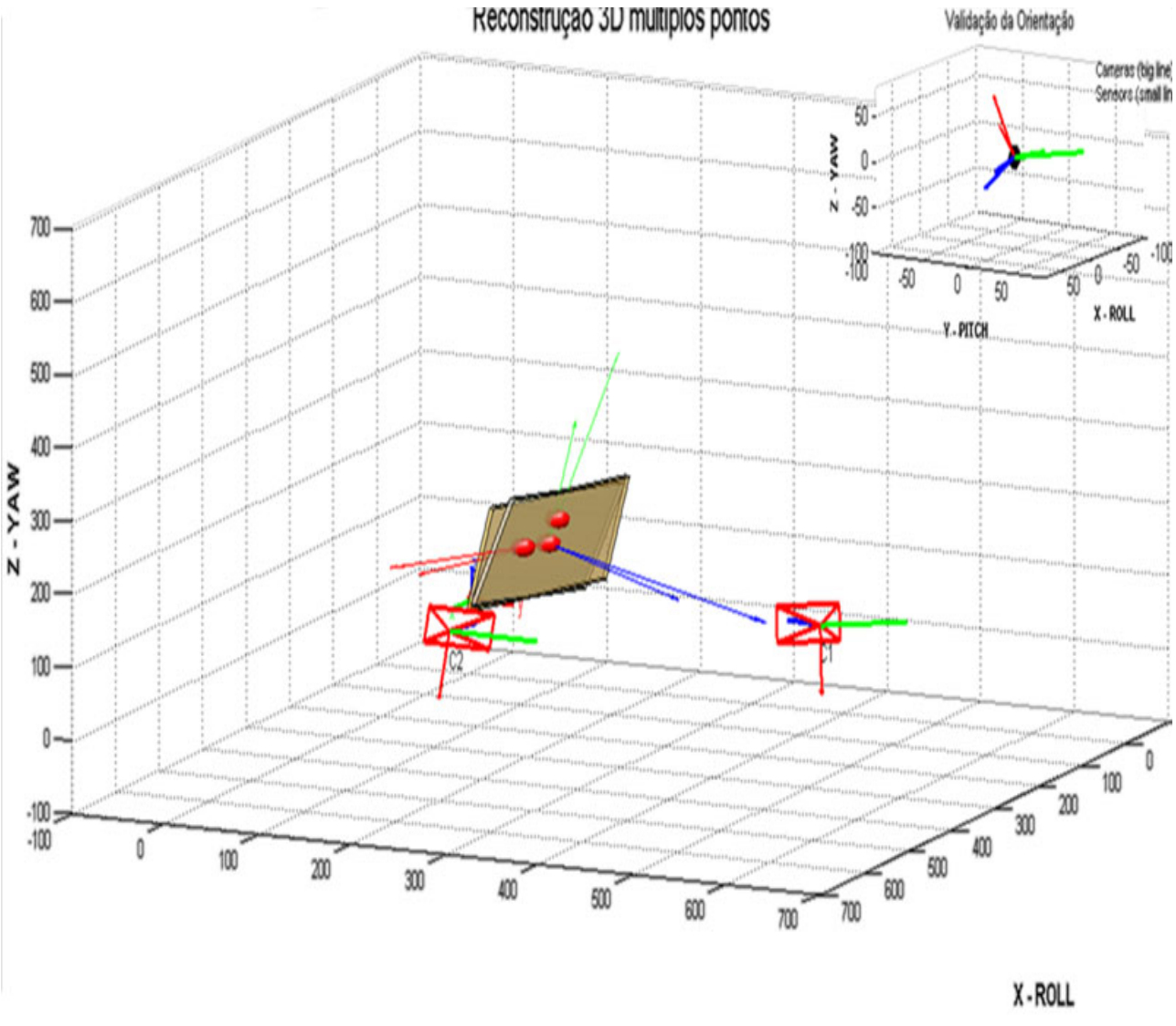
Figura 56 – Comparação entre planos de orientação dos planos do gimbal e da reconstrução de pontos – Teste 1



FONTE: A autora.

Nessa Fig. 56, os planos ficam com mesma orientação, isto é devido, a que o plano da reconstrução a partir dos pontos é igual ao plano do gimbal. Entretanto, a igualdade da orientação dos sistemas de coordenadas pode ser comprovada na imagem pequena acima à direita, onde os sistemas de coordenadas dos planos estão bem alinhados. Portanto, pode-se considerar a validação bem-sucedida, onde os pontos a partir do sistema de VE foram reconstruídos na posição certa.

Figura 57 – Comparação entre planos de orientação dos planos do gimbal e da reconstrução de pontos – Teste 2

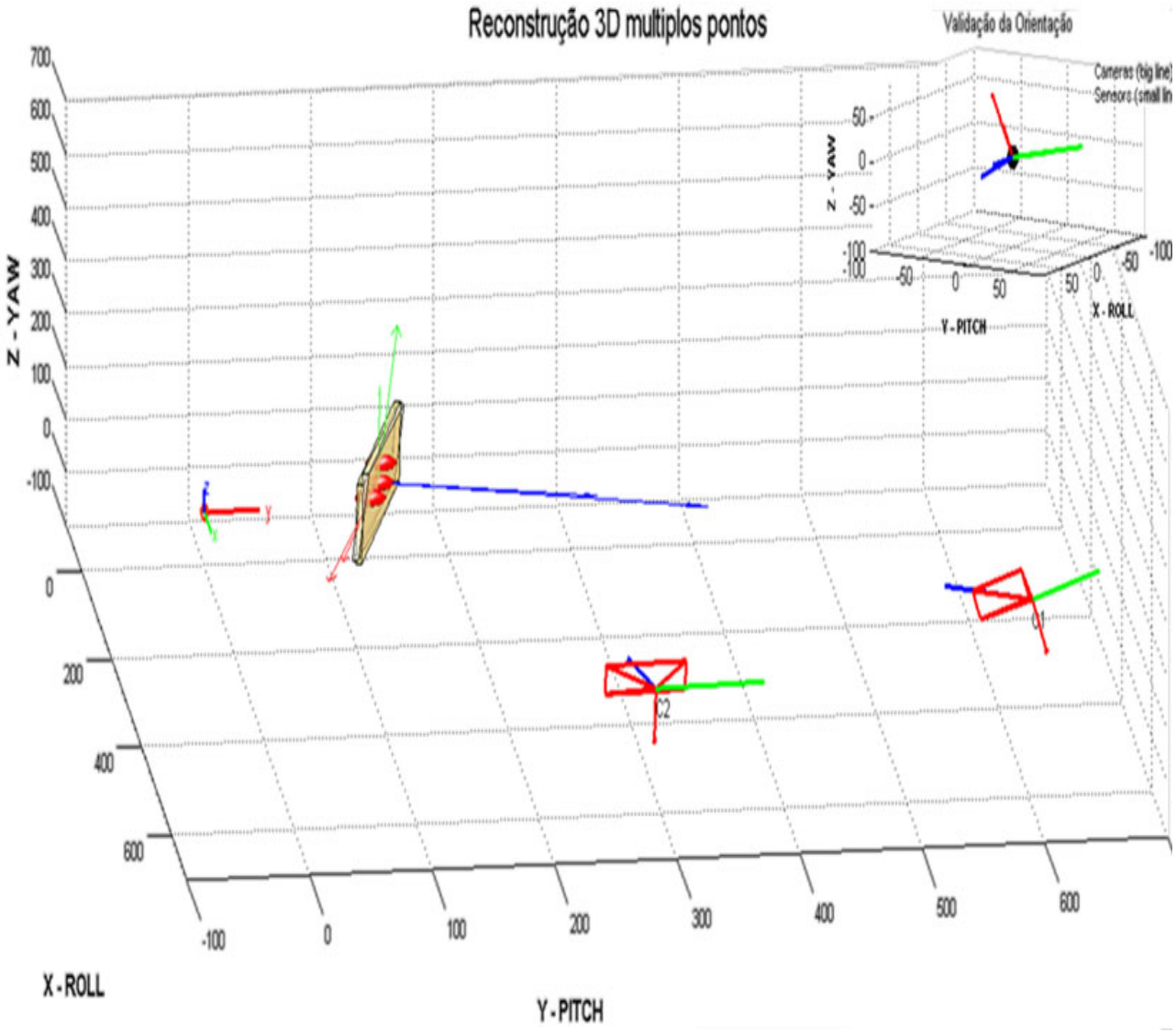


FONTE: A autora.

Para essa Fig. 57, o plano da reconstrução dos pontos teve um pequeno deslize em relação ao eixo de coordenadas Z. Nesse caso, na comparação entre os planos, existiu

uma diferença, a qual pode ter ocorrido devido a diferentes fatores: algum ponto não foi reconstruído na posição correta ou imagens mal capturadas. Apesar disso, em termos gerais, foi conseguido validar os pontos da reconstrução quase corretamente.

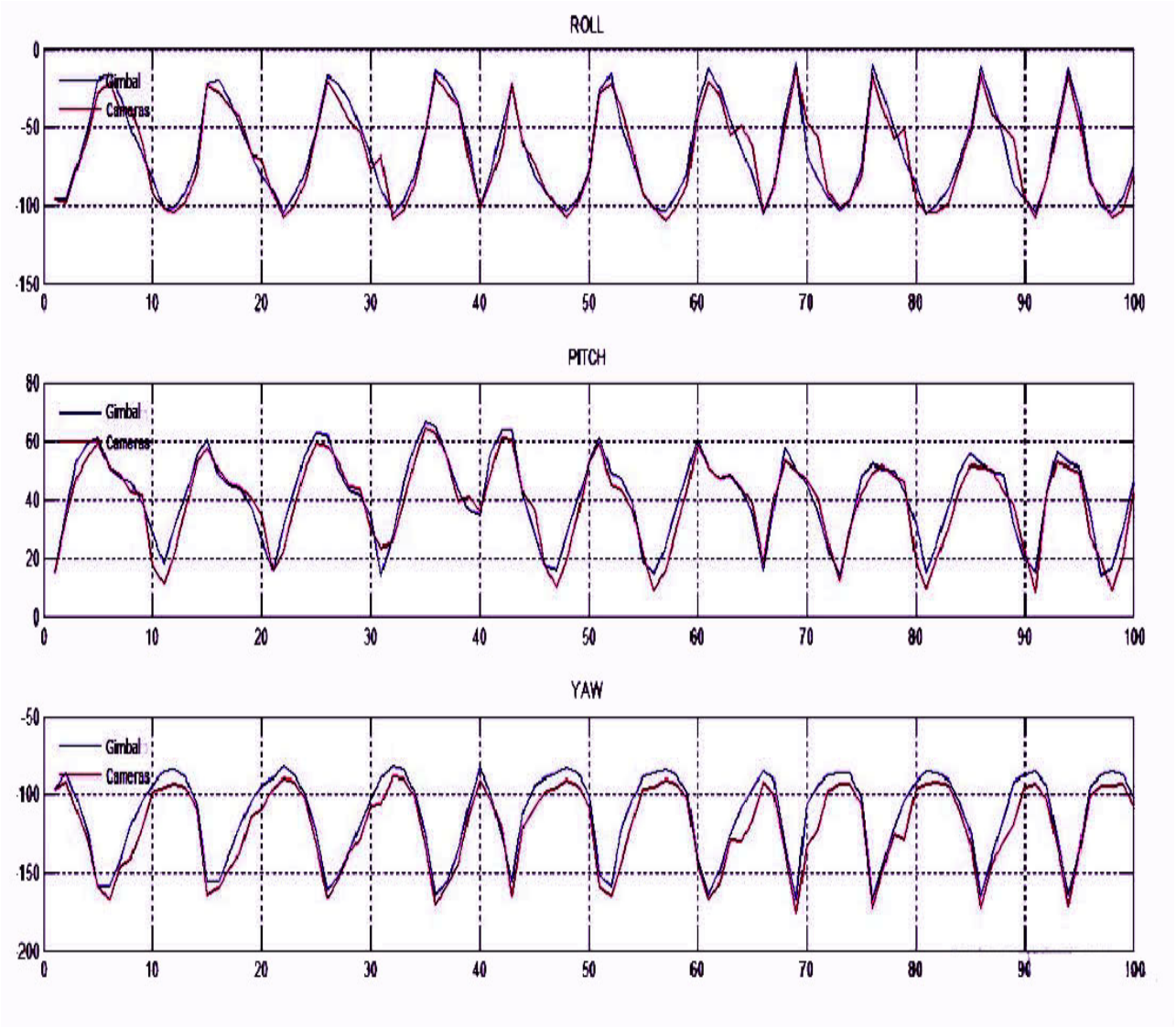
Figura 58 – Comparação entre planos de orientação dos planos do gimbal e da reconstrução de pontos – Teste 3



FONTE: A autora.

No caso dessa Fig. 58, a reconstrução foi mais precisa, uma vez que os dois planos foram alinhados. Na vista acima, pequena, pode ser comprovado o alinhamento, em que os dois sistemas de coordenadas seguem a mesma orientação. Portanto, a reconstrução e validação dos pontos é considerada bem-sucedida.

Figura 59 – Validação da reconstrução 3-D de múltiplos pontos



FONTE: A autora.

No caso dessa Fig. 59, apresenta-se uma comparação entre os planos de orientação da reconstrução de pontos e do gimbal, para um tamanho de 100 amostras. Utilizam-se as orientações dos planos para os três ângulos *Roll*, *Pitch* e *Yaw*. Todos os resultados estão organizados pelos ângulos dos movimentos do gimbal, por exemplo, no Roll, o movimento 40 tem um ângulo $-100°$.

Assim, na imagem acima observa-se o ângulo *Roll*, que varia desde $-10°$ até $-100°$. É visível que, quando os movimentos estão entre $20°$ e $30°$ ou $60°$ e $75°$, há uma variação nos ângulos. Isso significa que, na orientação desses intervalos, existem erros na reconstrução, mas nos outros movimentos a reconstrução foi bem-sucedida.

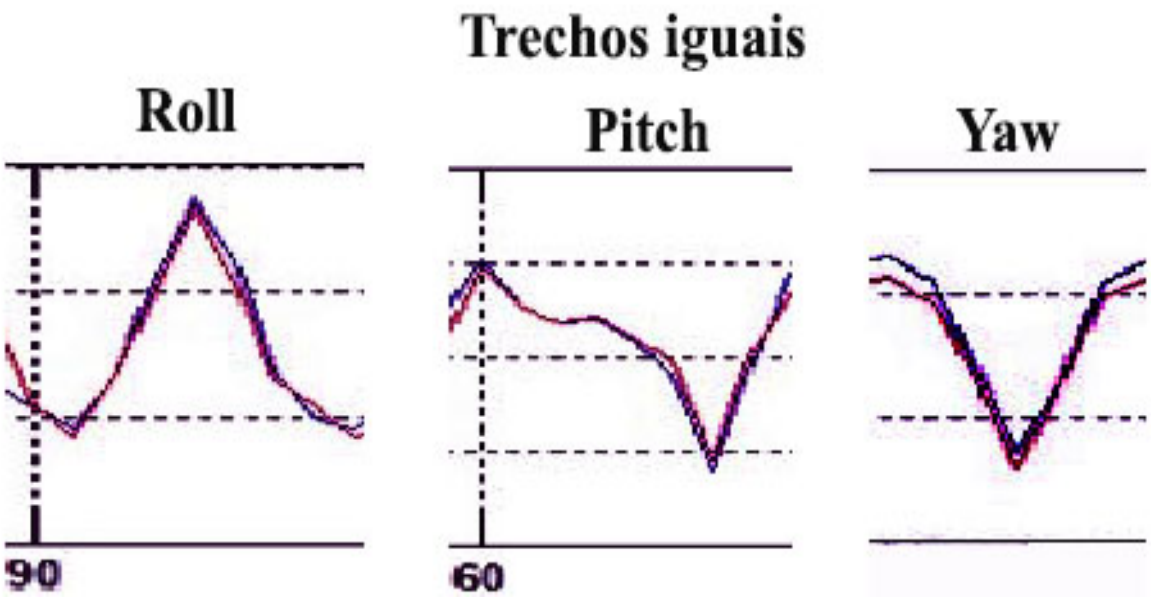
Na imagem do meio, apresenta-se o ângulo *Pitch* que varia desde 20° até 60°. Nos resultados da reconstrução, existe alguns erros entre os movimentos 10°, 30°, 55°, 80°. Assim, nos seguintes movimentos, a reconstrução foi realizada corretamente.

Na terceira imagem de baixo, apresenta-se o ângulo *Yaw*, que varia desde $-90°$ até $-160°$; neste eixo de coordenadas pode-se ver, que existe vários erros na reconstrução na maioria dos movimentos, entre o ângulo perto dos $-90°$, enquanto nos movimentos a partir de $-160°$ a validação da reconstrução dos pontos foi bem-sucedida.

Por outro lado, a seguir serão apresentados, numa análise mais detalhada, alguns trechos dos resultados. Assim, quando os planos de orientação são iguais, é porque a reconstrução foi bem-sucedida, (Fig. (60)). Mas, se os planos de orientação são diferentes, é porque existiram alguns erros na reconstrução (Fig. (61)).

Assim, os resultados para trechos iguais são apresentados a seguir na Fig. (60).

Figura 60 – Validação da reconstrução 3-D com trechos iguais

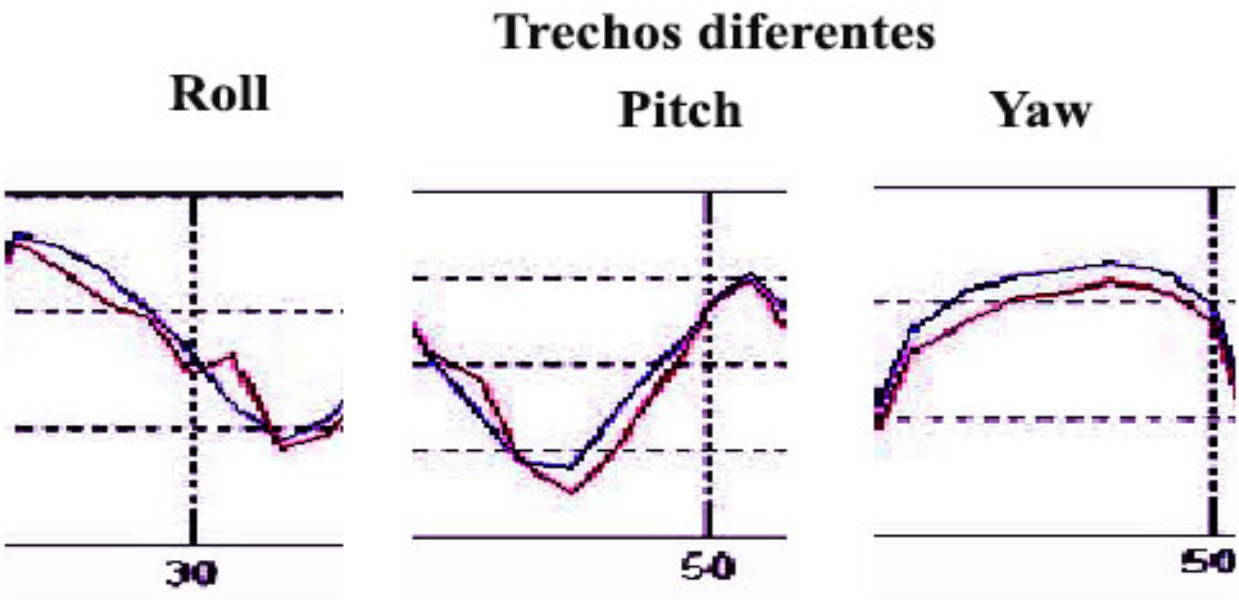


FONTE: A autora.

Nessa Fig. 60, é possível ver que alguns trechos da validação dos ângulos *Roll*, *Pitch* e *Yaw* são iguais; isso significa que as posições dos pontos reconstruídos e dos servomotores obtiveram os mesmos valores. Os planos da reconstrução de pontos e do gimbal tiveram a mesma orientação. Em conclusão, a validação da reconstrução de pontos foi bem-sucedida para esses trechos dos movimentos do gimbal.

Por outro lado, quando os trechos são diferentes, a análise dos resultados é apresentada na Fig. (61).

Figura 61 – Validação da reconstrução 3-D com trechos diferentes).



FONTE: A autora.

Na Fig. (61), pode ser visto que alguns trechos da validação do *Roll*, *Pitch* e *Yaw* possuem algumas diferenças. Isso significa que existiram alguns erros na reconstrução dos pontos, em alguns movimentos do gimbal. Esses erros podem ser causadas por ruído, alinhamento entre pontos, imagens mal capturadas, erro nos servomotores ou diferente tempo de resposta entre os servomotores e a captura das imagens. Entretanto, várias causas poderiam ter acontecido para que a reconstrução não tenha seguido uma trajetória igual aos dos servomotores. Apesar de existirem alguns erros na reconstrução, ambos os planos realizaram uma mesma trajetória na mesma direção. Portanto, em alguns intervalos de tempo, a reconstrução de pontos teve alguns erros, mas, em geral, a trajetória seguida pelos planos segue uma mesma orientação, então se teve uma reconstrução aceitável.

Em conclusão, os dois planos seguiram a mesma trajetória na mesma direção, sendo que ambos obtiveram no gráfico similares formas, apesar de existirem erros na posição de alguns pontos reconstruídos. Assim, é possível ver nas figuras, que os ângulos do gimbal seguem uma trajetória mais uniforme, sem muitas mudanças repentinas na posição. Ao contrário dos ângulos do plano da reconstrução que tem muitas mudanças repentinas de posição . Esses erros ou mudanças repentinas de posição podem ter ocorrido devido ao tempo de captura de imagens e processamento dos dados, já que os servomotores do gimbal realizam movimentos que não dependem do computador.

Sendo assim, o algoritmo de reconstrução deve realizar todas as tarefas ao mesmo tempo, do movimento do gimbal. Contudo, os resultados do algoritmo de reconstrução

foram bem-sucedidos, devido à correlação dos pontos corretamente, à posição dos pontos reconstruídos igual ao dos servomotores com pequenas variações e em razão da trajetória seguida entre os planos criados serem iguais em termos gerais.

5.4 Custo computacional do algoritmo da obtenção tridimensional de vários pontos

Neste tópico, é apresentado o custo computacional do algoritmo proposto para obter o espaço tridimensional para vários pontos

Para implementar esse algoritmo, diversas funções foram realizadas. A seguir na tabela 2, serão apresentadas as mais importantes, junto com sua descrição.

Tabela 2 – Funções principais do algoritmo da obtenção do espaço tridimensional de vários objetos

Nome de função	Descrição da função
$Main$	É a função principal desse programa. Essa função chama a todas as outras funções específicas.
$ProcessingAlgorith3D$	Obtém a profundidade 3-D dos objetos e aplica a teoria de visão estéreo.
$ProcessingGimbal$	Lê dados a partir do gimbal e obtém seu plano de orientação.
$Points2EulerR$	Cria o plano de orientação a partir dos pontos 3-D obtidos.
$CreateEnvioreme nt3D$	Cria o ambiente gráfico de todas as funções, sistemas de coordenadas, planos, linhas epipolares, projeções etc.
$AxisUpgrade$	Atualiza os sistemas de coordenadas às novas orientações em cada iteração do movimento dos objetos.
$RotatingPlane$	Cria o novo plano de orientação do novo movimento, utiliza a função $AxisUpgrade$.

FONTE: A autora.

Uma vez executado o programa, é verificado o custo computacional com as próprias ferramentas que tem o Matlab. Na tabela seguinte, mostra-se o custo computacional das funções mais importantes do programa proposto.

Tabela 3 – Custo computacional das funções principais do algoritmo da obtenção do espaço tridimensional de vários objetos

Nome de função	Númer de chamadas da função [s]	Tempo de execução de cada função [s]	Tempo total da função [s]
$AxisUpgrade$	300	12.691	38.123
$Main$	1	8.375	122.690
$RotatingPlane$	100	2.757	19.042
$CreateEnviorement3D$	1	1.681	5.976
$ProcessingAlgorith3D$	100	0.745	5.683
$ProcessingGimbal$	100	0.033	0.088
$Points2EulerR$	100	0.029	0.115

FONTE: A autora.

O custo computacional apresentado na Tabela 3, foi feito para 100 amostras. Essa tabela foi ordenada pela segunda coluna do tempo de execução de cada função. Também pode ser observado na primeira coluna o número de chamadas que a função tem no programa. A terceira e última coluna mostra o tempo de execução total da função mais as subfunções.

Quanto a uma análise mais detalhada sobre o custo computacional de cada função, é possível ver que a função "$AxisUpgrade$"ocupa o maior custo computacional no programa. Isso significa que mais tempo é gasto na criação e atualização do ambiente gráfico, sendo esse resultado consistente em razão do programa ter de atualizar o ambiente gráfico em cada movimento que o gimbal realiza. Por outro lado, a função principal "$Main$" é a segunda função que tem maior tempo de execução. Isso também é consistente, pois essa função chama a todas as outras funções e integra toda a lógica do programa. No entanto, a função da reconstrução de pontos "$ProcessingAlgorith3D$" é a terceira função que gasta menos custo computacional, podendo ser visto que para esse teste de 100 amostras foram gastos apenas 0.745 segundos em tempo computacional. Esse valor é muito pequeno comparado com 12.691 segundos que gasta a função "$AxisUpgrade$". Assim, o algoritmo de reconstrução "$ProcessingAlgorith3D$"pode ser implantado num sistema integrado com baixo custo computacional.

Portanto, em termos de custo computacional, o ambiente gráfico do programa gasta mais do que o programa da reconstrução de pontos.

6 Conclusão

A dissertação apresentada centra-se na reconstrução do espaço tridimensional de vários objetos (pontos) mediante um sistema de visão estéreo. Esse objetivo foi conseguido em duas etapas: o primeiro foi reconstruir o espaço tridimensional de um único objeto. A segunda etapa foi generalizar esse processo para vários objetos numa mesma cena. A fim de validar os resultados obtidos, um ambiente de testes foi construído junto com o gimbal em um sistema de visão estéreo.

Com o propósito de familiarizar ao leitor com o conteúdo dos capítulos sequentes, o capítulo 2 apresentou conceitos básicos usados no projeto. Esse é dividido em quatro etapas: representação de espaços euclidianos, calibração de câmeras, geometria epipolar para objetos e processamento de imagens. A etapa da representação de espaços euclidianos apresenta o espaço euclidiano em relação a um ambiente relativo e global, além de representar o movimento de um objeto num espaço euclidiano, através dos movimentos de rotação e translação. Contudo, a representação homogênea serviu para operações relacionadas entre matrizes e vetores afins. Na segunda etapa, foram expostos conceitos que usam calibração de câmeras, tal como são: a formação de imagens através de uma lente *pinhole*, modelo geométrico da formação de imagens através de uma câmera ideal, que obtêm parâmetros intrínsecos e extrínsecos. Na terceira etapa, apresentaram-se conceitos da geometria epipolar para objetos, sendo que a visão estéreo é uma parte importante do projeto. Foram tratados conceitos como a restrição epipolar, entidades geométricas epipolares, propriedades dos epipolos e linhas epipolares. Por último, analisaram-se conceitos de processamento de imagens, em que foi explicado o filtro da média, que ajudaria a melhorar as imperfeições das imagens capturadas no sistema de VE.

O Capítulo 3 foi dividido em duas partes principais: a calibração das câmeras e a reconstrução do espaço tridimensional para um objeto. A calibração de câmeras foi feita através do *"Tolbox "* de calibração feito em *Matlab*, desenvolvido pela Universidade de Caltech (BOUGUET, 2013). A seguir, são apresentados os resultados da calibração das câmeras: os parâmetros intrínsecos e extrínsecos, para a câmera 1 e, em seguida, para a câmera 2.

1. ***Resultados da calibração da câmera 1***

   - Parâmetros intrínsecos - Matriz K_1

$$K_1 = \begin{bmatrix} 684.6224 & 0 & 333.4842 \\ 0 & 686.8438 & 233.4144 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

- Parâmetros extrínsecos - Matriz g_1

$$g_1 = \begin{bmatrix} -0.7137 & 0.1917 & -0.6737 & 512.6399 \\ 0.6976 & 0.1071 & -0.7085 & 612.1398 \\ -0.0636 & -0.9756 & -0.2101 & 200.0854 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

2. ***Resultados da calibração da câmera 2***

- Parâmetros intrínsecos - Matriz K_2

$$K_2 = \begin{bmatrix} 689.3351 & 0 & 325.6238 \\ 0 & 690.4951 & 240.6365 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

- Parâmetros extrínsecos - Matriz g_2

$$g_2 = \begin{bmatrix} -0.1230 & 0.1853 & -0.9750 & 665.0527 \\ 0.9905 & -0.0372 & -0.1320 & 286.4502 \\ -0.0607 & -0.9820 & -0.1790 & 192.3744 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Os resultados da calibração das câmeras apresentados, como são os parâmetros intrínsecos e extrínsecos, foram muito importantes e necessários na utilização dos algoritmos de reconstrução.

A segunda parte do capítulo 3 é a reconstrução do espaço tridimensional para um objeto. Para cumprir com esse objetivo, foi utilizada a seguinte relação geral que compõe o sistema de VE:

$$\lambda_1 \boldsymbol{p}_{c1} = R\lambda_2 \boldsymbol{p}_{c2} + T$$

A partir dessa fórmula, e por meio de operações algébricas e matriciais, veio a se obter a profundidade de um objeto, a partir de um sistema de VE, apresentada a seguir:

$$\lambda = (M' \cdot M)^{-1} \cdot M' \cdot T$$

Por meio dessa equação, foi obtida a posição tridimensional de um objeto num sistema de VE. Para isso, foi gerado o diagrama de passos, na Fig. 29, o algoritmo 1 e o fluxograma da Fig. 30, dos passos seguidos para reconstruir um objeto.

A partir da formalização da reconstrução de um único objeto, no capítulo 4 foi generalizado o modelo matemático, para a reconstrução de vários objetos num ambiente de VE. Os passos para atingir esse objetivo foram cinco: melhoramento de imagens, número

de objetos no ambiente, correlação de objetos entre duas imagens, modelo matemático da obtenção da profundidade de vários objetos e o algoritmo. Nesse contexto, para o primeiro passo, que é melhorar as imagens e reduzir o ruído, utilizou-se o filtro da média com um tamanho de kernel de 3×3. Isso foi conseguido através da função predeterminada do *Matlab* "$Imagem_{filt}$=*filt2(imagem)*". Para o segundo passo, que é obter o número de objetos a serem tomados em conta, no ambiente do sistema de VE, foram obtidos através do algoritmo 2 e o algoritmo 3. Entretanto, para o terceiro passo, a correlação entre objetos num sistema de VE foi conseguida por meio da restrição epipolar, dada pela equação (36), apresentado a seguir:

$$\boldsymbol{x}_2{}^T E \boldsymbol{x}_1 = 0$$

Essa restrição ajudou a escolher os objetos a serem correlacionados. O critério utilizado foi escolher as menores restrições epipolares entre pontos. Essa escolha das menores restrições ajudou a conhecer os pontos são mais prováveis a serem correlacionados, entre duas imagens. Para ajudar na correlação, foram utilizadas as propriedades epipolares e um algoritmo de controle de pixels dos pontos entre as imagens. O seguinte passo foi a criação do modelo matemático da reconstrução tridimensional de vários objetos. Isso foi conseguido através da generalização do modelo matemático da obtenção tridimensional de um objeto, conseguido pela equação (54), apresentada a seguir:

$$\lambda^i = (M'_p \cdot M_p)^{-1} \cdot M'_p \cdot T$$

Desde que $i = 1, 2, ...n$ *objetos*

E o último passo, para a obtenção da profundidade de vários objetos, foi através da criação do algoritmo 4 e do fluxograma da Fig. 35.Esse algoritmo foi implementado no *Matlab* e testado no ambiente de experimentação da VE. Contudo, conseguiu-se obter a profundidade de vários objetos entre de duas imagens do sistema de VE.

No Capitulo 5, analisa-se que, após a implementação dos algoritmos da reconstrução, o próximo passo é a obtenção de resultados experimentais. Os resultados obtidos foram organizados em quatro passos: resultados da reconstrução da profundidade, para um objeto, para vários objetos, validação dos resultados e custo computacional dos algoritmos.

No primeiro passo, de resultados da obtenção da profundidade de um objeto, os testes foram feitos com a ajuda de uma bola de isopor e diferentes planos geométricos como quadrados ou círculos colocados no ambiente de experimentação. O parâmetro utilizado para determinar se os resultados atingidos tiveram sucesso ou não é por meio da comparação entre a o ambiente de experimentação e o ambiente gráfico. A conclusão dos resultados é apresentada a seguir:

1. **Resultados do teste 1 – trajeto do sistema de coordenadas:** com uma boa calibração de câmeras, conseguiu-se fazer a trajetória dos três eixos de coordenadas utilizando uma bola de isopor, o que pode ser visto na Fig. 36. Em conclusão no teste da trajetória, existiram algumas irregularidades na reconstrução entre os eixos de coordenadas x, y, mas pelo trajeto e a forma é considerada a reconstrução bem-sucedida.

2. **Resultados do teste 2 - desenho la letra "X":** esse teste foi feito à mão livre e está visível na Fig. 37, em que foi verificado que o trajeto realizado no ambiente gráfico é igual ao do ambiente de experimentação. Mas também, pode ser visto que existem algumas posições mal reconstruídas, perto da câmera à esquerda, causadas por ruídos externos no ambiente. Apesar disso, essa reconstrução é considerada bem-sucedida.

3. **Resultados do teste 3 - desenho de um circulo:** esse teste foi feito através do uso de uma bola de isopor e uma forma geométrica de um círculo colocada na base do ambiente do experimento. Enquanto, aos resultados obtidos na Fig. 38, o círculo reconstruído do ambiente gráfico é muito semelhante à forma geométrica do ambiente de experimentação, apesar de ter algumas irregularidades nos trajetos, isso foi devido ao atrito no trajeto, entre a bola de isopor e o ambiente de experimentação. Contudo, pela forma obtida, a reconstrução é considerada bem-sucedida.

4. **Resultados do teste 4 - desenho de dois quadrados:** : esse teste foi realizado com dois quadrados, um na base e o outro numa das paredes do ambiente de experimentação. Em relação aos resultados da Fig. 39, a forma obtida é igual ao ambiente gráfico comparado com o ambiente de experimentação. Portanto, a reconstrução é considerada bem-sucedida.

5. **Resultados do teste 5 - desenho da figura parecida a um diamante:** o resultado desse teste da Fig. 40 foi feito com um plano de um quadrado e uma bola de isopor. Em relação aos resultados, as duas formas do ambiente de experimentação e o ambiente gráfico são iguais, assim a reconstrução foi bem-sucedida.

Concluindo, a comparação das formas realizadas no ambiente de experimentação e o ambiente gráfico produziu bons resultados, pois as imagens reconstruídas foram exatamente iguais com as trajetórias buscadas.

O segundo passo deste capitulo é a reconstrução de vários objetos, através da generalização do algoritmo de um objeto. Esses testes foram realizados, com a ajuda de bolas de isopor e um plano com fundo preto com marcadores brancos. Os resultados dos testes realizados no ambiente são:

1. **Resultados da reconstrução de dois pontos:** esse teste foi realizado com duas bolas de isopor, dispostas em ordem diferente. Na correlação, pode-se ver que os dois pontos da primeira imagem pertencem, aos outros dois pontos da segunda imagem, visto na Fig. 41. A partir desta correlação, conseguiu-se reconstituir a posição tridimensional das duas imagens com sucesso.

2. **Resultados da reconstrução de três pontos:** nesse teste da Fig. 42, foi utilizado um fundo preto com três marcas brancas pintadas, dispostas em forma de triângulo. A correlação estéreo entre pontos foi realizada corretamente, verificado, através das cores dos asteriscos obtidos em cada ponto das imagens. Portanto, pode-se concluir que a reconstrução dos três objetos foi bem-sucedida.

3. **Resultados da reconstrução de doze pontos:** os resultados da Fig. 43 apresentam uma correlação parcial entre pontos, devido a que, só alguns pontos correlacionaram adequadamente enquanto outros não. Apesar dos erros de correlação, nos resultados da reconstrução do ambiente gráfico, obteve-se quase a mesma forma do que o ambiente de experimentação, portanto considera-se que existiu uma boa reconstrução.

4. **Resultados da reconstrução de vários pontos:** os resultados são apresentados na Fig. 44, em que na correlação entre objetos só alguns pontos não correlacionaram corretamente, entretanto os outros sim, ocorrendo, dessa forma, uma reconstrução parcial. Mas, apesar dos resultados da correlação parcial de pontos, as figuras formadas da reconstrução no ambiente gráfico e no ambiente de experimentação são similares. Assim, pode-se dizer que existiu uma boa reconstrução.

Em geral, a reconstrução foi bem-sucedida para quantidades menores de objetos, já que, quanto maior for a quantidade de objetos e menor a distância entre eles, existem mais erros por alinhamento. Mas, apesar disso, comparando os testes do ambiente gráfico com o ambiente de experimentação, ambos obtiveram formas iguais. Portanto, pode-se dizer que existiu uma boa reconstrução para vários objetos.

O último passo é a validação dos resultados. Para a validação, pensou-se na construção do gimbal. O processo de validação envolve a comparação dos dois planos de orientação: o primeiro plano é do gimbal (servomotores do gimbal) e o outro plano vem da reconstrução. Os resultados desse processo são apresentados a seguir:

1. **Resultados do plano de orientação do gimbal:** o gimbal apresentado na Fig. 46 foi construído com dois servomotores para calcular os ângulos dos eixos de coordenadas, um para o *Yaw* e o outro para o *Pitch*, enquanto o *Roll* foi deduzido baseado nos outros dois eixos de coordenadas. A partir dos ângulos dos servomotores, calculou-se o plano de orientação do gimbal, em relação ao sistema de coordenadas das câmeras,

em que se obteve a matriz de rotação e os ângulos de Euler. Depois de obter o plano de orientação do gimbal, esse foi levado em relação ao sistema de coordenadas do ambiente de experimentação {W} com a equação (5). Outro ponto a considerar é a visão das câmeras com o gimbal. Para melhorar a visão das câmeras a todos os objetos, o gimbal foi calibrado reduzindo a velocidade e o ângulo de rotação. A partir dessa calibração, conseguiu-se obter todos os movimentos dos objetos entre as duas câmeras. Os resultados dos ângulos formados do plano de orientação do gimbal são apresentados na Fig. 47 e na Fig. 48, além de apresentar o período de amostragem da Fig. 49.

2. **Resultados do plano de orientação da reconstrução dos objetos:** foi realizada a formação de um sistema de coordenadas baseado em três objetos, como mostra a Fig. 51. A formação desse sistema de coordenadas realizou-se através de um modelo matemático criado com a ajuda da teoria dos vetores entre os pontos . A partir desse modelo matemático, obteve-se a matriz de rotação da equação (66), que representa a orientação do plano da reconstrução dos três pontos. Assim, foram obtidos a matriz de rotação e os ângulos de Euler. Os resultados do plano de orientação da reconstrução de três objetos apresentam-se nas Figs. 54 e 55, os quais apresentaram bons resultados do movimento em tempo real.

3. **Comparação dos resultados entre os planos do gimbal e da reconstrução:** uma vez construídos os dois planos de orientações, o próximo passo é comparar os resultados, e, para isso, juntaram-se os dois algoritmos dos planos de orientações obtidos, do gimbal e da reconstrução. Os resultados da comparação entre os planos são mostrados nas Figs. 56, 57 e 58, nas quais é possível ver que em diferentes movimentos os planos ficam bem alinhados, enquanto em outros casos há algumas variações. Por outro lado, a comparação entre planos de orientação também foi apresentada mediante ângulos de Euler na Fig. 59. Nela, ambos obtiveram resultados muito similares, comparando-se a forma e a trajetória. Numa análise mais detalhada nos ângulos de Euler, verificou-se, na Fig. 60, que em diferentes movimentos os planos ficaram muito alinhados, mostrando uma reconstrução satisfatória. Entretanto, na Fig. 61, foi visto que existiram algumas variações em diferentes movimentos da reconstrução. Em conclusão, os dois planos de orientação tiveram formas muito similares, portanto a validação da reconstrução de três objetos no ambiente de VE foi bem-sucedida.

Já no último passo do Capitulo 5, da obtenção do custo computacional do algoritmo de reconstrução de vários objetos. Na Tabela 2, foram apresentadas todas as funções mais relevantes do programa. Para saber o custo computacional dessas funções, o teste foi feito para 100 amostras, e os resultados foram apresentados na Tabela 3, em que a

função "*AxisUpgrade*" ocupa o maior custo computacional no programa. Isso significa que mais tempo é gasto na atualização dos gráficos do sistema de coordenadas. Por outro lado, a função de reconstrução de pontos "*ProcessingAlgorith3D*" é a terceira função que gasta menos custo computacional. Portanto, no programa proposto, a função de criação e atualização do ambiente gráfico tem maior custo computacional do que a função da reconstrução de pontos.

Assim, como conclusão final do projeto, conseguiu-se apresentar um algoritmo da obtenção da profundidade de vários pontos, além de trazer a possibilidade de saber a orientação de um plano, a partir da reconstrução de três pontos, utilizando-se somente sensores de visão, como é o sistema de VE. Este projeto pode ser aplicado em diferentes robôs manipuladores, como são os seriais ou paralelos, mais precisamente o robô delta, que é a motivação do projeto. Mediante o programa, será possível saber a orientação do manipulador só com sensores de visão. Entretanto, o programa poderia ajudar em diferentes aplicativos que precisarem saber a orientação através de um sistema de VE.

Dessa forma, sugeri-se para trabalhos futuros: realizar uma lei de controle para ser aplicado no robô delta a partir da servovisão e aplicar o projeto em ambientes não controlados. Por fim, este projeto ajuda a obter a reconstrução de vários pontos tridimensionais, de modo que, pode ser aplicado e melhorado para outros fins.

Apêndices

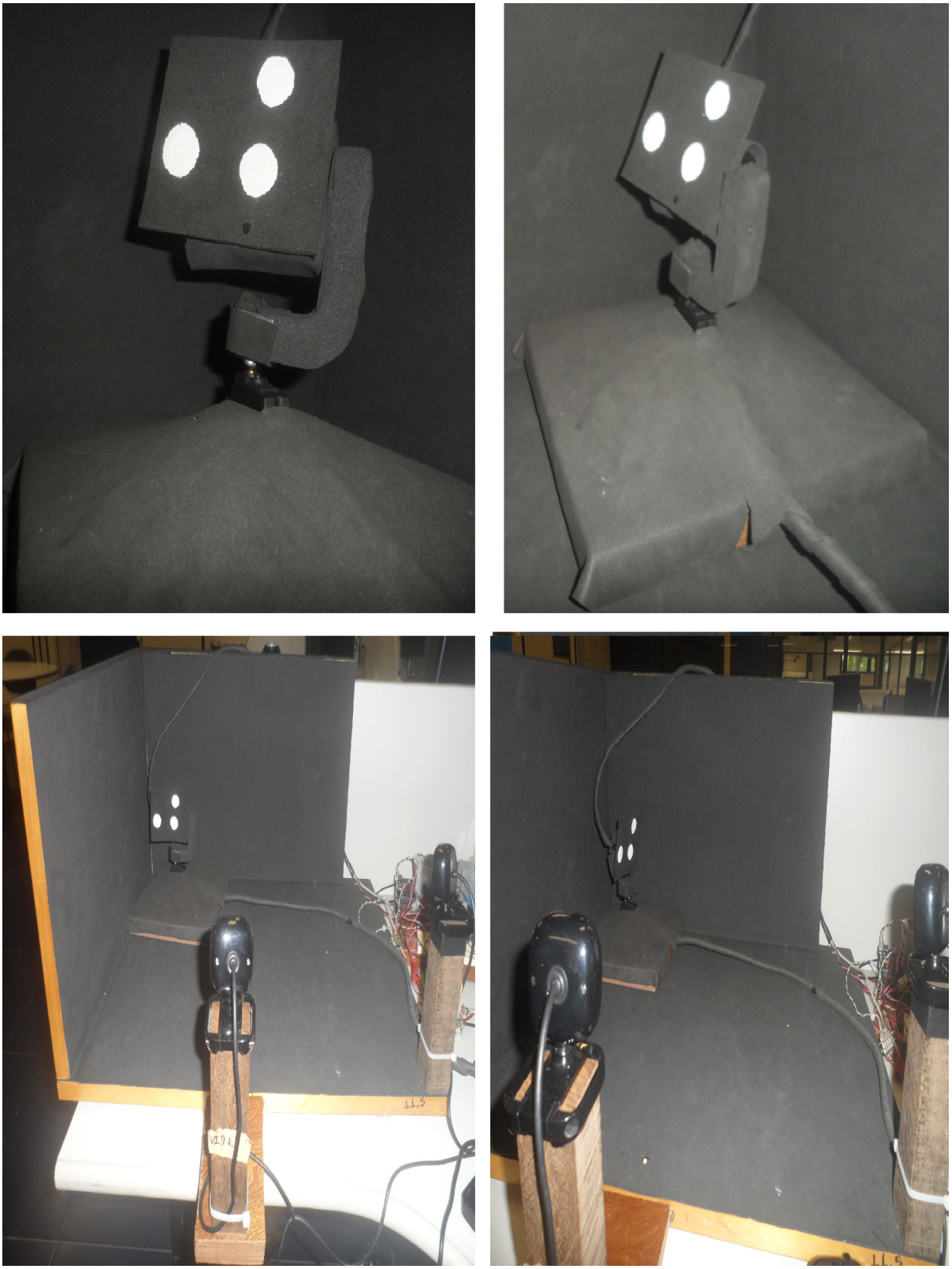
APÊNDICE A – Construção e funcionamento do Gimbal

O gimbal (Ver Fig. 62), esta construido com dois servo-motores, para os eixos Y e Z ou *Pitch* e *Yaw* respetivamente, sendo que o eixo X ou *Roll* será calculado baseado nos resultados dos eixos anteriores. Os servo-motores mencionados têm o propósito de enviar as posições angulares em cada movimento que realize o gimbal, cada servo-motor esta conectado numa *"ponte H"*, com o propósito de controlar o giro do servo-motor em ambos sentidos. Estes componentes dos servo-motores e pontes H, estão conectados numa eletrónica controlada por um programa armazenado numa placa de *arduino DUE*, com uma tensão de entrada de 12 Volts.

Eletrónica da construção do Gimbal

Depois da construção do hardware e o funcionamento da eletronica do gimbal, tem-se dois programas para controlar os dados enviados e recebidos dos sensores: Um programa que está construido no *Arduino*, tem dois controles para o *Pitch* e o *Yaw*, estes controles servem para aumentar ou diminuir a velocidade; todo este programa é guardado na placa do *Arduino* para depois ser utilizado pelo segundo programa; contudo a parte mais importante deste programa é o controle que é traduzido como a *"calibração do gimbal"* sobre a velocidade e os limites do giro dos servos (em graus). O segundo programa feito em *Matlab* recebe os dados direto da placa do *arduino* dos sensores(servo-motores), em cada movimento que o gimbal realiza; neste programa pode-se ver de modo mais gráfica o movimento do gimbal, também pode-se mudar as orientações dos planos baseados nos eixos(Y, Z), além de se calcular o terceiro eixo X, a partir disto pode-se transformar em ângulos de *Euler*, *DCM* ou *quaternions*, mediante funções do *Matlab*.

Figura 62 – Funcionamento e construção do gimbal



FONTE: A autora

APÊNDICE B – Implementação dos algoritmos de reconstrução 3-D

B.1 Implementação do algoritmo de reconstrução 3D - Um ponto

```
%% 19/03/2014 Attempt to creat a 3D visualization of a point and two
    cameras

clc
clear all
close all

ttotal=50;

rastro = 10;

% cameras' intrisic parameters
load K1
K1 = KK;
load K2
K2 = KK;

%% origin
figure(1)
clf
axis3plot([0 0 0]',[0 0 0]',' ',2);
view([130 20])
m = 600;
axis([ 20 m 20 m 0 400])
ylabel('y')
xlabel('x')
zlabel('z')
grid on
title('Reconst 3D')

%% Camera 1
load g1
cameraPlot(g1,K1)

%% Camera 2
load g2
cameraPlot(g2,K2)

```

```
%% Compute the relative pose between camera one and camera two
g12 = inv(g1)*g2;
T12 = g12(1:3,4);
R12 = g12(1:3,1:3);

%% Inicializar cameras
cam_config1;

%% Main
disp('Executando o programa principal...')

k=0;

tic
while toc < ttotal
    k = k+1;

    trigger(vid1);
    img1 = getdata(vid1); % get image from camera
    x1 = find_pixel(img1);% find object
    trigger(vid2);
    img2 = getdata(vid2); % get image from camera
    x2 = find_pixel(img2);% find object

    if isempty(x2)|isempty(x1)

    else
    % compute the orthonormal point X with respect to cameras [in mm]
    X1 = inv(K1)*[x1 1]';
    X2 = inv(K2)*[x2 1]';

    % From features and known relative pose, we can extract position
    M = [X1 R12*X2];
    Lambda = inv(M'*M)*M'*T12;
    XX1 = g1*[X1*Lambda(1); 1];
    XX2 = g2*[X2*Lambda(2); 1];
    XX(k,:) = (XX2);% + XX2)/2;

    % Plot the point
    k_init = k - rastro;

    if k_init < 1
        k_init = 1;
    end


    figure(1)
```

```
85
86      plot3(XX(k_init:k,1)    ,XX(k_init:k,2)    ,XX(k_init:k,3)    ,'ro')
87
88
89      if 0
90      figure(2)
91      subplot(1,2,1)
92      imshow(img1)
93      hold on
94      plot(x1(1),x1(2),'+')
95      hold off
96      subplot(1,2,2)
97      imshow(img2)
98      hold on
99      plot(x2(1),x2(2),'+')
100     hold off
101     end
102     end
103
104 end
105
106 %%
107 disp('Desligando as cameras...');
108 stop(vid1);
109 delete(vid1);
110 pause(1);
111 disp('Camera 01 deligada com sucesso!')
112 stop(vid2);
113 delete(vid2);
114 pause(1);
115 disp('Camera 02 deligada com sucesso!')
```

B.2 Implementação do algoritmo de reconstrução 3D - Múltiplos pontos

O programa principal é "main.m", apresentado a seguir:

```
1
2 %% Tridimensional reconstruction of multiples objects
3
4 clc,
5 clear all,
6 close all,
7 try
8 %% Intrinsinc parameters
9 load K1
```

```
10 K1 = KK;
11 load K2
12 K2 = KK;
13 %% Extrinsec parameters
14 load g1
15 g1_ = g1;
16 load g2
17 g2_ = g2;
18
19 %% Camera settings and gimbal configuration
20 disp('Initializing the cameras and the gimbal...')
21 cam_config1;
22
23 arduino=serial('COM21','BaudRate',115200);
24 flushinput(arduino);
25 fopen(arduino);
26 pause(2);
27
28 inicializa = 250;
29 me_envia   = 251;
30 finaliza   = 253;
31
32 %% variales inicializations
33 k = 1;
34 k_max = 50;
35 y = zeros( 2 , k_max);
36 Ts = zeros(1, k_max);
37 Ts_limit = 0.015;
38 R = eye(3);
39 R_sens = eye(3);
40 ypr_points_sens = zeros( 6 , k_max);
41 roll_in = zeros(1,k_max);
42 XX = zeros(3,3);
43 cont_oref = 0;
44 % reference point
45 A = [0 0 0];
46
47 %% Set environment to plot
48 onPlanePoints = 1;
49 onPlaneSensors = 0;
50 onPlotEpipolar = 0;
51 onAxisValidation = 1;
52 onPlotImageBinary = 0;
53 onPlotRPYSensorCam = 0;
54 onShowMaxEnvironment = 1;
55
56 %% Plot environment and data
```

```
figure(1)
hold on
CreateEnviorement3D;

figure(1)
hold on
ha1 = findobj('Type','axes','Tag','Axes1_main');
axes(ha1);
% plot_points =  plot3(XX(:,1), XX(:,2), XX(:,3), 'O', 'Linewidth',2, '
    color', [1 0.2 0]);hold on
[x_esph,y_esph,z_esph] = sphere(8); r = 13;
plot_sphera_1 = surf(r * x_esph + XX(1,1), r * x_esph + XX(1,2), r * x_esph
    + XX(1,3),'EdgeColor', 'none', 'FaceLighting', 'phong' );
plot_sphera_2 = surf(r * x_esph + XX(2,1), r * x_esph + XX(2,2), r * x_esph
    + XX(2,3), 'EdgeColor', 'none', 'FaceLighting', 'phong');
plot_sphera_3 = surf(r * x_esph + XX(3,1), r * x_esph + XX(3,2), r * x_esph
    + XX(3,3), 'EdgeColor', 'none', 'FaceLighting', 'phong');
colormap([1 0 0])
camlight rigth;

%%                                  PROGRAMA PRINCIPAL
disp('Initializing the main program...');
fwrite(arduino,inicializa,'uchar');
 while k <= k_max
     ProcessingGimbal;

     trigger(vid1);
     img1 = getdata(vid1);

     trigger(vid2);
     img2 = getdata(vid2);

%%
     ProcessingAlgorith3D;
     if isempty(XX1_XX2)
        XX = [];
     else
         %%
%         XX = XX1_XX2(:,4:6);
        XX = XX1_XX2(:,1:3);
        figure(1), hold on,
        switch size(XX,1)
         case 1
             figure(1), hold on
             set(plot_sphera_1, 'xdata', r * x_esph + XX(1,1), 'ydata', r *
                 y_esph + XX(1,2), 'zdata', r * z_esph + XX(1,3))
             drawnow;
```

```
99          case 2
100             figure(1), hold on
101             set(plot_sphera_1, 'xdata', r * x_esph + XX(1,1), 'ydata', r *
                    y_esph + XX(1,2), 'zdata', r * z_esph + XX(1,3))
102             set(plot_sphera_2, 'xdata', r * x_esph + XX(2,1), 'ydata', r *
                    y_esph + XX(2,2), 'zdata', r * z_esph + XX(2,3))
103             drawnow;
104         case 3
105             figure(1), hold on
106             set(plot_sphera_1, 'xdata', r * x_esph + XX(1,1), 'ydata', r *
                    y_esph + XX(1,2), 'zdata', r * z_esph + XX(1,3));
107             set(plot_sphera_2, 'xdata', r * x_esph + XX(2,1), 'ydata', r *
                    y_esph + XX(2,2), 'zdata', r * z_esph + XX(2,3));
108             set(plot_sphera_3, 'xdata', r * x_esph + XX(3,1), 'ydata', r *
                    y_esph + XX(3,2), 'zdata', r * z_esph + XX(3,3));
109             drawnow;
110             [roll, pitch, yaw, R, A] = Points2Euler_R(XX,2);
111             %% Inicializing the rotation reference
112             cont_oref = cont_oref + 1;
113             if cont_oref == 1
114                 R_sens = R;
115             end
116             if onPlanePoints == 1
117                 RotatingPlane(vert_0, fac, plot_cube_points, R, A, plot_u,
                        plot_v, plot_w, 300);
118             end
119             if onPlaneSensors == 1
120                 RotatingPlane(vert_0, fac, plot_cube_sensors, R_sens, A,
                        plot_u_sens, plot_v_sens, plot_w_sens, 200)
121             end
122            otherwise
123 %              set(plot_points, 'xdata', XX(:,1), 'ydata', XX(:,2), 'zdata',
    XX(:,3));
124         end
125     end
126
127 %% Plotting image data
128 if onPlotImageBinary == 1
129     clf(figure(2)),
130     if onPlotRPYSensorCam == 1
131         set(gcf,'units','normalized','outerposition',[0.7 0.05 0.30 .5]),
                hold on,
132     else
133         set(gcf,'units','normalized','outerposition',[0.7 0 0.30 1]),hold
                on,
134     end
135     hold on,
```

```
136     subplot(2,2,1),
137     imshow(img1), title('Image 1');
138     subplot(2,2,2),
139     imshow(binary1),title('Binary 1 ');
140     hold on,
141     subplot(2,2,3),
142     imshow(img2), title('Image 2');
143     subplot(2,2,4),
144     imshow(binary2),title('Binary 2');
145     if ~isempty(XX)
146        PlotPointsBelongTo;
147     end
148 end
149 %%  Plotting Epipolar data
150     if onPlotEpipolar == 1
151         if ~isempty(XX)
152             figure(1), hold on
153             PlotEpipolares;
154         end
155     end
156 %% Plotting angles from sensors and cameras
157     if onPlotRPYSensorCam == 1
158         figure(3),
159         hold on,
160         subplot(3,1,1)
161         set( plot_roll , 'xdata' , 1:k , 'ydata' , ypr_points_sens(4,1:k)),
                hold on,
162         set( plot_roll_points , 'xdata' , 1:k , 'ydata' , ypr_points_sens
                (1,1:k)),hold on,
163
164         subplot(3,1,2)
165         set( plot_pitch , 'xdata' , 1:k , 'ydata' , ypr_points_sens(5,1:k))
                ; hold on,
166         set( plot_pitch_points , 'xdata' , 1:k , 'ydata' , ypr_points_sens
                (2,1:k)); hold on,
167
168         subplot(3,1,3)
169         set( plot_yaw , 'xdata' , 1:k , 'ydata' , ypr_points_sens(6,1:k));
                hold on,
170         set( plot_yaw_points , 'xdata' , 1:k , 'ydata' , ypr_points_sens
                (3,1:k)); hold on,
171         drawnow
172     end
173  %% Plotting axis validation to compare equality
174     if onAxisValidation == 1
175         nroFig = 1;
176         T_ref = A;
```

```
177         g_vt = [R T_ref'; 0 0 0 1 ];
178         g_vti = inv(g_vt);
179         R_vt = g_vti(1:3, 1:3);
180         T_vt = g_vti(4, 1:3);
181         hold on,
182         AxisUpgrade( R_vt, T_vt, plot_u_vt, plot_v_vt, plot_w_vt, 80 ,
                nroFig);
183         %volta ao eixo principal desde o eixo do gimbal X180  y90
184         [roll_c, pitch_c, yaw_c ] = dcm2angle(R_vt,'XYZ');
185         ypr_points_sens(1:3,k) = 180/pi * [roll_c pitch_c yaw_c]';
186         hold on,
187         drawnow;
188
189         g_vts = [R_sens T_ref'; 0 0 0 1 ];
190         g_vtsi = inv(g_vts);
191         R_vts = g_vtsi(1:3, 1:3);
192         T_vts = g_vtsi(4, 1:3);
193         grid on,
194         AxisUpgrade( R_vts, T_vts, plot_u_svt, plot_v_svt, plot_w_svt, 40 ,
                nroFig);
195         %volta ao eixo principal desde o eixo do gimbal X180  y90
196         [roll_s, pitch_s, yaw_s,  ] = dcm2angle(R_vts,'XYZ');
197         ypr_points_sens(4:6,k) = 180/pi * [roll_s, pitch_s, yaw_s]';
198         ypr_points_sens(4,k) = ypr_points_sens(4,k)  13.0725; % offset
199         hold on,
200         drawnow;
201         ha2 = findobj('Type','axes','Tag','Axes2_orient');
202         axes(ha2);
203     end
204
205     k = k+1;
206  end
207 %% Ploting Roll, pitch yaw for sensors and cameras
208 if onShowMaxEnvironment == 1
209     k= k  1;
210     PlotRPY_sensors_camaras;
211 end
212
213 %% Ending videos and  Ending Gimbal
214 disp('Turning␣off␣the␣gimbal...')
215 fwrite(arduino,finaliza,'uchar');
216 pause(2);
217 fclose(arduino);
218 delete(arduino);
219 clear arduino;
220
221 disp('Turning␣off␣the␣cameras...');
```

```
stop(vid1);
delete(vid1);
pause(1);

stop(vid2);
delete(vid2);
pause(1);
catch exception
    %% Ending videos and gimbal
    disp('Turning off the gimbal...')
    fwrite(arduino,finaliza,'uchar');
    pause(2);
    fclose(arduino);
    delete(arduino);
    clear arduino;

    disp('Turning off the cameras...');
    stop(vid1);
    delete(vid1);
    pause(1);

    stop(vid2);
    delete(vid2);
    pause(1);

    disp('main')
    rethrow(exception)
end
```

E a função principal que realiza a reconstrução de múltiplos pontos é "ProcessingAlgorith3D.m", que foi chamado no programa principal de acima, e apresenta-se a seguir:

```
try
%%
[x1, binary1] = GetCentroids(img1,1);
[x2, binary2] = GetCentroids(img2,2);

if (isempty(x1) || isempty(x2))
    XX1_XX2 = [];
else
  %% relative pose between two cameras
  g12 = inv(g1)*g2;
  T12 = g12(1:3,4);
  R12 = g12(1:3,1:3);

  %Epipolar Data
```

```
15  E = uChapeu(T12)*R12;
16  e2 = T12;
17  e1 = R12'*T12;
18  re1 = sum(e2'*E);
19  re2 = sum(E*e1);
20
21  %% Points Correlation
22   size_x1 = size(x1,1);
23   size_x2 = size(x2,1);
24   s_x12 = min(size_x1, size_x2);
25   x1 = x1(1:s_x12,:);
26   x2 = x2(1:s_x12,:);
27  if size_x1 == size_x2
28    cont=0;
29    data_chosen = zeros(size_x1, 22);
30    data_all = zeros(size_x1 * size_x2, 22);
31    data_grouped = zeros(size_x2, 22);
32    for i = 1: size_x1
33        X1 = inv(K1)*[x1(i,:)  1]';
34        for j = 1: size_x2
35            X2 = inv(K2)*[x2(j,:)  1]';
36
37            l2 = E * X1;
38            l1 = E' * X2;
39
40            %epipolar constraints
41            re = (X1'*uChapeu(T12)*R12*X2);
42
43            rl1 = l1'*e1;
44            rl2 = l2'*e2;
45
46            rl13 = l1'*X1;
47            rl24 = l2'*X2;
48
49            cont=cont+1;
50            data_all(cont,:) = [abs(re) x1(i,:) x2(j,:) X1' X2' cont l1' l2
                  ' rl1 rl2 rl13 rl24 ];
51            data_grouped(j,:) = [abs(re) x1(i,:) x2(j,:) X1' X2' cont l1'
                  l2' rl1 rl2 rl13 rl24];
52        end
53    %% 1st Control to verify if a pixel point is repeating
54        if size_x1 == 1
55            data_chosen = sortrows(data_grouped,1);
56            continue;
57        end
58        % to verify a distance between the two first epipolar constraint
59        Gj = sortrows(data_grouped,1);
```

```
60          if abs(abs(Gj(1,1))    abs(Gj(2,1))) > 6
61              data_chosen(i,:) = Gj(1,:);
62          else
63              %there is a problem in the epipolar constraint
64              d1_px11 = abs(abs(Gj(1,4))    abs(Gj(1,2)));
65              d1_px12 = abs(abs(Gj(1,5))    abs(Gj(1,3)));
66              d2_px21 = abs(abs(Gj(2,4))    abs(Gj(2,2)));
67              d2_px22 = abs(abs(Gj(2,5))    abs(Gj(2,3)));
68              s_d1 = d1_px11 + d1_px12;
69              s_d2 = d2_px21 + d2_px22;
70              if s_d1 < s_d2
71                  data_chosen(i,:) = Gj(1,:);
72              else
73                  data_chosen(i,:) = Gj(2,:);
74              end
75          end
76       end
77      data_chosen = data_chosen(1:s_x12,:);
78
79      %% 2nd Control if a point pixel is repeting
80      control = [sortrows(x1, 1) sortrows(x2, 1)];
81      data_chosen = sortrows(data_chosen, 3);
82      for l = 1 : s_x12
83          if sum(control(l,:) == data_chosen(l, 2:5)) ~= 4
84              find  = 1;
85              cc1 = 0;
86              while find
87                  cc1 = cc1 + 1;
88                  if sum(control(l,:) == data_all(cc1, 2:5)) == 4
89                      data_chosen(l,:) = data_all(cc1,:);
90                      find = 0;
91                  end
92              end
93          end
94      end
95      %% Determina profundidades
96      if(cont > 0)
97          m = size(data_chosen,1);
98          XX1_XX2 = zeros(m,6);
99          L12 = zeros(m,2);
100         for i = 1 : size(data_chosen,1)
101             X1 = data_chosen(i,6:8)';
102             X2 = data_chosen(i,9:11)';
103             M = [X1   R12*X2];
104             L = inv(M'*M)*M'*T12;
105             L12(i,:)  = L';
106             XX1 = g1 * [X1 * L(1); 1];
```

```
107                 XX2 = g2 * [X2 * L(2); 1];
108                 XX1_XX2(i,1:end) = [XX1(1:3,1)' XX2(1:3,1)'];
109             end
110         else
111             XX1_XX2 = [];
112         end
113     else
114             XX1_XX2 = [];
115     end
116 end
117 catch exception
118             disp('Alg3D3')
119             rethrow(exception)
120 end
```

ANEXO A – Fatores básicos de Álgebra Linear

Definição A.1. Um espaço linear ou um espaço vetorial. *Um conjunto (de vetores) V é considerado um* espaço linear *sobre o campo $\mathbb{R}$ se os seus elementos, chamados vetores, estão fechados em duas operações básicas: multiplicação escalar e vetor soma "+".*

Ou seja, dados quaisquer dois vetores $v_1, v_2 \in V$ e qualquer dois escalares $\alpha, \beta \in \mathbb{R}$, a combinação linear $\alpha v_1 + \beta v_2$ é também um vetor em V. Além disso, a adição é comutativa e associativa, que tem uma identidade 0, e cada elemento tem uma inversa, "$-v$", de tal modo que $v + (-v) = 0$. A multiplicação escalar respeito a estrutura de $\mathbb{R}$; isto é $\alpha(\beta)u = (\alpha\beta)v, 1v = v$ e $0v = 0$. A adição e multiplicação escalar estão relacionadas pelas leis distributivas: $(\alpha + \beta)v = \alpha v + \beta v$ e $\alpha(v + u) = \alpha v + \alpha u$.

Por exemplo, $\mathbb{R}^n$ é um espaço linear sobre o campo de números reais $\mathbb{R}$. Para ser coerente, usa uma coluna para representar um vetor:

$$[x_1, x_2, x_3, ..., x_n]^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n \tag{67}$$

onde $[x_1, x_2, x_3, ..., x_n]^T$ significa "a (linha) vetor $[x_1, x_2, x_3, ..., x_n]$ transposta".

Dados dois escalares $\alpha, \beta \in \mathbb{R}$ e dois vetores $v_1 = [x_1, x_2, x_3, ..., x_n]^T \in \mathbb{R}^n$ e $v_2 = [y_1, y_2, y_3, ..., y_n]^T \in \mathbb{R}^n$, a sua combinação linear é um somatório de componente racional ponderada pelo α e β:

$$\begin{aligned} \alpha v_1 + \beta v_2 &= \alpha[x_1, x_2, ..., x_n]^T + \beta[y_1, y_2, ..., y_n]^T \\ &= [\alpha x_1 + \beta y_1, \alpha x_2 + \beta y_2, ..., \alpha x_n + \beta y_n]^T. \end{aligned}$$

definição A.2. produto interno. *A função*

$$\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$

é um produto interno *se*

1. $\langle u, \alpha v + \beta w \rangle = \alpha \langle u, v \rangle + \beta \langle u, w \rangle, \forall \alpha, \beta \in \mathbb{R}$

2. $\langle u, v\rangle = \langle v, u\rangle$

3. $\langle v, v\rangle \geq 0$ e $\langle v, v\rangle = 0 \Leftrightarrow v = 0$

Para cada vetor v, $\sqrt{\langle v, v, \rangle}$ *é chamado sua* norma.

O produto interno é também chamado uma *métrica*, Já que pode ser usado para medir o comprimento e ângulos.

Para simplificar, uma base standard é muitas vezes escolhido para o espaço vetorial $\mathbb{R}^n$ como o conjunto de vetores

$$e_1 = [1, 0, 0, ..., 0]^T, e_2 = [0, 1, 0, ..., 0]^T, e_n = [0, 0, 0, ..., 1]^T \tag{68}$$

O matriz $I = [e_1, e_2, ..., e_3]$ com esses vetores como colunas é exatamente o matriz identidade $n \times n$.

definição A.3. Ortogonalidade. *Dois vetores* x, y *são referidos como sendo ortogonais se o produto interno é zero:* $\langle x, y\rangle = 0$. *Isto é muitas vezes indicado como* $x \perp y$.

definição A.4. Produto Kronecker de duas matrizes. *Dadas duas matrizes* $A \in \mathbb{R}^{m\times n}$ *e* $B \in \mathbb{R}^{k\times l}$, *o seu produto Kronecker, denotado por* $A \otimes B$, *é uma nova matriz*

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mk\times nl} \tag{69}$$

Se A e B são dois vetores, isto é, $n = l = 1$, *o produto* $A \otimes B$ *também é um vetor mas de dimensão mk.*

Em Matlab, um pode facilmente calcular o produto Kronecker usando o comando C = Kron (A, B).

definição A.5. O grupo afim $A(n)$**.** *Uma transformação afim L de* $\mathbb{R}^n$ *para* $\mathbb{R}^n$ *é definida em conjunto por uma matriz* $A \in GL(n)$ *e um vetor* $b \in \mathbb{R}^n$ *de tal forma que*

$$L : \mathbb{R}^n \to \mathbb{R}^n; x \mapsto Ax + b \tag{70}$$

O conjunto de todas essas transformações afins é chamado o grupo afim de dimensão n e é denotado por $A(n)$.

Observe que o mapa L assim definido *não* é um mapa linear de $\mathbb{R}^n$ para $\mathbb{R}^n$, a menos que $b = 0$. No entanto, pode-se "encaixar"este mapa em um espaço uma dimensão maior, de modo que ainda pode ser representada por uma única matriz. Se se identifica

um elemento $x \in \mathbb{R}^n$ com $\begin{bmatrix} x \\ 1 \end{bmatrix} \in \mathbb{R}^{n+1}$ então L se torna um mapa do $\mathbb{R}^{n+1}$ para $\mathbb{R}^{n+1}$ no seguinte sentido:

$$L : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}; \begin{bmatrix} x \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}. \tag{71}$$

Assim, uma matriz da forma

$$\begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}, a \in GL(n), b \in \mathbb{R}^n, \tag{72}$$

descreve totalmente um mapa afim, e se chama de *matriz afim*. Esta matriz é um elemento do grupo linear geral $GL(n+1)$. Neste modo, $A(n)$ é identificado como um subconjunto (e, de facto, um subgrupo) de $GL(n+1)$. A multiplicação de matrizes duas afins no conjunto $A(n)$ é

$$\begin{bmatrix} A_1 & b_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} A_2 & b_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} A_1A_2 & A_1b_2 + b_1 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}, \tag{73}$$

que também é uma matriz afim em $A(n)$ e representa a composição de duas transformações afins.

Dada $\mathbb{R}^n$ e sua estrutura do produto interno standard, $\langle x, y\rangle x^T y, \forall x, y, \in \mathbb{R}^n$, vamos considerar o conjunto de transformações lineares (ou matrizes) que preservam o produto interno.

Referências

AESCHIMANN, R. Ground or obstacles? detecting clear paths in vehicle navigation. *IEEE International Conference on Robotics and Automation (ICRA)*, v. 1, n. 1, p. 8, 2015. ISSN 978-1-4799-6923-4. Citado na página 16.

ANTONIAZI, R. Construção de sistema ótico a partir de lentes de água. 2013. Disponível em: <http://www.ifi.unicamp.br/~accosta/>. Citado 2 vezes nas páginas 36 e 37.

AYKIN, M. Forward-look 2-d sonar image formation and 3-d reconstruction. *Oceans - San Diego, 2013 - IEEE*, v. 1, p. 1 – 10, 2013. ISSN 14115342. Nenhuma citação no texto.

BACKES, A. Filtragem espacial. 2014. Disponível em: <http://www.facom.ufu.br/~backes/gsi058/Aula06-FiltragemEspacial.pdf>. Citado na página 52.

BóDIS, A. *Stereo Vision*. [S.l.]: InTech, 2008. ISBN 978-953-7619-22-0. Citado 2 vezes nas páginas 14 e 15.

BEUL, M. A high-performance mav for autonomous navigation in complex 3d environments. *IEEE - International Conference on Unmanned Aircraft Systems (ICUAS)*, v. 1, n. 1, p. 8, 2015. ISSN 978-1-4799-6009-5. Citado na página 16.

BORGSTADT, J. Multi-modal localization algorithm for catheter interventions. *IEEE International Conference on Robotics and Automation (ICRA)*, v. 1, n. 1, p. 8, 2015. ISSN 978-1-4799-6923-4. Citado na página 16.

BOUGUET, J. *Camera calibration toolbox for MATLAB*. http://www.vision.caltech.edu/bouguetj/calib_doc/, 2013. Citado 3 vezes nas páginas 32, 55 e 108.

BRIGGS, W. *The DFT: An Owners' Manual for the Discrete Fourier Transform*. [S.l.]: Society for Industrial and Applied Mathematics, 1987. ISBN 0898713420. Citado na página 52.

BROGARDH, T. Robot control overview: An industrial perspective. *Modeling, Identification and Control*, Norwegian Society of Automatic Control, v. 30, n. 3, p. 167–180, 2009. Citado na página 18.

BRUSHLESSGIMBALS. Getting started with brushless gimbals. 2013. Disponível em: <http://brushlessgimbals.com/,http://www.hovership.com/2013/08/06/getting-started-with-brushless-gimbals/>. Citado na página 89.

BUCIOLI, A. A utilização da realidade aumentada no tratamento e simulação de sinais cardiológicos com biofeedback em tempo real. *WRVA08 - 5to Workshop de realidade virtual aumentada*, n. 31, p. 31–38, 2008. Nenhuma citação no texto.

CAI, C. Uncalibrated 3d stereo image-based dynamic visual servoing for robot manipulators. p. 63–70, 2013. Citado na página 19.

CAI, C. 6d image-based visual servoing for robot manipulators with uncalibrated stereo cameras. p. 736–742, 2014. Citado na página 19.

CHIANG, M.-H. Development of a 3d parallel mechanism robot arm with three vertical-axial pneumatic actuators combined with a stereo vision system. *PMC - US National Library of Medicine, National Institute of health*, National Center for Biotechnology Information, U.S. National Library of Medicine, v. 11, n. 12, p. 11476–11494, 2011. Citado 2 vezes nas páginas 18 e 19.

CORKE, P. *Robotics, Vision and Control*. [S.l.]: Springer-Verlang Berlin Heidelberg, 2011. (Springer Tracks in Advanced Robotics). ISBN 978-3-642-20143-1. Citado 2 vezes nas páginas 22 e 48.

COURROL, L. *Optica tecnica I*. [S.l.], 2006. Citado 2 vezes nas páginas 37 e 38.

DAVIS, J. Spacetime stereo: a unifying framework for depth from triangulation. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference*, v. 2, n. 7792768, p. II – 359–66, 2003. ISSN 1063-6919. Nenhuma citação no texto.

DECAROLI, A. *Matrizes vetores geometria analítica*. [S.l.]: Livraria Nobel S.A, 1976. (1ra ed.). Citado na página 96.

DEFARIAS, T. *Metodologia para reconstrução 3D baseada em imagens*. Tese (Doutorado) — Universidade Federal de Pernambuco, 2012. Citado na página 47.

DEOLIVEIRA, P. *Auto-localização e construção de mapas de ambiente para robôs móveis baseados em visão omnidirecional estéreo*. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2008. Nenhuma citação no texto.

DESOUSA, J. *Um método para determinação da profundidade combinando visão estereo e autocalibração para aplicação em robotica movel*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2007. Nenhuma citação no texto.

DESOUZA, R. Óptica geometrica. v. 1, n. 30, p. 33, 2012. Citado na página 36.

DHAWAN, A. Image formation. *Wiley-IEEE Press*, v. 1, p. 23 – 63, 2011. Citado 2 vezes nas páginas 34 e 36.

DOMINGUEZ, M. *Current Advancements in Stereo Vision*. [S.l.]: InTech, 2012. ISBN 978-953-51-0660-9. Citado na página 13.

DOSSANTOS, J. Confiabilidade inter e intraexaminadores nas mensurações angulares por fotogrametria digital e goniometria. *Creative Commons*, v. 24, n. 1,2, p. 389–400, 2011. ISSN 0103-5150. Nenhuma citação no texto.

DOUGHERTY, E. *An Introduction to Nonlinear Image Processing*. [S.l.]: SPIE - The international Society for Optical Engineering, 1994. (Volume TT 16). Citado na página 51.

DUC, T.; KANG, H.-J. Fusion of vision and inertial sensors for position-based visual servoing of a robot manipulator. Springer Berlin Heidelberg, v. 7995, p. 536–545, 2013. Disponível em: <http://dx.doi.org/10.1007/978-3-642-39479-9_63>. Citado na página 18.

DUFAUX, F. *Emerging Technologies for 3D Video Creation, Coding, Transmission and Rendering*. [S.l.]: John Wiley and Sons, 2013. ISBN 9781118355114. Citado na página 13.

FETZER, T. 3d interaction design: Increasing the stimulus-response correspondence by using stereoscopic vision. *IEEE*, v. 1, n. 1, p. 6, 2015. ISSN 978-1-4799-6026-2. Citado na página 17.

FINDEISEN, M. An omnidirectional stereo sensor for human behavior analysis in complex indoor environments. *IEEE - International Conference on Consumer Electronics (ICCE)*, v. 1, n. 1, p. 3, 2015. ISSN 978-1-4799-7543-3. Citado na página 16.

FU, C. Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system. *IEEE - International Conference on Unmanned Aircraft Systems (ICUAS)*, v. 1, n. 1, p. 6, 2015. ISSN 978-1-4799-6009-5. Citado na página 16.

GE, L.; JIE, Z. A real-time stereo visual servoing for moving object grasping based parallel algorithms. In: IEEE. *Industrial Electronics and Applications, 2007. ICIEA 2007. 2nd IEEE Conference on*. [S.l.], 2007. p. 2886–2891. Citado na página 19.

GOESELE, M. Multi-view stereo revisited. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference*, v. 2, p. 2402 – 2409, 2006. ISSN 1063-6919. Citado na página 46.

GOMES, L. Controle de um veiculo quadrirotor usando um sistema de captura de movimentos. *XX Congresso Brasileiro de Automática*, v. 1, n. 1, p. 1–8, 2014. Nenhuma citação no texto.

GONÇALVES, L. *Matching de Objetos em Imagens Estéreos*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, 2005. Citado na página 13.

HABIB, K. *Interdisciplinary Mechatronics : Engineering Science and Research Development*. [S.l.]: John Wiley and Sons, 2013. ISBN 9781848214187. Citado na página 19.

HAN, B. 3d dense reconstruction from 2d video sequence via 3d geometric segmentation. *ScienceDirect - ElSevier*, v. 22, n. 421, p. 421–431, 2011. Nenhuma citação no texto.

HAO, K. Trinocular matching realized by a monocular stereovision sensor for parallel manipulator. *Control, Automation, Robotics and Vision*, IEEE, v. 10, n. 1436, p. 1436–1441, 2008. Citado na página 19.

HARTLEY, R. *Multiple View Geometry in Computer Vision*. [S.l.]: Cambridge University Press, New York, 2004. (A Wiley-Interscience publication). ISSN 978-0-511-18618-9. Citado 2 vezes nas páginas 13 e 15.

HEIDEMAN, M. *Multiplicative Complexity, Convolution, and the DFT*. [S.l.]: Society for Industrial and Applied Mathematics, 1988. ISBN 139781461283997. Citado na página 52.

HEIKKILA, S. A four-step camera calibration procedure with implicit image correction. *IEEE - Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, v. 1, p. 1106–1112, 1997. Nenhuma citação no texto.

HERBST, E. *Camera Calibration by Corners Detection.* [S.l.], 2003. Nenhuma citação no texto.

HUNG, C.-H. Photometric stereo in the wild. *IEEE - Winter Conference on Applications of Computer Vision*, v. 1, n. 1, p. 8, 2015. ISSN 978-1-4799-6683-7. Citado na página 17.

IHRKE, I. *Digital elevation mapping using stereoscopic vision.* Dissertação (Mestrado) — Royal Institute of Technology, 2001. Nenhuma citação no texto.

JEONG, Y. Uncalibrated multiview synthesis based on epipolar geometry approximation digest of technical papers. *IEEE - International Conference on Consumer Electronics (ICCE)*, v. 1, n. 1, p. 2, 2015. ISSN 978-1-4799-7543-3 /15. Citado na página 17.

JINGHUA, G. Camera based automatic calibration for the varrier-system. *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference*, v. 1, p. 110, 2005. ISSN 1063-6919. Nenhuma citação no texto.

JURADO, P. *Filtros.* http://alojamientos.us.es/gtocoma/pid/Presentacion-PID.pdf, 2012,2013. Citado na página 51.

KANATANI, K. 3d euclidean versus 2d non-euclidean: two approaches to 3d recovery from images. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, v. 11, n. 3383959, p. 329 – 332, 1989. ISSN 0162-8828. Citado 2 vezes nas páginas 23 e 27.

KAREKAR, A. 2d to 3d image conversion and disparity map estimation ausing pso algorithms. *IEEE - International Conference on Computing Communication Control and Automation*, v. 1, n. 1, p. 5, 2015. ISSN 978-1-4799-6892-3. Citado na página 17.

KIM, J. Confocal disparity estimation and recovery of pinhole image for real-aperture stereo camera systems. *Image Processing, 2007. ICIP 2007. IEEE International Conference*, v. 5, n. 9820873, p. V – 229 – V – 232, 2007. ISSN 1522-4880. Citado na página 36.

KOO, J. Robust stereo matching algorithm for advanced drive assistance system. *IEEE - International Conference on Consumer Electronics (ICCE)*, v. 1, n. 1, p. 2, 2015. ISSN 978-1-4799-7543-3. Citado na página 17.

KWON, S. Selective attentional point-tracking through a head-mounted stereo gaze tracker based on trinocular epipolar geometry. *IEEE*, v. 1, n. 1, p. 5, 2015. ISSN 978-1-4799-6144-6. Citado na página 17.

LAI, Y. Distance estimation based on disparity analysis for vehicle applications. *IEEE-International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, v. 1, n. 1, p. 2, 2015. ISSN 978-1-4799-8745-0. Citado na página 17.

LARIBI, M. Analysis and dimensional synthesis of the {DELTA} robot for a prescribed workspace. *Mechanism and Machine Theory*, v. 42, n. 7, p. 859 – 870, 2007. ISSN 0094-114X. Citado na página 18.

LASEVITCH, H. *Controle Não-Linear do Veículo Omni-Direcional via Servovisão e Sensor Inercial.* Dissertação (Mestrado) — Pontifícia Universidade do Rio Grande do Sul – PUCRS, 2014. Nenhuma citação no texto.

LASEVITCH1, H. *Manipulador com Force-Feedback Baseado no Robô Delta*. [S.l.], 2012. Citado na página 18.

LAZZARI, F. Desenvolvimento de um robô paralelo delta associado com visão computacional para aplicações pick and place. Trabalho de conclusão de curso. 2008. Citado na página 19.

LI, Q.; WU, F. Control performance improvement of a parallel robot via the design for control approach. *Mechatronics*, v. 14, n. 8, p. 947 – 964, 2004. ISSN 0957-4158. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957415804000352>. Citado na página 18.

LI, S. Hetereogeneus multi-task learning for human pose estimation with deep convolutional neural network. *nternational Journal of Computer Vision*, v. 113, n. 1573, p. 1573–1405, 2015. ISSN 0920-5691. Citado na página 13.

LIANG, C. Light field analysis for modeling image formation. *Image Processing, IEEE Transactions*, v. 2, n. 11746811, p. 446 – 460, 2011. ISSN 1057-7149. Citado na página 34.

LIMA, J. *Optica*. [S.l.], 2010. Disponível em: <http://www.infoescola.com/optica/>. Citado 2 vezes nas páginas 38 e 39.

LIN, C.-T. Automatic age estimation system, for face images. *Intech, open scince open mind*, v. 9, n. 1, p. 1–9, 2012. ISSN 216:2012. Nenhuma citação no texto.

LIN, Y. Rotation, scaling, and translation resilient watermarking for images. *Image Processing, IET - IEEE*, v. 5, n. 12034904, p. 328 – 340, 2011. ISSN 1751-9659. Citado 2 vezes nas páginas 27 e 30.

LIU, Z. An improved minimum spanning tree stereo matching algorithm. *IEEE*, v. 27, n. 4, p. 1866–1869, 2015. Citado na página 17.

LOURAKIS, M. Using geometric constraints for matching disparate stereo views of 3d scenes containing planes. *Pattern Recognition, 2000. Proceedings. 15th International Conference*, v. 1, n. 6873676, p. 419 – 422 vol.1, 2000. ISSN 1051-4651. Nenhuma citação no texto.

MARTINET, P. *EMARO: Computer Vision*. [S.l.], 2013. Disponível em: <http://www.irccyn.ec-nantes.fr/~martinet>. Citado na página 13.

MARYUM, A. *Development of a stereo vision system fo outdoor mobile robots*. Dissertação (Mestrado) — University of Florida, 2006. Nenhuma citação no texto.

MATTHEWS, J. Convolution and correlation. 2002. Disponível em: <http://www.generation5.org/content/2002/convolution.asp>. Citado na página 50.

MCCOUN, J. *Binocular Vision: Development, Depth Perception and Disorders*. [S.l.]: Nova Science Publishers, Inc, 2010. ISBN 9781608765478. Citado na página 13.

MOHEBBI, A. An acceleration command approach to robotic stereo image-based visual servoing. v. 19, n. 1, p. 7239–7245, 2014. Citado na página 19.

MONACO, J. Epipolar spaces and optimal sampling strategies. *Image Processing, 2007. ICIP 2007. IEEE International Conference*, v. 6, n. 9821027, p. VI – 545 – VI – 548, 2007. ISSN 1522-4880. Citado na página 32.

MOUATS, T. Ieee transaction on image processing. *IEEE*, v. 24, n. 9, p. 2685–2700, 2015. Citado na página 17.

MOZEROV, M. Accurate stereo matching by two-step energy minimization. *IEEE*, v. 1, n. 1, p. 11, 2015. Citado na página 17.

NAKAMURA, N. Homogeneous stabilization for input affine homogeneous systems. *Automatic Control, IEEE Transactions*, v. 54, n. 10861437, p. 2271 – 2275, 2009. ISSN 0018-9286. Citado na página 31.

NAMMOTO, T. High speed/accuracy visual servoing based on virtual visual servoing with stereo cameras. p. 44–49, 2013. Citado na página 19.

NISHIMURA, C. Analise comparativa de algoritmos de correlação local baseados em intensidade luminosa. *radarciencia*, v. 1, n. oai-teses-usp-br-tde-14082008-082214, p. 6, 2008. Nenhuma citação no texto.

NIXON, M. *Feature extration and image processing*. [S.l.]: Elsevier, 2002. ISBN 0750650788. Citado na página 49.

OLEYNIKOVA, H. Reactive avoidance using embedded stereo vision for mav flight. *IEEE International Conference on Robotics and Automation (ICRA)*, v. 1, n. 1, p. 7, 2015. Citado na página 16.

OUELLET, J. Developing assistant tools for geometric camera calibration: assessing the quality of input images. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference*, v. 4, n. 8244164, p. 80 – 83, 2004. ISSN 1051-4651. Nenhuma citação no texto.

PARI, L. Image base visual servoing: Estimation of the image jacobian by using lines in a stereo vision system. p. 109–114, 2009. Citado na página 19.

PERES, F. *Scanner 3D: Problemas e soluções*. Dissertação (Mestrado) — Universidade Estadual de Londrina, 2013. Nenhuma citação no texto.

PERTUSA, J. *Técnicas de análisis de imagen*. [S.l.]: Universitat de València, 2011. (2da ed.). Citado na página 51.

RAMAZINI, F. *Modelagem e controle de um robô paralelo delta com três graus de liberdade*. [S.l.], 2011. Citado na página 18.

RENDULIC, I. Estimating diver orientation from video using body markers. *IEEE - MIPRO*, v. 1, n. 1, p. 6, 2015. Citado na página 17.

ROBERTO, R. Um estudo de aplicações de realidade aumentada para educação. *Universidade Federal de Minas Gerais, LBD - Laboratório de Banco de Dados*, v. 6, n. 0056, p. 6, 2012. Nenhuma citação no texto.

ROBERTZ, S.; KELKAR, S. Precise robot motions using dual motor control. IEEE, p. 5613–5620, 2010. Citado na página 18.

ROD, N. *Vergencia*. [S.l.], 2010. Disponível em: <http://hyperphysics.phy-astr.gsu.edu/hbasees/geoopt/vergence.html#c1>. Citado na página 37.

RODRIGUES, C. *Sistema de Visão Estéreo para Formas Polimétricas*. 1999. Nenhuma citação no texto.

SAEZ, J. Aerial obstacle detection with 3d mobile devices. *IEEE Journal of Biomedical and Health Informatics*, v. 1, n. 1, p. 7, 2014. Citado na página 17.

SAMANTARAY, A. Development of a 3d parallel mechanism robot arm with three vertical-axial pneumatic actuators combined with a stereo vision system. *Journal of engineering manufacture*, Proceedings of the Institution of Mechanical Engineers. Part B, v. 224, n. 169, p. 0954–4054, 2010. Citado na página 18.

SANTOS, A. Refinamento de reconstrução 3d através de seleção apropriada de keyframes. *GRVM - Virtual Reality and Multimedia Reserch Group*, v. 1, n. 1, p. 1–5, 2011. Nenhuma citação no texto.

SARRíA, F. *Técnicas de filtrado*. http://www.um.es/geograf/sigmur/teledet/tema06.pdf, 2005,2006. Citado na página 50.

SERRA, R. *Sistema de Visão Stereo Ativo aplicado aos Robôs do ISePorto*. Dissertação (Mestrado) — Instituto Superior de Engenharia do Porto - ISEP, 2012. Nenhuma citação no texto.

SHA, X. Three-dimensional positioning control based on stereo microscopic visual servoing system. *Optical Engineering*, Third International Conference on Machine Learning and Cybernetics, v. 54, n. 1, p. 1–5, 2014. Citado na página 19.

SHIMIZU, M. Calibration and rectification for reflection stereo. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference*, v. 1, n. 10139995, p. 1 – 8, 2008. ISSN 1063-6919. Citado na página 34.

SILVEIRA, G. Visual servo control of nonholonomic mobile robots. *COBEM 2001 - XVI Congresso Brasileiro de Engenharia Mecânica*, v. 1, n. 1, p. 1–9, 2001. Nenhuma citação no texto.

SILVEIRA, G. Switching between primary image-based visual servoing task - a first discussion with experimental results. *VI Simpósio Brasileiro de Automação Inteligente*, v. 6, n. 2, p. 1–6, 2003. Nenhuma citação no texto.

SIMõES, F. Realidade aumentada sem marcadores apartir de rastreamento baseado em textura - uma abordagem baseada em pontos de interesse e filtro de particulas. *5º Workshop de Realidade Virtual e Aumentada (WRVA2008), researchgate*, v. 1, n. 255666308, p. 5, 2011. Citado na página 17.

STRICKLAND, J. O que é um cardan - e o que isso tem a ver com a nasa? 2015. Disponível em: <http://science.howstuffworks.com/gimbal.htm>. Citado na página 89.

STURM, P. Multi-view geometry for general camera models. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference*, v. 1, n. 8588875, p. 206 – 212 vol. 1, 2005. ISSN 1063-6919. Citado na página 40.

SUCAR, L. *Visión Computacional*. [S.l.]: Departamento de Computación. ITESM, 2007. Citado na página 13.

TARAGLIO, S. *Advances in Theory and Applications of Stereo Vision*. [S.l.]: InTech, 2011. ISBN 978-953-307-516-7. Citado na página 13.

TAVARES, J. Obtenção de estrutura tridimensional a partir de movimento da câmara. *ResearchGate*, v. 1, n. 37649551, p. 3, 1995. Citado na página 13.

THOME, A. *Processamento de Imagens Processamento de Imagens Tratamento da Imagem Tratamento da Imagem - Filtros*. http://equipe.nce.ufrj.br/thome/p_grad/nn_img/transp/, 2010. Citado na página 51.

TORREAO, J. *Advances in Stereo Vision*. [S.l.]: InTech, 2011. ISBN 978-953-307-837-3. Citado 2 vezes nas páginas 14 e 45.

TWEED, D. Estimating rigid motions via the conformal model of euclidean space. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference - IEEE Xplore Digital Library*, v. 2, n. 8280081, p. 171 – 174, 2004. ISSN 1051-4651. Citado na página 26.

UCHIYAMA, T. Continuous path control of a 5-dof parallel-serial hybrid robot. *Journal of Mechanical Science and Technology*, v. 24, n. 47, p. 47–50, 2010. Citado na página 18.

URFALIOGLU, O. Robust estimation of camera rotation,translation and focal length at high outlier rates. *Computer and Robot Vision, 2004. Proceedings. First Canadian Conference - IEEE Xplore Digital Library*, v. 1, p. 464 – 471, 2004. Citado na página 34.

VISIONRT, C. *3d surface reconstruction. An overview of the key principles of the underlying imaging technology used by AlignRT, GateCT and GateRT*. [S.l.], 2009. Citado na página 45.

WANG, J. Analysis of a novel cylindrical 3-dof parallel robot. *Robotics and Autonomous Systems*, v. 42, n. 1, p. 31 – 46, 2003. ISSN 0921-8890. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0921889002002968>. Citado na página 18.

WANG, J. Computational model of stereoscopic 3d visual saliency. *IEEE Xplore Digital Library - Image Processing, IEEE Transactions*, v. 22, n. 13412760, p. 2151 – 2165, 2013. ISSN 1057-7149. Nenhuma citação no texto.

WEXLER, Y. Learning epipolar geometry from image sequences. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference*, v. 2, n. 7792762, p. II – 209–16, 2003. ISSN 1063-6919. Citado 2 vezes nas páginas 47 e 48.

XU, L. Stereo matching with optimal local adaptive radiometric compensation. *IEEE Signal Processing Letters*, v. 22, n. 2, p. 131–135, 2015. Citado na página 17.

YANG, L. Depth from water reflection. *IEEE Transactions on Image Processing*, v. 1, n. 1, p. 9, 2015. Nenhuma citação no texto.

YI, M. *An invitation to 3-D Vision from images to models*. [S.l.]: Universitat de València, 2004. Citado 11 vezes nas páginas 13, 22, 28, 29, 31, 32, 33, 45, 46, 47 e 48.

YLIMAKI, M. Fast and accurate multi-view reconstruction by multi-stage prioritised matching. *IET the institution of engeneering and technology*, v. 9, n. 576-578, p. 4, 2015. Nenhuma citação no texto.

YU, K. Stereo vision based robot navigation system using modulated potential field for implant surgery. *IEEE*, v. 1, n. 1, p. 6, 2015. Citado na página 17.

ZHANG, S. *Handbook of 3D Machine Vision*. [S.l.]: Hoboken: Taylor and Francis, 2013. ISBN 1-4398-7219-8. Citado 6 vezes nas páginas 13, 14, 15, 16, 45 e 47.

ZHOU, L. Extrinsic calibration of a camera and a lidar based on decoupling the rotation from the translation. *Intelligent Vehicles Symposium (IV), 2012 IEEE*, v. 4, n. 12849043, p. 642 – 648, 2012. ISSN 1931-0587. Citado na página 34.

ZIRAKNEJAD, N. Autonomous stereo camera parameter estimation for outdoor visual servoing. p. 157–162, 2007. Citado na página 19.

ZOU, W. Calibration of nonoverlapping in-vehicle cameras with laser pointers. *IEEE - Transaction on intelligent transportation system*, v. 1, n. 1, p. 12, 2014. Citado na página 17.