UNIVERSIDADE FEDERAL DE PERNAMBUCO
Centro de Informática

Pós-graduação em Ciência da Computação

APLICAÇÃO DO TESTE KENS PARA DETECÇÃO DE OUTLIERS EM FLUXO ÓTICO

Samuel Victor Medeiros de Macêdo

DISSERTAÇÃO DE MESTRADO

Recife
Março, 2013

UNIVERSIDADE FEDERAL DE PERNAMBUCO
Centro de Informática

Samuel Victor Medeiros de Macêdo

APLICAÇÃO DO TESTE KENS PARA DETECÇÃO DE OUTLIERS EM FLUXO ÓTICO

Trabalho apresentado ao Programa de Ps-graduao em Cincia da Computao do Centro de Informtica da UNIVERSIDADE FEDERAL DE PERNAMBUCO como requisito parcial para obteno do grau de Mestre em Cincia da Computao.

Orientadora: *Profa. Dra. Judith Kelner*

Recife
Março, 2013

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Macêdo, Samuel Victor Medeiros de
Aplicação do teste kens para detecção de outliers em fluxo ótico. / Samuel Victor Medeiros de Macêdo. - Recife: O Autor, 2013.
xix, 51 folhas, fig., tab.

Orientador: Judith Kelner.
Dissertação (mestrado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2013.

Inclui bibliografia e apêndice.

1. Ciência da computação. 2. Visão computacional. I. Kelner, Judith (orientador). II. Título.

004 CDD (23. ed.) MEI2013 – 074

Dissertação de Mestrado apresentada por **Samuel Victor Medeiros de Macêdo** à Pós Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Aplicação do Teste Kens para Detecção de Outliers em Fluxo Ótico**" orientada pela **Profa. Judith Kelner** e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Silvio de Barros Melo
Centro de Informática / UFPE

_____

Prof. Thiago Souto Maior Cordeiro de Farias
Departamento de Ciências da Terra – UPE/Caruaru

_____

Profa. Judith Kelner
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 1 de março de 2013

_____

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

A meus pais.

AGRADECIMENTOS

Pois é, agradecer é fácil. É só lembrar de todos que me ajudaram a edificar esse trabalho. O problema é que foram muitos e não caberia todos os nomes nesse documento...então falarei especificamente daqueles que não poderiam estar de forma alguma fora desse agradecimento.

Primeiramente queria a agradecer a minha orientadora e madrinha Judith Kelner até por que sem ela não existiria mestrado. Agradeço sua paciência, conselhos e apoio pessoal. A senhora sabe cobrar e sabe ser amiga.

Agradecer a meus velhos pais Edinaldo e Beatriz por terem se esforçado tanto pra me dar uma boa educação. Este mestrado e meu doutorado que começará em breve são os frutos colhidos depois de tanto tempo. Obviamente agradecer o companheirismo de meu irmão Isaac que esteve ao meu lado dando estímulo e tapas. Adamastor e Caniná também amo vocês.

Agradeço também aos meu amigos do GRVM e meus amigos do VOXAR. Todos aqueles que me ajudaram seja tecnicamente ou simplesmente com um abraço ou uma palavra de conforto ou uma coquinha em Bigo. Agradecimento especial aqueles que revisaram e tiveram paciência com as minhas teimosias e desabafos.

Agradeço a Sony Computer Entertainment por proporcionar meus tempos de lazer aos quais sem eles não teria forças pra terminar este trabalho.

Agradecimento especial a minha esposa Ana Paula por estar sempre do meu lado nos momentos em que estava cansado ou precisava desabafar. Obrigado por sempre me acordar na hora pois sem você eu dormiria a manhã toda e não terminaria esta dissertação. Além de agradecer por estar comigo, queria agradece-la pelo maravilhoso presente que é nossa filha Carolina. Obrigado por cuidar dela enquanto estou trabalhando e aguentar todos os "que cor?" que ela pergunta. É por vocês duas que saio todo dia pro trabalho e que volto cansado para descansar.

Por último agradeço ao CNPq, à Petrobrás e à CHESF pelo incentivo através dos projetos de pesquisa e pelas bolsas concedidas que proporcionaram a realização deste trabalho.

Make everything as simple as possible, but not simpler.

—ALBERT EINSTEIN

RESUMO

A área de reconstrução 3D tem sido bastante explorada, principalmente nos últimos anos, com a popularização de ferramentas para visualizar objetos tridimensionais. A busca por algoritmos eficientes que tornem o pipeline de reconstrução 3D mais eficiente é alvo de várias pesquisas universitárias e patentes tanto na indústria como na academia. Atualmente, alguns problemas existentes para reconstrução de malhas que possuem elevado número de pontos utilizando o *pipeline* de reconstrução [40] ainda persistem, mesmo aplicando apenas algumas restrições. Estes problemas são causados pela exigência de elevado poder computacional exigido pelas técnicas usuais. Dentre essas técnicas estão o rastreamento de pontos em imagens (*feature tracking*) [49] e a geração e avaliação de várias hipóteses de pose de câmera para encontrar a técnica que melhor se adequa à cena em questão [37].

A reconstrução 3D pode ser bastante útil em diversas áreas como: realidade aumentada sem marcadores, para a manipulação de objetos virtuais que interagem fisicamente com o mundo real e o tratamento de oclusão de objetos virtuais por objetos reais. Diante da problemática e da diversidade de aplicações, alterações no *pipeline* de reconstrução 3D que o tornem mais rápido e eficiente são interessantes tanto para a área de visão computacional quanto para a indústria.

No contexto desta problemática, esta dissertação propõe uma metodologia para otimização do *pipeline* de reconstrução 3D explorando os conceitos de inferência estatística, mais precisamente a área de teste de hipótese. O teste *kens* é um teste de hipótese estatístico desenvolvido nesta dissertação para verificar a suavidade de uma trajetória. Este teste será aplicado aos caminhos das *features* uma vez que o rastreamento das mesmas é feito utilizando fluxo ótico. Apesar de não ser provado matematicamente que *features inliers* percorrem caminhos suaves, este trabalho mostra indícios de uma relação entre suavidade e *inliers*, pois com a retirada das *features* que apresentaram caminhos não suaves a qualidade da reconstrução 3D apresentou resultados melhores.

Esta dissertação de mestrado descreve todo o ferramental teórico necessário para entendimento do *pipeline* de reconstrução 3D e do teste *kens*. A utilização da técnica em dois cenários será apresentada: sendo um cenário sintético e o outro real.

Palavras-chave: Detecção de *Outlier*; Fluxo Ótico; Visão Computacional; Reconstrução 3D; Inferência Estatística

ABSTRACT

The 3D reconstruction area has been widely explored in recent years especially with the popularization of resources to visualize three dimensional objects. The search for efficient algorithms to optimize the 3D reconstruction pipeline is at the heart of several university research projects and the subject of many industrial patents. Currently, there are problems for 3D reconstruction that have elevated quantities of points using the classic pipeline reconstruction [40]. These problems are caused by the requirement for high computational power demanded by the usual techniques. Among these techniques, there are tracking points in images (*feature tracking*) [49] and the generation and evaluation of pose camera hypotheses to find the technique that best suits the scene in question [37].

The 3D reconstruction can be useful in diverse areas such as augmented reality without markers, for the manipulation of virtual objects that physically interact with the real world and treatment of occlusion of virtual objects by real objects. Regarding these problems and the diversity of applications, changes in 3D reconstruction *pipeline* which would make it faster and more efficient are interesting for the computer vision area and industry.

This dissertation proposes a methodology for optimizing the 3D reconstruction *pipeline* exploring the concepts of statistical inference, specifically the area hypothesis test. The test *kens* is a statistical hypothesis test developed in this work to verify the smoothness of a trajectory. This test will be applied to *features* path since their tracking are made using optical flow. Although not proven mathematically that *features inliers*run smoother paths, this study shows evidence of a relationship between smoothness and *inliers*, because with the removal of *features* that the technique showed the existence of non-smooth paths, offering an improvement in the quality of 3D reconstruction. This work describes all the theoretical tools necessary for understanding the 3D reconstruction *pipeline* and the *kens* test. It will be shown using the technique in two scenarios: one synthetic and one real.

Keywords: *Outlier* Detection; Optical flow; Computer Vision; 3D Reconstruction; Statistical Inference

SUMÁRIO

Capítulo 1—Apresentação 1

1.1 *Pipeline* de Reconstrução 3D . . . 1
1.2 Problema . . . 2
1.3 Objetivos . . . 3
1.4 Relevância . . . 3
1.5 Principais Contribuições . . . 4
1.6 Organização da Dissertação . . . 4

Capítulo 2—Fundamentação Teórica 5

2.1 Rastreamento . . . 5
2.1.1 Fluxo Ótico . . . 5
2.1.2 Rastreamento de *Features* . . . 7
2.1.3 KLT . . . 8
2.2 Reconstrução 3D . . . 10
2.2.1 Calibração de Câmera . . . 10
2.2.2 Reconstrução Esparsa . . . 11
2.2.3 Reconstrução Densa . . . 13
2.2.4 Texturização . . . 14
2.3 Inferência Estatística . . . 14
2.3.1 Definições Básicas . . . 15
2.3.2 Teste de Hipótese . . . 15
2.3.3 Etapas do Teste de Hipótese . . . 16
2.4 Fatoração . . . 17

Capítulo 3—Estado da Arte 19

3.1 Tensor Trifocal Robusto . . . 19
3.2 TRAOD . . . 20
3.3 HEI . . . 21
3.4 Considerações . . . 22

Capítulo 4—Proposta: Teste *kens* 23

4.1 Seleção de *Features* . . . 23
4.2 Motivação . . . 24
4.3 Construção do Teste de Hipótese . . . 25

4.3.1 Variável Aleatória 25
4.3.2 Suposições 27
4.3.3 Estatística de Teste 27
4.3.4 Hipóteses do Teste 28
4.3.5 Distribuição de Probabilidade Aproximada 29
4.4 Aplicação do Teste Kens 30

Capítulo 5—Resultados 31

5.1 Vídeo Sintético 32
5.2 Vídeo Real 35
5.3 Considerações 38

Capítulo 6—Conclusão 39

6.1 Trabalhos Futuros 40

Apêndice A—Derivadas de 1º e 2º ordem 41

Apêndice B—Triângulo de Pascal 43

Apêndice C—Distribuição de probabilidade de K 45

LISTA DE FIGURAS

1.1 Exemplo de reconstrução 3D 1
1.2 Pipeline de Reconstrução 3D 2

2.1 Fluxo Óptico de *Pixels* 6
2.2 Exemplos de dificuldades encontradas no rastreamento de sequências de imagens: (a) cena original; (b) distorção por rotação; (c) borramento por movimentação; (d) distorção por escala; (e) oclusão parcial; (f) e (g) distorção por mudança da iluminação. 7
2.3 Influência do erro 2D no espaço 3D 8
2.4 Pipeline de Reconstrução 3D 10
2.5 Triangulação 11
2.6 Pipeline: Rastreamento 11
2.7 Pipeline: Matriz Fundamental 12
2.8 Pipeline: Pose de câmera 12
2.9 Pipeline: Triangulação 13
2.10 Pipeline: Reconstrução Densa 14
2.11 Pipeline: Texturização 14
2.12 (a) *features* (b) caminhos das *features* ao longo dos quadros 17

3.1 Conceito de t-partição 21
3.2 Círculo de Confiança 21

4.1 Trajetória Outlier 23
4.2 (a)Caminho da *feature a* (b)Caminho da *feature b* 26

5.1 Vídeo sintético - Casa 32
5.2 (a)*frame* 1: Vídeo sintético - Casa (b) *frame* 1(*kens*): Vídeo sintético - Casa 33
5.3 (a)*frame* 11: Vídeo sintético - Casa (b) *frame* 11(*kens*): Vídeo sintético - Casa 33
5.4 (a)*frame* 88: Vídeo sintético - Casa (b) *frame* 88(*kens*): Vídeo sintético - Casa 34
5.5 (a)Reconstrução: Vídeo sintético - Casa (b) Reconstrução(kens): Vídeo sintético - Casa 34
5.6 Vídeo sintético - Casa: Erro de reprojeção *frame* à *frame* 35
5.7 Vídeo real - Abacaxi 36
5.8 (a)*frame* 1: Vídeo real - Abacaxi (b) *frame* 1(*kens*): Vídeo real - Abacaxi 36
5.9 (a)*frame* 12: Vídeo real - Abacaxi (b) *frame* 12(*kens*): Vídeo real - Abacaxi 36

5.10 (a)Reconstrução: Vídeo real - Abacaxi (b)Reconstrução(kens): Vídeo real - Abacaxi 37
5.11 Vídeo real - Abacaxi: Erro de reprojeção *frame* a *frame* 37

A.1 (a) Inclinação da secante ao gráfico de f, (b) Inclinação da tangente à curva como a derivada de $f(x)$ 41
A.2 (a) Comportamento da derivada com a concavidade voltada para cima, (b) Comportamento da derivada com a concavidade voltada para cima 42

B.1 (a) Triângulo de Pascal (b) Triângulo de Pascal usando Binômio de Newton 43

C.1 Círculos pretos são os valores exatos e as linhas verdes são os valores aproximados 45
C.2 Círculos pretos são os valores exatos e as linhas verdes são os valores aproximados 46
C.3 Círculos pretos são os valores exatos e as linhas verdes são os valores aproximados 46

LISTA DE TABELAS

2.1 Tipos de erros em testes de hipóteses 15

4.1 Valores para as derivadas aproximadas de 1^{o} e 2^{o} ordem para as *features* a e b 26
4.2 Valores de k_i e K para as *features* a e b 27
4.3 Valores de K e suas probabilidades para um caminho com 7 *frames* . . . 28
4.4 Distribuição de probabilidade exata da variável K para n termos 28

CAPÍTULO 1

APRESENTAÇÃO

Reconstrução 3D a partir de imagens 2D exige conhecimento multidisciplinar em áreas tais como: processamento de imagem, visão computacional, álgebra linear, estatística, otimização de sistemas não lineares, entre outras [20],[13]. Estes domínios específicos são necessários em todas as fases do *pipeline* de reconstrução 3D, desde a aquisição da imagem 2D até o produto final, que no contexto desta dissertação se descreve como a geração de uma malha reconstruída (Figura 1.1).

Figura 1.1 Exemplo de reconstrução 3D

A visão computacional é uma área de pesquisa, com inúmeras contribuições a técnicas de reconstrução 3D, voltada principalmente à modelagem de informações do mundo real. Seus avanços envolvem a integração com algoritmos que executam em tempo real, abordagens estatísticas robustas, métodos de reconstrução 3D densa e aceleração baseada em hardware. Portanto, vários desafios e questões de pesquisa em aberto motivam o desenvolvimento de novas técnicas e a integração destas com outras já existentes.

Uma vez de posse de tais informações, podemos utilizá-las para os mais diversos fins, por exemplo, modelagem automática de objetos, sistemas de navegação autônoma de robôs, posicionamento de elementos virtuais em cenas reais, dentre outros [53].

Um dos maiores dilemas da visão computacional se concentra na escolha entre precisão ou tempo real. Algumas aplicações requerem tempo real enquanto que em outras o tempo é irrelevante desde que a solução seja precisa.

1.1 *PIPELINE* DE RECONSTRUÇÃO 3D

O *pipeline* de reconstrução 3D (segundo [40]) pode ser dividido em seis fases: aquisição de imagens e segmentação, estimativa da matriz fundamental, estimativa da matriz de pose, triangulação, reconstrução densa e texturização da reconstrução 3D (Figura 1.2).

A fase de aquisição é responsável pelo rastreamento e processamento de imagem, primeiramente ocorre a extração de pontos característicos (*features*) na imagem e depois procura-se o casamento dessas *features* ao longo da sequência do video. Após encontrar as correspondências em cada imagem, a fase seguinte é a estimativa da matriz fundamental.

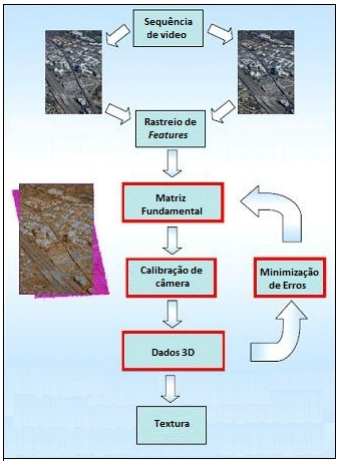

Figura 1.2 Pipeline de Reconstrução 3D

Essa matriz é responsável pela relação entre um par de imagens na mesma cena, desde que tenham pontos projetados em ambas as imagens. Existe então uma matriz fundamental para cada par de imagens.

Depois do cálculo da matriz fundamental, a terceira fase do *pipeline* consiste na estimativa da matriz de pose de câmera [20]. Essa matriz é definida como 3x4 e é constituída pela rotação e translação da câmera. Um ponto 3D é reconstruído por triangulação através dos pontos correspondentes em cada grupo de imagens e da correspondente pose de câmera. A quarta fase do *pipeline* consiste na reconstrução da cena esparsa.

A partir da reconstrução esparsa, entra-se na fase da reconstrução densa quando são adicionados pontos à reconstrução de modo a preencher o objeto reconstruído. A última fase consiste em gerar a textura das imagens resultando no modelo 3D reconstruído.

O *pipeline* será detalhado no capítulo 2 e mais informações sobre o mesmo podem ser encontradas em [20],[53] e [41].

1.2 PROBLEMA

Uma das formas mais difundidas de se realizar reconstrução 3D é utilizando sequências contíguas de imagens ou vídeos capturados por câmeras convencionais (monoculares) [51].

Quando a reconstrução 3D é feita a partir de dados reais existe uma introdução de erros que vão se acumulando ao longo do *pipeline*. A adição de ruídos começa na aquisição da imagem, essa etapa depende de parâmetros tais como: a resolução da imagem, o sensor da câmera e a iluminação. O processamento dessa imagem já vem acrescido de ruído e esse será passado para as próximas etapas.

Na fase de rastreamento existem também algumas dificuldades inerentes, como a oclusão de parte da cena e falsos casamentos. Isto se deve a áreas na imagem com texturas pobres e gradiente com baixa significância que também podem introduzir ruído no posicionamento das *features* ao longo do rastreio.

Considerando a priori que existem erros introduzidos pela aquisição e pelo rastreamento, é necessária uma abordagem que leve em conta esses erros durante o cálculo da

matriz de câmera e da matriz fundamental [20].

Só seria possível uma reconstrução "perfeita" se os dados vindos do rastreio fossem obtidos sem nenhum erro, ou seja, usando o *groundtruth*. Como a distribuição real dos erros é desconhecida, não é possível a princípio filtrar quais são as *features* fidedignas da cena (*inliers*) das *features* mal rastreadas (*outliers*).

O *groundtruth* pode ser obtido por métodos com rastreamento a laser ou com câmeras fixadas em uma cena pré concebida. Todas essas formas são dispendiosas e não são genéricas o suficiente a ponto do usuário querer reconstruir uma cena em qualquer lugar e apenas com uma câmera.

Para tentar resolver os problemas acima mencionados utiliza-se cálculo de estimativas robustas, para obter através de uma amostragem aleatória o modelo que melhor se ajusta a todas as observações. Assim, é possível separar os dados considerados corretos de acordo com as estimativas selecionadas.

Partindo-se de imagens reais, ruídos são adicionados ao dados, o que pode vir a gerar uma hipótese não fidedigna ao produto final desejado, seja a homografia ou qualquer outra relação.

1.3 OBJETIVOS

Considerando que é praticamente impossível evitar o acúmulo de erros passados de uma fase do *pipeline* para a outra, devido a erros numéricos, minimização de sistemas não-lineares e vários outras limitações do *pipeline*, uma solução para este problema é minimizar o máximo possível a quantidade de erros na primeira etapa do *pipeline* e fornecer ao algoritmo de reconstrução as melhores *features* possíveis.

Dentro deste contexto, o objetivo desta dissertação é especificar e definir uma metodologia baseada em métodos estatísticos para filtrar as *features* na fase de rastreamento do *pipeline* utilizando apenas os pontos 2D fornecidos pelo *tracker*.

Além disso essa dissertação propõe acrescentar no *pipeline* de reconstrução 3D um método robusto e com baixo custo computacional afim de reduzir o erro causado pelo *tracker* mantendo sempre que possível o tempo total de execução inalterado.

1.4 RELEVÂNCIA

Há algum tempo, a área de Reconstrução 3D vem sendo alvo de pesquisa e divulgação na comunidade científica. Problemas que antes se punham contra as restrições de tempo real agora podem ser remodelados e solucionados para tender estas restrições, como no caso dos algoritmos de visão computacional, especificamente focando em reconstrução 3D.

Propor modificações no *pipeline* original dessas técnicas, ou se for necessário criar um novo *pipeline*, pode ser uma contribuição para a comunidade científica de visão computacional, permitindo que novos algoritmos sejam utilizados neste contexto.

1.5 PRINCIPAIS CONTRIBUIÇÕES

Esta dissertação tem suas principais contribuições na área de rastreamento e teste de hipóteses, e destacam-se as seguintes:

- Desenvolvimento de um teste de hipótese estatístico (teste *kens*) para quantificar a suavidade de uma função;
- Aproximação do triângulo de Pascal por uma distribuição Normal;
- Metodologia para aplicação do teste *kens* ao *pipeline* de reconstrução 3D;
- Prototipação da metodologia utilizando o software estatístico R;
- Implementação da metodologia na linguagem C;

Publicações realizadas no período desta dissertação foram:

- Towards a Standalone Methodology for Robust Algorithms Evaluation: A Case Study in 3D Reconstruction. SIBGRAPI 2011 [35];
- Avaliação de Algoritmos de Estimativa de Pose para Reconstrução 3D e Realidade Aumentada. WRVA 2011 [9].
- Uma avaliação estatística do problema de registro de imagens de profundidade usando o Kinect. WRVA 2011.

1.6 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação de mestrado engloba seis capítulos incluindo este que já foi descrito. O capítulo 2 contemplará todo o ferramental teórico necessário para o entendimento da dissertação, onde serão expostos os conceitos de rastreamento, fluxo ótico, o *pipeline* de reconstrução 3D e os conceitos estatísticos básicos de teste de hipótese.

No capítulo 3 serão explicitadas algumas técnicas do estado da arte, principalmente detecção e remoção de *outliers*.

O quarto capítulo apresentará a metodologia proposta nesta dissertação - o teste *kens*. Neste capítulo serão descritos os conceitos matemáticos e estatísticos necessários para a utilização da técnica. Também serão demonstradas as fórmulas e aproximações que foram realizadas.

O penúltimo capítulo mostra a aplicação do teste *kens* em dois exemplos, o primeiro deles utiliza um vídeo sintético enquanto o segundo usa um vídeo real. Os resultados em relação à quantidade de erro *frame* a *frame* e o tempo de execução do *kens* são apresentados e ilustrados através de gráficos.

Por fim, o último capítulo irá explicitar as vantagens e conclusões da proposta desta dissertação.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

O objetivo deste capítulo é descrever alguns conceitos básicos da matemática e estatística necessários para facilitar o entendimento da metodologia proposta nesta dissertação. Serão abordados os conceitos relacionados a rastreamento, o arcabouço do *pipeline* de reconstrução 3D, inferência estatística e fatoração.

Estes tópicos são a base para duas áreas especificas: a implementação dos algoritmos presentes no *pipeline* de reconstrução 3D e a construção de um teste de hipótese estatístico.

O leitor que já se sente familiarizado com os assuntos abordados ao longo deste capítulo poderá concentrar sua leitura nos capítulos seguintes.

2.1 RASTREAMENTO

Uma das áreas que despertam estudos em visão computacional é área de rastreamento (*tracking*), pois é a etapa inicial de qualquer processo que envolve sequências de imagens. O rastreamento é a técnica utilizada para detectar um objeto no plano da imagem enquanto este se move durante uma cena [64]. Para captura de *features* é necessário detectar como o objeto em análise será representado, quais as características que servirão de parâmetros e qual a estratégia utilizada para o rastreamento [64]. Serão abordados aqui os conceitos e técnicas de rastreamento utilizados ao longo da pesquisa que motivou esta dissertação.

2.1.1 Fluxo Ótico

O movimento é uma poderosa característica em sequências de imagens, dessa forma é revelada a dinâmica da cena pela relação espacial das características da imagem de acordo com a mudança temporal [12]. A tarefa de análise de movimento permanece um desafio e um problema fundamental em visão computacional [27]. A sequência temporal em processamento visual é importante, pois o movimento aparente de objetos sobre um plano de imagem é uma forte sugestão visual para entender a estrutura e movimento 3D e isso é necessário para bem inferir propriedades do mundo real.

Fluxo óptico pode ser definido como a distribuição 2D da velocidade aparente do movimento dos padrões de intensidade no plano da imagem. Em outras palavras, o campo do fluxo óptico consiste de um campo denso de velocidade onde cada pixel no plano da imagem está associado com um único vetor de velocidade e se for conhecido o intervalo de tempo entre duas imagens consecutivas, os vetores da velocidade podem ser convertidos em vetores de deslocamento e vice-versa [50] descrevendo assim a trajetória da *feature* pela cena (Figura 2.1).

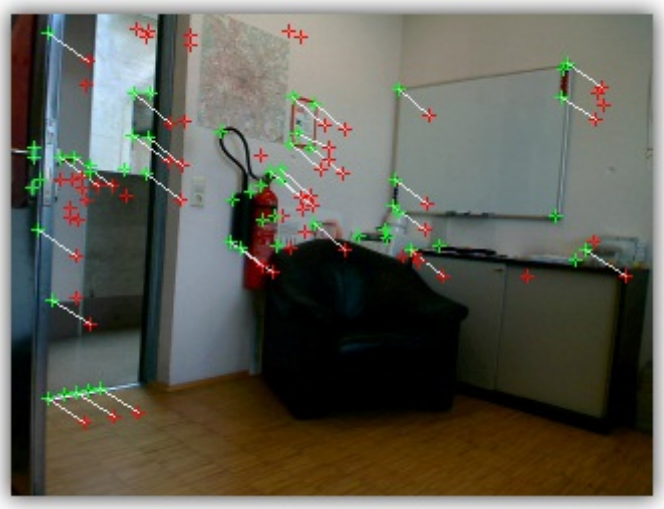

Figura 2.1 Fluxo Óptico de *Pixels*

A computação do fluxo óptico é um problema fundamental no processamento de sequências de imagens [1] e pode auxiliar em várias tarefas, tais como interpretação de cena, navegação exploratória, segmentação de objetos, codificação de vídeo, visão de robôs etc.

Para essa dissertação serão trabalhadas cenas estáticas com movimentação de câmera, o que caracteriza um problema mais complexo que a detecção de movimento [2]. Outros problemas mais complexos incluem a detecção de objetos (se movendo em um ambiente com vários objetos), a detecção da trajetória desses objetos e a predição das trajetórias.

A abordagem que será utilizada nesse trabalho é analisar o movimento a partir da computação do fluxo óptico onde existe uma pequena diferença de tempo entre as imagens e não devem ocorrer mudanças significativas entre duas imagens consecutivas. A computação do fluxo óptico nesse caso resulta numa estimativa da trajetória das *features* na imagem.

O fluxo óptico nem sempre corresponde ao verdadeiro campo de movimento por causa de alguns fatores que podem prejudicar as estimativas como distorção por rotação, borramento por movimentação, distorção por escala, oclusão parcial ou distorção dada por mudança da iluminação (Figura 2.2).

O principal problema com o cálculo do fluxo óptico é que todos os parâmetros estimados se baseiam em dados ruidosos saídos do rastreador, para o caso real é impossível medir a distribuição de probabilidade exata dos erros.

Dependendo da saída gerada pelo rastreador, os valores estimados e os fluxos reais podem apresentar diferenças bastante significativas. Cenas que apresentam bastante quantidade de ruído comprometem também a variância e por consequente prejudica a inferência em testes estatísticos e intervalos de confiança [35]. Outro grande problema nessa área é a questão do desempenho e a busca de algoritmos que atendam as condições de tempo real.

Figura 2.2 Exemplos de dificuldades encontradas no rastreamento de sequências de imagens: (a) cena original; (b) distorção por rotação; (c) borramento por movimentação; (d) distorção por escala; (e) oclusão parcial; (f) e (g) distorção por mudança da iluminação.

2.1.2 Rastreamento de *Features*

Muitos rastreadores de *features* vendo sendo propostos nessa área. O rastreamento é a identificação do posicionamento e/ou orientação de um determinado elemento ao longo do tempo e no âmbito de reconstrução 3D, para um detector de *features* ser considerado eficiente é necessário que seja capaz de identificar posições com sub-pixels de precisão.

Essa precisão é necessária por dois motivos, o primeiro é que deslocamentos pequenos na imagem muitas vezes podem resultar em erros grandes no espaço 3D (Figura 2.3).

O segundo motivo dessa busca por precisão se deve ao fato de que as posições das *features* são também a entrada principal para o *pipeline* de reconstrução 3D. Uma medida para medir essa precisão é o erro de projeção [20] e é dado pela equação 2.1:

$$E_{repro} = \sum_{x} d^2 \left(x, x'\right) \tag{2.1}$$

em que x é a posição da *feature* retornado pelo detector, x' é a posição do ponto na imagem associado ao ponto 3D após a reconstrução. A medida de distância d é a distância euclidiana e a soma é feita sobre todos os pontos da cena.

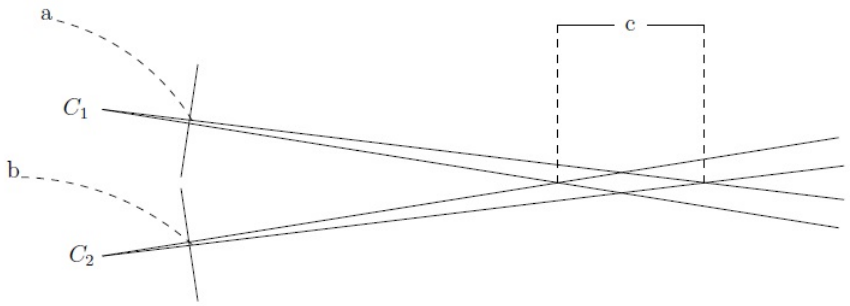

Figura 2.3 Influência do erro 2D no espaço 3D

Um conjunto de pontos em imagens distintas que representam a mesma *feature* é denominado *track*. Existem muitas técnicas e algoritmos destinados ao rastreamento de *features*. A maioria dos rastreadores escolhe suas *features* em áreas que representem arestas e cantos, pois são mais fáceis de rastrear. Escolher pontos facilmente identificáveis contribui para uma maior precisão no cálculo do posicionamento e diminui o custo computacional.

Existem vários fatores que dificultam o rastreamento. Dentre eles estão transformações afins (translação, rotação, escala e deformação), compressão das imagens, oclusões, distorções na lente, distorção de perspectiva e *motion blur* (borramento por movimentação). Exemplos desses problemas estão ilustrados na figura 2.2.

2.1.3 KLT

Em se tratando de fluxo ótico, um dos *trackers* mais utilizados é o KLT(Kanade-Lucas-Tomasi) [34]. O fluxo ótico (como decrito na seção 2.1.1) caracteriza-se por utilizar imagens geradas por sequência de vídeo, portanto mantém uma similaridade entre os quadros consecutivos além de possuir uma coerência temporal associada à ordem de exibição das imagens. O rastreamento ocorre basicamente em duas etapas:

- detecção de *features*;
- identificação ao longo dos quadros subsequentes.

Na fase de detecção de *features* utiliza-se o algoritmo GFTT (Good Features to Track) [49]. Este algoritmo é inspirado no detector de Harris [18], que extrai a informação de arestas e cantos presentes na imagem utilizando o operador de gradiente. A partir dos gradientes na direção horizontal e vertical é montada para cada pixel uma matriz de dimensão 2x2 denominada matriz de Harris. O detector de Harris baseia-se em um princípio central: em um canto, a intensidade da imagem vai mudar, em grande parte, em múltiplas direções. Isto pode ser alternativamente percebido através de uma análise nas alterações de intensidade devido a mudanças na janela de um local. Seguindo esta intuição, o detector de Harris utiliza a matriz de segundo momentos como base de suas decisões de canto e é intimamente ligado ao gradiente da imagem. A matriz A também pode ser chamada de matriz de auto correlação (Equação 2.2):

$$A(x) = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_y I_x \\ I_x I_y & I_y^2 \end{bmatrix} \tag{2.2}$$

assuma uma imagem I bidimensional em escala de cinza, I_x e I_y são as derivadas na direção de x e y de um ponto X na imagem respectivamente. Assuma também que $w(x,y)$ é uma função de peso que pode ser uniforme porém é mais típico ser usado uma gaussiana.

Analisando λ_x e λ_y, os autovalores da matriz A, a identificação da *feature* pode ser feito da seguinte forma:

- se λ_x e λ_y forem altos, a *feature* se trata de um canto;
- se um dos λ 's for alto e outro $\lambda \approx 0$, a *feature* se trata de uma borda;
- se $\lambda_x \approx 0$ e $\lambda_y \approx 0$ então a *feature* não é interessante.

Em outras palavras, quando os dois autovalores são representados por valores altos e positivos, a *feature* faz parte de um canto, sendo assim de fácil rastreio. Quando só um dos valores é alto (e o outro é próximo de zero) a *feature* faz parte de uma aresta, que também pode ser rastreada com facilidade, porém os cantos são as melhores opções. Quando os dois autovalores se aproximam de zero, o ponto não é considerado como *feature*.

A diferença entre o GFTT e a técnica de Harris pura está na definição do coeficiente de rastreabilidade (M_c calculado para cada pixel). Em ambas as técnicas, os coeficientes são calculados utilizando os dois autovalores da matriz de Harris.

Tendo observado que a extração dos autovalores é bastante custosa para ser feita para cada pixel da imagem, Harris propõe uma função de pontuação baseada nos autovalores (λ_x e λ_y) que pode ser computada sem extraí-los de fato. A equação 2.3 calcula o valor da função de Harris (M_c), onde A é a matriz de Harris e k representa o coeficiente de sensibilidade, que na literatura assume valores entre 0.04 e 0.06 [39]:

$$M_c = \lambda_x \lambda_y - k(\lambda_x + \lambda_x)^2 = det(A) - k(trace(A))^2. \tag{2.3}$$

Como já mencionado anteriormente, a diferença entre a extração de features na técnica de Harris e no GFTT é a função de cálculo do M_c, e no GFTT é descrito pela equação 2.4:

$$Mc = min(\lambda_x, \lambda_y). \tag{2.4}$$

O KLT usa uma abordagem piramidal na fase de rastreamento, nessa etapa são calculadas subamostragens da imagem original em diversos níveis, sendo a altura da pirâmide regulada por parâmetro indicado pelo usuário. Também são calculados os gradientes da imagem nas direções X e Y. Os resultados desses cálculos são utilizados para buscar a nova posição do ponto no próximo quadro, onde a *feature* a ser procurada é o ponto que tem o menor autovalor no espaço de busca.

2.2 RECONSTRUÇÃO 3D

Como dito na seção 1.1, o *pipeline* de reconstrução 3D é dividido em seis fases. As primeiras quatro etapas podem ser denominadas como uma só e chamada de reconstrução esparsa. O produto final da reconstrução esparsa serão dois: uma nuvem de pontos 3D esparsamente reconstruída e as poses de câmera da cena bem como seus parâmetros estimados. O mais importante na reconstrução esparsa, na verdade, são as poses de câmera pois é com elas que serão triangulados e estimados os novos pontos e poder então realizar a reconstrução densa.

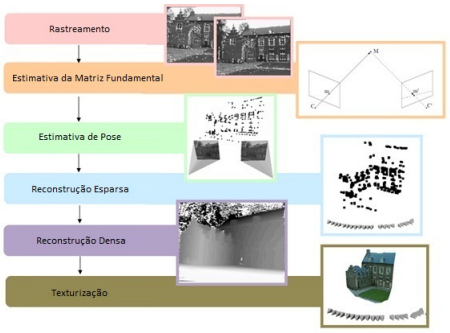

Figura 2.4 Pipeline de Reconstrução 3D

O *pipeline* de reconstrução pode se baseado em estrutura de movimento(Structure from Motion - SfM) [62] que é uma abordagem clássica para calcular a estrutura da cena e movimento de câmera assumindo que esta informação é desconhecida [40].

2.2.1 Calibração de Câmera

A partir de duas câmeras posicionadas de forma correta, com suas posições e direcionamentos conhecidos, é possível determinar a posição de qualquer ponto neste espaço, desde que este ponto possa ser localizado dentro de cada uma das imagens capturadas pelas câmeras [20]. Para isso, torna-se necessário que haja uma região comum, de proporção razoável, aparecendo em ambas imagens.

Embora haja tentativas de criar modelos a partir de apenas uma imagem [42],[47], os melhores resultados geralmente são obtidos utilizando mais imagens, estabelecendo correspondências e usando técnicas de triangulação de pontos 3D.

Para o processo de triangulação (Figura 2.5), além do posicionamento relativo das câmeras e das suas propriedades técnicas, devem também ser conhecidos os parâmetros extrínsecos e intrínsecos da câmera.

Em geral, existem duas abordagens para obter estas informações. A primeira é medir os parâmetros da câmera usada antes de iniciar o procedimento de reconstrução. Este procedimento é chamado de calibração [59],[65]. Por um lado, uma câmera já calibrada simplifica substancialmente o processo de reconstrução, mas por outro lado, impõe inflexibilidade pois suas configurações não podem ser alteradas.

Outra abordagem é tentar calcular os parâmetros de câmera juntamente com os pontos

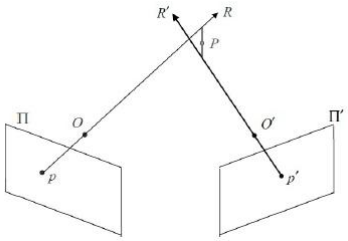

Figura 2.5 Triangulação

3D, apenas confiando nas correspondências estabelecidas entre as imagens observadas. Mais detalhes sobre esses sistemas e suas melhorias podem ser encontrados em [20],[32], [48],[41],[60].

Em comparação com a abordagem quando a câmera já está calibrada, esta abordagem projetiva tem um maior custo computacional e é menos precisa. Além disso, é necessário o uso de técnicas de auto calibração [20] afim de eliminar a distorção projetiva.

2.2.2 Reconstrução Esparsa

O processo de reconstrução 3D se inicia a partir da coleta de um número de imagens do mesmo objeto. Essas imagens devem conter as características de interesse para a modelagem. É feita então a extração de pontos característicos(*features*) na imagem e depois procura-se o casamento dessas *features* ao longo da sequência do video (Figura 2.6). Em outras palavras, uma *feature* posicionada na primeira imagem I_1 deve ser detectada na segunda imagem I_2 e assim sucessivamente.

Figura 2.6 Pipeline: Rastreamento

A uma sequência de posições de uma *feature* advindas de um conjunto de imagens chamamos correspondência. Se as câmeras e suas posições no espaço são conhecidas, é possível reconstruir os pontos 3D do objeto de interesse. Esse é o processo de triangulação e pode ser conseguido pela interseção dos raios dos centros da câmera através das correspondências espaciais das *features*, podendo pode ser usado apenas com imagens [23].

Embora haja também abordagens que trabalham sem correspondências dadas [10], as informações contidas nas correspondências em geral são suficientes para reconstruir pontos da superfície de objetos. Após encontrar as correspondências em cada imagem, a fase seguinte é a estimativa da Matriz fundamental (Figura 2.7).

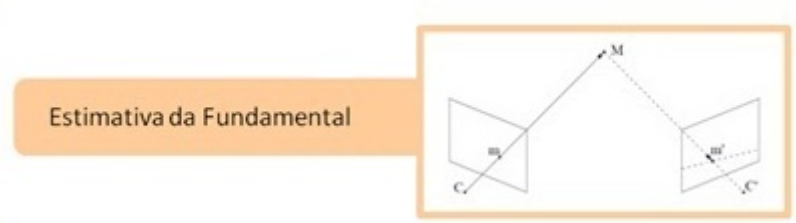

Figura 2.7 Pipeline: Matriz Fundamental

Essa matriz é responsável pela relação entre quaisquer duas imagens na mesma cena, desde que tenham pontos projetados em ambas as imagens e é definida pela seguinte equação:

$$x'^T F x = 0 \tag{2.5}$$

onde x = $(x, y, 1)^T$ e x' = $(x', y', 1)^T$ são as coordenadas dos pontos correspondentes em duas imagens e F é a própria matriz fundamental. Para o cálculo da matriz fundamental é necessário que existam pelo menos 7 correspondências.

A forma mais simples de calcular a matriz fundamental é usando o algoritmo chamado 8-points [33] que necessita apenas de uma resolução de sistemas lineares. Atualmente é mais comum normalizar os dados, aplicar o algoritmo 8-points e depois desnormalizar. Esse processo se chama normalized 8-points [20]. Apesar desse algoritmo necessitar apenas de 8 correspondências, é mostrado em [19] que os resultados são melhores se usarmos como entrada nove ao invés de oito pontos.

Depois do cálculo da matriz fundamental, a terceira fase do *pipeline* consiste na estimativa da matriz de pose de câmera (Figura 2.8). A matriz de câmera é uma transformação composta de parâmetros intrínsecos (distância focal, skew, ponto principal, etc.) e extrínsecos (a própria da pose) da câmera. A dimensão da matriz de câmera é 3x4 e é composta por uma rotação e uma translação.

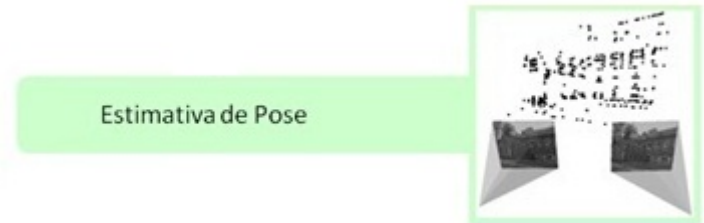

Figura 2.8 Pipeline: Pose de câmera

Existem na literatura muitos algoritmos que se propõem a estimar a matriz de câmera. Algumas buscam a solução algébrica através de um sistema linear, como o DLT [20], outros utilizam um argumento geométrico para descobrir a pose, como o POSIT [11], PNP [44] e EPNP [31]. Uma avaliação desses algoritmos podem ser vista em [9].

De posse da matriz de câmera, os pontos 3D pode ser reconstruídos por triangulação (Figura 2.9) através dos pontos correspondentes em cada grupo de imagem e da correspondente pose de câmera.

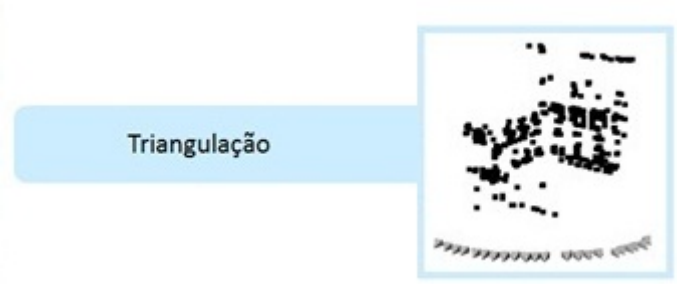

Figura 2.9 Pipeline: Triangulação

Com os pontos devidamente triangulados, será utilizado então técnicas de *Bundle Adjustment*(BA) [20] [58] que são quase sempre utilizadas como um dos últimos passos dos algoritmos de visão computacional.

Tendo como entrada as estimativas iniciais dos parâmetros de câmera e uma nuvem de pontos (obtida a partir de vários pontos de vista) o algoritmo BA pode refinar simultaneamente o movimento e estrutura, minimizando o erro reprojeção entre os pontos de imagem observados e previstos. O algoritmo mais utilizado atualmente é o *Sparse Bundle Adjustment*(SBA) e pode ser obtido em [29]. A minimização de sistemas neste algoritmo é tipicamente realizada com o auxílio do algoritmo de Levenberg-Marquardt(LM) [38].

Quando se trata da minimização do erro, a distribuição probabilística deste erro é bastante importante na elaboração da solução do problema. Na maior parte do tempo estima-se o ruído na imagem como sendo Gaussiano de média zero. Sendo assim, BA pode ser representado pelo estimador de máxima verossimilhança(MLE - Maximum Likelihood Estimator) [4].

No entanto, devido ao grande número de operações envolvidas, a aplicação do LM incorre em custos computacionais, quando aplicada ao problema de minimização definido no contexto de BA e assim, com a aplicação do SBA, o limite do tempo real não pode ser alcançado.

2.2.3 Reconstrução Densa

Após ter sido gerada a reconstrução esparsa, entra-se na fase da reconstrução densa quando adicionam-se pontos à reconstrução para preencher o objeto reconstruído.

A reconstrução densa (Figura 2.10) é o processo no qual uma nuvem esparsa de pontos 3D em conjunto com os parâmetros extrínsecos das câmeras envolvidas na reconstrução são utilizados para gerar uma nuvem de pontos 3D em que cada pixel das imagens corresponde a um ponto 3D da nuvem. Este efeito é alcançado através de melhoramentos no rastreamento das *features* utilizando informações da geometria epipolar. Com estas informações é possível rastrear mais *features* e consequentemente melhorar a precisão da nuvem densa a ser criada.

Figura 2.10 Pipeline: Reconstrução Densa

Entre as diversas técnicas de reconstrução densa pode ser citado o plane sweep [8] e o multi-view stereo [15]. A principal diferença é que o plane sweep utiliza apenas a visão de uma câmera enquanto que o multi-view stereo utiliza a informação de várias câmeras calibradas para adicionar um casamento massivo de *features*.

Uma abordagem mais aprofundada dessas técnicas podem ser encontradas em [13]

2.2.4 Texturização

Finalmente, a última fase consiste em gerar a textura das imagens tornando-o o modelo 3D reconstruído como mostra a figura 2.11.

Figura 2.11 Pipeline: Texturização

2.3 INFERÊNCIA ESTATÍSTICA

Em geral, intervalos de confiança são a forma mais informativa de apresentar os achados principais de um estudo. Contudo, algumas vezes existe um particular interesse em verificar determinadas afirmações ou conjecturas.

Em muitas situações temos interesse em tomar a decisão de aceitar ou rejeitar determinada afirmação baseando-se em um conjunto de evidências. Isto é, feita uma determinada afirmação sobre uma população, usualmente sobre um parâmetro desta, desejamos saber se os resultados de uma amostra contrariam ou não tal afirmação.

Posto que não se pode descartar a possibilidade da ocorrência de erros, as técnicas tem que ser cuidadosamente estudadas e seguidas suas suposições e restrições.

2.3.1 Definições Básicas

Serão formalizado nessa seção os princípios básicos do estudo de inferência. A seguir será apresentado a noção de hipótese estatística.

Definição 1. *Uma hipóteses estatística é uma afirmação ou conjectura sobre o parâmetro, ou parâmetros, da distribuição de probabilidades de uma característica, X, da população ou de uma v.a.*

Denotado por H_0(Hipótese nula) a hipótese de interesse. Caso H_0 seja rejeitada, aceita-se como verdade a alternativa H_1. Sendo a variável aleatória X distribuída de acordo com a função de densidade(ou de probabilidade) $f(x|\theta)$, com $\theta \in \Theta$, é dito que a distribuição de X está totalmente especificada quando se tem $f(x|\theta)$ e θ e parcialmente especificada quando se tem $f(x|\theta)$ e não tem θ.

Associados às hipóteses H_0 e H_1, são definidos os conjuntos Θ_1 e Θ_2, ou seja, H_0 afirma que $\theta \in \Theta_0$(formalmente $H_0 : \theta \in \Theta_0$) e H_1 afirma que $\theta \in \Theta_1$(formalmente $H_1 : \theta \in \Theta_1$).

No caso que $\Theta_0 = \{\theta_0\}$ diz-se que H_0 é simples, caso contrário, composta. Da mesma forma vale para H_1.

2.3.2 Teste de Hipótese

Para a construção de um teste de hipótese, para um parâmetro populacional, primeiramente precisa-se definir essa situação: uma variável X em uma dada população e uma hipótese sobre algum parâmetro θ dessa população.

Tendo essas duas premissas colhe-se uma amostra dessa população e calcula-se o parâmetro de interesse e chama-o de θ_0. Tendo calculado o parâmetro amostral, explicita-se claramente a hipótese colocada à prova. Essa hipótese é chamada de hipótese nula.

Em seguida deve-se explicitar também a hipótese que, caso H_0 seja rejeitada, será considerada verdadeira, a essa é chamada de hipótese alternativa, ou H_1. Um exemplo de como é escrita formalmente está em 2.6.

$$\begin{aligned} H_0 &: \theta = \theta_0 \\ H_1 &: \theta \neq \theta_0 \end{aligned} \tag{2.6}$$

Qualquer decisão que seja tomada está sujeita a erros. Os riscos α e β são conhecidos na literatura [4] como erros do tipo I e tipo II respectivamente. Mais precisamente, o erro do tipo I ocorre se rejeitamos H_0 quando H_0 é verdadeira e o erro do tipo II é não rejeitar H_0 quando H_0 é falsa. A tabela 2.1 ilustra de maneira resumida.

Tabela 2.1 Tipos de erros em testes de hipóteses

Decisão	H_0 é verdadeira	H_0 é falsa
Aceitar H_0	Decisão Correta	Erro do tipo II
Rejeitar H_0	Erro do tipo I	Decisão Correta

Então de maneira mais formal, pode-se definir α e β como:

$$\alpha = \text{P(erro do tipo I)} = \text{P(Rejeitar } H_0 | H_0 \text{ é verdadeira)} \tag{2.7}$$
$$\beta = \text{ P(erro do tipo II)} = \text{P(Não rejeitar } H_0 | H_0 \text{ é falsa)} \tag{2.8}$$

O objetivo do teste de hipótese é afirmar, através da estatística $\hat{\theta}$ obtida de uma amostra, se a hipótese H_0 é aceitável.

Em todo teste estatístico existe uma região que é denominada região crítica [5], caso o valor observado da estatística $\hat{\theta}$ pertença à região crítica, H_0 é rejeitada em prol de H_1. Essa região crítica é construída de modo que:

$$\alpha = P(\hat{\theta} \in \text{região crítica} | H_0 \text{ é verdadeira}) \tag{2.9}$$

A probabilidade α é uma valor arbitrário e recebe o nome de nível de significância do teste. Quanto menor o valor para α mais significante é rejeitar H_0. Em outras palavras, o nível de significância α de um teste é a probabilidade máxima com que deseja-se correr o risco de um erro do tipo I.

Usualmente se usam valores para α fixados em 5%, 1% ou 0,1%, mas não há nada que justifique formalmente o uso destes valores em particular [5].

Um enfoque alternativo(e mais comumente utilizado na comunidade estatística) consiste em calcular o menor nível de significância para o qual H_0 é rejeitada, para o valor observado da estatística de teste. Esta quantidade é chamada nível critico, probabilidade de significância ou p-valor.

Definição 2. *O p-valor é a probabilidade sob H_0 de a estatística de teste tomar valores tão ou mais desfavoráveis à hipótese nula do que o seu valor observado, ou seja, o menor nível de significância que conduz à rejeição de H_0.*

O cálculo do p-valor permite que não seja necessário fixar previamente o nível de significância de um teste de hipótese. A ideia é que, após calcular o p-valor o pesquisador possa escolher o seu próprio nível de significância como sendo a probabilidade máxima tolerável para um erro tipo I.

2.3.3 Etapas do Teste de Hipótese

Qualquer teste de hipóteses paramétrico segue os seguintes passos:

- **Formular as hipóteses:** Estabelecer as hipóteses nula e alternativa.
- **Estabelecer a estatística a ser utilizada:** Após fixar as hipóteses é necessário determinar se a diferença entre a estatística amostral e o suposto valor do parâmetro da população é suficiente para rejeitar a hipótese. A estatística utilizada deve ser definida e sua distribuição teórica determinada.
- **Fixar o nível de significância do teste:** Fixar a probabilidade de ser cometer erro do tipo I, isto é, estabelecer o nível de significância do teste.

- **Calcular a estatística teste:** Através da amostra obtida calcular a estimativa que servirá para aceitar ou rejeitar a hipótese nula.

- **Tomar a decisão:** Se o valor da estatística estiver na região crítica rejeitar H_0, caso contrário, não rejeitar H_0.

- **Formular a conclusão:** Com base na aceitação ou rejeição da hipótese nula, enunciar qual a decisão a ser tomada na situação do problema.

2.4 FATORAÇÃO

Métodos de fatoração são populares em visão por computador [54], [55] e [36]. As aplicações mais populares de métodos de fatoração incluem a recuperação da estrutura do movimento (SFM) [55], separação de estilo e conteúdo [54] e decomposição de peças locais [36].

A utilização de técnicas de fatoração difere em aspectos tais como a forma e como fatores são extraídos, interpretados e modelados. Tomasi e Kanade [55] recuperaram a estrutura da cena e movimento de câmera a partir de uma sequência de imagens tiradas com uma câmera em movimento. Essas matrizes são fatoradas usando Decomposição de Valor Singular(SVD) [26].

Uma das ideias utilizadas em trabalhos sobre detecção de *outliers* é utilizar a própria característica de quadros em sequência e assim analisar o caminho que a *feature* percorre (Figura 2.12).

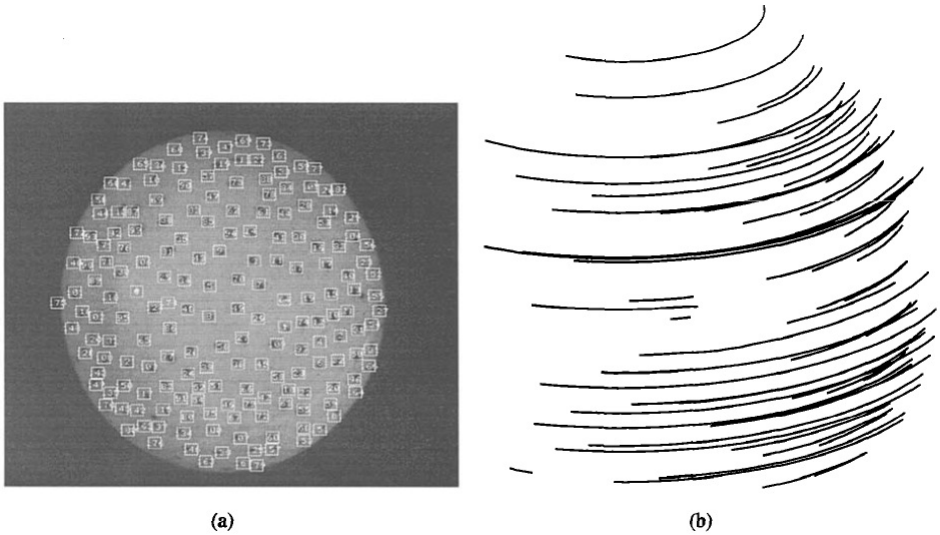

Figura 2.12 (a) *features* (b) caminhos das *features* ao longo dos quadros

Em [56] foi introduzido o método de fatoração, entre vários detalhes desse trabalho, um destaque é a forma de representação da matriz de pontos 2D, que trata as *features* como caminhos no eixo X(horizontal) e eixo Y(vertical) separadamente.

Essa forma de representar será utilizada nesse trabalho com uma pequena alteração na notação, mudança essa feita apenas por conveniência do problema. Como será utilizada no desenvolvimento da pesquisa é necessária ser definida formalmente, suponha que foram rastreadas P *features* ao longo de F quadros de uma cena. Assim são obtidas as trajetórias das coordenadas da imagem na forma:

$$(U_{pf}, V_{pf})|p = 1, \ldots, P; f = 1, \ldots, F \tag{2.10}$$

onde as coordenadas das *features* são escritas horizontalmente em uma matriz do tipo PxF. Tem-se então duas matrizes U e V que representam o caminho que as *features* fazem no eixo X(matriz U) e caminho que as *features* fazem no eixo Y(matriz V). As matrizes U e V pode ser visualizadas abaixo:

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,F} \\ u_{2,1} & u_{2,2} & \cdots & u_{1,F} \\ \vdots & \vdots & \vdots & \vdots \\ u_{P,1} & u_{P,2} & \cdots & u_{P,F} \end{bmatrix}$$

$$V = \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,F} \\ v_{2,1} & v_{2,2} & \cdots & v_{1,F} \\ \vdots & \vdots & \vdots & \vdots \\ v_{P,1} & v_{P,2} & \cdots & v_{P,F} \end{bmatrix}$$

A ideia de se trabalhar com as matrizes U e V é que as linhas dessas matrizes formam uma função no tempo e podem ser analisadas como tal [61].

CAPÍTULO 3

ESTADO DA ARTE

Este capítulo descreve algumas técnicas relacionadas à área de remoção de *outliers*. Durante o tempo deste mestrado foi feita uma pesquisa bibliográfica extensiva e não foi encontrada nenhuma técnica similar à apresentada nesta dissertação, ou seja, remoção de *outliers* baseada em suavidade de caminho .

Contudo, existem na literatura, outras técnicas que também se propõem a remover *outliers*. Historicamente, existem dois tipos de métodos para remoção de *outliers* [63]; o primeiro usa *Relaxation Iteration* [66], [6] enquanto o segundo é baseado em RANSAC [14]. O método de *Relaxation Iteration* adota um esquema de correspondência de *features* utilizando aplicação de geometria epipolar para remover *outliers* discrepantes. Esse procedimento é repetido iterativamente até que resultados satisfatórios sejam obtidos, no entanto, este método é dispendioso computacionalmente [63]. O método baseado em RANSAC é um método iterativo robusto para estimar os parâmetros de um modelo matemático. O funcionamento do RANSAC está descrito na seção 4.1.

3.1 TENSOR TRIFOCAL ROBUSTO

O tensor trifocal é uma matriz 3x3x3 que incorpora todas as relações projetivas geométricas entre três pontos de vista [20]. É responsável por relacionar as coordenadas de pontos correspondentes e é independente da estrutura da cena, dependendo apenas do movimento relativo (isto é, pose) entre os três pontos de vista e os seus parâmetros de calibração intrínsecas.

Assim, o tensor trifocal pode ser considerado como a generalização da matriz fundamental em três vistas [20]. Nota-se que, apesar do tensor ser composto por 27 elementos, apenas 18 deles são realmente independentes.

Para estabilizar o rastreamento, restrições adicionais devem ser usadas. As restrições epipolares derivadas de sequências temporais [22] são menos restritivas, porém mesmo assim ainda recaem em *outliers* nas linhas epipolares. Portanto, para um rastreamento consistente cada *feature* é verificada ao longo de três *keyframes* sucessivos utilizando o tensor trifocal.

Esse tensor trifocal implica em restrições geométricas sobre todos os pares dentro de um trio de imagens. Na prática, ele transfere um par de pontos correspondentes na terceira imagem onde a sua posição pode ser verificada. Apenas se essa *feature* atender às restrições do tensor ela é considerada como um possível *inlier*.

O tensor trifocal pode ser calculado com pelo menos seis pontos de correspondências em três imagens [22]. Se todos os pontos estão em um mesmo plano, o tensor não pode ser calculado corretamente.

Portanto, um teste de homografia sobre todos os *inliers* que são consistentes com o tensor trifocal é sugerido. Para obter uma estimativa robusta do tensor trifocal, pode se

usar o GASAC [45] que uma é outra abordagem baseada no RANSAC.

Dependendo do ruído, um tensor derivado de um conjunto com poucos pontos pode conduzir a distâncias epipolares grandes entre os *inliers*. Para resolver esse problema uma otimização não linear deve ser considerada como um passo final, ou pode-se também usar um conjunto sobredeterminado para o tensor trifocal proposto em [21].

O algoritmo 1 demonstra o passo-a-passo resumido para a computação do tensor trifocal robusto.

Algoritmo 1 Tensor Trifocal Robusto

1. Estimar o tensor trifocal com o GASAC usando 6 pontos
2. Determinar todos os *inliers* de acordo com o tensor
3. Testar se os *inliers* definem uma homografia?

Se Sim **então**
    Remova 2/3 das *features* planares dos dados
    Testa novamente com 6 pontos
    Se Sim **então**
        Volte para o passo 1 com o banco de dados reduzido
    Senão
        Use a homografia com todos os *inliers*
    Fim Se
SenãoTensor trifocal é válido
Fim Se

4. Opcional: Usar otimização não linear em todos os *inliers*

Embora o tensor trifocal robusto seja uma eficiente ferramenta para detectar *outliers*, este método apresenta algumas desvantagens:

- GASAC é mais lento que o RANSAC;
- depende da quantidade de ruído;
- o processo como um todo é lento pois necessita de testes de homografia e otimização não linear.

3.2 TRAOD

O algoritmo TRAOD(TRAjectory Outlier Detection) foi proposto por [30]. Esse trabalho assim como esta dissertação, visa detectar as *features outliers* pelas suas trajetórias.

Este algoritmo tem como entrada um conjunto de trajetórias $I = TR_i$ onde i é a quantidade de *features* na cena, e terá como saída um conjunto de *outliers* . O TRAOD também necessita ter como entrada três parâmetros: D(distância máxima), p e F que são dados pelo usuário.

Uma t-partição é uma abreviação para partição da trajetória. A t-partição é um segmento de reta $p_ip_j(i < j)$, onde p_i e p_j são pontos escolhidos na mesma trajetória

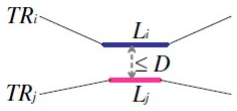

Figura 3.1 Conceito de t-partição

e é considerada distante se ela não for suficientemente "semelhante" à t-partição das trajetórias vizinhas a ela (Figura 3.1).

A primeira fase consiste em particionar as trajetórias em base unitária. Uma base unitária é definida em [30] como sendo a menor distância significativa naquele cenário, ou seja, é uma unidade de medida dependente do experimento. O próximo passo do algoritmo é a fase de detecção. Nessa etapa, o algoritmo encontra as t-partições e as usa para detectar as trajetórias. Após a detecção um conjunto de *outliers* é dado como saída.

Esse trabalho apresenta como desvantagem a escolha dos parâmetros. Todos os parâmetros são empíricos e dependem do experimento. Os autores sugerem valores para p e F devido à a experiência com o algoritmo [30] mas não definem nenhum valor para D, alegando a sensibilidade do parâmetro.

3.3 HEI

O algoritmo HEI (Heterogeneity Elimination Individually) proposto por [63] é um método de detecção de *outliers* que utiliza fluxo ótico. O HEI calcula a variância e a distância média entre os pontos rastreados e os pontos projetados. Com esses dados é construído um círculo de confiança (Figura 3.2) e quem estiver fora é considerado como *outlier*. Esse círculo de confiança é delimitado por um *threshold* definido pelo usuário.

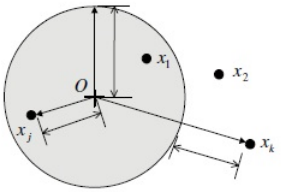

Figura 3.2 Círculo de Confiança

A diferença principal entre HEI e RANSAC é que o último adota esquema de seleção aleatória. Como RANSAC é baseado em seleção e validação, alguns pontos próximos da fronteira *inlier/outlier* podem possivelmente ser validados de forma equivocada de acordo com a amostra aleatória escolhida. Enquanto que o HEI é um método determinístico, quando o *threshold* é determinado, as *features inliers* no final serão sempre as mesmas.

A desvantagem do HEI é precisar ter como entrada um parâmetro (o *threshold*) e ele só apresenta maior performance que o RANSAC para cenas com até 50% de *outls*. A

partir disso o RANSAC é mais rápido.

O experimento realizado em [63] indica tempo real porém utiliza apenas 27 *features* e os experimentos feitos nesta dissertação utilizam mais de mil pontos.

3.4 CONSIDERAÇÕES

As três técnicas já citadas apresentam alguns pontos negativos. O tensor trifocal robusto demanda esforço computacional, o HEI não apresenta bons resultados quando há uma grande quantidade de *outliers* e o TRAOD necessita da interação do usuário para entrar com parâmetros, ou seja, não é automático.

O teste *kens* proposto nesta dissertação, além de ser eficaz na remoção de *outliers*, apresenta as seguintes características:

- não é necessária a interferência do usuário para informar os parâmetros;
- funciona mesmo quando os dados apresentam muito *outliers*;
- o custo computacional não é elevado.

Todo o ferramental teórico para entendimento do teste *kens* e suas características serão descritos no capítulo a seguir.

CAPÍTULO 4

PROPOSTA: TESTE *KENS*

A reconstrução 3D exige várias áreas específicas de conhecimento e que são necessárias em todas as etapas do *pipeline*, desde a aquisição das imagens, até o produto final, que é caracterizado pela nuvem de pontos e pelos parâmetros das câmeras que compõem a cena. Além dos parâmetros de calibração das câmeras, os parâmetros extrínsecos (pose formada pela posição e orientação das câmeras) também são recuperados com a reconstrução 3D.

O que acontece na prática é que a cada estágio do pipeline ocorre normalmente uma corrupção acumulativa dos dados da reconstrução 3D. Desde o início, na aquisição da imagem, dependendo das condições de iluminação e da resolução do sensor da câmera, o processamento provavelmente já estará comprometido por ruídos quando repassado para o próximo estágio do pipeline. Na fase de rastreamento, diversos fatores como oclusão de *features* (características), falsos casamentos, *drift* devido a áreas da imagem pobres em textura ou com gradientes pouco significativos, também inserem ruídos no posicionamento das *features* ao longo das *tracks*.

Um *outlier* é um objeto dentro de um banco de dados que é grosseiramente diferente ou inconsistente com os demais objetos nesse banco de dados [17].

As *features inliers* sempre seguem caminhos condizentes com a cena enquanto que *features outliers* em geral não seguem. Se todas as *features* em uma determinada região da cena seguem um padrão de trajetória e uma não segue, a *feature* que a gerou tem grande chance de ser um *outlier*. Um exemplo de trajetória *outlier* pode ser visto na figura 4.1.

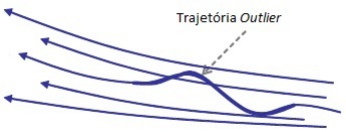

Figura 4.1 Trajetória Outlier

Uma vez que, um *outlier* percorre um caminho ruidoso, a técnica proposta nesta dissertação propõe uma metodologia para identificar e ranquear as *features* em ordem de suavidade.

4.1 SELEÇÃO DE *FEATURES*

Como os dados extraídos do mundo real já vêm comprometidos (com ruídos), os cálculos de reconstrução não irão gerar o produto final esperado e sim uma hipótese (um produto intermediário). Um dos métodos para avaliação de hipóteses é o algoritmo RANSAC (Random Sampling Consensus) [14], que consiste em um método iterativo

para estimar parâmetros de um modelo matemático tendo como base um conjunto de observações que contém erros.

O seu funcionamento segue os seguintes passos, uma hipótese gerada a partir de dados é escolhida aleatoriamente e testada com o universo completo dos dados. Caso os testes aprovem a hipótese através de um limiar determinado pelo usuário, esta hipótese é guardada como possível candidata ao produto final. Depois de estimados um determinado número de hipóteses, a melhor hipótese, de acordo com fatores pré-definida no processo, é escolhida como o produto final.

As observações que forem fiéis ao modelo são denominadas de *inliers* e as que não satisfizerem ao conjunto de parâmetros pré-determinados são consideradas como *outliers*. Para determinar se uma observação é *inlier* ou *outlier*, compara-se o erro gerado pelas hipóteses com um limiar de tolerância determinado pelo usuário. O RANSAC puro utiliza como fator de avaliação final das hipóteses a contagem de *inliers*, dessa forma, é sempre escolhida a hipótese com o maior número de *inliers*.

Baseado no RANSAC, outros algoritmos surgiram, dentre eles o LMedS (Least Median Square) [46], cuja principal diferença consiste em como o LMedS calcula o desvio padrão robusto utilizado na definição do limiar de aceitação de *inliers*. O algoritmo LMedS, primeiro gera as hipóteses, acumula os residuais de cada uma das hipóteses para que possa ser calculado o desvio padrão e a mediana. Após este passo cada hipótese gera uma mediana dos residuais de todos os pontos do conjunto. A menor mediana é guardada e denominada M_j. O desvio padrão é calculado por:

$$\hat{\sigma} = 1.4826 \left(1 + \frac{5}{n-p}\right) \sqrt{M_j} \tag{4.1}$$

Este desvio padrão é importante para parametrizar o cálculo do *threshold*, que equivale a 2,5s (onde s é o desvio-padrão amostral). Após o cálculo do threshold o LMedS filtra os inliers para cada hipótese, e escolhe a melhor hipótese similar ao RANSAC, ou seja, aquela que contiver o maior número de inliers.

O LMedS é uma abordagem muito custosa computacionalmente quando comparada ao RANSAC, pois precisa percorrer as hipóteses duas vezes e também é necessária uma ordenação no momento do cálculo da mediana.

Atualmente existem vários algoritmos que como o LMedS, são ajustes, aprimoramentos ou releitura do RANSAC. Entre eles podem ser citados o StaRSaC [7], MSAC [57] e MLESAC [57].

4.2 MOTIVAÇÃO

Na área de reconstrução 3D é de vital importância a busca pelo tempo real. A inclusão de mais um passo no *pipeline* deve ser visto com cautela pois pode prejudicar o desempenho do sistema como um todo.

O RANSAC e suas variações são indispensáveis para o *pipeline* e já são soluções estáveis, o problema é que quando são adicionados muitas *features* igualmente é acrescido mais ruido e *outliers*, diminuindo assim a precisão do RANSAC e provavelmente fugindo do tempo real.

Para isso é necessário um filtro com baixo custo computacional afim de selecionar as melhores *features* e diminuir o tempo de execução do RANSAC.

A intenção não é suavizar os caminhos, isso poderia ser realizado utilizando curvas de Bézier ou B-Splines [43]. Aproximar as *features* a curvas suaves utilizando essas técnicas não garante a correspondências nos *frames* da sequência, sem contar que essa abordagem também gera custo computacional.

De mesma forma não é interessante modelar a série temporal pois para isso se fazem necessários cálculos de mínimos quadrados [61] o que leva à computação de matrizes inversas, gerando custos computacionais.

A ideia proposta por esta dissertação é fornecer como entrada para o RANSAC as melhores *features* com a menor quantidade de tempo possível. Uma vez que apenas *inliers* são dados como entrada o RANSAC será mais preciso e executará menos iterações melhorando assim a performance do *pipeline* inteiro.

Outro importante fator na técnica proposta é a não necessidade de introdução de parâmetros dados pelo usuário. Em diversas técnicas utilizadas na área é comum que alguns parâmetros sejam introduzidos fazendo com que o teste se comporte de diferentes formas dependendo das entradas dadas pelo o usuário(vide seção 3.2).

Para isso foi desenvolvido nesta pesquisa um teste estatístico de suavidade. As *features* serão ranqueadas pelo p-valor do teste *kens* e como cada algoritmo de reconstrução demanda uma quantidade mínima de *features* necessária, serão fornecidas ao RANSAC apenas a quantidade mínima de *features* de acordo com a ordem de suavidade.

4.3 CONSTRUÇÃO DO TESTE DE HIPÓTESE

Em estatística, um teste de hipóteses é um método para verificar se os dados amostrais são compatíveis com alguma hipótese que se tenha interesse de verificar.

Um Teste de Hipóteses pode ser paramétrico ou não-paramétrico. Testes paramétricos são baseados em parâmetros da amostra, por exemplo média e desvio padrão. O uso tanto dos testes paramétricos como dos não-paramétricos está condicionado à dimensão da amostra e à respectiva distribuição da variável em estudo.

Como visto na seção 2.3.2, para a construção do teste de hipótese precisa-se definir duas coisas: uma variável X em uma dada população e uma hipótese sobre algum parâmetro θ dessa população. Depois disso é necessário definir uma estatística de teste e uma regra de decisão.

4.3.1 Variável Aleatória

Para construir o teste precisa-se primeiro entender como se comporta a variável aleatória. Como visto em 2.4 podemos entender o caminho de uma *feature* como uma função.

Suponha duas *features* a e b em uma sequência de video. Em 10 *frames* a percorreu as coordenadas C_a =(1, 8, 12, 14, 13, 8, 4, 5, 8, 12) e b percorreu as coordenadas C_b =(1, 8, 12, 10, 13, 8, 9, 5, 8, 12) gerando assim dois caminhos que pode ser visto na figura 4.2.

A *feature* a tem um comportamento suave enquanto a *feature* b claramente está rui-

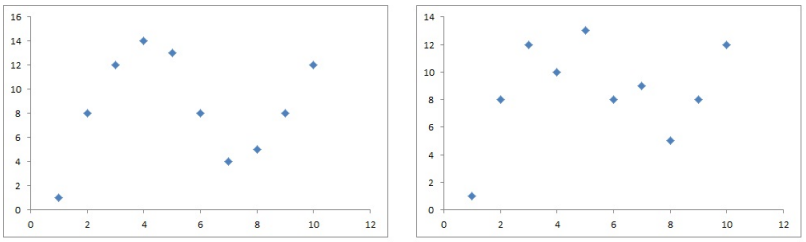

Figura 4.2 (a)Caminho da *feature a* (b)Caminho da *feature b*

dosa. Para verificar essa suavidade, utilizaremos a segunda derivada da função que gerou o caminho da *feature* e que como são desconhecidas as derivadas terão que ser aproximadas(Apêndice A). O valor das derivadas está na tabela 4.1.

Tabela 4.1 Valores para as derivadas aproximadas de 1º e 2º ordem para as *features a* e *b*

a	b	a'	b'	a''	b''
1	1	7	7	-3	-3
8	8	4	4	-2	-6
12	12	2	-2	-3	5
14	10	-1	3	-4	-8
13	13	-5	-5	1	6
8	8	-4	1	5	-5
4	9	1	-4	2	7
5	5	3	3	1	1
8	8	4	4		
12	12				

Percebe-se que para a *feature a* as segundas derivadas alternam de sinal apenas uma vez devido ao ponto de inflexão, enquanto que para a *feature b* a segunda derivada alterna 5 vezes.

Em resumo, houve 1 alternância de sinal em 8 valores de a'' contra 5 alternâncias de sinais para a mesma quantidade de valores em b''. Se faz necessário então um teste de hipótese que avalie até quantas alternâncias de sinais são validas para considerar um caminho suave.

Concluindo, a alternância dos sinais da segunda derivada será a variável aleatória a ser estudada. Foi desenvolvido então nesta dissertação um teste não-paramétrico denominado *kens* para a variável aleatória $K = \sum_i k_i$ de forma que:

$$k_i = \begin{cases} 1, & \text{se} \quad f''[i+1]f''[i] > 0 \\ -1, & \text{se} \quad f''[i+1]f''[i] < 0 \\ 0, & \quad c.c. \end{cases} \tag{4.2}$$

onde $f''[i]$ e a sequência de segundas derivadas do caminho em questão.

4.3.2 Suposições

Como todo teste de hipótese estatístico, fazem-se necessárias as suposições as quais os dados devem atender para que o teste seja devidamente preciso e assim não ocasione decisões erradas.

- garantir que a sequência dada seja oriunda de um caminho
- dados em ordem temporal

Um fator interessante a lembrar é que se uma sequência for extremamente sinuosa a quantidade de pontos de inflexão pode prejudicar o teste tornando-o mais rigoroso do que o necessário. Neste trabalho não foi abordado um estudo sobre a quantidade de pontos de inflexão, pois uma sequência de video em geral contém 20 *frames* por segundo e por isso a quantidade de pontos de inflexão pode ser considerada desprezível.

4.3.3 Estatística de Teste

Como definido na seção 4.3.1, a variável aleatória em questão será a soma da quantidade de transição de sinais nos valores aproximados das segundas derivadas($K = \sum_i k_i$). Analisando o exemplo dado na seção 4.3.1, o valor de K para a *feature* a e b é ilustrado pela tabela 4.2

Tabela 4.2 Valores de k_i e K para as *features* a e b

a	b	a''	b''	$k_i(a)$	$k_i(b)$
1	1	-3	-3	+1	+1
8	8	-2	-6	+1	-1
12	12	-3	5	+1	-1
14	10	-4	-8	-1	-1
13	13	1	6	+1	-1
8	8	5	-5	+1	-1
4	9	2	7	+1	+1
5	5	1	1		
8	8				
12	12			$\sum_i k_i(a) = 5$	$\sum_i k_i(b) = -3$

Percebe-se que a *feature* a tem um K maior que a *feature* b, ou seja, a *feature* b apresenta mais variações que a *feature* a. Em outras palavras, quanto maior mais suave.

Mas como se comporta a varável aleatória? Para responder a este questionamento é necessário agora calcular como se comporta K. Para isso vamos usar um exemplo mais simples.

Suponha que uma *feature* com 7 *frames* terá seis valores para a primeira derivada, cinco para segunda e logo terá quatro alteração de sinais. Existem dezesseis casos possíveis:

(+ + + +), (+ + + -), (+ + - +), (+ - + +), (+ + - -), (+ - + -), (+ - - +), (+ + + +)
(- + + +), (- + + -), (- + - +), (- - + +), (- + - -), (- - + -), (- - - +), (- + + +)

Nesse exemplo o caso ótimo seria $K = 4$ (+ + + +) e acontece com probabilidade 0,0625. Na verdade os cálculos de probabilidade são derivados de um binômio de Newton(Apêndice B.1). Os valores K e suas respectivas probabilidades para um caminho com 7 *frames* estão na tabela 4.3:

Tabela 4.3 Valores de K e suas probabilidades para um caminho com 7 *frames*

K	-4	-2	0	2	4
$P(K = k_i)$	$\frac{\binom{4}{0}}{2^4}$	$\frac{\binom{4}{1}}{2^4}$	$\frac{\binom{4}{2}}{2^4}$	$\frac{\binom{4}{3}}{2^4}$	$\frac{\binom{4}{4}}{2^4}$

Percebe-se que se o caminho tiver n termos teremos valores máximos de $K = n - 3$ e valores mínimos de $K = -(n - 3)$, por isso o caminho necessita ter pelo menos 4 termos.

A distribuição de probabilidade exata da variável K para n termos é dada pela tabela 4.4:

Tabela 4.4 Distribuição de probabilidade exata da variável K para n termos

K	-(n-3)	-(n-5)	...	(n-5)	(n-3)
$P(K = k_i)$	$\frac{\binom{n}{0}}{2^{n-3}}$	$\frac{\binom{n}{1}}{2^{n-3}}$	...	$\frac{\binom{n}{n-1}}{2^{n-3}}$	$\frac{\binom{n}{n}}{2^{n-3}}$

4.3.4 Hipóteses do Teste

De acordo com a variável K e visto que quanto maior o valor de K mais suave é o caminho, o teste de suavidade é um teste unilateral à direita e tem as seguintes hipóteses:

$$\begin{cases} H_0 : \text{a } \textit{feature} \text{ é } \textit{outlier} \\ H_1 : \text{a } \textit{feature} \text{ é } \textit{inlier} \end{cases} \tag{4.3}$$

$$\begin{cases} H_0 : K \leqslant k_i \\ H_1 : K > k_i \end{cases} \tag{4.4}$$

4.3.5 Distribuição de Probabilidade Aproximada

Quando n cresce a computação para $n!$ fica custosa e assim está fora do escopo desta dissertação, dessa forma é necessário verificar uma aproximação para a distribuição de probabilidade $P(K = k_i)$.

A ideia mais intuitiva é tentar primeiramente aproximar a uma distribuição normal pois o próprio binômio de Newton tem características simétricas.

O problema do teorema central do limite [28] trata da convergência em distribuição das somas parciais normalizadas,

$$\frac{S_n - E(S_n)}{\sqrt{Var(S_n)}} \tag{4.5}$$

para a distribuição normal padrão $N(0,1)$. Para tanto, supõe-se que todas as variâncias sejam finitas e que pelo menos uma delas seja estritamente positiva. O problema reside em achar condições sob as quais:

$$\frac{S_n - E(S_n)}{\sqrt{Var(S_n)}} \xrightarrow{D} N(0,1) \tag{4.6}$$

Para esse propósito precisa-se calcular E(K) e Var(K). Supondo uma *feature* com um caminho contendo n termos, logo $n-3$ transições, tem-se para E(K):

$$E(K) = \frac{-(n-3)\binom{n}{0} - (n-5)\binom{n}{1} \cdots + (n-5)\binom{n}{n-1} + (n-3)\binom{n}{n}}{2^{n-3}} \tag{4.7}$$

$$= \frac{0}{2^{n-3}} \tag{4.8}$$

$$= 0 \tag{4.9}$$

Como Var(K) = E(K^2) - (E(K))^2 e foi demonstrado acima que E(K) =0, Var(K) = E(K^2) e é dado da seguinte forma:

$$E(K^2) = \frac{(n-3)^2\binom{n}{0} + (n-5)^2\binom{n}{1} \cdots + (n-5)^2\binom{n}{n-1} + (n-3)^2\binom{n}{n}}{2^{n-3}} \tag{4.10}$$

$$= \frac{2^{n-3}(n-3)}{2^{n-3}} \tag{4.11}$$

$$= n-3 \tag{4.12}$$

Quando usamos a aproximação Normal estamos aproximando uma variável discreta (que só assume valores inteiros) por uma variável contínua (que pode assumir quaisquer valores dentro de um intervalo de número reais). É de se esperar que algum ajuste deva ser feito. Este ajuste é denominado de correção de continuidade [5].

A correção feita é diminuir 1 ao valor de K antes de calcular o valor da normal. Dessa forma, dado uma *feature* com n termos, a aproximação da distribuição de probabilidade de K pela distribuição normal é da forma:

$$\frac{K-1}{\sqrt{n-3}} \xrightarrow{D} N(0,1) \tag{4.13}$$

A demonstração e simulações sobre a aproximação estão no apêndice C.

Um ponto a se ressaltar é que pela própria característica da equação 4.13, *features* com caminhos maiores serão consideradas mais suaves que as *features* com caminhos menores, uma vez que quando n cresce o numerador cresce mais rápido que o denominador. Para exemplificar suponha uma *feature* com dez coordenadas e K igual a dez e outra com vinte coordenadas e K igual a vinte. Ambas têm todas as segundas derivadas com o mesmo sinal, indicando o máximo de suavidade porém o p-valor da primeira, a com dez coordenadas, é $3x10^{-4}$ enquanto que a *feature* com vinte coordenadas teria p-valor igual $2x10^{-6}$ e por isso seria posto primeiro na ordem de suavidade que a menor.

4.4 APLICAÇÃO DO TESTE KENS

A metodologia para aplicar o teste *kens* é dada por:

- **Passo 1**: Rastrear a cena usando o rastreador.
- **Passo 2**: Decompor a entrada dada pelo rastreador em (X,Y). Ou seja, para cada *feature* teremos o caminho que ela faz em X e o caminho que ela faz em Y, como visto na seção 2.4.
- **Passo 3**: Aplicar o *kens* em todas as *features* tanto para os caminhos em X quanto para os caminhos em Y e ordenar o p-valor em ordem crescente, ou seja, por ordem de suavidade.
- **Passo 4**: Verificar os requisitos do algoritmo de reconstrução 3D pois cada um tem os seus próprios, como por exemplo, quantidade de *features* em um determinado *frame*, tempo em que uma *feature* deve ser rastreada de um *frame* i a um *frame* j, entre outros.
- **Passo 5**: Dar como entrada as *features* mais suaves que preencham os requisitos do algoritmo de reconstrução. Uma *feature* só é considerada suave quando é suave em X e em Y.

Se o vídeo a ser reconstruído for predominantemente filmado em um único eixo, seja X ou Y, a metodologia pode ser toda feita aplicando o *kens* a apenas esse eixo e dessa forma diminuir o tempo de execução.

CAPÍTULO 5

RESULTADOS

Este capítulo apresenta os principais resultados obtidos com a utilização da metodologia proposta nesta dissertação, ilustrados com uma aplicação em dois cenários distintos: um vídeo sintético e um vídeo real. Como explicitado no capitulo 4 as *features* serão ranqueadas pelo p-valor do teste *kens* ou seja em ordem de suavidade.

A escolha de não optar por um nível de significância para o teste *kens* e sim ranquear pelos p-valores se deve ao fato de que não termos conhecimento preciso se a quantidade de *features* selecionadas como suaves seria suficiente para execução do algoritmo de reconstrução 3D.

É importante salientar que quando uma *feature* não for selecionada pelo teste *kens*, não significa necessariamente que esta *feature* é não suave, significa apenas que esta é menos suave que as demais que foram selecionadas. Da mesma forma, se uma *feature* for selecionada, não necessariamente significa que esta *feature* é suave e sim que ela é mais suave que as demais não selecionadas.

O sistema utilizado para reconstrução é o R3D [13] desenvolvido pelo GRVM [16]. Este sistema utiliza o *pipeline* de reconstrução 3D descrito na seção 2.2. Cada algoritmo de reconstrução tem suas peculiaridades e o R3D, em específico, necessita de pelo menos trinta *features* rastreadas entre o primeiro e segundo *keyframe*, em outras palavras, trinta *features* devem existir e serem rastreadas nos primeiros dois *keyframes*. Se simplesmente aplicássemos um nível de significância α ao *kens* e selecionássemos as *features inliers* não poderíamos garantir as restrições do R3D. Por este motivo é necessário fornecer como entrada ao R3D as *features* mais suaves e que atendam aos requisitos do algoritmo de reconstrução. A aplicação do teste *kens* ao R3D segue o algoritmo 2.

Algoritmo 2 Aplicação do *kens* ao sistema R3D

1. Aplicar o teste *kens* nas *features* rastreadas
2. Ordenar de forma crescente, ou seja, por suavidade(vide seção 4.4)
3. Selecionar as primeiras trinta *features*

Enquanto não existirem 30 *features* que foram rastreadas do primeiro ao segundo *keyframes* **Faça**

Acrescentar ao conjunto a próxima *feature* seguindo a ordem de suavidade

Fim Enquanto

4. Este resultado é o novo conjunto de entrada para o R3D

Em algumas reconstruções a malha 3D pode ficar visualmente comprometida, pois quando o *kens* é aplicado há uma drástica redução na quantidade de *features*. Na prática, na maioria dos casos a malha 3D é descartada a princípio, pois o mais importante é a estimativa de pose de câmera [13], uma vez que de posse de uma boa estimativa de pose

pode ser gerada, a partir de algoritmos de reconstrução densa [15], uma malha 3D ainda melhor que a gerada inicialmente. Para avaliar [20] a qualidade de uma reconstrução deve-se verificar o erro de reprojeção (seção 2.1.2) *frame* a *frame*. A melhor reconstrução então é aquela que possuir o menor erro médio *frame* a *frame*. Esta média pode ser calculada como a média do erro de reprojeção de todas as *features* em todos os *frame*s ou pode ser calculada também como a média das médias do erro de reprojeção por *frame*. Estes dois resultados são semelhantes [3].

A seguir será demonstrado a aplicação do teste *kens* em dois vídeos sendo um sintético e um real.

5.1 VÍDEO SINTÉTICO

O vídeo denominado casa (Figura 5.1) é um vídeo sintético produzido pelo GRVM [16]. Este vídeo foi feito com uma resolução 640x480 *pixels* e tem 102 *frames* ao longo de dez segundos.

Figura 5.1 Vídeo sintético - Casa

A etapa inicial da metodologia consiste em rastrear a cena. Para o vídeo da casa o rastreador utilizado foi o KLT com detecção de *features* utilizando o algoritmo GFTT (vide seção 2.1.2), fornecendo as 1024 melhores *features* (Figura 5.2a) de acordo com a classificação do GFTT. Em seguida, o próximo passo é decompor os dados de entrada fornecido pelo rastreador em (X,Y) como demonstrado na seção 4.4. É interessante notar que na Figura 5.3a os caminhos feitos pelas *features* seguem basicamente um movimento horizontal e por esse motivo poderia o teste *kens* ser aplicado apenas no eixo X, diminuindo assim o tempo de processamento. A última etapa consiste em aplicar o *kens* nas 1024 *features* fornecidas inicialmente pelo rastreador e ordenar crescentemente os p-valores obtidos pelo teste *kens*. Quanto menor o p-valor mais suave o caminho.

Como dito na introdução deste capítulo, o R3D necessita de pelo menos trinta *features* rastreadas entre o primeiro e o segundo *keyframe*. Aplicando a metodologia, a cena é reconstruída utilizando apenas as primeiras 120 *features*, o que representa uma redução de aproximadamente 88,3% do total inicialmente rastreado. Em outras palavras, a cena foi reconstruída utilizando as 120 *features* mais suaves e dentro dessas 120 *features* existem

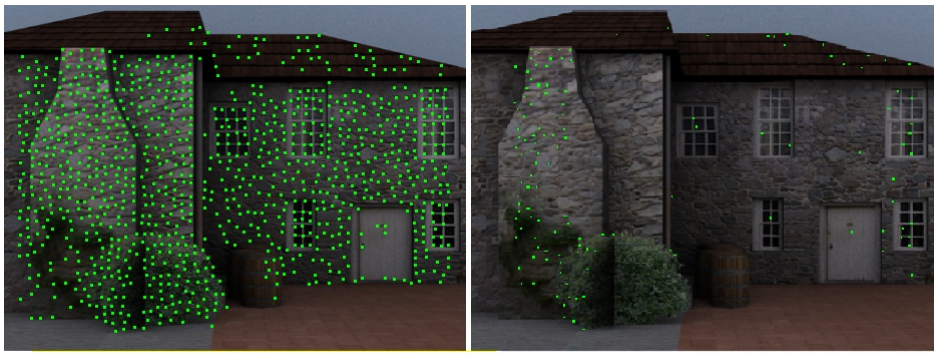

Figura 5.2 (a)*frame* 1: Vídeo sintético - Casa (b) *frame* 1(*kens*): Vídeo sintético - Casa

trinta *features* rastreadas entre o primeiro e o segundo *keyframe*. As *features* indicadas como as mais suaves pelo *kens* podem ser vistas na Figura 5.2b.

Comparando visualmente, percebe-se que ao longo do vídeo rastreado com as 1024 *features*, algumas delas apresentaram movimentos não suaves como mostrado, por exemplo, na (Figura 5.3a). Os retângulos brancos na figura destacam os caminhos mais relevantes em relação a não suavidade. Quando aplicado apenas às 120 *features* dadas pelo teste *kens*, estas *features* desapareceram como ilustra a Figura 5.3b, isso se deve ao fato de que estas *features* apresentaram comportamento menos suave, ou seja, apresentaram p-valor no teste *kens* maior que as 120 selecionadas. Outro exemplo de não suavidade também ocorre na Figura 5.4a, no canto superior direito, essas mesmas *features* não aparecem na sequência quando utilizado o *kens* (Figura 5.4b), pois também foram classificadas como menos suaves que as selecionadas pelo teste *kens*.

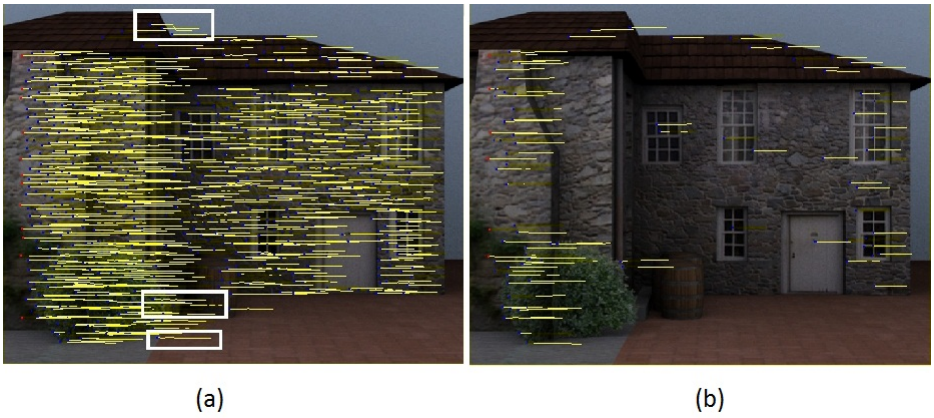

Figura 5.3 (a)*frame* 11: Vídeo sintético - Casa (b) *frame* 11(*kens*): Vídeo sintético - Casa

Como a intenção da metodologia é diminuir a quantidade de *features* para reduzir o tempo de processamento, a malha 3D pode apresentar menos detalhes que a malha com os pontos inicialmente rastreados, porém como dito no início deste capítulo, o mais

Figura 5.4 (a)*frame* 88: Vídeo sintético - Casa (b) *frame* 88(*kens*): Vídeo sintético - Casa

importante são os erros de reprojeção *frame* a *frame* e as estimativas de pose. A malha 3D e o caminho de câmera pode ser visto na Figura 5.5.

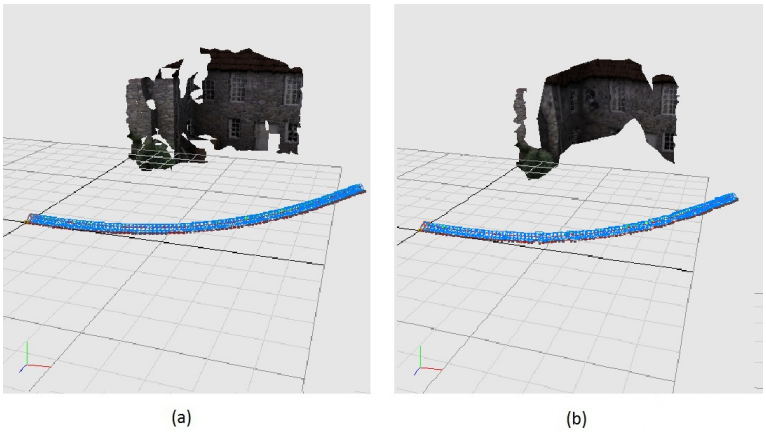

Figura 5.5 (a)Reconstrução: Vídeo sintético - Casa (b) Reconstrução(kens): Vídeo sintético - Casa

Utilizando as 1024 *features* iniciais, o vídeo da casa apresentou um erro de reprojeção *frame* a *frame* médio de 0,2327. Quando utilizado o *kens*, a cena reconstruída apresentou um erro de reprojeção médio de 0,2180, o que significa uma redução de 6,3%. Os erros de reprojeção *frame* a *frame* podem ser vistos na figura 5.6.

Como o erro de reprojeção médio entre todas as *features* e o erro médio entre as *features* utilizadas pelo *kens* foram muito próximos (apenas 6,3% de diferença), foi realizado um teste não paramétrico de Wilcoxon [24] para comparação das médias, afim de verificar se existe diferença estatística entre as mesmas. Este teste foi selecionado pois o erro de reprojeção *frame* a *frame* gerado nesse caso não atende os aspectos da distribuição normal [24]. O teste de Wilcoxon obteve um p-valor de 0,3764, o que indica que não existe diferença significativa entre as médias dos erros, ou seja, a cena reconstruída utilizando o *kens* não alterou o erro médio de reprojeção.

Vale a pena ressaltar que o vídeo da casa é sintético e apresenta pouco ruído na sequência. O ganho principal nesse cenário se deve à redução expressiva da quantidade de *features* utilizadas diminuindo assim o tempo de processamento do RANSAC e que

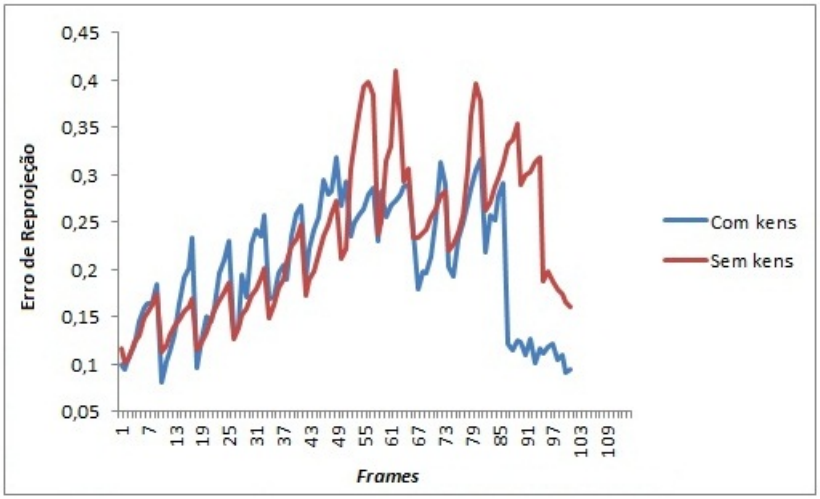

Figura 5.6 Vídeo sintético - Casa: Erro de reprojeção *frame* à *frame*

mesmo com a redução de 88% das *features* foi mantido estatisticamente o mesmo erro de reprojeção.

Outro ponto interessante é a questão da performance. O *kens* foi implementado primeiramente no software estatístico R [25] e depois na linguagem C. A aplicação do *kens* foi utilizando uma maquina com processador Intel(R) Core(TM) i3-2120 CPU 3.3GHz. Em R o tempo médio de execução do *kens* foi de aproximadamente 0,36 segundos enquanto que na linguagem C o tempo de processamento médio foi de aproximadamente 0,4 milissegundos. Ou seja, na implementação em C o incremento no tempo de execução é praticamente imperceptível.

5.2 VÍDEO REAL

A segunda demonstração utilizou um vídeo real denominado abacaxi que foi produzido com uma resolução de 960x544, composto por 164 *frames* ao longo de cinco segundos (Figura 5.7). Este vídeo foi criado artesanalmente no GRVM [16].

Assim como no caso anterior, a etapa inicial da metodologia consiste em rastrear a cena. Para o vídeo do abacaxi o rastreador utilizado foi o KLT com detecção de *features* utilizando o algoritmo GFTT (vide seção 2.1.2), fornecendo as 2000 melhores *features* (Figura 5.8a) de acordo com a classificação do GFTT. Em seguida, o próximo passo é decompor os dados de entrada fornecido pelo rastreador em (X,Y) como demonstrado na seção 4.4. É interessante notar que na Figura 5.9a os caminhos feitos pelas *features* seguem basicamente um movimento vertical (ao contrário do que foi observado quando utilizado o vídeo sintético) e por esse motivo poderia o teste *kens* ser aplicado apenas no eixo Y, diminuindo assim o tempo de processamento. A última etapa consiste em aplicar o *kens* nas 2000 *features* fornecidas inicialmente pelo rastreador e ordenar crescentemente os p-valores obtidos pelo teste *kens*. Quanto menor o p-valor mais suave o caminho.

O R3D necessita de pelo menos trinta *features* rastreadas entre o primeiro e o segundo

Figura 5.7 Vídeo real - Abacaxi

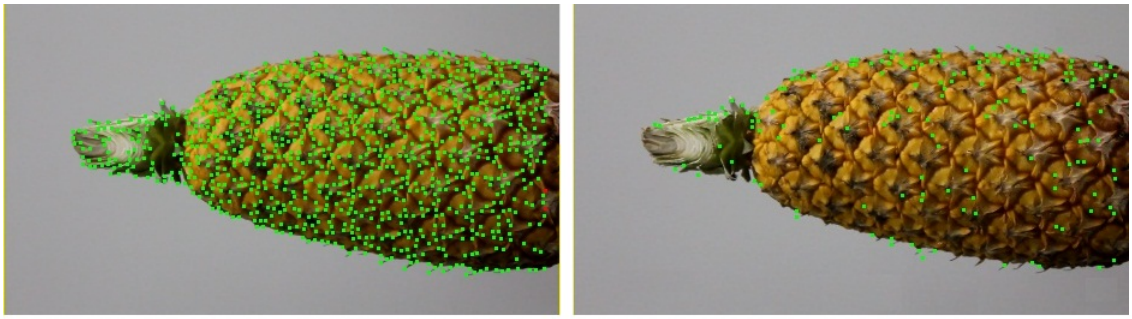

Figura 5.8 (a)*frame* 1: Vídeo real - Abacaxi (b) *frame* 1(*kens*): Vídeo real - Abacaxi

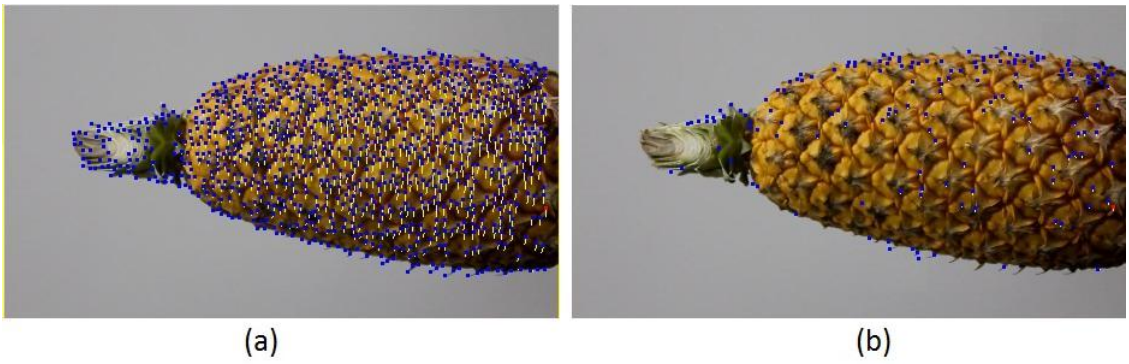

Figura 5.9 (a)*frame* 12: Vídeo real - Abacaxi (b) *frame* 12(*kens*): Vídeo real - Abacaxi

keyframe. Dada a aplicação do algoritmo 2, a cena é reconstruída utilizando apenas as primeiras 153 *features*, o que representa uma redução de aproximadamente 92,3% do total inicialmente rastreado. Em outras palavras, a cena foi reconstruída utilizando as 153 *features* mais suaves e dentro dessas 153 *features* existem trinta *features* rastreadas entre o primeiro e o segundo *keyframe*. As *features* indicadas como as mais suaves pelo

kens podem ser vistas na Figura 5.8b.

A malha 3D gerada com os pontos selecionado pelo teste *kens* sofreu discretas alterações mesmo tendo utilizado apenas 7,6% do total de *features*. A malha 3D e o caminho de câmera podem ser vistos na Figura 5.10.

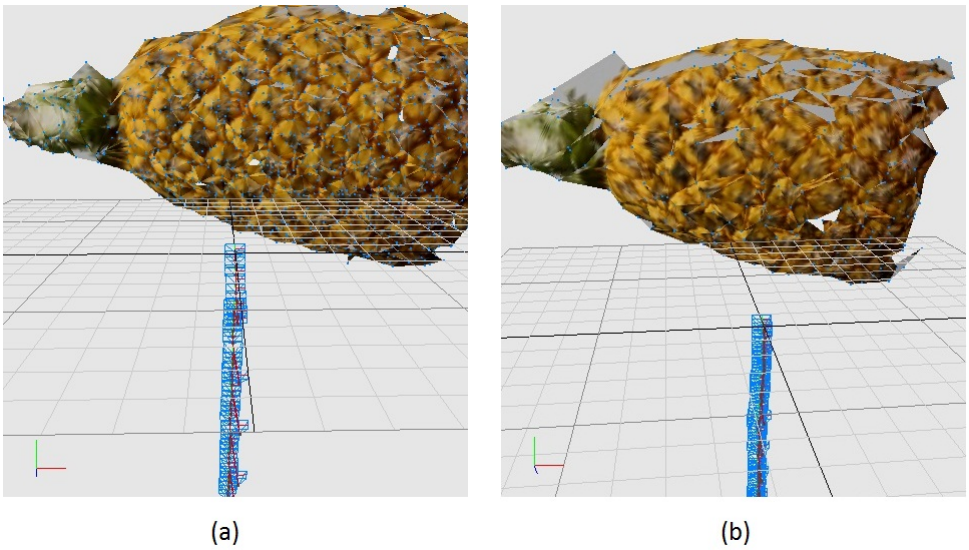

Figura 5.10 (a)Reconstrução: Vídeo real - Abacaxi (b)Reconstrução(kens): Vídeo real - Abacaxi

Utilizando as 2000 *features* iniciais, o vídeo do abacaxi apresentou um erro de reprojeção *frame* a *frame* médio de 1,689. Quando utilizado o *kens*, a cena reconstruída apresentou um erro de reprojeção médio de 0,90, o que significa uma expressiva redução de 46,9%. Os erros de reprojeção *frame* a *frame* podem ser vistos na Figura 5.11.

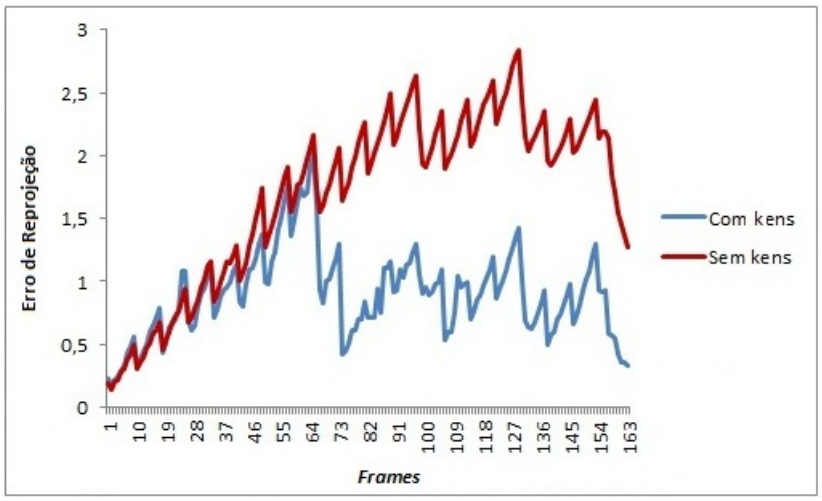

Figura 5.11 Vídeo real - Abacaxi: Erro de reprojeção *frame* a *frame*

Apesar da diferença entre os erros de reprojeção mostrados na Figura 5.11 (principal-

mente do *frame* 70 em diante), foi realizado também o teste não paramétrico de Wilcoxon [24] para confirmação estatística dessa diferença. Assim como no caso anterior, o teste não paramétrico foi utilizado pois o erro de reprojeção *frame* a *frame* gerado nesse caso não atende as condições da distribuição normal [24]. O teste de Wilcoxon obteve um p-valor menor que 10^{-5} o que indica que existe diferença significativa entre as médias dos erros, ou seja, a cena reconstruída utilizando o *kens* reduziu o erro médio de reprojeção.

Em relação à performance, a aplicação do *kens* também foi feita em uma maquina com processador Intel(R) Core(TM) i3-2120 CPU 3.3GHz. O tempo médio de execução do *kens* no R foi de aproximadamente 0,40 segundos. Já em linguagem C o tempo médio de execução foi de aproximadamente 0,5 milissegundos. Ou seja, na implementação em C o incremento no tempo de execução é praticamente imperceptível.

Para esse cenário, a utilização do *kens* reduziu a quantidade de *features* em 92,3%, não alterou praticamente o tempo de execução do *pipeline* e reduziu em 46,9% o erro de reprojeção *frame* a *frame*.

5.3 CONSIDERAÇÕES

Os resultados obtidos neste capítulo foram satisfatórios em demonstrar os resultados utilizando o *kens*. Nota-se que a principal característica da técnica proposta não é reduzir a quantidade de *features*, isto é uma consequência da técnica. O foco do *kens* é a seleção das melhores *features* e uma vez de posse delas, os cálculos do *pipeline* serão mais precisos e dessa forma serão geradas reconstruções mais fidedignas às cenas.

Os exemplos mostrados dão indícios de que *features* com caminhos suaves são de fato *features inliers*. Essa afirmação pode ser observada, pois com as *features* selecionadas pelo *kens*, os erros de reprojeção ou diminuíram ou pelo menos não foram alterados. Além do mais, fica evidenciado que não havia necessidade de utilizar a quantidade total das *features* rastreadas inicialmente, pois, houve um erro menor utilizando menos *features* e gerando a mesma reconstrução.

CAPÍTULO 6

CONCLUSÃO

Quando a reconstrução 3D é feita a partir de dados reais, existe uma introdução de erros que vão se acumulando ao longo do *pipeline*. O processamento dessas imagens virá acrescido de ruído e estes serão passados para as próximas etapas do *pipeline*. Devido a esse acúmulo de erros, as fases finais do *pipeline* podem ser comprometidas ou até mesmo não executar devido à grande quantidade de ruídos que já foi acumulada desde o início. Considerando que é praticamente impossível evitar o acúmulo de erros passados de uma fase do *pipeline* para a outra, uma proposta para minimizar ao máximo possível a quantidade de erros na primeira etapa e fornecer ao algoritmo de reconstrução as melhores *features* possíveis foi desenvolvida como objetivo principal desta dissertação. O foco para atacar este problema foi investir na fase de rastreamento.

É de senso geral, porém não provado formalmente até a presente data, que *features inliers* sigam caminhos suaves enquanto que *features outliers* não apresentem essa suavidade no caminho de câmera. Não foi encontrado na literatura um teste ou técnica com baixo custo computacional que se propusesse a quantificar a suavidade de um caminho. Por esse motivo foi desenvolvido nesta dissertação um teste de hipótese com a capacidade de avaliar a suavidade de um caminho, no caso em questão o caminhos das *features*.

O teste de hipótese especificado e implementado, denominado de *kens*, foi usado para avaliar a suavidade de caminhos. Este teste foi aplicado no contexto de fluxo ótico, aos caminhos gerados pelas *features* rastreadas na cena. Com esse teste, a hipótese de que uma *feature* com caminho suave é *inlier* não pode ainda ser considerada como prova formal, mas pode ser considerada como indício, pois não foi encontrado nos estudos realizados nesta dissertação uma *feature inlier* que apresentasse caminho sem ser suave. Por outro lado, mesmo que existisse uma *feature inlier* que não apresentasse caminho suave o *kens* a descartaria do conjunto final em detrimento das demais *features inliers*, então, sem perda de generalidade, esta técnica é válida para qualquer cenário.

Um fator importante da metodologia proposta é a não necessidade do usuário entrar com parâmetros para executar o *kens*. Dessa forma tem-se um método automático, determinístico e sem necessidade da intervenção do usuário para que a metodologia realize o experimento.

A aplicação da metodologia vista no capítulo 5 foi satisfatória, no caso sintético com baixo ruído, as *features* selecionadas pelo *kens* obtiveram o mesmo erro de reprojeção médio das *features* rastreadas originalmente, porém esse resultado foi obtido utilizando apenas 11,7% do total de *features* fornecidas pelo *tracker*. Para o caso real com maior ruído, além de reduzir o erro de reprojecão médio em 46,9%, esse resultado foi obtido utilizando apenas 7,7% das *features* fornecidas pelo *tracker*.

Uma grande dificuldade encontrada foi em desenvolver uma metodologia que obtivesse resultados em tempo real e que não comprometesse o desempenho do *pipeline* de recon-

strução 3D. A implementação do *kens*, em linguagem C, utilizando uma máquina com processador Intel(R) Core(TM) i3-2120 CPU 3.3GHz obteve tempo médio aproximado de 0,5 milissegundos para uma cena com 2000 *features*. Este tempo representa apenas 1,5% do necessário para tempo real (aproximadamente 33,3 milissegundos).

Vale salientar que tanto no vídeo sintético como no vídeo real a metodologia proposta foi validada tanto para uma sequência de vídeo considerando o eixo X (Vídeo sintético - casa) como em uma que considera apenas o eixo Y (Vídeo real - abacaxi). Ou seja, independente de como foi gerada a sequência de vídeo os resultados após a aplicação do teste *kens* são significativos.

6.1 TRABALHOS FUTUROS

Como a metodologia apresenta baixo custo computacional, um dos próximos passos a serem realizados nesta pesquisa é a implementação desta técnica para retirar os *outliers* em tempo real e não depois de um pós-processamento. Isto seria muito interessante pois poderia ser aplicado em outras áreas onde a necessidade de tempo real é fundamental como por exemplo, realidade aumentada.

Não existe ainda uma metodologia que selecione a quantidade ótima de *features* necessárias para uma reconstrução. O ordenamento por suavidade junto com as restrições dos algoritmos de reconstrução 3D tem como saída a quantidade mínima necessária para gerar uma reconstrução, mas não existe garantia que com mais *features inliers* dadas como entrada no video o erro de reprojeção *frame* a *frame* diminuiria ou não. Uma proposta de trabalho futuro que já está em andamento é a questão da quantidade ótima de *features* necessárias para a reconstrução.

APÊNDICE A

DERIVADAS DE 1º E 2º ORDEM

Em cálculo, a derivada representa a taxa de variação instantânea de uma função [52]. Seja I um intervalo com mais do que um ponto do conjunto $\mathbb{R}$ dos números reais e seja f uma função de I em $\mathbb{R}$. Se o ponto $p \in I$, diz-se que f é derivável em p se existir o limite [52] e o mesmo for finito, então $f'(p)$ é a derivada no ponto p.

$$f'(p) = \lim_{x \to p} \frac{f(x) - f(p)}{x - p} = \lim_{h \to 0} \frac{f(p+h) - f(p)}{h} \tag{A.1}$$

Note-se que a derivada de f em p , se existir, é única. Segundo esta definição, a derivada de uma função de uma variável é definida como um processo de limite. Considera-se a inclinação da secante, quando os dois pontos de intersecção com o gráfico de f convergem para um mesmo ponto. No limite, a inclinação da secante é igual à da tangente, como mostra a figura A.1

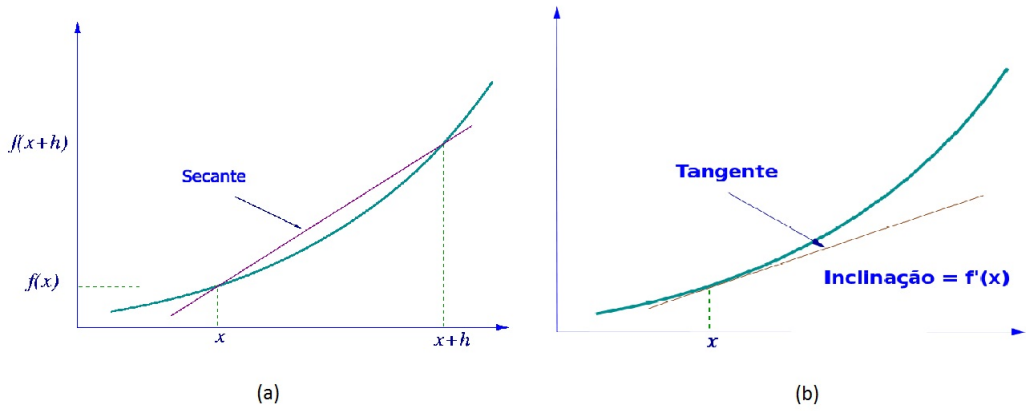

Figura A.1 (a) Inclinação da secante ao gráfico de f, (b) Inclinação da tangente à curva como a derivada de $f(x)$

A derivada de segunda ordem de uma função, ou segunda derivada, representa a derivada da derivada desta função. Matematicamente, a derivada de segunda ordem pode ser representada por $f''(p)$ e pode ser mostrada como:

$$f''(p) = \lim_{h \to 0} \frac{f(p+2h) - 2f(p+h) + f(p)}{h^2} \tag{A.2}$$

A segunda derivada será utilizada nessa proposta para verificar a suavidade do caminho pois em uma função suave $f(p)$ não deverá ter alternância nos sinais da derivada ao longo sequência (fig A.2), a não ser nos pontos de inflexão [52] onde $f''(p) = 0$.

Figura A.2 (a) Comportamento da derivada com a concavidade voltada para cima, (b) Comportamento da derivada com a concavidade voltada para cima

Como não é conhecido a função que origina as coordenadas da *feature* é necessário utilizar derivadas numérica em funções discretizadas.

A diferenciação numérica é a técnica de obter valores aproximados para a derivada de uma função usando valores da função em um conjunto de pontos e propriedades conhecidas da função. O método mais simples de se aproximar a derivada de uma função real f em um certo ponto x consiste em aproximar a declividade da reta tangente pela declividade de alguma reta secante próxima ao ponto.

Quando o resultado de uma função está discretizado em pares ordenados (x, y) e não se conhece a expressão para $y = f(x)$, então a derivação numérica mais simples é dada por:

$$f'(x) = \frac{y[i+1] - y[i]}{x[i+1] - x[i]} \tag{A.3}$$

onde $(x[1], y[1]), (x[2], y[2]), ..., (x[n], y[n])$ são os n pontos resultantes da discretização.

APÊNDICE B

TRIÂNGULO DE PASCAL

O triângulo de Pascal é um triângulo numérico infinito formado por coeficientes binomiais $\binom{n}{k}$(equação B.1) onde n representa o número da linha (posição vertical) e k representa o número da coluna (posição horizontal).

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{B.1}$$

Foi descoberto pelo matemático chines Yang Hui (1238-1298) e posteriormente várias de suas propriedades foram estudadas por Blaise Pascal (1623-1662).

Uma das formas (a mais usual) de construir um triângulo de Pascal é na forma de um triângulo isósceles (Figura B.1).

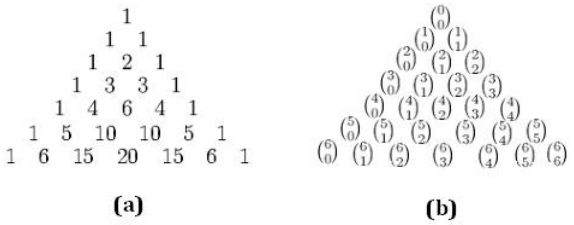

Figura B.1 (a) Triângulo de Pascal (b) Triângulo de Pascal usando Binômio de Newton

O triângulo de Pascal tem várias propriedades interessantes e frequentemente utilizadas. Entre as todas, duas são mais conhecidas: a relação de Stifel(equação B.2) em que se pode pode construir o triângulo de forma recursiva pois cada número é igual a soma dos dois números imediatamente acima dele; a outra é a relação de termos equidistantes(equação B.3) também chamados de números binomiais complementares,, propriedade esta que garante a simetria do triângulo de pascal.

$$\binom{n-1}{k-1} + \binom{n-1}{k} = \binom{n}{k} \qquad n,k \in \mathbb{N}^*; \quad n > k \tag{B.2}$$

$$\binom{n}{k} = \binom{n}{n-k} \quad \forall n,k \tag{B.3}$$

Existem uma relação menos conhecida(e que será útil para os cálculos feitos nessa dissertação) é que para uma linha n qualquer, a soma dos elementos é igual a 2^n. Esse é o teorema das linhas, que estabelece a seguinte relação:

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n \tag{B.4}$$

APÊNDICE C

DISTRIBUIÇÃO DE PROBABILIDADE DE K

As figuras C.1, C.2 e C.3 ilustram a aproximação da distribuição normal aos valores de K quando um caminho possui tamanho 10, 50 e 75 respectivamente. Também foi realizado o teste de Kolmogorov Smirnov [24]. Este teste serve para inferir se duas distribuições de probabilidade são semelhantes, o que para o nosso caso, o interesse é verificar se a distribuição de probabilidade aproximada descrita no seção C é semelhantes a distribuição de es: a dos valores exatos de K e a dos valores aproximados. Nos 3 casos

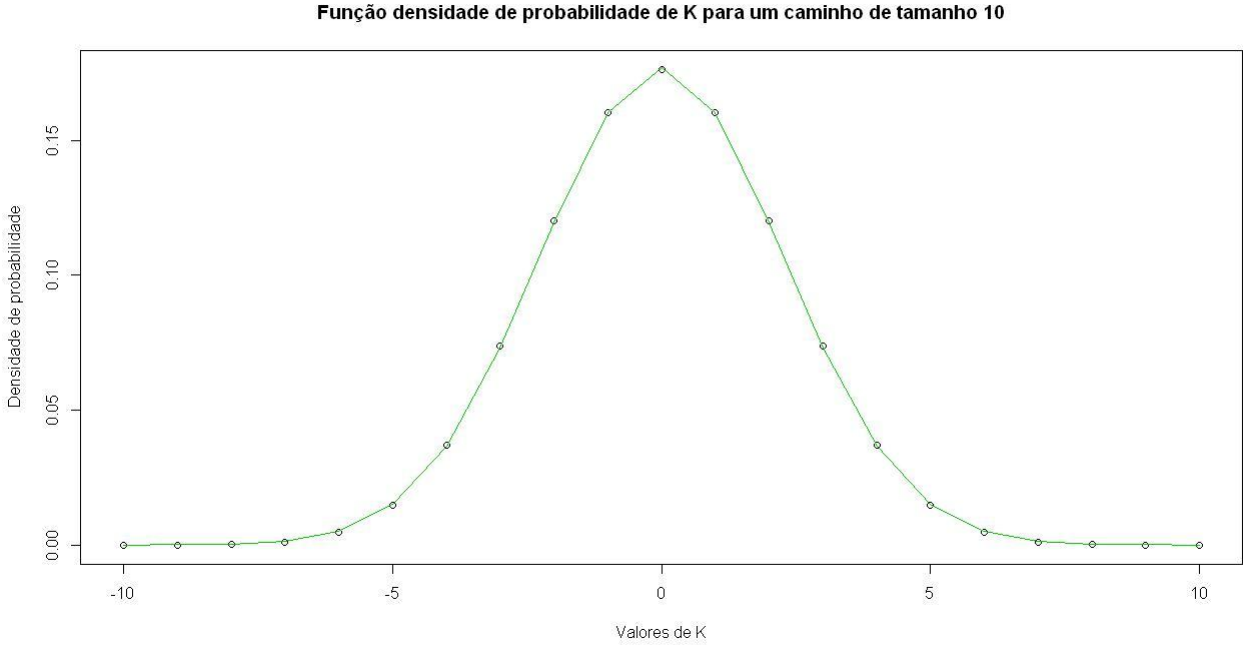

Figura C.1 Círculos pretos são os valores exatos e as linhas verdes são os valores aproximados

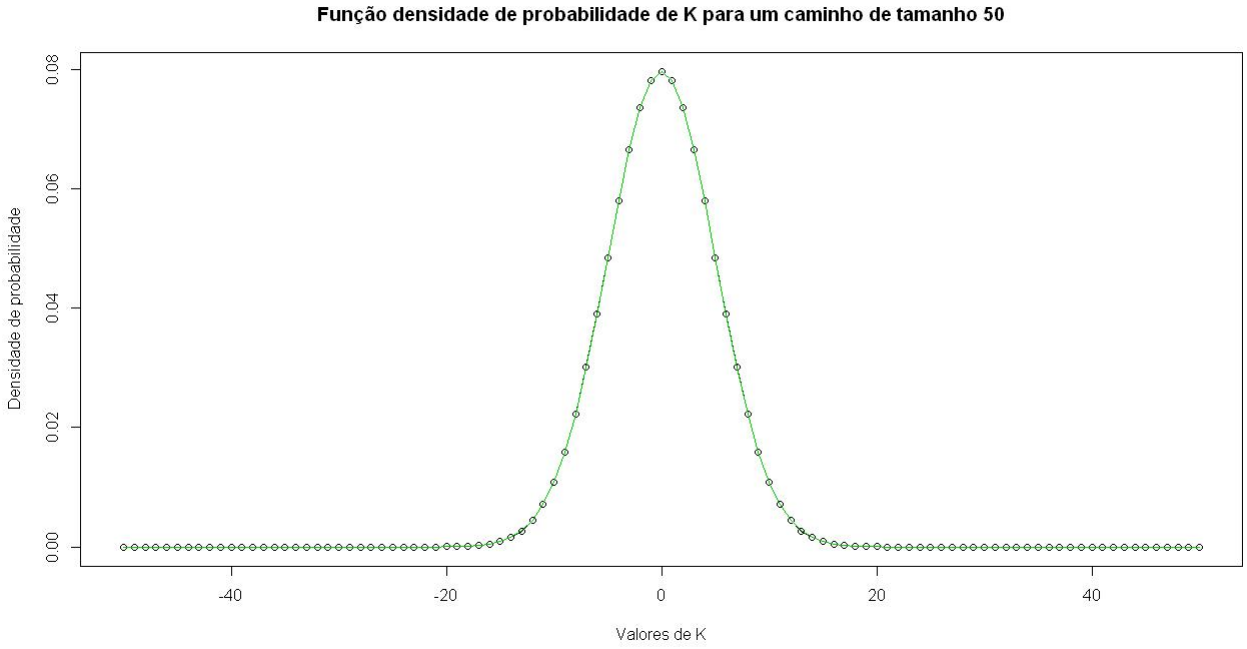

Figura C.2 Círculos pretos são os valores exatos e as linhas verdes são os valores aproximados

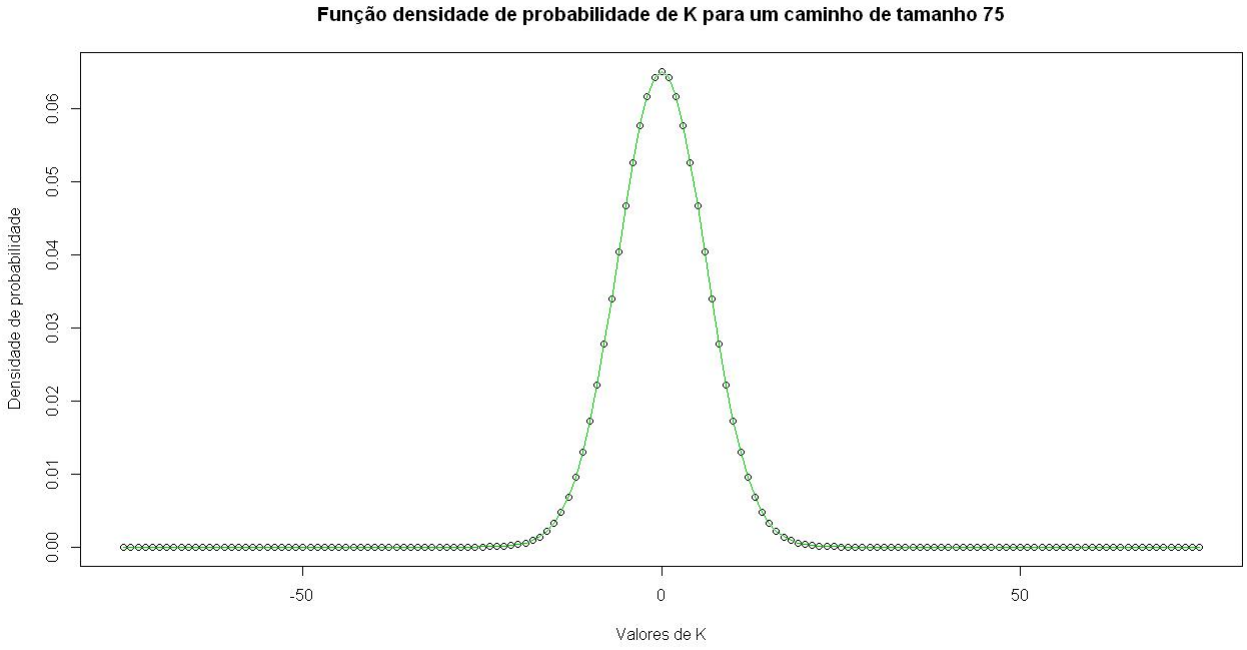

Figura C.3 Círculos pretos são os valores exatos e as linhas verdes são os valores aproximados

REFERÊNCIAS BIBLIOGRÁFICAS

[1] R. L. Barbosa. *Caminhamento Fotogramétrico Utilizando o Fluxo Óptico Filtrado.* PhD thesis, UNESP, 2006.

[2] R.L. Barbosa, B. Gallis, J.F. Silva, and M.M. Junior. A computação do fluxo óptico em imagens obtidas por um sistema móvel de mapeamento terrestre. *Revista Brasileira de Cartografia*, 2005.

[3] H. Bolfarine and W. O. Bussab. *Elementos de Amostragem.* Editora Edgard Blucher, São Paulo, 2005.

[4] H. Bolfarine and M.C. Sandoval. *Introdução a inferência estatística.* Coleção Matemática Aplicada. SBM, 2001.

[5] W. Bussab and P.A. Morettin. *Estatística Básica.* Editora Saraiva, 2002.

[6] Cordelia Schmid. C. and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:530–535, 1997.

[7] J. Choi and G.G. Medioni. Starsac: Stable random sample consensus for parameter estimation. *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 675–682, 2009.

[8] R. T. Collins. A space-sweep approach to true multi-image matching. *IEEE Computer Vision and Pattern Recognition*, 1996.

[9] V. M. César, T. Farias, Samuel Macêdo, J. Kelner, and I. Santos. Avaliação de algoritmos de estimativa de pose para reconstrução 3d e realidade aumentada. In *Workshop de Realidade Virtual e Aumentada*, WRVA 2011, 2011.

[10] F. Dellaert, S. M. Seitz, C. E. Thorpe, and S. Thrun. Structure from motion without correspondence. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2000.

[11] D. F. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–141, 1995.

[12] A. W. C. Faria. Fluxo Óptico. Technical report, Universidade Federal de Minas Gerais, 1992.

[13] T. S. M. C. Farias. *Metodologia para Reconstrução 3D Baseada em Imagens.* PhD thesis, UFPE, 2012.

[14] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Morgan Kaufmann Publishers Inc.*, 1987.

[15] Y. Furukawa and J. Ponce. Accurate, dense and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[16] GRVM. Virtual reality and multimedia research group, 1998.

[17] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. 2nd ed, 2006.

[18] C. Harris and M. Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[19] R. I. Hartley and T Saxena. In defense of the eigth-points algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997.

[20] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer*. 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[21] R.I. Hartley. Projective reconstruction from line correspondences. In *In Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 903–907. IEEE Computer Society Press, 1994.

[22] M. Heinrichs, O. Hellwich, and V. Rodehorst. Robust spatio temporal feature track. *XXI Congress of the Int. Society for Photogrammetry and Remote Sensing (ISPRS)*, 2008.

[23] K. Häming and G. Peters. The structure from motion reconstruction pipeline: a survey with focus on short image. *Kybernetika*, 2010.

[24] M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods, 2nd Edition*. Wiley-Interscience, 2 edition, jan 1999.

[25] K. Hornik. The r faq, 2012. ISBN 3-900051-08-9.

[26] W. Härdle and L. Simar. *Applied Multivariate Statistical Analysis*. Springer, 2nd edition, sep 2007.

[27] B. Jahne, H.W. Haussecker, and P. Geissler. *Handbook of Computer Vision and Applications*. Academic Press, 1999.

[28] B. R. James. *Probabilidade: Um curso em nível intermediário*. IMPA, Rio de Janeiro, 2002.

[29] K. Konolige. Sparse sparse bundle adjustment. In *British Machine Vision Conference*, Aberystwyth, Wales, 08/2010 2010.

[30] J.G. Lee, J. Han, and X. Li Li. Trajectory outlier detection: A partition and detect framework. *ICDE IEEE*, 2006.

[31] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o(n) solution to the pnp problem, 2008.

[32] M. Lhuillier and Quan L. A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2005.

[33] H. C. Longuet-Higgins. Readings in computer vision: issues, problems, principles, and paradigms. In Martin A. Fischler and Oscar Firschein, editors, *Morgan Kaufmann Publishers Inc.*, chapter A computer algorithm for reconstructing a scene from two projections. 1981.

[34] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision (ijcai). In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 674–679, April 1981.

[35] S.V.M. Macêdo, T.S.M.C. de Farias, J.C.B.O. Lima, J. Kelner, and E. Albuquerque. Towards a standalone methodology for robust algorithms evaluation: A case study in 3d reconstruction. In *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on*, pages 220 –227, aug. 2011.

[36] R. Nikolaus. Learning the parts of objects using non-negative matrix factorization (nmf), 2007.

[37] David Nistér. Preemptive ransac for live structure and motion estimation. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 199–, Washington, DC, USA, 2003. IEEE Computer Society.

[38] J.A. Nocedal and S.J. Wright. *Numerical Optimization: With 85 Illustrations.* Springer Series in Operations Research Series. Springer-Verlag GmbH, 1999.

[39] U. Orguner and F. Gustafsson. Statistical characteristics of harris corner detector. In *Proceedings of the 2007 IEEE/SP 14th Workshop on Statistical Signal Processing*, SSP '07, pages 571–575, Washington, DC, USA, 2007. IEEE Computer Society.

[40] M. Pollefeys. *Self-calibration and metric 3D reconstruction uncalibrated image sequeces.* PhD thesis, ESAT-PSI, 1999.

[41] M. Pollefeys, L. Van Gool, and M. Vergauwen. Visual modeling with a hand-held camera. *Journal Comput. Vision*, 2004.

[42] M. Prasad and A.W. Fitzgibbon. Single view reconstruction of curved surfaces. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2:1345–1354, 2006.

[43] H. Prautzsch, W. Boehm, and M. Paluszny. *Bezier and B-Spline Techniques.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[44] L. Quan and Z. Lan. Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1999.

[45] V. Rodehorst and O. Hellwich. Genetic algorithm sample consensus (gasac) - a parallel strategy for robust parameter estimation. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, page 103, june 2006.

[46] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection.* John Wiley, 1987.

[47] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Depth perception from a single still image. *Periódico Desconhecido*, 2008.

[48] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or 'how do i organize my holiday snaps?'. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 1, pages 414–431. Springer-Verlag, 2002.

[49] J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593–600, 1994.

[50] Y. Q Shi and H. Sun. *Image and video compression for multimedia engineering : fundamentals, algorithms, and standards.* Boca Raton, Fla. ; London : CRC Press, 2000.

[51] D. Silva. *Uma avaliação de algoritmos de rastreamento 2D para uso em reconstrução 3D.* PhD thesis, UFPE, 2010.

[52] J. Stewart. *Curso de cálculo volume 1.* Coleção Matemática Aplicada. São Paulo: Pioneira Thomson Learning, 2002.

[53] R. Szeliski. *Computer Vision: Algorithms and Applications.* Springer, 1st edition. edition, nov 2010.

[54] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, pages 1247–1283, 2000.

[55] C. Tomasi and T. Kanade. Shape and motion without depth. In *ICCV*, pages 91–95. IEEE, 1990.

[56] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 1992.

[57] P. H. S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:2000, 2000.

[58] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment: a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*, pages 298–375. Springer Verlag, 2000.

[59] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 1987.

[60] M. Vergauwen and L. Van Gool. Web-based 3d reconstruction service. *Machine Vision and Applications*, 2006.

[61] W. W. S. Wei. *Time Series Analysis: Univariate and Multivariate Methods*. Pearson, 2006.

[62] T Xiang and L.F. Cheong. Understanding the behavior of sfm algorithms: A geometric approach. *International Journal of Computer Vision*, 2003.

[63] M. Yang, J. Tao, L. Shi, K. Mu, and J. Che. An outlier rejection scheme for optical flow tracking. In *IEEE Workshop on Machine Learning for Signal Processing*, pages 1–4, 2011.

[64] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), dec 2006.

[65] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1998.

[66] Z. Zhang, R. Deriche, O. Faugeras, and Q. T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry, October 1995.