TRABALHO DE GRADUAÇÃO

RECONSTRUÇÃO ESPARSA EM 3D A PARTIR DE MÚLTIPLAS IMAGENS

LUCAS ARAGÃO BESSA

Brasília, 16 de dezembro de 2015.

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia Elétrica

TRABALHO DE GRADUAÇÃO

RECONSTRUÇÃO ESPARSA EM 3D A PARTIR DE MÚLTIPLAS IMAGENS

Lucas Aragão Bessa

Relatório submetido como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Banca Examinadora

Prof. Adolfo Bauchspiess, UnB/ ENE (Orientador)	_____
Prof. Francisco Assis de Oliveira Nascimento	_____
Prof. Cristiano Jacques Miosso Rodrigues Mendes	_____

Brasília, 16 de dezembro de 2015.

FICHA CATALOGRÁFICA

LUCAS ARAGÃO BESSA.
Reconstrução esparsa em 3D a partir de múltiplas imagens, [Distrito Federal] 2015. xvii, 67p., 297 mm (FT/UnB, Engenheiro, Elétrica, 2015). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1.Reconstrução 3D 1 2. visão computacional 2
3.Processamento de Imagens 4.Múltiplos Pontos de vista

I. Elétrica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

ARAGÃO BESSA, LUCAS, (2015). Reconstrução esparsa em 3D a partir de múltiplas imagens. Trabalho de Graduação em Engenharia Elétrica, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF.

CESSÃO DE DIREITOS

AUTOR: Lucas Aragão Bessa.

Reconstrução esparsa em 3D a partir de múltiplas imagens: Desenvolvimento de um sistema de visão computacional que processa imagens de múltiplos pontos de vista de um objeto e realiza reconstrução 3D esparsa.

GRAU: Engenheiro ANO: 2015

Lucas Aragão Bessa
SHIN QI 14 – Lago Norte.
70382-090 Brasília – DF – Brasil.

AGRADECIMENTOS

Agradeço a todos que me apoiaram para conclusão desse trabalho.

Lucas Aragão Bessa.

RESUMO

A reconstrução 3D de um objeto geralmente é realizada a partir de dois pontos de vista utilizando visão estéreo. Essa abordagem não permite visualizar o extensivo número de características de um objeto. Esse trabalho trata o tema de uma forma diferente empregando múltiplos pontos de vista ao invés de apenas dois. O principal objetivo desse projeto é reconstruir uma nuvem de pontos esparsa de um objeto aplicando múltiplos pontos de vista adquiridos em sequência com apenas uma câmera. O trabalho é realizado encontrando os principais pontos de interesse em uma imagem e realizando a correspondência com os pontos da imagem adjacente. Em sequência, os parâmetros de câmera são encontrados utilizando estas correspondências. Após isso, os pontos são reprojetados no espaço a partir dos parâmetros de câmera e por fim uma nuvem de pontos única é gerada a partir da unificação de pontos. O resultado obtido encontrou um erro de reprojeção médio de 2 pixels com uma quantidade de pontos insuficiente para ser utilizado em classificação das linhas de transmissão. Porém um dos objetos retém sua curvatura na reconstrução 3D. Trabalhos futuros podem melhorar o trabalho com nuvens de pontos densas e novos métodos de localização de pontos de interesse além de posicionamento mais preciso.

Palavras Chave: Reconstrução 3D, Visão Computacional, Processamento de Imagens, Múltiplos pontos de vista

ABSTRACT

The 3D reconstruction of an object is usually done with two views and using stereo vision. This approach lead to a subpar performance to visualize the extensive number of characteristics of an object. This work approaches the task in a different manner using multiple views instead of only two. The main goal of this project is to reconstruct a sparse point cloud of an object by means of multiple views. The work is done by finding the main key points in an image and matching them with the key points of adjacent images. Subsequently, the camera parameters are found with those matches. After that, the points are reprojected in space using the camera parameters. And lastly the point clouds are unified using bundle adjustment. The results obtained don't reproduce the objects evaluated with the similarity necessary to recognize them. But they reproduce characteristics that resemble the object. Future projects may increment this work with dense point clouds and new key point localization methods.

Keywords: 3D Reconstruction; Computer Vision; Image Processing; Multiple View Geometry

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

x_i	ponto na imagem
X_i	Ponto no espaço
l_i	Reta em Coordenadas Homogêneas
f	Distância Focal
C	Centro ótico da câmera
Pi	Plano de imagem
P_i	Matriz de câmera
K_{3x3}	Matriz de valores intrínsecos
R_{3x3}	Matriz de rotação
r_{ij}	Elementos da matriz de rotação
r_3	Última linha da matriz de rotação
I_3	Matriz Identidade
t_{3x1}	Vetor de Translação
p_x	Ponto principal da imagem no eixo x
p_y	Ponto principal da imagem no eixo y
m_x	Quantidade de pixels por mm na direção x
m_y	Quantidade de pixels por mm na direção y
x_0	Coordenadas do ponto principal em pixels na direção x
y_0	Coordenadas do ponto principal em pixels na direção y
e	epipolo
$\boldsymbol{l}$	reta epipolar
F	Matriz fundamental
F_{ij}	Elementos da matriz fundamental
T	Transformação de normalização para matriz fundamental
$\boldsymbol{U}$	Matriz unitária
$\boldsymbol{V}$	Matriz ortonormal
$\boldsymbol{S}$	Matriz diagonal contendo valores singulares
E	Matriz essencial
$L(x, y, \sigma)$	Espaço de escala
$G(x, y, \sigma)$	Variável de escala gaussiana
$I(x, y)$	Imagem
$D(x, y, \sigma)$	Diferença de Gaussianas

$m(x, y)$	Magnitude do gradiente do *keypoint*
$d(p, q)$	Distância euclidiana
a	ponto na árvore-kD
$\hat{X}_i$	Estimativa de projeção
$\hat{x}_i$	Estimativa de reprojeção
$R_x(\theta)$	Matriz de rotação em função do ângulo na direção x
$R_y(\theta)$	Matriz de rotação em função do ângulo na direção y
$R_z(\theta)$	Matriz de rotação em função do ângulo na direção z
v	matriz de rotação da forma *axis-angle*
k_{rot}	matriz de rotação da forma *axis-angle*

Símbolos Gregos

α_x	Distância focal da câmera em pixels no eixo x
α_y	Distância focal da câmera em pixels no eixo y
$\alpha_{médio}$	média entre a distância focal para os eixos x e y
$\theta(x, y)$	Direção do vetor de *keypoints*
τ	Método de transformação projetiva
H	Transformação projetiva

Grupos Adimensionais

Nu	Número de Nusselt
Re	Número de Reynolds

Subscritos

$médio$	médio
$caneca_{ij}$	referente à caneca
$cubo_{ij}$	referente ao cubo
$origami_{ij}$	referente ao origami

Sobrescritos

‘	Par

Siglas

CEO	Câmera Escura de Orifício

RANSAC	*Random Sample Consensus*
SIFT	*Scale-Invariant Feature Transform*
MATLAB	Matrix Laboratory
3D	Três dimensões
VANT	Veículo Aéreo Não-Tripulado
CCD	Charged-Coupled Device

1 INTRODUÇÃO

A visão é a habilidade de processar as informações que estão contidas na luz visível, interpretar, reconstituir e reconhecer objetos. O processo de visão se inicia quando a córnea e o cristalino focam os feixes de luz vindos do ambiente na retina que é a membrana fotossensível localizada no fundo do olho [17]. Neste ponto, as células especialistas em receber esses feixes de luz produzem impulsos elétricos e enviam sinais elétricos ao cérebro. Todo esse sistema permite aos seres humanos ver o mundo. Apesar de ser um processo natural, a visão é um dos mecanismos mais sofisticados realizados pelo corpo humano. Estima-se que a visão é responsável por dois terços da atividade elétrica do cérebro realizando entre dois e três bilhões de excitações por segundo. Vários pesquisadores tentam compreender como reproduzir esse sistema artificialmente a partir da visão computacional [4], [5], [9].

A visão computacional é uma área da inteligência artificial que objetiva a reconstrução do processo natural de extrair informações de imagens ou vídeos, sendo amplamente utilizada em aplicações do cotidiano.

Porém, por se tratar de um assunto muito complexo, as aplicações em visão computacional são divididas em sistemas especialistas que resolvem problemas de variados setores da sociedade que necessitam de análise rápida de informações visuais, quais sejam, por exemplo:

Trânsito: Controle em tempo real do tráfego nas ruas e sistemas de piloto automático que mantém o carro em uma mesma faixa de trânsito (Mobileye);

Esporte: Análise de cenas com rastreamento preciso da bola e análise de vídeo para coleta de estatísticas sobre jogos (Sportvision);

Medicina: Análise de imagens médicas e ferramentas de aprendizado médico com reconstrução 3D(CLmTec);

Indústria: Sistemas de inspeção e controle de processos em manufatura (RSIP Vision);

Agricultura: Controle, inspeção e mapeamento de plantações, classificação e separação das colheitas(Geosys);

Segurança: Sistemas biométricos de reconhecimento, monitoramento e rastreamento para câmeras de segurança (IBG).

A reconstrução da forma 3D de um objeto utilizando fontes 2D é o foco deste projeto dentro da área de visão computacional. Para se reconstruir a forma de um objeto é possível escolher

abordagens variadas. Como exemplo é possível citar estéreo, movimentação, sombreamento, texturas e *defocus*.[18]

Neste projeto será adotado o método de reconstrução 3D utilizando uma sequência de imagens adquiridas a partir do movimento de uma mesma câmera. O processo é similar à visão estéreo na qual é preciso encontrar pontos correspondentes entre diferentes imagens para então projetá-los em coordenadas em três dimensões.

O método é vantajoso considerando que a quantidade de informações adquiridas durante a sequência de imagens permite uma reprodução mais fidedigna do objeto original. Porém o elevado tempo de processamento torna as aplicações em tempo real difíceis de serem reproduzidas

A motivação original para construir esse tipo de sistema veio da dissertação de mestrado do Doutor Elder Oroski que utilizava visão estéreo e redes neurais para identificar falhas em linhas de transmissão [1]. Para esse trabalho, esse conceito será expandido para adquirir mais pontos de interesse dos espaçadores.

1.1 Objetivos

- O objetivo geral do trabalho é fazer um sistema de reconhecimento de falhas em linhas de transmissão a partir imagens adquiridas por um Veículo Aéreo Não Tripulado (VANT).
- O objetivo específico do trabalho é desenvolver um sistema capaz de receber imagens de um objeto sob diversos pontos de vista e reconstruir uma nuvem de pontos esparsa que reproduz a forma desse objeto. Posteriormente, o sistema pode ser adaptado para identificar falhas nas linhas de transmissão utilizando a nuvem de pontos adquiridas pelo algoritmo produzido nesse projeto.

1.2 Organização do trabalho

Serão apresentados cinco capítulos.

O segundo capítulo detalha os fundamentos matemáticos e os conceitos para o entendimento do projeto como um todo, apresenta o conceito de geometria projetiva, expõe os modelos de câmeras adotados em algoritmos de reconhecimento e também as relações entre imagens que possibilitam a reconstrução 3D. Aponta ainda como é realizada a extração

dos pontos de interesse das imagens e como é feita a calibração da câmera. E por fim como se unem todas as nuvens de pontos e um único conjunto de pontos.

A exposição sobre a implementação do sistema encontra-se no capítulo três, juntamente com o passo a passo desde a calibração do sistema, passando pelas funções que implementam a extração de pontos de interesse, até o final onde temos a extração final da nuvem de pontos esparsa do objeto.

O quarto capítulo apresenta os resultados experimentais obtidos em cada etapa da implementação do projeto comparados aos resultados esperados durante o processo de desenvolvimento do sistema.

No último capítulo encontra-se a parte final do trabalho e apresenta a conclusão de tudo o que foi realizado durante o projeto. Além disso, também informa quais são possíveis melhorias para o sistema construído e o que poderá ser abordado em trabalhos futuros.

2 FUNDAMENTOS

Neste capítulo serão apresentados os conceitos matemáticos de geometria projetiva necessários para realizar a construção 3D e os modelos de câmera mais apropriados para esta técnica, incluindo a câmera CCD utilizada no desenvolvimento do projeto. Além disso será descrita qual a relação entre as câmeras, como utilizar essa relação para definir os pontos de interesse e a triangulação necessária para a reconstrução 3D. Por fim será explicado como funciona a unificação dos pontos que permite unir todas as nuvens de pontos para apenas uma.

2.1 GEOMETRIA PROJETIVA

A geometria euclidiana é amplamente utilizada no cotidiano por descrever o mundo real em três dimensões com precisão, porém não descreve de forma satisfatória as relações para o processamento de imagens 2D [4].

Em uma câmera, tamanhos e ângulos não são preservados e retas paralelas podem se intersectar, fatos que não ocorrem no mundo real. A geometria euclidiana pode ser descrita como um subconjunto da geometria. Outro subconjunto será o foco dessa seção, a geometria projetiva [4].

O estudo da geometria projetiva se iniciou há mais de quinhentos anos quando artistas começaram a utilizar a perspectiva em seus quadros. Eles notaram que as distâncias medidas em um quadro não representavam as distâncias euclidianas [19].

A geometria projetiva é o estudo das propriedades descritivas das figuras geométricas que são invariantes às transformações projetivas. Isso significa que ela não segue os mesmos princípios básicos da geométria euclidiana pois permite uma gama maior de transformações. Porém, a geometria projetiva não pode ser amplamente utilizada porque medidas como distância, ângulo, paralelismo não são preservadas. Exemplo de medidas conservadas em transformações projetivas são rotação, translação, escala uniforme e projeção de perspectiva. Portanto, as transformações projetivas são necessárias [2] para o processo de mapeamento de um ponto em três dimensões para um plano em duas dimensões.

O sistema de coordenadas homogêneas é utilizado na geometria projetiva. Esse sistema adiciona uma dimensão ao espaço euclidiano, ou seja, um ponto no espaço projetivo de dimensão N é representado por um vetor de tamanho $N + 1$. A dimensão adicionada terá um valor constante em relação ao espaço euclidiano e será um múltiplo comum entre as outras

dimensões [3]. Por isso, em coordenadas homogêneas os pontos de duas e três dimensões serão respectivamente:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \begin{bmatrix} kx_1 \\ kx_2 \\ k \end{bmatrix} \tag{2.1}$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix} = \begin{bmatrix} kX_1 \\ kX_2 \\ kX_3 \\ k \end{bmatrix}. \tag{2.2}$$

Dois vetores com $n+1$ elementos x_n e y_{n} representam o mesmo ponto se e somente se existir um escalar não nulo λ tal que $x_i = \lambda y_i$ para $1 \leq i \leq n+1$. Isso mostra que a correspondência entre pontos no espaço projetivo não é de um para um [3].

O ponto foi o primeiro elemento matemático do espaço projetivo a ser descrito. A reta é outro elemento fundamental.

As retas em um plano euclidiano podem ser representadas pela equação $ax_1 + bx_2 + c = 0$. Considerando que o ponto no espaço de duas dimensões é representado por $x = [x_1\ x_2]$, podemos concluir que a reta pode ser descrita por

$$x^T l = 0, \tag{2.3}$$

sendo $l = (a\ b\ c)$. Toda reta que satisfaz essa equação será uma reta no plano projetivo.

Segundo a geometria euclidiana, a intersecção de duas retas é um ponto. O mesmo conceito ocorre na geometria projetiva. A intersecção é calculada pelo produto vetorial entre duas retas

$$l_1 \times l_2 = x. \tag{2.4}$$

Por último, outra definição análoga a geometria euclidiana é aquela pela qual dois pontos definem uma reta. Portanto, na geometria projetiva temos que dois pontos pertencentes a uma linha definem a reta de forma que

$$x_1 \times x_2 = l. \tag{2.5}$$

2.2 MODELO UTILIZANDO UMA CÂMERA

Uma câmera é o mapeamento entre o mundo 3D e uma imagem 2D. A primeira câmera a ser explicada nesse projeto representa o modelo ideal, a Câmera Escura de Orifício (CEO).

2.2.1 Modelo Geométrico

A câmera escura de orifício é representada por uma caixa fechada com um furo em uma de suas faces. A luz entra por esse orifício e é projetada na face oposta da caixa, sofrendo inversão.

Na figura 2.1 é possível observar esse processo. **{tk} verificar imagem**

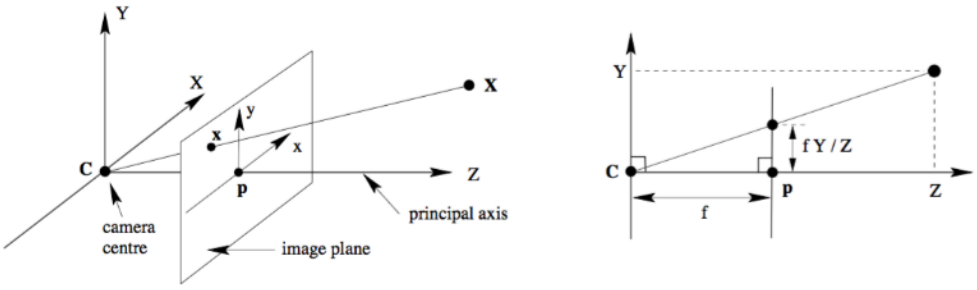

Figura 2.1 Principais objetos matemáticos de um modelo de câmera (Fonte: [4])

O modelo geométrico que é mostrado na figura acima consiste dos seguintes objetos matemáticos:

- A distância f que representa a distância focal entre a câmera e o plano de imagem;
- O ponto C é o centro ótico ou centro da câmera, localizado a uma distância f do plano de imagem P;
- O plano de imagem Pi que representa as coordenadas $z = f$.

O modelo apresentado pode modelar com precisão sistemas de câmeras modernos.

2.2.2 Matriz de Câmera

A luz refletida de objetos em um ambiente em três dimensões é capturada pela câmera e forma uma imagem em duas dimensões. Essa é uma transformação não-linear no espaço euclidiano.

A geometria projetiva é ideal para trabalhar com esse tipo de transformação. O uso de coordenadas homogêneas permite que essa projeção não-linear no espaço euclidiano seja representada por uma equação linear da forma [4]:

$$\begin{bmatrix} x_1 \\ x_2 \\ w \end{bmatrix} = P_{3x4} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ T \end{bmatrix}, \tag{2.6}$$

$$x = PX. \tag{2.7}$$

O vetor $\boldsymbol{x}$ representa as coordenadas homogêneas em duas dimensões que são um ponto na imagem. Já o vetor $\boldsymbol{X}$ é um ponto no espaço em três dimensões e P_{3x4} é a matriz de câmera ou matriz de projeção.

Para que seja possível utilizar a Eq. 2.8 é necessário adquirir os parâmetros da matriz de câmera que quando generalizada pode ser descrita por:

$$P = K_{3x3} R_{3x3} [I_3 | \, \mathrm{t}_{3x1}], \tag{2.8}$$

Onde K é um valor específico para cada câmera (nesse caso *pinhole*) que representa os parâmetros internos ou intrínsecos da câmera. R e t são relacionados à orientação e posição da câmera em relação ao exterior e portanto são chamados de parâmetros externos ou extrínsecos. Segue a Fig. 2.2 que representa essa transformação.

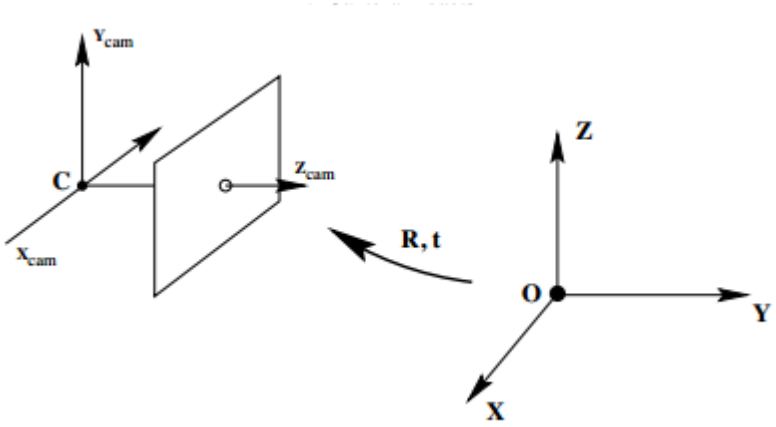

Figura 2.2 Parametrização de um sistema de coordenadas do mundo real para um sistema de coordenadas calibrado com os valores da câmera. (Fonte: [4])

Para cálculos utilizando o modelo *Pinhole* os parâmetros intrínsecos são representados por

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.9}$$

Onde $p_x e\ p_y$ são as coordenadas do ponto principal que é o centro do plano de imagem e f é a distância focal para a câmera utilizada.

Quando utilizada a câmera digital é necessário fazer adaptações à matriz de parâmetros intrínsecos. Nesse tipo de câmera há possibilidade de existir pixels não quadrados e é preciso adicionar fatores de escala em cada direção.

Modelo CCD:

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

onde $\alpha_x = f m_x$ e $\alpha_y = f m_y$ representam a distância focal da câmera em termos de dimensão de pixels nas direções x e y respectivamente. m_x e m_y representam a quantidade de pixels por unidade de distância nas direções x e y respectivamente. Assim como no modelo *Pinhole* (x_0, y_0) são as coordenadas do ponto principal. Sendo $x_0 = m_x p_x$ e $y_0 = m_y p_y$. A câmera CCD possui dez parâmetros.

Os valores dos parâmetros intrínsecos podem ser encontrados no processo de calibração da câmera que será descrito adiante no capítulo.

Para concluir a descrição da matriz de câmera é necessário abordar os parâmetros externos. R é uma matriz 3x3 que representa a matriz de rotação da câmera. Ela fornece informação sobre as coordenadas do mundo em relação a câmera. t é um vetor coluna 3x1 que fornece a posição da origem das coordenadas do mundo em relação à câmera.

Para estimar os valores de R e t é preciso adicionar informações sobre outros pontos de vista.

2.3 MODELO UTILIZANDO DUAS OU MAIS CÂMERAS

Na seção anterior foi descrito um sistema com uma câmera. Nessa seção será descrito como extrair as informações de dois pontos de vista para conseguir completar as informações sobre a matriz de câmera. Essa seção irá descrever a geometria epipolar e a partir dela derivar a matriz fundamental. Em seguida, utilizando os parâmetros intrínsecos das câmeras será possível derivar a matriz essencial e por fim obter os parâmetros externos de cada ponto de vista.

2.3.1 Geometria Epipolar

A geometria epipolar é a geometria que descreve as relações projetivas entre duas imagens [4]. Ela independe de um sistema de coordenadas global, sendo derivada apenas dos parâmetros intrínsecos e da posição relativa entre as duas câmeras.

A geometria epipolar é motivada pela busca de pontos correspondentes entre duas imagens. Dado ponto x em uma imagem representa o ponto X no espaço. As relações matemáticas da geometria epipolar permitem encontrar um ponto x' em outra imagem que representa o mesmo ponto X no espaço.

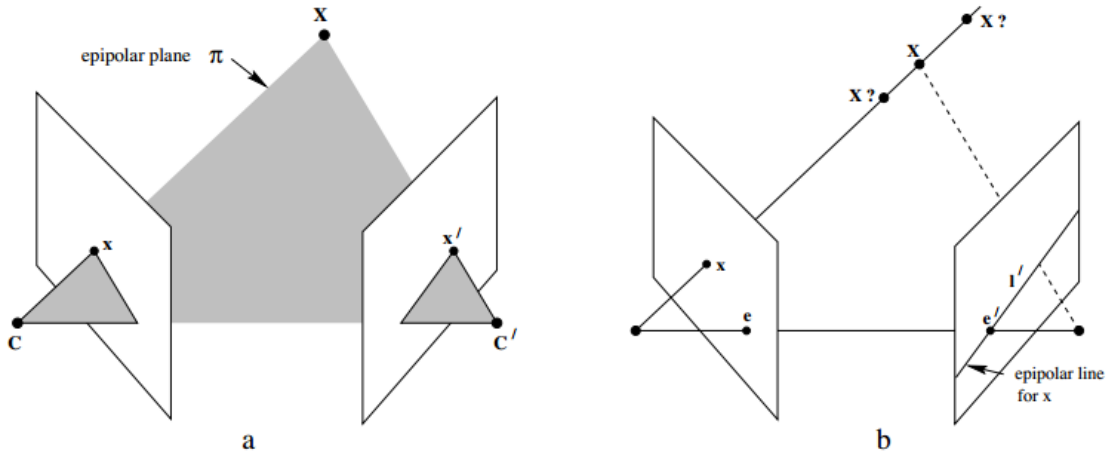

Figura 2.3 (a) Plano epipolar e como ele está relacionado com os pontos correspondentes nas duas imagens (b) Epipolos e as retas epipolares de pontos correspondentes. (Fonte: [4])

A figura 2.3 (a) mostra que os pontos x e x' são coplanares pois os raios projetados dos pontos saindo dos centros de câmera C e C' se encontram no mesmo ponto X. Esse plano denomina-se Plano Epipolar.

Na figura 2.3 (b) o ponto e que é definido pela intersecção entre o segmento de reta que liga os centros de câmera e os planos da imagem denomina-se epipolo. Já os segmentos de

reta que ligam o ponto x com o epipolo e e o ponto x' com o epipolo e' são chamadas de retas epipolares($\boldsymbol{l}$ e $\boldsymbol{l}'$).

2.3.2 Matriz fundamental (F)

A matriz fundamental é a representação algébrica da geometria epipolar [4]. Ela é uma matriz 3x3 que descreve as relações matemáticas entre dois pontos correspondentes para dois pontos de vista. Para cada par de pontos (x, x') existe um par de epipolos (e,e') que são ligados por um par de retas epipolares($\boldsymbol{l}, \boldsymbol{l}'$) . O mapeamento entre esses pontos é representado pela matriz fundamental.

Esse mapeamento expressa que a matriz fundamental (F) deve satisfazer a seguinte condição [4]:

$$x'^{T} F x = 0. \tag{2.11}$$

Essa expressão permite que a matriz fundamental seja independente da matriz de câmera, possibilitando que a matriz seja calculada partindo apenas das correspondências entre imagens. A interpretação geométrica para essa equação será a de que x'está é parte da reta epipolar Fx e isso ocorre apenas quando a matriz fundamental F mapeia as retas epipolares.

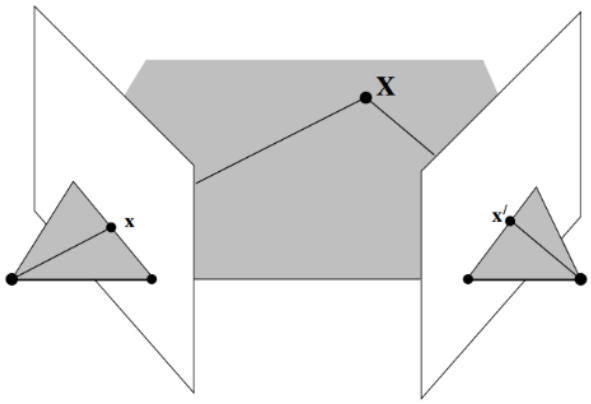

Figura 2.4 A imagem mostra o mapeamento dos pontos nas imagens para um só ponto no espaço.(Fonte: [4])

2.3.2.1 Cálculo da Matriz Fundamental

A matriz fundamental é uma matriz 3x3 homogênea de posto dois que tem oito graus de liberdade. Apesar de possuir nove elementos sua determinante é nula. Esse fato remove um grau de liberdade. Portanto, o cálculo da matriz fundamental a partir dos pontos correspondentes entre as imagens será feito pelo algoritmo de oito pontos normalizados [4]. Esse algoritmo determina a matriz fundamental a partir de oito ou mais pontos correspondentes. É possível aplicar esse algoritmo em quatro passos.

O primeiro passo é a normalização. Para realizá-la é preciso transformar as coordenadas de x_i e x_i' para Tx_i e $T'x_i'$ respectivamente, onde T e T' são transformações de normalização que consistem em mudanças de escala e translação. Essa transformação muda o centroide dos pontos para origem e transforma a distância média entre os pontos para $\sqrt[2]{2}$.

O segundo passo é a solução linear. A partir da equação (2.11) é possível escrever um conjunto de no mínimo oito equações que resolvem linearmente a matriz fundamental. Os pontos $x_i = [x_i, y_i, 1]^T$ e $x_i' = [x_i', y_i', 1]^T$ formam o i-ésimo par de correspondências e $f = [F_{11}\ F_{12}\ F_{13}\ F_{21}\ F_{22}\ F_{23}\ F_{31}\ F_{32}\ F_{33}]^T$ é o vetor que contém os elementos da matriz fundamental. Com isso, pode-se deduzir a equação

$$A = \begin{pmatrix} x_1'x_1 & x_1'y_1 & x_1' & y_1'x_1 & y_i'y_1 & y_1' & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n'x_n & x_n'y_n & x_n' & y_n'x_n & y_n'y_n & y_n' & x_n & y_n & 1 \end{pmatrix}$$

$$\boldsymbol{Af} = 0 \tag{2.12}$$

A solução é extraída por SVD(*Singular Value Decomposition*). Aplicando SVD em A resulta na decomposição $\boldsymbol{USV^T}$ sendo $\boldsymbol{U}$ e $\boldsymbol{V}$ matrizes ortonormais e $\boldsymbol{S}$ uma matriz diagonal contendo os valores singulares. Os valores serão positivos e em ordem decrescente. A matriz fundamental será a coluna de $\boldsymbol{V}$ que corresponde ao menor valor singular.

O terceiro passo é forçar a limitação de $\det F = 0$. A coluna de $\boldsymbol{V}$ será a matriz fundamental num caso ideal, porém num caso real com presença de ruído é preciso limitar a matriz para que ela possua posto 2. Essa limitação é feita com *SVD*. A matriz é decomposta e reconstruída com os dois maiores valores singulares.

O quarto e último passo é a denormalização que é necessária para voltar os pontos correspondentes aos dados iniciais. Isso é realizado com a equação:

$$F = T'F\,T \tag{2.13}$$

2.3.3 Matriz essencial (E)

A matriz essencial é um caso particular da matriz fundamental em que as coordenadas da imagem estão normalizadas com os parâmetros intrínsecos da câmera. A matriz pode ser encontrada segundo [4]

$$E = K'^{T} F K. \tag{2.14}$$

A matriz essencial possui apenas cinco graus de liberdade. A partir da matriz essencial é possível calcular os termos finais das matrizes de câmera.

2.3.4 Cálculo da matriz de câmera

O cálculo da matriz de câmera depende da obtenção do vetor de translação t e da matriz de rotação R. Para encontrá-los será utilizada a seguinte relação:

$$E = [t]_x R \tag{2.15}$$

A decomposição da matriz essencial atrelada a Eq. 2.15 pode ser reescrita como $E = SR$. Onde S é a matriz antissimétrica que tem t como vetor nulo à direita [8].

A decomposição de S segundo [4] pode ser feita como $S = UZU^T$. Em que U é a matriz ortogonal da decomposição de valores singulares de E e

$$Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

que também pode ser escrito como

$$Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$Z = WD.$$

Unindo as equações descritas nessa seção temos:

$$SR = UDWU^T R \tag{2.16}$$

O que implica segundo [4] que fazendo a decomposição em valores singulares de E utilizando a mesma notação, R só poderá ser dois valores:

$$R_1 = UW^TV^T e\ R_2 = UWV^T$$

E t que é o vetor nulo à direita de S também só poderá ter dois valores:

$$t_1 = u_3\ e\ t_2 = -u_3$$

Sendo u_3 a última coluna da matriz U.

Por fim, a fórmula para matriz de câmera será [4]:

$$P' = K[UWV^T|u_3]\ ou\ K[UWV^T|-u_3] ou\ K[UW^TV^T|+u_3]\ ou\ K[UW^TV^T|+u_3] \qquad (2.17)$$

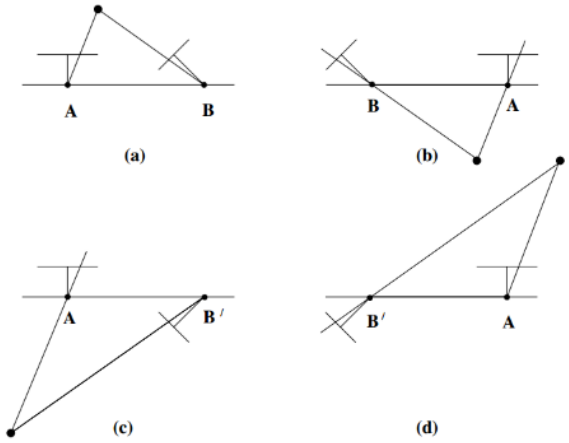

Figura 2.5 Configuração de cada possível solução de R, t referentes a um E. (Fonte: [4])

Agora que R e t foram adquiridos é preciso verificar qual combinação de valores é a melhor para ser utilizada. A melhor combinação será escolhida a partir da reprojeção dos pontos. Os pontos serão reprojetados no espaço e comparados com cada uma das alternativas para R e t.

A comparação será feita partindo da premissa de que os pontos estão na frente da câmera. O ponto reprojetado X estará na frente da câmera caso [4]

$$(X + R^T t) * r_3 > 0$$

Sendo r_3 a última linha da matriz R.

A combinação que possuir mais pontos na frente da câmera irá ser escolhida.

2.4 DETECÇÃO DE CARACTERÍSTICAS

O processo de detecção de características é fundamental para análise de imagens que demandam grande poder de processamento. Localizar os pontos mais importantes alivia o gasto computacional devido a redução da quantidade de dados a serem analisados após esse passo.

Os pontos de interesse são aqueles que permitem descrever as características de um objeto mesmo sem qualquer outra informação.

O processo de extração de características pode ser abordado com diferentes métodos. Porém, o ideal é que os pontos extraídos possam ser detectados em outras imagens mesmo com mudanças de escala, níveis de ruído e iluminação.

O projeto detalhará o algoritmo de detecção e descrição de características locais SIFT (*Scale-Invariant Feature Tranform*) criado por David Lowe em 1999 [5]. Tal algoritmo é amplamente utilizado no campo de reconstrução 3D pois ele consegue extrair pontos que em conjunto com um vetor de características locais descrevem uma imagem com certa precisão. São vetores invariáveis: translação, escala, rotação; e parcialmente invariáveis a mudanças de iluminação. Também são robustos à presença de distorção geométrica. Segundo Lowe [5], tais propriedades são similares às respostas dos neurônios no córtex temporal inferior na visão dos primatas.

O algoritmo SIFT extrai os pontos de interesse da imagem e gera um conjunto de vetores que descreve numericamente as propriedades da região em volta desses pontos. Esses vetores podem ser usados posteriormente para fazer as correspondências de características. Nas seções a seguir será detalhado esse processo.

2.4.1 Identificação de pontos de interesse

As propriedades invariáveis à escala são conseguidas através do emprego de uma filtragem em estágios. O primeiro estágio identifica localidades importantes, chamadas *keypoints*, no espaço de escala. Essa filtragem busca por locais que são máximos ou mínimos de uma diferença-de-gaussianas. O espaço de escala da imagem é definido como $L(x, y, \sigma)$, e é produzido pela convolução entre a variável de escala Gaussiana,$G(x, y, \sigma)$, com a imagem, $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.18}$$

onde $*$ é a convolução e

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.19}$$

A diferença de gaussianas entra como uma alternativa computacionalmente menos exigente do que o laplaciano de gaussianas. O uso do espaço de escala gaussiano permite a reutilização dos valores já computados das gaussianas e com isso D será apenas uma subtração.

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y)$$
$$= L(x, y, k\sigma) - L(x, y, \sigma).$$

O cálculo pode ser feito eficientemente construindo uma pirâmide de imagens com reamostragem para suavização em cada nível. Esse método torna flexível a mudança de escala pois as características buscadas devem ser estáveis em todas as escalas.

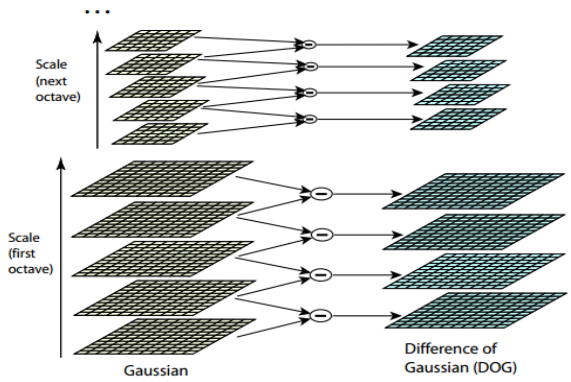

Figura 2.6 - Para cada escala a imagem é repetidamente convolucionada com gaussianas para produzir o conjunto de imagens mostradas à esquerda. As imagens são subtraídas para produzir as diferenças de gaussianas e reamostradas por um fator de 2. (Fonte: [6])

Para a determinação dos mínimos ou máximos locais cada ponto é comparado aos seus oito vizinhos na mesma escala e também a seus dezoito vizinhos nas escalas superiores e inferiores, conforme a figura abaixo (2.7).

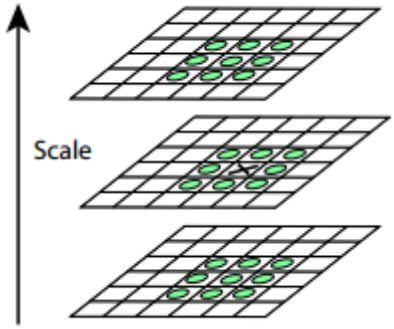

Figura 2.7 - Cada ponto é comparado a seus 26 vizinhos. Sendo 8 na mesma escala e 18 nas escalas superior e inferior.(Fonte: [6])

2.4.2 Localização e filtragem de *Keypoints*

Após a seleção dos *keypoints* é preciso realizar uma localização, ou seja, adequar os pontos coletados a região próxima. Essa nova seleção irá rejeitar pontos que apresentam baixo contraste e são localizados em bordas. A diferença de gaussianas gera pontos que estão mal colocados em bordas instáveis que são suscetíveis a ruído.

Para eliminar os pontos de baixo contraste é utilizada a expansão de Taylor até o elemento quadrático da função $D(x, y, \sigma)$ transladada tal que a origem é o ponto a ser avaliado.

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{x} x^T \frac{\partial^2 D}{\partial x^2} x \tag{2.20}$$

D é avaliada no ponto testado e $x = (x, y, \sigma)^T$ é o offset deste ponto. Para encontrar o máximo é necessário derivar e igualar a zero. O algoritmo utiliza o offset maior que 0.5 como sendo o limite área aquele *keypoint* específico. Caso o valor seja maior que esse, o máximo provavelmente estará em outro ponto.

$$\hat{\mathrm{x}} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x}$$

Ao substituir uma equação na outra temos,

$$D(\hat{\mathrm{x}}) = D + \frac{1}{2} \frac{\partial D^T}{\partial x} \hat{\mathrm{x}}$$

Lowe [6] utiliza valores menores que 0.03 para serem descartados.

Figura 2.8 – Pontos registrados nos 3 passos. A primeira imagem registra os pontos após a aplicação da diferença de gaussianas. A segunda imagem mostra os *keypoints* após a rejeição de pontos de baixo contraste. A terceira imagem mostra os pontos restantes após a retirada daqueles localizados em bordas instáveis.(Fonte: [16])

2.4.3 Atribuição de direção

Para que os *keypoints* sejam inalteráveis a rotação de ver dada uma direção ao vetor. A direção do vetor é dada a partir de:

$$m(x,y) = \sqrt{\left(\left(L(x+1,y) - L(x-1,y)\right)^2 + \left(L(x,y+1) - L(x,y-1)\right)^2\right)},$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))).$$

Sendo que L é a imagem com escala mais próxima do *keypoint*. e $m(x,y)$ será a magnitude gradiente do ponto, e sua orientação $\theta(x,y)$.

2.4.4 Keypoint Descriptors

O último passo para o algoritmo SIFT é a criação do *keypoint descriptors*. O *descriptor* é um conjunto de vetores que descreve numericamente as propriedades em torno dos pontos de interesse. Esse passo utiliza a região em volta do keypoint para encontrar valores que descrevem essa região. Esse *descriptor* pretende deixar o sistema robusto para alterações de iluminação e ponto de vista tridimensional.

O *keypoint descriptor* é calculado a partir do gradiente de magnitude e orientação dos pontos em volta dos keypoints para a imagem suavizado do espaço de escala mais próxima em uma região de dezesseis por dezesseis pixels. Os vizinhos são unidos em regiões de quatro por quatro pixels e os gradientes somados para encontrar os valores em oito

direções. Isso resulta em 4x4x8 = 128 vetores de características para cada ponto de interesse. Os vetores representam o histograma de cada região e para uma mudança suave entre regiões é realizada uma interpolação tri linear que ajuda na distribuição dos valores entre os histogramas do ponto, ou seja, os vetores são multiplicados por um peso de $(1 - d)$ onde d é igual a distância até o ponto central do descriptor. Para melhorar ainda mais a performance para variação de iluminação, o descriptor é normalizado. Essa descrição pode ser percebida na Fig. 2.9.

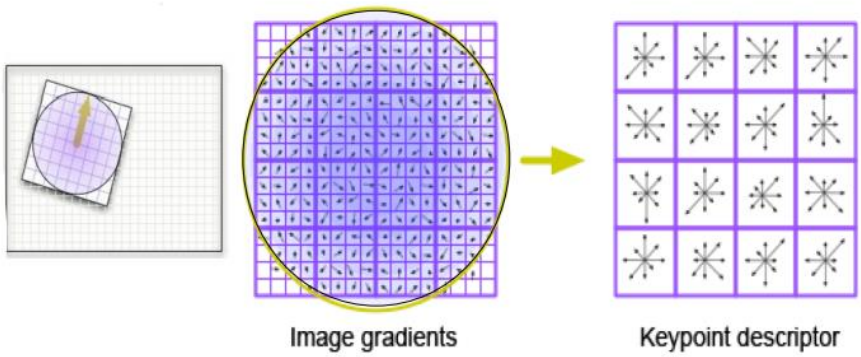

Figura -2.9 – Processo de construção dos keypoint descriptors. (Fonte: Jonas Hurreimann)

2.5 CORRESPONDÊNCIA DE CARACTERÍSTICAS

Com a obtenção dos *keypoint descriptors* e da localização dos pontos de interesse, o próximo passo é comparar as características entre os descriptors encontrados para encontrar quais serão os pontos correspondentes. [6]

Para encontrar os pontos correspondentes será utilizada a distância euclidiana entre os vetores. Como visto na seção anterior, os *keypoint descriptors* são vetores com cento e vinte e oito elementos. A fórmula para a distância euclidiana em n-dimensões é

$$d(p,q) = \sqrt[2]{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2 \dots (p_n - q_n)^2}$$

Para um vetor de cento e vinte e oito dimensões será:

$$d(p,q) = \sqrt[2]{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2 \dots (p_{128} - q_{128})^2}. \qquad (2.21)$$

Medir distâncias com vetores de alta dimensionalidade para todos os pontos torna-se um grande gasto computacional para o projeto. Por esse motivo será implementado um método de indexação que permite reduzir o custo computacional. O método denomina-se árvore-kD [11].

A árvore-kD é um tipo de estrutura de dados que guarda e organiza um conjunto finito de pontos de alta dimensionalidade na forma de árvore binária de busca. Cada nó da árvore representa uma dimensão que divide o hiperplano em dois lados. Os pontos à esquerda serão representados por uma sub-árvore e os pontos à direita também até que sejam esgotadas as dimensões do ponto. Para escolher o lado dos pontos em cada subdivisão deve ser seguir o seguinte padrão:

Para um nó que representa a dimensão i do ponto $a = [a_1\ a_2\ a_3 \dots a_i \dots a_n]$. Estarão a esquerda do nó os valores que forem menores que a_ie estarão à direita do nó os valores que forem maiores que a_i. Essa divisão se repete até a n-ésima dimensão.

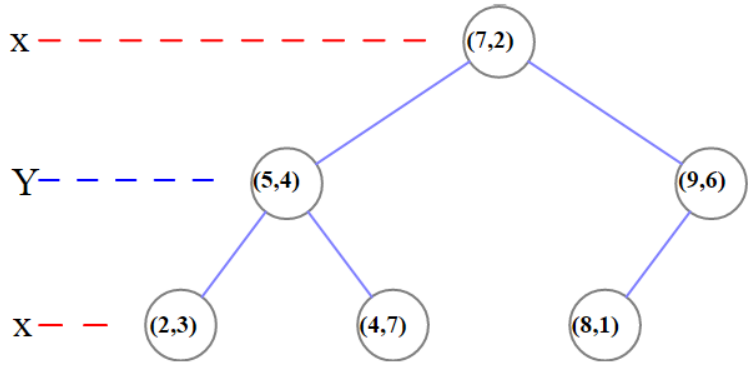

Figura 2.10 – Método de divisão para a estrutura de dados árvore kD em um conjunto de pontos bidimensionais.

Para comparar dois pontos é feita a busca dentro da árvore seguindo da raiz até a última folha para encontrar qual será o vetor mais próximo. Esse algoritmo de busca denomina-se “busca de vizinho mais próximo” (*Nearest Neighbour search*). Quando a busca parar na última folha da árvore, é calculada a distância euclidiana somente para os pontos mais próximos e o ponto com a menor distância é escolhido como o ponto correspondente na outra imagem.

Esse processo reduz amplamente o gasto computacional na busca pelos pontos correspondentes.

2.6 FILTRAGEM DE *OUTLIERS* COM RANSAC

A correspondência de características encontra pontos similares entre as imagens, porém os pontos similares podem não representar o mesmo ponto no espaço. Caso isso ocorra, a estimativa da matriz fundamental pode não ser precisa para o conjunto de imagens. Por isso é necessário utilizar um algoritmo que filtra os pontos que durante a correspondência pareciam iguais, porém não são.

O algoritmo *Random Sample Consensus* (RANSAC) é um algoritmo iterativo que estima parâmetros de um modelo matemático a partir de um conjunto de dados que contém *outliers*, ou seja, dado um conjunto de dados experimentais o algoritmo iterativamente ajusta um modelo matemático que inclui os pontos que estão de acordo com os dados e exclui os pontos que não estão de acordo com esse modelo [10].

Diferente de outros algoritmos de ajuste de dados que tentam trabalhar com os dados como um todo, o RANSAC inicia o algoritmo com um subconjunto pequeno de dados que são selecionados aleatoriamente.

Após a seleção inicial dos dados um modelo inicial é ajustado para esses elementos sorteados. Esse modelo é testado para todos os pontos do conjunto utilizando o desvio padrão. Caso existam pontos suficientes do conjunto que se encaixam no modelo, se inicia o próximo passo. Se o modelo não for bom o suficiente, serão sorteados novos pontos. Segundo [10] a quantidade de vezes que um subconjunto de pontos deve ser sorteado aleatoriamente é trinta e seis vezes. Essa quantidade garante com probabilidade de 90% que será escolhido um subconjunto sem *outliers.* Caso não seja possível encontrar um modelo satisfatório em trinta e seis tentativas o algoritmo termina em falha.

Com um número suficiente de pontos que se encaixam no modelo o algoritmo tenta reduzir o erro com o método dos mínimos quadrados. Todos os pontos que tem um erro maior que o limite imposto são considerados *outliers* e são filtrados do conjunto.

2.7 CALIBRAÇÃO DA CÂMERA

Como mencionado no início do capítulo, considerando que não são conhecidos os parâmetros internos da câmera, é necessária a calibração da câmera para conhecer os valores das matrizes de câmera, os quais dependem principalmente dos seguintes fatores:

distância focal, coordenadas do ponto principal, fator de distorção e distorção da lente. Sendo que nesse trabalho serão aferidos apenas a distância focal e as coordenadas do ponto principal em pixels.

Uma boa calibração é importante, pois sem os parâmetros medidos com acurácia o erro será propagado e ao final do processo de reconstrução, o objeto não terá a forma semelhante ao real.

Nesse projeto será utilizada uma câmera CCD e, portanto, a matriz de valores intrínsecos será igual à da Eq. 2.22. A presença de dois valores de distância focal nessa matriz indica a premissa de que os pixels não são quadrados. Esse fato ocorre devido aos diversos processos de compressão e transmissão que a imagem sofre desde a captura até ser analisada no monitor do computador. Para calcular a distância focal em pixels será utilizada a equação

$$\alpha_x = fm_x \; e \; \alpha_y = fm_y. \tag{2.22}$$

A distância focal em mm é fornecida pelo fabricante da câmera. A quantidade de pixels por unidade de distância será verificada comparando o tamanho do pixel na imagem com o tamanho do pixel para uma distância real medida.

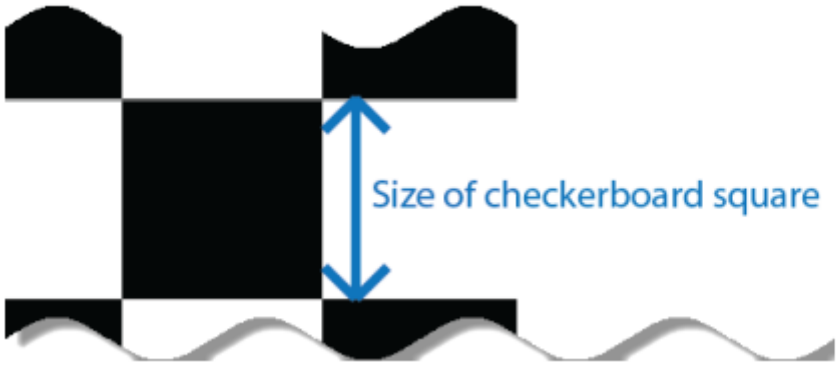

Figura 2.11 A medição do tamanho do padrão é feita em mm e em pixels para encontrar o tamanho de cada pixel. (Fonte: Camera Calibration Toolbox – MATLAB)

Utilizando as medições baseadas na Fig. 2.11. A equação para medir isso será:

$$m_x = \frac{checkerboard_x(mm)}{checkerboard_x(pixels)} \; e \; m_y = \frac{checkerboard_y(mm)}{checkerboard_y(pixels)}.$$

Agora que a distância focal em pixels foi aferida é preciso verificar o local do ponto principal(x_0, y_0) na imagem em pixels. Será assumido que o ponto principal da imagem é o centro da imagem, portanto o ponto principal é:

$$x_0 = m_x p_x \; e \; y_0 = m_x p_x.$$

Como m_x e m_y foram calculados no passo anterior, temos as coordenadas dos pontos principais.

2.8 RECONSTRUÇÃO 3D

Para realizar a reconstrução 3D dos pontos correspondentes será utilizado o “Teorema da reconstrução projetiva” que segundo [4] diz: Se um conjunto de pontos correspondentes determinam uma matriz fundamental única, então a cena e a câmera podem ser reconstruídas dessas correspondências. Caso aferidos os parâmetros de câmera essa projeção poderá ser escalada para as coordenadas do mundo.

Partindo da Eq. 2.6 e 2.11 e dadas as matrizes de câmera será possível reconstruir a os pontos dado que:

$$x_i = PX_i \; e \; x_i' = P'X_i \; para \; todo \; i$$

$$x_i' F x_i = 0$$

Sendo x_i e x_i' pontos correspondentes na duas imagens. Essas serão as amarras impostas pelo problema. As amarras epipolares impõem que os raios projetados a partir do centro de câmera (C) passando pelos pontos x_i e x_i' se intersectam somente em um ponto. Isso ocorre pois as retas estão no mesmo plano epipolar.

Para um caso real, os raios projetados não irão se intersectar devido a ruídos inerentes aos sistemas reais.

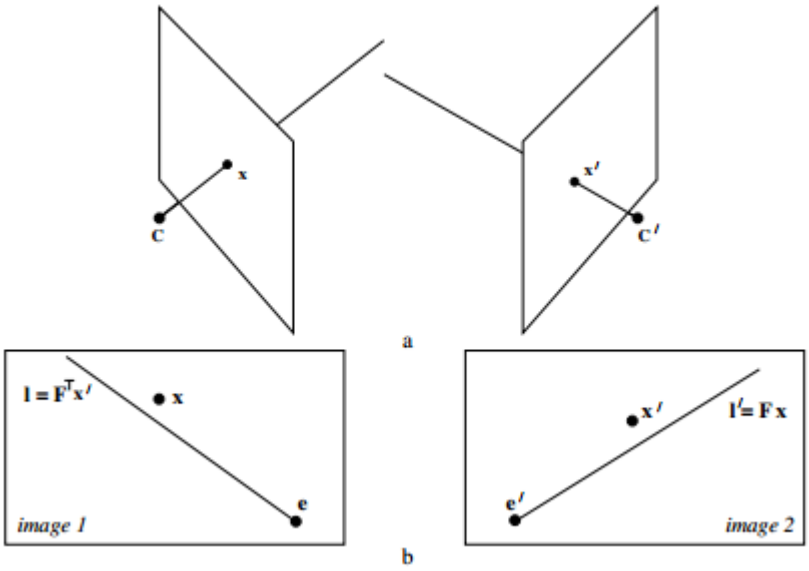

Figura 2.12 Os raios projetados a partir dos centros de câmera estão distorcidos devido aos ruídos.(Fonte: [4])

Portanto, para calcular o ponto X deve ser utilizado um método τ que considera esse erro e também deve ser invariante a transformações projetivas [4].

$$X_i = \tau(x_i, x_i', P, P') \tag{2.23}$$

$$\tau(x_i, x_i', P, P') = H^{-1}\tau(x_i, x_i', PH, P'H)$$

Para avaliar o método será utilizado o erro de reprojeção.

2.8.1 Erro de reprojeção

Para uma triangulação desse tipo, fica claro que é inapropriado minimizar o erro no espaço em três dimensões pois não existe um valor de referência para ser comparado. Porém os pontos correspondentes nas imagens são um valor que pode ser empregado como referência para calcular o erro na projeção.

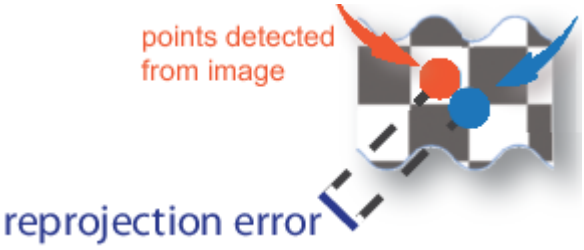

Figura 2.13 A imagem acima mostra graficamente o que é o erro de reprojeção. (Fonte: Camera Calibration Toolbox – MATLAB)

Esse método será utilizado fazendo uma estimativa de projeção de onde o ponto $\hat{X}_i$ estará no espaço. A partir daí as projeções desse ponto$(\hat{x}_i, \hat{x}'_i)$ na imagem serão computadas. O erro de reprojeção será a distância entre os pontos em pixels.

$$\sum_i d(x_i, \hat{x}_i)^2 + d(x'_i, \hat{x}_i)^2 \ para\ x'_i F x = 0 \qquad (2.24)$$

O objetivo na reconstrução é minimizar esse tipo de erro.

2.8.2 Cálculo da Reconstrução 3D

O cálculo da reconstrução 3D será realizado pelo método de triangulação linear aplicado em pontos não-homogêneos. Portanto, primeiro é preciso mudar os pontos homogêneos para não-homogêneos. Isso é feito dividindo o vetor de coordenadas dos pontos pelo último elemento do vetor.

$$x_{NH} = \left[\frac{x_1}{x_3} \ \frac{x_2}{x_3} \ \frac{x_3}{x_3}\right] = \left[\frac{x_1}{x_3} \ \frac{x_2}{x_3} \ 1\right]$$

$$X_{NH} = [X_1 \quad X_2 \quad X_3 \quad 1]$$

O método da triangulação linear é similar àquele utilizado na seção 2.3.2 para calcular a matriz fundamental. Cada ponto terá a relação $x_i = PX_i \leftrightarrow x'_i = P'X_i$ equivalente e as equações poderão ser combinadas na forma $AX = 0$, que é uma equação linear de X.

$$A = \begin{bmatrix} x_1 p_3^T - p_1^T \\ x_2 p_3^T - p_1^T \\ x_1' p_3^T - p_1^T \\ x_2' p_3^T - p_1^T \end{bmatrix} \tag{2.25}$$

$$AX = 0$$

Para p_i^T sendo as linhas da matriz de câmera. Esse conjunto de equações resultará em um conjunto de quatro equações lineares e três variáveis desconhecidas. Para resolver a equação é usado o método dos mínimos quadrados onde $AX = b$, onde b não será uma solução única uma vez que se tem mais equações que variáveis. Deve-se encontrar a solução em que $\|Ax - b\|$ é minimizada. Solução essa que é deduzida a partir de uma decomposição em valores singulares [4].

$$A = UDV^T \tag{2.26}$$

Substituindo $y = V^T X \; e \; b' = U^T b$ na equação a ser minimizada.$\|Dy - b'\|$ será a equação a ser resolvida. Segundo [4] a solução será $X = Vy$ para $y_i = b_i / d_i$.

2.9 UNIFICAÇÃO DOS PONTOS

É possível observar um problema ao calcular mais de duas reconstruções 3D para duas vistas. Os valores de R e t estão com referenciais diferentes para cada valor descrito. Conforme Tab. 2.1:

Tabela 2.1 Matrizes de câmera para os diferentes pontos de vista

	Ponto 1	Ponto 2
Imagem 1	$P_1 = K[R_1 \mid t_1]$	
Imagem 2	$P_2 = K[R_2 \mid t_2]$	$P_2' = K[R_2' \mid t_2']$
Imagem 3		$P_3 = K[R_3 \mid t_3]$

Para realizar a junção das matrizes de câmera para um único referencial deverá ser feito a unificação dos pontos, que consiste em juntar as nuvens de pontos com referenciais diferentes. Em seguida, encontrar os pontos em comum para as duas nuvens e uni-los

minimizando o erro de reprojeção através do algoritmo de otimização não linear de Levenberg-Marquardt. A matriz de rotação e deverá ser transformada em *axis-angle* e devido a presença de senos e cossenos será utilizada a otimização não linear. A matriz de rotação é descrita por:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -sen(\theta) \\ 0 & \text{sen}(\theta) & \cos(\theta) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \text{sen}(\theta) \\ 0 & 1 & 0 \\ -sen(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

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

Porém os valores obtidos de R estão em termos de x, y, z. Para converter para axis-angle em termos de θ utiliza-se a fórmula [15]

$$\begin{cases} \theta_x = atan2(r_{32}, r_{33}) \\ \theta_y = atan2\left(-r_{31}, \sqrt[2]{r_{31}^2 + r_{33}^2}\right) \\ \theta_z = atan2(r_{21}, r_{11}) \end{cases} \quad (2.27)$$

Sendo r_{ij} componentes da matriz de rotação(R).

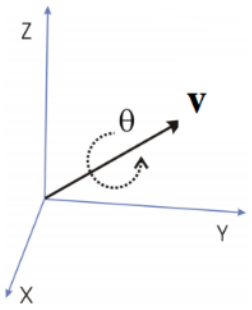

Figura 2.14 Representação na forma Axis-Angle para coordenadas x,y,z

A rotação será feita seguindo formula de rotação de Rodrigues [14]:

$$k_{rot} = kcos\theta + (v \times k)sin\theta + v(v \cdot k)(1 - cos\theta) \quad (2.28)$$

Sendo v_{1x3} e k_{1x3} e θ representações das matrizes de rotação na forma *axis-angle*. Após as rotações calcula-se o novo erro de reprojeção otimizando para o menor erro possível. Com isso todos os R e t serão unidos no mesmo referencial. Porém é preciso voltar a matriz de rotação para as coordenadas x, y, z. Isso será feito através da Eq. 2.9.4[15].

$$R_{3x3} = \begin{bmatrix} \cos\theta + v_x^2(1-cos\theta) & cos\theta + v_x v_y(1-cos\theta) & v_y \sin\theta + v_x v_z(1-cos\theta) \\ v_z sin\theta + v_x v_y(1-cos\theta) & cos\theta + v_y^2(1-cos\theta) & -v_x \sin\theta + v_y v_z(1-cos\theta) \\ -v_y sin\theta + v_x v_z(1-cos\theta) & v_x \sin\theta + v_y v_z(1-cos\theta) & cos\theta + v_z^2(1-cos\theta) \end{bmatrix} \quad (2.29)$$

Por fim, todos os pontos foram reconstruídos e estão no mesmo referencial.

3 IMPLEMENTAÇÃO DO SISTEMA

Nesse capítulo será descrito o sistema a ser utilizado no projeto. O detalhamento matemático foi explicado no capítulo 2. Portanto, agora preocupa-se apenas com aspectos da implementação.

3.1 INTRODUÇÃO

A reconstrução 3D esparsa de um objeto de teste será feita nesse projeto a partir da aquisição de onze imagens. A reconstrução do objeto é feita a partir da reprojeção dos pontos em duas dimensões para três dimensões através da matriz de câmera. Para encontrar as matrizes de câmera para cada ponto de vista é necessário obter os parâmetros intrínsecos e os parâmetros extrínsecos das câmeras.

Os parâmetros intrínsecos serão os mesmos para cada ponto de vista pois se referem a valores específicos da câmera. Estes serão obtidos pela calibração.

Os parâmetros extrínsecos são obtidos a partir da matriz fundamental, que por sua vez é adquirida pela correspondência de pontos de interesse em imagens adjacentes. Os 6 passos que levam até a reconstrução do objeto estão listados na Fig. 3.1.

Figura 3.1 Fluxograma que representa os passos do sistema proposto para reconstrução 3D a partir de múltiplas imagens.

Todos os passos serão descritos com maior detalhe no capítulo que se segue.

3.2 EQUIPAMENTOS UTILIZADOS

No sistema implementado, foram utilizados os seguintes equipamentos:

- Câmera Fujifilm X-E1 Mirrorless com lente Fujifilm - 35mm f/1.4 XF;
- Tripé Joby GorillaPod Hybrid;
- Flash Yongnuo Yn-560ii com transmissor sem fio;

- Software MATLAB 2015b;
- Software MeshLab.

3.3 CALIBRAÇÃO DA CÂMERA

Para realizar a reconstrução em três dimensões de um objeto é necessário verificar quais são os parâmetros da câmera. Para que os parâmetros sejam determinados é possível realizar a calibração da câmera a partir de um padrão “tabuleiro de damas” em diferentes ângulos.

A calibração será realizada a partir da “Camera Calibration Toolbox” presente no MATLAB 2015b. O *software* fornece o padrão “tabuleiro de damas” presente na Fig. 3.3 para impressão e a instruções para calibração são simples.

Figura 3.2 - Os passos para calibrar uma imagem utilizando a toolbox do MATLAB. (Fonte: Camera Calibration Toolbox - MATLAB)

O programa recomenda fotografar de 10 a 20 imagens do padrão em variados ângulos para adquirir os parâmetros com maior precisão. Em seguida, é preciso adicionar as imagens a *toolbox* e calibrar. A *toolbox* calcula o erro de reprojeção e caso esteja muito alto recomenda adquirir novas imagens. Os principais valores a serem utilizados são àqueles necessários para encontrar a matriz K descrita no capítulo anterior para câmeras CCD.

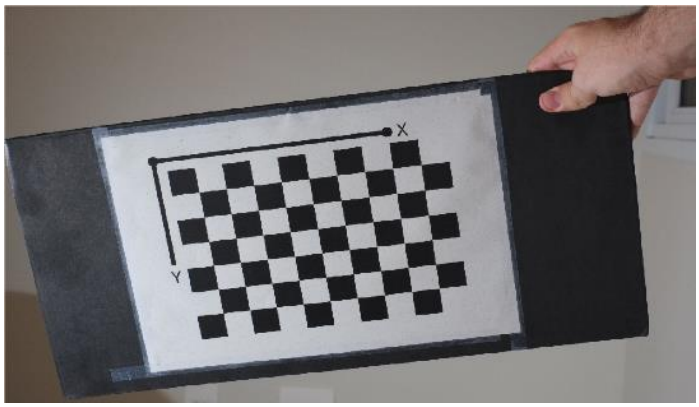

Figura 3.3 – Tabuleiro de damas fornecido pelo programa.

3.4 AQUISIÇÃO

A aquisição das imagens é feita com a câmera Fujifilm X-E1 Mirrorless com lente Fujifilm - 35mm f/1.4 XF em um tripé Joby GorillaPod Hybrid em conjunto com o Flash Yongnuo Yn-560ii. Os objetos a serem fotografados são uma caneca, um cubo mágico e um icosaedro de origami.

A câmera foi configurada no modo manual com abertura em f/8.0, velocidade do obturador de 1/250 s (velocidade de sincronismo do flash). O foco para aquisição das imagens foi o mesmo utilizado para calibração não alterando a distância focal.

São adquiridas 11 imagens com aproximadamente 10º de diferença entre cada foto e a uma distância de aproximadamente 54cm até o objeto. A posições da câmera são delimitadas a partir de marcações no chão com as distâncias medidas com régua. Uma das pernas do tripé é posicionado na demarcação para cada posição da câmera. Após a aquisição, as fotos são exportadas para o computador para serem processadas pelo MATLAB.

Figura 3.4 - Esquema para as fotografias serem realizadas.

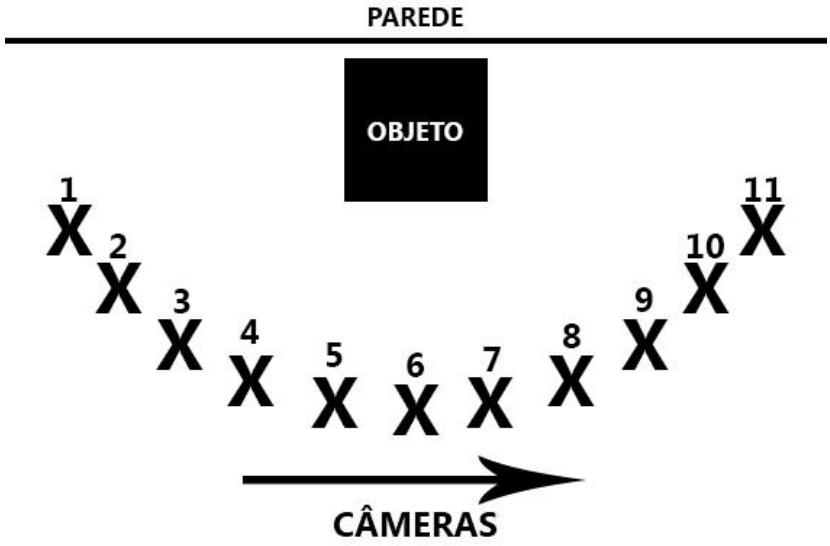

Figura 3.5 – Diagrama da ordem e posição para a aquisição das fotografias.

Para os testes são fotografados três objetos: uma caneca, uma icosaedro de origami e um cubo mágico. Todos os objetos foram capturados com o mesmo processo.

3.5 OBTENÇÃO DOS PONTOS DE INTERESSE

As imagens são carregadas com tamanho 2496x1664 pixels. Porém, para realizar a reconstrução 3D não é necessário utilizar todos os pontos. É preciso encontrar os pontos mais importantes para a formação do objeto que são chamados de pontos de interesse. Eles geralmente são localizados em regiões de bordas da figura. Nesse projeto será utilizado o algoritmo SIFT (seção 2.3).

Os pontos de interesse serão obtidos a partir da função vl_sift.m presente na biblioteca VLFEAT que implementa diversos algoritmos de visão computacional em C e possui interface com MATLAB.

A rotina que implementa a função recebe duas imagens e as reduz para 1024x683 pixels com objetivo de diminuir o tempo de processamento. A redução das imagens é feita pela função *imresize.m* que por padrão utiliza o algoritmo de interpolação bicúbica e *antialiasing* para mudar o tamanho das imagens. A função vl_sift.m atua em seguida para adquirir a localização dos pontos de interesse e os *keypoint descriptors* com informação da vizinhança.

Com essas informações já é possível realizar a correspondência entre as duas imagens.

3.6 CORRESPONDÊNCIA

A correspondência é realizada para encontrar pontos entre as duas imagens que representam o mesmo ponto no mundo. É importante porquanto permite comparar os pontos adquiridos no passo anterior.

Nesse trabalho a correspondência é feita comparando *keypoint descriptors* das duas imagens pela sua distância euclidiana utilizando indexação kd-tree(seção 2.4) para uma busca mais rápida. O algoritmo foi implementado pela biblioteca VLFEAT pela função vl_kdtreequery e vl_kdtreebuild. A implementação verifica os pontos falsos positivos com um limite de distância entre os dois melhores resultados da busca. Os pontos correspondentes serão utilizados no próximo passo.

3.7 MATRIZ FUNDAMENTAL(F)

A matriz fundamental irá descrever a relação matemática entre duas imagens, mapeando um ponto na primeira imagem para uma linha na segunda. Assim sendo, para encontrar a posição do ponto no espaço é preciso encontrar a matriz fundamental.

A forma utilizada para encontrar a matriz fundamental será o algoritmo de 8 pontos normalizados, descrito no capítulo anterior. Esse algoritmo é auxiliado pela função RANSAC (seção 2.6) que filtra os pontos iterativamente retirando aqueles que são divergentes do modelo.

A função recebe os pontos os filtra por RANSAC os valores que normalizados estão a uma distância maior que 0.002 do modelo. A partir daí os pontos que restaram são enviados para o algoritmo de oito pontos normalizados. Com esse passo, a matriz fundamental é obtida, mas a extração da matriz essencial(E) para que seja realizada a triangulação.

A matriz fundamental é verificada através do determinante. O determinante ser igual a zero é um indicador de que a matriz está correta.

A matriz essencial é extraída conforme a Eq. 2.14.

3.8 MATRIZ DE CÂMERA

A matriz de câmera permite a tradução dos pontos em uma imagem para a reconstrução em três dimensões, sendo necessário fazer a triangulação para encontrar os últimos parâmetros.

A triangulação localiza a câmera no espaço. R e t são ,respectivamente, a matriz de rotação e o vetor de translação referentes à câmera para cada imagem. Para encontrar R e t é utilizada a decomposição da matriz essencial(E) em valores singulares. Todavia, conforme visto na seção 2.3.3, cada matriz essencial decomposta resulta em quatro soluções de R e t para matriz de câmera.

O próximo passo é testar os Rs e ts e verificar qual terá melhor performance para a matriz de câmera completa. O teste é realizado inserindo cada combinação de R e t na matriz fundamental e recuperando os valores dos pontos no espaço. Partindo dos pontos recuperados, verifica-se qual combinação terá mais pontos na nuvem de pontos recuperada.

Com a seleção de R e t já é possível extrair a matriz de câmera para aquele ponto de vista.

3.9 RECONSTRUÇÃO 3-D

Após adquirir a matriz de câmera, é possível obter o ponto de interesse no espaço 3D. Para obter tais coordenadas será utilizada a função vgg_X_from_xP_nonlin.m. Essa função estima a posição de um ponto no espaço com o teorema de reconstrução projetiva descrito na seção (2.8) em conjunto com o método iterativo de Newton para reduzir o erro de reprojeção. A nuvem de pontos para um par de imagens é obtida ao repetir esse processo para todos os pontos de interesse encontrados.

3.10 UNIFICAÇÃO DE PONTOS

A unificação dos pontos é a parte final do projeto e reúne todas as nuvens de pontos em uma só. Consiste em reduzir ao máximo a distância entre os pontos reprojetados e os pontos na imagem para todos os pontos de vista. A unificação dos pontos é realizada em três partes, conforme o algoritmo descrito na seção (2.9).

Primeiramente é preciso unificar as matrizes de rotação e o vetor de translação para cada par de imagens para somente um sistema de referências. Em seguida, sobrepor as nuvens de pontos para conseguir uma com todos os pontos de interesse adquiridos em cada par de imagens. O último passo é reduzir o erro de reprojeção utilizando o algoritmo de otimização

de Levenberg-Marquardt. Nesse trabalho a função *bundleadjustment.m* faz o papel de reduzir o erro de reprojeção e a função *merge2graphs.m* realiza a união dos pontos de vista. Para avaliar a unificação de pontos será utilizada o erro de reprojeção descrito na seção 2.8.1. Caso o erro de reprojeção seja maior que 2 pixels e 5 graus os pontos são removidos da nuvem de pontos. Se tais pontos não forem removidos o erro de reprojeção irá aumentar a cada unificação de nuvens.

Por fim, o programa salva a função no formato ply para que seja aberta pelo *software* $MeshLab$ para visualização.

4 RESULTADOS

Nesse capítulo serão abordados os resultados obtidos com a implementação do sistema descrito no capítulo 3 e serão apresentadas imagens e tabelas como evidencias dos estudos realizados.

4.1 CALIBRAÇÃO

Foram fotografadas 18 poses do padrão de tabuleiro de damas. Seguem imagens:

Figura 4.1 Imagens Carregadas na Camera Calibration Toolbox no MATLAB

As imagens foram carregadas no MATLAB e incluídas na Camera Calibration Toolbox e em seguida processadas após o botão *calibrate* ser pressionado e o preenchimento do tamanho do quadrado impresso. As imagens foram avaliadas e o programa recusou uma das imagens devido a problemas de iluminação. A quinta imagem da primeira fileira apresentou problemas na iluminação para o programa. Tal problema impedia a avaliação do quadriculado corretamente. 17 foram avaliadas e seguem abaixo:

Figura 4.2 – Imagens aceitas e processadas pelo pela câmera Calibration Toolbox no MATLAB

Foram selecionadas duas imagens com o intuito de verificar os detalhes com as anotações feitas

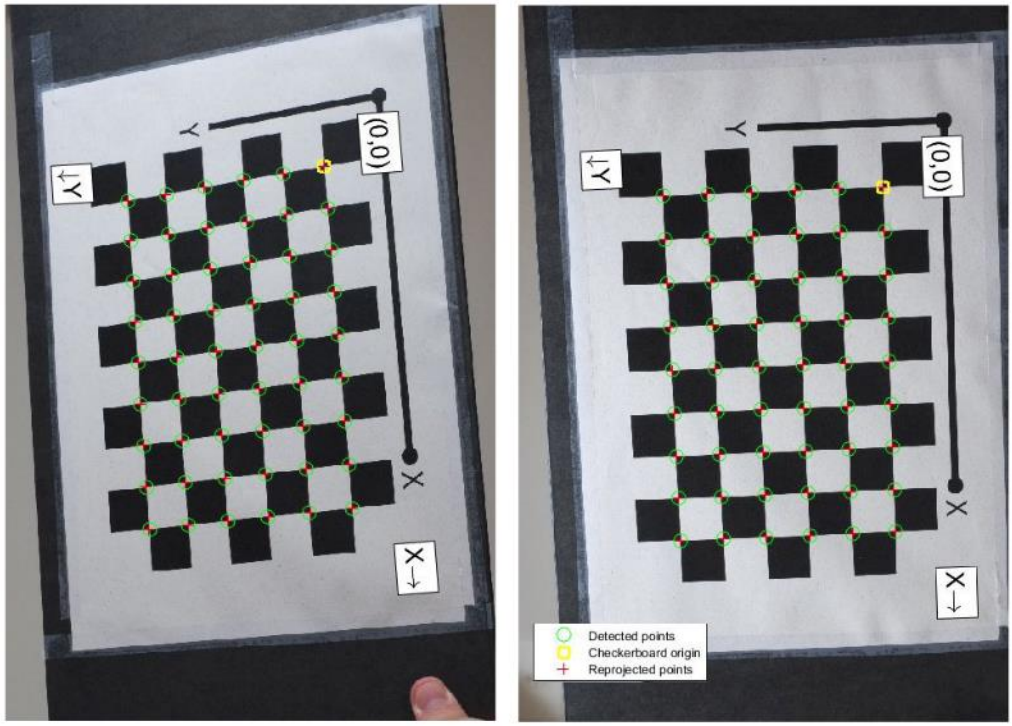

Figura 4.3 – Processamento da Camera Calibration Toolbox do MATLAB. Observa-se que as intersecções entre os quadrados são detectadas e são computados os pontos detectados e os pontos reprojetados para ajustar a matriz da câmera.

O MATLAB calcula os erros de reprojeção para os pontos correspondentes entre os quadriculados. O erro de reprojeção para calibração teve uma média de 0.78 pixel com pico de aproximadamente 1.2 pixel, com visto na fig. 4.4. Considerando que a imagem possui medidas 2496x1664 pixels a medida foi precisa.

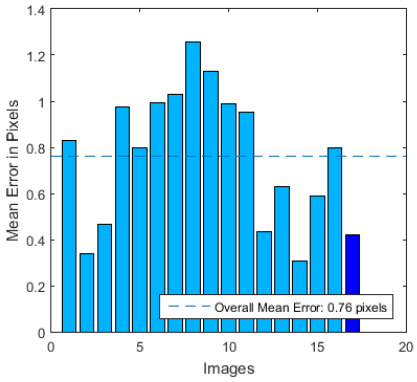

Figura 4.4 - O gráfico mostra a performance com relação à reprojeção de erro de cada imagem. A média é de 0.76 pixels

Por fim, os valores importantes para o avanço no trabalho são a distância focal adquirida e a posição dos pontos principais em pixels foram medidos. A matriz de valores intrínsecos (K) com os valores medidos está expressa na Eq. 4.1.

$$K = \begin{bmatrix} 3910 & 0 & 1284{,}5 \\ 0 & 3899 & 877{,}3 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Como esperado, os valores da distância focal para o eixo x e para o eixo y são diferentes. Como citado no capítulo 2, provavelmente as divergências são por causas de ruído na compressão ou no processamento da imagem pela câmera. A distância equivalente a um pixel no mundo real é calculada utilizando a distância focal média para os as duas direções.

$$\alpha_{médio} = \frac{m_{médio}}{f}, \tag{4.2}$$

Onde f é a distância focal que o fabricante coloca para o conjunto lente e sensor. Nesse caso é de $f = 53mm$. O $\alpha_{médio}$ será a média entre os valores de distância focal para os eixos x e y. Nesse caso $\alpha_{médio} = 3904{,}5$. Portanto, o valor de será:

$$m_{médio} = 73{,}7\,\frac{px}{mm}.$$

4.2 AQUISIÇÃO

O processo de aquisição foi feito a partir do formato descrito no capítulo 3, ou seja, onze imagens. Foram escolhidos 3 objetos para servirem de exemplo no programa. Os objetos são uma caneca, um cubo mágico e um icosaedro de origami. As figuras 4.5,4.6 e 4.7 mostrarão as fotografias.

Figura 4.5 – A imagem acima possui as fotografias das canecas.

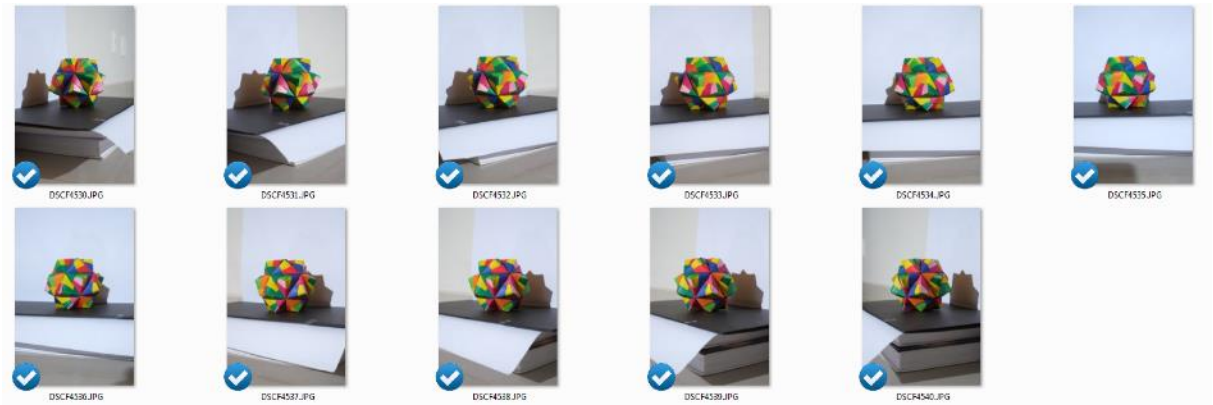

Figura 4.6 – A imagem acima contém as fotografias do origami

Figura 4.7 - A imagem acima contém as fotografias do cubo mágico

Inicialmente foram encontradas dificuldades para fotografar o objeto pois as correspondências não eram feitas de forma correta se a câmera não estivesse com a mesma distância focal utilizada na calibração. Após a montagem do sistema com as demarcações de

distância corretas e de acordo com a distância focal calibrada foi possível obter as imagens com rapidez. As imagens são de 2496x1664 pixels.

4.3 PONTOS DE INTERESSE E CORRESPONDÊNCIA DE CARACTERÍSTICAS

Os pontos de interesse foram computados pelo algoritmo SIFT em conjunto com a correspondência pelo algoritmo árvore-kd. A Fig. 4.8, 4.9 e 4.10 mostram todas as correspondências entre imagens adjacentes adquiridas.

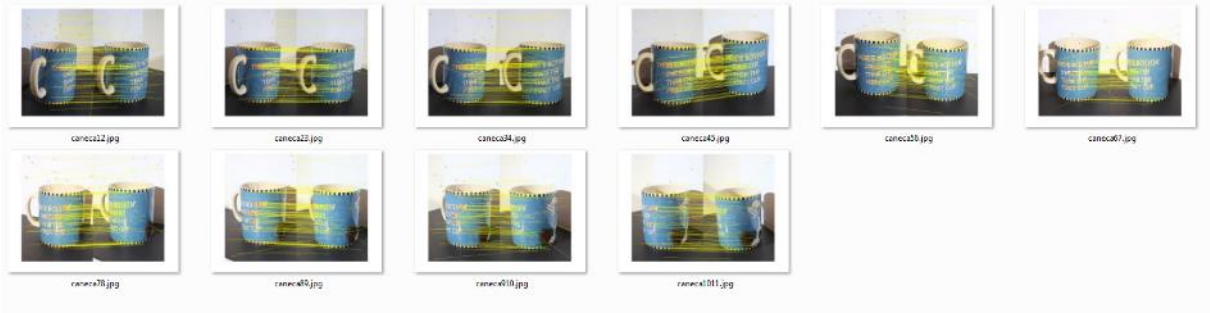

Figura 4.8 Imagens adjacentes com correspondências para as imagens da caneca

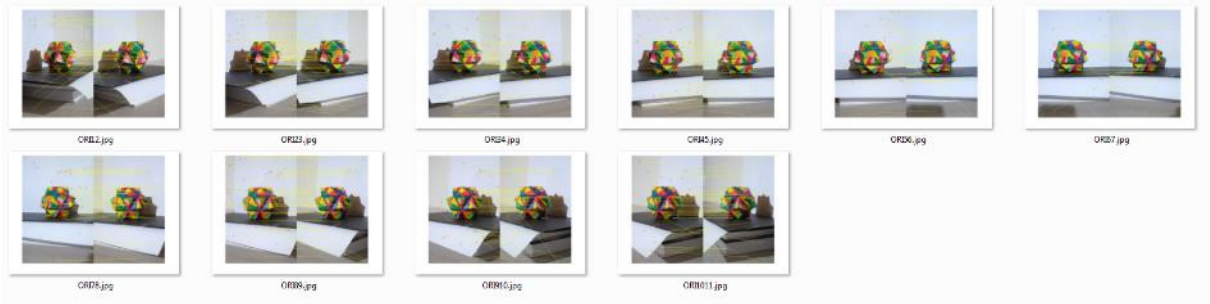

Figura 4.9 - Imagens adjacentes com correspondências para as imagens do origami

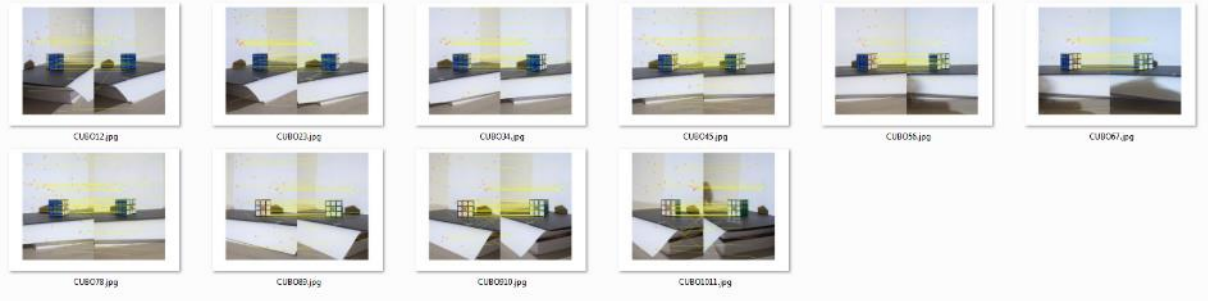

Figura 4.10- Imagens adjacentes com correspondências para as imagens do cubo mágico.

Devido a quantidade de imagens nem todas foram mostradas em detalhe no texto. Apenas duas imagens para cada objeto seguem nas Fig. 4.11, 4,12 e 4.13. As imagens da caneca a serem analisadas são a da Fig. 4.11.

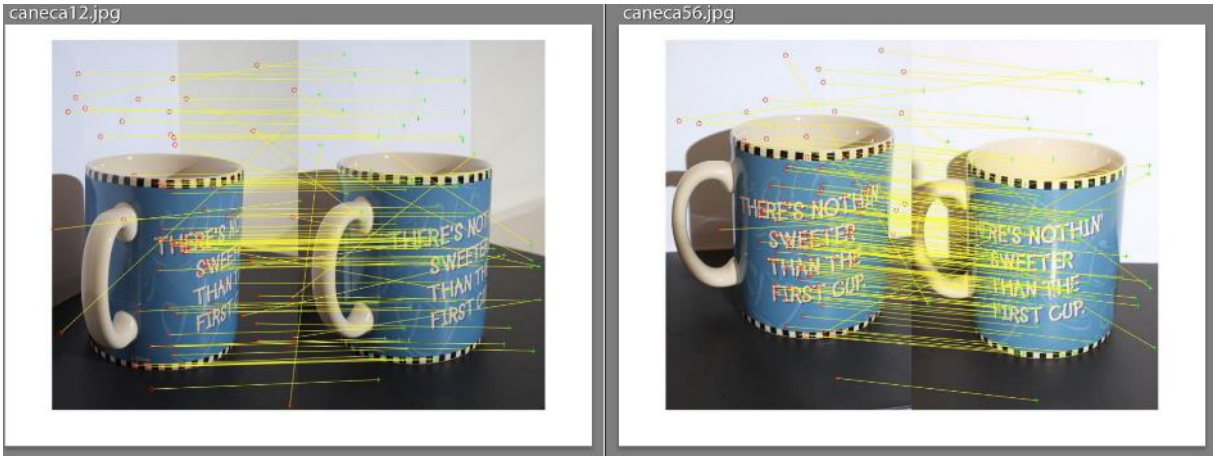

Figura 4.11 Imagens de correspondências do primeiro e quinto par de fotos.

Observando no detalhe as fotos das correspondências da caneca, as regiões contínuas de cores não são consideradas na imagem como pontos de interesse. Exemplo disso é a falta de pontos de interesse na região azul da caneca.

A grande quantidade de pontos de interesse correspondentes no texto da caneca é relevante pois mostra que essa região têm uma grande chance de ser representada na reconstrução 3D. Mais um fato relevante são as correspondências, que mesmo não indicando o mesmo ponto no espaço são marcadas erroneamente. Em passos seguintes tais correspondências são eliminadas.

As figuras 4.12 e 4.13 são referentes as imagens do cubo e do origami respectivamente. Elas serão analisadas em conjunto.

Figura 4.12 – Imagens de correspondências do sexto e segundo par de fotos respectivamente.

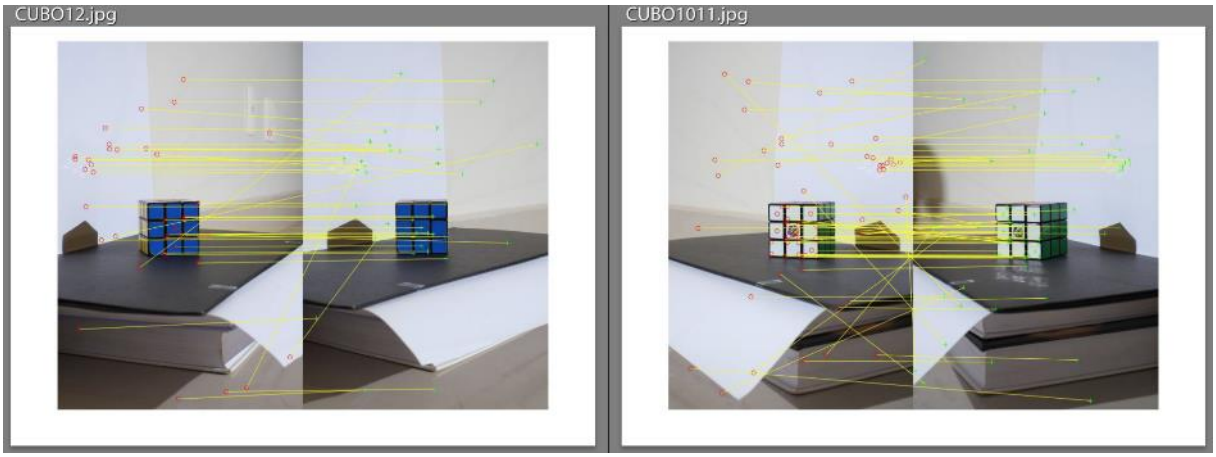

Figura 4.13 - Imagens de correspondências do primeiro e décimo par de fotos respectivamente.

Os dois objetos apresentam comportamento parecido frente ao algoritmo. O origami detém de variações de cores fortes, no entanto não apresenta contraste grande o suficiente nas bordas. O algoritmo SIFT analisa a diferença de gaussianas da imagem em escala de cinza, e por esse motivo o programa não conseguiu fazer tantas correspondências quanto necessário.

O mesmo ocorre no cubo, contudo percebe-se uma clara diferença quando o contraste é maior. O lado branco do cubo possui muito mais pontos de interesse do que o lado azul. Há também um conjunto de correspondências no fundo acima do cubo que devem ser causadas pelo papel amassado.

São reiterados os aspectos discutidos sobre as fotografias quando observada a quantidade de pontos para cada par de imagens são mostradas nas figuras 4.14 e 4.15.

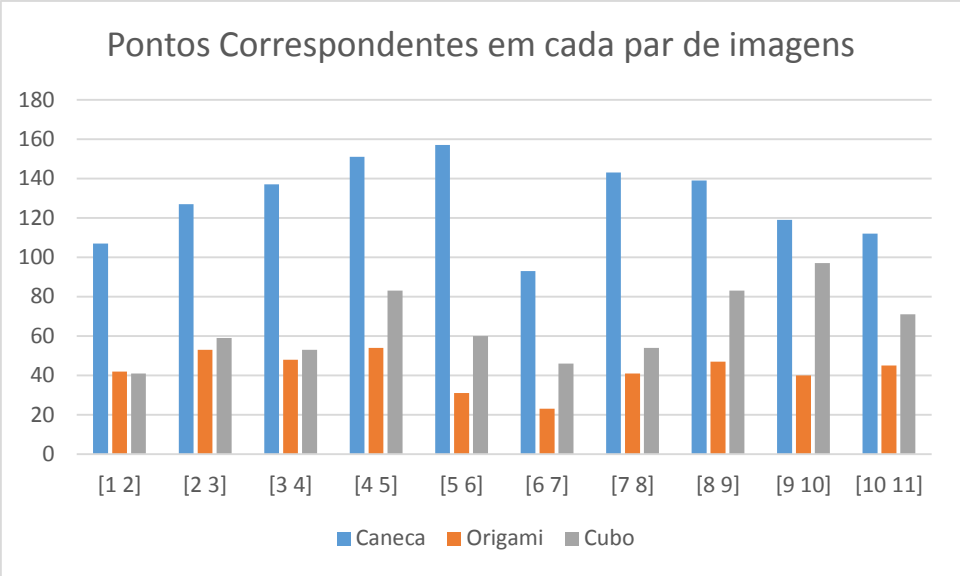

Figura 4.14 – Pontos correspondentes entre um par de imagens

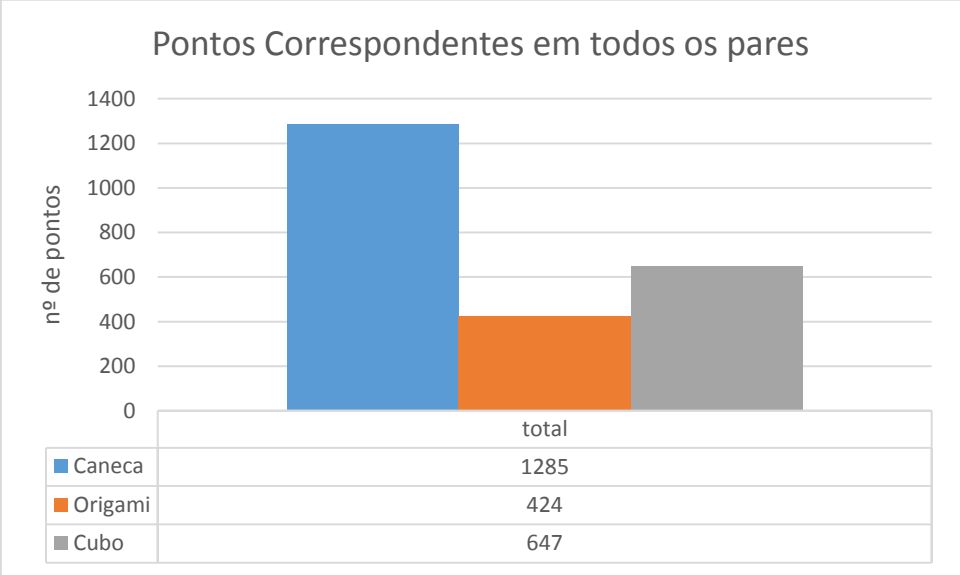

Figura 4.15 – Correspondência total para todas imagens.

4.4 MATRIZ FUNDAMENTAL

As matrizes fundamentais foram calculadas com sucesso já que todos os pares de imagens para cada objeto tiveram suas matrizes fundamentais computadas. Pode-se afirmar que as matrizes foram calculadas corretamente devido aos valores do determinante. Se a matriz for igual a zero ou muito próxima de 0 a matriz está correta. Os valores das determinantes estão na tabela 4.1.

Tabela 4.1 – A tabela mostra os valores dos determinantes das matrizes fundamentais para cada par de imagens

	det(F)		
Par	ORIGAMI	CUBO	CANECA
[1 2]	3,89E-24	-1E-23	-8,4E-24
[2 3]	0	-7,6E-24	3,37E-24
[3 4]	-2,8E-24	1,62E-24	-3,9E-24
[4 5]	-1E-24	-2,4E-25	0
[5 6]	-2,2E-22	-3,8E-22	0
[6 7]	6,69E-23	-1,4E-23	1,15E-23
[7 8]	-1,7E-24	-1,6E-23	1,78E-24
[8 9]	3,98E-25	1,19E-23	4,01E-24
[9 10]	6,68E-23	5,82E-24	-2,6E-25
[10 11]	0	3,92E-22	3,34E-25

Todos os valores são iguais a 0 e isso mostra uma forte evidência das matrizes estarem corretas. Nesse passo do código também são calculados pontos outliers que não estão de acordo com o modelo estimado do algoritmo RANSAC. As correspondentes que não passaram nesse teste foram filtradas. As figuras 4.16 e 4.17 mostram a nova quantidade de pontos.

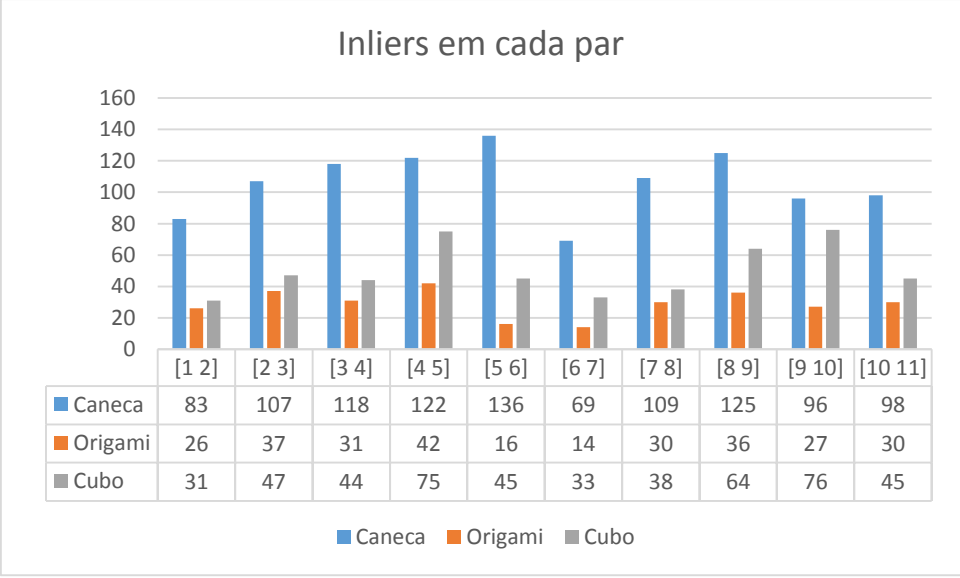

Figura 4.16 O gráfico acima mostra a quantidade de pontos que sobraram depois dessa etapa para cada par de imagem.

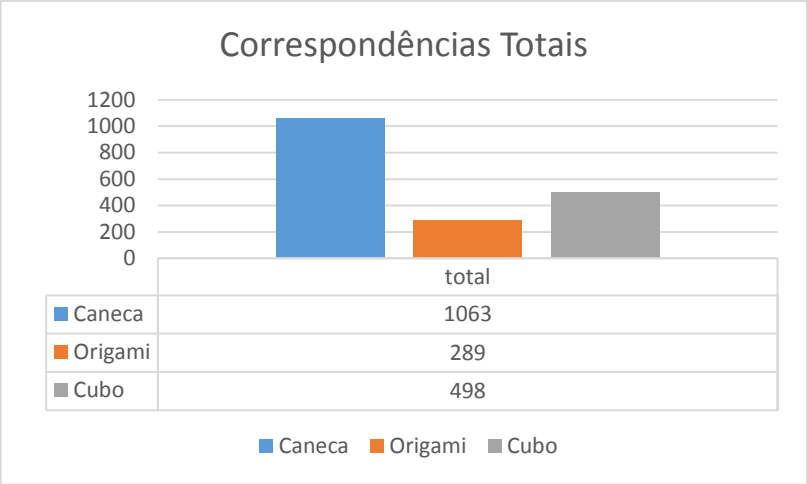

Figura 4.17 O gráfico mostra a quantidade total de pontos que sobraram para cada objeto no total.

Conforme a Fig. 4.17, houve uma redução de 17% na quantidade de pontos para a caneca. No cubo, uma redução de 23% nas correspondências e por fim uma diminuição de 32% para os pontos aferidos no origami. Esses números confirmam o que foi mencionado na seção 4.3 sobre as correspondências que não representavam o mesmo ponto no espaço.

A partir da matriz fundamental foi calculada a matriz essencial. Devido a grande quantidade de valores será inserido um exemplo do primeiro par de cada objeto.

$$E_{caneca_{21}} = \begin{bmatrix} -0.1973 & -1.461 & -0.597 \\ 3.472 & 0.0653 & -9.854 \\ 0.1140 & -9.264 & -0.0607 \end{bmatrix}$$

$$E_{cubo_{21}} = \begin{bmatrix} -0.0504 & -6.3279 & -0.1509 \\ 6.327 & -0.309 & -10.498 \\ 0.216 & 9.771 & 0.003 \end{bmatrix}$$

$$E_{origami_{21}} = \begin{bmatrix} -0.193 & -2.931 & -0.403 \\ -0.223 & 0.222 & -8.467 \\ 0.549 & 8.848 & -0.041 \end{bmatrix}$$

4.5 MATRIZ DE CÂMERA

Nessa etapa são estimados os valores da matriz de rotação e do vetor de translação que em seguida formam a matriz de câmera. Nas equações abaixo estarão os exemplos com o primeiro par de cada objeto.

$$P_{caneca_{21}} = \begin{bmatrix} 1568.3 & -78.2 & -309 & 1577 \\ 77.2 & 1597.7 & -12.4 & 98.8 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P_{cubo_{21}} = \begin{bmatrix} 700{,}7 & -29.2 & 1438.5 & -1343.5 \\ -54.4 & -1598.7 & -5.9 & 19 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$P_{origami_{21}} = \begin{bmatrix} 1504.3 & -14 & -546 & 1517 \\ 25.3 & 1599.2 & 28.6 & -74.7 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

4.6 RECONSTRUÇÃO 3D e UNIFICAÇÃO DE PONTOS

O processo de reconstrução em três dimensões foi realizado com sucesso pelo programa. Os resultados para o erro de reprojeção para o passo de reconstrução 3D de cada objeto estão nas tabelas 4.2, 4.3 e 4.4. Todos os valores estão em pixel.

Tabela 4.2 – Erros de reprojeção para cada par de imagem do cubo em pixels.

	CUBO	
Par	**Erro de Reprojeção Inicial (pixel)**	**Erro de Reprojeção Final(pixel)**
[1 2]	58,363	2,719
[2 3]	13,060	1,582
[3 4]	41,346	1,407
[4 5]	51,435	3,710
[5 6]	30,007	0,771
[6 7]	32,355	0,511
[7 8]	11,903	1,240
[8 9]	16,275	0,928
[9 10]	35,111	0,722
[10 11]	1,784	0,753

Tabela 4.3 – Erros de reprojeção para cada par de imagem do origami em pixels

	ORIGAMI	
Par	**Erro de Reprojeção Inicial(pixel)**	**Erro de Reprojeção Final(pixel**
[1 2]	11,988	2,210
[2 3]	18,197	1,031
[3 4]	4,496	1,990
[4 5]	12,733	1,949
[5 6]	25,047	5,740
[6 7]	66,329	4,628
[7 8]	4,514	0,660
[8 9]	46,361	2,466
[9 10]	64,046	1,479
[10 11]	144,637	4,753

Tabela 4.4 – Erros de reprojeção para cada par de imagem da caneca em pixels

	CANECA	
Par	Erro de Reprojeção Inicial(pixel)	Erro de Reprojeção Final(pixel)
[1 2]	84,880	2,415
[2 3]	37,036	1,999
[3 4]	69,234	2,128
[4 5]	93,392	1,589
[5 6]	34,528	0,985
[6 7]	29,868	1,094
[7 8]	43,395	2,375
[8 9]	11,068	1,463
[9 10]	8,061	1,449
[10 11]	39,642	2,099

Os erros de reprojeção para etapa de reconstrução 3D apresentam um valor impraticável caso fossem levados diretamente a unificação dos pontos. Isso se deve à propagação do erro desde a correspondência dos pontos até esse ponto. Todos os parâmetros exceto os valores intrínsecos da câmera são estimativas. Porém a otimização nesse passo foi suficiente para reduzir o erro consideravelmente para um valor aceitável próximo a 2 pixels. Valor esse que não apresenta problemas na próxima etapa.

Por fim o passo final é a união das nuvens de pontos. A união entre as nuvens aumenta o erro de reprojeção devido a mudança de referenciais e portanto, o erro inicial é muito maior para todos os casos. Na tabela 4.5 estão os valores de erro inicial e final para cada objeto.

Tabela 4.5 – Erros de reprojeção para etapa de unificação de pontos.

OBJETO	Erro de Reprojeção Inicial(pixel)	Erro de Reprojeção Final(pixel)
CANECA	509,371	2,335
CUBO	118,457	0,753
ORIGAMI	341,100	1,382

Nessa etapa ocorre uma filtragem massiva dos pontos que não conseguem se adequar à unificação de pontos. São os pontos que sozinhos possuem erro de reprojeção maior do que 2 pixels. Pontos com erros maiores propagavam o erro para a próxima iteração o que gerava dificuldades e gasto computacional desnecessário. Na Fig. 4.18 estará a quantidade de pontos filtrada nessa última etapa.

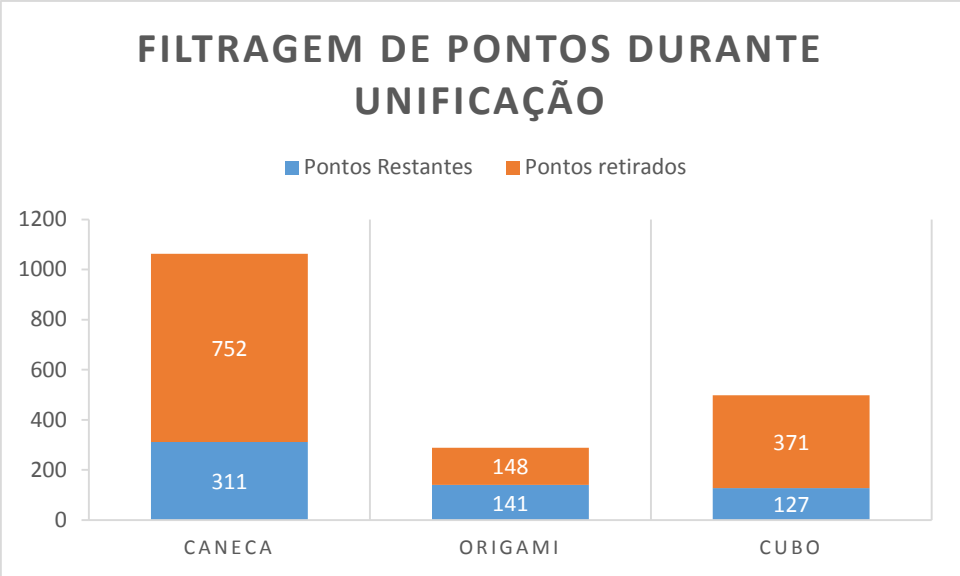

Figura 4.18 –Quantidade de pontos que foi filtrada durante a unificação de pontos.

A Figura 4.18 mostra que foi realizada uma retirada massiva dos pontos no espaço. Da caneca foram excluídos 71% dos pontos, do origami 51% e do cubo foram retirados 74%.

Após finalizar esse processo foi adquirida a nuvem de pontos para cada objeto. As Fig. 4.19, 4.20 e 4.21 contém a nuvem esparsa de pontos dos objetos.

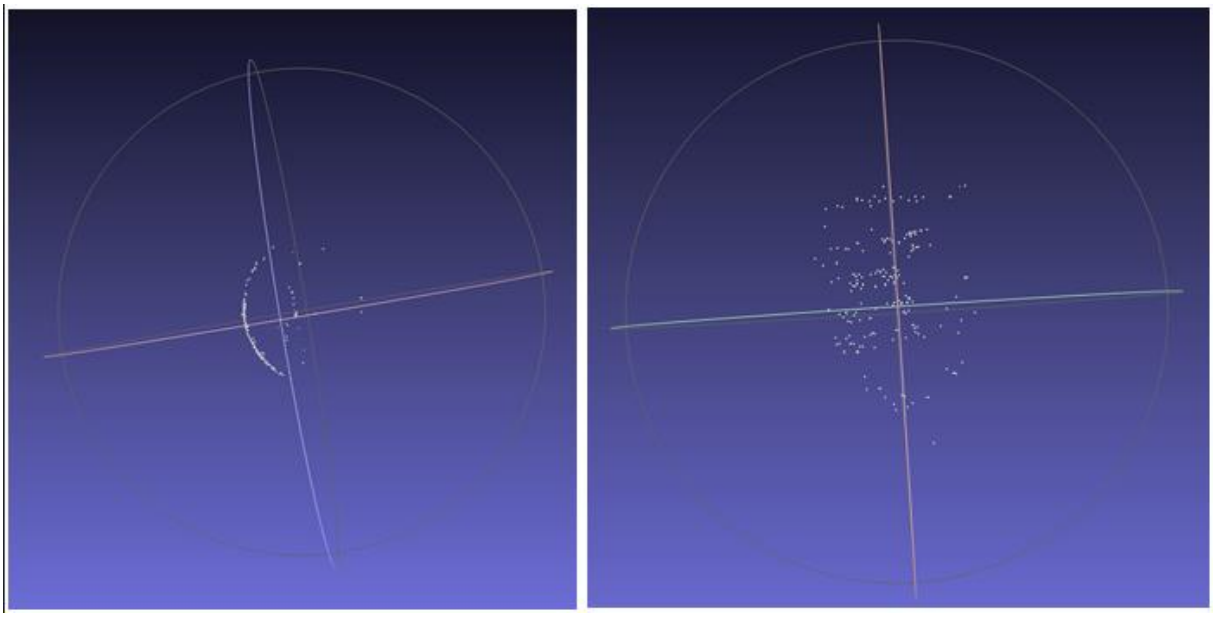

Figura 4.19 – Nuvem de pontos da Caneca exibidos no Software Meshlab. Visão superior e visão frontal, respectivamente

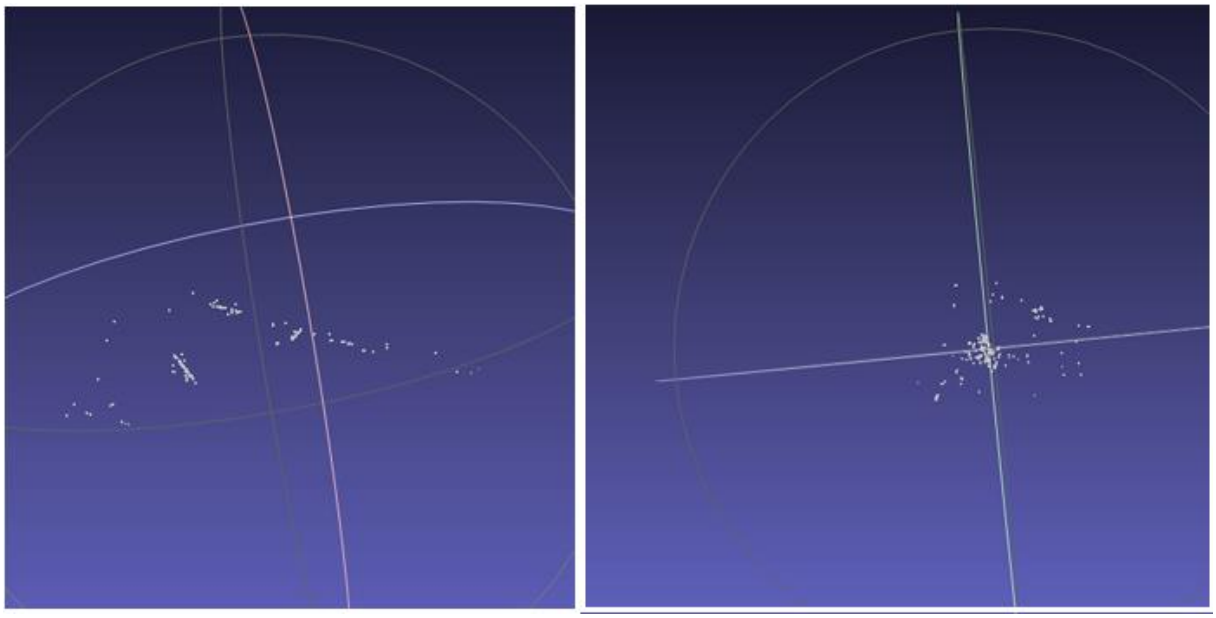

Figura 4.20 Nuvem de pontos da Caneca exibidos no Software Meshlab. Visão superior e visão frontal, respectivamente.

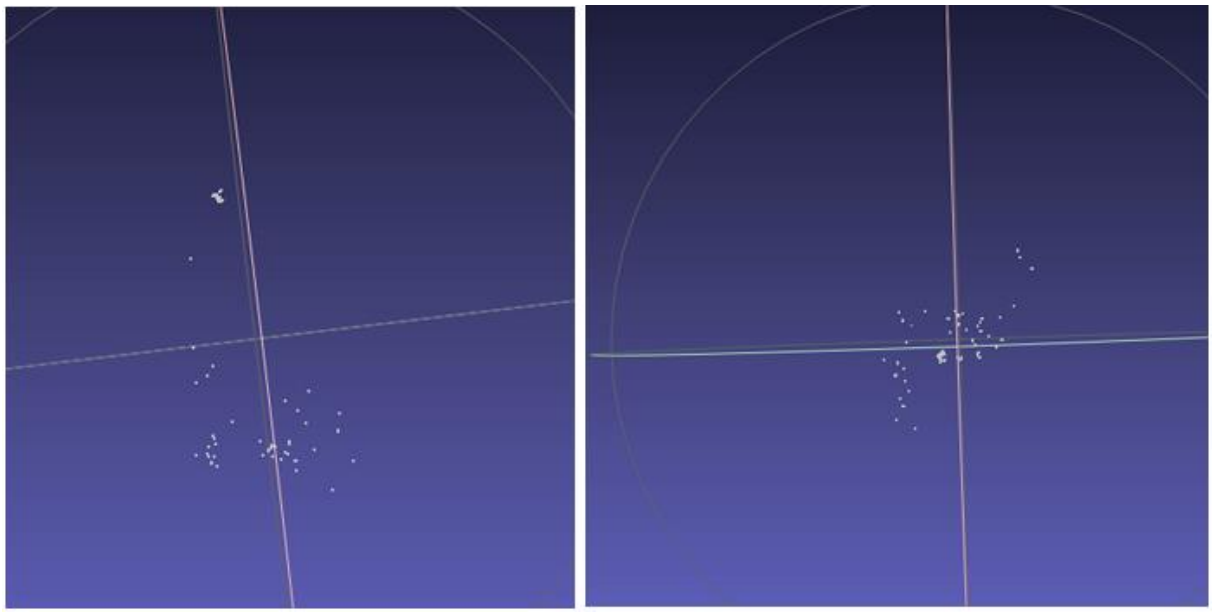

Figura 4.21 – Nuvem de pontos do Origami exibidos no MeshLab. Visão supeior e visão frontal, respectivamente.

A nuvem de pontos da caneca foi a única que conseguiu reproduzir características que lembram o objeto. A visão superior mostra que a curvatura da caneca está presente e a visão frontal apresenta pontos que representam o texto presente nela. Apesar disso, os pontos não representam a caneca em sua totalidade.

Os outros dois objetos não foram recuperados de forma interessante. Os pontos adquiridos não foram suficientes para reproduzir a complexidade nem do cubo nem do origami. Como comentado na seção de correspondência de pontos o algoritmo usado para gerar os pontos de interesse não conseguiu extrair informações das regiões que mais predominavam, que eram regiões de extensa variação de cores.

Utilizando o aplicativo comercial *VisualSFM* como base da reconstrução densa em três dimensões para comparar com a mais completa realizada nesse projeto segue a figura 4.22. É possível verificar que os ponto reconstruído no trabalho seguem o formato da caneca.

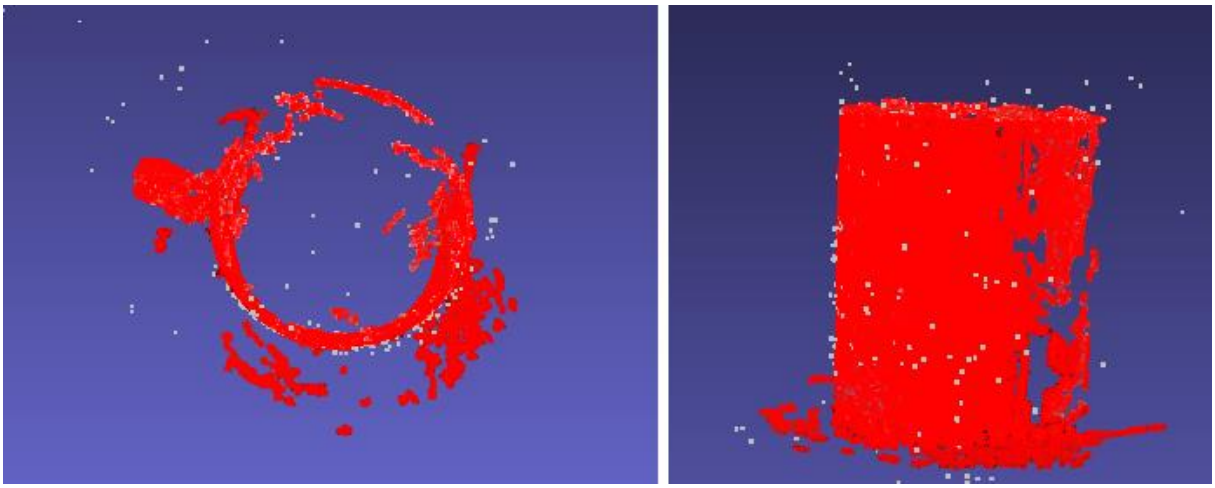

Figura 4.22 – Nuvem de pontos esparsa da caneca comparada com a reconstrução densa do aplicativo *VisualSFM* os pontos em vermelhos representam a reconstrução densa do aplicativo e os pontos branco a reconstrução esparsa do trabalho.

5 CONCLUSÃO E PERSPECTIVAS

Nesse projeto foi abordada a reconstrução esparsa em três dimensões de objetos a partir de múltiplas imagens. Esse método é uma continuação da reconstrução estéreo que utiliza apenas 2 imagens. Utilizar múltiplas imagens permite ter uma reprodução de maior fidelidade às características do objeto devido ao número maior de dados adquiridos na imagem. Por isso esse método foi utilizado

No trabalho foram fotografados três objetos sendo eles uma caneca, um icosaedro de origami e um cubo mágico. Os três objetos foram escolhidos por serem bem distintos um do outro. Cada objeto foi fotografado onze vezes sendo cada imagem com a câmera em uma posição diferente.

Realizou-se o processo em cinco etapas: Calibração, detecção e correspondência dos pontos de interesse, cálculo dos parâmetros da câmera, reconstrução 3D e unificação de pontos.

A calibração foi realizada com o programa de calibração fornecido pelo *MATLAB calibration toolbox* e foi possível encontrar os parâmetros intrínsecos da câmera a partir de dezessete imagens. A calibração teve resultados excelentes obtendo erro de reprojeção menores que um pixel.

A detecção e correspondência dos pontos de interesse foram computadas com a ajuda dos algoritmos SIFT e árvore-kd. O algoritmo SIFT utiliza uma diferença de gaussianas em diferentes espaços de escala para gerar vetores que descrevem os pontos de interesse. O método de indexação árvore-kd fez a comparação entre esses vetores para gerar as correspondências. Esse método prioriza a detecção de pontos localizados em regiões de alto contraste para imagens em escala de cinza e isso foi prejudicial em relação aos objetos escolhidos. Tanto o origami quanto o cubo apresentam alta variação de cores, porém com pouco contraste em escala de cinza. Com isso, a quantidade de pontos de interesse para esses objetos foi baixa.

O processo para computar os parâmetros das câmeras também serviu para filtrar pontos correspondentes que não representavam o mesmo ponto no espaço. A filtragem foi feita por meio do algoritmo RANSAC. Isso foi feito pois as matrizes de câmera são muito sensíveis a ruído e quaisquer pontos que possam causar

divergências devem ser retirados. Com isso, os pontos de interesse que já eram reduzidos foram diminuídos ainda mais nessa etapa. Contudo, as equações das matrizes de câmera necessitam apenas de 8 pontos correspondentes e os parâmetros foram calculados.

Com todos os parâmetros em mãos, as reconstruções 3D e a unificação dos pontos foram obtidas. Para evitar a propagação de erros, novamente foi necessário aplicar uma filtragem de pontos. Essa filtragem foi realizada para retirar pontos que não estavam reprojetados no local correto mesmo depois de feitas as otimizações.

Como resultado final uma nuvem de pontos esparsa única foi produzida a partir onze imagens dos objetos. A nuvem de pontos da caneca foi a única que conseguiu reproduzir características que lembram o objeto. Apresentando uma curvatura que é distintiva à caneca. Os outros dois objetos não foram recuperados de forma interessante. Os pontos adquiridos não foram suficientes para reproduzir a complexidade nem do cubo nem do origami.

O sistema não foi robusto o suficiente para conseguir reproduzir diferentes objetos de forma a reconhecer as formas completas do objeto. O principal problema para o cubo e para o origami foi uma aquisição de pontos de interesse pouco eficiente que não levava os principais atributos do objeto em conta na hora de obter os pontos. Apesar da aquisição de poucos pontos esses pontos estão posicionados na posição correta do objeto no espaço.

5.1 TRABALHOS FUTUROS

Diversas alternativas são aplicáveis para melhorar o sistema. É possível substituir o algoritmo de detecção de características para obter os pontos de interesse de diferentes formas. Pode também ser utilizada a obtenção de uma nuvem de pontos densa para representar o objeto completamente. A nuvem de pontos densa aproveitaria os poucos pontos obtidos e faria uma propagação na região em volta para obter uma grande quantidade pontos. A implementação da autocalibração poderá retirar o problema de posicionamento da câmera. A análise de 3 pontos de vista ao mesmo tempo com a implementação do tensor trifocal pode ser outra saída para trabalhos futuros pois diminui o gasto computacional na união dos pontos de vista.

Por fim, o sistema conseguiu reconstruir parte da caneca com uma nuvem de pontos esparsa, porém não foi tão robusto quanto esperado pois não é possível reconhecer o objeto por completo apenas com os pontos capturados por essa técnica. Espera-se que com as melhorias citadas ele possa ser utilizado em outras aplicações como por exemplo na detecção de falhas em linhas de transmissão.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] OROSKI, E. Identificação de Falhas em Espaçadores de Linhas de Transmissão Utilizando Visão Estéreo e Redes Neurais Artificiais, Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.DM-456/2011, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF,124p.

[2] Birchfield, **S.** (1998). An introduction to projective geometry (for computer vision). Unpublished Note, Stanford University, 1–22. Retrieved from http://isa.umh.es/doct/v3d/projective.pdf

[3] O. Faugeras. Three-dimensional computer vision: A geometric viewpoint. The MIT press, Cambridge,MA, 1993.

[4] Richard Hartley and Andrew Zisserman (2003). Multiple View Geometry in computer vision. Cambridge University Press. ISBN 0-521-54051-8.

[5] Lowe, D. G. (1999). Object recognition from local scale-invariant features. Proceedings of the Seventh IEEE International Conference on Computer Vision, 2([8), 1150–1157. http://doi.org/10.1109/ICCV.1999.790410.

[6] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91–110.

[7] M.A. Fishler and R.C. Boles. "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography". Comm. Assoc. Comp, Mach., Vol 24, No 6, pp 381-395, 1981

[8] Pinto, Daniel de Carvalho Cayres "Sala Inteligente Utilizando Geometria de Múltiplas Imagens" – Rio de Janeiro: Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2013.

[9] Besl, P. J., & McKay, H. D. (1992). A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence. http://doi.org/10.1109/34.121791

[10] Martin A. Fischler, Robert C. Bolles (Junho 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography" (inglês). Comm. of the ACM, Vol 24, pp 381-395.

[11] Bentley, J. L.(1975). "Multidimensional binary search trees used for associative searching". Communications of the ACM 18 509.

[12] Santos, R. F., Traina, A. J. M., & Traina, C. (1999). Algoritmos para Indexação em Bases de Dados Através de Estruturas. Universidade de São Paulo

[13] CAETANO S. J., H. (2011). “Reconhecimento de falhas em espaçadores de linhas de transmissão utilizando reconstrução 3D e redes neurais artificiais”, Trabalho de Graduação em Engenharia de Controle e Automação, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 91p.

[14] Don Koks, (2006) Explorations in Mathematical Physics, Springer Science Business Media, LLC. ISBN 0-387-30943-8. Ch.4, pps 147 et seq. A Roundabout Route to Geometric Algebra'

[15] http://labts.troja.mff.cuni.cz/~machl5bm/sift/

[16] http://www.decom.ufop.br/menotti/rp122/slides/

[17] Gonzalez, R. C., & Woods, R. E. (2009). Digital Image Processing, Third Edition. Journal of Biomedical Optics, 14(2), 029901.

[18] Seitz, Steve. Passive 3D photography. Carnegie Melon University. https://www.cs.cmu.edu/~seitz/course/CVPR99/seitz-passive.pdf.

[19] Coxeter, H. S. M., 2003. Projective Geometry, 2nd ed. Springer Verlag.

ANEXO

Código utilizado para rodar o programa:

```
%% CÓDIGO FEITO PARA O TCC - LUCAS ARAGÃO BESSA
%  Reconstrução 3D objetos a partir de múltiplas imagens


addpath(genpath('vl_feat'));
addpath(genpath('Rt'));
addpath(genpath('vgg'));


%% Incluir imagens

data.imagem{1}='images/CANECA/CANECA001.jpg';
data.imagem{2}='images/CANECA/CANECA002.jpg';
data.imagem{3}='images/CANECA/CANECA003.jpg';
data.imagem{4}='images/CANECA/CANECA004.jpg';
data.imagem{5}='images/CANECA/CANECA005.jpg';
data.imagem{6}='images/CANECA/CANECA006.jpg';
data.imagem{7}='images/CANECA/CANECA007.jpg';
data.imagem{8}='images/CANECA/CANECA008.jpg';
data.imagem{9}='images/CANECA/CANECA009.jpg';
data.imagem{10}='images/CANECA/CANECA010.jpg';
data.imagem{11}='images/CANECA/CANECA011.jpg';

% data.imagem{1}='images/ICOSAEDRO/ORI001.jpg';
% data.imagem{2}='images/ICOSAEDRO/ORI002.jpg';
% data.imagem{3}='images/ICOSAEDRO/ORI003.jpg';
% data.imagem{4}='images/ICOSAEDRO/ORI004.jpg';
% data.imagem{5}='images/ICOSAEDRO/ORI005.jpg';
% data.imagem{6}='images/ICOSAEDRO/ORI006.jpg';
% data.imagem{7}='images/ICOSAEDRO/ORI007.jpg';
% data.imagem{8}='images/ICOSAEDRO/ORI008.jpg';
% data.imagem{9}='images/ICOSAEDRO/ORI009.jpg';
% data.imagem{10}='images/ICOSAEDRO/ORI010.jpg';
% data.imagem{11}='images/ICOSAEDRO/ORI011.jpg';

% data.imagem{1}='images/CUBO/CUBO001.jpg';
% data.imagem{2}='images/CUBO/CUBO002.jpg';
% data.imagem{3}='images/CUBO/CUBO003.jpg';
% data.imagem{4}='images/CUBO/CUBO004.jpg';
% data.imagem{5}='images/CUBO/CUBO005.jpg';
% data.imagem{6}='images/CUBO/CUBO006.jpg';
% data.imagem{7}='images/CUBO/CUBO007.jpg';
% data.imagem{8}='images/CUBO/CUBO008.jpg';
% data.imagem{9}='images/CUBO/CUBO009.jpg';
% data.imagem{10}='images/CUBO/CUBO010.jpg';
% data.imagem{11}='images/CUBO/CUBO011.jpg';



%% Parametros da Camera

fprintf('\n Adquirindo Par□tros da C□ra...\n');
```

```
load('images/calibration/CameraParams.mat');
data.length = length(data.imagem);
data.imsize = size(imread(data.imagem{1}));
data.focal_length = cameraParams.FocalLength;

sizeFrame = 1024;

data = change_size(data,sizeFrame);

data.K = f2K(data.focal_length);


% ajustar o seed para que sempre seja o mesmo valor sempre
s = RandStream('mcg16807','Seed',10);
    RandStream.setGlobalStream(s);

%% SIFT e Correspondencia de Pontos
fprintf('\n Fazendo correspondencias...\n');
for foto = 1:data.length-1
    % Calcular os pontos de interesse e as correspondências
    par(foto) = match2viewSIFT(data,foto,foto+1);
end

%% Matriz Fundamental e matriz Essencial
fprintf('\n Encontrando matriz fundamental e essencial...\n');
for foto = 1:data.length-1
    %Estimar a matriz fundamental
    [par(foto).F,inliers] = ransacfitfundmatrix(par(foto).matches(1:2,:),
par(foto).matches(3:4,:), 0.002, 0);

    %Retirar os pontos outliers
    fprintf('%d inliers / dos %d Pontos Correspondentes = %.2f%%\n',
length(inliers), size(par(foto).matches,2),
100*length(inliers)/size(par(foto).matches,2));
    par(foto).matches = par(foto).matches(:,inliers);

    %Estimar para matriz essencial
    par(foto).E = transpose(data.K) * par(foto).F * data.K;

end


%% Matriz de camera
fprintf('\n Estimando matriz de c□ra...\n');
for foto = 1:data.length-1
    %Estimar posição da camera
    par(foto).Rt = RtFromE(par(foto),data);

    %Matriz de camera
    par(foto).P(:,:,1) = data.K*[eye(3) [0;0;0]];
    par(foto).P(:,:,2) = data.K*par(foto).Rt;

end

%% Reprojecao 3D para 2 pontos de vista
fprintf('\n Reprojecao 3D para dois pontos de vista...\n');
for foto = 1:data.length-1

    par(foto).X = triangulate(par(foto),data);

end
```

```
%% Bundle Adjustment
fprintf('\n Iniciando Bundle Adjustment...\n\n');
for foto = 1:data.length-1
    % Ajustar a função para o script escrito por Jianxiong Xiao de
Princeton
    Graph{foto} = pair2graph(par(foto),data);

    %Bundle Adjustment inicial para reduzir o erro para dois pontos de
    %vista
    Graph{foto} = bundleAdjustment(Graph{foto});
end

fprintf('\n Juntando nuvem de pontos...\n\n');
%Variaveis para juntar as nuvens de pontos
mergedGraph = Graph{1};


for foto = 2:data.length-1

    %Juntar as nuvens
    mergedGraph = merge2graphs(mergedGraph,Graph{foto});
    disp(mergedGraph.frames);

    %Ajustar as nuvens para o mesmo referencial
    mergedGraph = bundleAdjustment(mergedGraph);

%     %Remover os pontos com diferença de reprojeção maior que 2 pixels e 5
graus
[mergedGraph outliers] = removeOutlierPts(mergedGraph, 10,5);

    if sum(outliers)>0
        %Ajustar com a retirada de pontos
        mergedGraph = bundleAdjustment(mergedGraph);
    end
 end

ptCloud = pointCloud(mergedGraph.Str);

pcwrite(ptCloud,'sparse', 'PLYformat', 'binary');
```