CENTRO UNIVERSITÁRIO DA FEI

ABEL AUGUSTO RIBEIRO GUIMARÃES

CORRESPONDÊNCIA ENTRE REGIÕES DE IMAGENS POR MEIO DO ALGORITMO ITERATIVE CLOSEST POINT (ICP)

São Bernardo do Campo

2015

ABEL AUGUSTO RIBEIRO GUIMARÃES

CORRESPONDÊNCIA ENTRE REGIÕES DE IMAGENS POR MEIO DO ALGORITMO ITERATIVE CLOSEST POINT (ICP)

Dissertação de Mestrado apresentada ao Centro Universitário da FEI para a obtenção do título de Mestre em Engenharia Elétrica, orientada pelo Prof. Dr. Paulo Eduardo Santos.

São Bernardo do Campo

2015

Guimarães, Abel Augusto Ribeiro
Correspondência entre regiões de imagens por meio do algoritmo Iterative Closest Point (ICP) / Abel Augusto Ribeiro Guimarães. São Bernardo do Campo, 2015.
135 f. ill.

Dissertação - Centro Universitário da FEI.
Orientador: Prof. Dr. Paulo Eduardo Santos.

1. ICP. 2. ASIFT - Algoritmo. 3. SIFT – Algoritmo. 4. SURF - Algoritmo. 5. Kinect. I. Santos, Paulo Eduardo, orient. II. Título.

CDU 510.5

CENTRO UNIVERSITÁRIO DA FEI

APRESENTAÇÃO DE DISSERTAÇÃO
ATA DA BANCA EXAMINADORA

Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica

Mestrado

PGE-10

Aluno: Abel Augusto Ribeiro Guimarães **Matrícula:** 112108-6

Título do Trabalho: Correspondência entre regiões de imagens por meio do algoritmo iterative closest point (ICP).

Área de Concentração: Inteligência Artificial Aplicada à Automação

Orientador: Prof. Dr. Paulo Eduardo Santos

Data da realização da defesa: 25/06/2015

ORIGINAL ASSINADA

A Banca Examinadora abaixo-assinada atribuiu ao aluno o seguinte:

APROVADO ☒ REPROVADO ☐

São Bernardo do Campo, 25 de Junho de 2015.

MEMBROS DA BANCA EXAMINADORA

Prof. Dr. Paulo Eduardo Santos

Ass.: _____

Prof. Dr. Paulo Sérgio Silva Rodrigues

Ass.: _____

Prof. Dr. Roberto Hirata Júnior

Ass.: _____

VERSÃO FINAL DA DISSERTAÇÃO

ENDOSSO DO ORIENTADOR APÓS A INCLUSÃO DAS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

Este trabalho é dedicado a minha querida
esposa Carla, seu pai José e sua mãe Sylvia.
Ao meu pai Alfredo (In memoriam) e minha mãe Diva
e aos meus irmãos José, Diva, Fátima e Rachel (In memoriam),
que tanto incentivaram a minha vocação à ciência.

AGRADECIMENTOS

Os agradecimentos principais são direcionados ao meu Prof. Dr. Paulo Eduardo Santos , que com sua paciência e dedicação dirigiu a minha dissertação, para a conclusão deste trabalho. Agradeço também a todos os professores que se dedicam ao mestrado, bem como a todos os professores doutores participantes desta banca.

"Os que se encantam com a prática sem a ciência são como os timoneiros que entram no navio sem timão nem bússola, nunca tendo certeza do seu destino."

Leonardo da Vinci

RESUMO

Na literatura atual sobre correspondência de pontos podem-se destacar três algoritmos principais: ASIFT, SIFT e SURF, que procuram pontos correspondentes entre imagens através de descritores locais, fazendo a correspondência de pontos entre imagens semelhantes em várias cenas. No entanto, quando a correspondência entre a imagem de referência e a imagem a ser correspondida possui uma variação de latitude e longitude expressiva, esses algoritmos perdem em precisão e revocação. Diante disso, esta dissertação apresenta um algoritmo capaz de realizar a correspondência entre regiões das imagens, onde tem-se variações de latitude e longitude expressivas, em diferentes cenas e pontos de vista. O algoritmo desenvolvido neste trabalho foi chamado de Método de Correspondência utilizando o ICP (MCICP). Para a produção das regiões relevantes da imagem, foi utilizado um algoritmo de segmentação em grafo, que produz regiões que se aproximam dos objetos em estudo. Para a correspondência das regiões produzidas, utilizou-se o algoritmo Iterative Closest Point (ICP) como comparador dessas regiões, produzindo assim, a correspondência entre imagens.

Foram criados três cenários para testes, no primeiro cenário utilizamos somente objetos volumétricos, no segundo cenário somente objetos planos e no terceiro uma combinação entre eles. O sensor Kinect foi utilizado para a obtenção do mapa de profundidade de cada imagem.

Os resultados obtidos comprovam que o nosso modelo conseguiu fazer a correspondência entre regiões.

Nesta dissertação, comprovou-se que o ICP pode ser utilizado na correspondência de regiões, em que a métrica de comparação é o erro produzido pelas nuvens.

Palavras-chaves: ICP. Correspondência. Segmentação. Kinect. ASIFT. SIFT. SURF.

ABSTRACT

In the current literature on image correspondence, three algorithms can be highlighted: ASIFT, SIFT and SURF, that seek corresponding points on pairs of images using local descriptors, matching those points between similar images in various scenes. However, when the correspondence between the reference image and the image to be matched has an expressive latitude and longitude variation, these algorithms lose their precision and recall. Thus, in this dissertation we present an algorithm that performs the correspondence between regions of images where there is a significant latitude and longitude variation in different scenes and views. The algorithm developed in this work is called Method of Correspondence using ICP (MCICP). For the production of the relevant image regions, the algorithm used a graph segmentation algorithm, which produces regions approaching the objects under study. For the correspondence of the regions produced we used the algorithm Iterative Closest Point (ICP) to compare regions between the images, thereby matching these regions.

Three scenarios were created for evaluation proposes, the first scenario only uses volumetric objects, the second scenario only uses planar objects and the third uses a combination of them. A Kinect sensor was used for obtaining the depth map for each image.

The results show that our model has achieved the correspondence between regions.

This dissertation shows that ICP serves as an algorithm of correspondence where the comparing metric is the error produced by the analysed clouds. Thus, it can be used in matching areas for comparing images.

Keywords: ICP. Correspondence. Segmentation. Kinect. ASIFT. SIFT. SURF.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Correspondência entre imagens é considerado um dos problemas mais relevantes no desenvolvimento da visão computacional. Entre as várias abordagens da correspondência entre imagens, destacamos as que são realizadas através de pontos de interesse em imagens bidimensionais. Esse procedimento consiste em encontrar pontos de interesse que possuam características locais de uma vizinhança de estrutura invariante com as transformações tanto geométricas quanto fotométricas.

Um ponto de interesse, também denominado ponto-chave, é representado por uma pequena vizinhança local e se define pelas coordenadas de imagem, tamanho e forma da estrutura da vizinhança.

Tais coordenadas representam alguma característica local descriminante.

Quando se trata da *correspondência de pontos* entre imagens, pelo menos três fatores devem ser considerados:

a) invariância à mudança de escala;

b) invariância às transformações afins;

c) descritores locais.

No caso de mudanças de escala significativa, ou seja, quando a imagem está se distanciando ou aproximando em relação ao eixo focal da câmera, é preciso determinar quais os pontos de interesse que são invariantes em relação à forma da estrutura local de sua vizinhança, para que não haja perda de sua significância.

A escala de uma estrutura local de um ponto de interesse está diretamente relacionada à resolução em que a estrutura é representada em uma imagem, respeitado o mínimo de resolução para que uma estrutura local permaneça íntegra; abaixo deste mínimo a estrutura perde a significância. Sem a propriedade de invariância de escala, a complexidade de um algoritmo de correspondência aumenta. Para que a estrutura local mantenha a sua integridade nas várias resoluções do espaço escalar, é preciso que as imagens tenham uma resolução alta, o que significa que o tamanho das imagens a serem combinadas deva ser grande, acarretando assim um aumento da complexidade do algoritmo como também um aumento do tempo de computação.

Além dos ajustes de escala, para que haja correspondência dos pontos de interesse, também é necessário que haja a invariância sob transformações afins.

Uma transformação afim é uma transformação linear seguida de uma translação.

Na transformação afim, o importante é resolver sua influência nos principais parâmetros que definem a aparência de uma estrutura local: sua localização, seu tamanho e a forma da vizinhança do ponto de interesse.

Outro ponto importante em relação à correspondência são os descritores locais.

Uma vez que as características locais sejam identificadas na imagem, as suas propriedades devem ser capturadas por descritores. Há muitas possibilidades para descrever as estruturas locais de uma imagem. A descrição é necessária para comparar e encontrar estruturas semelhantes entre as imagens comparadas. O ponto crítico é a complexidade dos descritores, muita complexidade na descrição dificultará a computação.

Pode-se utilizar a transformação afim da forma da vizinhança do ponto de interesse como um descritor. Dadas as regiões covariantes afins, pode-se compensar a deformação geométrica para se calcular um descritor invariante afim. No entanto, a invariância à rotação e mudanças de iluminação também devem ser tratadas no processo de identificação de um descritor para se obter um completo cálculo de similaridade entre os pontos.

Alguns dos principais algoritmos que utilizam a abordagem da correspondência entre pontos em imagens bidimensionais são: *Scale Invariant Feature Transform* (SIFT) (LOWE, 1999), em que a principal contribuição é um método para a obtenção da invariância da escala, *Affine Scale Invariant Transform* (ASIFT) (MOREL; YU, 2009), em que a principal contribuição é na invariância à transformada afim e *Speed Up Robust Features (SURF)* (BAY et al., 2006), em que a principal contribuição é o uso de filtros derivativos melhorando assim a performance do sistema.

Nesta dissertação, deu-se ênfase para obter correpondência considerando aspectos globais da imagem e não correspondência local dos pontos da imagem.

O método de correspondência global utilizando regiões desenvolvido neste trabalho foi baseado no *Iterative Closest Point* (ICP) (KJER et al., 2010).

O ICP consegue fazer o alinhamento de uma nuvem de pontos considerada como modelo em comparação com outra nuvem de pontos considerada como dados, trazendo como resposta a transformação (rotação, translação e escala) que sofreu a nuvem de dados ao se alinhar com a nuvem modelo e o erro da distância entre os pontos das nuvens comparadas.

O principal uso do ICP é a produção do escaneamento 3D de uma cena, pois o alinhamento das nuvens de pontos traz o alinhamento das superfícies dos objetos em cena, formando assim, o escaneamento de toda a cena.

Esta dissertação desenvolveu um algoritmo com o ICP, criando um método para correspondência entre imagens, usando o erro das nuvens de pontos como comparador das regiões.

Pretende-se, aqui, apresentar um novo foco para a correspondência de imagens, substituindo o foco dos descritores locais de uma imagem bidimensional por um aspecto global com estrutura geométrica tridimensional onde a correspondência depende do mapa de profundidade da cena para poder comparar as regiões nelas contidas.

O objetivo principal desta dissertação é o estudo e o desenvolvimento de um algoritmo baseado no ICP capaz de fazer a correspondência entre regiões de imagens, mesmo que essas regiões sofram uma transformação significante (rotação, translação e escala) de pontos de vista distintos de uma mesma cena.

Utilizou-se um sensor Kinect para o fornecimento de imagens RGBD, onde D é a disparidade da imagem (captura da profundidade da imagem) e o RGB é a cor da imagem, que neste caso foi capturada com uma resolução de 640x480 pixels. Pixels representam os pontos das imagens. Eles têm cores e intensidades diferentes e são dispostos em uma matriz bidimensional. Chama-se mapa de disparidade a matriz bidimensional que informa a profundidade de cada pixel da imagem RGB.

Para a obtenção das regiões nas imagens que servirão de "modelos" e de "dados" na comparação entre elas, será necessário utilizar um algoritmo de segmentação, que nesta dissertação, foi o algoritmo proposto por (FELZENSWALB; HUTTENLOCHER, 2004). Um algoritmo de segmentação em grafo produz dois tipos de regiões: as regiões "finas" e as regiões "grosseiras" (FELZENSWALB; HUTTENLOCHER, 2004). As regiões finas apresentam um maior número de detalhes, refinando as informações dos seus componentes, produzindo, assim, mais regiões. Enquanto que as regiões grosseiras produzem regiões mais globais da imagem. As regiões "finas" são muitas vezes regiões de áreas muito pequenas; isso pode acarretar em erro na comparação das regiões, pois quanto menor a área de uma região, maior a probabilidade de correspondência com qualquer outra região, pois essa perde a significância de sua forma.

Diante disso, buscou-se trabalhar, aqui, com as regiões grosseiras, que possuem maior significância na correspondência das imagens. À partir daí, o ICP foi utilizado para a obtenção da correspondência das regiões entre as imagens.

Nesta dissertação desenvolveu-se uma nova função para o algoritmo ICP, que não o método de escaneamento 3D. O ICP é aqui utilizado como um comparador de regiões entre imagens, utilizando para isso o erro da distância entre as nuvens. O *Iterative Closest Point* (ICP) implementado nesta dissertação foi proposto por (KJER et al., 2010), que contém a técnica da árvore *k-d* que é um facilitador na procura da correspondência das nuvens.

No próximo capítulo, serão abordados os modelos de correspondência de pontos mais utilizados, que compõem a revisão bibliográfica deste trabalho; em seguida, o capítulo dos conceitos fundamentais, que apresentará as teorias utilizadas dos algoritmos que fazem

parte do nosso algoritmo. No capítulo método da correspondência de regiões entre imagens por meio do ICP é descrito detalhadamente todo o funcionamento do MCICP desde a captura da imagem até sua parte de correspondência. No capítulo do arranjo experimental detalharemos as cenas e como as várias posições de captura das imagens são tomadas e como os objetos foram escolhidos e dispostos. No capítulo 6 mostraremos os resultados descrevendo todos os gráficos precisão versus revocação de todas as cenas. Em seguida entraremos na discussão revendo os resultados e analisando-os determinando o porquê de tais resultados.

2 TRABALHOS RELACIONADOS

O embasamento teórico desta dissertação tem origem no estudo da visão computacional. Visão computacional é uma vasta área de conhecimento que tem como um de seus tópicos a correspondência de pontos em imagens. Neste capítulo serão tratadas as teorias relacionadas à correspondência de pontos entre imagens.

As principais abordagens para a correspondência de pontos entre imagens utilizaram características locais de correspondência através de pontos de interesse. Esta área teve seu inicio com os detectores que buscam por pontos que possuem um forte gradiente de intensidade em relação aos seus pontos vizinhos.

O detector Hessiano foi proposto no trabalho de (BEAUDET, 1978) e utiliza uma matriz com derivada de segunda ordem no ponto de interesse em relação aos pontos da vizinhança; essa matriz procura pontos da vizinhança (limitados por uma janela) que possuem uma forte variação em duas direções ortogonais. Esta técnica utiliza uma Gaussiana para a suavização da superfície da vizinhança. Outro detector relevante é o Harris/Förstner *detector* de (FÖRSTNER; GÜLCH, 1987, HARRIS; STEPHENS, 1988) utilizado na busca por pontos de interesse com características locais (MORAVEC, 1983). O detector Harris utiliza uma Gaussiana, como peso nos pontos da vizinhança, para obter sua primeira derivada. O ponto forte destes detectores é a possibilidade de produzir pontos de interesse como um canto (esquina), embora eles não só detectem pontos de canto, mas também incluam vários pontos em gradientes de sua vizinhança.

Os detectores Hessiano e Harris, quando não se tem uma mudança expressiva da escala, são robustos em relação às transformações no plano de rotação, ruído e iluminação; porém, se a mudança de escala é significativa, ocorre um aumento da não-correspondência, em virtude da Gaussiana aplicada possuir uma base de escala σ fixa para as derivadas da estrutura da vizinhança.

Para que a estrutura da vizinhança permaneça detectável nos vários níveis de escala é preciso obter uma *"função de assinatura"* ou "*kernel*" que irá simular as várias escalas obtendo os pontos-chaves que serão invariantes à mudança de escala, produzindo assim um espaço escalar (WITKIN, 1983). O operador que produz este espaço escalar normalizado é a Gaussiana $G(\mathbf{x}, \sigma)$ e suas derivadas (LINDEBERG, 1994).

Na mesma linha de raciocínio, (LINDEBERG, 1998) propôs um operador que busca por uma variação extrema do espaço escalar, chamado de operador Laplaciano da Gaussiana (LoG), definido matematicamente como:

$$L(\mathbf{x}, \sigma) = \sigma^2(I_{xx}(\mathbf{x}, \sigma) + I_{yy}(\mathbf{x}, \sigma)). \quad (1)$$

O operador LoG, figura 1, corresponde a uma estrutura de centro com rebordo circular, com pesos positivos na região do centro e pesos negativos na estrutura de anel circundante. Com esse formato, ele irá produzir respostas máximas, se e somente se aplicado a uma vizinhança que contém uma forma semelhante a um círculo com escala correspondente. Ao simular vários espaços de escala, o operador LoG irá enfatizar as estruturas em forma circular. Desta forma, o ponto de interesse que poderá ser obtido, tanto por Harris quanto pelo Hessiano, será aceito quando o ponto for encontrado nos níveis subsequentes de escala simulados.

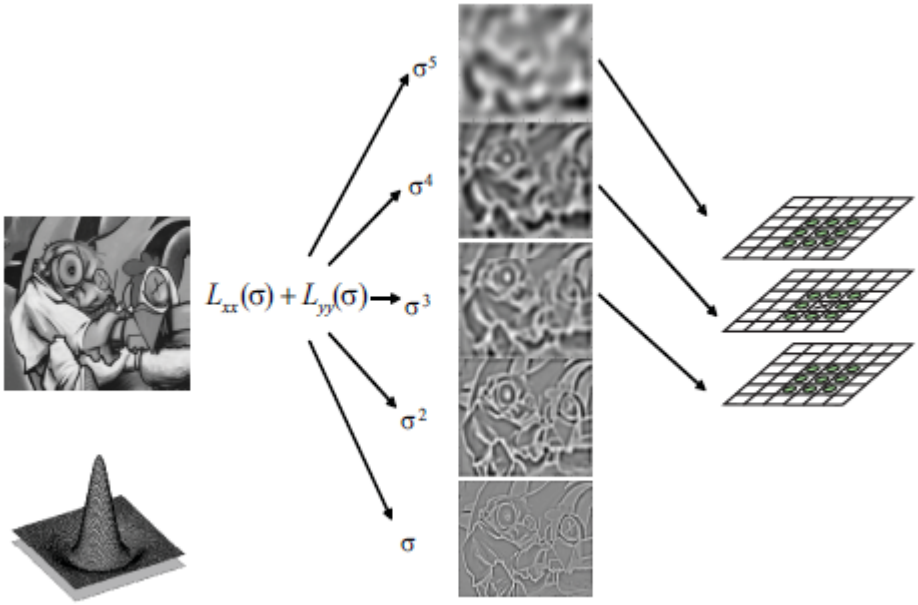


Figura 1 – O operador Laplaciano da Gaussiana (LoG).
Fonte: Autor "adaptado de" (MIKOLAJCZYK, 2002).
Nota: Este operador busca por uma variação extrema do espaço escalar 3D (3x3x3) da função LoG.

Outro operador conhecido é o *Difference of Gaussian (DoG) (WILSON; GIESE, 1977);* ele é obtido pela diferença entre dois espaços escalares Gaussianos, que são aproximados por diferenças Gaussianas $D(x, \sigma)$; é mais eficiente quando obtido por duas diferentes escalas adjacentes separadas por um fator k:

$$D(x, \sigma) = (G(x, k\sigma).G(x, \sigma)) \star I(x), \tag{2}$$

onde $G(x, k\sigma)\, e\, G(x, \sigma)$ são dois níveis da escala Gaussiana que faz a convolução com a imagem $I(x)$. Quando o fator k for constante, ocorre à normalização da escala (LOWE, 2004). Pode-se, portanto, dividir cada oitava de escala em um número igual de K intervalos, tal que $k = 2^{1/K}$ e $\sigma_n = k^n \sigma_0$. Para um cálculo mais eficiente, o resultado do espaço escalar pode ser implementado com a pirâmide Gaussiana, na qual as imagens são remontadas

numa escala menor para cada oitava (figuras 2 e 3). Onde na figura 2 a imagem foi reduzida em 5 oitavas e a figura 3 mostra para cada oitava da pirâmide a produção de 5 níveis de Gaussiana, aplicando a diferença a cada dois níveis subjacentes (DoG). E à direita temos a busca persistência do ponto de interesse nos vários níveis.

Os pontos de interesse, usando DoG, são definidos como locais extremos no plano da imagem. Tais pontos são encontrados comparando-se o valor de cada ponto $D(x, \sigma)$ com os 8 vizinhos do próprio nível de escala e com os 9 vizinhos acima e abaixo da escala, em um total de 26 vizinhos (figura 2).

Tanto o operador LoG quanto o operador DoG, proporcionam uma mudança de escala, facilitando aos detectores Harris e Hessiano a determinação da escolha do ponto de interesse, ou seja, aqueles pontos que se mantêm nos vários níveis de escala.

Existem algumas técnicas que combinam os detectores com os operadores. O detector Harris Laplaciano em combinação ao operador Laplaciano LoG foi proposto para aumentar o poder discriminativo da vizinhança do ponto (MIKOLAJCZYK; SCHMID, 2001, MIKOLAJCZYK; SCHMID, 2004). Ele combina a especificidade do operador LoG com a determinação do ponto de interesse do detector Harris para estruturas de canto com o mecanismo de seleção por escala (LINDEBERG, 1998). O método constrói os níveis dos espaços de escala utilizando o operador Laplaciano. Em seguida, ele usa a função de Harris para localizar pontos candidatos em cada nível de escala e seleciona aqueles para os quais o Laplaciano alcançará o extremo das escalas. Os pontos resultantes são robustos às alterações na escala, rotação de imagem, iluminação, e ao ruído da câmera. Além disso, eles são altamente discriminativos (MIKOLAJCZYK; SCHMID, 2001, MIKOLAJCZYK; SCHMID, 2005). Como desvantagem, no entanto, o detector de Harris Laplaciano normalmente retorna um número muito menor de pontos.

Uma versão atualizada do detector de Harris Laplaciano foi proposta com base em um critério menos rigoroso (MIKOLAJCZYK, 2002) a fim de simplificar a restrição adicional de que cada ponto tenha que cumprir simultaneamente duas condições máximas, uma para a função multi-escalar Harris e outra para o LoG, foi proposto um modelo simplificado em que pontos iniciais da função multi-escalar Harris determinam um limiar para o LoG. Como resultado, este detector modificado tem mais pontos de interesse com uma precisão ligeiramente inferior, o que resulta em melhor desempenho (MIKOLAJCZYK et al., 2005).

Como no caso do Harris Laplaciano, a mesma ideia pode ser aplicada para o Hessiano Laplaciano (MIKOLAJCZYK et al., 2005). O detector Hessiano Laplaciano tipicamente retorna mais pontos de interesse do que Harris Laplaciano como uma menor variação da repetibilidade.

Ambos os detectores Harris Laplaciano e Hessiano Laplaciano podem ser estendidos para produzir regiões covariantes afins. Isto é feito pelo seguinte esquema de estimativa

iterativa. O procedimento é inicializado com uma região circular devolvida pelo detector de escala invariante (MIKOLAJCZYK et al., 2005). Em cada iteração, constrói-se uma matriz de segundo momento da região e calcula-se os autovalores dessa matriz. Isto produz uma forma elíptica [1], o que corresponde a uma deformação local afim. Em seguida, aplica-se uma transformada (translação, rotação e escala) na imagem, de tal forma que estrutura da vizinhança do ponto de interesse que possui uma forma elíptica é transformada em uma estrutura circular, atualizando-se a localização e estimativa da escala da imagem transformada. Este procedimento é repetido até que os autovalores da matriz de segundo momento sejam aproximadamente iguais.

Como resultado deste esquema de estimação iterativa, obtém-se um conjunto de regiões elípticas que são adaptadas para os padrões de intensidade locais, de modo que as mesmas estruturas locais são cobertas apesar das deformações provocadas por alterações do ângulo de visão.

Em contraste com a abordagem de pontos, uma abordagem diferente para encontrar regiões covariantes afins foi proposta por (MATAS et al., 2002). Esta abordagem começa a partir de uma perspectiva de segmentação. Aplica-se um algoritmo de segmentação chamado de "divisor de águas" (*watershed*) para a imagem e se extraem regiões homogêneas e estáveis, alcançando a máxima estabilidade de uma região extrema *Maximally Stable Extremal Regions* (MSER).

Outras abordagens contemplam detectores e descritores para o ponto de interesse, melhorando assim a capacidade da correspondência dos pontos entre as imagens. Estas abordagens se referem ao *Scale Invariant Feature Transform* (SIFT), *Speed Up Robust Feature* (SURF) e *Affine Scale Invariant Feature* (ASIFT), estas três abordagem são consideradas as mais utilizadas e complementam todas as invariâncias em termos de escala, transformações afins de ponto de vista, iluminação ou ruído. Desta forma estas foram escolhidas como base de comparação em termos de correspondência para este trabalho e foram abordadas em detalhes nas próximas seções.

Outro detector que aborda a invariância afim é o *Affine Seeped Up Robust Feature* (ASURF) de (PANG et al., 2012) que segue os procedimentos similares aos do ASIFT em termos das transformações covariantes e simulações para obtenção da invariância afim, a única diferença é o método utilizado para a invariância em escala que neste caso utiliza-se o SURF, não sugerindo um resultado expressivo em relação ao ASIFT.

2.1 Scale Invariant Feature Transform (SIFT)

SIFT (LOWE, 2004) é um algoritmo que extrai características locais, pontuais e invariantes em termos de escala e translação, além de ter um detector, este possui um

[1] Como mostrado na figura 28 nos anexos.

descritor dos pontos de interesse que produz um histograma da orientação dos gradientes da vizinhança do ponto.

Na continuação do aprimoramento do trabalho descrito em, (LOWE, 2004), as características locais dos dados da imagem são transformadas em coordenadas de escala invariante. Um aspecto importante desta abordagem é que ela gera um grande número de características que densamente cobrem a imagem ao longo de toda a gama do espaço de escalas da imagem e suas características locais. Uma imagem típica do tamanho de 500 x 500 pixels irá dar origem a cerca de 2000 características estáveis. A seguir é descrito como obter este espaço escalar e de que forma os extremos dos pontos de interesse são detectados, como também os passos do algoritmo SIFT.

2.1.1 Detecção de extremos do espaço escalar do algoritmo SIFT

Na detecção de extremos do espaço escalar procura-se a detecção de pontos chave, em que é utilizada uma abordagem de filtragem em cascata na qual um algoritmo eficiente examina os pormenores de candidatos locais em várias escalas identificando-os desta forma. A detecção destes pontos locais, que são invariantes à mudança de escala em uma imagem, pode ser realizada através da busca dos recursos estáveis em todas as escalas possíveis, utilizando uma função contínua de escala conhecida como espaço de escala (WITKIN, 1983, KOENDERINK, 1984, LINDEBERG, 1994) demonstraram que o espaço escalar pode ser obtido por uma função Gaussiana.

O primeiro passo do algoritmo é reduzir a resolução da imagem por oitavas; isto é, uma oitava em resolução reduz pela metade o tamanho da imagem produzindo assim uma pirâmide de oitavas como pode ser observado nas figuras 2, 3 e 4.

Para cada oitava é aplicada uma Gaussiana $G(x, y, \sigma)$, definindo um espaço escalar como uma função, $L(x, y, \sigma)$, que é produzida pela convolução da imagem de entrada $I(x, y)$ e uma variável de escala Gaussiana $G(x, y, \sigma)$, desta forma tem-se:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{3}$$

onde $*$ é o operador de convolução em x e y, e

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{4}$$

Para se detectar eficientemente os locais no espaço escalar onde pontos-chave estáveis se encontram, a proposta de (LOWE, 1999) foi utilizar a detecção de extremos no espaço escalar aplicando a função diferença Gaussiana (DOG) na convolução da imagem, $D(x, y, \sigma)$, que pode ser calculada pela diferença de duas escalas na qual foi aplicado um fator

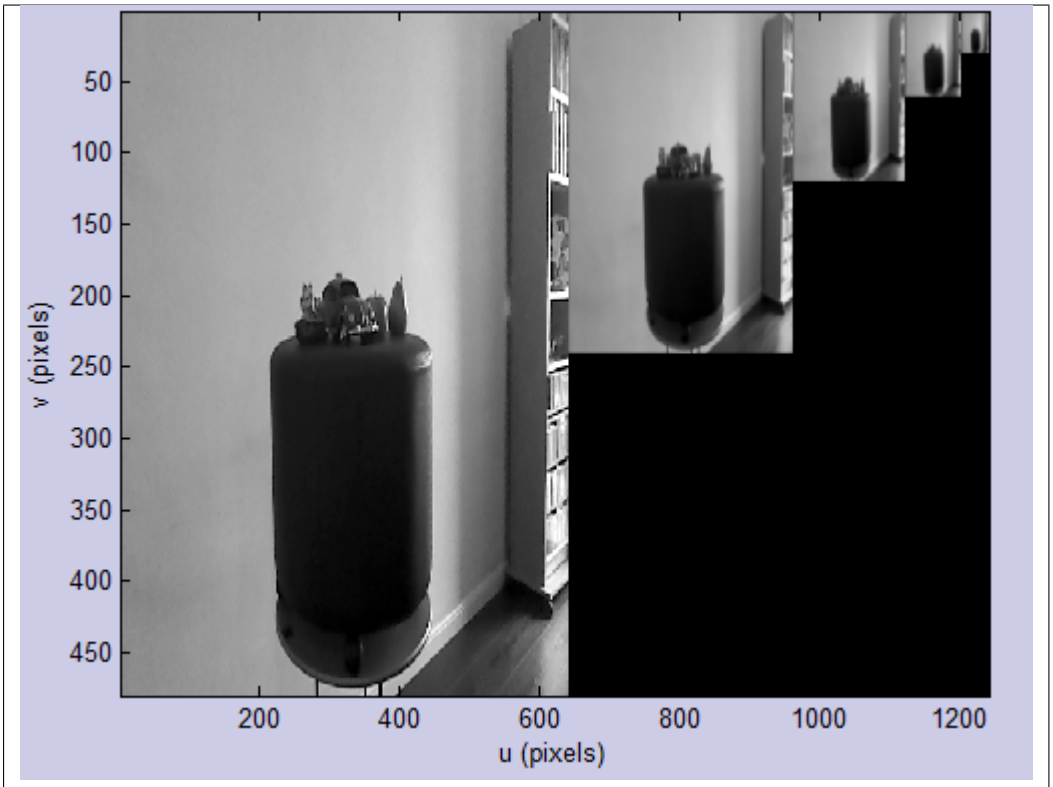
multiplicativo k:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{5}$$

$$= L(x, y, k\sigma) - L(x, y, \sigma), \tag{6}$$

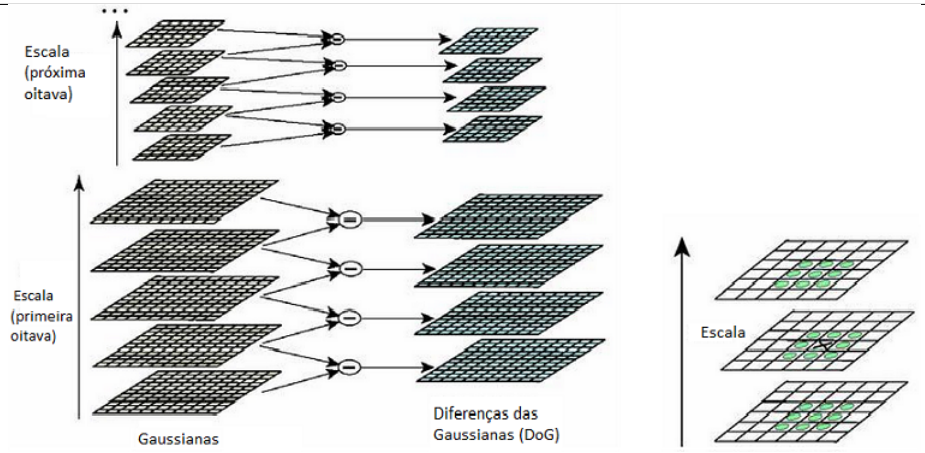
cujo exemplo de um resultado pode ser visto nas figuras 4b e 4d.

O próximo passo é eliminar os candidatos a pontos-chaves que possam ser instáveis, por terem pouco contraste com a vizinhança ou por pertencerem a uma fronteira.



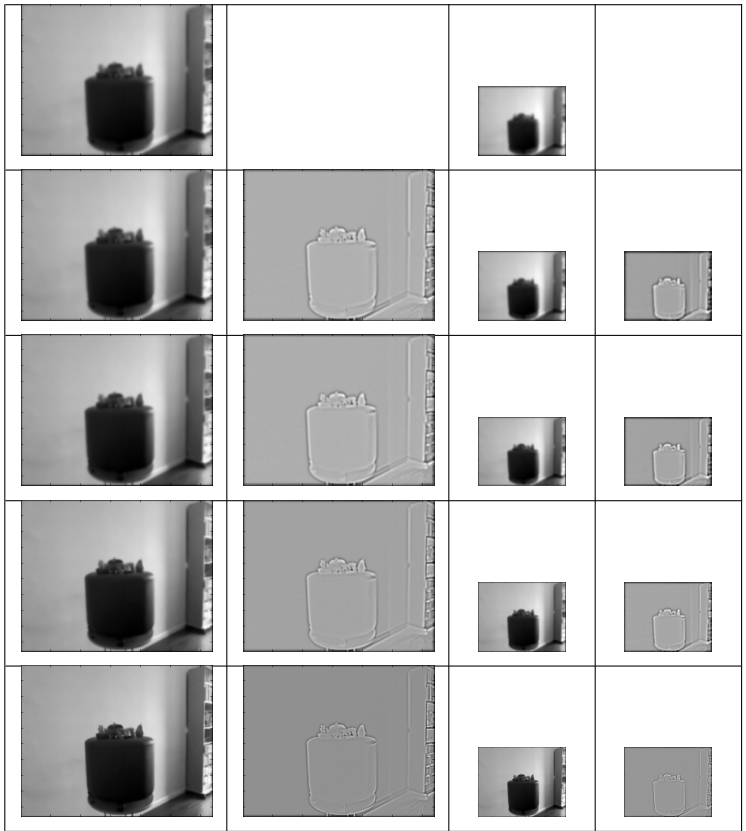
Pirâmide Gaussiana com cinco oitavas

Figura 2 – Pirâmide Gaussiana.
Fonte: Produzido pelo próprio autor



Espaço Escalar de DoG com as oitavas e à direita a procura dos vizinhos nas escalas subjacentes.

Figura 3 – Processo do espaço escalar em oitavas.
Fonte: Adaptado de (LOWE, 2004)



(a) $1^a oitava$ com 5 níveis de $L(x, y, \sigma)$	(b) (DoG) da $1^a oitava$	(c) Segunda Oitava	(d) (DoG) da $2^a oitava$

Figura 4 – Exemplo do processo do espaço escalar em oitavas.
Fonte: Produzido pelo próprio autor.

2.1.2 Precisão da localização do ponto-chave

Uma vez que os candidatos ao ponto-chave foram encontrados, quando da comparação do pixel e seus vizinhos, o passo seguinte é selecionar a localização exata de cada um dos pontos-chave selecionados no passo anterior utilizando o método de (BROWN; LOWE, 2002). Este método determina a localização interpolada máxima, nos pontos-chaves da pirâmide DoG, utilizando uma função quadrática $3 \times 3 \times 3$ em torno do ponto-chave localizado em $\mathbf{x} = (x, y, \sigma)$, procurando além da vizinhança do ponto no nível atual, as vizinhanças dos níveis acima e abaixo da oitava da pirâmide num total de 26 vizinhos. Os experimentos de (BROWN; LOWE, 2002) mostram que as buscas em níveis subsequentes proporcionam uma melhora substancial na correspondência; essa abordagem usa a expansão de Taylor (até os termos quadráticos) da função de espaço de escala, $D(x, y, \sigma)$, obtendo uma mudança de modo que a origem está no ponto de máxima ou mínima que será escolhido:

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial x}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial x^2}\mathbf{x}. \tag{7}$$

A mínima ou a máxima é localizada em

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial^2 x}\frac{\partial D}{\partial x}. \tag{8}$$

O valor de $D(\mathbf{x})$ na mínima ou máxima tem que ser maior que um limiar, $|D(\mathbf{x})| > th$.

2.1.3 Orientação de pontos-chave

A orientação de um ponto-chave θ, que corresponde à direção predominante do gradiente em torno do ponto-chave, é uma informação essencial para conferir invariância à rotação dos descritores SIFT. A orientação é calculada ao nível da pirâmide Gaussiana mais próxima à escala onde se detectou o ponto-chave. Computa-se o histograma de orientações dos gradientes desta imagem numa vizinhança em torno do ponto-chave. O histograma cobre 360^0 em 36 faixas de 10^0 de largura. A contribuição de cada ponto da vizinhança é ponderada pela magnitude do gradiente e por uma função que decresce com a distância ao centro da vizinhança. A figura 5 mostra um exemplo de imagem em que estão indicados os pontos-chave e as correspondentes orientações.

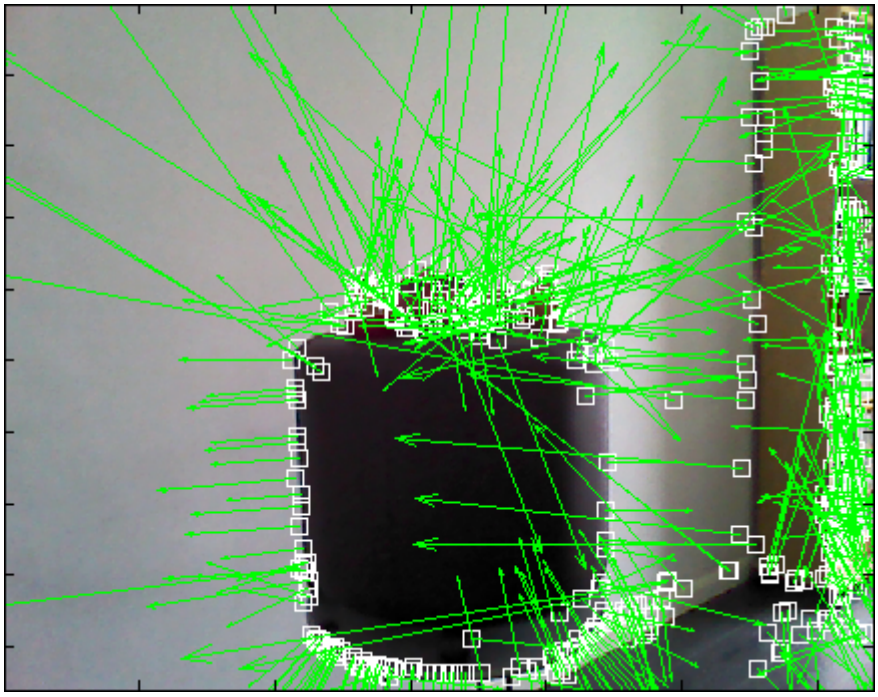


Figura 5 – Resultado das orientações dos pontos-chave.
Fonte: Produzido pelo próprio autor.

2.1.4 O descritor SIFT

No passo seguinte é obtido o descritor SIFT, que serve como uma característica do ponto para a correspondência entre outros pontos de outra imagem. O descritor SIFT localiza um conjunto de orientações de gradientes numa grade regular de 16x16 ao redor do ponto-chave escolhido no espaço de escala descrito acima.

A partir disso o descritor SIFT prepara um histograma desses gradientes criando uma grade de 4×4 na qual cada grade possui 8 orientações desses gradientes formando assim um vetor $4 \times 4 \times 8 = 128$ caracterizando unicamente a chave local como mostrado na figura 6.

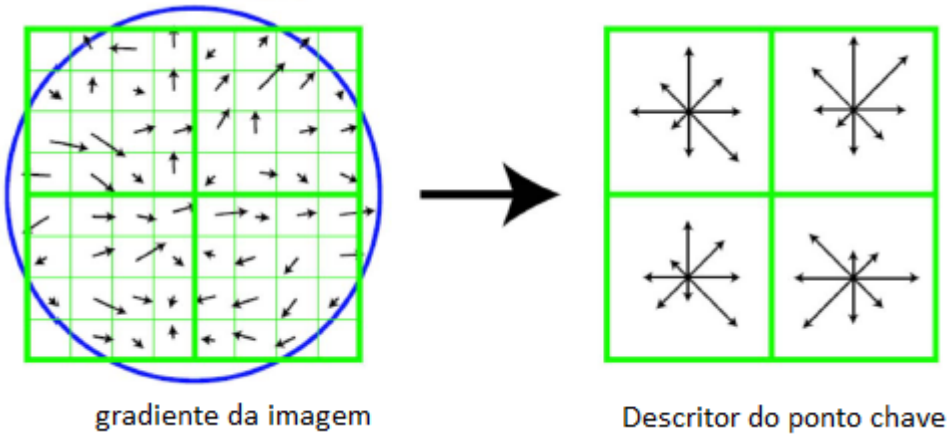


Figura 6 – Descritor SIFT.
Fonte: Autor "adaptado de" (LOWE, 2004)

2.2 Speeded Up Robust Features (SURF)

Seguindo os mesmos passos do SIFT, que utiliza a diferença das Gaussianas para a obtenção dos descritores em um espaço escalar, o *Speeded Up Robust Features* (SURF) *Detector* (BAY et al., 2006) baseia-se na matriz Hessiana para a obtenção dos seus descritores no espaço escalar, melhorando a performance do sistema.

2.2.1 Descritor SURF

O descritor SURF pode ser descrito da seguinte forma:

Dado um ponto $p = (x, y)$, numa imagem I, a matriz Hessiana $H(p, \sigma)$ em p na escala σ, é definida como:

$$H(p, \sigma) = \begin{bmatrix} Lxx(p, \sigma) & Lxy(p, \sigma) \\ Lxy(p, \sigma) & Lyy(p, \sigma) \end{bmatrix}, \tag{9}$$

onde $Lxx(p, \sigma)$ é a convolução Gaussiana da derivada de segunda ordem $\frac{\partial^2}{\partial^2 x} g(\sigma)$ com a imagem I no ponto p, seguindo também esta forma para obtenção de $Lxy(p, \sigma)$ e $Lyy(p, \sigma)$.

O SURF combina um detector Hessiano-Laplace com um filtro próprio formando um descritor de característica seguindo a estratégia do SIFT porém, ao invés de construir um histograma baseado nas orientações dos gradientes da vizinhança, como é feito no SIFT, é computado um somatório estatístico $\sum dx, \sum |dx|, \sum dy$ e $\sum |dy|$, que resulta em um descritor de dimensão 64. Os filtros, que são simples janelas em $2D$ ditas "*Haar wavelets*", são mostrados na figura 7.

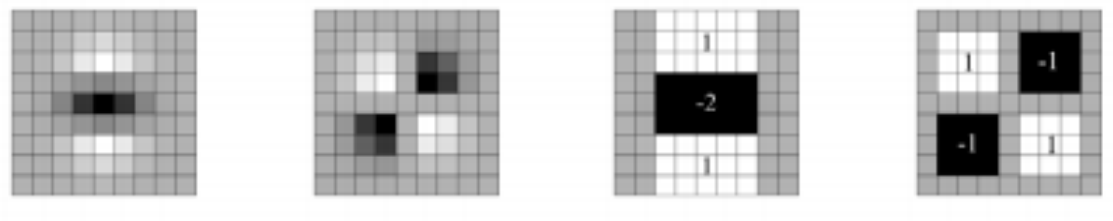


Figura 7 – Filtros do descritor SURF.
Fonte: Autor "adaptado de" (BAY et al., 2006)

2.3 Affine Scale Invariant Feature Transform (ASIFT)

O *Affine Scale Invariant Feature Transform* (ASIFT) (MOREL; YU, 2009) é um algoritmo que propõe uma extensão do algoritmo SIFT que seja totalmente invariante à transformação afim.

Pode-se modelar uma câmera afim quando da aquisição de uma imagem digital como pode ser visto na figura 8.

Um objeto plano pode ser descrito como:

$$\mathbf{u} = \mathbf{S_1 G_1 A \Gamma u_o}, \tag{10}$$

onde $\mathbf{u}$ é uma imagem digital e u_0 é uma vista do objeto plano com uma resolução frontal infinita (ideal); $\mathbf{\Gamma}$ e $\mathbf{A}$ são, respectivamente, o plano transladado e um mapa que projeta o plano devido ao movimento da câmara; G_1 é uma convolução Gaussiana que modela a resolução da imagem (como um borrão ótico) e S_1 é o operador de amostragem padrão em grade regular com malha unitária. O *kernel* Gaussiano é suficientemente amplo para garantir que não haja *aliasing* na amostragem, ou seja, o *kernel* Gaussiano vai conter a amostragem, a interpolação e a Gaussiana, desta forma tem-se: $\mathbf{I S_1 G_1 A \Gamma u_o} = \mathbf{G_1 A \Gamma u_o}$, onde $\mathbf{I}$ denota o operador de interpolação.

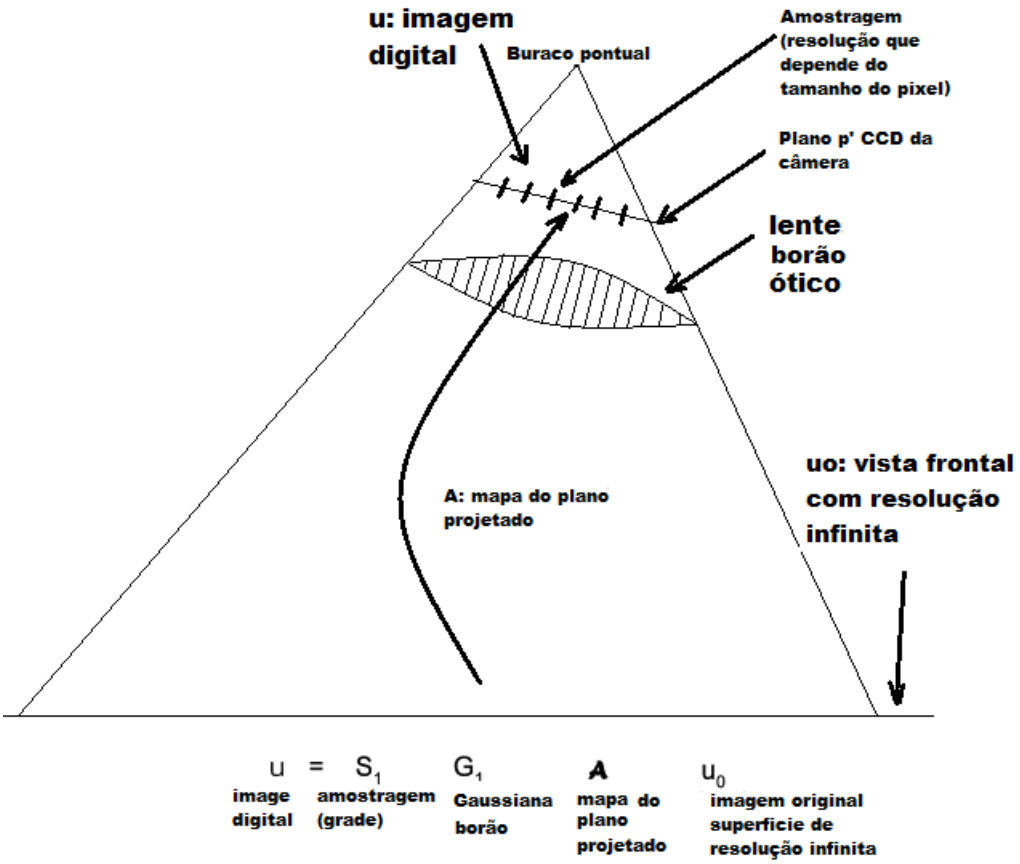


Figura 8 – No modelo de câmera projetiva $\mathbf{u} = \mathbf{S_1 G_1 A \Gamma u_o}$. $\mathbf{A}$ é uma transformação da projeção do plano (uma homografia). $\mathbf{G_1}$ é um filtro *anti-aliasing* Gaussiano. $\mathbf{S_1}$ é a amostragem do sensor CCD da câmera.
Fonte: Autor "adaptado de" (MOREL; YU, 2009).

Diferente de todos os outros algoritmos como: MSER, Harris-affine e Hessian-affine os quais normalizam todos os seis parâmetros de uma matriz afim, tal qual a matriz M apresentada na equação (11):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}}_{M} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{11}$$

onde u e v são as coordenadas da imagem após a transformação da matriz afim, o ASIFT simula três parâmetros e normaliza o resto com o SIFT.

Na simulação dos três parâmetros é feita a decomposição da matriz $\mathbf{A}$ da seguinte forma:

Simplifica-se o modelo acima, reduzindo $\mathbf{A}$ a um mapa afim.

Seja a matriz $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ uma mapa afim.

Pode-se desconsiderar a translação (e e f) da equação (11), assumindo que o eixo da câmera se encontra com o plano de imagem em um ponto fixo (ver figura 9).

Seguindo o princípio de decomposição singular do valor (SVD), a matriz **A** decompõe-se em:

$$A = \mathbf{H}_\lambda \mathbf{R}_1(\psi)\mathbf{T}_t\mathbf{R}_2(\phi) = \lambda \begin{bmatrix} cos\psi & -sin\psi \\ sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{bmatrix}, \qquad (12)$$

onde ϕ e $\theta = arccos(1/t)$ são respectivamente a longitude e a latitude dos ângulos em certo ponto de vista, ψ parametriza o spin da câmera e λ corresponde ao zoom. A figura 9 mostra a geometria desta decomposição.

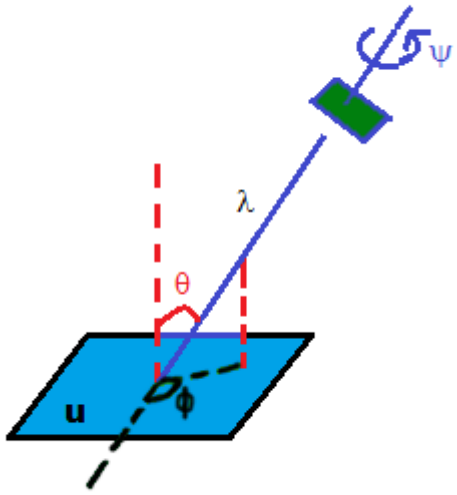


Figura 9 – Interpretação geométrica da decomposição equação (12).
Fonte: Autor "adaptado de" (MOREL; YU, 2009).
Nota: Na imagem, **u** em azul é um objeto físico plano. O pequeno paralelogramo em verde no topo à direita representa uma câmera olhando para **u**. Os ângulos ϕ e θ são, respectivamente, o eixo óptico da câmara sua longitude e latitude em relação a normal do plano **u**. Um terceiro ângulo ψ parametriza o spin da câmera, e λ é um parâmetro de zoom.

Dessa forma o ASIFT simula os três parâmetros da mudança de eixo da câmera que são: a mudança em relação à escala (zoom), a mudança do ângulo da latitude e a mudança do ângulo da longitude e, em seguida, aplica o SIFT que simula a mudança de escala e normaliza a rotação e a translação. Com isso o ASIFT de (MOREL; YU, 2009), consegue a invariância de uma transformação afim.

2.3.1 O algoritmo ASIFT

Algoritmo 1 Algoritmo retratado por (MOREL; YU, 2009)

a) Cada imagem é transformada para simular todas as possíveis transformações afins causadas pela mudança da orientação do eixo óptico da câmera de sua posição frontal. Estas transformações dependem de dois parâmetros: a longitude ϕ e a latitude θ. As imagens que são submetidas às rotações ψ, possuem uma inclinação do parâmetro $t = |\frac{1}{cos\theta}|$ (uma inclinação t na direção de x é uma operação $u(x, y) \rightarrow u(tx, y)$). Para imagens digitais, a inclinação é feita com subamostras direcionais. É requerida uma aplicação anterior de um filtro *anti-aliasing* na direção de x, utilizando a convolução por uma Gaussiana com um desvio padrão $c\sqrt{t^2 - 1}$. O valor utilizado para c=0.8 escolhido por (LOWE, 2004) certifica o melhor erro de *anti-aliasing* (MOREL; YU 2008).

b) Estas rotações e inclinações são praticadas com um número pequeno de variações na longitude e latitude. Isto faz com que a simulação de novas posições da imagem para ϕ e θ, esteja perto das que serão testadas para a sua correspondência em uma nova imagem.

c) Todas essas simulações são posteriormente comparadas utilizando o método SIFT para a sua correspondência.

A amostra da simulação das latitudes e das longitudes é especificada abaixo:

a) As latitudes são amostradas de forma que as associadas inclinações sigam a série geométrica $1, a, a^2, ..., a^n$, com $a > 1$. A escolha $a = \sqrt{2}$ é boa, pois tem comprometimento entre precisão e dispersão. O valor de n pode chegar até 5 ou mais. Desta forma as inclinações transitivas sobem para 32 ou além.

b) As longitudes para cada inclinação seguem uma série aritmética $0, b/t, ..., kb/t$, onde $b = 72^0$, parece também um bom comprometimento, e k é o último inteiro tal que $kb/t < 180^0$.

A figura 10 ilustra o resultado da saída do algoritmo ASIFT em duas imagens.

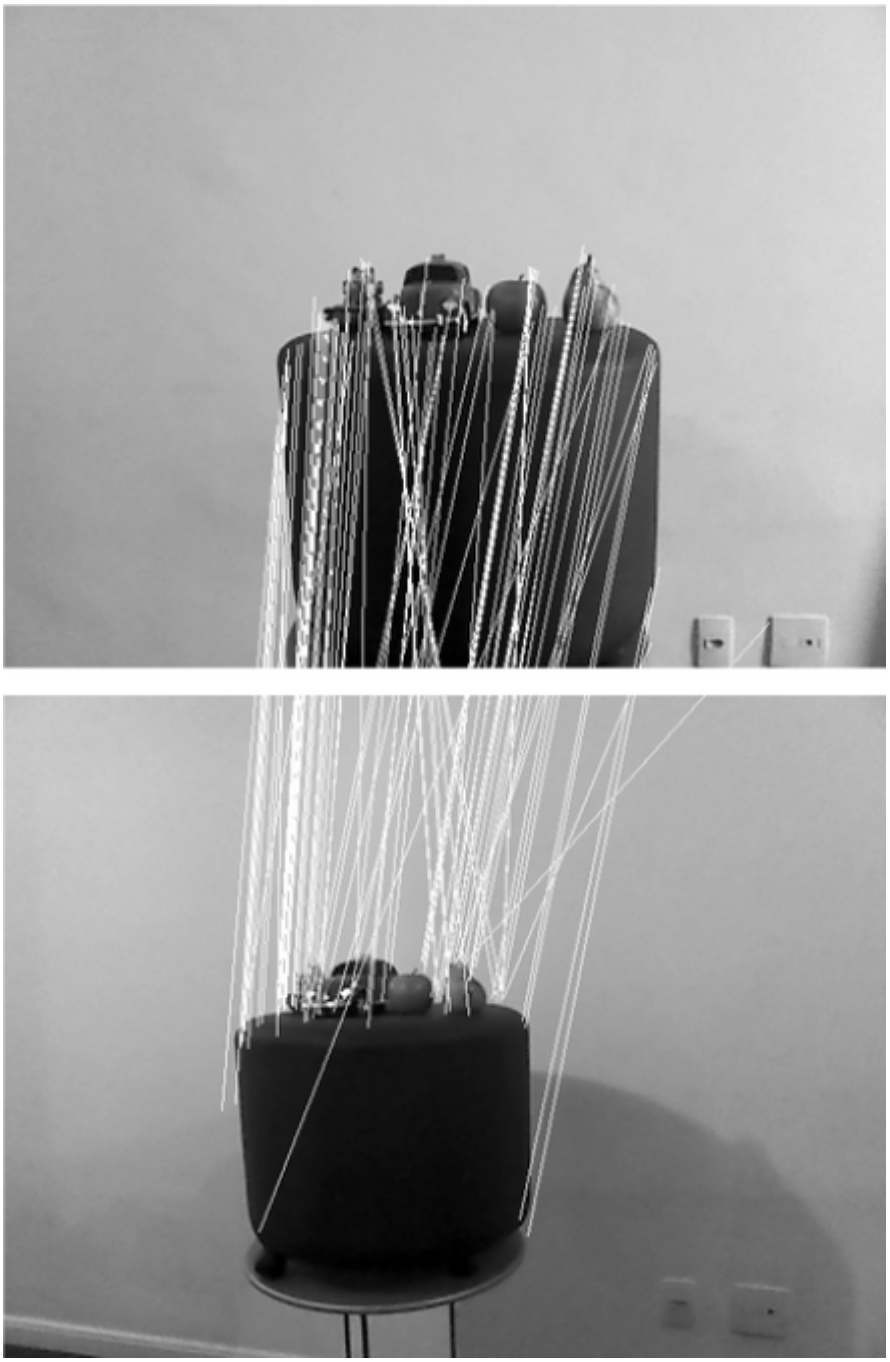


Figura 10 – Resultado do ASIFT na obtenção da correspondência dos pontos de interesse.
Fonte: Produzido pelo próprio autor.

Todas as técnicas apresentadas neste capítulo conseguem reconhecer pontos correspondentes de imagens até uma determinada transformação da pose na imagem, isto é, essas técnicas possuem uma limitação no alcance da transformação.

Assim, a motivação deste trabalho é usar o ICP que não utiliza a correspondência de pontos e sim, o erro entre as nuvens de pontos para a correspondência de áreas entre imagens em pontos de vista distintos. Considerou-se aqui o ICP que possue como variável

a profundidade e não simplesmente imagens bidimensionais. Partiu-se, assim, para o mapa de disparidade da imagem, dividindo a imagem em regiões para serem comparadas de uma forma global ao invés da comparação entre pontos, usando, para isso, um algoritmo de segmentação em grafo. No próximo capítulo, serão tratadas as técnicas que estudam segmentação, disparidade e correspondência por meio do *Iterative Closest Point* (ICP).

3 CONCEITOS FUNDAMENTAIS

Neste capítulo são introduzidos alguns conceitos básicos sobre Segmentação e ICP, que são as técnicas principais utilizadas neste trabalho.

A segmentação é utilizada para obter regiões que serão comparadas entre imagens de pontos de vista distintos. Se tais regiões tiverem a informação de profundidade podemos utilizar o método do ICP como o método de comparação, fazendo assim a sua correspondência. Para obter a profundidade da imagem, foi utilizado o sensor Kinect, o qual gera uma imagem RGBD. Sua calibração é descrita a seguir.

3.1 Calibração no Kinect

O Kinect possui diferentes sensores; no caso da captura da profundidade da imagem este utiliza um padrão de projeção emitido por um feixe de laser no qual é capturado por um sensor *infra-red* que mede o quanto de desvio um ponto da cena sofreu segundo este padrão.

Mesmo sendo menos preciso que um sensor a laser como o *Laser Imaging, Detection and Ranging* (LIDAR), para invariância em certo limite de profundidade e ambiente fechado o Kinect consegue um mapa de profundidade consistente.

Como as câmeras RGB e *infra-red* estão em posições distintas, é preciso obter as suas calibrações para corrigir o efeito paralaxe de ambas registrando-as em pontos correspondentes. Para isso (HERRERA et al., 2012) em seu trabalho produziu um algoritmo de calibração para o Kinect , que faz uso do plano homográfico obter a calibração. Este algoritmo foi utilizado para a calibração das imagens RGBD dessa dissertação.

3.2 Segmentação de Imagens

Um dos primeiros métodos utilizados na extração de regiões de uma cena através de suas características é o método da segmentação. A segmentação é uma operação monádica, ou seja, aplica-se uma função à imagem de entrada e tem-se como resultado uma imagem de saída com mesma dimensão. Em termos matemáticos temos: $\mathbf{O}[u, v] = f(\mathbf{I}[u, v]), \forall (u, v) \in \mathbf{I}$, onde (u, v) são as coordenadas dos pixels da imagem $\mathbf{I}$. Neste processo de análise, os pixels da imagem serão classificados e agrupados formando assim regiões de interesse (CORKE, 2011).

3.2.1 Segmentação Simples

O processo de segmentação de imagem é constituído de três subproblemas, o primeiro é a *classificação,* que é um processo de decisão aplicado a cada pixel e atribuído a uma determinada região R de um conjunto inteiro $r \in \{0, 1, ..., R-1\}$, onde R é o número de regiões. Comumente se usa $R = 2$ conhecido como classificação binária. Como exemplo de um caso simples, pode-se classificar uma imagem formada por duas regiões (classificação binária $\{0, 1\}$) quando $r = 0$, o pixel é classificado como sendo de uma região da imagem da frente que se destaca da imagem de fundo; $r = 1$, o pixel é classificado como sendo de uma região da imagem de fundo.

A regra de classificação binária dos pixels pode ser obtida por:

$$\mathbf{r}(u,v) = \begin{cases} 0 & \mathbf{I}(u,v) < T \\ 1 & \mathbf{I}(u,v) \geq T \end{cases} \forall (u,v) \in \mathbf{I}, \tag{13}$$

onde a classificação do pixel é baseada na referência de corte T (comumente chamada de *limiar*), e o domínio de $I(u,v)$ é $\mathbf{I} = \{I(u,v) \in \mathbb{R} | 0 \leq I(u,v) \leq 1\}$.

Um exemplo desse resultado pode ser visto na figura 11.

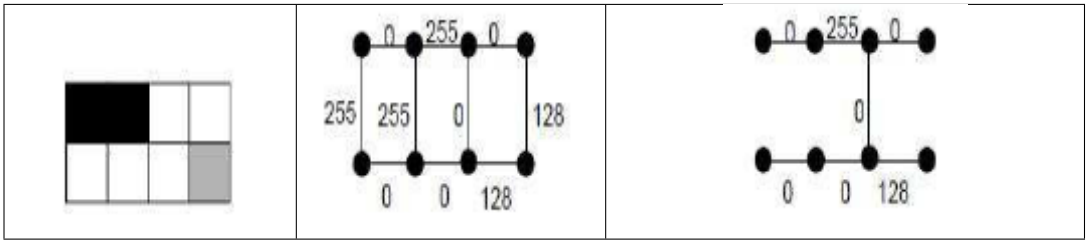
Figura 11 – Exemplo de segmentação binária onde o limiar é de $T \geq 0.6$.
Fonte: Produzido pelo próprio autor.

A seguir, o método de segmentação por grafo é abordado.

3.2.2 Segmentação Baseada em Grafo

Um grafo consiste de um conjunto de vértices, denotados por $\mathbf{V}$, ligados por um conjunto de arestas, denotadas por $\mathbf{E}$, onde:

$$\mathbf{G} = (\mathbf{V}, \mathbf{E})$$



(a) Imagem inicial.	(b) Grafo associado.	(c) Árvore de cobertura mínima.

Figura 12 – Exemplo de árvore de cobertura mínima.
Fonte: Autor "adaptado de" (ZIRARI et al., 2012)

$$\mathbf{V} = \{v_1, v_2, ..., v_n\}$$

$$\mathbf{E} = \{(v_i, v_j) | v_i \in \mathbf{E}, v_j \in \mathbf{E}\}$$

Onde temos as seguintes definições:

a) Um grafo conectado é um grafo onde para quaisquer dois vértices, i e j pode-se achar um caminho que começa em i e termina em j;

b) Um grafo não direcionado é um grafo em que as arestas não possuem orientação. A aresta (i, j) é idêntica a aresta (j, i), i.e., elas não são pares ordenados;

c) Uma árvore é um grafo conectado não cíclico, o qual conecta um subconjunto de todos os vértices. Uma árvore de cobertura é uma árvore que conecta todos os nós;

d) Uma árvore de cobertura mínima ou árvore de amplitude mínima ou árvore geradora mínima, do inglês *minimum spanning tree* (MST) de um grafo com arestas ponderadas, é uma árvore de cobertura cujo peso (a soma dos pesos de cada aresta dessa árvore) não é maior do que o peso de qualquer outra árvore de cobertura.

Todos esses conceitos são ilustrados neste exemplo de um grafo simples modelado na imagem da Figura 12. Neste exemplo, os vértices são associados aos pixels da imagem e as arestas são ponderadas pela diferença das intensidades entre dois pixels.

3.2.2.1 Método de segmentação

O método de segmentação baseado em grafo utilizado nessa dissertação vem do trabalho de (FELZENSWALB; HUTTENLOCHER, 2004). Esta abordagem pode ser descrita da seguinte forma:

Seja $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ um grafo não direcionado com vértices $v_i \in \mathbf{V}$, o conjunto de elementos que deve ser segmentado, e arestas $(v_i, v_j) \in \mathbf{E}$ correspondendo a pares de

vértices vizinhos. Cada aresta $e = (v_i, v_j)$ tem um peso correspondente $w(e)$, que é uma medida não negativa da desigualdade entre elementos vizinhos v_i e v_j. No caso de uma imagem segmentada, os elementos em $\mathbf{V}$ são pixels e o peso de uma aresta é alguma medida de dissimilaridade entre os dois pixels ligados por essa aresta (por exemplo, a diferença em intensidade, cor, movimento, localização ou algum outro atributo local).

Uma segmentação $\mathbf{S}$ é uma partição de $\mathbf{V}$ em componentes de tal modo que cada um dos componentes (ou regiões) $\mathbf{C} \in \mathbf{S}$ corresponde a um componente conectado em um grafo $\mathbf{G}' = (\mathbf{V}, \mathbf{E}')$, em que $\mathbf{E}' \subseteq \mathbf{E}$. Em outras palavras, qualquer segmentação é induzida por um subconjunto de arestas em $\mathbf{E}$.

Existem diferentes maneiras de medir a qualidade de uma segmentação, mas em geral queremos que os elementos de um componente se agrupem em semelhança. Isto significa que as arestas entre dois vértices no mesmo componente devem ter pesos relativamente baixos, e arestas entre os vértices em componentes diferentes devem ter pesos mais elevados.

3.2.2.2 Comparação em pares de regiões segundo um critério

Definindo um critério D para avaliar se há ou não evidências de uma fronteira entre dois componentes em uma segmentação (duas regiões de uma imagem), ou seja, se existir uma fronteira pode-se determinar a que componente um vértice pertence.

O critério D é um predicado (conforme a equação (16)) que é baseado na medição da dissimilaridade entre elementos ao longo da fronteira dos dois componentes em relação a uma medida de dissimilaridade entre elementos vizinhos, em cada um dos dois componentes. O critério resultante compara as diferenças entre componentes de modo adaptativo para dentro de cada componente com respeito às características locais dos dados.

Desse modo é definida a diferença interna de um componente $\mathbf{C} \subseteq \mathbf{S} \subseteq \mathbf{V}$ como sendo o maior peso na *árvore de cobertura mínima* do componente, $MST(\mathbf{C}, \mathbf{E})$. Isto é,

$$Int(\mathbf{C}) = \max_{e \in MST(\mathbf{C},\mathbf{E})} w(e). \tag{14}$$

Uma intuição subjacente a esta medida é que um determinado componente $\mathbf{C}$ só permanece conectado quando, pelo menos, o peso da aresta da fronteira de $Int(\mathbf{C})$ é considerado.

A diferença entre dois componentes $\mathbf{C}_1, \mathbf{C}_2 \subseteq \mathbf{V}$ é definida como sendo o mínimo de peso de uma fronteira que liga esses dois componentes, em que $e = (v_i, v_j)$. Isto é,

$$Dif(\mathbf{C}_1, \mathbf{C}_2) = \min_{v_i \in \mathbf{C}_1, v_j \in \mathbf{C}_2, (v_i, v_j) \in E} w(e), \tag{15}$$

Se não houver uma aresta de ligação $\mathbf{C}_1$ e $\mathbf{C_2}$ então $Dif(\mathbf{C}_1, \mathbf{C}_2) = \infty$. Esta medida de diferença poderia, em princípio, ser problemática, pois reflete apenas o menor peso da

fronteira entre dois componentes. Na prática verificou-se que a medida funciona muito bem, apesar dessa limitação aparente. Além disso, alterando a definição de utilizar a mediana do peso, ou outra quantidade, a fim de torná-la mais robusta, faz com que o problema de encontrar uma boa segmentação se torne NP-hard. Assim, uma pequena mudança para o critério de segmentação muda consideravelmente a complexidade do problema (FELZENSWALB; HUTTENLOCHER, 2004).

O predicado D compara regiões e avalia se há evidências de uma fronteira entre um par de componentes, verificando se a diferença entre os componentes, $Dif(\mathbf{C}_1, \mathbf{C}_2)$, é grande em relação à diferença interna dentro de, pelo menos, um dos componentes, $Int(\mathbf{C}_1)$ e $Int(\mathbf{C}_2)$. Uma função de corte é usada para controlar o grau no qual a diferença entre os componentes deva ser maior do que o mínimo interno desta diferença. Definindo o predicado de comparação aos pares como:

$$D(C1, C2) = \begin{cases} true & if\ Dif(C_1, C_2) > MInt(C_1, C_2) \\ false & caso\, contrario \end{cases}, \tag{16}$$

onde a diferença mínima interna, $MInt$, é definida como,

$$MInt(C_1, C_2) = min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)). \tag{17}$$

A função de corte τ controla o grau ao qual a diferença entre dois componentes deve ser maior do que suas diferenças internas, a fim de que haja evidência de uma fronteira entre eles (ou seja, D deve ser verdadeiro). Para pequenos componentes, $Int(C)$ não é uma boa estimativa das características dos dados locais. No caso extremo, quando $|C| = 1$, $Int(C = 0)$ tem-se um exemplo de haver nenhuma característica para o dado local. Por conseguinte, uma função de corte com base no tamanho do componente,

$$\tau(C) = {}^{k}/_{|C|}, \tag{18}$$

onde $|C|$ denota o tamanho de C, e k é um parâmetro constante. Isto é, para pequenos componentes é exigida uma evidência mais forte para um limite de corte. Na verdade, k define uma escala de observação, em que um k maior provoca uma preferência de componentes maiores, definindo-se assim o tamanho da segmentação. No entanto, nota-se que k não é um componente de tamanho mínimo. Componentes pequenos são aceitos quando existe uma diferença suficientemente grande entre componentes vizinhos.

Qualquer função não negativa para um único componente pode ser usada para o τ sem alterar o resultado do algoritmo. Por exemplo, é possível se ter um método de segmentação que prefira certo tipo de formas, se for definido um τ que seja grande para componentes que não se encaixam em alguma forma e pequenos para outros que se

Algoritmo 2 Algoritmo de Segmentação (FELZENSWALB; HUTTENLOCHER, 2004)

A entrada do algoritmo é um grafo $G = (V, E)$ que possui n vértices e m arestas. A saída é a segmentação de V em componentes $S = (C_1, ..., C_r)$.

a) Ler imagem a ser segmentada.

b) Aplica para cada canal de cor rgb uma convolução Gaussiana para suavização da imagem.

c) Construção do grafo com as arestas e seus respectivos pesos, sendo que cada peso da aresta é a diferença das intensidades entre dois pixels vizinhos.

d) Segmentação da imagem seguindo os passos descrito de a) até e).

a) Ordene E em $\pi = (o_1, ..., o_m)$, por um peso ascendente.

b) Comece com um segmento S^0, onde cada vértice v_i esta no seu próprio componente.

c) Faça o próximo item variando $q = 1, ..., m$.

d) Dado S^{q-1} construa S^q como a seguir. Faça v_i e v_j como representantes dos vértices conectados pela $q - \acute{e}xima$ aresta ordenada, como exemplo, $o_q = (v_i, v_j)$. Se v_i e v_j são componentes disjuntos de S^{q-1} e o peso $w(o_q)$ é pequeno comparado com a diferença interna de ambos os componentes, então combine os dois componentes ou então ignore. Mais formalmente, Faça C_i^{q-1} ser o componente de S^{q-1} contendo v_i e C_j^{q-1} o componente contendo v_j. Se $C_i^{q-1} \neq C_j^{q-1}$ e $w(o_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ então S^q é obtido de S^{q-1} por combinar C_i^{q-1} com C_j^{q-1}. De outra forma faça $S^q = S^{q-1}$.

e) Retorne $S = S^m$.

encaixam. Isto poderá acarretar em uma segmentação na qual o algoritmo converge os componentes em certas formas que não foram desejadas.

3.2.2.3 Algoritmo de Segmentação Baseado em Grafo e suas propriedades

Nesta seção é descrito o algoritmo de segmentação utilizando o critério de decisão D introduzido acima. Mostra-se que a segmentação produzida por este algoritmo obedece às propriedades de ser nem muito fino (detalhado) nem demasiadamente grosseiro.

O algoritmo de segmentação a seguir é bem parecido com o algoritmo de Kruskal (CORMEN et al., 1990), para a construção de uma árvore de cobertura mínima de um grafo. Ele pode ser implementado para funcionar em um tempo de $O(mlogm)$, em que m é o número de arestas no grafo.

Descrição do algoritmo 3.1:

O algoritmo lê a imagem a ser segmentada, suaviza essa imagem com uma Gaussiana e constrói o grafo com as arestas e seus respectivos pesos. Será utilizado o exemplo da Figura 12 para uma melhor compreensão.

No exemplo da Figura 12, temos o conjunto das arestas $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}\}$, onde $e_q = (v_i, v_j, w)$ é uma aresta com peso w. Tem-se neste caso que $E = \{(v_1, v_2, 0), (v_2, v_3, 255), (v_3, v_4, 0), (v_4, v_5, 0), (v_1, v_6, 255), (v_2, v_7, 255), (v_3, v_8, 0), (v_4, v_9, 0), (v_5, v_{10}, 128), (v_6, v_7, 0), (v_7, v_8, 0), (v_8, v_9, 0), (v_9, v_{10}, 128)\}$. A seguir, descreveremos a sequência da segmentação. O passo 1 do algoritmo 3.1 pede que se ordene E, segundo o peso w em ordem ascendente, desta forma $\pi = \{e_1, e_3, e_4, e_7, e_8, e_{10}, e_{11}, e_{12}, e_9, e_{13}, e_2, e_5, e_6\}$, ou seja, $E = \{(v_1, v_2, 0), (v_3, v_4, 0), (v_4, v_5, 0), (v_3, v_8, 0), (v_4, v_9, 0), (v_6, v_7, 0), (v_7, v_8, 0), (v_8, v_9, 0), (v_5, v_{10}, 128), (v_9, v_{10}, 128), (v_2, v_3, 255), (v_1, v_6, 255), (v_2, v_7, 255)\}$. O passo 2 pede para começar com um segmento S^0, onde cada vértice v_i está no seu próprio componente, ou seja, $S^0 = \{v_1, v_2, v_3, v_4, v_1, v_5, v_6, v_7, v_8, v_9, v_{10}\}$. O passo 3 pede que se faça o próximo passo variando $q = 1, ..., m$. Faremos então essa simulação começando com $q = 1$, portanto S^{q-1} com $q = 1$ fica com o valor S^0, em seguida ele pede para construir S^q, ou seja, construa S^1 com a $q - \acute{e}xima$ aresta ordenada, neste caso $o_1 = e_1 = (v_1, v_2, 0)$ e como v_1 e v_2 são componentes disjuntos de S^0 e o peso $w(e_1) = 0$ é pequeno comparado com a diferença interna dos componentes, combinamos v_1 com v_2 da aresta e_1; e como $S^q = S^1$ obtido de $S^{q-1} = S^0$, tem-se $S^1 = \{\{v_1, v_2\}, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$. Continuando a simulação com $q = 2$, construa S^2 com a $q - \acute{e}xima$ aresta ordenada, neste caso $o_2 = e_3 = (v_3, v_4, 0)$ e como v_3 e v_4 são componentes disjuntos de S^1 e o peso $w(e_3) = 0$ é pequeno comparado com a diferença interna dos componentes, combinamos v_3 com v_4 da aresta e_3; e como S^2 é obtido de S^1, tem-se $S^2 = \{\{v_1, v_2\}, \{v_3, v_4\}, v_5, v_6, v_7, v_8, v_9, v_{10}\}$. Substituindo para $q = 3$, construa S^3 com a $q - \acute{e}xima$ aresta ordenada, neste caso $o_3 = e_4 = (v_4, v_5, 0)$ e como v_4 e v_5 são componentes disjuntos de S^2 e o peso $w(e_4) = 0$ é pequeno comparado com a diferença interna dos componentes, combinamos v_4 com v_5 da aresta e_4; em que S^3 é obtido de S^2, tem-se $S^3 = \{\{v_1, v_2\}, \{v_3, v_4, v_5\}, v_6, v_7, v_8, v_9, v_{10}\}$. Continuando, tem-se $q = 4$, construa S^4 com a $q - \acute{e}xima$ aresta ordenada, neste caso $o_4 = e_7 = (v_3, v_8, 0)$, e como v_3 e v_8 são componentes disjuntos de S^3 e o peso $w(e_7) = 0$ é pequeno comparado com a diferença interna dos componentes, combinamos v_3 com v_8 da aresta e_7; em que S^4 é obtido de S^3, e desta forma tem-se $S^4 = \{\{v_1, v_2\}, \{v_3, v_4, v_5, v_8\}, v_6, v_7, v_9, v_{10}\}$. A simulação continua com $q = 5$, $q = 6$ e $q = 7$. Com $q = 8$, construa S^8 com a $q - \acute{e}xima$ aresta ordenada, neste caso $o_8 = e_{12} = (v_8, v_9, 0)$ e como v_8 e v_9 não são componentes disjuntos de S^7, temos outra saída onde $S^q = S^{q-1}$, ou seja, $S^8 = S^7 = \{\{v_1, v_2\}, \{v_3, v_4, v_5, v_6, v_7, v_8, v_9\}, v_{10}\}$. Continuando com $q = 9$, construa S^9 com a $q - \acute{e}xima$ aresta ordenada, neste caso $o_9 = e_9 = (v_5, v_{10}, 128)$ e v_5 e v_{10} são componentes disjuntos de S^8, porém o peso $w(e_9) = 128$ supera o peso da diferença interna dos componentes, portanto temos a mesma saída que $q = 8$, ou seja, $S^9 = S^8 = S^7 = \{\{v_1, v_2\}, \{v_3, v_4, v_5, v_6, v_7, v_8, v_9\}, v_{10}\}$. A simulação continua para $q = 10$, $q = 11$ e $q = 12$. E por último, tem-se a simulação com $q = 13$, construa S^{13} com a $q - \acute{e}xima$ aresta ordenada, neste caso $o_{13} = e_6 = (v_2, v_7, 255)$ e v_2 e v_7 são componentes disjuntos de S^{12}, o peso $w(e_6) = 255$ supera o peso da diferença interna dos componentes, portanto temos a mesma saída que $q = 12$, ou seja, $S^{13} = S^{12} = S^{11} = S^{10} = S^9 = S^8 =$

$S^7 = \{\{v_1, v_2\}, \{v_3, v_4, v_5, v_6, v_7, v_8, v_9\}, v_{10}\}$. O quinto e último passo é à saída do sistema onde $S = S^m = S^{13} = \{\{v_1, v_2\}, \{v_3, v_4, v_5, v_6, v_7, v_8, v_9\}, v_{10}\}$. Perceba que S possui três componentes internos, formando assim três regiões na figura 12 que são as três regiões visíveis da figura.

A seguir, veremos o funcionamento do algoritmo Iterative Closest Point (ICP) e de que forma esse pode ser usado para obtermos a correspondência entre regiões em diferentes imagens.

3.3 Introdução ao Iterative Closest Point (ICP)

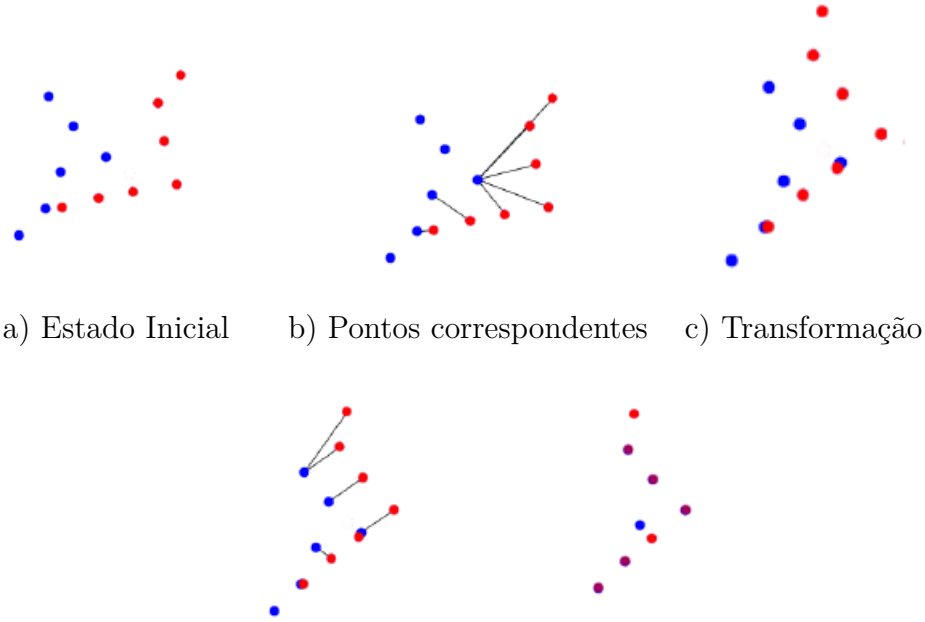
O algoritmo Iterative Closest Point (ICP) foi desenvolvido por (BESL; MCKAY, 1992), é um algoritmo desenvolvido para registrar duas nuvens de pontos de formas tridimensionais ou bidimensionais num sistema de igual coordenada. O algoritmo ICP calcula a orientação de duas nuvens de pontos registrando-as.

Nesta dissertação o objetivo não é um escaneamento em 3D e sim utilizar o ICP para obter a correspondência das nuvens de pontos e desta forma associar as regiões que delimitaram estas nuvens a partir de pares de imagens. Quando o ICP compara as nuvens no final do processo, este determina a pose final entre as duas nuvens e também produz um erro que irá servir como valor para se determinar a correspondência das nuvens (regiões entre as imagens) neste trabalho.

O algoritmo ICP funciona da seguinte forma: dois conjuntos de pontos são entradas para o algoritmo, sendo um considerado como "modelo" de comparação enquanto o outro conjunto é considerado como "dados" para comparação. É feita uma estimativa inicial da transformação (rotação, translação e escala) entre o "modelo" de pontos e o "dados" de pontos (inicialização do algoritmo). Todas as transformações que ocorrerão no processo do ICP têm um valor de posicionamento inicial (por exemplo, ele pode ser inicializado igualando os centroides de ambas às nuvens) e valores que limitam o processo de alinhamento das nuvens de pontos que são: o número máximo de iterações, o erro máximo permitido e o estabelecimento de um limite no caso de não convergência.

Um exemplo de convergência para obtenção da pose entre as nuvens é mostrado na figura 13. Na figura 13a tem-se o estado inicial das duas nuvens, que é representado pelo conjunto de pontos em azul como "modelo" e um conjunto de pontos em vermelho como "dados". Na próxima etapa, o ponto mais próximo do modelo (azul) é usado para verificar a distância para cada ponto do conjunto "dados", como mostrado na figura 13b. Em seguida, as novas matrizes de transformação são calculadas de acordo com o erro minimizado da Equação (25) e aplicado ao conjunto "dados". Depois disso, o erro atual é comparado ao erro anterior e se este for maior, a iteração recomeça. Inicializando a estimativa de transformação (rotação, escala e translação) como mostrada na figura 13c.

Este processo interativo se repete até que o erro seja pequeno o suficiente, como na figura 13e ou que a iteração máxima seja alcançada.



a) Estado Inicial b) Pontos correspondentes c) Transformação

d) Nova Correspondência e) Estado final

Figura 13 – Fases do ICP básico .
Fonte: Autor "adaptado de" (OSWALD, 2010).

3.3.1 Busca pelo ponto de vizinhança mais próximo

O ICP requer basicamente a computação de vizinhos mais próximos. Dado um conjunto de n pontos pertencentes à nuvem modelo M, onde $M = \{m_1, m_2, ..., m_n\}$, e um ponto d, o problema consiste em encontrar o ponto em M que está mais próximo de d. A ideia básica é calcular as distâncias entre cada ponto de M e d e escolher aquele com a menor distância. Para uma melhor eficiência, uma possibilidade é pré-computar regiões no espaço que estejam mais próximas de um determinado ponto em M, reduzindo buscas subsequentes. Esta é a abordagem seguida pela triangulação de Delaunay (WATSON, 1981, BOWYER, 1981). Outro método consiste em dividir os pontos de M em grupos de acordo com sua localização. A busca do vizinho mais próximo é acelerada pela exclusão condicional de muitos pontos, reduzindo assim o número de cálculos de distância. Um desses métodos é a *árvore k-d* (BENTLEY, 1975).

3.3.2 Etapas do ICP

Desde a introdução do ICP (CHEN; MEDIONI, 1991, BESL; MCKAY, 1992), muitas variantes foram introduzidas. O ICP utilizado nesta dissertação vem da tese proposta por (KJER et al., 2010) que possui as seguintes etapas:

a) Seleção de um conjunto de pontos em uma ou ambas as nuvens.

b) Correspondência, harmonizando os pontos de amostras na outra nuvem.

c) Ponderação dos pares correspondentes de forma adequada.

d) Rejeitar certos pares (outliers) (SIMON, 1996).

e) Atribuição de uma métrica de erro com base nos pares de pontos.

f) Minimização do erro (RUSINKIEWICZ; Levoy, 2001).

3.3.2.1 Etapa 1 - Seleção

Antes de se aplicar o algoritmo ICP, pode ser vantajoso considerar apenas alguns pontos tanto da nuvem "dados" quanto da nuvem "modelo". Por exemplo, valores atípicos podem ser eliminados com base em algum limite; para minimizar a complexidade computacional, a quantidade de pontos pode ser reduzida por uma sub amostragem aleatória ou uniforme. Isso irá acelerar os cálculos, principalmente no passo da correspondência.

A ideia por trás da amostragem aleatória é modificar a amostra a cada iteração do algoritmo (MASUDA et al., 1996), a fim de evitar qualquer tendência de se ter valores atípicos.

Regiões planas nas nuvens de pontos podem conter informações redundantes das quais algumas podem ser descartadas. Se os objetos subjacentes como: cor, curvatura ou tangentes normais estão disponíveis, uma estratégia é fazer com que essas características possam ser representadas em alguma distribuição predefinida. Assim, a referida estratégia pode ser usada para aumentar a densidade de amostragem em regiões ricas nesses recursos. Isso pode não só reduzir a complexidade, mas também aumentar a convergência.

3.3.2.2 Etapa 2 - Correspondência

Correspondência serve como medida do emparelhamento de pontos a partir da nuvem de pontos de dados para a nuvem de pontos modelo. Encontrar os vizinhos mais próximos é geralmente o passo computacional mais intensivo no algoritmo ICP. O método de força bruta simplesmente calcula distâncias para todos os candidatos vizinhos e escolhe o mais próximo.

Técnicas podem ser empregadas para acelerar a busca pelo vizinho mais próximo. Estes incluem árvores k-d e triangulações Delaunay, ambos fornecem um grande aprimoramento no tempo de pesquisa, em detrimento de algum pré-processamento adicional. Na maioria das iterações do algoritmo ICP, os dados e os modelos de nuvens de pontos estão bastante próximos uns dos outros. Portanto, há um favorecimento ao se utilizar a árvore k-d, acelerando assim o passo da correspondência.

3.3.2.3 Etapa 3 - Ponderação

Cada par de pontos correspondido pode ser ponderado conforme a característica local. Na prática isso significa multiplicar sempre um termo da métrica de erros com um fator específico w. O fator pode ser baseado na distância, cor, curvatura ou direções normais tangentes.

O peso de um par de pontos pode ser calculado utilizando as normais $\vec{n_p}$ do ponto p da nuvem "modelo" e $\vec{n_q}$ do ponto q da nuvem "dados", tal como foi sugerido no artigo de (RUSINKIEWICZ; Levoy, 2001) usando o peso

$$w = \vec{n_p}.\vec{n_q}. \tag{19}$$

No trabalho de (KJER et al., 2010) é proposto um peso Gaussiano para pontos com valores de curvatura c_p e c_q que estejam num intervalo de $[0; 1]$ como sendo:

$$w = e^{-(c_p - c_q)^2}. \tag{20}$$

.Para ponderar o par (p, q) de acordo com a distância, (GODIN et al., 1994) estabelece o peso como sendo:

$$w = 1 - \frac{dist(p, q)}{dist_{max}}, \tag{21}$$

na qual $dist_{max}$ é a distância máxima entre todos os pares de pontos.

3.3.2.4 Etapa 4 - Rejeição

O objetivo desta etapa é rejeitar uma dada porcentagem dos pares de pontos que possuem as piores distâncias Euclidianas entre eles.

Pares de pontos podem ser rejeitados após cada passo de correspondência. Isto pode ser feito por uma avaliação estatística das distâncias da vizinhança.

3.3.2.5 Etapa 5 - Atribuição da métrica de erro

A métrica de erro define uma função que tem como objetivo a minimização do erro a cada iteração. Duas métricas são comumente usadas:

Minimização *ponto a ponto (BESL; MCKAY, 1992),* é a soma dos quadrados das distancias dos pontos "dados" com os pontos "modelo" e que pode ser expressa como:

$$E = \sum_{i=1}^{N} ||Rp_i + \vec{T} - q_i||^2. \tag{22}$$

Minimização do *ponto ao plano* é a soma das distâncias dos pontos "dados" com os planos tangentes no qual o modelo correspondente reside. Sua formulação matemática é:

$$E = \sum_{i=1}^{N} [(Rp_i + \vec{T} - q_i).\vec{n_i}]^2, \tag{23}$$

onde $\vec{n_i}$ denota a normal da tangente estimada no i -ésimo ponto do "modelo". Isto faz com que planos se correspondam na minimização o que pode ser uma vantagem.

A cada iteração do ICP, a transformação (rotação, escala e translação) dos pontos que pertencem a uma determinada região pode ser calculada por diferentes métodos.

3.3.2.6 Etapa 6 - Minimização

A solução existente para a minimização da métrica de erro ponto a ponto desenvolvida no trabalho de (ARUN et al., 1987) pode ser encontrada no anexo C. Esta minimização é baseada no valor singular decomposto (SVD) (anexo E). Para a formulação dos quaternios equivalentes existe uma solução na forma fechada (FAUGERAS; HEBERT, 1986, HORN, 1987). A métrica de erro do ponto ao plano tem apenas uma solução de forma fechada após linearização da matriz de rotação R, que é mostrada no anexo D. A minimização foi proposta e derivada por (CHEN; MEDIONI, 1991). Para outras métricas, em geral, não se espera a existência de uma solução na forma fechada. Nesse caso, podem ser utilizados métodos não lineares. Usando métodos não lineares tem que se ter um equilíbrio entre uma precisão aceitável e o custo do cálculo, porém, tem-se a liberdade para modelar a função proposta.

3.3.3 Algoritmo Básico do ICP

Considere dois conjuntos de pontos $M, D \subseteq R^d$ onde $|M| = n$ e $|D| = b$. O interesse é ter uma função de correspondência um para um $\mu : M \to D$ que minimize a raiz quadrada da distância média ($RMSD$) entre M e D . Matematicamente, queremos minimizar o seguinte:

$$RMSD(M, D, \mu) = \sqrt{\frac{1}{n} \sum_{m \in M} ||m - \mu(m)||^2}. \tag{24}$$

Incorporando a rotação e translação na correspondência, queremos achar

$$\min_{\mu: M \to D, t \in R^d, R \in SO(d)} \sum_{m \in M} ||Rm - t - \mu(m)||^2, \tag{25}$$

onde R é a matriz de rotação, t é o vetor de translação e $SO(d)$ é o conjunto especial de matrizes ortogonais na dimensão d.

Descrição do algoritmo 3.2:

Algoritmo 3 Algoritmo Básico do ICP.

$ICP(M, D)$

a) Inicialização $R = I = (Matriz\ Identidade), t = 0$.

b) Correspondência: Dado R e t , calcular a otimização μ , procurando minimizar μ em $RMSD(M, D, \mu)$.

c) Transformação: Dado μ , calcular a otimização de R e t , procurando minimizar (R, t) em $RMSD(RM - t, D, \mu)$.

d) Volte à etapa 2 se μ mudar.

O algoritmo inicia a matriz de rotação com o valor da matriz identidade, em seguida, procura minimizar o $RMSD$ alternando as etapas de correspondência e transformação. Na etapa de correspondência, dada uma rotação R e uma translação t, a otimização da correspondência entre as nuvens M e D é calculada de forma a minimizar o $RMSD$ conforme a Equação (24), minimizando o erro μ. Na etapa de transformação, dada uma correspondência otimizada μ, calcula-se a rotação R e translação t de forma a minimizar (R, t). Esse processo iterativo termina quando a correspondência μ fica inalterada nas iterações sucessivas.

A seguir é descrito o método utilizado para a correspondência de regiões entre imagens desenvolvido neste trabalho.

4 MÉTODO DA CORRESPONDÊNCIA DE REGIÕES ENTRE IMAGENS POR MEIO DO ICP

Como mencionado na introdução desta dissertação, o principal objetivo deste trabalho é o estudo e o desenvolvimento de um algoritmo capaz de identificar regiões correspondentes entre as imagens obtidas a partir de pontos de vista distintos de uma mesma cena.

Este método propõe uma nova aplicação para o ICP que é a comparação de regiões entre imagens, para isso a função que retorna o erro entre as nuvens das regiões comparadas irá servir como valor de comparação entre essas regiões, sendo que quanto menor for o erro entre as nuvens das regiões, maior a correspondência entre elas.

4.1 Fluxograma do Método proposto

O método proposto segue as etapas descritas abaixo (as etapas estão esquematizadas na figura 14 e desenvolvidas nos Algoritmos 4 e 5):

Etapa 1. Leitura da Cena

Etapa 2. Normalização

Etapa 3. Segmentação da Cena

Etapa 4. Seleção das Regiões no Mapa de Disparidade

Etapa 5. ICP

Etapa 6. Correspondência das Regiões de Menor Erro

A figura 14 representa o método desenvolvido. Nesta figura tem-se como entrada (etapa 1) a captura das imagens e disparidades obtidas pelos sensores, logo em seguida tem-se o bloco da etapa 2 onde é feita a normalização entre imagem e disparidade. Em seguida (no bloco da etapa 3) é gerada a segmentação de cada imagem, que resulta em regiões a serem comparadas pela etapa 5. Essa segmentação obedece a alguns parâmetros como, por exemplo, o limiar de área das regiões resultantes. Após a etapa 3 tem-se o bloco da etapa 4 que selecionará a região com sua respectiva disparidade. Com essa disparidade forma-se a nuvem de pontos da região a ser comparada. Na etapa 5 as nuvens produzidas a partir de cada região obtida em pares de imagens serão comparadas pelo método do ICP, cada nuvem de um conjunto de m nuvens obtida pela segmentação da imagem, que foi capturada por um sensor é comparada com n nuvens obtida pela segmentação da imagem, que foi capturada pelo outro sensor, ou seja, é uma pesquisa de força bruta. Após comparamos uma nuvem de um sensor com n nuvens do outro sensor, a etapa 6 escolhe o menor erro de comparação revelando assim o par de nuvens correspondentes.

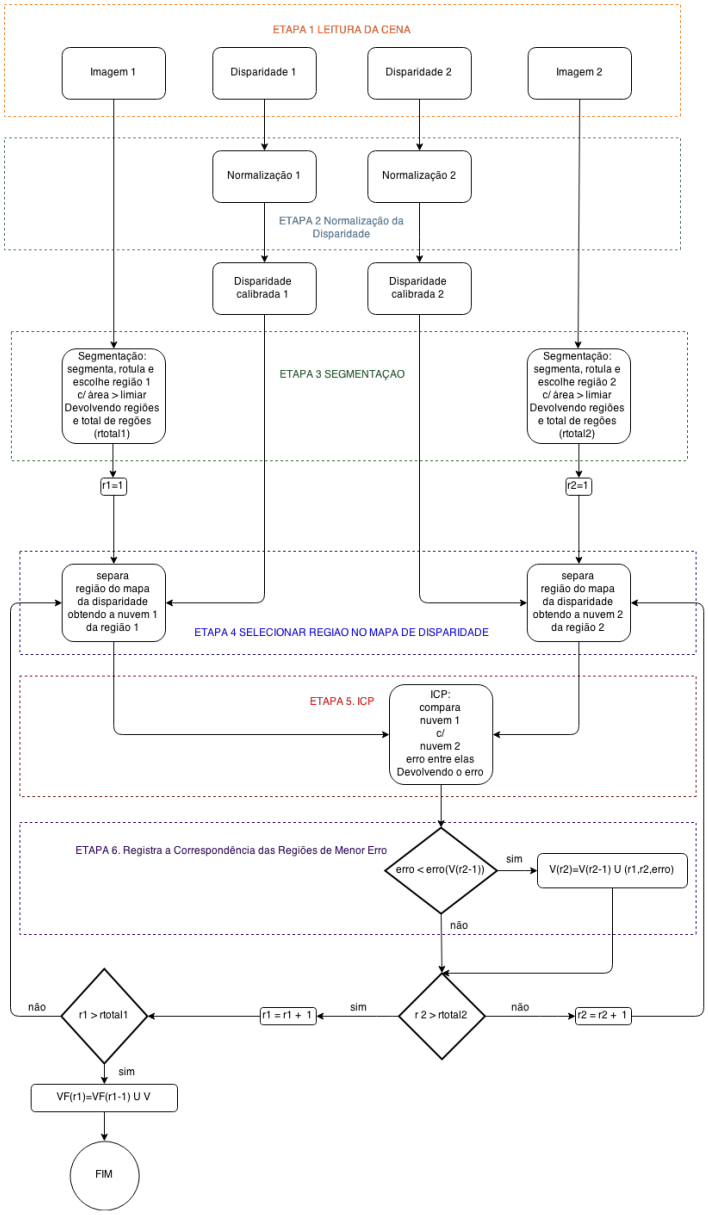


Figura 14 – Fluxograma do modelo de correspondência de áreas.
Fonte: Produzido pelo próprio autor.

4.2 Método Proposto

O método proposto neste trabalho é denominado: Método de Correspondência baseado no ICP (MCICP).

O pseudo código apresentado no algoritmo 4 representa o MCICP. As linhas 1,2,3,4 e 5 representam os parâmetros iniciais. As linhas 1,2 e 3 representam os parâmetros da função de segmentação. O parâmetro k da linha 1 serve como limiar da área a ser segmentada condicionando o resultado em regiões grosseiras ou finas, no nosso caso escolhemos um valor alto para k, pois estamos interessados em formar regiões grosseiras. O parâmetro min, da linha 2, limita o tamanho mínimo de área resultante da segmentação. Em seguida

na linha 3, o parâmetro σ é a Gaussiana utilizada na função de segmentação que suaviza a imagem a ser segmentada. O parâmetro max é a máxima iteração que o ICP pode executar na comparação da etapa 5. O parâmetro $MinAreaThreshold$ é o limiar da razão mínima entre as áreas a serem comparadas, ele determina se as regiões podem ser comparadas conforme a razão entre as suas áreas. Por último, na linha 6, temos a inicialização do vetor variável da região correspondente e regiões correspondentes, essas variáveis serão utilizadas na etapa 6 (Correspondência das regiões de menor erro).

A Etapa 1 (Leitura da Cena) é explicada em detalhes na subseção 4.3.1, essa etapa corresponde às linhas 7 e 8 que recebem as imagens e disparidades de cada sensor.

A Etapa 2 (Normalização) é explicada em detalhes na subseção 4.3.2, essa etapa corresponde às linhas 9 e 10 que fazem a correspondência dos pontos da imagem RGB como os pontos da disparidade de cada sensor.

A Etapa 3 (Segmentação da Cena) é explicada em detalhes na subseção 4.3.3, essa etapa corresponde às linhas 12 e 13. A linha 12 chama a função de segmentação para formar as regiões da imagem 1 e a linha 13 chama a função de segmentação para formar as regiões da imagem 2. A função de segmentação (segmentação em grafo) utiliza o algoritmo de (FELZENSWALB; HUTTENLOCHER, 2004) que segmenta a imagem devolvendo um vetor de regiões e o total de regiões formadas.

A Etapa 4 (Selecionar região no mapa de disparidade) é explicada em detalhes na subseção 4.3.4, essa etapa corresponde às linhas 14 a 29. As linhas 14 e 15 iniciam o laço externo onde as regiões da imagem 1 serão escolhidas para serem comparadas. A seguir, na linha 17 é separada a região da imagem 1 a ser comparada e na linha 19 é calcula a área 1 dessa região, que servirá para o cálculo da razão entre as áreas selecionadas, esse cálculo junto com o limiar $MinAreaThreshold$ irá selecionar se essas áreas devem ser comparadas. A linha 21 utiliza a região selecionada da imagem 1 para selecionar a nuvem 1 no mapa da disparidade 1. As linhas 22 e 23 iniciam o laço interno onde as regiões da imagem 2 serão escolhidas, da mesma forma que no laço externo (linhas 14 a 21), na linha 25 é separada a região da imagem 2 e a seguir na linha 27 é calculada a área 2 e também da mesma forma que na linha 21, a linha 29 seleciona a região da imagem 2 que irá produzir a nuvem 2 utilizando o mapa de disparidade 2.

Antes da Etapa 5 na linha 31, tem-se a avaliação das razões entre as áreas a serem comparadas, se a razão for superior ao limiar $MinAreaThreshold$ essas regiões (agora em forma de nuvens) serão comparadas.

Algoritmo 4 Pseudocódigo do Algoritmo de Correspondência

1 // k é um parâmetro escalar não sendo um valor mínimo, porem tem uma maior pendência na escolha de áreas maiores.
2 // min é o tamanho mínimo de área que uma região pode ter em pixels.
3 // σ é a Gaussiana interna que é utilizada para suavizar a imagem inicialmente.
4 // max é a máxima iteração do ICP.
5 // $MinAreaThreshold$ é a razão limiar entre as regiões comparadas.
6 $RegiãoCorrespondente(0) \leftarrow \oslash$,$RegioesCorrespondentes(0) \leftarrow \oslash$
7 **imagem1** $\leftarrow ler(imagem1)$,**imagem2** $\leftarrow ler(imagem2)$
8 **disparidade1** $\leftarrow ler(disparidade1)$,**disparidade2** $\leftarrow ler(disparidade2)$
9 $\mathbf{D}_1 \leftarrow Normalização(\mathbf{imagem1}, \mathbf{disparidade1})$
10 $\mathbf{D}_2 \leftarrow Normalização(\mathbf{imagem2}, \mathbf{disparidade2})$
11 // Cria as regiões (segmentação) das imagens 1 e 2 , devolvendo as regiões e o total delas.
12 $[\mathbf{R}_1, tamanhoR_1] \leftarrow igraphseg(\mathbf{imagem1}, k, min, \sigma)$
13 $[\mathbf{R}_2, tamanhoR_2] \leftarrow igraphseg(\mathbf{imagem2}, k, min, \sigma)$
14 $m \leftarrow 1$
15 **while** $m < tamanhoR_1$
16 // Separa a região da imagem 1 a comparar.
17 $\mathbf{R}_1 \leftarrow \begin{cases} R_1(u,v) \leftarrow true, & Se\ R_1(u,v) = m \\ R_1(u,v) \leftarrow false, & Caso\ contrario \end{cases}$
18 // Calcula a área da região da imagem 1.
19 $area1 \leftarrow area(\mathbf{R}_1)$
20 // Separa a região da disparidade 1 um no mapa de disparidade.
21 $\mathbf{P}_1 \leftarrow \mathbf{R}_1.\mathbf{D}_1$
22 $n \leftarrow 1$
23 **while** $n < tamanhoR_2$
24 // Separa a região da imagem 2 a comparar.
25 $\mathbf{R}_2 \leftarrow \begin{cases} R_2(u,v) \leftarrow true, & Se\ R_2(u,v) = n \\ R_2(u,v) \leftarrow false, & Caso\ contrario \end{cases}$
26 // Calcula a área da região da imagem 2.
27 $area2 \leftarrow area(\mathbf{R}_2)$
28 // Separa a região da imagem 1 um no mapa de disparidade.
29 $\mathbf{P}_2 \leftarrow \mathbf{R}_2.\mathbf{D}_2$
30 // Avalia se a razão entre as áreas das regiões e menor que $MinAreaThreshold$
31 **if** $area2/area1 > MinAreaThreshold \ || \ area1/area2 > MinAreaThreshold$ **then**
32 // ICP onde$\mathbf{P}_1$ é a nuvem "Modelo", $\mathbf{P}_2$ é a nuvem "Dados", usando Kdtree
33 $erro \leftarrow ICP(\mathbf{P}_1, \mathbf{P}_2, 'kDtree', max)$
34 // Verifica se a correspondência entre as regiões (m, n) possui um erro menor que a correspondência anterior
35 **if** $erro < Erro(RegioesCorrespondentes(n-1))$ **then**
36 $RegiaoCorrespondente(n) \leftarrow RegiaoCorrespondente(n-1) \cup (erro, m, n)$
37 **end if**
38 **end if**
39 $n \leftarrow n + 1$
40 **end while**
41 $RegioesCorrespondentes(m) \leftarrow RegioesCorrespondentes(m-1) \cup RegiaoCorrespondente(n)$
42 $m \leftarrow m + 1$
43 **end while**

A Etapa 5 (ICP) é explicada em detalhes na subseção 4.3.5, essa etapa corresponde à linha 33, onde as nuvens 1 e 2 serão submetidas à função ICP que retornará o erro entre ambas.

A Etapa 6 (Correspondência das regiões de menor erro) é explicada em detalhes na subseção 4.3.6, essa etapa corresponde às linhas 36 e 41. Na linha 36 o erro de comparação do par (m, n) é verificado com o erro do par anterior $(m, n-1)$ prevalecendo aquele com o menor erro o qual será adicionado ao vetor das regiões correspondentes na linha 41.

4.3 Descrições das Etapas do Método proposto

A seguir cada uma das etapas do método proposto é descrito em detalhes.

4.3.1 Etapa 1: Leitura da Cena

Esta etapa recebe pares de imagens RGB e dois mapas de disparidade correspondentes, representadas abaixo como imagem 1, imagem 2, disparidade 1 e disparidade 2. Estas são imagens e disparidades capturadas pelo Kinect em diferentes posições de uma mesma cena e se diferem em relação ao ponto de vista, ou seja, a imagem 1 e disparidade 1 correspondem à posição do sensor num determinado local e a imagem 2 e disparidade 2 em outro local observando os mesmos objetos de uma cena. Assim a posição do sensor onde foi obtida a imagem 1 e disparidade 1 em relação à posição do sensor onde foi obtida a imagem 2 e disparidade 2 está separada segundo uma mudança de rotação, escala e translação. Essas imagens são fornecidas automaticamente pelo Kinect, porém, como veremos na próxima etapa, é necessária a execução de um processo de normalização que depende da matriz de calibração da câmera, que representa a correspondência da imagem RGB com sua imagem de disparidade.

4.3.2 Etapa 2: Normalização

A etapa de normalização consiste em determinar os pontos da imagem RGB que correspondem aos pontos do mapa de disparidade, para que isso ocorra é preciso obter a calibração de ambos os sensores tanto a calibração da câmera RGB do Kinect quanto o sensor que forma o mapa de disparidade.

Pelo fato do Kinect não produzir a imagem da câmera RGB calibrada à imagem do sensor de disparidade, foi necessário fazer essa calibração para a obtenção da matriz de calibração.

O processo de calibração consiste em determinar quais os parâmetros intrínsecos e extrínsecos de um sensor referentes ao sistema de coordenadas global. Estas técnicas dependem de um conjunto conhecido de pontos tridimensionais que correspondem a um conjunto de pontos do plano da imagem.

O programa utilizado para executar a calibração do mapa de disparidade com a imagem RGB é o proposto no trabalho de (HERRERA et al., 2012).

Após obter a matriz de calibração de ambos os sensores foi possível calcular o conjunto de pontos do mapa de disparidade da imagem projetando de volta para o quadro de referência da câmera RGB em coordenada métrica. O algoritmo 5 descreve este processo.

Algoritmo 5 Algoritmo de Normalização da disparidade.

a) Ler mapa de disparidade do Kinect.

b) Tirar a distorção radial do mapa de disparidade.

c) Converter o valor da disparidade no mapa de disparidade do Kinect para o valor de profundidade, tornado assim um mapa de profundidade.

d) Normalizar os pontos do mapa de profundidade projetando o raio que passa pelo centro da câmera no plano da imagem da disparidade.

e) Projetar de volta ao plano de imagem RGB, transformando os pontos, ou seja, aplicar translação, escalonamento e rotação, utilizando a matriz de calibração do mapa de disparidade para esse fim.

O processo de normalização de disparidade é de grande importância para que as regiões que serão comparadas contenham as profundidades respectivas, ou seja, sem esta calibração a imagem RGB possuirá o erro de paralaxe em relação à imagem do mapa de profundidade afetando o resultado da etapa 5 que é a correspondência do ICP.

A figura 15 mostra a correspondência entre as duas imagens (RGB e mapa de disparidade) e o registro final da imagem RGB com o mapa de disparidade.

4.3.3 Etapa 3: Segmentação da Cena

A etapa da segmentação utiliza alguns parâmetros como já exposto na seção 3.2.2 (segmentação por grafo), são eles:

a) k: o parâmetro de escala;

b) min: a área da região mínima em pixels;

c) σ: o parâmetro que a Gaussiana utiliza inicialmente para a suavização da imagem.

A saída é uma imagem etiquetada formando as regiões como mostra a figura 16, onde r mostra o valor da etiqueta de cada região.

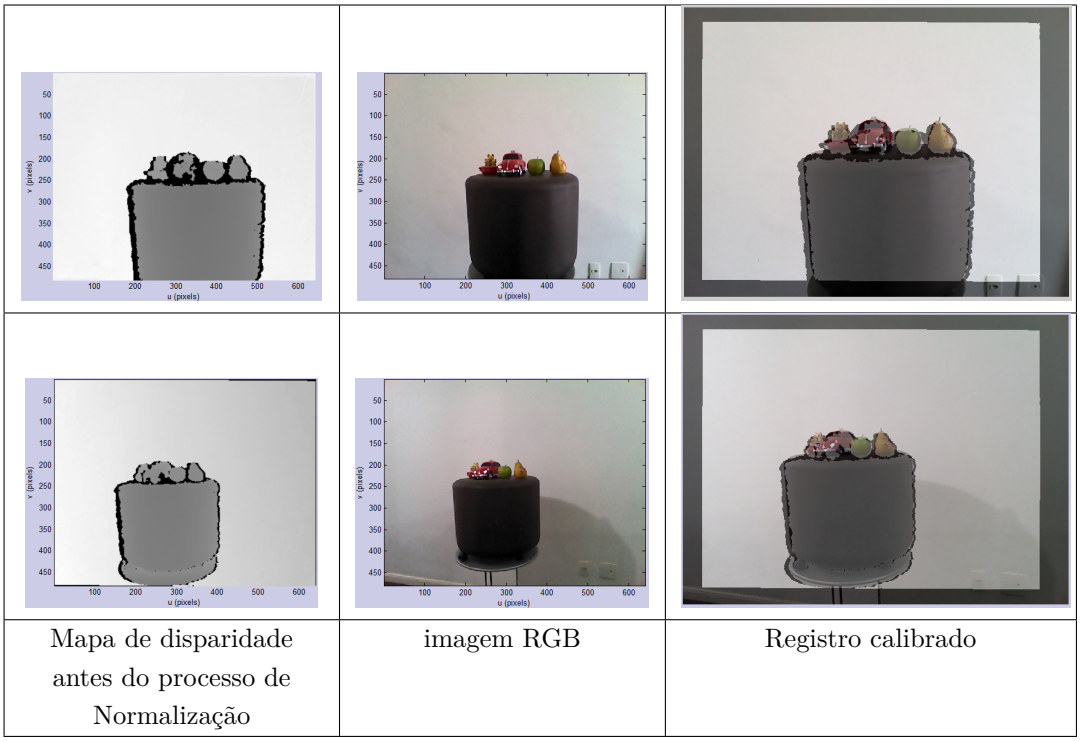


Figura 15 – Calibração da imagem RGB com o mapa de disparidade.
Fonte: Produzido pelo próprio autor.

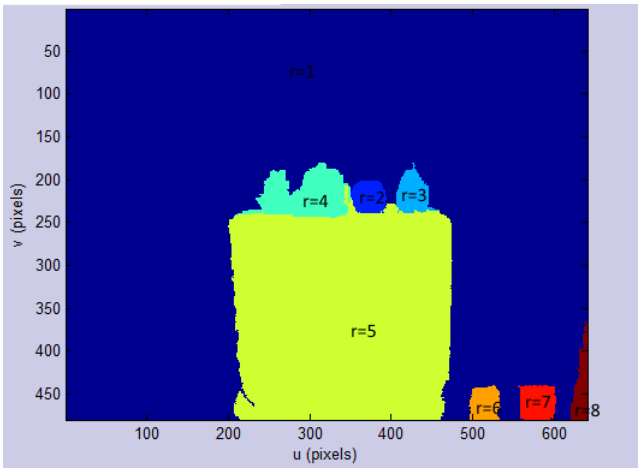


Figura 16 – Segmentação em grafo.
Fonte: Produzido pelo próprio autor.

4.3.4 Etapa 4: Seleção das Regiões no Mapa de Disparidade

Após a segmentação tem-se uma matriz $\mathbf{S_i}$ de coordenadas (u, v) onde i é a i -ésima imagem segmentada. A matriz $\mathbf{S_i}$ é formada por regiões R_k, ou seja, $\mathbf{S_i} = (R_1, R_2, ..., R_k)$, onde k é a k-ésima região da imagem segmentada, como também k é o valor da etiqueta da região, como mostra a figura 17(a) e 17(b). Para selecionar a região a ser comparada utilizamos a seguinte forma de seleção:

$$\mathbf{S}_{ik} = \begin{cases} S_i(u, v) = 1, & Se\ S_i(u, v) = k \\ S_i(u, v) = 0, & caso\ contrario \end{cases}, \tag{26}$$

onde, $S_i(u, v)$ recebe o valor 1 (true) se a coordenada (u, v) pertencer à região de etiqueta k e 0 (false) se esta coordenada (u, v) não pertencer à região de etiqueta k, separando assim a região da imagem $\mathbf{S}_i$ obtendo uma matriz binária $\mathbf{S}_{ik}$ de valores lógicos (0 e 1).

Podemos agora aplicar a matriz de valores binários $\mathbf{S}_{ik}$ multiplicando-a a matriz do mapa de profundidade $\mathbf{D}_i$:

$$\mathbf{P_{ik}} = \mathbf{S_{ik}}.\mathbf{D_i}, \tag{27}$$

obtendo dessa forma uma nuvem de pontos $\mathbf{P}_{ik}$ a ser comparada no ICP que é mostrado na etapa seguinte.

A figura 17 mostra um exemplo visual da seleção de duas regiões da imagem 1 e imagem 2 obtidas a partir de pontos de vista distintos, segundo um deslocamento de translação, escala e rotação. A imagem 1 está a 0^0 de longitude e a imagem 2 esta a 15^0 em longitude, e ambas estão a 0^0em latitude, ou seja, no mesmo plano. Daremos agora um exemplo onde queremos selecionar a região da imagem 1 que contém a etiqueta de valor $k = 5$ e na imagem 2 iremos selecionar a região que contém a etiqueta de valor $k = 4$. Na figura 17(a) temos o resultado da etapa de segmentação da imagem 1 e na figura 17(b) tem-se o resultado da etapa de segmentação da imagem 2, onde é observado o valor da etiqueta de cada região. Seguindo o processo representado na equação (26) seleciona-se a região que contém a etiqueta $k = 5$ de $\mathbf{S}_1$, tendo como resultado a imagem binária $\mathbf{S}_{15}$ como mostrado na figura 17(c). Da mesma forma aplicamos a $\mathbf{S}_2$ o mesmo processo da equação (26) com etiqueta $k = 4$, obtendo a matriz binária $\mathbf{S}_{24}$ como mostra a figura 17(d). Em seguida utilizando-se da equação (27), da multiplicação escalar $\mathbf{S}_{15}.\mathbf{D}_1$ obtém-se a nuvem $\mathbf{P}_{15}$ que é mostrada na figura 17(e) e aplicando o mesmo processo de multiplicação escalar $\mathbf{S}_{24}.\mathbf{D}_2$ tem-se como resultado a nuvem $\mathbf{P}_{24}$ que é mostrado na figura 17(f).

4.3.5 Etapa 5: ICP

A Etapa 5 irá comparar as duas nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{24}$ de ambas as imagens produzidas na Etapa 4. O algoritmo ICP (descrito na seção (3.5)) irá alinhar as duas nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{24}$,

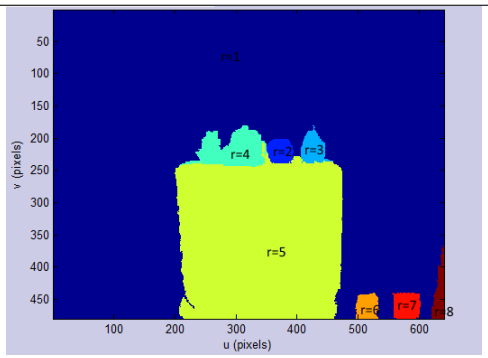
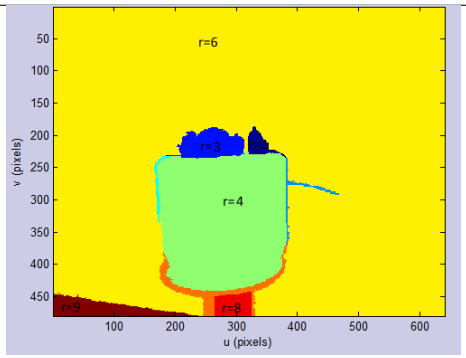
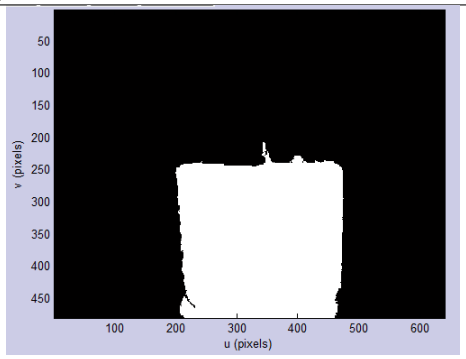
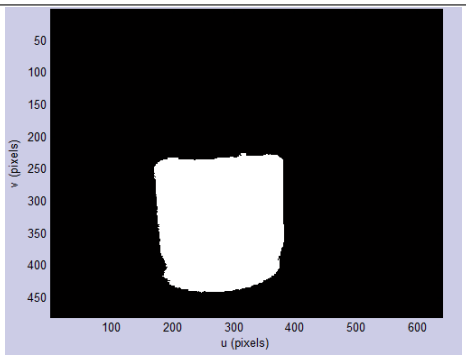
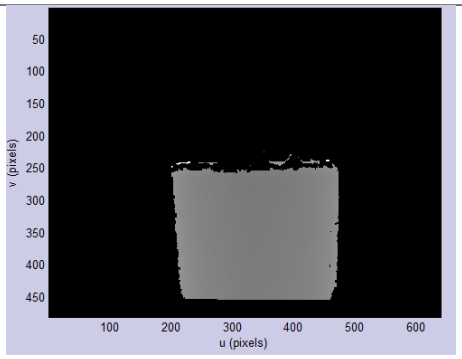
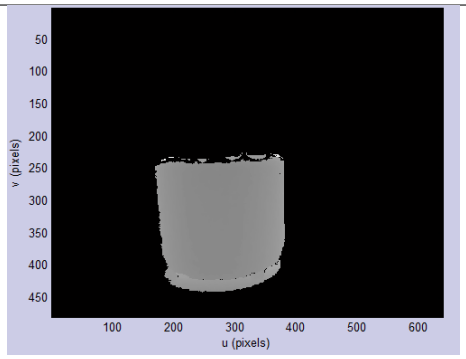
Imagem 1	Imagem 2
	
$(a)\mathbf{S}_1$ regiões etiquetadas da imagem 1	$(b)\mathbf{S}_2$ regiões etiquetadas da imagem 2
	
(c) Seleção da região de etiqueta $k = 5$ resultando na imagem binária $\mathbf{S}_{15}$	(d) Seleção da região de etiqueta $k = 4$ resultando na imagem binária $\mathbf{S}_{24}$
	
(e) Seleção no mapa de profundidade aplicando a multiplicação escalar $\mathbf{P}_{15} = \mathbf{S}_{15}.\mathbf{D}_1$	(f) Seleção no mapa de profundidade aplicando a multiplicação escalar $\mathbf{P}_{24} = \mathbf{S}_{24}.\mathbf{D}_2$

Figura 17 – Exemplo das etapas da seleção das regiões das imagens 1 e 2.
Fonte: Produzido pelo próprio autor.

obtendo como resultado a pose relativa de ambas as nuvens, bem como o erro calculado entre elas, este erro servirá como limiar na Etapa 6.

A figura 18 mostra a sequência do alinhamento do ICP na comparação das duas regiões da imagem 1 e imagem 2 exemplificadas na figura 17(c).

Continuando o exemplo dado pela figura 17, as duas regiões de etiquetas $k = 5$ e $k = 4$ estão agora na forma de nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{24}$. A nuvem $\mathbf{P}_{15}$ pode ser visualizada na cor azul (ou cinza escuro) da figura 18(a) na sua posição inicial e a nuvem $\mathbf{P}_{24}$ pode ser visualizada na cor vermelha (ou cinza claro) da figura 18(a) também em sua posição inicial. O ICP irá agora calcular a distância Euclidiana de cada ponto em $\mathbf{P}_{24}$ que esteja mais próximo de um ponto da nuvem $\mathbf{P}_{15}$ como foi explicado na subseção 3.3.1. Minimizando a distância dos pontos mais próximos de $\mathbf{P}_{15}$ em relação a $\mathbf{P}_{24}$, a nuvem $\mathbf{P}_{24}$ sofre uma transformação em termos de translação, escala e rotação produzindo assim uma nova matriz de pose para $\mathbf{P}_{24}$ e o primeiro erro que pode ser visualizado no resultado da figura 19, essa nova translação de $\mathbf{P}_{25}$ pode ser visualizada na figura 18(b). A figura 18(c) já mostra um estágio adiantado da comparação entre as nuvens, onde o erro entre elas diminuiu consideravelmente. A figura 18(d) mostra o resultado final, após ter atingido 15 iterações que foi igual ao limiar escolhido neste exemplo.

A figura 19 mostra a sequência do erro produzido pelo ICP a cada iteração entre as nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{24}$ do exemplo anterior. O método de busca utilizado foi o da árvore k-d (explicado no anexo G) e a máxima iteração utilizada foi de 15 iterações. Podemos perceber pelo gráfico da figura 19, que o erro vai se aproximando de uma constante (desvio pequeno) após a sétima comparação.

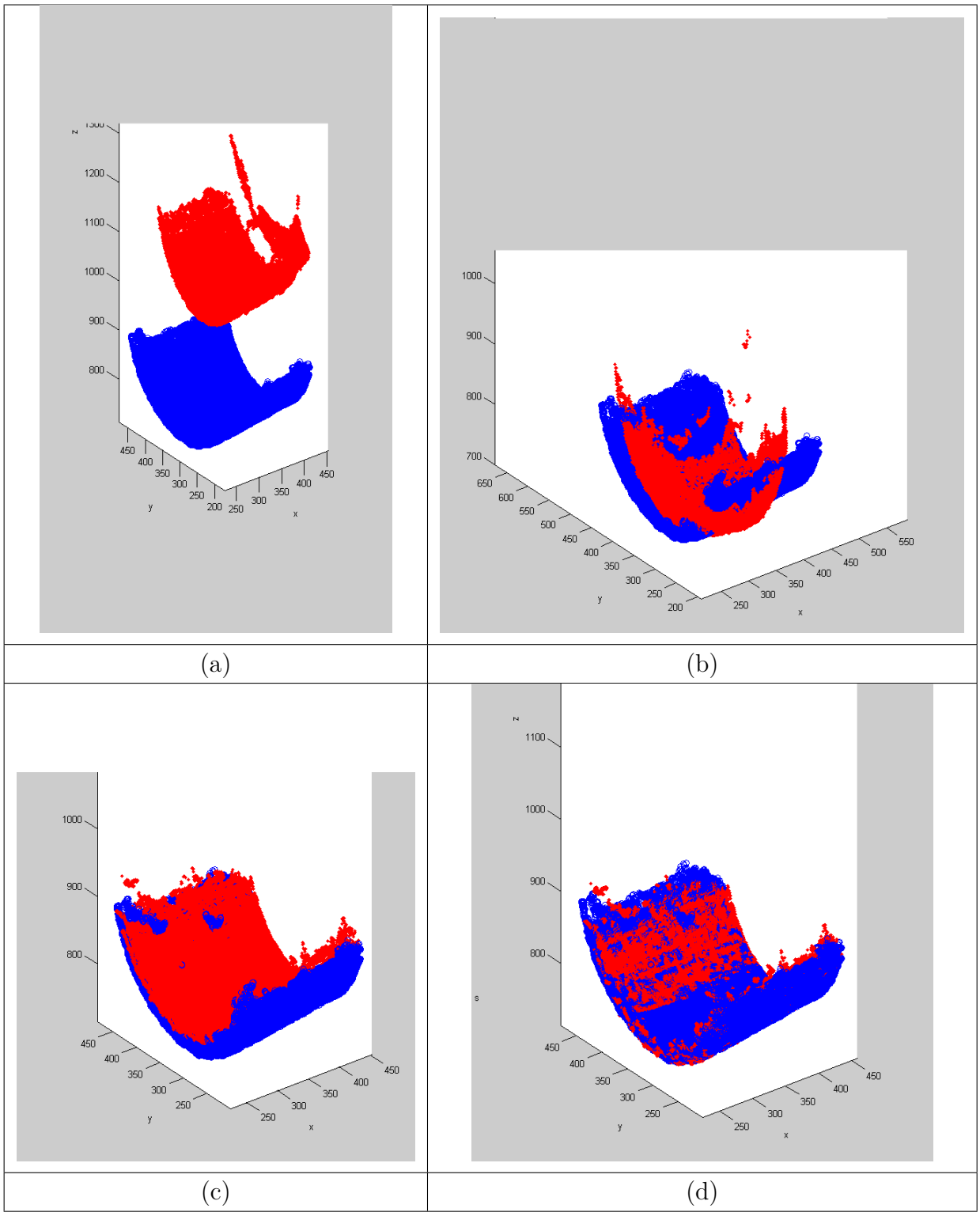


Figura 18 – Exemplo de etapas do alinhamento ICP entre duas nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{24}$.
Fonte: Produzido pelo próprio autor.

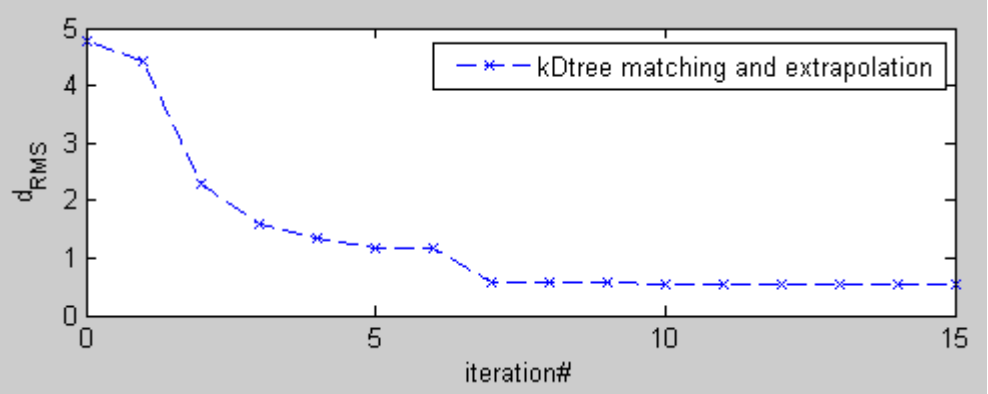


Figura 19 – Exemplo da iteração do ICP mostrando o erro a cada iteração.
Fonte: Produzido pelo próprio autor.

4.3.6 Etapa 6: Correspondência das Regiões de Menor Erro

Seguindo o exemplo anterior da Etapa 5, após a comparação entre as nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{24}$, o ICP devolve o erro entre ambas as nuvens, neste exemplo o $erro(\mathbf{P}_{15}, P_{24})$ equivalente às regiões de etiquetas $k_{15} = 5$ para imagem 1 e $k_{24} = 4$ para imagem 2. Podemos agora comparar com o erro das nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{23}$, onde $\mathbf{P}_{23}$ é a nuvem equivalente à região do laço interno anterior, cuja etiqueta da região da imagem 2 tem o valor $k_{23} = 3$. O resultado da comparação do ICP para o laço anterior entre as nuvens pode ser visto na figura 20(a), e o seu erro: $erro(\mathbf{P}_{15}, P_{23})$ pode ser visto na figura 20(b). O erro: $erro(\mathbf{P}_{15}, P_{24})$ produzido da figura 19 é comparado com o erro do laço anterior $erro(\mathbf{P}_{15}, P_{23})$ da figura 20(b). Podemos observar para esse exemplo que o erro da figura 19 é menor que o erro da figura 20(b), e dessa forma temos que a correspondência entre as nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{24}$ é maior que a correspondência entre as nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{23}$, sendo assim o par de maior correspondência é mantido, formando o vetor de correspondência $[erro(\mathbf{P}_{15}, P_{24}), k_{15}, k_{24}]$. O laço interno continua até atingir o total de regiões da imagem 2 obtendo o par de regiões de menor erro.

(a) Resultado inicial e final das nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{23}$ da direita para a esquerda, após 15 iterações.	(b) Resultado do $erro(\mathbf{P}_{15}, \mathbf{P}_{23})$, após 15 iterações.

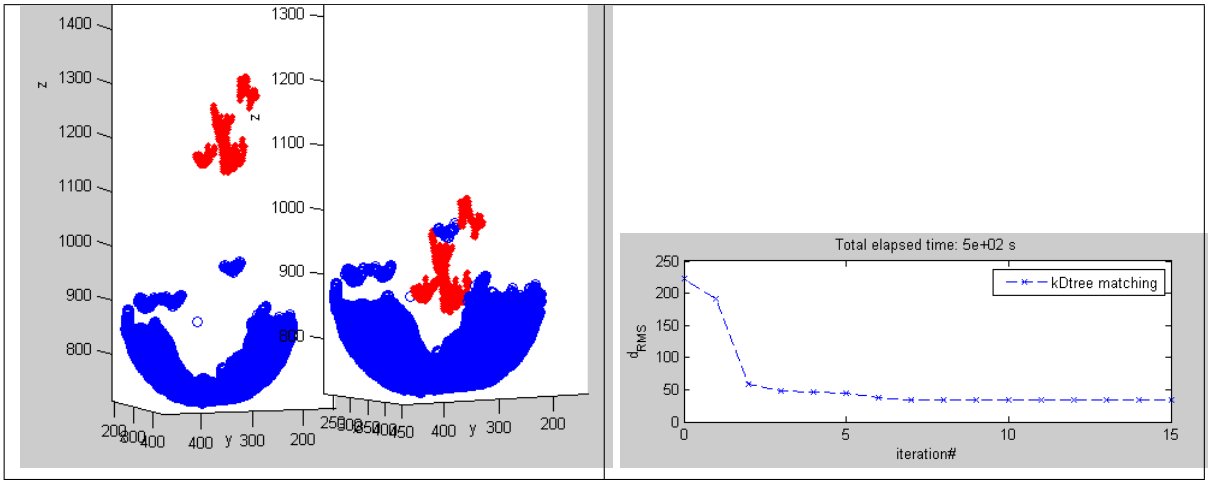


Figura 20 – Resultado final do ICP das nuvens $\mathbf{P}_{15}$ e $\mathbf{P}_{23}$.
Fonte: Produzido pelo próprio autor.

5 ARRANJO EXPERIMENTAL

O arranjo experimental adotado nesta dissertação utilizou como sensor o Kinect da Microsoft, que capturou imagens RGBD, a fim de avaliar o algoritmo de correspondência proposto neste trabalho. Esse arranjo teve como objetivo estressar o ponto de vista de uma cena capturando as imagens de forma a abranger a maioria dos pontos de vista possíveis de uma determinada cena. Além disso, tivemos o cuidado de incluir objetos de variadas formas e cores. O arranjo experimental sofreu as variações descritas abaixo:

a) Dos tipos de objetos utilizados;

b) Da quantidade de cenas;

c) Do ponto de vista de cada cena.

5.1 Dos tipos de objetos utilizados

Tivemos o cuidado de inserir objetos planos (por exemplo, utilizamos o objeto quadro como objeto plano) e objetos volumétricos (Objetos que não possuem superfície plana, como por exemplo: Garrafa Térmica).

Foram inseridos os seguintes objetos:

a) Frutas;

b) Brinquedos;

c) Almofada;

d) Garrafa Térmica;

e) Mesinha de centro;

f) Quadros Coloridos.

Observação: Outros objetos entram em cena conforme o ponto de vista. Isso ocorre devido à variação em latitude e longitude, em que além dos objetos em estudo o sensor começa a capturar objetos que entram na cena. Isso é importante, pois a complexidade da cena aumenta porque acaba aumentando o número total de regiões na execução da etapa 3, visto na subseção 4.3.3.

5.2 Da quantidade de cenas

Foram produzidas três cenas internas para este trabalho de forma a abranger objetos planos e objetos volumétricos. Para cada cena foram geradas 31 imagens.

5.2.1 Cena 1 Objetos volumétricos

A primeira cena, doravante cena 1, possui somente objetos volumétricos e coloridos, foram incluídos:

a) Frutas;

b) Brinquedos;

c) Almofada.

A figura 21 mostra um exemplo da imagem da cena 1. Essa cena tem o intuito de provocar uma maior complexidade para os algoritmos ASIFT, SIFT e SURF, pois esses algoritmos dependem de imagens planas para obter a correspondência dos pontos de interesse. Como esta cena não contém nenhum objeto plano será de enorme interesse ver qual o resultado da correspondência de cada algoritmo para esta questão.

Figura 21 – Exemplo de imagem da primeira cena.
Fonte: Produzido pelo próprio autor.

5.2.2 Cena 2 Objetos Planos

Na segunda cena temos somente objetos planos.

a) Quadro Colorido.

A figura 22 mostra um exemplo da imagem da segunda cena, doravante cena 2. Essa cena possui apenas objetos planos, veremos como os algoritmos de pontos e o MCICP reage a este tipo de cena.

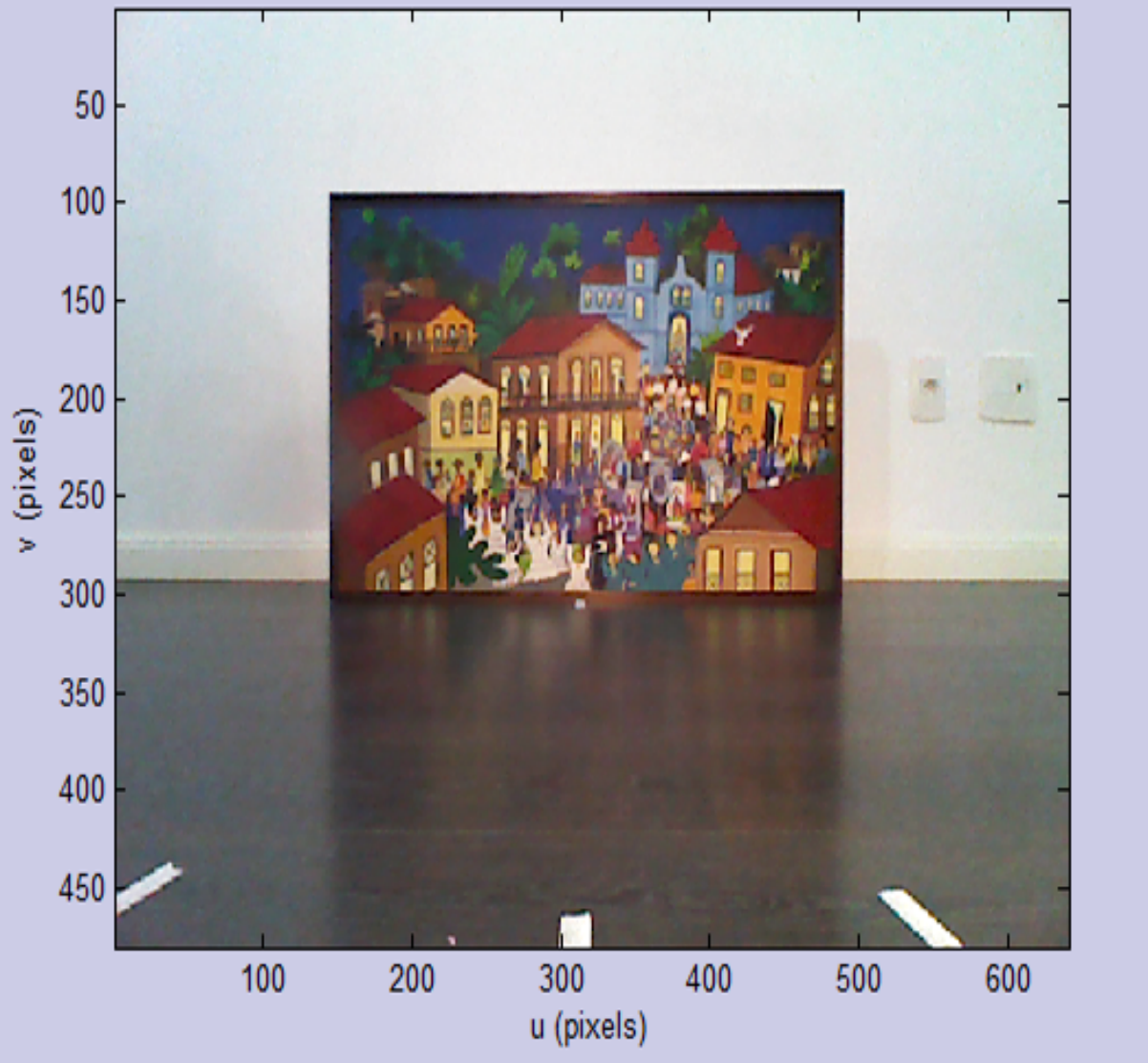


Figura 22 – Exemplo de imagem da segunda cena.
Fonte: Produzido pelo próprio autor.

5.2.3 Cena 3 Combinação de Objetos Planos com Objetos Volumétricos.

Na última cena temos uma combinação de objetos planos com objetos volumétricos.

a) Garrafa Térmica;

b) Mesinha de centro;

c) Quadros Coloridos.

A figura 23 mostra um exemplo da imagem da última cena, doravante cena 3. Essa cena possui tanto objetos planos como objetos volumétricos, veremos então como todos os algoritmos reagem com cenas mais comuns, onde há os dois tipos de objetos.

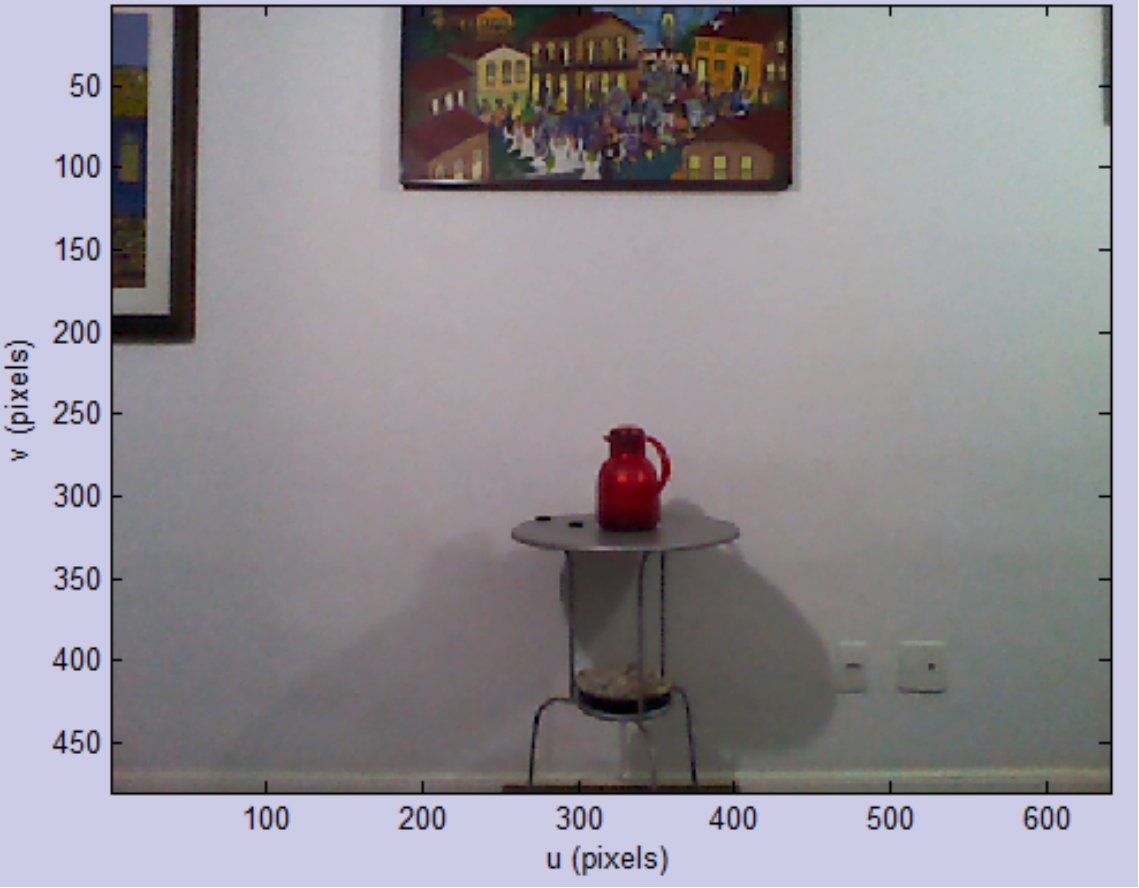


Figura 23 – Exemplo de imagem da última cena.
Fonte: Produzido pelo próprio autor.

5.3 Variação do ponto de vista de cada cena

O ponto de vista de cada cena foi alterado tanto na latitude ao redor da cena como na sua longitude, a diferença entre os ângulos em longitude é de aproximadamente 15^0; não necessariamente foi preservada a escala, bem como rotação, ou seja, esses fatores podem variar para mais ou para menos a cada translação de latitude e longitude.

Não foram avaliados controladamente as seguintes variáveis: rotação, escala, ruído, luminosidade, embaçamento (*blur*) e compressão. Somente foi avaliada a variação do ponto de vista de cada cena, alterando latitude e longitude do ponto de observação ao redor da cena.

Cada imagem de cada cena foi obtida variando a posição do sensor em sua latitude e longitude. Todas as imagens que foram variadas em latitude e longitude são comparadas com a mesma imagem de latitude 0^0 e longitude 0^0 . Dessa forma teremos as comparações de correspondência das regiões com variações de latitude de até 45^0 e longitude de até $\pm 90^0$.

Para cada ponto de vista de uma imagem, a latitude tem suas variações aproximadas

de 0^0, 30^0e 45^0, e a longitude tem suas variações aproximadas de 0^0,$\pm15^0$,$\pm30^0$,$\pm45^0$,$\pm60^0$,$\pm75^0$ e $\pm90^0$.

A figura 24 mostra alguns exemplos das variações do ponto de vista das imagens.

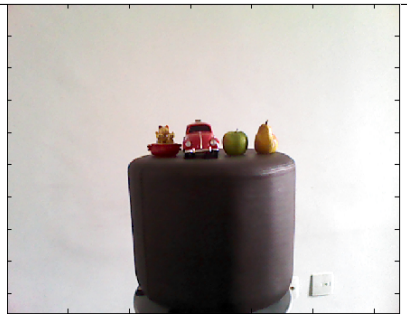
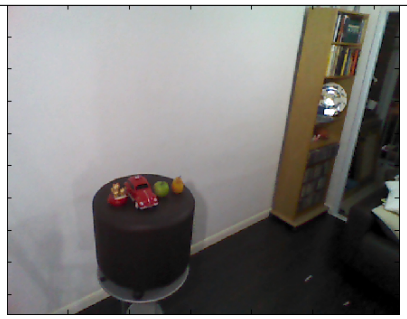
<table>
<tr><td colspan="2">Cena 1</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Ponto de Vista onde a latitude é de 0^0 e a longitude é de -15^0.</td><td>Ponto de Vista onde a latitude é de 30^0 e a longitude é de 90^0.</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Ponto de Vista onde a latitude é de 45^0 e a longitude é de -45^0.</td><td>Ponto de Vista onde a latitude é de 45^0 e a longitude é de 0^0.</td></tr>
</table>

Figura 24 – Exemplos de pontos de vista utilizados na primeira cena.
Fonte: Produzido pelo próprio autor.

A figura 25 mostra um exemplo de como foi feita a captura das imagens nas cenas. Ao implementarmos o arranjo da figura 25 procurou-se avaliar o quanto os algoritmos de pontos conseguem recuperar os verdadeiros pontos correspondentes, e o quanto o nosso algoritmo baseado no ICP consegue recuperar as verdadeiras regiões correspondentes. Na figura 25 observamos que a posição do sensor inicial se encontra na latitude de 0^0 e longitude de 0^0 em cima de uma base alaranjada (sensor está pintado de verde), essa é a principal posição capturada da imagem, pois é ela que vai ser comparada com todas as imagens capturadas em outras posições de latitude e longitude, ou seja, é a imagem de referência para a obtenção de correspondência. Cada imagem capturada do sensor em todas as variações de latitude e longitude tenta ter como foco o objeto ou objetos de estudo que no caso da figura 25 é o objeto central em cima da base cilíndrica de forma hexagonal e cor magenta. Com a variação da latitude, outros objetos que não o objeto ou objetos

de estudo podem entrar no campo de visão da imagem pelo fato do sensor transladar em latitude e longitude. Pode-se observar que, com a variação da latitude e longitude, os objetos de estudo podem ficar totalmente ou em parte oclusos, isso é interessante, pois aumenta a complexidade da cena em relação à correspondência. A figura 25 também mostra os sentidos horários e anti-horários da figura, no sentido anti-horário teremos as longitudes positivas de 0^0 à 90^0 e no sentido horário as longitudes negativas de 0^0 à -90^0. A variação das latitudes vão de 0^0 à 60^0 e, como podemos observar na figura 25, os sensores em verde estão posicionados na latitude de 0^0, os sensores em azul estão posicionados na latitude de 30^0 e os sensores em vermelho estão posicionados na latitude de 60^0.

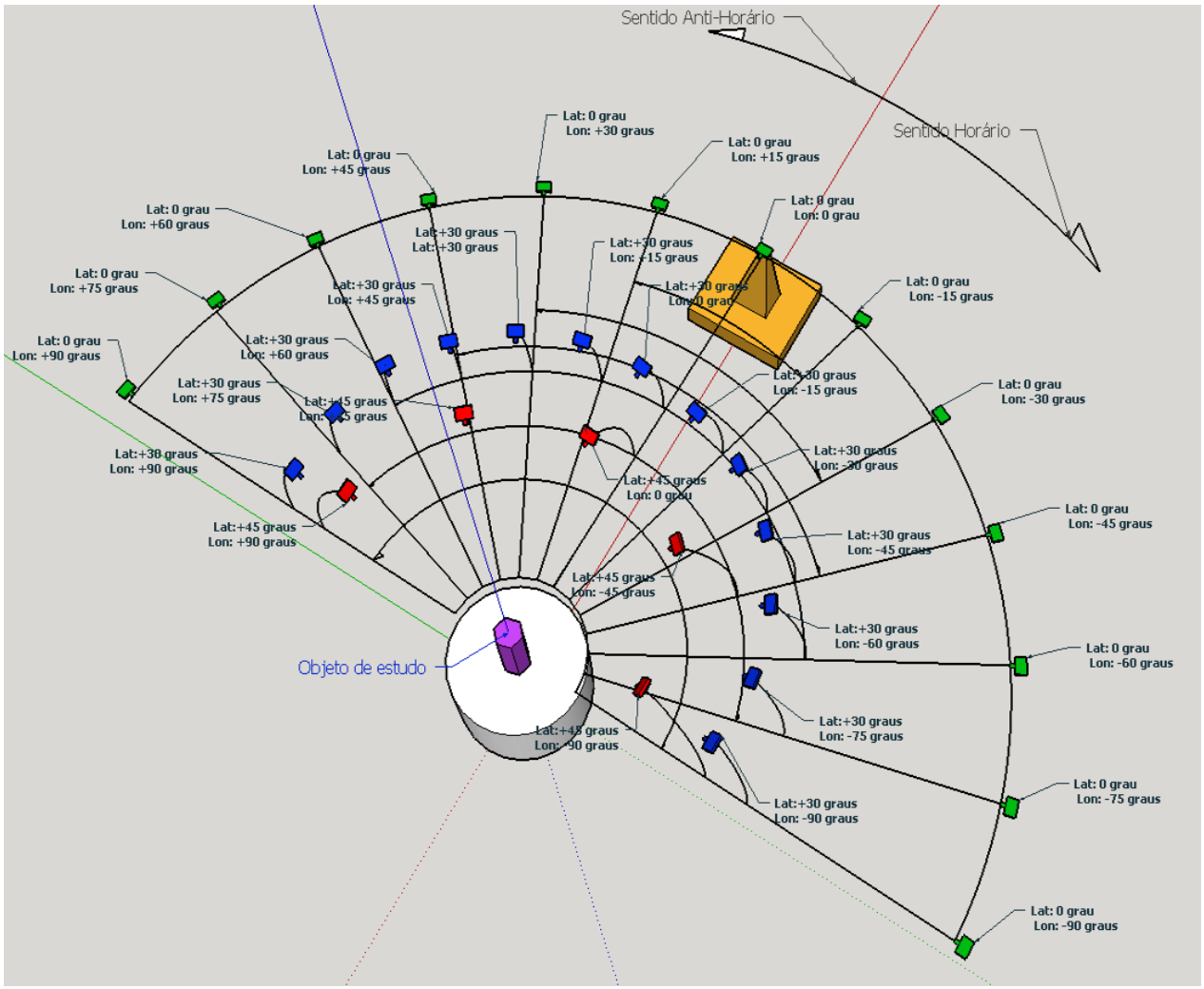


Figura 25 – Disposição da captura das imagens na cena.
Fonte: Produzido pelo próprio autor.

5.4 Cenas Internas e Externas

Esse banco de imagens foi construído a partir de três cenas internas, sendo que, de cada cena, foi extraído um total de 31 imagens a partir de diversos pontos de vista.

Figuras de cenas externas não puderam ser obtidas, pois o infravermelho do Kinect não captura as imagens de disparidade por causa do ruído produzido pela iluminação natural. O trabalho de (MANKOFF; RUSSO 2013) expõe os problemas e limitações do Kinect. Estas limitações possuem três categorias: hardware do dispositivo, propriedades ambientais e as propriedades da superfície. Aplicações do Kinect podem ser limitadas pelo intervalo de medição $(0,5 - 5m)$, FOV (até $\sim 5m^2$, mas frequentemente menor para melhorar a resolução de distância). Alterações térmicas ao longo do tempo podem reduzir a precisão. A limitação ambiental principal é que o Kinect não funciona sob luz solar intensa. Ele emite e detecta a luz IR e não consegue distinguir o sinal de retorno do fundo, o sol satura o sensor. Certas superfícies absorvem IR e não podem ser detectadas com precisão.

No próximo capítulo iremos descrever o padrão ouro de anotação, como foi gerado o padrão ouro automático com o algoritmo que gera uma imagem a priori para obtenção dos verdadeiros pontos chaves das figuras da cena 2 e 3. Iremos também analisar os resultados obtidos das três cenas.

6 RESULTADOS

Este capítulo expõe os resultados obtidos na correspondência entre as imagens e seus pontos de vista comparando MCICP (método desenvolvido deste trabalho) com os algoritmos ASIFT, SIFT e SURF em experimentos controlados. A avaliação é efetuada por um critério do índice de acertos de pontos, ou regiões, conforme o método utilizado.

Foi adotado um padrão ouro para anotação manual do MCICP porque o MCICP é baseado em regiões e temos poucas regiões para fazer a correspondência; esse processo foi aplicado para o MCICP nas três cenas. As correspondências anotadas do ASIFT, SIFT e SURF foram feitas manualmente para cena 1 (anotação padrão ouro manual). Para as cenas 2 e 3 foi desenvolvido um programa automático, para obtenção das verdadeiras correspondências dos pontos chaves dos algoritmos ASIFT, SIFT e SURF. Esse algoritmo calcula de forma automática a transformação da imagem de referência, produzindo uma imagem sintética que corresponde à imagem alvo. Esse programa será explicado detalhadamente na seção 6.2.

6.1 Padrão Ouro Anotado Manualmente

Um padrão ouro de anotação manual foi utilizado para avaliar a correspondência entre regiões. No caso do nosso algoritmo, foi desenvolvido um programa para essa anotação. Esse programa funciona da seguinte forma: Primeiro as duas imagens (imagem de referência e imagem alvo) são segmentadas pelo mesmo processo da etapa 3 subseção 4.3.3, formando um vetor de **n** regiões na imagem de referência e um vetor de **m** regiões na imagem alvo. Em seguida ele apresenta a primeira região do vetor **n** na imagem de referência envolvida por um retângulo (para melhor ser observada pelo usuário) e na imagem alvo ele apresenta a primeira região do vetor **m** envolvida por um retângulo; ambos os retângulos então são ligados por uma linha, mostrando dessa forma as regiões a serem manualmente comparadas em ambas às imagens, como mostra a figura 26(a); em seguida o programa pergunta ao usuário se a região envolvida pelo retângulo da imagem à esquerda corresponde à região envolvida pelo retângulo na imagem à direita, ou melhor dizendo, se a região à esquerda está contida na ou contém a região da imagem à direita. Em seguida um vetor de correspondência $[c, n, m, a]$ é gerado onde c é a variável de correspondência lógica com o valor 1 se o usuário responder sim e 0 caso contrário; (n, m) é o par da região em questão e a é o índice da imagem alvo. Como temos um total de 31 imagens alvo, o programa vai percorrer essas 31 imagens produzindo um total aproximado de $n \times m \times 31$ perguntas ao usuário para cada cena, produzindo assim uma tabela padrão ouro para todas as imagens alvo em relação à imagem de referência para este trabalho. Outro programa, similar ao programa do padrão ouro do MCICP, faz a anotação da correspondência de pontos dos

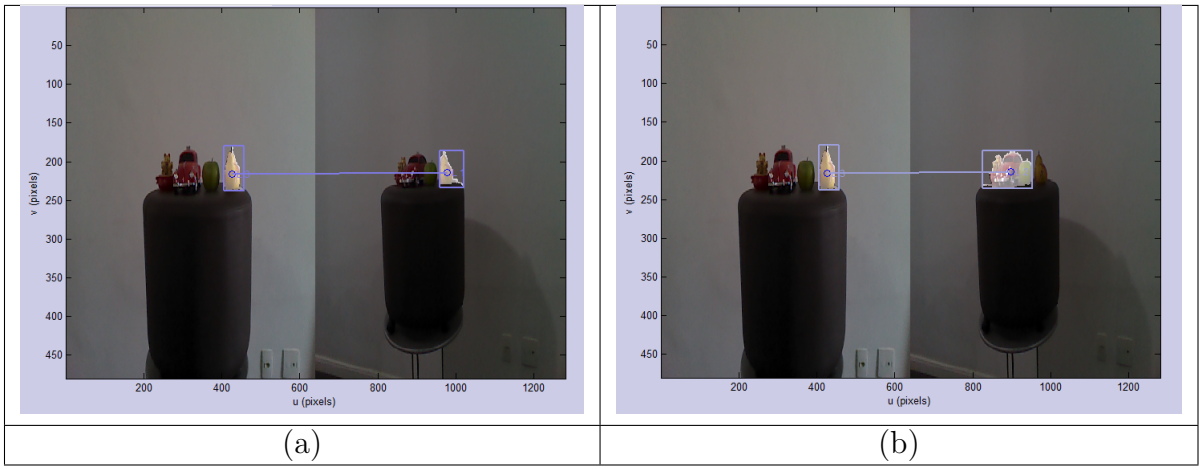


Figura 26 – Programa de formação da tabela padrão ouro no que diz respeito à comparação das regiões. Fonte: Produzido pelo próprio autor.

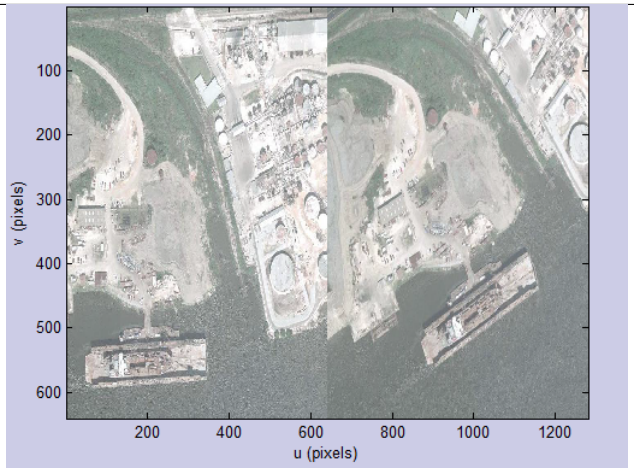
algoritmos ASIFT, SIFT e SURF, somente para a cena 1, perguntando ao usuário se os pontos interligados são correspondentes entre a imagem de referência e a imagem alvo. Da mesma forma, este programa produz um vetor de correspondência $[c, p, q, a]$ onde p é o ponto correspondente à imagem de referência e q é o ponto da imagem alvo, percorrendo as 31 imagens alvo produzindo no final uma tabela padrão ouro de pontos correspondentes, isso foi executado para cada um dos algoritmos ASIFT, SIFT e SURF.

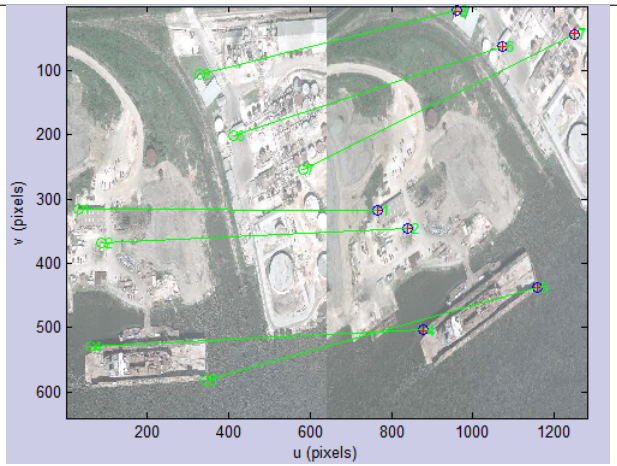
6.2 Geração Automática de Dados Padrão Ouro Para a Correspondência dos Pontos-Chave

Inicialmente executamos os algoritmos ASIFT, SIFT e SURF em cada imagem para identificar os pontos-chave. O objetivo é obter, para cada par de imagens, uma lista de pontos-chave. Uma vez que cada algoritmo gera centenas ou milhares de pontos-chave por imagem, para fazermos a correspondência de todos os pontos-chave manualmente para todos os 31 pontos de vista de cada cena seria extremamente demorado e potencialmente sujeito a erros. Felizmente, tendo uma matriz de homografia que mapeia os pontos entre a imagem de referência e a imagem alvo (imagem onde variou-se a latitude e longitude, visto no capítulo 5), irá gerar uma imagem sintética, que corresponde a imagens alvo. Tendo a imagem sintética é possível resolver automaticamente o problema de correspondência. Quando temos imagens que sofrem transformações e geram sinteticamente a imagem de destino, essas são conhecidas como imagem a priori. A figura 27 mostra um exemplo da geração de uma imagem sintética, onde a matriz de homografia calculada é igual a

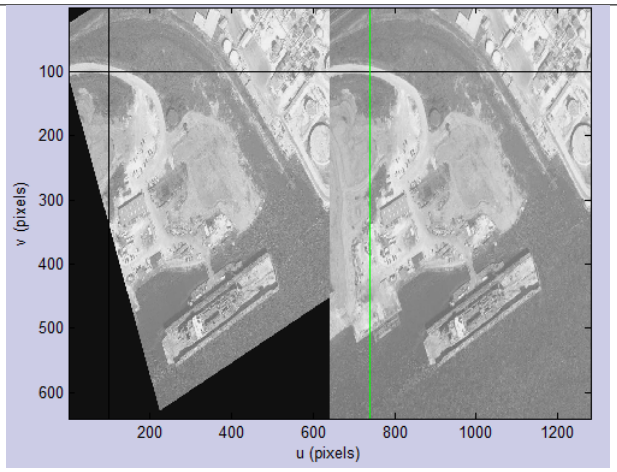
$$H = \begin{bmatrix} 0,9177 & 0,3891 & -23,8747 \\ -0,3948 & 0,9198 & 40,0929 \\ 0 & 0 & 1 \end{bmatrix}.$$

À esquerda tem-se a imagem a ser transformada e à direita, a imagem alvo.





Pontos selecionados manualmente para o cálculo da matriz de homografia.



À esquerda tem-se a imagem gerada sinteticamente e à direita tem-se a imagem alvo original

Figura 27 – Exemplo de uma imagem gerada sinteticamente.
Fonte: Produzido pelo próprio autor.

Para a transformação, que é expressa como uma homografia entre duas imagens de uma cena é usado o método a seguir: Seja T_{ij} o mapa de transformação entre o ponto da imagem i para seu ponto correspondente à imagem j. Um ponto de interesse na primeira imagem, P_i, mapeia a segunda imagem da seguinte forma: $P_i^{'} = T_{ij}(P_i)$. O ideal esperado seria que a correspondência do ponto de interesse da segunda imagem estivesse exatamente no mesmo local da transformação do ponto da primeira, ou seja, $P_j = P_i^{'}$. Na prática, consideramos uma correspondência válida se P_j e P_i estão suficientemente perto um do outro no espaço Euclidiano e em escala. Dois pontos são considerados suficientemente perto um do outro no espaço Euclidiano se a distância entre eles for menor que δ pixels. Utilizamos para isso o que foi proposto por (LOWE, 2004) onde, para que a probabilidade de que uma correspondência de pontos seja correta, deve-se determinar o cálculo da razão da distância do pixel vizinho mais próximo ao pixel do ponto chave com a distância do segundo pixel do vizinho mais próximo ao pixel do ponto chave. Essa razão deve ser menor que 0.8, para que o pixel da vizinhança mais próximo se corresponda com o pixel do ponto chave. A figura 28 mostra o gráfico da Função Densidade de Probabilidade (PDF) das razões das distâncias entre o ponto mais perto do ponto chave com o segundo mais perto do ponto chave. O gráfico da figura 28 foi calculado utilizando um banco de dados de 40.000 pontos-chave. Utilizamos esse método para fazer a correspondência dos pontos-chave da imagem de referência que contém a latitude 0^0 e longitude 0^0 com os pontos-chave das imagens de latitude e longitude de cada ponto de vista do nosso estudo.

Para se ter a correspondência dos pontos-chave para valores maiores que 300 pontos, foi desenvolvido um algoritmo que utiliza a técnica de retificação (FUSIELLO et al., 2000) e mais uma transformação homográfica discutida nesta seção. Quando a imagem possui uma cena com objetos planos, que é o caso da cena 2 e cena 3, a transformação entre as imagens pode ser modelada com uma retificação em conjunto com uma homografia 2D plana. Este método segue os passos descritos no algoritmo 6.1.

A figura 29 mostra um exemplo onde temos uma imagem de referência latitude 0^0 e longitude 0^0 mostrado na figura 29(a) e uma imagem alvo de latitude 0^0 e longitude 15^0 mostrado na figura 29(b). Os resultados seguem as etapas do algoritmo 6.1, a figura 29(c) mostra a etapa de retificação das imagens, a figura 29(d) mostra a etapa da matriz de homografia, a figura 29(e) mostra uma ampliação do ponto selecionado da figura sintética à esquerda com o seu ponto correspondente ampliado na figura 29(f) à direita.

6.3 Critério de Avaliação

No critério de avaliação, as medidas de precisão e revocação são calculadas entre a imagem de referência, ou seja, a imagem de latitude 0^0 e longitude 0^0 e as outras imagens do conjunto.

O critério de avaliação serve para a comparação entre os modelos padrões de

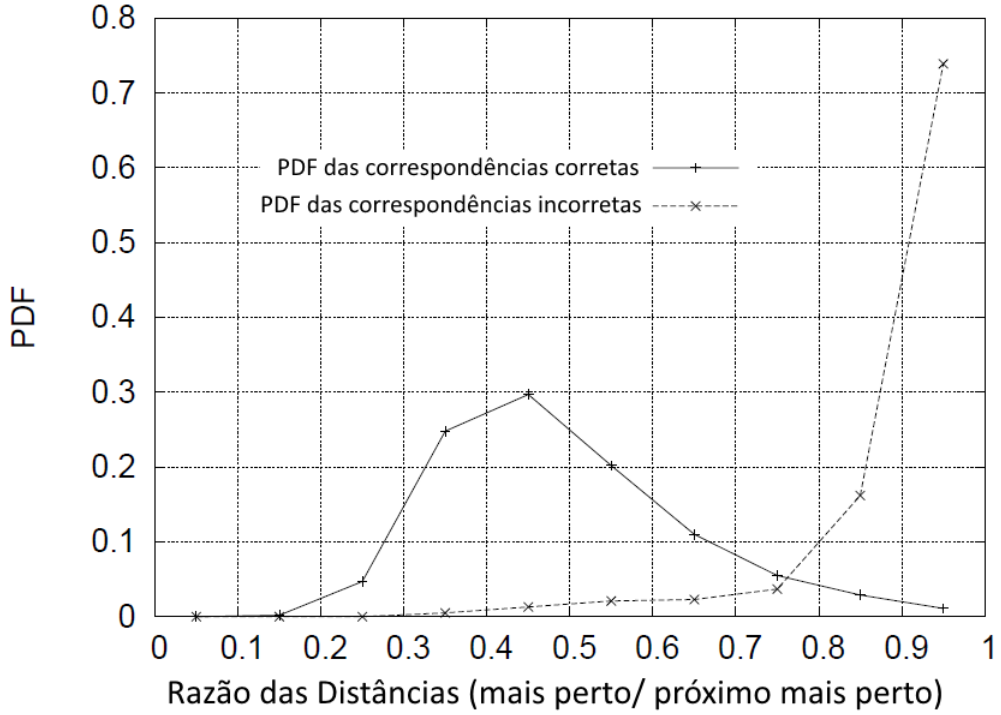


Figura 28 – Gráfico PDF versus Razão entre as distâncias do ponto mais próximo, com o segundo ponto mais próximo ao ponto-chave.
Fonte: Autor "adaptado de" (LOWE, 2004).

Algoritmo 6 Algoritmo desenvolvido para a formação das imagens a priori.

a) Aplica-se o SIFT para se ter a correspondência de pontos entre a imagem de referência e a imagem alvo; esses pontos servirão na formação da geometria epipolar, que será utilizada no processo de retificação a seguir.

b) Ambas as imagens sofrem uma retificação (ver o anexo I); a retificação irá transformar ambas as imagens de forma que os pontos epipolares (ver o anexo H) de ambas as figuras apontem para o infinito e desta forma as linhas epipolares de ambas as imagens fiquem paralelas ao eixo horizontal das imagens. Isto significa que o ponto correspondente vai estar na mesma linha horizontal da imagem retificada.

c) Aplica-se novamente o SIFT para se ter a correspondência de pontos entre as imagens; esses pontos agora servirão para o cálculo da matriz de homografia do passo seguinte.

d) Para encontramos este ponto correspondente na linha horizontal, utilizamos agora a matriz de homografia e fazemos a transformação na segunda imagem, onde o ponto correspondente sofre a seguinte transformação $P_i^{'} = T_{ij}(P_i)$, obtendo dessa forma duas imagens sintéticas correspondentes.

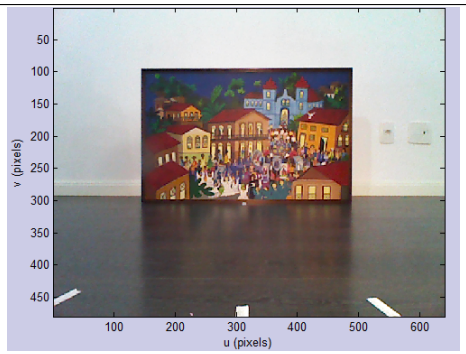
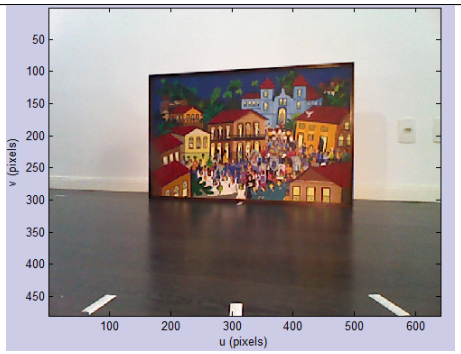
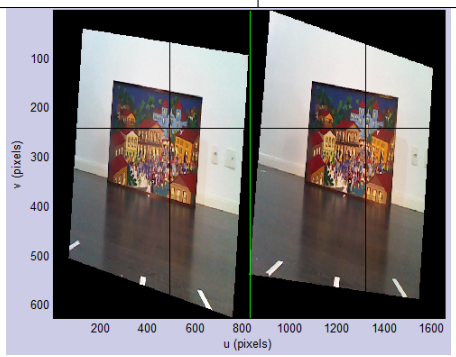
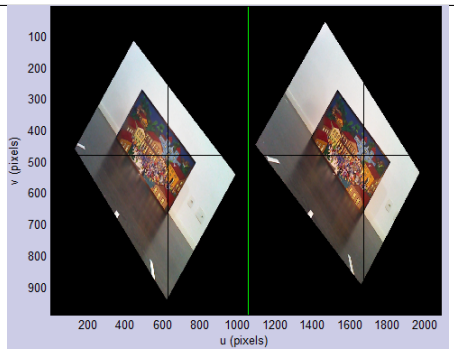
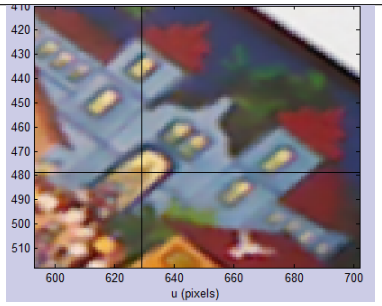
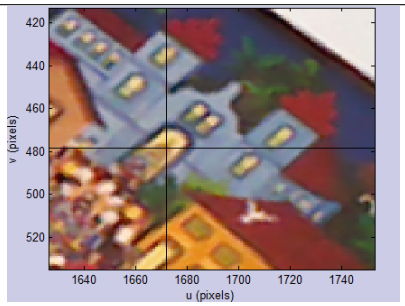
Imagem de referência latitude 0^0e longitude 0^0	Imagem alvo de latitude 0^0e longitude 15^0
	
a	b
	
c	
	
d	
	
e	f

Figura 29 – Exemplo das etapas do resultado do algoritmo 6.1 na formação das imagens a priori.
Fonte: Produzido pelo próprio autor.

correspondência de pontos, ou seja, ASIFT, SIFT e SURF aqui utilizados, como também, inclui o nosso algoritmo de correspondência de regiões, que utiliza a segmentação e o ICP; esse critério constitui-se na contagem de regiões correspondentes (no caso do método MCICP) e na contagem de pontos correspondentes (nos casos onde se tem a correspondência de pontos).

A contagem da precisão fornece em termos de porcentagem o quanto o algoritmo é preciso em sua correspondência, ou seja, provê a razão (porcentagem) dos pontos correspondentes positivos ou regiões correspondentes positivas pelo total de pontos correspondentes ou regiões correspondentes (verdadeiros e falsos positivos), ou seja,

$$Precisão = \frac{numero\,de\,verdadeiros\,positivos}{numero\,de\,verdadeiros\,positivos + numero\,de\,falsos\,positivos}. \tag{28}$$

Como a precisão não consegue capturar o quanto o algoritmo consegue recuperar as verdadeiras correspondências de todos os pontos-chave dos pares de imagens comparadas, tem-se que calcular o quanto desses pontos pressupostos foi recuperado, ou seja,

$$Revocação = \frac{numero\,de\,verdadeiros\,positivos}{numero\,de\,correspondencia\,verdadeiras}. \tag{29}$$

6.3.1 O Fator F

O Fator F é uma medida que combina e avalia a medida de compensação entre Precisão e Revocação (POWERS, 2011).

F é uma média harmônica ponderada, expressa por

$$F = \frac{1}{\alpha\frac{1}{Precisão} + (1-\alpha)\frac{1}{Revocação}} = \frac{(\beta^2+1)Precisão * Revocação}{\beta^2 Precisão + Revocação}. \tag{30}$$

Usualmente utiliza-se a fórmula balanceada de ponderação, onde $\beta = 1$, expressa por

$$F_1 = 2 * Precisão * Revocação/(Precisão + Revocação). \tag{31}$$

6.4 Curvas Precisão-Revocação

Considere um conjunto de amostras onde temos etiquetas e pontuações, a pontuação é tipicamente a saída de um classificador, onde classificações altas equivalem a etiquetas positivas. De preferência, os dados de pontuação são ordenados em ordem decrescente deixando em primeiro lugar as amostras positivas e em último lugar as amostras negativas. Na prática, a classificação não é perfeita e o posicionamento da pontuação não é ideal.

Um classificador binário classifica os dados como pontos positivos ou negativos. Se nós, antecipadamente, sabemos qual a verdadeira classificação, a performance do classificador é dada por uma tabela 2×2 chamada de tabela de contingencia ou matriz de confusão. O quadro 1 mostra como a matriz é formada.

	Dados Corretos (pontos positivos)	Dados Incorretos (pontos negativos)
Dados Selecionados (etiquetas positivas)	Verdadeiros Positivos	Falso Positivos (Erro Tipo I)
Dados Não Selecionados (etiquetas negativas)	Falso Negativos (Erro Tipo II)	Verdadeiros Negativos

Quadro 1 - Matriz de confusão

A maioria dos classificadores possui um limiar para sua classificação; por exemplo, o nosso algoritmo MCICP possuí como limiar o erro das nuvens selecionadas, pois ele está interessado em classificar a correspondência das regiões. Um algoritmo de correspondência de pontos pode ter como limiar de classificação a distância Euclidiana dos descritores da vizinhança de um ponto.

Um classificador binário irá dispor os dados do que ele acredita ser o correto como dados selecionados e o que ele acredita ser incorreto como dados não selecionados; se soubermos previamente quais dados são corretos desse conjunto de dados, podemos classificá-los em dados corretos e dados incorretos segundo a tabela 1 da seguinte forma:

a) Para dados selecionados pelo classificador, iremos selecionar os dados corretos classificando-os como verdadeiros positivos. Os dados selecionados que sobraram são os dados incorretos e iremos classificá-los como falsos positivos, preenchendo assim a primeira linha da tabela.

b) Para os dados não selecionados pelo classificador, iremos selecionar os dados corretos classificando-os como falsos negativos. Os dados não selecionados que sobraram são os dados incorretos e iremos classificá-los como verdadeiros negativos, preenchendo assim a segunda linha da tabela.

Utilizando a análise de precisão versus revocação poderíamos construir esse gráfico para o nosso algoritmo MCICP. Para cada comparação entre a imagem de referência e as imagens deslocadas pela latitude e longitude teríamos um gráfico de precisão versus revocação para o MCICP como mostra Figura 30(a). Teríamos também que prover os gráficos de precisão versus revocação para os algoritmos de pontos ASIFT, SIFT e SURF, para cada

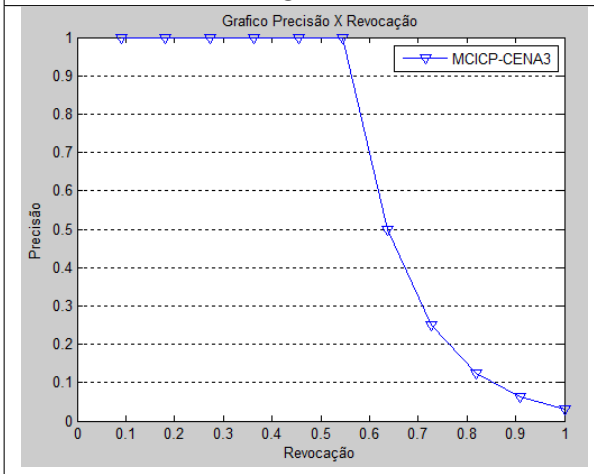
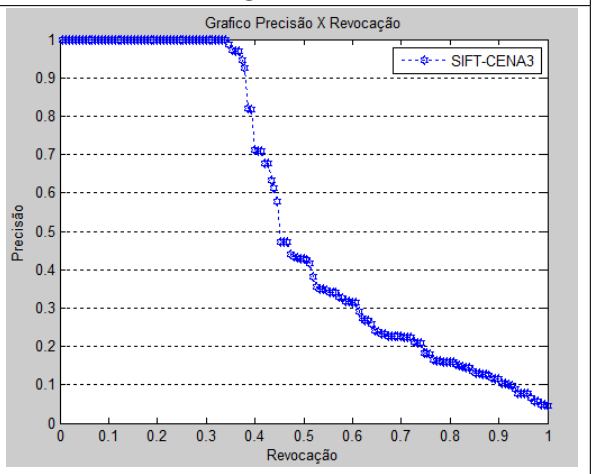
Precisão versus Revocação do MCICP entre a imagem de referencia a latitude 0^0 e a longitude 0^0 e a imagem de latitude 0^0 e a longitude 15^0	Precisão versus Revocação do SIFT entre a imagem de referencia a latitude 0^0 e a longitude 0^0 e a imagem de latitude 0^0 e a longitude 15^0
	
(a)	(b)

Figura 30 – Precisão versus Revocação do MCICP e SIFT entre a imagem de referência a latitude 0^0 e a longitude 0^0 e a imagem de latitude 0^0 e a longitude 15^0.
Fonte: Produzido pelo próprio autor.

comparação entre a imagem de referência e as imagens deslocadas pela latitude e longitude como mostra Figura 30(a). A figura 30 mostra o resultado de precisão versus revocação da cena 3 entre a imagem de referencia a latitude 0^0 e a longitude 0^0 e a imagem de latitude 0^0 e a longitude 15^0, porém sabemos que o limiar de classificação do MCICP é diferente do limiar dos algoritmos de pontos, pois o nosso algoritmo MCICP se baseia no erro entre as nuvens de pontos, enquanto que os algoritmos de pontos ASIFT,SIFT e SURF se baseiam em uma distância Euclidiana. E dessa forma não faz sentido ter uma análise em conjunto.

Poderíamos fazer somente a análise da precisão de todos os algoritmos MCICP, ASIFT, SIFT e SURF para cada ponto de vista para a cena 2 e 3, como é feito para cena 1 onde não temos como calcular os valores de revocação dos algoritmos ASIFT, SIFT e SURF, mas dessa forma não saberíamos o quanto cada algoritmo retorna sobre as verdadeiras correspondências contidas na comparação da imagem de referência com as imagens dos diferentes pontos de vista. Como é possível calcular o valor da revocação nas cenas 2 e 3, iremos contornar essa situação propondo neste trabalho uma alternativa diferente. Iremos calcular um valor de precisão versus revocação para cada ponto de vista para as cenas 2 e 3, utilizando as equações 28 e 29, onde os valores de revocação são ordenados em ordem crescente. Em seguida iremos tirar o valor da média harmônica ponderada, porém na sua forma balanceada $F1$, fazendo em seguida uma média aritmética dos valores calculados do fator F_1 para cada cena.

A seção a 6.5 utiliza essa proposta, plotando um gráfico para obtermos qual o

melhor limiar da variável t para o algoritmo MCICP.

6.5 Critério de Avaliação do Limiar t e sua Análise

O algoritmo desenvolvido nesta dissertação possui uma variável t que representa a razão entre as áreas que serão comparadas na correspondência. Por não sabermos qual o melhor valor de razão entre as áreas, pois não queremos comparar áreas muito pequenas, foi preciso obter um gráfico onde comparamos a Precisão com a Revocação para a análise do melhor valor de t para o algoritmo desenvolvido nesta dissertação. Foram analisados os seguintes valores para t: 0.05,0.10,0.20,0.25,0.30,0.4,0.5,0.6,0.7,0.8 e 0.9. Os resultados obtidos para cada um desses valores estão representados no gráfico da figura 31.

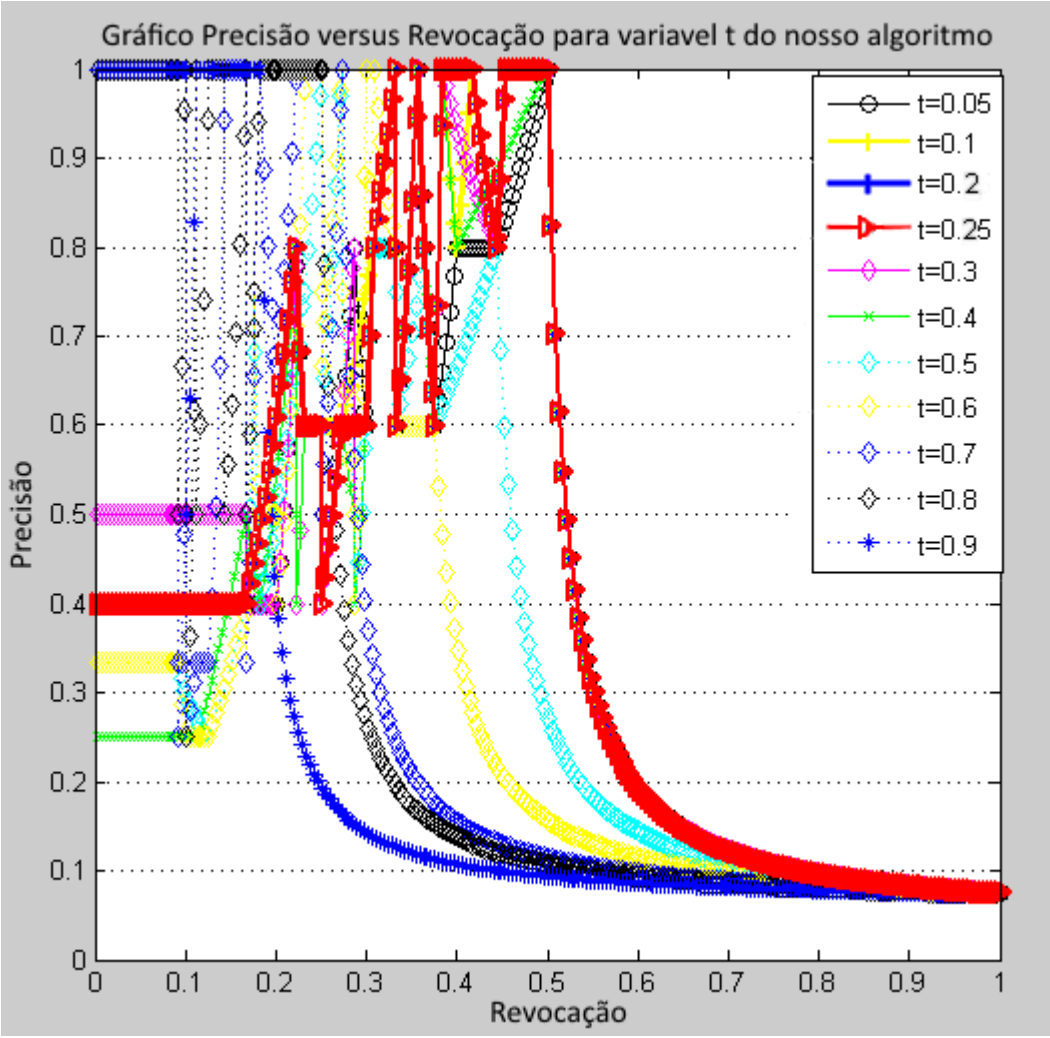


Figura 31 – Gráfico Precisão x Revocação do valor da variável t do nosso algoritmo.
Fonte: Produzido pelo próprio autor.

Como pode-se observar no gráfico da figura 31 o melhor valor que produz um bom resultado entre Revocação x Precisão é o valor de 0.25 para a variável t. No algoritmo 4, essa variável possui o nome *MinAreaThreshold*.

6.6 Resultados obtidos para Cena 1

A tabela 2 mostra os resultados para cena obtidos nas correspondências entre a imagem de referência e as imagens de cada ponto de vista com sua latitude e longitude informada na primeira coluna da tabela 2. A segunda coluna informa os resultados dos falsos positivos do MCICP para cada ponto de vista. A terceira coluna informa os resultados dos verdadeiros positivos do MCICP para cada ponto de vista. A quarta coluna informa os resultados das correspondências verdadeiras do MCICP para cada ponto de vista. O mesmo acontece para as últimas colunas onde temos os resultados obtidos pelo algoritmo ASIFT. Quando temos como valor um traço, isso nos informa que não teve como formar a correspondência verdadeira, pois não houve como formar uma imagem a priori conforme mencionado na seção 6.2.

A tabela 3 mostra os resultados obtidos nas correspondências entre a imagem de referência e as imagens de cada ponto de vista com sua latitude e longitude informadas na primeira coluna da tabela 3. A segunda coluna informa os resultados dos falsos positivos do SIFT para cada ponto de vista. A terceira coluna informa os resultados dos verdadeiros positivos do SIFT para cada ponto de vista. A quarta coluna informa os resultados das correspondências verdadeiras do SIFT para cada ponto de vista. O mesmo acontece para as três últimas colunas onde tem-se os resultados obtidos pelo algoritmo SURF.

6.6.1 Resultado da precisão versus revocação da cena 1

Apresentaremos o resultado da cena 1, avaliando a precisão versus revocação, ou somente a precisão se não houver como calcular a revocação do algoritmo.

O resultado da figura 32 mostra a precisão versus a revocação da cena 1 onde somente se apresenta o nosso algoritmo MCICP, pois não foi possível obter uma matriz de homografia que pudesse ser utilizada para corresponder os pontos no cálculo de revocação para os algoritmos ASIFT,SIFT e SURF. Não foi possível comparar com os outros algoritmos somente com este gráfico. Como foi apresentado na seção 6.4 esse gráfico não segue uma curva comum dos gráficos de precisão versus revocação pois este não utiliza um pontuador para classificar as amostras de correspondência obtidas, mas sim utiliza o resultado final de todas as correspondências calculando os valores de precisão e revocação para cada ponto de vista. Um valor muito utilizado e que nos permite saber o quanto um algoritmo pode nos retornar de informação do sistema, balanceando a precisão versus revocação, é utilizar o cálculo do fator F para os gráficos de precisão versus revocação já apresentado na seção 6.3.1 na sua forma balanceada F_1, para o gráfico da figura 32 este fator é de $F_1 = 46,11\%$; como sabemos, essa é uma média harmônica ponderada e é uma média muito conservativa, ou seja, na comparação dos dois valores (precisão versus revocação) ela vai obter um valor perto do mínimo entre os dois valores. Para melhor esclarecer o que isso significa, para este gráfico encontramos uma média de $revocação = 0,33$ e uma média

Tabela 1 – Os verdadeiros e falsos positivos e correspondências verdadeiras de cada ponto de vista dos métodos MCICP e ASIFT apresentados da cena 1.

	MCICP falsos positivos	MCICP verdadeiros positivos	MCICP correspondências verdadeiras	ASIFT falsos positivos	ASIFT verdadeiros positivos	ASIFT correspondências verdadeiras
Lat:0^0 Lon:15^0	1	4	8	38	39	-
Lat:0^0 Lon:30^0	0	5	8	29	23	-
Lat:0^0 Lon:45^0	0	5	7	38	23	-
Lat:0^0 Lon:60^0	0	5	5	39	23	-
Lat:0^0 Lon:75^0	0	5	8	25	3	-
Lat:0^0 Lon: 90^0	1	4	9	27	2	-
Lat:0^0 Lon:-15^0	1	4	14	6	262	-
Lat:0^0 Lon:-30^0	0	5	9	12	131	-
Lat:0^0 Lon:-45^0	1	4	8	30	55	-
Lat:0^0 Lon:-60^0	1	4	8	15	16	-
Lat:0^0 Lon:-75^0	2	3	9	14	12	-
Lat:0^0 Lon:-90^0	2	3	7	0	0	-
Lat:30^0 Lon:0^0	3	2	10	34	32	-
Lat:30^0 Lon:15^0	0	5	10	38	17	-
Lat:30^0 Lon:30^0	1	4	9	32	3	-
Lat:30^0 Lon:45^0	0	5	9	36	4	-
Lat:30^0 Lon:60^0	0	5	6	21	3	-
Lat:30^0 Lon:75^0	2	3	5	0	0	-
Lat:30^0 Lon:90^0	2	3	6	0	0	-
Lat:30^0 Lon:-15^0	2	3	10	39	12	-
Lat:30^0 Lon:-30^0	0	5	9	34	23	-
Lat:30^0 Lon:-45^0	1	4	7	39	11	-
Lat:30^0 Lon:-60^0	1	4	5	11	2	-
Lat:30^0 Lon:-75^0	3	2	6	17	1	-
Lat:30^0 Lon:-90^0	2	3	8	0	0	-
Lat:60^0 Lon:0^0	2	3	10	25	4	-
Lat:60^0 Lon:45^0	2	3	5	21	0	-
Lat:60^0 Lon:90^0	2	3	5	0	0	-
Lat:60^0 Lon:-45^0	1	4	5	18	3	-
Lat:60^0 Lon:-90^0	1	4	7	0	0	-

Fonte: Produzido pelo próprio autor.

Tabela 2 – Os verdadeiros e falsos positivos de cada vista dos métodos SIFT e SURF apresentados da cena 1.

	SIFT falsos positivos	SIFT verdadeiros positivos	SIFT correspondências verdadeiras	SURF falsos positivos	SURF verdadeiros positivos	SURF correspondências verdadeiras
Lat:0^0 Lon:15^0	22	12	-	1	2	-
Lat:0^0 Lon:30^0	13	8	-	0	0	-
Lat:0^0 Lon:45^0	17	4	-	0	0	-
Lat:0^0 Lon:60^0	27	2	-	0	0	-
Lat:0^0 Lon:75^0	28	2	-	0	0	-
Lat:0^0 Lon: 90^0	22	2	-	0	0	-
Lat:0^0 Lon:-15^0	17	47	-	0	12	-
Lat:0^0 Lon:-30^0	10	24	-	0	5	-
Lat:0^0 Lon:-45^0	13	8	-	0	2	-
Lat:0^0 Lon:-60^0	25	2	-	0	1	-
Lat:0^0 Lon:-75^0	22	0	-	0	0	-
Lat:0^0 Lon:-90^0	26	3	-	0	0	-
Lat:30^0 Lon:0^0	26	8	-	1	1	-
Lat:30^0 Lon:15^0	20	8	-	0	0	-
Lat:30^0 Lon:30^0	12	6	-	0	0	-
Lat:30^0 Lon:45^0	20	1	-	0	0	-
Lat:30^0 Lon:60^0	21	2	-	0	0	-
Lat:30^0 Lon:75^0	16	3	-	0	0	-
Lat:30^0 Lon:90^0	28	0	-	0	0	-
Lat:30^0 Lon:-15^0	16	11	-	0	1	-
Lat:30^0 Lon:-30^0	8	4	-	0	1	-
Lat:30^0 Lon:-45^0	21	1	-	1	0	-
Lat:30^0 Lon:-60^0	19	1	-	0	0	-
Lat:30^0 Lon:-75^0	16	1	-	0	0	-
Lat:30^0 Lon:-90^0	20	2	-	0	0	-
Lat:60^0 Lon:0^0	12	3	-	0	0	-
Lat:60^0 Lon:45^0	14	1	-	0	0	-
Lat:60^0 Lon:90^0	16	1	-	0	0	-
Lat:60^0 Lon:-45^0	16	1	-	0	0	-
Lat:60^0 Lon:-90^0	22	1	-	0	0	-

Fonte: Produzido pelo próprio autor.

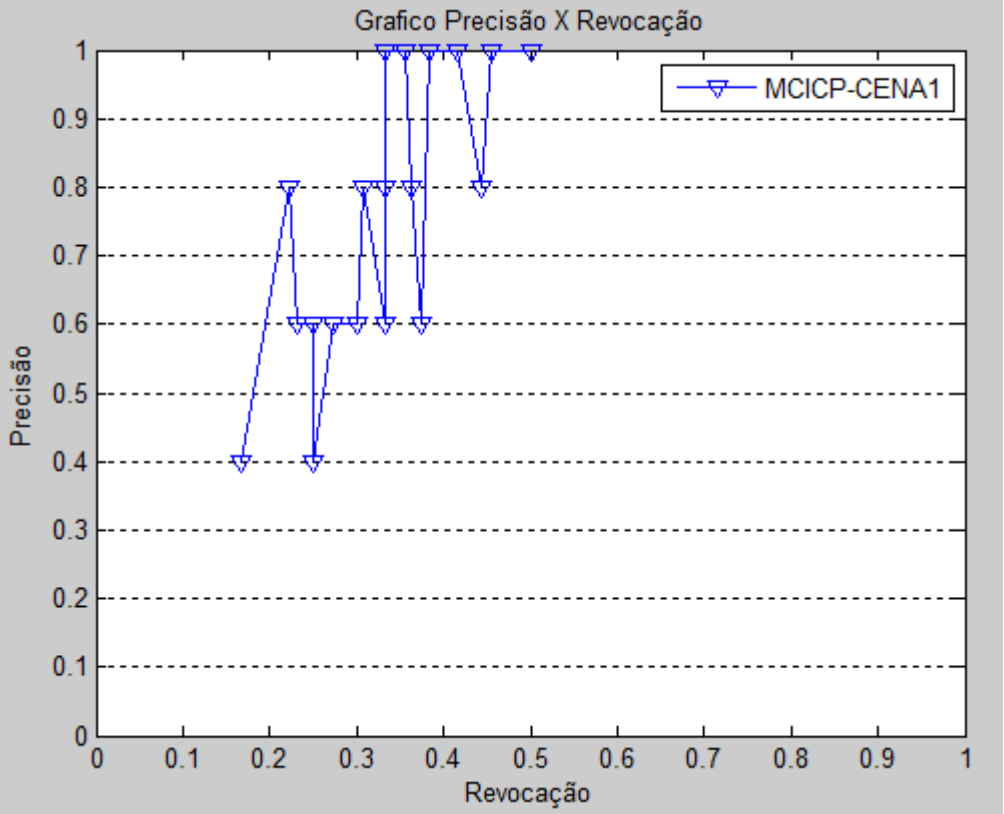


Figura 32 – Gráfico Precisão x Revocação Cena 1.
Fonte: Produzido pelo próprio autor.

de $precisão = 0,75$ resultando em um valor de $F_1 = 0,46$. Perceba que o valor de F_1 está perto do valor de *revocação*, podemos então dizer que o nosso algoritmo tem um valor médio em torno de $46,11\%$, tanto para a revocação quanto para a precisão para o nosso algoritmo MCICP.

6.6.2 Resultado de precisão da cena 1

Apresentamos agora o resultado da precisão entre os algoritmos para cena 1.

Na análise do gráfico da figura 33 na cena 1 que não possui objetos planos, o resultado mostra a precisão dos algoritmos. Na figura 33 o ponto de vista varia da longitude 15^0 até 90^0 no plano de latitude 0^0. Nessa figura na longitude de 15^0, o ASIFT começa com uma precisão em torno de 50% que é abaixo do resultado do SURF $\sim 68\%$ e do MCICP que é de 80%; o SIFT começa com uma precisão de $\sim 35\%$. Na longitude de 30^0 o SURF cai sua precisão a *zero* e mantém este valor até o final, o ASIFT diminui sua precisão para $\sim 43\%$ mantendo uma média de 40% de precisão até os 60^0, a partir disso ele tem uma queda atingindo 10% de precisão mas ainda é superior ao SIFT e o SURF. Isso condiz com o resultado do artigo (MOREL; YU, 2009) onde na correspondência em pontos de vista o ASIFT supera o SIFT. Observamos também, que o MCICP, ao contrário dos outros algoritmos, sobe a 100% e se mantém até 75^0 de longitude, caindo a 80% de

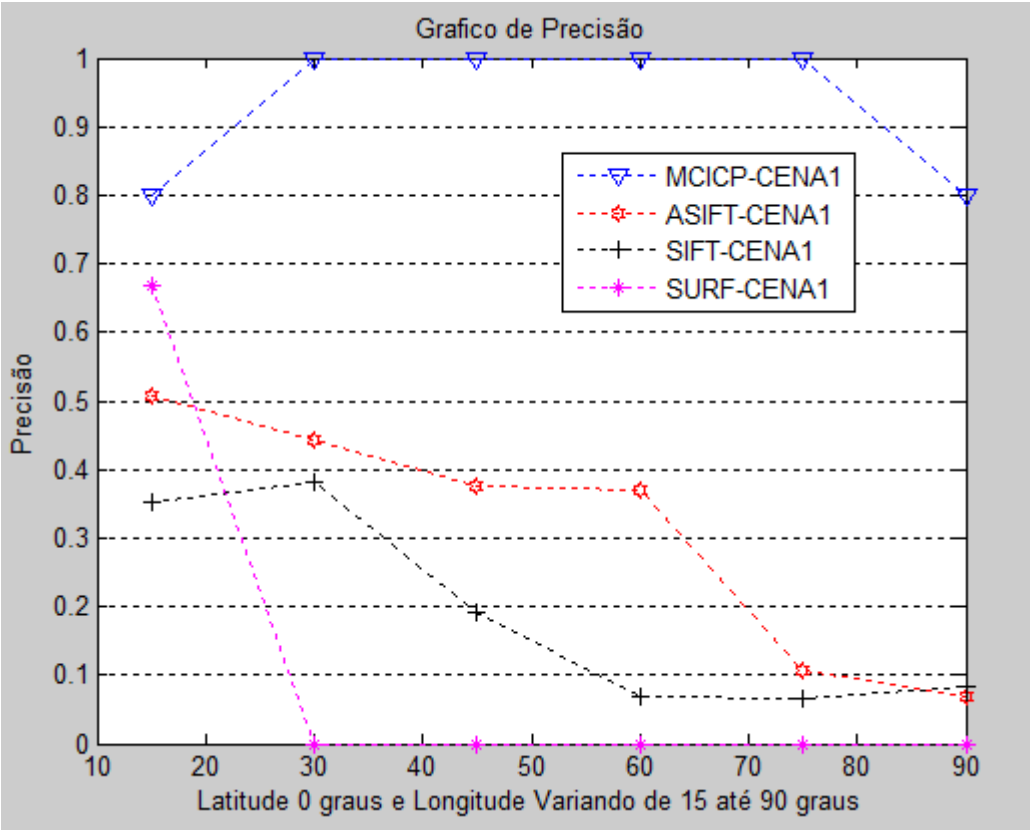


Figura 33 – Porcentagem de acertos em cada mudança de visão onde a longitude varia de 15^0 até 90^0 no plano de latitude 0^0.
Fonte: Produzido pelo próprio autor.

precisão na longitude de 90^0, superando todos os outros algoritmos com uma média de $\sim 93\%$ de precisão, enquanto que a média do ASIFT é de $\sim 31\%$ de precisão, a média do SIFT é de $\sim 20\%$ e a média do SURF é de $\sim 11\%$. Portanto o melhor resultado da figura 33 é o obtido pelo nosso método MCICP, onde mantém um alto índice de precisão até os 90^0 de longitude.

Continuando a análise da cena 1, o resultado de precisão mostrado pela figura 34, onde temos a latitude de 0^0 e longitudes variando de -15^0 até -90^0, nos mostra que o ASIFT começa com uma precisão em torno de $\sim 100\%$ na latitude de -15^0 quase igualando ao resultado do SURF que é de 100%, acima do MCICP que é de 80% e também do SIFT que é de $\sim 73\%$, na longitude de -30^0 o MCICP ultrapassa o ASIFT indo a 100% igualando com o SURF onde o ASIFT diminui para $\sim 90\%$. A partir desse ponto, o SURF mantém 100% até o 60^0 caindo abruptamente em seguida a *zero* continuando com esse valor até o final. O MCICP diminui chegando a 60% em 90^0. O SURF mantém uma superioridade, porem até 60^0, o MCICP nesse caso tem uma média superior de $\sim 76\%$ contra $\sim 67\%$ do SURF, onde o ASIFT teve uma média de $\sim 58\%$ e o SIFT uma média de $\sim 35\%$.

Já na figura 35 para cena 1, o gráfico mostra a precisão dos algoritmos agora

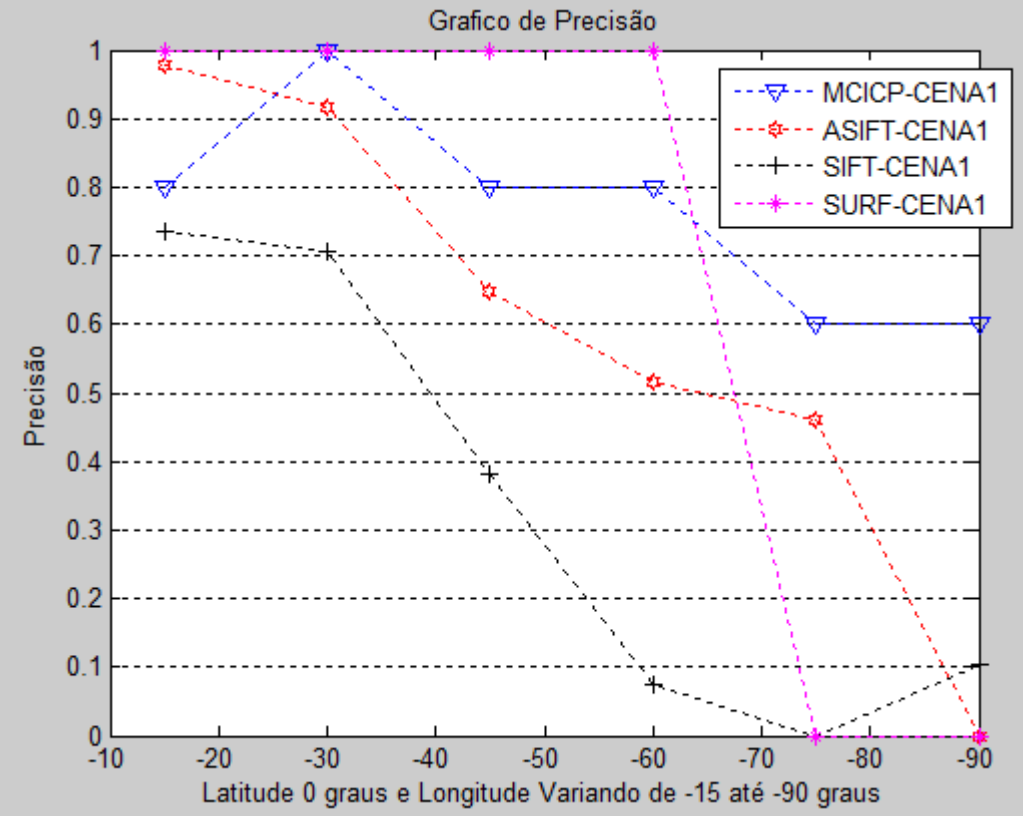


Figura 34 – Porcentagem de acertos em cada mudança de visão onde a longitude varia de -15^0 até -90^0 no plano de latitude 0^0.
Fonte: Produzido pelo próprio autor.

variando da longitude 0^0 até 90^0 na latitude 30^0. O resultado de precisão nos mostra que o ASIFT junto com o SURF iniciam com uma precisão em torno de $\sim 50\%$ na longitude 0^0 acima do MCICP que é de 40% e também do SIFT que é de $\sim 23\%$. Na longitude de 15^0 o MCICP ultrapassa o ASIFT indo a 100% o SURF cai para *zero* e continua assim até o final. O ASIFT cai para $\sim 10\%$ em 30^0 mantendo assim até 60^0 depois cai à *zero*, continuando assim até o final. O SIFT supera o ASIFT chegando a $\sim 33\%$ a 30^0 mas logo em seguida cai mantendo uma média de $\sim 10\%$ até 75^0, caindo a *zero* logo em seguida. Fica bem claro neste gráfico a superioridade do MCICP mantendo em média um valor de precisão de $\sim 77\%$ até 90^0.

Analogamente na figura 36 o gráfico mostra a precisão dos algoritmos agora variando da longitude -15^0 até -90^0 na latitude 30^0. O resultado de precisão mostra que o SURF começa com uma precisão em torno de 100% na longitude de -15^0 acima do MCICP que é de 60% e também do SIFT que é de $\sim 40\%$ e do ASIFT que começa com uma precisão de $\sim 23\%$. Na longitude de -30^0 o MCICP iguala o SURF indo a 100%, o ASIFT supera o SIFT e atinge $\sim 40\%$ e o SIFT fica um abaixo em $\sim 33\%$, a partir desse ponto o SURF cai à *zero* mantendo essa posição até o final. O SIFT e o ASIFT caem igualando a $\sim 5\%$ a 75^0, em seguida o SIFT supera mas não chega a 10%. Aqui também o MCICP é superior

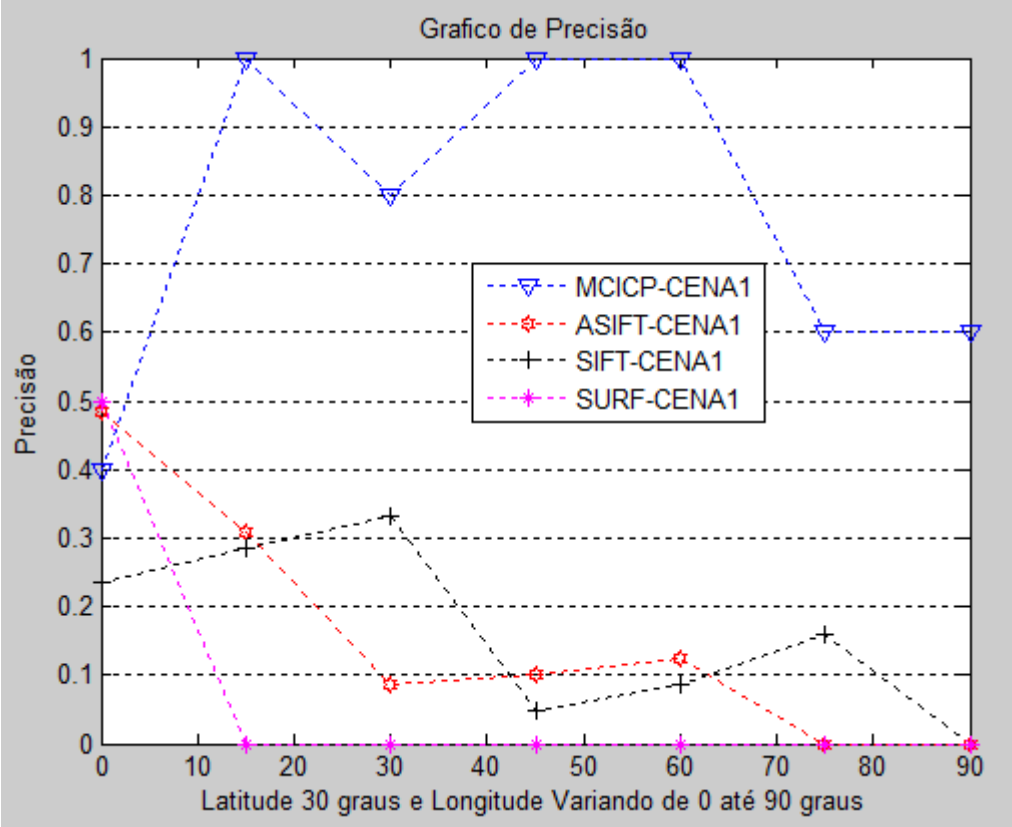


Figura 35 – Porcentagem de acertos em cada mudança de visão onde a longitude varia de 0^0 até 90^0 na latitude 30^0.
Fonte: Autor Produzido pelo próprio autor.

mantendo em média um valor de precisão de $\sim 70\%$ até 90^0.

O gráfico da figura 37 é a conjunção do gráfico 35 com 36 variando da longitude -90^0 até 90^0 na latitude 30^0. Pode-se observar pela figura 37 que o melhor resultado agora é o obtido pelo MCICP como foi mostrado nos gráficos da figura 35 e 36.

O resultado de precisão mostrado pela figura 38 para cena 1, onde há um aumento da latitude agora tendo o valor de 60^0 e longitudes variando de -90^0,-45^0,0^0,45^0 e 90^0. Faremos a análise do centro do gráfico onde temos a latitude de 60^0 e longitude 0^0 para as laterais direita e esquerda. Vemos que o SURF não pontua neste gráfico. O SIFT começa com uma precisão de 20% abaixo do MCICP com precisão de 60% e diminui na lateral direita caindo para $\sim 7\%$ mantendo esse valor até 90^0, simetricamente ocorre aproximadamente o mesmo à esquerda até -90^0. O ASIFT começa abaixo do SIFT com um valor de $\sim 14\%$, à direita ele acompanha o SIFT na queda em 45^0 de longitude com um valor de precisão de $\sim 5\%$ caindo pra *zero* em 90^0, e à esquerda ele tem uma leve subida de $\sim 15\%$ e logo cai à zero em -90^0. O MCICP supera todos tendo à direita um valor de precisão constante de 60% e à esquerda ele sobe para 80% e mantém constante até noventa. É interessante notar que o ASIFT e o MCICP têm um desempenho melhor na variação da longitude à esquerda e com o SIFT este mantém constante o final ocorrendo o

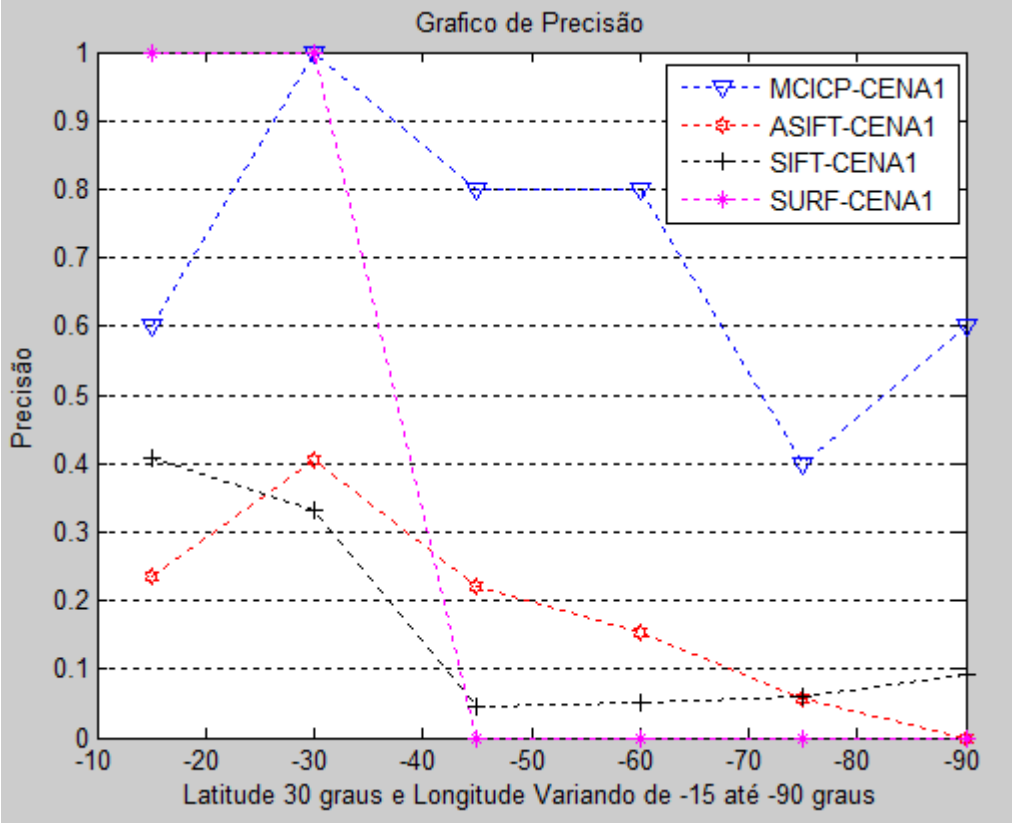


Figura 36 – Porcentagem de acertos em cada mudança de visão onde a longitude varia de -15^0 até -90^0 na latitude 30^0.
Fonte: Produzido pelo próprio autor.

mesmo no MCICP. Pode-se observar pela figura 38 que o melhor resultado é o obtido pelo MCICP , onde a sua média de precisão é de 68%.

6.7 Resultados Obtidos para Cena 2

A tabela 4 mostra os resultados obtidos para a cena 2 para correspondências entre a imagem de referência e as imagens de cada ponto de vista com sua latitude e longitude informada na primeira coluna da tabela 4. A segunda coluna informa os resultados dos falsos positivos do MCICP para cada ponto de vista. A terceira coluna informa os resultados dos verdadeiros positivos do MCICP para cada ponto de vista. A quarta coluna informa os resultados das correspondências verdadeiras do MCICP para cada ponto de vista. O mesmo acontece para as últimas colunas onde temos o algoritmo ASIFT. Quando temos como valor um traço, isso nos informa que não teve como formar a correspondência verdadeira pois não houve como formar uma imagem a priori.

A tabela 5 mostra os resultados obtidos nas correspondências entre a imagem de referência e as imagens de cada ponto de vista com sua latitude e longitude informada na primeira coluna da tabela 5. A segunda coluna informa os resultados dos falsos positivos do SIFT para cada ponto de vista. A terceira coluna informa os resultados dos verdadeiros

Tabela 3 – Os verdadeiros e falsos positivos e correspondências verdadeiras de cada vista dos métodos MCICP e ASIFT apresentados da cena2.

	MCICP verdadeiros positivos	MCICP falsos positivos	MCICP correspondências verdadeiras	ASIFT falsos positivos	ASIFT verdadeiros positivos	ASIFT correspondências verdadeiras
Lat:0^0 Lon:15^0	3	2	6	835	1158	5762
Lat:0^0 Lon:30^0	2	3	10	1966	2	4737
Lat:0^0 Lon:45^0	2	3	6	428	232	5318
Lat:0^0 Lon:60^0	2	3	6	-	-	-
Lat:0^0 Lon:75^0	2	3	5	-	-	-
Lat:0^0 Lon: 90^0	1	4	5	-	-	-
Lat:0^0 Lon:-15^0	1	4	4	736	846	5782
Lat:0^0 Lon:-30^0	2	3	7	402	421	5436
Lat:0^0 Lon:-45^0	2	3	6	616	11	4872
Lat:0^0 Lon:-60^0	2	3	6	-	-	-
Lat:0^0 Lon:-75^0	4	1	6	-	-	-
Lat:0^0 Lon:-90^0	2	3	4	-	-	-
Lat:30^0 Lon:0^0	3	2	9	810	142	5123
Lat:30^0 Lon:15^0	2	3	7	484	432	5480
Lat:30^0 Lon:30^0	3	2	8	485	90	5023
Lat:30^0 Lon:45^0	2	3	6	-	-	-
Lat:30^0 Lon:60^0	0	5	5	-	-	-
Lat:30^0 Lon:75^0	1	4	5	-	-	-
Lat:30^0 Lon:90^0	0	5	5	-	-	-
Lat:30^0 Lon:-15^0	3	2	8	334	281	5574
Lat:30^0 Lon:-30^0	2	3	8	339	206	5350
Lat:30^0 Lon:-45^0	1	4	6	277	105	5321
Lat:30^0 Lon:-60^0	3	2	7	-	-	-
Lat:30^0 Lon:-75^0	1	4	6	-	-	-
Lat:30^0 Lon:-90^0	1	4	4	-	-	-
Lat:60^0 Lon:0^0	1	4	7	-	-	-
Lat:60^0 Lon:45^0	2	3	6	-	-	-
Lat:60^0 Lon:90^0	0	5	5	-	-	-
Lat:60^0 Lon:-45^0	3	2	6	-	-	-
Lat:60^0 Lon:-90^0	0	5	5	-	-	-

Fonte: Produzido pelo próprio autor.

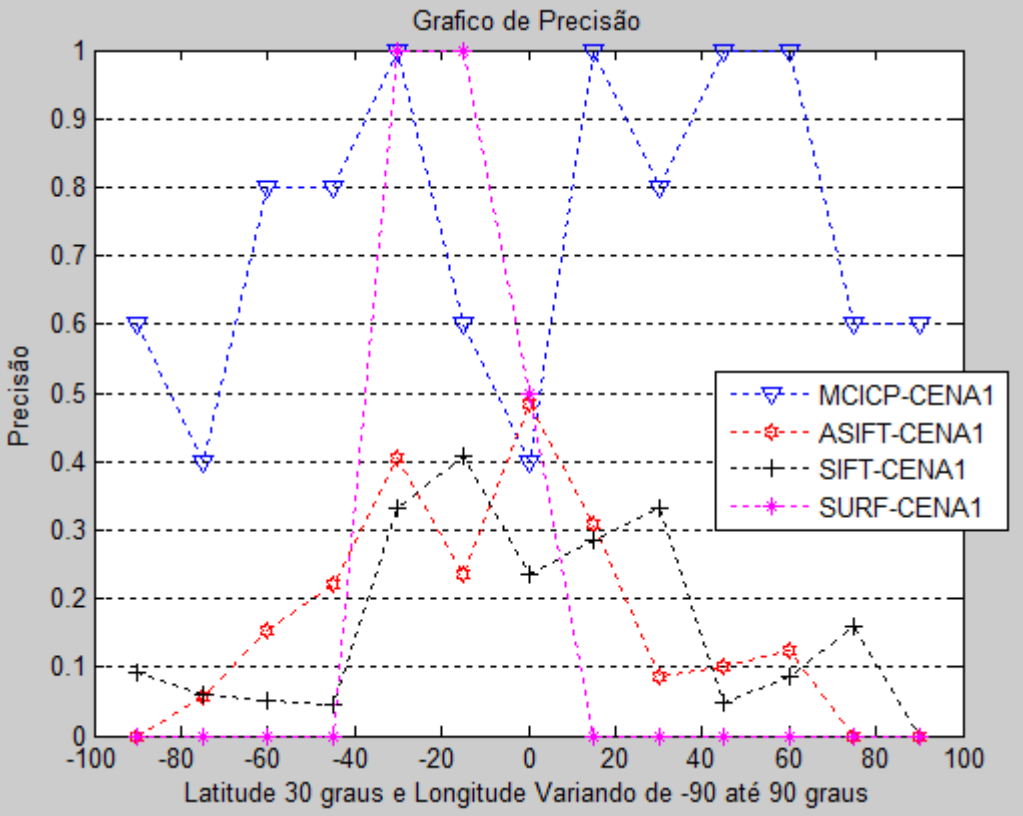


Figura 37 – Porcentagem de acertos em cada mudança de visão onde a longitude varia de -90^0 até 90^0 na latitude 30^0.
Fonte: Produzido pelo próprio autor.

positivos do SIFT para cada ponto de vista. A quarta coluna informa os resultados das correspondências verdadeiras do SIFT para cada ponto de vista. O mesmo acontece para as últimas colunas onde temos o algoritmo SURF.

6.7.1 Resultado da precisão versus revocação da cena 2

Apresentaremos o resultado da cena 2, avaliando a precisão versus revocação. Esse resultado engloba todos os pontos de vista, onde a imagem de referência de latitude 0^0 e longitude 0^0 é comparada com todas as outras 31 imagens dispostas em latitude e longitude conforme o arranjo apresentado na seção 5.3 para todos os algoritmos onde teve resultados não vazios de revocação (vazios está representado com traço nas tabelas 4 e 5).

Na análise do resultado da cena 2 onde temos somente um objeto plano, o resultado de precisão versus revocação, mostrado pela figura 39, para o ASIFT foi uma média de $revocação = 0,06$ e uma média de $precisão = 0,32$ o que resulta em um valor de $F_1 = 0,1$. Podemos então dizer que o algoritmo ASIFT tem um valor médio em torno de 10%, tanto para a revocação quanto para a precisão. Da mesma forma encontramos para o SURF uma média de $revocação = 0,10$ e uma média de $precisão = 0,88$ o que resulta em um valor de $F_1 = 0,17$. Podemos então dizer que o algoritmo SURF tem um valor médio

Tabela 4 – Os verdadeiros e falsos positivos e correspondências verdadeiras de cada vista dos métodos SIFT e SURF apresentados da cena2.

	SIFT falsos positivos	SIFT verdadeiros positivos	SIFT correspondências verdadeiras	SURF falsos positivos	SURF verdadeiros positivos	SURF correspondências verdadeiras
Lat:0^0 Lon:15^0	35	327	475	11	197	878
Lat:0^0 Lon:30^0	53	263	491	8	99	828
Lat:0^0 Lon:45^0	52	226	458	10	42	779
Lat:0^0 Lon:60^0	-	-	-	-	-	-
Lat:0^0 Lon:75^0	-	-	-	-	-	-
Lat:0^0 Lon: 90^0	-	-	-	-	-	-
Lat:0^0 Lon:-15^0	42	265	482	5	164	852
Lat:0^0 Lon:-30^0	49	252	502	5	82	831
Lat:0^0 Lon:-45^0	54	163	519	0	45	814
Lat:0^0 Lon:-60^0	-	-	-	-	-	-
Lat:0^0 Lon:-75^0	-	-	-	-	-	-
Lat:0^0 Lon:-90^0	-	-	-	-	-	-
Lat:30^0 Lon:0^0	70	250	445	23	53	797
Lat:30^0 Lon:15^0	51	270	489	5	92	834
Lat:30^0 Lon:30^0	52	181	442	15	41	756
Lat:30^0 Lon:45^0	-	-	-	-	-	-
Lat:30^0 Lon:60^0	-	-	-	-	-	-
Lat:30^0 Lon:75^0	-	-	-	-	-	-
Lat:30^0 Lon:90^0	-	-	-	-	-	-
Lat:30^0 Lon:-15^0	46	217	443	4	62	834
Lat:30^0 Lon:-30^0	37	186	487	19	35	809
Lat:30^0 Lon:-45^0	36	91	461	0	12	782
Lat:30^0 Lon:-60^0	-	-	-	-	-	-
Lat:30^0 Lon:-75^0	-	-	-	-	-	-
Lat:30^0 Lon:-90^0	-	-	-	-	-	-
Lat:60^0 Lon:0^0	-	-	-	-	-	-
Lat:60^0 Lon:45^0	-	-	-	-	-	-
Lat:60^0 Lon:90^0	-	-	-	-	-	-
Lat:60^0 Lon:-45^0	-	-	-	-	-	-
Lat:60^0 Lon:-90^0	-	-	-	-	-	-

Fonte: Produzido pelo próprio autor.

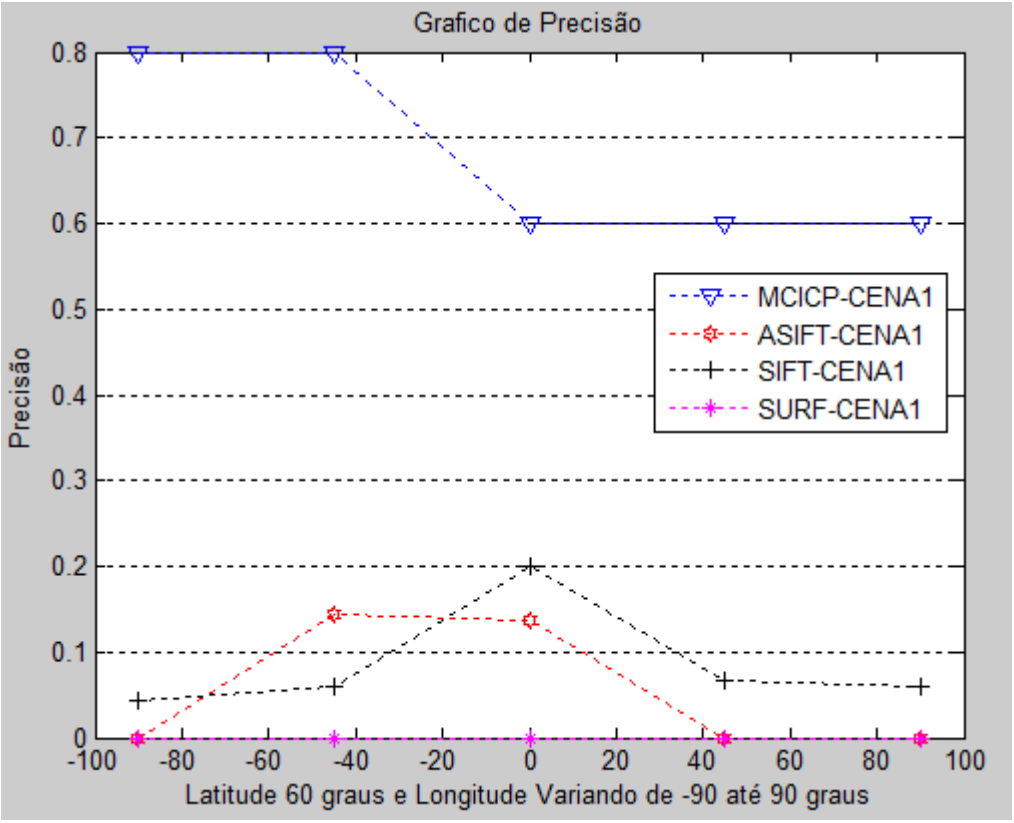


Figura 38 – Porcentagem de acertos em cada mudança de visão onde a longitude varia de -90^0 até 90^0 na latitude 60^0.
Fonte: Produzido pelo próprio autor.

em torno de 17%, tanto para a revocação quanto para a precisão. No caso do algoritmo SIFT tem-se uma média de $revocação = 0,47$ e uma média de $precisão = 0,81$ o que resulta em um valor de $F_1 = 0,60$. Novamente podemos então dizer que o algoritmo SIFT possui um valor médio em torno de 60% tanto para a revocação quanto para a precisão. E por último encontramos os valores para o nosso algoritmo MCICP, que obteve uma média de $revocação = 0,33$ e uma média de $precisão = 0,41$ o que resulta em um valor de $F_1 = 0,36$, ou seja, possui um valor médio em torno de 36% tanto para a revocação quanto para a precisão.

6.8 Resultados Obtidos para Cena 3

A tabela 6 mostra os resultados obtidos nas correspondências entre a imagem de referência e as imagens de cada ponto de vista com sua latitude e longitude informada na primeira coluna da tabela 6. A segunda coluna informa os resultados dos falsos positivos do MCICP para cada ponto de vista. A terceira coluna informa os resultados dos verdadeiros positivos do MCICP para cada ponto de vista. A quarta coluna informa os resultados das correspondências verdadeiras do MCICP para cada ponto de vista. O mesmo acontece para as últimas colunas onde temos o algoritmo ASIFT. Quando temos como valor um

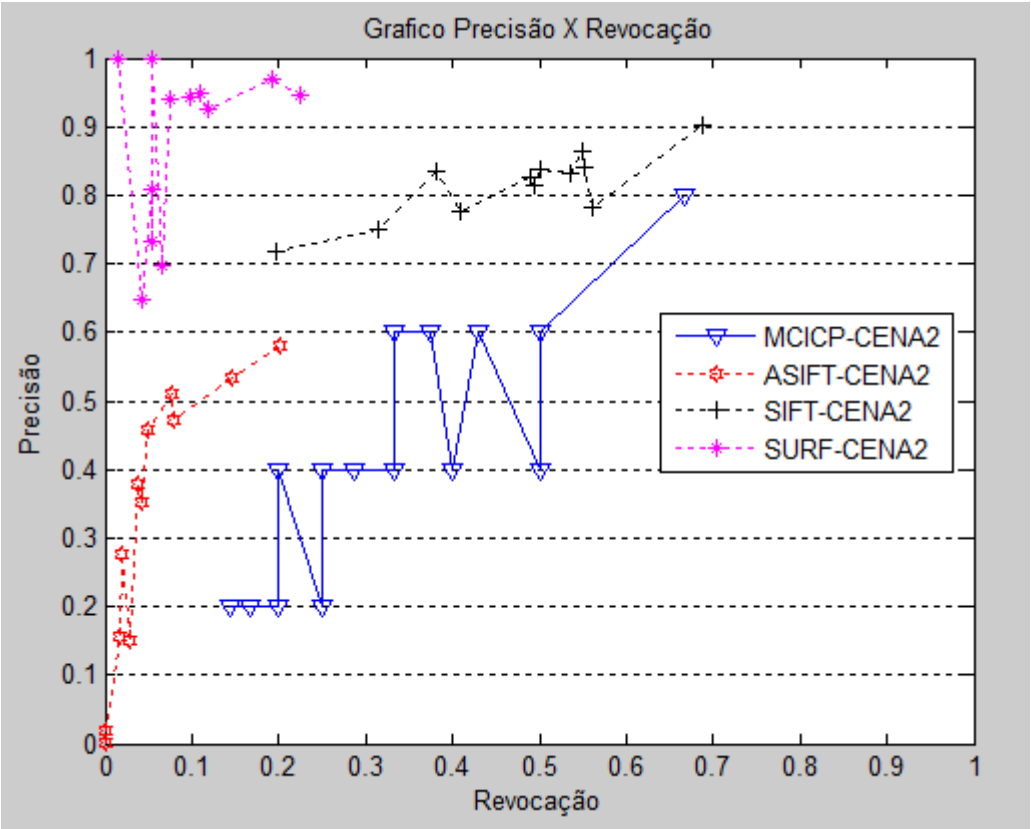


Figura 39 – Gráfico Precisão x Revocação Cena 2.
Fonte: Produzido pelo próprio autor.

traço, isso nos informa que não teve como formar a correspondência verdadeira pois não houve como formar uma imagem a priori.

A tabela 7 mostra os resultados obtidos nas correspondências entre a imagem de referência e as imagens de cada ponto de vista com sua latitude e longitude informada na primeira coluna da tabela 7. A segunda coluna informa os resultados dos falsos positivos do SIFT para cada ponto de vista. A terceira coluna informa os resultados dos verdadeiros positivos do SIFT para cada ponto de vista. A quarta coluna informa os resultados das correspondências verdadeiras do SIFT para cada ponto de vista. O mesmo acontece para as últimas colunas onde temos o algoritmo SURF.

6.8.1 Resultado da precisão versus revocação da cena 3

Apresentaremos o resultado da cena 3, avaliando a precisão versus revocação. Esse resultado engloba todos os pontos de vista, onde a imagem de referência de latitude 0^0 e longitude 0^0 é comparada com todas as outras 31 imagens dispostas em latitude e longitude conforme o arranjo apresentado na seção 5.3 para todos os algoritmos onde teve resultados não vazios de revocação (vazios está representado com traço nas tabelas 6 e 7).

A análise do resultado da cena 3 onde temos objetos planos e objetos volumétricos,

Tabela 5 – Os verdadeiros e falsos positivos e correspondências verdadeiras de cada vista dos métodos MCICP e ASIFT apresentados da cena3

	MCICP falsos positivos	MCICP verdadeiros positivos	MCICP correspondências verdadeiras	ASIFT falsos positivos	ASIFT verdadeiros positivos	ASIFT correspondências verdadeiras
Lat:0^0 Lon:15^0	0	6	11	181	137	1956
Lat:0^0 Lon:30^0	2	4	11	170	70	1905
Lat:0^0 Lon:45^0	2	4	9	114	39	1709
Lat:0^0 Lon:60^0	3	3	10	-	-	-
Lat:0^0 Lon:-15^0	3	3	9	315	202	2090
Lat:0^0 Lon:-30^0	3	3	7	222	172	2227
Lat:0^0 Lon:-45^0	3	3	7	143	34	1887
Lat:0^0 Lon:-60^0	4	2	8	-	-	-
Lat:30^0 Lon:0^0	4	2	7	36	14	1420
Lat:30^0 Lon:15^0	4	2	10	42	6	1404
Lat:30^0 Lon:30^0	3	3	14	35	7	1397
Lat:30^0 Lon:45^0	3	3	13	-	-	-
Lat:30^0 Lon:60^0	3	3	11	-	-	-
Lat:30^0 Lon:-15^0	4	2	6	-	-	-
Lat:30^0 Lon:-30^0	4	2	8	-	-	-
Lat:30^0 Lon:-45^0	4	2	6	-	-	-
Lat:30^0 Lon:-60^0	4	2	9	-	-	-

Fonte: Produzido pelo próprio autor.

pode-se observar qual a reação dos algoritmos que se baseiam em figuras planas como foi comentado no capítulo 5, o resultado de precisão versus revocação, mostrado pela figura 40, encontrou-se para o ASIFT uma média de $revocação = 0,04$ e uma média de $precisão = 0,29$ o que resulta em um valor de $F_1 = 0,07$. Podemos então dizer que o algoritmo ASIFT tem um valor médio em torno de 7%, tanto para a revocação quanto para a precisão. Da mesma forma encontramos para o SURF uma média de $revocação = 0,05$ e uma média de $precisão = 0,65$ o que resulta em um valor de $F_1 = 0,1$. Podemos então dizer que o algoritmo SURF tem um valor médio em torno de 10%, tanto para a revocação quanto para a precisão. No caso do algoritmo SIFT tem-se uma média de $revocação = 0,20$ e uma média de $precisão = 0,27$ o que resulta em um valor de $F_1 = 0,23$. Novamente podemos então dizer que o algoritmo SIFT possui um valor médio em torno de 23% tanto para a revocação quanto para a precisão. E por último encontramos os valores para o nosso algoritmo MCICP, onde este obteve uma média de $revocação = 0,32$ e uma média de $precisão = 0,50$ onde resulta em um valor de $F_1 = 0,39$, ou seja, tem um valor médio em torno de 39%, tanto para a revocação quanto para a precisão.

Tabela 6 – Os verdadeiros e falsos positivos e correspondências verdadeiras de cada vista dos métodos SIFT e SURF apresentados da cena3.

	SIFT falsos positivos	SIFT verdadeiros positivos	SIFT correspondências verdadeiras	SURF falsos positivos	SURF verdadeiros positivos	SURF correspondências verdadeiras
Lat:0^0 Lon:15^0	50	67	150	13	49	538
Lat:0^0 Lon:30^0	55	16	122	7	25	549
Lat:0^0 Lon:45^0	53	9	136	13	9	513
Lat:0^0 Lon:60^0	-	-	-	-	-	-
Lat:0^0 Lon:-15^0	-	-	-	-	-	-
Lat:0^0 Lon:-30^0	-	-	-	-	-	-
Lat:0^0 Lon:-45^0	55	51	114	11	54	521
Lat:0^0 Lon:-60^0	46	39	125	4	47	527
Lat:30^0 Lon:0^0	55	14	82	6	5	483
Lat:30^0 Lon:15^0	-	-	-	-	-	-
Lat:30^0 Lon:30^0	48	0	125	5	9	321
Lat:30^0 Lon:45^0	61	0	90	9	7	265
Lat:30^0 Lon:60^0	52	2	283	7	11	281
Lat:30^0 Lon:-15^0	-	-	-	-	-	-
Lat:30^0 Lon:-30^0	-	-	-	-	-	-
Lat:30^0 Lon:-45^0	-	-	-	-	-	-
Lat:30^0 Lon:-60^0	-	-	-	-	-	-

Fonte: Produzido pelo próprio autor.

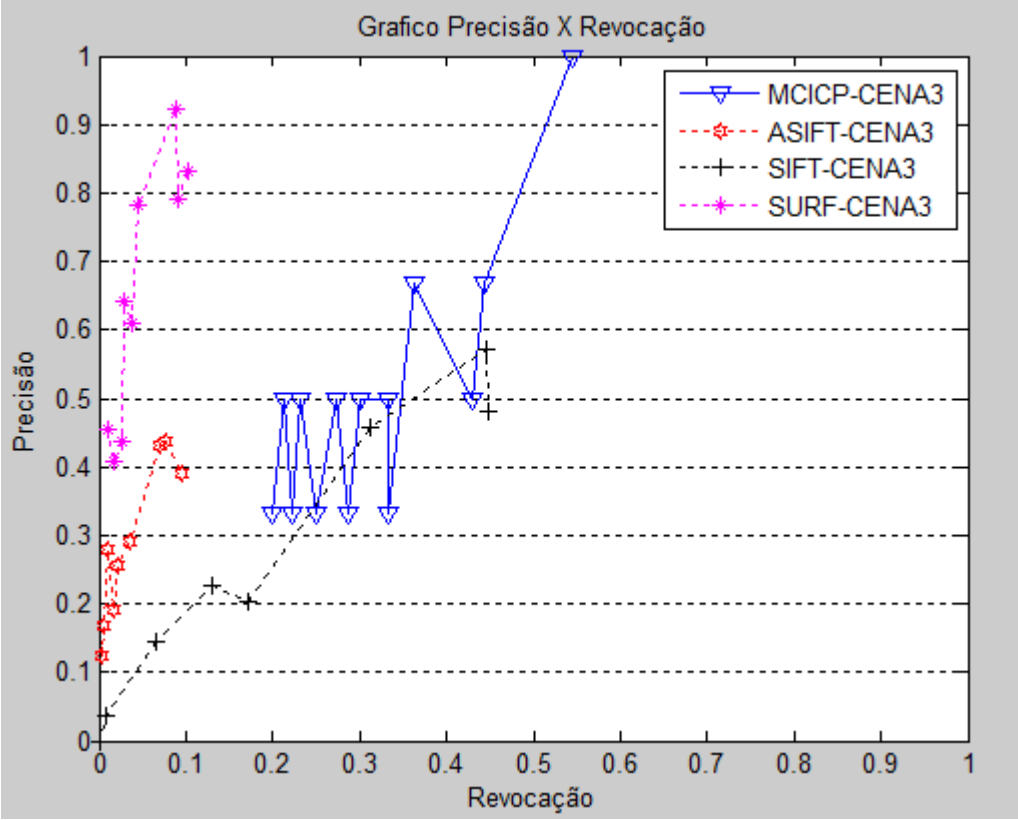


Figura 40 – Gráfico Precisão x Revocação Cena 3.
Fonte: Produzido pelo próprio autor.

7 DISCUSSÃO

Foram tratados neste trabalho os algoritmos: ASIFT, SIFT e SURF de correspondência de pontos e o nosso algoritmo MCICP de correspondência de regiões.

O algoritmo ASIFT propõe fazer uma correspondência de pontos simulando a latitude e longitude de um plano da figura com transformadas afins. Essa simulação abrange várias inclinações de um plano em uma figura, sendo uma região espacial de longitude ϕ e a latitude θ em que a simulação de longitude segue uma série geométrica e a simulação da latitude segue uma série aritmética com uma latitude máxima de 72^0, conforme descrito no algoritmo 2.1. Para cada simulação, o ASIFT faz a correspondência das imagens utilizando o algoritmo SIFT. Em virtude das várias simulações, o ASIFT acumula um número maior de pontos para cada imagem. Seu algoritmo de correspondência é um algoritmo similar ao SIFT. A escolha do ASIFT para esta dissertação vem do fato dele simular latitudes e longitudes,de acordo com o arranjo desta dissertação. Além disso ele utiliza, como base de comparação o SIFT, que é um dos algoritmos considerados estado da arte pela literatura.

O algoritmo SIFT é considerado um dos melhores algoritmos de detecção de pontos, sendo utilizado pela maioria dos algoritmos de correspondência. Na detecção de extremos do espaço escalar, ele utiliza uma filtragem em cascata aplicando uma função de diferença Gaussiana (DOG), procurando a detecção de pontos- chave, que examina os pormenores de candidatos locais em várias escalas, identificando-os desta forma. A escolha do SIFT vem do fato deste ser um algoritmo padrão para correspondência de pontos, e sua principal vantagem é produzir um espaço de escala, para obter a invariância dos pontos de interesse em termo de escala.

O algoritmo SURF segue os mesmos passos do SIFT, que utiliza a diferença das Gaussianas para a obtenção dos descritores em um espaço escalar; porém este se baseia na matriz Hessiana para a obtenção dos seus descritores no espaço escalar. SURF combina um detector Hessiano-Laplace com um filtro próprio, formando um descritor de característica que segue a estratégia do SIFT; porém, ao invés de construir um histograma baseado nas orientações dos gradientes da vizinhança como é feito no SIFT, ele calcula um somatório estatístico que resulta em um descritor de dimensão 64. A escolha do SURF é ser este também um padrão em termos de correspondência e por ter uma precisão maior em relação ao SIFT (BAY et al., 2006).

Todos esses algoritmos (ASIFT, SIFT e SURF) são algoritmos considerados de baixo nível, pois procuram descrever as características locais de uma imagem. Muitos trabalhos de alto nível utilizam os algoritmos de correspondência de pontos como apoio

para fazer a correspondência de seus objetos (GRAUMAN; LEIBE, 2011). A ideia do algoritmo proposto neste trabalho é executar a correspondência entre imagens a partir de características consideradas globais. O nosso algoritmo possui uma etapa global de segmentação dada pelo algoritmo de segmentação de imagens baseado em grafo, como foi discutido na subseção 3.2.2 deste trabalho. Escolhemos utilizar este tipo de algoritmo pois ele consegue produzir regiões consideradas grosseiras (sem muitos detalhes). Tais regiões em muito casos se aproximam dos objetos reais, como pode ser observado no apêndice A em que mostramos alguns resultados visuais da correspondência do MCICP.

O algoritmo ICP é bastante eficiente na formação de imagens 3D. Ele é em geral utilizado como gerador de superfícies tridimensionais encaixando as nuvens de pontos como em um quebra-cabeça, ou seja, alinhando as nuvens de pontos determinando uma pose final entre as nuvens. Dessa forma, o ICP consegue gerar a superfície dos objetos tridimensionais quando esses são observados a partir de vários pontos de vista. Como o ICP, além de determinar a pose, também gera o erro entre as distâncias dos pontos das nuvens; vimos que esta métrica poderia servir como um limiar de correspondência entre as regiões formadas pela segmentação, mas para que isso pudesse acontecer, teríamos que capturar, na cena, a imagem bidimensional mais a sua profundidade. Resolvemos esta parte utilizando o Kinect para esse fim.

Em uma análise geral dos resultados da precisão da cena 1, o resultado de precisão mostrado pela figura 33, para a latitude de 0^0 e longitudes variando de 15^0 até 90^0 (capítulo 6), nos mostrou que o nosso algoritmo MCICP é melhor que todos os algoritmos, pois mantém uma média de precisão em 90%, enquanto que os outros algoritmos iniciam com uma precisão acima de 50% mas tem essa precisão reduzida rapidamente com o aumento da longitude.

Continuando a análise da cena 1, o resultado de precisão visto na figura 34 do capítulo 6, para a latitude de 0^0 e longitudes variando de -15^0 até -90^0, nos mostrou que o SURF manteve uma superioridade em precisão até 60^0 reduzindo sua precisão rapidamente após este ponto, mas o MCICP manteve uma média superior a $\sim 76\%$ contra $\sim 67\%$ do SURF. Além disso percebemos no gráfico da figura 34 a mesma reação do gráfico anterior em que o MCICP varia, porém consegue manter uma média de precisão até o final da longitude, que é de 90^0, sendo que os outros algoritmos caem rapidamente.

Observando o gráfico da figura 37, que é uma fusão dos gráficos das figuras 35 e 36 para cena 1, em que temos um aumento da latitude que agora é de 30^0 e longitudes variando de -90^0 até 90^0, observa-se que, mais uma vez, o resultado segue a mesma reação dos resultados anteriores para o MCICP, como também a mesma reação para o resultado dos algoritmos de pontos aqui estudados, onde a média do MCICP girou em torno de $\sim 74\%$.

O resultado de precisão mostrado na figura 38 para cena 1, em que há um aumento

da latitude agora tendo o valor de 60^0 e longitudes variando de -90^0,-45^0,0^0,45^0 e 90^0, mais uma vez, como nos resultados anteriores, revela certa constante da precisão do MCICP em torno de $\sim 70\%$. E, mais uma vez, a mesma reação dos algoritmos ASIFT e SIFT, caindo quando da mudança em latitude e longitude; nesse caso o SURF nem sequer pontua para essa latitude de 60^0. Dessa forma vimos que o MCICP superou em precisão todos os algoritmos de ponto em todos os gráficos.

A redução em precisão dos algoritmos de pontos, em relação às variações de longitude e latitude já era esperada, pois os resultados análogos são citados na maioria dos artigos aqui referenciados. Uma das causas disso, é que uma figura plana em uma imagem sempre sofre uma transformação afim quando há variação em latitude e longitude em uma cena, deformando-se na perspectiva da cena e desta forma tem-se um aumento de erros na correspondência dos pontos.

No caso do MCICP, cada ponto de uma imagem possui sua profundidade na cena, quando há transformação em latitude e longitude, o ponto acompanha a perspectiva da cena, ele independe do objeto a que pertence podendo ser ele de um objeto plano ou de um objeto volumétrico na cena.

Além disso, os pontos, conforme a variação em latitude e longitude, ficam oclusos por outros pontos da cena, sendo que isto é válido para todos os algoritmos. Logo houve um maior número de erros quanto maior a variação.

O resultado da figura 32 mostra a precisão versus a revocação da cena 1 somente com o nosso algoritmo MCICP, porque foram utilizadas somente figuras volumétricas na cena, as quais dificultam a produção de uma imagem sintética com pontos correspondentes, para que possamos obter o número de verdadeiros pontos dos algoritmos de pontos ASIFT, SIFT e SURF. Podemos observar neste gráfico, que os valores variam bruscamente, isso se deve ao fato que as regiões comparadas na correspondência formam uma amostra pequena. Neste gráfico (figura 32) encontramos uma média de $revocação = 0,33$ e uma média de $precisão = 0,75$ em que o resultado é um valor de $F_1 = 0,46$. Sendo assim podemos então dizer que o nosso algoritmo tem um valor médio em torno de 46%, tanto para a revocação quanto para a precisão para o algoritmo MCICP.

Na apresentação do resultado da cena 2, (capítulo 6), onde há somente um objeto plano, conforme a figura 39, o melhor resultado quanto à precisão versus revocação foi o do algoritmo SIFT. O SIFT obteve uma média de $revocação = 0,47$ e uma média de $precisão = 0,81$ com um valor de $F_1 = 0,60$, possuindo um valor médio em torno de 60% tanto para a revocação quanto para a precisão. No caso do MCICP, tem-se uma média de $revocação = 0,33$ e uma média de $precisão = 0,41$ o que resulta em um valor de $F_1 = 0,36$. Sendo assim podemos então dizer que o algoritmo MCICP possui um valor médio em torno de 36% tanto para a revocação quanto para a precisão.

Na discussão do resultado da cena 3, o algoritmo SIFT teve como resultado uma média de $revocação = 0,20$ e uma média de $precisão = 0,27$ e um valor de $F_1 = 0,23$. Novamente podemos então dizer que o algoritmo SIFT possui um valor médio em torno de 23% tanto para a revocação quanto para a precisão. Por último encontramos os valores para o MCICP, que obteve uma média de $revocação = 0,32$ e uma média de $precisão = 0,50$ que resulta em um valor de $F_1 = 0,39$. Sendo assim podemos então dizer que o algoritmo MCICP possui um valor médio em torno de 39% tanto para a revocação quanto para a precisão, mostrando um melhor resultado balanceado entre Precisão versus Revocação. O fato agora é que com a entrada de objetos volumétricos houve uma maior deformação na transformação afim das imagens dessa cena e, em consequência, um maior número de erros de correspondência para os algoritmos de ponto.

8 CONCLUSÕES E TRABALHOS FUTUROS

O objetivo desta dissertação foi desenvolver um algoritmo utilizando o ICP capaz de executar a correspondência entre as regiões de imagens de uma mesma cena observada a partir de pontos de vista distintos. A literatura apresenta, com bastante ênfase, algoritmos capazes de corresponder pontos em imagens: ASIFT; SURF; SIFT. Quando se trata de imagens que sofrem variação em latitude e longitude, a correspondência de pontos sofre uma perda em precisão e revocação.

Nesta dissertação foi apresentado o algoritmo ICP que, embora não tivesse essa funcionalidade específica, conseguiu efetivar a correspondência em regiões semelhantes nas diversas cenas apresentadas, em várias poses, funcionando desta forma como um algoritmo de correspondência.

Foram estudadas aqui várias ferramentas de correspondência e suas particularidades.

Em relação ao SURF, destaca-se o seguinte resultado: na cena 2 apresentou 88% de precisão e 10% de revocação e na cena 3, 65% de precisão e 5% de revocação. Analisando o resultado do SURF, ele apresentou uma precisão e revocação maior na cena 2. Em relção ao ASIFT, destaca-se o seguinte resultado: na cena 2 apresentou 32% de precisão e 6% de revocação e na cena 3, 29% de precisão e 38% de revocação. Analisando o resultado do ASIFT, ele apresentou uma precisão maior na cena 2 e uma maior revocão na cena 3. Em relação ao SIFT, destaca-se o seguinte resultado: na cena 2 apresentou 81% de precisão e 147% de revocação e na cena 3, 27% de precisão e 20% de revocação. Analisando o resultado do SIFT, ele apresentou uma precisão e revocação maior na cena 2.

Analisando o resultado do MCICP, destaca-se: na cena 1 apresentou 75% de precisão e 33% de revocação; na cena 2, 41% de precisão e 33% de revocação e na cena 3, 50% de precisão e 32% de revocação. Analisando o resultado do MCICP, ele apresentou uma maior precisão na cena 1 e manteve aproximadamente a mesma revocação nas três cenas.

O ICP, algoritmo que compõe nosso modelo MCICP, foi utilizado como algoritmo de correspondência de regiões; utilizando como métrica de comparação, o erro da pose entre as nuvens de pontos. Uma das contribuições deste trabalho foi confirmar que o erro dado pelo ICP pode servir como métrica de comparação.

Foi proposto, como teste empírico, a aplicação dos algoritmos na correspondência de pontos e imagens das cenas apresentadas no arranjo experimental no capítulo 5. As cenas foram produzidas pelo autor através de fotografias de objetos planos (objetos que possuem superfícies planas) e volumétricos (objetos que não possuem superfícies planas), organizados numa sequência de poses, diferenciando em latitude e longitude do ponto de observação, sem análise de alteração da distância, entre a imagem e o objeto de referência.

Utilizou-se, para a produção das imagens, o sensor *Kinect*. Para o ajuste das imagens, foi utilizada a calibração, normalizando a imagem de profundidade com a imagem RGB. A segmentação foi utilizada para a produção das regiões relevantes. Na sequência, a fim de corresponder tais regiões, utilizou-se o algoritmo Iterative Closest Point (ICP) como comparador, no modelo proposto MCICP. Os resultados são comparados a uma tabela padrão ouro, a partir do qual pode-se comparar os dados do nosso algoritmo MCICP com as verdadeiras correspondências, e assim, produzir resultados de precisão e revocação.

Para os algoritmos de pontos, foram utilizados programas manuais ou automáticos para produzir uma tabela padrão ouro. Esses programas foram desenvolvidos pelo autor para a produção de imagens sintéticas que servem para a obtenção dos verdadeiros pontos de imagens que serão comparadas com os resultados da correspondência dos algoritmos de pontos.

A pesquisa apresentou a seguinte tabela 8

Tabela 7 – Resultado Geral das Cenas

Cena 1		
MCICP		
$revocação = 33\%$	$precisão = 75\%$	$F_1 = 46\%$
Cena 2		
MCICP		
$revocação = 33\%$	$precisão = 41\%$	$F_1 = 36\%$
ASIFT		
$revocação = 6\%$	$precisão = 32\%$	$F_1 = 10\%$
SIFT		
$revocação = 47\%$	$precisão = 81\%$	$F_1 = 60\%$
SURF		
$revocação = 10\%$	$precisão = 88\%$	$F_1 = 17\%$
Cena 3		
MCICP		
$revocação = 32\%$	$precisão = 50\%$	$F_1 = 39\%$
ASIFT		
$revocação = 38\%$	$precisão = 29\%$	$F_1 = 7\%$
SIFT		
$revocação = 20\%$	$precisão = 27\%$	$F_1 = 23\%$
SURF		
$revocação = 5\%$	$precisão = 65\%$	$F_1 = 9\%$

Fonte: Produzido pelo próprio autor.

Como observado no Capitulo 6 não pudemos obter os valores de revocação dos algoritmos ASIFT, SIFT e SURF para a cena 1.

Diante disso, analisando todos os resultados, pode-se concluir que ocorreu correspondência de regiões semelhantes entre imagens distintas através do algoritmo desenvolvido MCICP.

O ponto fraco do MCICP está relacionado ao seu sistema de busca, pois executa uma busca exaustiva entre as regiões a serem comparadas, isso representará uma demora na resposta do sistema. Além disso, para regiões que possuem um grande número de pontos nas nuvens, o ICP leva um tempo maior em retornar um resultado na busca da pose final.

Propomos então as seguintes melhorias para o MCICP, a serem abordadas em trabalhos futuros: para a melhoria da busca exaustiva, será preciso refazer o sistema de comparação das regiões de imagens, utilizando outra forma de busca, por exemplo, pode-se utilizar a árvore $k - d$ para reduzir o número de comparações. Podemos também utilizar filtros para suavizar o ruído das nuvens, para uma melhor precisão da correspondência entre elas. Para a melhora do tempo de comparação do ICP, será preciso diminuir o número de pontos das nuvens a ser comparadas, sem perder o formato das mesmas. Outra melhoria proposta, pode ser feita no arranjo experimental, aumentando a resolução das fotos utilizadas, bem como utilizando cenas tridimensionais externas, através de uma câmera stereo. Pode-se também incluir descritores em relação a cor das nuvens de pontos.

REFERÊNCIAS

ARUN, K. S.; HUANG, T. S.; BLOSTEIN,S. D. Least-squares fitting of two 3-d point sets. **IEEE transactions on pattern analysis and machine intelligence**, Urbana, v. 9, n. 5, p. 698-700, 1987.

BAY, H.; TUYTELAARS, T.; VAN GOOL, L. SURF: Speeded-Up Robust Features In: EUROPEAN CONFERENCE ON COMPUTER VISION, 9, 3951., 2008, Graz. **Proceedings...** Berlin: Springer-Verlag, 2006. p. 404-417.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. **Communications of the ACM**, New York, v. 18, n. 9, p. 509-517, 1975.

BESL, P.; MCKAY, N. A Method for Registration of 3-D Shapes. **IEEE transactions on pattern analysis and machine intelligence**, Washington, v.14, n. 2, p. 239-256, 1992.

BEAUDET, P. Rotationally invariant image operators In: INTERNATIONAL JOINT CONFERENCE ON PATTERN RECOGNITION, 4, 1166., 1978, Kyoto. **Proceedings...** Long Beach, Calif.: IEEE Computer Society, 1979. p. 579-583.

BOWYER, A. Computing Dirichlet tessellations. **The Computer Journal**, London, v. 24, n. 2. p. 162-166, 1981.

BROWN, M.; LOWE, D. Invariant features from interest point groups. In: THE BRITISH MACHINE VISION CONFERENCE, 2002, London. **Proceedings...** London: BMVA, 2002. p. 656-665.

CHEN, Y.; MEDIONI, G. Object Modeling by Registration of Multiple Range Images. In: IEEE CONF. ON ROBOTICS AND AUTOMATION, 3., 1991, Sacramento. **Proceedings...** New York: IEEE, 1991. p. 2724-2729.

CORKE, P. **Robotics, Vision and Control: Fundamental Algorithms in Matlab**. Berlin: Springer, 2011.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L. **Introduction to Algorithms**. Cambridge: The MIT Press, 1990.

FAUGERAS, O.; HEBERT, M. The Representation, Recognition, and Locating of 3-D Objects. **The International Journal of Robotic Research**, Utah, v. 5, n. 3, p. 27-52, 1986.

FELZENSZWALB, P. F.; HUTTENLOCHER, D. P. Efficient Graph-Based Image Segmentation. **International Journal of Computer Vision**, Berlin, v. 59, n. 2, p. 167-181, 2004.

FÖRSTNER, W.; GÜLCH, E. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In: ISPRS INTERCOMMISSION, 1987, Interlaken, **Workshop...** Interlaken: ISPRS, 1987. p. 149–155.

FRIEDMAN, J. H.; BENTLEY, J. L.; FINKEL, R. A. An algorithm for finding best matches in logarithmic expected time. **ACM Transactions on Mathematical Software (TOMS)**, New York, v. 3, n. 3, p. 209-226, 1977.

FUSIELLO, A.; TRUCCO, E.; VERRI, A. A compact algorithm for rectification of stereo pairs. **Machine Vision and Applications**, Berlin, v. 12, n. 1, p. 16-22, 2000.

GODIN, G.; RIOUX, M.; BARIBEAU, R. Three-dimensional registration using range and intensity information. **Proceedings of the SPIE - The International Society for Optical Engineering**, Boston, v. 2350, n. 2350, p. 279-290, 1994.

GRAUMAN, K.; LEIBE, B. Visual Object Recognition. **Synthesis Lectures on Artificial Intelligence and Machine Learning**, San Francisco, v. 5, n. 2, p. 1-181, 2011.

HARRIS, C.; STEPHENS M. A combined corner and edge detector. **Procedings of the Alvey Vision Conference**, Manchester, p. 147-151, 1988.

HERRERA C. D.; KANNALA, J.; HEIKKILÄ, J. Joint depth and color Câmera calibration with distortion correction. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, New Your, v. 34, n. 10, p. 2058-2064, 2012.

HORN, B. Closed-Form Solution of Absolute Orientation Using Unit Quaternions. **Journal of the Optical Society of America A**, v. 4, n. 4, p. 629-642, 1987.

KHOSHELHAM, K; ELBERINK, S. O. Accuracy and resolution of kinect depth data for indoor mapping applications. **Sensors**, v. 12, n. 2, p. 1437-1454, 2012.

KJER, H. M.; WILM, J. **Evaluation of surface registration algorithms for PET motion correction**, 2010. 87 f. Thesis (PhD in Informatics and Mathematical Modelling) - Technical University of Denmark, University of Denmark, 2010.

KOENDERINK, J. J. The structure of images. **Biological Cybernetics**, Berlin, v. 50, n. 5, p. 363-396, 1984.

LINDEBERG, T. Scale-space theory: A basic tool for analysing structures at different scales. **Journal of Applied Statistics**, Abingdon, v. 21, n. 2, p. 224-270, 1994.

LINDEBERG. T. Feature detection with automatic scale selection. **International Journal of Computer Vision**, Berlin, v. 30, n. 2, p. 79-116, 1998.

LOWE, D. G. Object recognition from local scale-invariant features. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 7, v. 2, 1999, Corfu. **Proceedings...** New York: IEEE, 1999. p. 1150-1157.

LOWE, D. G. Distinctive image features from scale-invariant key points. **International Journal of Computer Vision**, Berlin, v. 60, n. 2, p. 91-110, 2004.

MANKOFF, K. D.; RUSSO, T. A. The Kinect: A low-cost, high-resolution, short range 3d Camera. **Earth Surface Processes and Landforms**, Lausanne, v. 38, n. 9, p. 926-936, 2013.

MASUDA, T. et al. Registration and integration of multiple range images for 3D model construction. INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 13, v. 1, n. 1, 1996, Washington. **Proceedings...**, p. 879-883.

MATAS, J. et al. Robust wide baseline stereo from maximally stable extremal regions. In: BRITISH MACHINE VISION CONFERENCE, 13, 2002, Cardiff. **Conference...**, Cardiff: BMVC, 2002. p. 384-393.

MIKOLAJCZYK, K.; SCHMID, C. Indexing based on scale invariant interest points. IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, v. 1, n. 0, 2001, Los Alamitos. **Proceedings...**, p. 525-531.

MIKOLAJCZYK, K. **Detection of local features invariant to affine transformations Application to matching and recognition**. 2002. 171 f. Thesis (Ph. D. in Imaging Vision and Robotics) - Imaging Vision and Robotics, Institut National Polytechnique de Grenoble, France.

MIKOLAJCZYK, K.; SCHMID, C. Scale and affine invariant interest point detectors. **International Journal of Computer Vision**, Berlin, v. 60, n. 1, p. 63-86, 2004.

MIKOLAJCZYK, K.; SCHMID, C. A performance evaluation of local descriptors. **IEEE transactions on pattern analysis and machine intelligence**, New Youk, v. 27, n. 10, p. 1615-1630, 2005.

MIKOLAJCZYK, K. et al. A comparison of affine region detectors. **International Journal of Computer Vision**, Berlin, v. 65, n. 1-2, p. 43-72, 2005.

MORAVEC, H. The Stanford cart and the CMU rover. **Proceedings of the IEEE**, v. 71, n. 7, p. 872-884, 1983.

MOREL, J. M.; YU, G. On the Consistency of the SIFT method; **Thechnical Report, CMLA, ENS Cachan**, Cachan, 2008.

MOREL, J. M.; YU, G. ASIFT: A New Framework for Fully Affine Invariant Image Comparison. **SIAM Journal on Imaging Sciences**, Philadelphia, v. 2, n. 2, p. 438-469, 2009.

OSWALD, L. Recent development of the Iterative Closest Point algorithm **Autonomous systems lab, Swiss Federal Institute** p. 39, 2010.

PANG, Y.; LI, W.; YUAN, Y.; PAN, J. Fully affine invariant SURF for image matching. **Neurocomputing**, v. 85, n. 0, p.6-10, 2012.

POWERS, D. M. W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation. **Journal of Machine Learning Technologies**, v. 2, n. 1, p. 37-63, 2011.

RUSINKIEWICZ, S.; LEVOY, M. Effcient variants of the ICP algorithm. 3-D DIGITAL IMAGING AND MODELING, 2001. INTERNATIONAL CONFERENCE ON. IEEE, 3, 2001, **Proceedings...**, p. 145-152.

SIMON D. A. **Fast and Accurate Shape-Based Registration**, 1996. 196 f. Theses (Ph. Doctoral), Carnegie Mellon University, 1996.

TUYTELAARS, T.; MIKOLAJCZYK, K. Local invariant feature detectors: a survey. **Foundations and Trends in Computer Graphics and Vision**, v. 3, n. 3, p. 177-280, 2008.

ZIRARI, F. et al. A graph based approach for heterogeneous document segmentation. **Image and Signal Processing**, p. 424-431, 2012.

WATSON, D. F. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. **Computer Journal**, v. 24, n. 2, p.167-172, 1981.

WILSON, H. R.; GIESE, S. C. Threshold visibility of frequency gradient patterns. **Vision Research**, v. 17, p. 1177-1190, 1977.

WITKIN, A. P. Scale-space filtering. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2., 1983, Karlsruhe. **Conference...** Berlin: Morgan Kaufmann Publishers Inc., 1983. p. 1019-1022.

APÊNDICE A – RESULTADO VISUAL DAS REGIÕES CORRESPONDENTES

Aqui está visualmente o resultado parcial da correspondência.

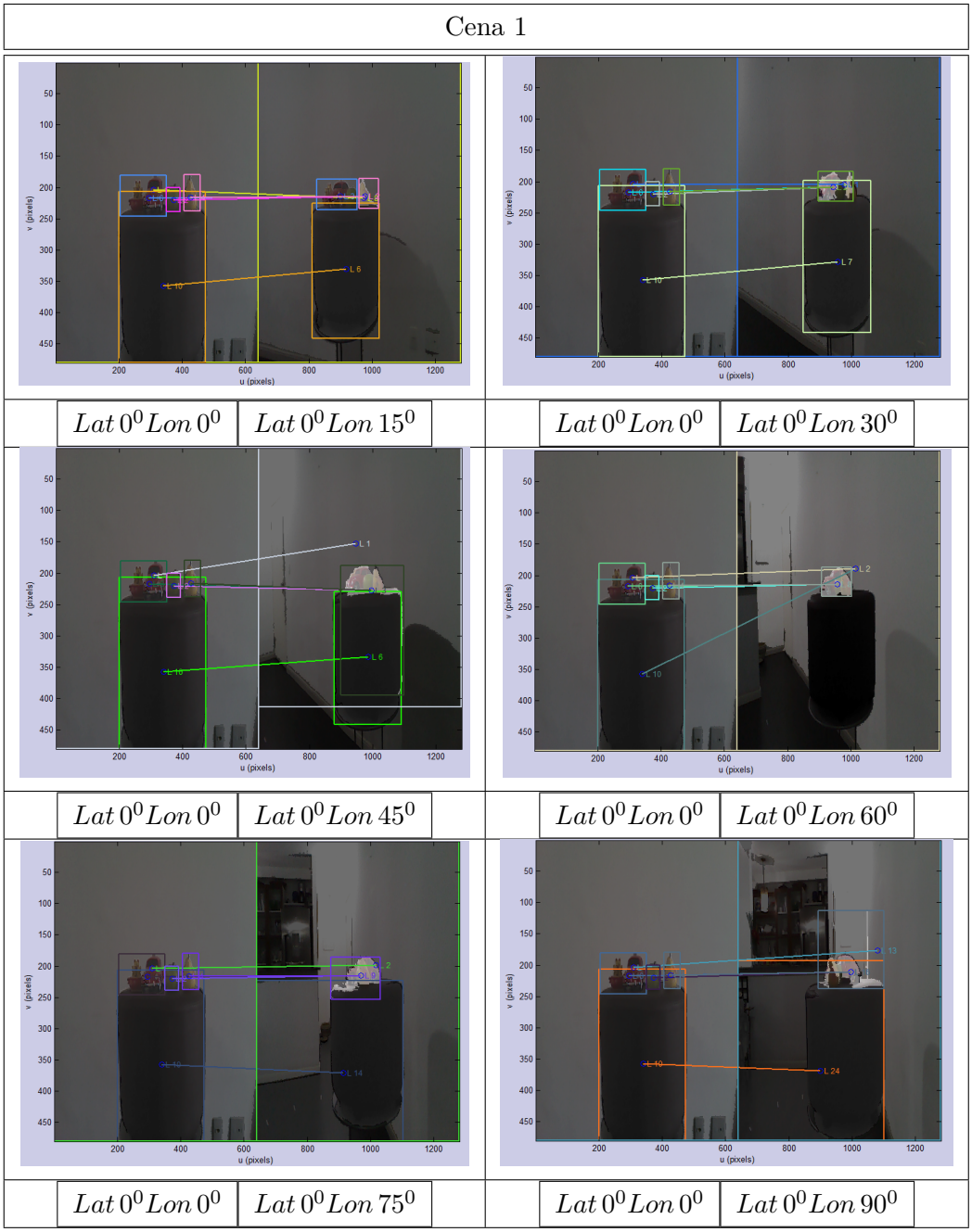


Figura 41 – Cena 1 resultados da correspondência na variação de 0^0 até 90^0.
Fonte: Produzido pelo próprio autor.

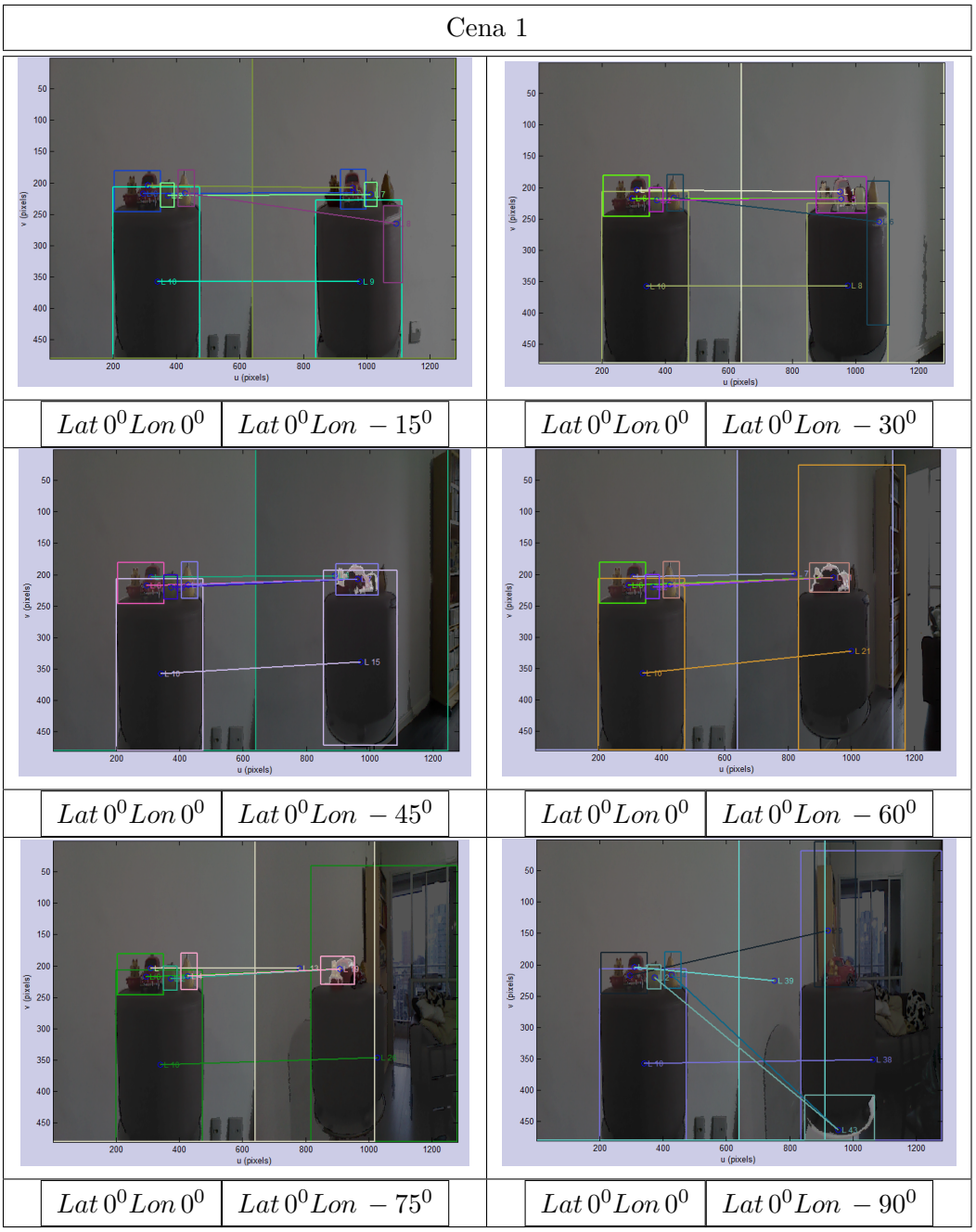


Figura 42 – Cena 1 resultados da correspondência na variação de 0^0 até -90^0.
Fonte: Produzido pelo próprio autor.

APÊNDICE B – KINECT

O sensor Kinect XBOX 360 utilizado nesta dissertação, consiste de um emissor de laser infravermelho, uma câmara de infravermelho e uma câmera RGB. Onde a imagem da câmera RGB e a câmera de infravermelho possui uma resolução de 640 x 480 pixels.

A calibração foi feita utilizando o programa kinect toolbox de (HERRERA et al., 2012).

A Matriz de Calibração final possui os seguintes parâmetro da tabela 9:

Tabela 8 – Parâmetros da matriz de calibração

Parâmetros de Calibração		Câmera RGB	Sensor IR
Foco	f_x	1060.27± 0.89 [pixels]	579.83 ± 0.51 [pixels]
	f_y	1053.03± 0.88 [pixels]	586.73 ± 0.50 [pixels]
Ponto Principal	x_0	619.70± 0.70 [pixels]	321.55 ± 0.42 [pixels]
	y_0	513.69 ± 0.62 [pixels]	235.01 ± 0.42 [pixels]
Dimensão do Frame	w	5.996 ± 0.001 [mm]	6.012 ± 0.002 [mm]
	h	4.5 [mm]	4.5 [mm]
Tamanho do pixel	p_x	9.3 [±mm]	9.3 [±mm]
	p_y	9.3 [±mm]	9.3 [±mm]
Distorção Radial simétrica	K_1	0.2150 ± 0.0029	0
	K_2	-0.6704 ± 0.0140	0
	K_3	-0.0038 ± 0.0002	0
Distorção descentrada	P_1	0.0012 ± 0.0002	0
	P_2	0.6422 ± 0.0187	0
Pose Relativa	$Rotação$	1.00000 0.00000 0.00000 0.00000 1.00000 0.00000 0.00000 0.00000 1.00000	1.00000 0.00066 0.00260 −0.00063 0.99990 −0.01397 −0.00261 0.01397 0.99990
	$Translação$	0.00000 0.00000 0.00000	−0.02308 0.00481 −0.00186

Fonte: Produzido pelo próprio autor.

Cálculo da profundidade por triangulação

A medição da profundidade como um processo de triangulação, utiliza uma fonte laser que emite um único feixe. Esse feixe é dividido em múltiplos feixes por uma rede de difração para criar um padrão constante de manchas projetadas na cena. Esse padrão é capturado pela câmera infravermelha e está correlacionada com outro padrão de referência.

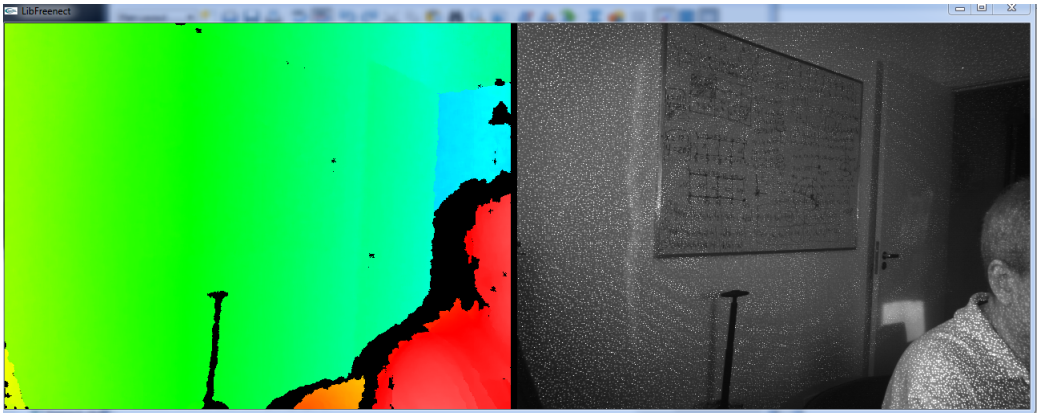


Figura 43 – Disparidade obtida atraves de um padrão de manchas.
Fonte: Produzido pelo próprio autor.

O padrão de referência é obtido pela captura de um plano a uma distância conhecida a partir do sensor e é armazenado na memória do sensor.

Quando uma mancha é projetada sobre um objeto fora do plano de referência do kinect, essa mancha é deslocada na direção da linha de base entre o projetor de laser e o centro de perspectiva do sensor infravermelho do kinect. A imagem de disparidade é a medida do deslocamento das manchas, através de um procedimento de correlação simples da distância. A figura 43 ilustra o resultado da medição da disparidade à partir da configuração de manchas.

A figura 44 ilustra a relação entre a distância de um ponto k, um plano de referência e a disparidade d. As coordenadas $3D$ dos pontos do objeto formam um sistema de coordenadas de profundidade, com sua origem no centro do sensor infravermelho. O eixo$Z$ é ortogonal ao plano da imagem, o eixo X é perpendicular ao eixo Z na direção da linha de base b entre o centro do sensor infravermelho e o projetor de laser, e o eixo Y é ortogonal ao X e Z, apresentando um sistema de coordenadas.

Assumindo um plano de referência a uma distância Z_0 em relação ao sensor infravermelho, uma mancha no plano do objeto é projetada sobre o plano de imagem da câmara de infravermelhos. Se essa mancha é deslocada em relação ao sensor, a localização da mancha no plano de imagem será deslocada também na direção do eixo X. Esta medida é conhecida como disparidade d.

A partir da semelhança de triângulos temos:

$$\frac{D}{b} = \frac{Z_0 - Z_k}{Z_0} \tag{32}$$

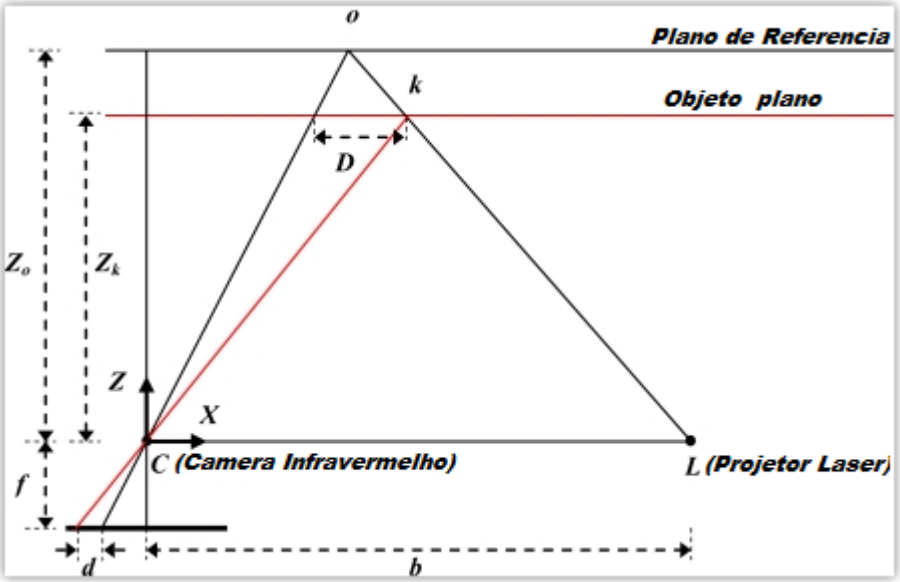


Figura 44 – Figura da relação entre as distâncias do padrão de referência de manchas e um ponto k no objeto, para o cálculo da disparidade.
Fonte: Autor "adaptado de" (KHOSHELHAM; ELBERINK, 2012)

e:

$$\frac{d}{f} = \frac{D}{Z_k}, \tag{33}$$

em que Z_k indica a distância (profundidade) do ponto k do objeto, b é o comprimento da base, f é a distância focal da câmara de infravermelhos, D é o deslocamento do ponto k do objeto, e d é a disparidade observada na imagem. Substituindo D a partir da equação (33) na equação (32) e expressa em termos de Z_k temos:

$$Z_k = \frac{Z_0}{1 + \frac{Z_0}{fb}d}. \tag{34}$$

A equação (34) representa o modelo matemático para a determinação da profundidade da disparidade, desde que os parâmetros sejam constantes Z_0, f, e b possam ser determinados pela calibração (anexo F).

ANEXO A – DETECTOR HESSIANO

O detector Hessiano (BEAUDET, 1978) procura locais na imagem que exibam forte derivadas em duas direções ortogonais. É baseado em uma matriz que contém as segundas derivadas, que é conhecida como Hessiana. Como os operadores de derivadas são sensíveis ao ruído, sempre é usado um operador de derivada que combina a Gaussiana com o parâmetro de desvio σ que irá desfocar a imagem eliminando o ruído.

$$H(x,\sigma) = \begin{bmatrix} I_{xx}(x,\sigma) & I_{xy}(x,\sigma) \\ I_{xy}(x,\sigma) & I_{yy}(x,\sigma) \end{bmatrix} \tag{35}$$

O detector computa as segundas derivadas $I_{xx}(x,\sigma)$, $I_{xy}(x,\sigma)$ e $I_{yy}(x,\sigma)$ para cada ponto da imagem em seguida ele procura por pontos onde o determinante do Hessiano é máximo:

$$det(H) = I_{xx}I_{yy} - I_{xy}^2 \tag{36}$$

Após obter os valores dos determinantes da Hessiana, em seguida, aplica-se a supressão não máxima utilizando uma janela de 3×3. Neste procedimento, a janela de pesquisa varre toda a imagem, mantendo apenas os pixels cujo valor é maior do que os valores de todos os oito vizinhos imediatos no interior da janela. O detector em seguida, retorna todos os locais restantes, cujo valor é superior a um limiar θ. Como mostrado na figura 45 (superior esquerdo), as respostas dos detectores resultantes estão localizados principalmente em cantos e em áreas de imagem fortemente texturizadas.

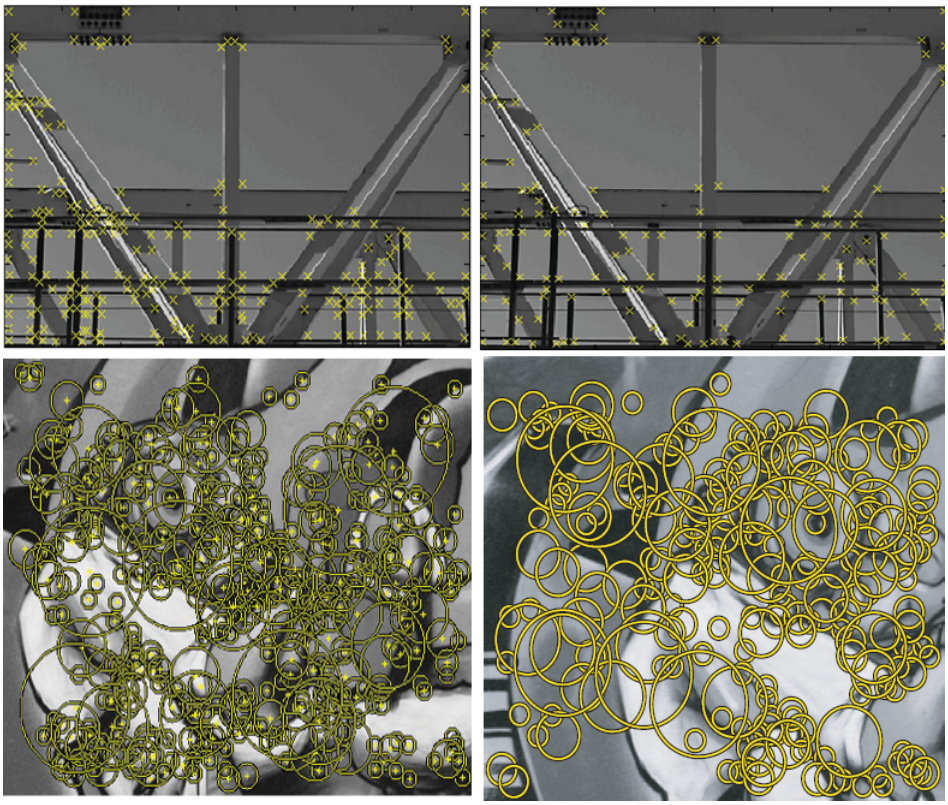


Figura 45 – Exemplo de resultado do detector Hessiano (esquerda acima); (direita acima) detector Harris; (esquerda abaixo) detector Laplaciano Gaussiano; (direita abaixo) detector Diferença Gaussiana.
Fonte: Autor “adaptado de” (TUYTELAARS et al., 2008)

ANEXO B – DETECTOR HARRIS

O detector Harris [Forstner and Gulch, 1987, Harris and Stephens, 1988] foi explicitamente preparado para se ter uma estabilidade geométrica. Este define os pontos-chaves como "pontos que possuem uma máxima local com precisão de auto correspondência quando se transladam em um modelo de correspondência dos mínimos quadrados" [Triggs, 2004]. Na pratica, estes pontos-chaves sempre correspondem a uma estrutura com formato de esquina. O processo de detecção é visualizado na figura 46. O detector Harris procura por pontos x onde o segundo momento da matriz C ao redor de x tem 2 grandes auto-valores. A matriz C pode ser processada obtendo a primeira derivada deu uma janela em torno de x, aplicando um peso por uma Gaussiana $G(x, \tilde{\sigma})$:

$$C(x, \sigma, \tilde{\sigma}) = G(x, \tilde{\sigma}) \star \begin{bmatrix} I_x^2(x, \sigma) & I_x I_y(x, \sigma) \\ I_x I_y(x, \sigma) & I_y^2(x, \sigma) \end{bmatrix} \tag{37}$$

Nesta formulação, a convolução com a Gaussiana $G(x, \tilde{\sigma})$ assume o papel de soma sobre todos os pixels em uma vizinhança local circular, onde cada pixel sofre uma contribuição adicional que é ponderada pela sua proximidade com o ponto central. Ao

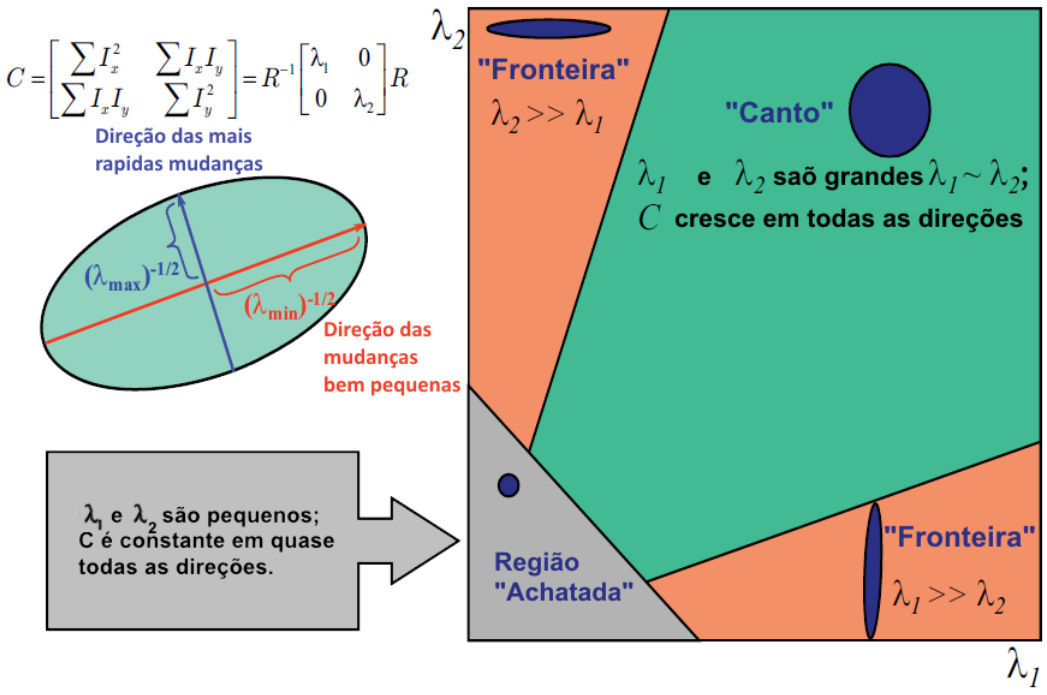


Figura 46 – O detector Harris procura por vizinhança em que o segundo momento da matriz C contenha dois grandes auto valores, correspondendo a duas orientações dominantes que formam uma elipse. Os pontos resultantes muitas vezes correspondem a cantos como estrutura.
Fonte: Autor "adaptado de" (GRAUMAN; LEIBE, 2011).

invés de explicitamente computar os autovalores de C, a equivalência seguinte é utilizada

$$det(C) = \lambda_1 \lambda_2 \tag{38}$$

$$trace(C) = \lambda_1 + \lambda_2 \tag{39}$$

para checar se sua razão $r = \frac{\lambda_1}{\lambda_2}$ está abaixo de certo limiar.

$$\frac{trace2(C)}{det(C)} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} = \frac{(r\lambda_2 + \lambda_2)^2}{r\lambda_2^2} = \frac{(r+1)^2}{r} \tag{40}$$

Com isto pode ser expresso pela seguinte condição,

$$det(C) - \alpha trace^2(C) > t \tag{41}$$

no qual não é preciso ter o cálculo exato do autovalor. Os valores típicos de α estão em um range de (0.04 - 0.06). O parâmetro $\tilde{\sigma}$ é usualmente igualado a 2σ, de modo que a vizinhança considerada é ligeiramente maior do que o suporte do operador derivativo utilizado.

ANEXO C – MINIMIZAÇÃO PONTO A PONTO

A seguir é descrito a minimização utilizando os mínimos quadrados para minimizar o alinhamento do erro. Deixe p_i e q_i denotar os pares de pontos de N correspondências. As normais correspondentes aos pontos do "modelo" são denominados $\vec{n_i}$. Os centroides são definidos como:

$$\bar{p} = \frac{1}{m}\sum p \; \bar{q} = \frac{1}{n}\sum q, \tag{42}$$

onde m e n são respectivamente a quantidade de pontos do "modelo" e quantidade de pontos de "dados". Os desvios dos pontos em relação ao centroide são dadas por:

$$p_i^{'} = p_i - \bar{p} \; q_i^{'} = q_i - \bar{q}. \tag{43}$$

A tarefa é achar a matriz $\mathbf{R}$ de rotação e o vetor $\mathbf{T}$ de translação que minimiza o erro

$$E = \sum_{i=1}^{N} ||\mathbf{R}p_i + \mathbf{T} - q_i||^2. \tag{44}$$

Usando as definições acima, isso pode ser reescrito como:

$$= \sum_{i=1}^{N} ||\mathbf{R}(p_i^{'} + \bar{p}) + \mathbf{T} - (q_i^{'} + \bar{q})||^2 = \sum_{i=1}^{N} ||\mathbf{R}p_i^{'} - q_i^{'} + (R\bar{p} - \bar{q} + \mathbf{T})||^2. \tag{45}$$

A fim de minimizar a métrica de erro, o vetor de translação $\mathbf{T}$ deve ser escolhido para mover o centroide rotacionado do "dados" para o centroide "modelo"

$$\mathbf{T} = \bar{q} - \mathbf{R}\bar{p}, \tag{46}$$

o qual simplifica a expressão do erro

$$E = \sum_{i=1}^{N} ||\mathbf{R}p_i^{'} - q_i^{'}||^2 = \mathbf{R}\mathbf{R}^T \sum_{i=1}^{N} ||p_i^{'}||^2 - 2tr\left(\mathbf{R}\sum_{i=1}^{N} p_i^{'} q_i^{'T}\right) + \sum_{i=1}^{N} ||q_i^{'}||^2 \tag{47}$$

$$= \sum_{i=1}^{N} ||p_i^{'}||^2 - 2tr\left(\mathbf{R}\sum_{i=1}^{N} p_i^{'} q_i^{'T}\right) + \sum_{i=1}^{N} ||q_i^{'}||^2. \tag{48}$$

Agora, faça $\mathbf{N} = \sum_{i=1}^{N} p_i^{'} q_i^{'T}$. Para minimizar o erro E o traço da matriz $tr(\mathbf{RN})$ tem que ser maximizado. Deixe as colunas de $\mathbf{N}$ e as fileiras de $\mathbf{R}$ ser denotado c_i e r_i, respectivamente, onde $i \in \{1, 2, 3\}$. O traço de $\mathbf{RN}$ pode ser expandido à

$$tr(\mathbf{RN}) = \sum_{i=1}^{3} r_i.c_i \leq \sum_{i=1}^{3} ||r_i|| \, ||c_i||, \tag{49}$$

onde a desigualdade é apenas uma reformulação da desigualdade de Cauchy-Schwarz. Uma vez que a matriz de rotação $\mathbf{R}$ é ortogonal por definição, toda a sua linha de vetores têm a unidade como comprimento. Isto implica

$$tr(\mathbf{RN}) \leq \sum_{i=1}^{3} \sqrt{c_i^T.c_i} = tr\left(tr\sqrt{\mathbf{N^T N}}\right). \tag{50}$$

Considere o valor singular decomposição de $\mathbf{N}$

$$\mathbf{N} = \mathbf{U\Sigma V^T}, \tag{51}$$

escolhendo o vetor de rotação

$$\mathbf{R} = \mathbf{VU^T}, \tag{52}$$

o traço do $\mathbf{RN}$torna-se:

$$tr(\mathbf{VU^T U\Sigma V^T}) = tr(\mathbf{V\Sigma V^{-1}}) = tr\left(\sqrt{\mathbf{V\Sigma^T \Sigma V^{-1}}}\right) = tr\left(\sqrt{\mathbf{N^T N}}\right), \tag{53}$$

que de acordo com (D. 9) é tão grande quanto possível.

ANEXO D – MINIMIZAÇÃO DO PONTO AO PLANO

No ponto de minimização do plano, o objetivo geral é o de trazer os pontos de "dados" próximos aos planos em que os pontos "modelo" correspondentes residem. Matematicamente, isso pode ser feito através da minimização dos produto escalar dos vetores $\overrightarrow{p_i q_i}$ e normais $\vec{n_i}$. A métrica de erro pode ser escrita como:

$$E = \sum_{i=1}^{N} [(Rp_i + \vec{T} - q_i).\vec{n_i}]^2. \tag{54}$$

A fim de resolver a equação analiticamente, a matriz de rotação deve ser linearizado como dada pela equação (61). Esta aproximação só faz sentido para pequenos ângulos α, como são esperados no final das iterações do algoritmo ICP.

Usando essa aproximação, a métrica do erro torna-se:

$$\begin{aligned} E = \sum_{i=1}^{N} [&(p_{i,x} - \gamma p_{i,y} + \beta p_{i,z} + T_x - q_{i,x}) n_{i,x} + \\ &(\gamma p_{i,x} + p_{i,y} - \alpha p_{i,z} + T_y - q_{i,y}) n_{i,y} + \\ &(-\beta p_{i,x} + \alpha p_{i,y} + p_{i,z} + T_z - q_{i,z}) n_{i,z}]^2 \end{aligned}, \tag{55}$$

$$\begin{aligned} E = \sum_{i=1}^{N} [&(p_i - q_i).\vec{n_i} + T_x.\vec{n_i} + \\ &\alpha(p_{i,y} n_{i,z} - p_{i,z} n_{i,y}) + . \\ &\beta(p_{i,z} n_{i,x} - p_{i,x} n_{i,z}) + \\ &\gamma(p_{i,x} n_{i,y} - p_{i,y} n_{i,x})]^2 \end{aligned} \tag{56}$$

Definindo $c_i = p_i \times \vec{n_i}$ e $\vec{r} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$ o erro torna-se:

$$E = \sum_{i=1}^{N} [(p_i - q_i).\vec{n_i} + \vec{T}.\vec{n_i} + \vec{r}.c_i]^2 \tag{57}$$

As derivadas parciais que tem relação aos seis graus de liberdade são:

$$\frac{\partial E}{\partial \alpha} = \sum_{i=1}^{N} 2c_{i,x}[(p_i - q_i).\vec{n_i} + T.\vec{n_i} + \vec{r}.c_i] = 0 \tag{58}$$

$$\frac{\partial E}{\partial \beta} = \sum_{i=1}^{N} 2c_{i,y}[(p_i - q_i).\vec{n_i} + T.\vec{n_i} + \vec{r}.c_i] = 0 \tag{59}$$

$$\frac{\partial E}{\partial \gamma} = \sum_{i=1}^{N} 2c_{i,z}[(p_i - q_i).\vec{n_i} + T.\vec{n_i} + \vec{r}.c_i] = 0 \tag{60}$$

$$\frac{\partial E}{\partial T_x} = \sum_{i=1}^{N} 2n_{i,x}[(p_i - q_i).\vec{n_i} + T.\vec{n_i} + \vec{r}.c_i] = 0 \tag{61}$$

$$\frac{\partial E}{\partial T_y} = \sum_{i=1}^{N} 2n_{i,y}[(p_i - q_i).\vec{n_i} + T.\vec{n_i} + \vec{r}.c_i] = 0 \tag{62}$$

$$\frac{\partial E}{\partial T_z} = \sum_{i=1}^{N} 2n_{i,z}[(p_i - q_i).\vec{n_i} + T.\vec{n_i} + \vec{r}.c_i] = 0. \tag{63}$$

Na forma de matriz, este sistema de equações torna-se:

$$\sum_{i=1}^{N} \begin{bmatrix} c_{i,x}c_{i,x} & c_{i,x}c_{i,y} & c_{i,x}c_{i,z} & c_{i,x}n_{i,x} & c_{i,x}n_{i,y} & c_{i,x}n_{i,z} \\ c_{i,y}c_{i,x} & c_{i,y}c_{i,y} & c_{i,y}c_{i,z} & c_{i,y}n_{i,x} & c_{i,y}n_{i,y} & c_{i,y}n_{i,z} \\ c_{i,z}c_{i,x} & c_{i,z}c_{i,y} & c_{i,z}c_{i,z} & c_{i,z}n_{i,x} & c_{i,z}n_{i,y} & c_{i,z}n_{i,z} \\ n_{i,x}c_{i,x} & n_{i,x}c_{i,y} & n_{i,x}c_{i,z} & n_{i,x}n_{i,x} & n_{i,x}n_{i,y} & n_{i,x}n_{i,z} \\ n_{i,y}c_{i,x} & n_{i,y}c_{i,y} & n_{i,y}c_{i,z} & n_{i,y}n_{i,x} & n_{i,y}n_{i,y} & n_{i,y}n_{i,z} \\ n_{i,z}c_{i,x} & n_{i,z}c_{i,y} & n_{i,z}c_{i,z} & n_{i,z}n_{i,x} & n_{i,z}n_{i,y} & n_{i,z}n_{i,z} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ T_x \\ T_y \\ T_z \end{bmatrix} \tag{64}$$

$$= -\sum_{i=1}^{N} \begin{bmatrix} c_{i,x}(p_i - q_i).\vec{n_i} \\ c_{i,y}(p_i - q_i).\vec{n_i} \\ c_{i,z}(p_i - q_i).\vec{n_i} \\ n_{i,x}(p_i - q_i).\vec{n_i} \\ n_{i,y}(p_i - q_i).\vec{n_i} \\ n_{i,z}(p_i - q_i).\vec{n_i} \end{bmatrix}, \tag{65}$$

que podem ser resolvidos por métodos padrões.

ANEXO E – SVD

O Valor singular de decomposição ou SVD de uma matriz A é

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}}, \tag{66}$$

onde $\mathbf{U} \in R^{m\times m}$ e $\mathbf{V} \in R^{n\times n}$ são ambas as matrizes ortogonais, compreendendo, como colunas, o vetores singulares correspondentes u_i e v_i. $\Sigma \in R^{m\times n}$ é uma matriz diagonal dos valores singulares

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & 0 & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & 0 & & & \ddots & \\ & & & & & 0 \end{bmatrix} \tag{67}$$

, onde $r = rank(A)$ é o rank de A. Para o caso em que $r < n$ a diagonal terá zero elementos, como mostrado. O número de condição de uma matriz A é $max\,\sigma i/min\,\sigma i$ e um valor alto significa que a matriz está perto de uma matriz singular ou "mal condicionada".

A forma quadrática matriz

$$s = x^T Ax \tag{68}$$

(D.4) é um escalar. Para o caso em que A é diagonal este pode ser escrita

$$s = \sum_{i=1}^{n} A_{ii}x_i^2 \tag{69}$$

que é a soma ponderada de quadrados. Se A é simétrica, em seguida,

$$s = \sum_{i=1}^{n} A_{ii}x_i^2 + 2\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}x_i x_j \tag{70}$$

o resultado também inclui produtos ou correlações entre os elementos de x.

Matrizes reais são um subconjunto de todas as matrizes. Para o caso geral de matrizes complexas o termo Hermitiano é o análogo de simétrico, e unitário do análogo de ortogonal. A^H denota a transposta Hermitiana, a transposta conjugada complexa da matriz complexa A.

Nós frequentemente precisamos resolver sistemas de equações lineares

$$Ax = b \tag{71}$$

onde $A \in R^{n \times m}$ e $b \in R^n$ são conhecidos, e $x \in R^m$ é desconhecida. Se $n = m$ então A é quadrada, e se A é não singular, então, a solução é obtida usando a matriz inversa

$$x = A^{-1}b \tag{72}$$

Se $n > m$ o sistema é mais constrangido, onde usamos o inverso pseudo

$$x = A^{+}b \tag{73}$$

que dá para x a minimização da norma do resíduo $|Ax - b|$. Usando SVD, onde $A = U\Sigma V^T$ isto é

$$x = V\Sigma^{-1}U^T b \tag{74}$$

onde agora Σ^{-1} é simplesmente o inverso que multiplica elemento a elemento dos não-zeros elementos diagonais da matriz Σ^T.

Se a matriz é singular, ou o sistema está sob limitação $n < m$, então há um número infinito de soluções. Nós podemos usar novamente a abordagem SVD, onde desta vez Σ^{-1} é o inverso que multiplica elemento a elemento dos não-zero elementos da diagonal de Σ, todos os outros zeros são deixados no local.

Decomposição em valores singulares também podem ser usados para estimar uma matriz de rotação dado um conjunto de vetores$\{(p_i, q_i), i = 1...N\}$ para que $q_i = Rp_i$. Nós primeiro calculamos a matriz de momento

$$M = \sum_{i=1}^{N} q_i p_i^T \tag{75}$$

e tomar em seguida, o calculo do SVD $M = U\Sigma V^T$. A estimativa mínimos quadrados da matriz de rotação é

$$R = UV^T \tag{76}$$

o que garante ser uma matriz ortogonal.

ANEXO F – MODELO DE CALIBRAÇÃO

Modelo Intrínseco da Câmera Colorida

Este modelo $\mathcal{L}_c = \{f_c, p_{0c}, k_c\}$ é similar ao modelo de (Heikkil) que consiste no modelo do "pequeno orifício" com correção da distorção radial e tangencial da imagem. A projeção de um ponto tridimensional com coordenadas $\mathbf{x}_c = [x_c, y_c, z_c]^T$ para o plano da imagem da câmera colorida $\mathrm{p}_c = [u_c, v_c]^T$ é obtido através das seguintes equações. O ponto é normalizado $\mathbf{x}_n = [x_n, y_n]^T = [x_c/z_c, y_c/z_c]^T$ em seguida a distorção é aplicada ao ponto:

$$x_g = \begin{bmatrix} 2k_3 x_n y_n + k_4(r^2 + 2x_n^2) \\ k_3(r^2 + 2y_n^2) + 2k_4 x_n y_n \end{bmatrix} \tag{77}$$

$$x_k = (1 + k_1 r^2 + k_2 r^4 + k_5 r^6) x_n + x_g, \tag{78}$$

onde $r^2 = x_n^2 + y_n^2$ e $k_c = [k_1, ..., k_5]$ é um vetor que contém os coeficientes da distorção da lente.

Finalmente as coordenadas da imagem são obtidas:

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} f_{cx} & 0 \\ 0 & f_{cy} \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} u_{0c} \\ v_{0c} \end{bmatrix}, \tag{79}$$

onde $f_c = [f_{cx}, f_{cy}]$ são os tamanhos focais e $p_{0c} = [u_{0c}, v_{0c}]$ é o ponto principal.

Modelo Intrínseco da Câmera de Profundidade

A transformação entre coordenadas tridimensional da câmera de profundidade $x_d = [x_d, y_d, z_d]^T$ e as coordenadas da câmera de profundidade da imagem $p_d = [u_d, v_d]^T$ segue um modelo semelhante ao utilizado para a câmera de cor.

ANEXO G – MODELO DE BUSCA ÁRVORE K-D

A fim de manter pontos em uma árvore k-d, um conjunto P é dividido pela mediana das coordenadas de todos os pontos. O ponto que corresponde à mediana torna-se a raiz da árvore. Em seguida, os dois subconjuntos resultantes são divididos com base na mediana de suas segundas coordenadas. O processo conhecido como particionamento binário do espaço, gera uma árvore binária equilibrada contendo todos os pontos de P.

A Figura 47 ilustra a criação de uma árvore k-d para pontos em duas dimensões. Para ilustrar o processo pode-se facilmente pensar numa prolongação em espaços de dimensões superiores. Em R^3, que é o caso desta dissertação, nuvens de pontos podem ser posteriormente divididos em dois grupos de igual tamanho por planos que são ortogonais ao eixo $x, y\, e\, z$.

A complexidade computacional de criar uma árvore k-d não é maior do que $O(kNlog_2N)$ (FRIEDMAN et al., 1977). Isto corresponde à triagem do ponto com relação a cada uma de suas coordenadas, que é k vezes por $O(Nlog_2N)$ operação de classificação.

Ao usar árvores k-d, o tempo de computação para encontrar vizinhos mais próximos, em métricas Euclidiana, pode ser reduzido drasticamente. Assume-se que uma árvore k-d foi construído contendo os pontos de conjunto de P. Localizando o vizinho mais próximo de q dentro P, o algoritmo pode ser feito da seguinte maneira.

a) Mover para baixo a árvore, começando na raiz, compare as coordenadas de acordo com a atual divisão da dimensão até o nó da folha ser atingido.

b) Marque o ponto localizado no nó da folha como o melhor ponto, e calcule a distância entre d e q com o nó atual da folha.

c) Mover um nível acima na árvore k-d e determinar a distância de q para este nó. Se a distância é mais curta do que d, defina este nó como sendo o melhor nó atual e a distância a ele como d. Se a distância de q para os correntes nós dividem o plano em distâncias maiores que d, exclua os nós do outro ramo lateral e continue a mover para cima até que a raiz seja alcançada. Caso contrário, outro ramo lateral ao nó é pesquisado.

d) Quando o nó superior é alcançado e todos os ramos laterais necessários foram pesquisados, escolher o mais curto de todos os candidatos a partir da busca principal e eventuais pesquisas dos sub-ramos.

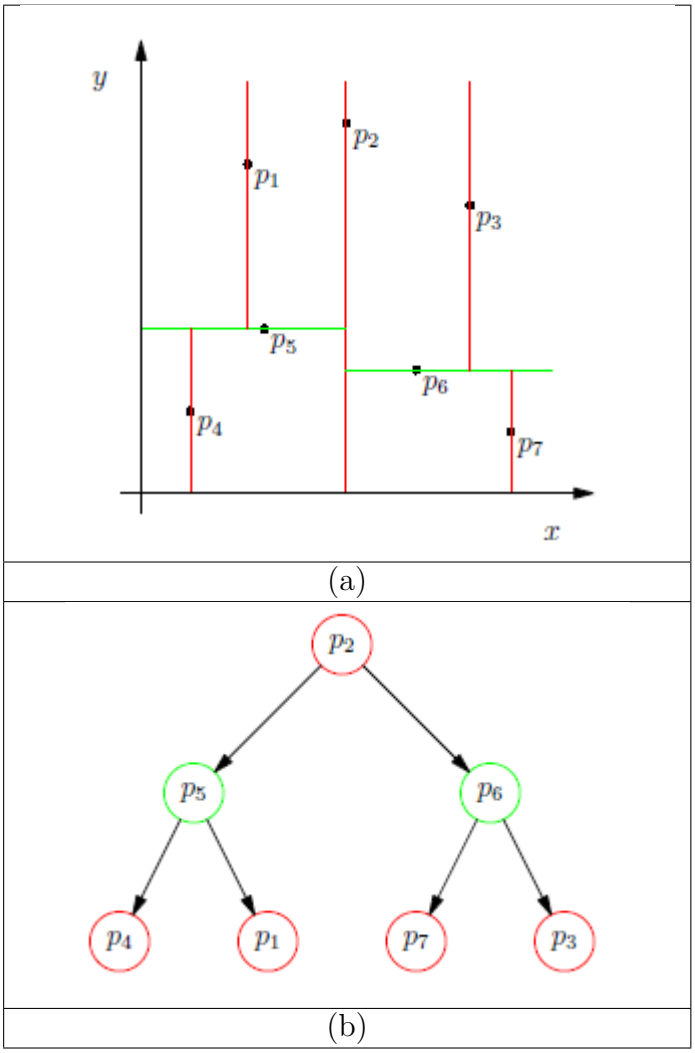


Figura 47 – (a): Uma nuvem de pontos que foi dividida utilizando a partição binária do espaço. (b): A árvore k-d correspondente.
Fonte: Autor “adaptado de” (KJER et al., 2010)

ANEXO H – GEOMETRIA EPIPOLAR

A Geometria Epipolar serve para identificar os pontos correspondentes entre duas imagens

Da figura 48 tem se:

Linha Epipolar

É o conjunto de pontos da cena que quando projetado na imagem esquerda forma um único ponto e que quando projetado na imagem direita forma uma linha chamada de linha epipolar. Como exemplo é o conjunto de pontos formados pela linha $\mathbf{x_2e_2}$ da imagem da câmera direita projeta no ponto $\mathbf{x_1}$ da imagem da câmera esquerda.

Plano Epipolar

É o plano definido pelos centros das câmeras $\mathbf{c_1}$ e $\mathbf{c_2}$ e um ponto da cena. Como exemplo é o plano formado pelos pontos $\mathbf{c_1}, \mathbf{c_2}$ *e* $\mathbf{w}$ ou o plano que contém as linhas epipolares $\mathbf{x_2e_2}$ e $\mathbf{x_1e_1}$.

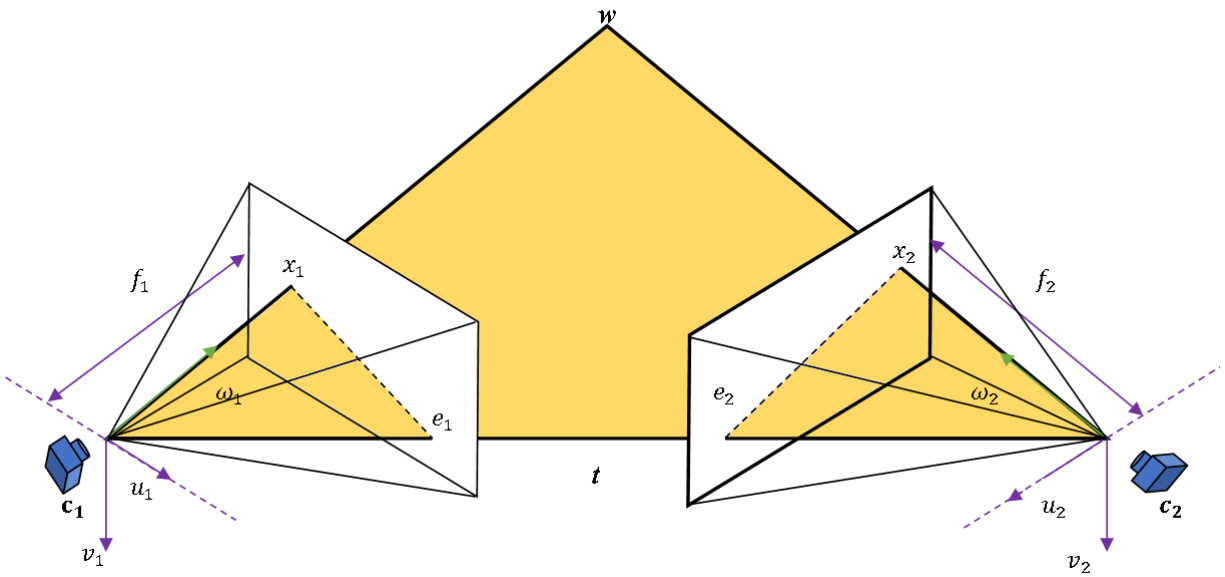


Figura 48 – Geometria Epipolar.
Fonte: Produzido pelo próprio autor.

Epipolo

É a intersecção do plano da imagem com a linha que conecta os centros das câmeras. Como exemplo o ponto $\mathbf{e_1}$ da câmera esquerda ou o ponto $\mathbf{e_2}$ da câmera direita.

Coordenadas Normalizadas

Sabe-se que o raio projetado do ponto $\mathbf{w}$ intercepta a imagem esquerda em $\mathbf{x_1}$ segundo uma projeção $\mathbf{P_1}$ sendo assim tem-se:

$$\mathbf{x_i} = \mathbf{P_i w} \tag{80}$$

onde o índice $\mathbf{i}$ possui os valores $\{\mathbf{1}, \mathbf{2}\}$ respectivamente das câmeras com centro em $\mathbf{c_1}$ e $\mathbf{c_2}$

A matriz de projeção da câmera é dada por:

$$\mathbf{P_i} = \mathbf{K_i}[\mathbf{R_i}|\mathbf{t_i}] \tag{81}$$

onde $\mathbf{K}$ é a matriz de calibração intrínseca de formato triangular com valor $\begin{bmatrix} \alpha_u & \gamma & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ e $\alpha_u = -f\mathbf{k}_u$, $\alpha_v = -f\mathbf{k}_v$ são as dimensões horizontais e verticais dos pixels, f é o tamanho do foco em milímetros, $\mathbf{k}_u$ e $\mathbf{k}_v$ são os números efetivos de pixels por milímetro ao longo do eixo u e v e (u_0, v_0) são as coordenadas do ponto principal do plano da imagem e γ é o fator (*skew factor*) que modela os eixos não ortogonais dos eixos u e v onde na maioria da câmeras este fator é muito próximo de zero.

A interpretação de uma câmera normalizada é que sua matriz de calibração intrínseca se torna a matriz identidade $\mathbf{I}$, sendo assim o foco f e as dimensões do pixels α_x e α_y serão quadradas com o valor unitário.

A matriz $\mathbf{K_i}$ depende somente dos parâmetros intrínsecos, pode-se considerar aqui que as duas câmeras possuem os mesmos valores intrínsecos, desta forma $\mathbf{K_1} = \mathbf{K_2} = \mathbf{K}$.

Ao se escolher um pixel da imagem $\mathbf{x_i}$, aplica-se a inversa da matriz de calibração onde obtêm-se um ponto $\hat{\mathbf{x}}_\mathbf{i}$ normalizado (coordenada normalizada) sendo assim temos:

$$\widehat{\mathbf{x_i}} = \mathbf{K^{-1}x_i} \tag{82}$$

Substituindo $\mathbf{P_i}$ da equação (77) pelo valor de $\mathbf{P_i}$da equação (78) e $\mathbf{x_i}$ da equação (79) tem-se:

$$\widehat{\mathbf{x_i}} = [\mathbf{R_i}|\mathbf{t_i}]\mathbf{w} \tag{83}$$

Lembrando que $\mathbf{R_i}$ é a matriz de rotação que alinha o sistema de coordenada da câmera com o sistema de coordenada global.

Coplanaridade

Os vetores $\overrightarrow{\mathbf{c_1x_1}}$, $\overrightarrow{\mathbf{c_2x_2}}$ e $\overrightarrow{\mathbf{c_1c_2}}$ estão contidos no plano epipolar da figura 48 ou seja eles são coplanares.

Isso implica que:

$$\overrightarrow{\mathbf{c_1x_1}} \bullet (\overrightarrow{\mathbf{c_1c_2}} \times \overrightarrow{\mathbf{c_2x_2}}) = \mathbf{0} \tag{84}$$

$\omega_\mathbf{1}$ e $\omega_\mathbf{2}$ são os vetores de direção respectivos ao sistema de referência de cada câmera:

$$\omega_\mathbf{i} = \frac{\mathbf{K^{-1}x_i}}{||\mathbf{K^{-1}x_i}||} = \frac{\widehat{\mathbf{x_i}}}{||\widehat{\mathbf{x_i}}||} \tag{85}$$

Como exemplo para a obtenção de $\widehat{\mathbf{x_i}}$ podemos obter a partir de $\omega_\mathbf{1}$ da seguinte forma $\omega_\mathbf{1} = \begin{bmatrix} \omega_\mathbf{1x} \\ \omega_\mathbf{1y} \\ \omega_\mathbf{1z} \end{bmatrix}$ portanto, $\widehat{\mathbf{x_i}} = \begin{bmatrix} \frac{\omega_\mathbf{1x}}{\omega_\mathbf{1z}} \\ \frac{\omega_\mathbf{1y}}{\omega_\mathbf{1z}} \\ \mathbf{1} \end{bmatrix}$

Observe que $\hat{\mathbf{x}}_\mathbf{i}$ tem a mesma direção de $\omega_\mathbf{i}$.

Matriz Essencial

Sejam as projeções $\mathbf{P_1}$ e $\mathbf{P_2}$ das câmeras normalizadas tem-se que $\mathbf{P_1} = [\mathbf{R_1}|\mathbf{t_1}]$ e $\mathbf{P_2} = [\mathbf{R_2}|\mathbf{t_2}]$, onde $\check{\mathbf{w}}$ é o ponto $\mathbf{w}$ em coordenada não homogênea, resultando assim,

$$\widehat{\mathbf{x_i}} = [\mathbf{R_i}|\mathbf{t_i}]\mathbf{w} = \mathbf{R_i}\check{\mathbf{w}} + \mathbf{t_i} \tag{86}$$

portanto o ponto $\check{\mathbf{w}}$ do espaço Euclidiano normalizado será:

$$\check{\mathbf{w}} = \mathbf{R_i^{-1}}\widehat{\mathbf{x_i}} - \mathbf{R_i^{-1}t_i} \tag{87}$$

Ambas as câmeras $\mathbf{c_1}$ e $\mathbf{c_2}$ podem reconstruir o ponto $\check{\mathbf{w}}$ sem perder a generalidade, podendo assim, assumir que o sistema de referência da câmera $\mathbf{c_1}$ será exatamente igual ao sistema global.

Alinhando o sistema de referência da câmera $\mathbf{c_1}$ com o sistema de referência global deste sistema normalizado, a matriz de rotação é substituída pela matriz identidade atrelada à origem, exemplo $\mathbf{P_1} = [\mathbf{I}|\mathbf{0}]$ portanto,

$$\mathbf{R_1^{-1}}\widehat{\mathbf{x_1}} - \mathbf{R_1^{-1}}\mathbf{t_1} = \mathbf{R_2^{-1}}\widehat{\mathbf{x_2}} - \mathbf{R_2^{-1}}\mathbf{t_2} \tag{88}$$

Assumindo que o sistema global tem a mesma referência que o sistema da câmera $\mathbf{c_1}$, assim $\mathbf{R_1^{-1}} = \mathbf{I}$ e $\mathbf{t_1} = \mathbf{0}$ tem-se:

$$\widehat{\mathbf{x_1}} = \mathbf{R_2^{-1}}\widehat{\mathbf{x_2}} - \mathbf{R_2^{-1}}\mathbf{t_2} \tag{89}$$

multiplicando por $\mathbf{R_2}$

$$\mathbf{R_2}\widehat{\mathbf{x_1}} = \widehat{\mathbf{x_2}} - \mathbf{t_2} \tag{90}$$

portanto,

$$\widehat{\mathbf{x_2}} = \mathbf{R_2}\widehat{\mathbf{x_1}} + \mathbf{t_2} \tag{91}$$

Para simplificar a equação (88) retirando a soma da equação usaremos o produto vetorial $\mathbf{t_2}\times$ em ambos os lados da equação

$$\mathbf{t_2} \times \widehat{\mathbf{x_2}} = \mathbf{t_2} \times \mathbf{R_2}\widehat{\mathbf{x_1}} + \mathbf{t_2} \times \mathbf{t_2} \tag{92}$$

onde $\mathbf{t_2} \times \mathbf{t_2} = 0$, e para trocar o produto vetorial pelo produto escalar, utiliza-se a matriz equivalente do produto vetorial $[\mathbf{t}]_\mathbf{x}$ sendo $[\mathbf{t}]_\mathbf{x} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$ uma matriz antissimétrica: $[\mathbf{t}]_\mathbf{x}^\mathbf{T} = -[\mathbf{t}]_\mathbf{x}$, assim tem-se:

$$[\mathbf{t_2}]_\mathbf{x}\widehat{\mathbf{x_2}} = [\mathbf{t_2}]_\mathbf{x}\mathbf{R_2}\widehat{\mathbf{x_1}} \tag{93}$$

Fazendo o produto com o ponto $\hat{\mathbf{x_2}}$ em ambos os lados da equação (90) tem-se:

$$\widehat{x_2} \bullet [\mathbf{t_2}]_\mathbf{x}\widehat{\mathbf{x_2}} = \widehat{\mathbf{x_2}} \bullet [\mathbf{t_2}]_\mathbf{x}\mathbf{R_2}\widehat{\mathbf{x_1}} \tag{94}$$

trocando o produto vetorial pela transposta

$$\hat{\mathbf{x}}_\mathbf{2}^\mathbf{T}[\mathbf{t_2}]_\mathbf{x}\widehat{\mathbf{x_2}} = \hat{\mathbf{x}}_\mathbf{2}^\mathbf{T}[\mathbf{t_2}]_\mathbf{x}\mathbf{R_2}\widehat{\mathbf{x_1}} \tag{95}$$

Finalmente

$$\hat{\mathbf{x}_2^{\mathbf{T}}}[\mathbf{t_2}]_{\mathbf{x}}\mathbf{R_2}\widehat{\mathbf{x_1}} = \mathbf{0} \tag{96}$$

$$\widehat{\mathbf{x}_2^{\mathbf{T}}}\mathbf{E}\widehat{\mathbf{x}_1} = \mathbf{0} \tag{97}$$

Pois $[\mathbf{t_2}]_{\mathbf{x}}\widehat{\mathbf{x_2}} = \mathbf{t_2} \times \widehat{\mathbf{x_2}} = \mathbf{0}$ e $\mathbf{t_2} \times \widehat{\mathbf{x_2}}$ é ortogonal $\widehat{\mathbf{x_2}}$. A equação (94) é conhecida como restrição epipolar e $\mathbf{E} = [\mathbf{t_2}]_{\mathbf{x}}\mathbf{R_2}$ é conhecido com a matriz essencial.

Mapeando o ponto a linha

A matriz $\mathbf{E}$ mapeia $\widehat{\mathbf{x_1}}$ da imagem 1 à linha $\mathbf{l_2} = \mathbf{E}\widehat{\mathbf{x_1}}$ na imagem 2, pois $\widehat{\mathbf{x_2^T}}\mathbf{l_2} = \mathbf{0}$.

Todas essas linhas passam pelo epipolo $\mathbf{e_2}$.

O epipolo é a projeção do vetor $\mathbf{t}$ sobre a imagem.

Simetria

Se $\widehat{\mathbf{x_2^T}}\mathbf{E}\widehat{\mathbf{x_1}} = \mathbf{0}$ então:

$$\widehat{\mathbf{x_2^T}}\mathbf{E}\widehat{\mathbf{x_1}} = \widehat{\mathbf{x_2}} \bullet \mathbf{E}\widehat{\mathbf{x_1}} = \mathbf{E}\widehat{\mathbf{x_1}} \bullet \widehat{\mathbf{x_2}} = (\mathbf{E}\widehat{\mathbf{x_1}})^{\mathbf{T}}\widehat{\mathbf{x_2}} = \mathbf{0} \tag{98}$$

$$\widehat{\mathbf{x_1^T}}\mathbf{E^T}\widehat{\mathbf{x_2}} = \mathbf{0} \tag{99}$$

A matriz essencial $\mathbf{E^T}$ mapeia $\widehat{\mathbf{x_2}}$ na imagem 2 para a linha epipolar $\mathbf{l_1} = \mathbf{E^T}\widehat{\mathbf{x_2}}$ na imagem 1, pois $\widehat{\mathbf{x_1}}\mathbf{l_1} = \mathbf{0}$.

Todas essas linhas passam pelo epipolo $\mathbf{e_1}$.

Matriz Fundamental

A vantagem de trabalhar com a matriz fundamental ao invés da matriz essencial é que na matriz essencial é preciso calcular os parâmetros intrínsecos utilizados na matriz da câmera $\mathbf{K}$.

Para ver como isto funciona tem-se a matriz essencial dada por $\widehat{\mathbf{x_2^T}}\mathbf{E}\widehat{\mathbf{x_1}} = \mathbf{0}$ deixe:

$$\widehat{\mathbf{x_2}} = \mathbf{K^{-1}}\check{\mathbf{x}}_{\mathbf{2}} \tag{100}$$

$$\widehat{\mathbf{x_1^T}} = (\mathbf{K^{-1}}\check{\mathbf{x}}_{\mathbf{1}})^{\mathbf{T}} = \check{\mathbf{x}}_{\mathbf{1}}{}^{\mathbf{T}}\mathbf{K^{-T}} \tag{101}$$

então

$$\mathbf{\check{x}_1}^{\mathbf{T}}\mathbf{K}^{-\mathbf{T}}\mathbf{E}\mathbf{K}^{-\mathbf{1}}\mathbf{\check{x}_2} \tag{102}$$

ou identificando o centro da equação (99) como $\mathbf{F} = \mathbf{K}^{-\mathbf{T}}\mathbf{E}\mathbf{K}^{-\mathbf{1}}$, aonde $\mathbf{F}$ é a matriz fundamental tem-se:

$$\mathbf{\check{x}_1}^{\mathbf{T}}\mathbf{F}\mathbf{\check{x}_2} = \mathbf{0} \tag{103}$$

que é a equação que relaciona os pontos não normalizados notando que ainda pode-se reconstruir as linhas epipolares através da matriz fundamental $\mathbf{F}$.

Solução da Matriz Fundamental

Para solucionar $\mathbf{F}$ utiliza-se os mesmos métodos utilizados na solução da matriz essencial $\mathbf{E}$, exceto que agora os pontos correspondentes estão em coordenadas não normalizadas.

Com isso tem-se:

$$\mathbf{\check{x}_1}^{\mathbf{T}}\mathbf{F}\mathbf{\check{x}_2} = \mathbf{0}$$

onde

$$\begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \mathbf{0}$$

Escrevendo na forma $\mathbf{Ax} = \mathbf{0}$, aonde $\mathbf{x} = (\mathbf{F_{11}}, \mathbf{F_{12}}, \mathbf{F_{13}}, ..., \mathbf{F_{33}})$ tem-se:

$$\begin{bmatrix} x1x2 & x1y2 & x1 & y1x2 & y1y2 & y1 & x2 & y2 & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ \vdots \\ F_{33} \end{bmatrix} = \mathbf{0}$$

Sendo $\mathbf{Ax} = \mathbf{0}$ um sistema de equações homogêneas, pode-se utilizar neste caso o valor de decomposição singular (SVD).

E como pré condição, primeiramente a uma translação e um escalonamento dos pontos de forma a ficarem com a origem centrada e a distância média à origem é $\sqrt{2}$.

E também como pós condição é que os valores de **F** não são independentes. Tem-se aqui somente cinco parâmetros independentes, para que isso aconteça tem-se que forçar **F** a ter um *rank=2.*

ANEXO I – RETIFICAÇÃO

Retificação é um caso especial de rotação e translação dos planos da imagem esquerda e direita, isto acontece quando o centro da câmera esquerda $\mathbf{c_1}$ (vide figura 49) estiver no plano focal da câmera à direita, o epipolo da direita $\mathbf{e_2}$, vai estar no infinito formando linhas epipolares paralelas na imagem direita. Um caso muito especial é quando os dois epipolos $\mathbf{e_1}$ e $\mathbf{e_2}$ estão no infinito, isto acontece quando a linha $\mathbf{c_1c_2}$ (linha da base) está contida em ambos os planos focais, formando linhas paralelas e horizontais. Este tipo de procedência é chamado de retificação (MORAVEC, 1983).

Retificação das Matrizes das Câmeras

Assumindo a calibração das câmeras o *Perspective Projection Matrix (PPM)* $\widetilde{\mathbf{P_{o1}}}$ *e* $\widetilde{\mathbf{P_{o2}}}$ são conhecidos. A ideia por trás da retificação é definir dois novos *PPMs* $\widetilde{\mathbf{P_{n1}}}$ *e* $\widetilde{\mathbf{P_{n2}}}$ rotacionando $\widetilde{\mathbf{P_{o1}}}$ *e* $\widetilde{\mathbf{P_{o2}}}$ em torno de seus centros ópticos, até que os planos focais se tornem coplanares, desta forma contendo a linha da base. Fazendo isto temos os epipolos no infinito portanto as linhas epipolares de ambas as imagens ficam paralelas. Para ter linhas epipolares horizontais, a linha da base tem que ficar paralela ao novo eixo u de ambas as câmeras. Adicionado a isto para se ter uma apropriada retificação, pontos conjugados precisão estar na mesma coordenada vertical. Isto é obtido quando as câmeras possuem os mesmos parâmetros intrínsecos.

Escrevendo *PPM* como

$$\tilde{\mathbf{P}} = \left[\begin{array}{c|c} \mathbf{q_1^T} & q_{14} \\ \mathbf{q_2^T} & q_{24} \\ \mathbf{q_3^T} & q_{34} \end{array}\right] = [\mathbf{Q}|\tilde{\mathbf{q}}] \tag{104}$$

Em coordenadas Cartesianas, a projeção pode ser escrita

$$\begin{cases} u = \frac{\mathbf{q_1^T w} + q_{14}}{\mathbf{q_3^T w} + q_{34}} \\ v = \frac{\mathbf{q_2^T w} + q_{24}}{\mathbf{q_3^T w} + q_{34}} \end{cases} \tag{105}$$

O plano focal é o plano paralelo ao plano da imagem que contêm o centro óptico $\mathbf{c}$, onde a coordenada de $\mathbf{c}$ é dada por

$$\mathbf{c} = -\mathbf{Q^{-1}}\tilde{\mathbf{q}} \tag{106}$$

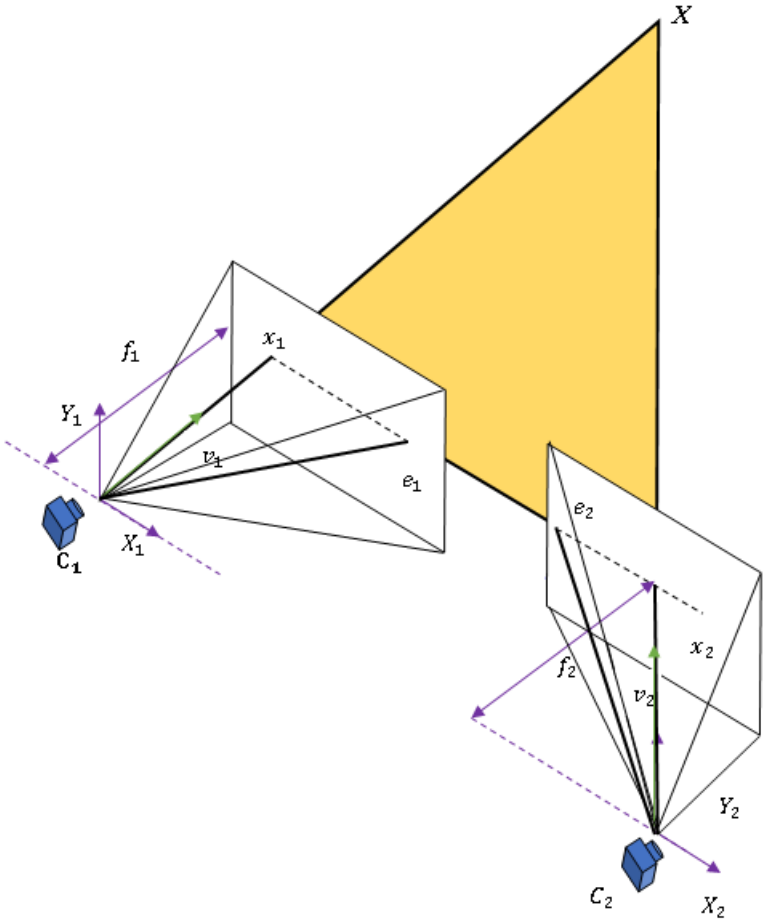


Figura 49 – Retificação.
Fonte: Produzido pelo próprio autor.

Desta forma $\tilde{\mathbf{P}}$ pode ser escrito como:

$$\tilde{\mathbf{P}} = [\mathbf{Q}| - \mathbf{Qc}]. \tag{107}$$

O raio óptico associado ao ponto da imagem $\mathbf{x}$ é a linha $\mathbf{xc}$, onde o conjunto de pontos em 3D $\{\mathbf{w} : \tilde{\mathbf{x}} \cong \tilde{\mathbf{P}}\tilde{\mathbf{w}}\}$. Sua forma paramétrica será:

$$\mathbf{w} = \mathbf{c} + \lambda \mathbf{Q}^{-1}\tilde{\mathbf{m}}, \ \textit{onde} \ \lambda \in \mathbb{R} \tag{108}$$

Fatorizando os novos *PPMs* temos da equação (104)

$$\widetilde{\mathbf{P_{n1}}} = \mathbf{K}[\mathbf{R}| - \mathbf{Rc_1}], \ \widetilde{\mathbf{P_{n2}}} = \mathbf{K}[\mathbf{R}| - \mathbf{Rc_2}] \tag{109}$$

Lembrando que a matriz intrínseca $\mathbf{K}$ é a mesma em ambas as câmeras. Os centros ópticos $\mathbf{c_1}$e $\mathbf{c_2}$ são calculados pela equação (103) com valores antigos e a matriz $\mathbf{R}$ que

proporciona a pose é a mesma para ambos os *PPMs.* podendo ser especificado pelo vetor linha:

$$\mathbf{R} = \begin{bmatrix} \mathbf{r_1^T} \\ \mathbf{r_2^T} \\ \mathbf{r_3^T} \end{bmatrix} \tag{110}$$

que são os eixos X,Y e Z, expresso em coordenadas globais.

Produzindo agora os novos eixos de rotação teremos:

a) O novo eixo X paralelo à linha da base será: $\mathbf{r_1} = (\mathbf{c_1} - \mathbf{c_2})/||\mathbf{c_1} - \mathbf{c_2}||$.

b) O novo eixo Y ortogonal a X será: $\mathbf{r_2} = \mathbf{k} \times \mathbf{r_1}$.

c) O novo eixo Z ortogonal a XY será: $\mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2}$.

Sendo que $\mathbf{k}$ é um vetor unitário arbitrário, que fixa a posição do novo eixo Y no plano ortogonal à X. Fazendo com que $\mathbf{k}$ seja igual ao antigo vetor unitário do eixo Z da matriz esquerda, desta forma o eixo Y é restringido de forma a ser ortogonal a ambos o novo X e o velho eixo esquerdo Z.

A Transformação retificadora

Ao retificar a imagem esquerda, é preciso calcular a transformação que mapeia o plano da imagem de $\tilde{\mathbf{P}}_{\mathbf{o1}} = [\mathbf{Q_{o1}}|\tilde{\mathbf{q}}_{\mathbf{o1}}]$, para o novo plano da imagem $\tilde{\mathbf{P}}_{\mathbf{n1}} = [\mathbf{Q_{n1}}|\tilde{\mathbf{q}}_{\mathbf{n1}}]$. Pode-se observar que a transformação é colinear dada por uma matriz 3 x 3 $\mathbf{T_1} = \mathbf{Q_{n1}Q_{o1}^T}$. O mesmo resultado é aplicado à imagem direita.

Para qualquer ponto 3D, pode ser escrito

$$\begin{cases} \tilde{x}_{o1} = & \tilde{P}_{o1}\tilde{w} \\ \tilde{x}_{n1} = & \tilde{P}_{n1}\tilde{w} \end{cases}$$

e de acordo com a equação (105) do raio óptico (desde que retificação não move o centro óptico):

$$\begin{cases} \mathbf{w} = \mathbf{c_1} + \lambda_{\mathbf{o}}\mathbf{Q_{o1}^{-1}}\tilde{\mathbf{m}}_{\mathbf{o1}} & onde\ \lambda_{\mathbf{o}} \in \mathbb{R} \\ \mathbf{w} = \mathbf{c_1} + \lambda_{\mathbf{n1}}\mathbf{Q_{n1}^{-1}}\tilde{\mathbf{m}}_{\mathbf{n1}} & onde\ \lambda_{\mathbf{n}} \in \mathbb{R} \end{cases}$$

desta forma,

$$\tilde{\mathbf{m}}_{\mathbf{n1}} = \lambda \mathbf{Q_{n1}Q_{o1}^{-1}}\tilde{\mathbf{m}}_{\mathbf{o1}} onde\ \lambda \in \mathbb{R} \tag{111}$$

A transformação $\mathbf{T_1}$ é aplicada então a imagem original da esquerda produzindo desta forma uma imagem retificada.