***Estimação de movimento
a partir de imagens RGBD
usando homomorfismo entre grafos***

David da Silva Pires

TESE APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
DOUTOR EM CIÊNCIAS

Programa: **Computação**
Orientador: **Prof. Dr. Roberto M. Cesar-Jr**
Coorientador: **Prof. Dr. Luiz Velho**

Durante a elaboração deste trabalho, o autor recebeu auxílio financeiro da CAPES

- São Paulo, 13 de maio de 2013 -

***Estimação de movimento
a partir de imagens RGBD
usando homomorfismo entre grafos***

Esta versão da tese contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 14/12/2012. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Roberto Marcondes Cesar-Jr. (orientador) - IME-USP
- Prof. Dr. Carlos Hitoshi Morimoto - IME-USP
- Prof. Dr. Roberto Hirata Júnior - IME-USP
- Prof. Dr. Marcio Lobo Netto - EP-USP
- Prof^a. Dr^a. Soraia Raupp Musse - PUC-RS

*A meu amigo Jesús P. Mena-Chalco,
certamente culpado por mudar o curso de meu doutorado,
eventualmente culpado por mudar o curso de minha vida.*

*A minha namorada Bartira,
eventualmente culpada por mudar o curso de meu doutorado,
certamente culpada por mudar o curso de minha vida,
sol que ilumina meus dias.*

Resumo

PIRES, D. S. **Estimação de movimento a partir de imagens RGBD usando homomorfismo entre grafos**. 105 p. Tese - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2012.

Recentemente surgiram dispositivos sensores de profundidade capazes de capturar textura e geometria de uma cena em tempo real. Com isso, diversas técnicas de Visão Computacional, que antes eram aplicadas apenas a texturas, agora são passíveis de uma reformulação, visando o uso também da geometria. Ao mesmo tempo em que tais algoritmos, tirando vantagem dessa nova tecnologia, podem ser acelerados ou tornarem-se mais robustos, surgem igualmente diversos novos desafios e problemas interessantes a serem enfrentados. Como exemplo desses dispositivos podemos citar o do Projeto Vídeo 4D, do IMPA, e o Kinect™, da Microsoft. Esses equipamentos fornecem imagens que vêm sendo chamadas de RGBD, fazendo referência aos três canais de cores e ao canal adicional de profundidade (com a letra ‘D’ vindo do termo *depth*, profundidade em inglês). A pesquisa descrita nesta tese apresenta uma nova abordagem não-supervisionada para a estimação de movimento a partir de vídeos compostos por imagens RGBD. Esse é um passo intermediário necessário para a identificação de componentes rígidos de um objeto articulado. Nosso método faz uso da técnica de casamento inexato (homomorfismo) entre grafos para encontrar grupos de pixels (blocos) que se movem para um mesmo sentido em quadros consecutivos de um vídeo. Com o intuito de escolher o melhor casamento para cada bloco, é minimizada uma função custo que leva em conta distâncias tanto no espaço de cores RGB quanto no XYZ (espaço tridimensional do mundo). A contribuição metodológica consiste justamente na manipulação dos dados de profundidade fornecidos pelos novos dispositivos de captura, de modo que tais dados passem a integrar o vetor de características que representa cada bloco nos grafos a serem casados. Nosso método não usa quadros de referência para inicialização e é aplicável a qualquer vídeo que contenha movimento paramétrico por partes. Para blocos cujas dimensões causem uma relativa diminuição na resolução das imagens, nossa aplicação roda em tempo real. Para validar a metodologia proposta, são apresentados resultados envolvendo diversas classes de objetos com diferentes tipos de movimento, tais como vídeos de pessoas caminhando, os movimento de um braço e um casal de dançarinos de samba de gafieira. Também são apresentados os avanços obtidos na modelagem de um sistema de vídeo 4D orientado a objetos, o qual norteia o desenvolvimento de diversas aplicações a serem desenvolvidas na continuação deste trabalho.

Palavras-chave: estimação de movimento, segmentação de movimento, casamento entre grafos, imagens RGBD.

Abstract

PIRES, D. S. **Motion estimation from RGBD images using graph homomorphism.** 105 p. Ph. D. Thesis
- Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2012.

Depth-sensing devices have arisen recently, allowing real-time scene texture and depth capture. As a result, many computer vision techniques, primarily applied only to textures, now can be reformulated using additional properties like the geometry. At the same time that these algorithms, making use of this new technology, can be accelerated or be made more robust, new interesting challenges and problems to be confronted are appearing. Examples of such devices include the 4D Video Project, from IMPA, and Kinect™, from Microsoft. These devices offer the so called RGBD images, being related to the three color channels and to the additional depth channel. The research described on this thesis presents a new non-supervised approach to estimate motion from videos composed by RGBD images. This is an intermediary and necessary step to identify the rigid components of an articulated object. Our method uses the technique of inexact graph matching (homomorphism) to find groups of pixels (patches) that move to the same direction in subsequent video frames. In order to choose the best matching for each patch, we minimize a cost function that accounts for distances on RGB color and XYZ (tridimensional world coordinates) spaces. The methodological contribution consists on depth data manipulation given by the new capture devices, such that these data become components of the feature vector that represents each patch on graphs to be matched. Our method does not use reference frames in order to be initialized and it can be applied to any video that contains piecewise parametric motion. For patches which allow a relative decrease on images resolution, our application runs in real-time. In order to validate the proposed methodology, we present results involving object classes with different movement kinds, such as videos with walking people, the motions of an arm and a couple of samba dancers. We also present the advances obtained on modeling an object oriented 4D video system, which guide a development of different applications to be developed as future work.

Keywords: motion estimation, motion segmentation, graph matching, RGBD images.

Sumário

Resumo

Abstract

Lista de Figuras

Lista de Tabelas

1 **Introdução**

Motivação , 1 • Comentários preliminares, 2 • Objetivos , 4 • Contribuições, 4 • Organização do texto , 5.

7 **Revisão bibliográfica**

Estimação de movimento , 7 • Mapa de profundidade , 9 • Escaneamento 3D , 10 • Vídeo 3D e vídeo 4D , 13.

17 **Metodologia**

Perspectiva geral, 17 • Sequência de etapas, 18 • Dados de entrada, 18 • Processamento da textura, 21 • Processamento do mapa de profundidade, 22 • Abordagem baseada em grafos, 25 • Visualização dos resultados, 28 • Desempenho, 30.

33 **Resultados experimentais**

41 **Conclusão**

Discussão, 41 • Trabalhos futuros, 41.

45 **Detalhes técnicos da implementação**

Porte do vídeo 3D para o sistema operacional GNU/Linux, 45 • Modelagem do sistema de vídeo 4D , 45 • Equipamento e programa da estimação de movimento, 57.

59

Manual de captura de vídeo 3D

Início rápido , 59 • Detalhamento da captura , 60.

69

Pôster SIBGRAPI 2012

71

Artigo 2013

Referências Bibliográficas

Lista de Figuras

1.1	Dois dispositivos sensores de profundidade em uso atualmente	3
1.2	Dados de entrada típicos	3
1.3	Resultado típico	3
2.1	Sequência de passos do sistema do IMPA	14
2.2	Etapas que constituíram o trabalho	15
3.1	Fluxo de dados através do método implementado	19
3.2	Imagem RGBD	21
3.3	Volume de visualização ou <i>frustum</i>	22
3.4	Controlando os limiares do volume de visualização	23
3.5	Exemplo de mapeamento de textura	24
3.6	Construção e casamento de grafos	26
3.7	Exemplo de casamento entre grafos	27
3.8	Opções de exibição dos blocos	29
3.9	Rótulos que indicam a direção do movimento	30
3.10	Janela principal do aplicativo	31
3.11	Caixa de diálogo do desenho do grafo e do casamento	32
4.1	Dados de entrada e saída típicos	33
4.2	Imagem de profundidade capturada	34
4.3	Imagem de textura	34
4.4	Sequência de vídeo com pessoa movendo seu braço	34
4.5	Visualização dos casamentos como setas	35
4.6	Resultado refinado de segmentação de movimento usando blocos de 2×2 pixels	35
4.7	Teste dos valores extremos de α : 0 e 1	36
4.8	Textura e mapa de profundidade	37
4.9	Comparação entre casamentos	39
4.10	Teste de robustez da classificação	40
5.1	Desempenho do Kinect™ na presença de luz solar	42
5.2	Falha de captura na imagem de textura	42
A.1	Casos de uso em alto nível	46
A.2	Edição de um vídeo 4D	48
A.3	Salvamento e abertura de um vídeo 4D	49
A.4	Exportação de um vídeo 4D	50
A.5	Importação de textura e geometria	52
A.6	Diagrama de implantação	53
A.7	Sequência de passos do sistema do IMPA	54

A.8	Diagrama de colaboração	55
A.9	Diagrama de estados I	55
A.10	Diagrama de estados II	56
A.11	Diagrama de atividades	56
B.1	Câmera e projetor em um tripé	61
B.2	Câmera usada no sistema de captura de vídeo 3D	62
B.3	Enquadramento da cena na câmera	62
B.4	Botão para fazer ajuste manual da íris da câmera	63
B.5	Projetor usado no sistema de captura de vídeo 3D	63
B.6	<i>Genlock</i>	64
B.7	Monitores usados no sistema de captura de vídeo 3D	65
B.8	O programa v4dFaixas	65
B.9	Ajuste dos objetos na cena a ser gravada	66
B.10	Utensílios	67

Lista de Tabelas

B.1	Atalhos do programa v4dFaixas: quantidade de faixas exibidas	66
B.2	Atalhos do programa v4dFaixas: tipo de slides exibidos	66

Agradecimentos

DOZE ANOS. Exatamente uma dúzia de anos é o tempo que se passou desde que minha amiga Ana Beatriz me apresentou, como um aluno interessado em fazer iniciação científica, ao professor que a tantos fascinava com seu jeito descontraído de dar as aulas de Visão Computacional e Processamento de Imagens do IME-USP: o professor Roberto Cesar. Da iniciação científica veio o mestrado, do mestrado o doutorado e, não é de surpreender que, em meio a tanto tempo, a amizade. Sempre pronto a perdoar minhas falhas em questões que sempre tive dificuldades e disposto a ajudar até em questões pessoais, vejo em você, Roberto, uma pessoa muito humana, à qual sou muito grato por todo o aprendizado. Sei que continuarei sempre sendo seu aluno.

Creio que também sou tão grato quanto a meu amigo Jesús P. Mena-Chalco, ao qual dedico esta obra. Assim como pude acompanhar a rápida evolução acadêmica do Roberto, também vi você, Jesús, galgando passo-a-passo uma escalada de sucesso. Seus comentários sempre pertinentes ao meu trabalho, as animadas discussões que tivemos sobre computação e, claro, sua interferência direta em meu doutorado, quando me emprestou, sem data para devolução, o Kinect™ que tinha acabado de comprar e com o qual só havia brincado por um dia, desenharam em minha memória lembranças agradáveis que quero sempre manter. Agora, que cada um vai seguindo por seu caminho, é hora de vivermos coisas diferentes, fortalecendo essa amizade.

Agradeço ao professor Luiz Velho, por sempre ter me recebido tão bem quando fui ao IMPA e pelas ótimas orientações que recebi nos poucos contatos (*mea maxima culpa*) que tivemos.

Também agradeço a todos os colegas do Laboratório de Processamento de Imagens, Bioinfo I, com os quais convivi tantos anos estudando, trabalhando e me divertindo.

Nesta reta final do doutorado, pude contar com a ajuda de muitas pessoas que doaram seu tempo e suas habilidades para que eu pudesse concluir a minha tese. Agradeço especialmente ao Pedro Losco Takecian e ao Jesús P. Mena-Chalco (de novo!) pelas ótimas revisões que fizeram em meu texto final, aos meus professores de dança Camilla Varjão e Ricardo Oliveira por embelezarem o meu trabalho com sua arte do samba de gafieira, e à minha sobrinha Manuela (e a seus pais, minha irmã Deise Pires Hala e meu cunhado George Roberto Hala) por ter sido a estrela em uma captura com o Kinect ao mostrar que já domina a arte de andar.

Certamente essa tese não teria sido possível sem todo o amor que recebi de meus pais, Antonio Pires Eustáquio e Dulcineia da Silva. Acho que só entenderei a razão de todos os sacrifícios que vocês fizeram, e fazem, por mim quando tiver meus próprios filhos.

E claro, eu não poderia deixar de agradecer à minha namorada Bartira Ferraz Martiniano, a quem também dedico esta tese e que, apesar de nem imaginar o quê o operador ++ faz em C, sempre me ouviu com muita atenção quando eu falava sobre as minhas dificuldades no doutorado, demonstrando toda sua paciência. Não houve um só dia que não tenhamos tratado desse assunto nessa reta final; você se fez presente como ninguém, em meu trabalho e em meus pensamentos.

David da Silva Pires
São Paulo, 17 de agosto de 2012.

Capítulo 1

Introdução

Esta tese apresenta um método para a estimação densa de movimento com base no formalismo de algoritmos em grafos, mais especificamente homomorfismo entre grafos. Tais algoritmos têm sido aplicados a vários problemas de processamento e análise de imagens, incluindo filtragem e segmentação, classificação, representação e descrição de objetos.

A matemática discreta fornece uma estrutura elegante para o processamento de imagens, sendo rica em algoritmos eficientes, com provas de corretude. Como consequência, muitos métodos recentes de processamento de imagem têm sido modelados como problemas de busca e otimização em grafos. Métodos tradicionais podem também ser reformulados com base em grafos, levando a implementações mais eficientes, e/ou favorecendo análises teóricas. Por outro lado, as peculiaridades do processamento de imagem exigem adaptações específicas, o que gera novos desafios de pesquisa e oportunidades, visto a crescente quantidade de publicações na área [Lézoray & Grady, 2012].

1.1 Motivação

HÁ DIVERSOS PROBLEMAS a serem resolvidos e metodologias que vem sendo aplicadas com relação à estimação de movimento 3D em vídeo. Em alguns casos, algumas soluções tem sido adotadas e implementadas e já há aplicações sendo usadas em situações reais, beneficiando-se da evolução que a área sofreu nos últimos anos. Muitas dessas aplicações lidam apenas com o movimento que é percebido a partir de um vídeo comum, com informação apenas da textura da cena. Outras usam marcadores nos objetos para inferir uma representação tridimensional dos mesmos.

Dado que o escaneamento 3D tradicional, envolvendo objetos rígidos e estáticos, já foi resolvido para variadas situações práticas, muito se tem dado atenção à presença de movimento nas cenas capturadas em um escaneamento 3D, também envolvendo objetos deformáveis (maleáveis).

Ao mesmo tempo em que essa necessidade é verificada, tornou-se disponível no mercado câmeras de vídeo de profundidade, permitindo que, juntamente com a informação de intensidade, seja capturada também a distância de cada pixel até o dispositivo sensor. Essa disponibilidade permitiu que fossem desenvolvidos produtos que realizam o reconhecimento de gestos para interação natural, usando estes como um meio de controlar dispositivos eletrônicos, tais como televisores e *vídeo-games*.

Dentre as principais aplicações que motivam este trabalho, podemos citar:

- análise de cenas de tráfego de automóveis;
- análise 3D do modo de caminhar humano;
- estimação do esqueleto de uma pessoa;
- rastreamento de pessoas;

- rastreamento de animais;
- reconhecimento de gestos para interação natural [Suma *et al.*, 2011];
- escaneamento 3D de cenas dinâmicas;
- escaneamento completo 3D de objetos articulados;
- mapeamento de texturas em superfícies;
- identificação de componentes rígidos em objetos articulados [Kumar *et al.*, 2005];
- remoção de ruído e outros artefatos de vídeo;
- estabilização de imagem durante gravação de vídeo com câmera digital [Srinivasan *et al.*, 2005];
- desentrelaçamento de vídeo;
- interpolação de quadros em vídeo (câmera lenta);
- compressão de vídeo [Gall, 1991; Lee *et al.*, 1997];
- *morphing* automatizado;
- realidade aumentada;
- vigilância automatizada;
- automação na condução de automóveis;
- edição de vídeo.

1.2 Comentários preliminares

O MÉTODO DESENVOLVIDO durante o doutorado, apresentado neste documento, faz parte do desenvolvimento de uma plataforma de edição e integração de dados, interação com usuário e visualização de vídeo 3D. Trata-se do conceito que chamamos de vídeo 4D, o qual é explicado nos capítulos seguintes. Este trabalho é uma continuação de uma parceria com o professor Luiz Velho, do Instituto Nacional de Matemática Pura e Aplicada (IMPA), dando sequência a uma colaboração que se iniciou durante o mestrado.

Nesse contexto, fez-se uso do equipamento de captura de vídeo 3D desenvolvido no IMPA, o qual usa uma técnica baseada em luz estruturada para inferir a distância dos objetos até a câmera. Com o acesso que obtivemos a esse hardware e com o posterior surgimento de dispositivos como o Kinect™ (da Microsoft®) e o Xtion™ (da ASUS®), que podem capturar imagens de textura e mapa de profundidade de uma cena, pudemos iniciar os experimentos tomando estes dados como entrada e enfrentar os muitos desafios e problemas interessantes que surgiram. A Figura 1.1 mostra os dois sensores de profundidade citados (as imagens não estão em escala).

O estudo de movimento em Visão Computacional é um passo adiante em relação a problemas que envolvem uma única imagem ou duas imagens adquiridas ao mesmo tempo: para que o movimento possa ser percebido, faz-se necessário o processamento de imagens adquiridas *no decorrer do tempo*. O interesse passa a ser a informação visual que pode ser extraída a partir das mudanças espaciais e temporais que ocorrem em uma sequência de imagens.

Assumindo que as condições de iluminação não variam, mudanças nas imagens podem ser causadas por um movimento relativo entre a câmera e a cena: a câmera pode se mover em frente a uma cena estática, partes da cena podem se mover em frente a uma câmera estacionária ou, em um caso mais geral, tanto câmera quanto objetos podem se mover com diferentes movimentos.

Dada uma sequência de vídeo, tal como a exibida na Figura 1.2, nosso método constrói um grafo para

Figura 1.1: Dois dispositivos sensores de profundidade em uso atualmente.

Figura 1.2: Os dados de entrada típicos constituem uma sequência de vídeo com uma imagem de textura e outra de profundidade para cada quadro. A imagem de textura fornece valores RGB enquanto que a do mapa de profundidade nos dá triplas (x, y, z) .

cada quadro e os compara, entre quadros consecutivos, encontrando um casamento baseado nas distâncias nos espaços RGB de cores e XYZ de coordenadas do mundo. Tais casamentos são codificados como rótulos coloridos com o propósito de visualização.

A principal aplicação de dados de entrada como os apresentados é relativa a interação natural, seja para controlar um televisor ou no ramo do entretenimento, para interagir com jogos de *vídeo-games*. Além desse tipo de aplicação, também podemos usar a informação adicional fornecida pela imagem de profundidade para melhorar algoritmos que, até agora, eram aplicados apenas a imagens de textura. É justamente nesse sentido que desenvolvemos o método de estimação de movimento aqui apresentado, conforme exibido na Figura 1.3.

Nossa abordagem pode ser usada em situações mais gerais que as que contêm humanos fazendo gestos em frente ao dispositivo de captura. O método apresentado pode lidar com oclusão e com continuidade es-

Figura 1.3: Detecção e segmentação de movimento (à direita) usando nosso método aplicado às imagens de textura (à esquerda) e de profundidade (meio). As cores vermelho e azul indicam movimento para a direita e para a esquerda, respectivamente.

pacial e é independente de quadros que sejam usados para inicialização. O movimento deve ser paramétrico por partes, a fim de que seja possível identificar componentes rígidos que estejam se movendo.

1.2.1 Divulgação científica

O software encontra-se na página <http://www.vision.ime.usp.br/~davidsp/4dvideoviewer/>. O programa pode ser baixado e instalado em plataformas GNU/Linux e permite capturar, ler arquivos (vide base de dados abaixo), gravar arquivos, calcular o casamento e fazer a visualização dos resultados.

Nós criamos uma base de dados para trabalhar e tornamos tal base pública, acessível a partir do endereço: <http://www.vision.ime.usp.br/~davidsp/kinectdatabase/>.

Todo material que documenta o desenvolvimento do software e o modo de usá-lo encontra-se disponível em <http://www.vision.ime.usp.br/~davidsp/4dvideoviewer/v4dviewer/doc>, incluindo documentação gerada a partir do código-fonte (com o Doxygen), diagramas UML sobre o sistema, manual de usuário e a versão final desta tese. Também são distribuídos outros tipos de documentos, tais como pôsteres e conjuntos de slides de apresentação, os quais foram usados em conferências e seminários dados sobre o trabalho desenvolvido, e o arquivo bibTeX usado nesta tese, o qual contém as principais referências para quem quiser se aprofundar nos detalhes, teóricos e técnicos, aqui descritos.

1.3 Objetivos

ESTE TRABALHO visa desenvolver um método, baseado em modelos estruturais 3D dinâmicos, representados por casamentos entre grafos, que explore os benefícios ao se usar a informação adicional dada pela imagem de profundidade registrada com a imagem de textura, apresentando um algoritmo que detecta a direção do movimento (fluxo óptico) a taxas de tempo real. O procedimento desenvolvido mostra, por meio de rótulos identificados por cores, para qual direção cada grupo de pixels (denominado *bloco*) está se movendo. Dados de profundidade, quando usados em conjunto com os tradicionais valores RGB e coordenadas de textura, facilitam a delimitação de objetos de interesse e tornam os resultados substancialmente melhores quando considerados como uma característica descritiva para cada pixel. Esse fator é uma vantagem sobre métodos que dependem da presença de um padrão de textura bem definido, como uma sequência de tabuleiros quadriculados [Tron & Vidal, 2007].

1.4 Contribuições

HÁ TEMPOS VEM SENDO DESENVOLVIDA uma metodologia no Grupo Creativision (Grupo de Visão Criativa do IME-USP, coordenado pelo professor Roberto Cesar), que vem pesquisando o uso da técnica de casamento inexato entre grafos (homomorfismo). Cada vez mais essa técnica tem se mostrado robusta e genérica, como pode ser verificado nas teses de doutorado de Noma [2010] e Graciano [2012]. Dentre as aplicações, encontram-se segmentação interativa de imagem natural, colorização assistida por computador e rastreamento de objetos.

A presente tese é mais uma contribuição para essa linha de pesquisa. A metodologia desenvolvida aqui generaliza os casos tratados em trabalhos anteriores (que baseavam-se apenas na textura) para casos 3D, que também fazem uso da informação de profundidade de uma cena. A inovação que é trazida com o aumento de uma dimensão é explorada de diversas formas no texto aqui apresentado. Seu emprego é feito na tentativa de resolução de um problema muito desafiador: a estimação de movimento em cenas sem restrições quanto à iluminação ou quanto ao tipo de objetos se movimentando (que pode ser rígido ou maleável, articulado ou não).

O trabalho desenvolvido também contribuiu para a aquisição de *know-how* no grupo com relação a uma nova proposta de uso de modelos de grafos com integração (fusão) de atributos oriundos de diferentes fontes (textura, geometria e vídeo). Foi a primeira vez que tivemos acesso a dispositivos sensores de profundidade de última geração, o que serviu de grande motivação para criar um método que permitisse a comparação entre os diferentes tipos de entrada.

Não obstante, outro fruto resultante deste trabalho foi a implementação de um *software* que permite a execução, a experimentação e a análise dos resultados de toda a pesquisa realizada.

1.5 Organização do texto

O CAPÍTULO 2 contém uma revisão bibliográfica sobre estimação de movimento, mapas de profundidade e uma seção que explicita o estado da arte na área de Computação Gráfica em relação aos algoritmos já implementados para a reconstrução de objetos articulados, bem como dos problemas inerentes a essa tarefa. Ainda no Capítulo 2, a Seção 2.4 fornece uma breve visão sobre o sistema de captura de vídeo 3D desenvolvido no IMPA e tenta definir os principais tópicos que caracterizarão o vídeo 4D. Nesse sentido, é apresentada a modelagem já existente para este. Já no Capítulo 3, é descrita a metodologia que foi desenvolvida para a estimação de movimento a partir de imagens RGBD. A fim de validar a metodologia proposta, no Capítulo 4 são apresentados os resultados que obtivemos com a gravação de objetos móveis, resultados estes que são discutidos no Capítulo 5. O Apêndice A apresenta a modelagem desenvolvida para o vídeo 4D e detalhes técnicos da implementação no uso do Kinect™. O Apêndice B, por sua vez, é um manual de captura de vídeo 3D. Por fim, no Apêndice C encontra-se o pôster referente ao método desenvolvido nesta tese que foi apresentado no Sibgrapi 2012 (*Congress on Graphics, Patterns and Images*), seguido, no Apêndice D, por um artigo que pretendemos submeter oportunamente.

Capítulo 2

Revisão bibliográfica

ESTE LEVANTAMENTO BIBLIOGRÁFICO sobre trabalhos relacionados ao tema da tese inicia com uma breve descrição sobre pesquisas realizadas acerca da estimação de movimento na Seção 2.1. Procurou-se, de forma sucinta, dar uma ideia geral sobre fluxo óptico, listar aplicações que façam uso dele e, na medida do possível, relacionar tais trabalhos com o método desenvolvido.

Na Seção 2.2, são discutidas algumas aplicações do mapa de profundidade, o formato de dados fornecido por dispositivos como o Kinect™. Já na Seção 2.3, são apresentados diversos trabalhos que tratam de um assunto que se refere mais à área de Computação Gráfica que à de Visão Computacional: a reconstrução tridimensional de objetos a partir de mapas de profundidade. Embora tal discussão pareça se distanciar do tema principal da tese, deve-se lembrar que a estimação de movimento é uma etapa necessária para a identificação das articulações de objetos rígidos articulados. Assim, este parece ser um caminho natural a ser seguido em projetos futuros e exemplifica a forte conexão que pode haver entre assuntos que, de antemão, possam parecer tão díspares.

Por fim, na Seção 2.4 são descritos o sistema de captura de vídeo 3D do IMPA e o conceito de vídeo 4D que é construído sobre ele.

2.1 Estimação de movimento

Movimento é um aspecto da visão tridimensional que nós, humanos, podemos interpretar com facilidade. O estudo de conceitos teóricos básicos envolvidos no seu entendimento permite tratar de problemas reais nos quais o movimento é crucial. É neste contexto que surge o conceito de fluxo óptico. Em oposição ao rastreamento de características, a estimação densa de movimento baseada na intensidade, também chamada de fluxo óptico, é um tópico que lida diretamente com os valores armazenados nos pixels de uma imagem.

A estimação de movimento pode ser usada em diversas aplicações práticas tais como a estabilização de vídeos com a remoção de tremores causados durante a gravação [Hansen *et al.*, 1994; Irani *et al.*, 1997; Matsushita *et al.*, 2006; Morimoto & Chellappa, 1997; Srinivasan *et al.*, 2005], o monitoramento de tráfego de veículos, o rastreamento de pessoas (e.g., em sistemas de vigilância) ou a inserção de figuras 2D ou modelos 3D em vídeos por meio do rastreamento automático de pontos de referência próximos.

A área de estimação acurada de movimento vem sofrendo uma rápida evolução, com significantes avanços no desempenho dos algoritmos ocorrendo a cada ano. A página de avaliação de fluxo óptico (<http://vision.middlebury.edu/flow/>) é uma ótima fonte de ponteiros para algoritmos de alto desempenho desenvolvidos recentemente.

Os modelos mais simples de movimento que existem são aqueles que modelam apenas movimentos translacionais de toda a imagem [Wedel *et al.*, 2009]. Tais modelos lidam com tópicos como estimação de movimento hierárquico (*coarse-to-fine*) [Anandan, 1989; Bergen *et al.*, 1992; Quam, 1984], técnicas baseadas na transformada de Fourier [Fleet & Jepson, 1990] e refinamento iterativo.

Algumas abordagens [Baker & Matthews, 2004; Bergen *et al.*, 1992; Fuh & Maragos, 1991; Lucas & Kanade, 1981; Rehg & Witkin, 1981; Shashua & Toelg, 1997; Shashua & Wexler, 2001] realizam uma modelagem do movimento por meio de parâmetros globais válidos também para toda a imagem, porém identificando e compensando um número maior de alterações tais como translação, rotação e aproximação (*zoom*), além de movimento perspectivo planar. No entanto, enquanto modelos de movimento paramétrico são úteis em uma variada gama de aplicações (tais como estabilização de vídeo e mapeamento em superfícies planares), a maioria dos movimentos em imagens é muito complicada para ser capturada por tais modelos de baixa dimensão.

Modelos de movimento baseados em *splines* comumente encontram largo uso no registro de imagens médicas [Bajcsy & Kovacic, 1989; Christensen *et al.*, 1997; Szeliski & Lavallée, 1996]. Técnicas de registro podem ser usadas tanto para rastrear o desenvolvimento ou progresso de um paciente com o passar do tempo quanto para casar imagens de diferentes pacientes, a fim de encontrar partes comuns e detectar variações ou patologias.

No outro extremo, tradicionalmente, algoritmos de fluxo óptico calculam uma estimativa de movimento independente para cada pixel, isto é, o número de vetores de fluxo calculados é igual ao número de pixels de entrada.

Abordagens quantitativas à Visão Computacional, tais como algoritmos de fluxo óptico baseados na intensidade das imagens, surgiram primeiramente no início da década de 80 [Horn & Schunck, 1981; Huang, 1981; Lucas & Kanade, 1981; Nagel, 1986]. O processamento de dados de profundidade tridimensionais (aquisição, casamento, modelagem e reconhecimento) também foi ativamente explorado desde então [Agin & Binford, 1976; Banno *et al.*, 2008; Besl & Jain, 1985; Curless & Levoy, 1996; Faugeras & Hebert, 1987; Kanade, 1987; Montenegro *et al.*, 2005; Pires, 2007; Pires *et al.*, 2005; Sá *et al.*, 2002; Velho *et al.*, 2004; Vieira *et al.*, 2005]. Algoritmos de rastreamento com técnicas baseadas diretamente na intensidade das imagens também foram desenvolvidos durante as décadas de 80 e 90 [Lucas & Kanade, 1981; Rehg & Kanade, 1994; Shi & Tomasi, 1994], sendo frequentemente aplicados ao rastreamento de faces [Lanitis *et al.*, 1997; Matthews & Baker, 2004; Matthews *et al.*, 2007]. Outras abordagens que fazem uso de fluxo óptico generalizado por pixel incluem modelos de movimento aprendido e em camadas. Aplicações dessas técnicas incluem *morphing* automatizado, interpolação de quadros em vídeo (câmera lenta) e interfaces de usuário baseadas em movimento (gestos).

Os métodos de fluxo óptico foram sendo melhorados com o passar do tempo [Anandan, 1989; Baker *et al.*, 2007; Barron *et al.*, 1994; Bergen *et al.*, 1992; Black & Anandan, 1996; Bolles *et al.*, 1987; Bruhn *et al.*, 2005; Horn & Weldon Jr., 1988; Nagel & Enkelmann, 1986; Papenberg *et al.*, 2006].

Outra aplicação prática que envolve a estimação de movimento e, portanto, faz parte da motivação do desenvolvimento de métodos que a calculem, é o processo de remoção de ruído e outros artefatos de vídeos [Kokaram, 2004]. Em contraste com removedores de ruído de uma única imagem, nos quais somente a informação disponível encontra-se na figura corrente, removedores de ruído em vídeo podem calcular a média ou tomar emprestado informação de quadros adjacentes. No entanto, para realizar esta tarefa sem introduzir borramento ou *jitter* (movimento irregular), é necessário estimar um acurado movimento por pixel [Gai & Kang, 2009].

Algoritmos para alinhamento de imagens e estimação de movimento em sequências de vídeo estão entre os mais amplamente usados em Visão Computacional. Como um exemplo, o alinhamento de imagens com capturas a taxa de quadro ($\frac{1}{24}$ s, conhecida como *frame rate* na literatura) ou a taxa de campo ($\frac{1}{30}$ s, do inglês *field rate*) é muito usado em filmadoras e câmeras digitais para implementar uma característica chamada estabilização de imagem.

Também referente a uma aplicação que envolve a estimação de movimento por pixel que é comumente usada é o desentrelaçamento de vídeo, processo de converter um vídeo formado por campos de linhas pares e ímpares, que se alternam, em um sinal não entrelaçado que contenha ambos os campos em cada quadro [de Haan & Bellers, 1998]. Duas técnicas de desentrelaçamento simples são a *bob*, que copia a linha acima ou abaixo da linha ausente do mesmo campo, e a *weave*, que copia a linha correspondente do campo posterior ou anterior. Os nomes vêm dos artefatos visuais gerados por estas duas técnicas simples: o *bob* introduz um movimento para cima e para baixo (conhecido como *bobbing*) ao longo de linhas horizontais fortes; o *weave* pode levar a um efeito de “zipeamento” ao longo de arestas que se transladam horizontal-

mente. Substituir esses operadores de cópia por valores médios pode ajudar, mas não elimina completamente tais artefatos [Szeliski, 2011]. Uma grande variedade de técnicas de aperfeiçoamento tem sido desenvolvidas para este processo, o qual frequentemente vem embarcado em *chips* de processamento digital de sinais encontrados em placas digitalizadoras de vídeo em computadores (já que vídeos com recepção via antena são geralmente entrelaçados, enquanto que os monitores de computadores não o são). Uma grande classe dessas técnicas estima movimentos locais por pixel e interpola os dados ausentes a partir de informação disponível em campos adjacentes no espaço e no tempo [Dai *et al.*, 2009].

O método que desenvolvemos neste trabalho de doutorado pode ser usado em aplicações mais gerais que aquelas que exigem que pessoas façam gestos em frente ao dispositivo de captura, permitindo lidar com reconstrução de cenas 3D e realidade aumentada. Naquela, pode-se atingir melhores registros de dados, uma vez que parâmetros do movimento de um objeto, tais como direção e velocidade, são conhecidos. Nesta, objetos artificiais da realidade aumentada podem responder melhor com relação ao movimento do usuário.

Neste trabalho, nós usamos a teoria de casamento entre grafos para encontrar a correspondência entre dois conjuntos de pontos. Esta abordagem tem sido usada para resolver muitos problemas de Visão Computacional tais como segmentação interativa de imagem natural [Noma, 2012], casamento de formas [Belongie *et al.*, 2002], colorização assistida por computador [Noma, 2010], rastreamento de objetos [Graciano, 2007] e casamento de pontos para registro não-rígido [Chui & Rangarajan, 2003], dentre outros [Conte *et al.*, 2004].

Nossa abordagem também é um passo intermediário na identificação de componentes rígidos de um objeto articulado [Kumar *et al.*, 2005; Torr & Zisserman, 1999]. Uma vez que sabemos, para cada bloco, a direção do movimento, nós podemos agrupar blocos similares em relação a esta característica. Por meio do acúmulo de tais similaridades através de uma sequência de vídeo, é provável que os componentes rígidos tornem-se claramente identificáveis e, por meio de comparação da proximidade entre tais componentes e levando em conta as diferenças nas direções de seus movimentos, poderia ser possível identificar os pontos de articulação também.

Outra área de pesquisa que poderia tirar vantagem da informação adicional fornecida pelo mapa de profundidade é o rastreamento 3D de movimento humano. Acreditamos que seja possível melhorar bons trabalhos desenvolvidos nesta área [Zhao & Liu, 2008].

2.2 Mapa de profundidade

A ENVOLVENTE TECNOLOGIA inerente aos atuais dispositivos sensores de profundidade foi inicialmente criada para prover interação natural a consoles de *vídeo-games*. Porém, toda pesquisa que vem sendo feita em áreas como Computação Gráfica e Visão Computacional tem mostrado muitos outros usos interessantes. Por isso vem sendo dada tanta importância ao desenvolvimento de programas que facilitem o uso desse equipamento.

Embora essa tecnologia seja relativamente nova, não será surpreendente se, em poucos anos, a mesma novidade estiver presente também em clientes computacionais leves, tais como *smart-phones* e *tablets*. Com o recente crescimento nas resoluções de tela e no poder de processamento gráfico desses pequenos dispositivos, tem sido possível alavancar essas tendências e implementar programas que lidam com dados estereoscópicos 3D, mas ainda não sem um atraso nos dados processados [Olson *et al.*, 2011]. Daí a urgência por algoritmos eficientes e rápidos.

A principal aplicação de dados capturados por tais equipamentos é geralmente referente a interação natural. Tais aplicações usam algoritmos antropométricos para estimar pose, esqueleto e o número de usuários em frente ao dispositivo. Alguns sistemas possuem algoritmos especializados para reconhecer seus usuários, mesmo que haja gêmeos idênticos entre eles [Leyvand *et al.*, 2011]. O reconhecimento de gestos tem sido aplicado para usar o Kinect™ para controlar outros dispositivos, objetivando uma interação mais fácil ou mais natural e permitindo o uso de computadores por meio de grande acessibilidade. Exemplos de trabalhos como estes incluem interfaces para usar em apresentações com slides [da Silveira, 2011] e a indústria de jogos [Microsoft®, 2012].

Dentre as abordagens usadas para identificação de usuários desses dispositivos sensores de profundidade,

destaca-se a tecnologia chamada *Kinect™ Identity* [Leyvand *et al.*, 2011], a qual seleciona um conjunto de três técnicas de identificação do usuário independentes: reconhecimento de faces, rastreamento da cor da vestimenta e estimação da altura. Tais técnicas foram selecionadas a partir de um conjunto maior e demonstraram ser as melhores que, ao mesmo tempo, eram robustas, sem exigir um processamento intenso por parte da CPU ou um consumo intenso de memória, e tão independentes quanto possível uma em relação à outra. Tal escolha indica a importância do desenvolvimento de algoritmos de treinamento que usam ambos os tipos de dados: textura e mapa de profundidade.

A necessidade de um melhor tratamento dos dados de profundidade em conjunto com os dados RGB e, conseqüentemente, o uso destes em algoritmos de segmentação de movimento é sentido até mesmo por desenvolvedores de programas especializados no reconhecimento de gestos, como o FFAST (sigla em inglês de *Flexible Action and Articulated Skeleton Toolkit*) [Suma *et al.*, 2011]. Tais programadores demonstraram interesse no desenvolvimento de rastreamento da cabeça em tempo real e estimação da torção do braço de um usuário. Nenhum destes é fornecido pela *middleware* OpenNI™ e as soluções para estes problemas certamente exigem que técnicas de Visão Computacional sejam aplicadas sobre ambos os dados de entrada.

2.3 Escaneamento 3D

APLICAÇÕES DE ESCANEAMENTO 3D TRADICIONAIS envolvem somente objetos estáticos e o principal desafio em tais aplicações é a produção de um modelo digital preciso da geometria da cena. Na última década, uma gama enorme de algoritmos têm sido proposta para tratar desse problema e atualmente pode-se considerar que ele esteja resolvido em variadas situações práticas. Assim, a atenção tem mudado para a lide com cenas dinâmicas, ou seja, aquelas nas quais os objetos estão se movendo.

Uma vez que a cena é dinâmica, à primeira vista parece que o problema não está bem definido. O que significa escanear uma cena em que a geometria é alterada constantemente? O que se espera como saída desse processo? O problema torna-se mais complexo quando se leva em conta o fato de que, para capturar qualquer movimento de forma acurada, é preciso capturar a cena a taxas em tempo real, o que, por si só, é um desafio tecnológico para o dispositivo de escaneamento.

Em relação a este último desafio, o sensor mais comumente usado para cenas dinâmicas é a chamada *câmera de vídeo de profundidade*. Tal câmera fornece uma imagem da cena, na qual cada pixel contém não somente a tradicional informação de intensidade como também a distância geométrica da câmera ao objeto naquele pixel. Algumas câmeras comerciais que geram esse tipo de informação a taxas de vídeo têm surgido nos últimos anos [3DVSystems, 2009; Canesta, 2009; Primesense, 2009; Vialux, 2009] e o estado da arte das tecnologias envolvidas vem melhorando rapidamente. Os preços também caíram e, com o surgimento do Kinect™, é possível adquirir atualmente uma câmera de vídeo de profundidade a um custo razoável. No nosso caso, em particular, temos acesso ao sistema de captura de vídeo 3D desenvolvido no IMPA pelo professor Luiz Velho [Velho *et al.*, 2004; Vieira *et al.*, 2005] e também a um dispositivo Kinect™.

A versão mais simples do escaneamento de uma cena dinâmica é a captura de movimento de um objeto 3D rígido por partes (tal como uma pessoa). Isto significa que, como resultado, não se está interessado na geometria precisa do objeto, mas sim no movimento bruto de um “esqueleto” representando suas partes rígidas, as quais geralmente são poucas, atingindo no máximo a ordem de algumas dezenas. A captura de movimento vinha sendo realizada, até pouco tempo, por meio do uso de equipamentos elaborados, os quais envolviam marcadores situados no objeto, sendo útil ter um dispositivo capaz de realizar a captura de movimento sem marcadores, baseado somente em câmeras de profundidade.

Uma versão mais desafiadora do problema é o escaneamento completo 3D de objetos dinâmicos tridimensionais rígidos por partes. A saída desejada não é só um modelo esquelético do objeto, mas também um modelo 3D completo da superfície de cada um dos componentes rígidos e uma descrição do movimento de cada componente no tempo. Seria ideal se tudo isso pudesse ser calculado em tempo real. No entanto, mesmo uma computação em um pós-processamento poderia ser útil para muitas aplicações, principalmente naquelas em que é impossível impedir que o objeto se mova durante o processo de escaneamento. Modelos de tais corpos articulados têm sido muito usados em Visão Computacional e em Computação Gráfica, em aplicações como detecção de pose e rastreamento de objetos em vídeo [Bregler & Malik, 1998; Hogg, 1983;

Sigal *et al.*, 2003; Yu *et al.*, 2002] e em dados 3D [Lin, 1999], além de casos que incluem a estimação e a renderização de movimento 3D [Allen *et al.*, 2002]. Na grande maioria das aplicações, a estrutura e os parâmetros do esqueleto articulado são especificados à mão, embora algumas aplicações otimizem os parâmetros das juntas individuais [Allen *et al.*, 2002; Gavrilu & Davis, 1996; Mikić *et al.*, 2001]. Um algoritmo que recupere, de um modo completamente não supervisionado, modelos articulados a partir de dados 3D pode diminuir enormemente a dependência, em muitas dessas aplicações, de interferência humana para determinação das articulações do modelo.

Mais difícil ainda de ser resolvido é o problema que envolve objetos na cena que se deformam com o passar do tempo de uma maneira não-rígida. Aqui, a descrição do movimento do objeto é mais complicada e, em alguns casos, nem mesmo é paramétrica.

Gool *et al.* [2002] ataca dois problemas que são de grande interesse. O primeiro deles é o casamento de imagens obtidas a partir de pontos de vista muito diferentes, o que ocasiona pouca sobreposição de informação. A solução baseia-se em características locais invariantes ao ponto de vista no momento da aquisição da imagem. É de grande interesse estender essa solução para dados de entrada compostos por imagens RGBD, uma vez que, com isso, o registro da textura e da geometria poderia ser feito a partir de quadros do vídeo que estivessem bem distantes no tempo, acabando com a necessidade de pertencerem a quadros consecutivos. O segundo problema tratado nesse artigo remete a como combinar modelos 3D parciais, obtidos a partir de um sistema de luz estruturada, sem a necessidade de uma boa inicialização de suas posições relativas. Essa necessidade é uma característica intrínseca do algoritmo ICP (do inglês *Iterative Closest Point*, ponto mais próximo iterativo), que é largamente usado para realizar o registro entre duas nuvens de pontos.

Outro artigo que trata de escaneamento 3D e que descreve um método que também poderia ser aplicado aos mapas de profundidade que capturamos é o de Wand *et al.* [2009], intitulado “*Efficient reconstruction of nonrigid shape and motion from real-time 3D scanner data*”. Nele, é discutida uma nova técnica para a reconstrução de uma única forma e seu movimento não-rígido a partir de dados de escaneamento 3D. O algoritmo descrito toma um conjunto de pontos não-estruturados amostrados com variância no tempo, os quais capturam visões parciais de um objeto deformável, e reconstrói uma forma única e um campo de deformação que casa os dados. Tal representação gera correspondências densas para a sequência inteira, bem como uma forma 3D completa em cada quadro. Não obstante, o algoritmo remove automaticamente artefatos ruidosos, tanto espaciais quanto temporais, a partir dos dados de entrada brutos.

O livro “Fotografia 3D”, de Montenegro *et al.* [2005], apresenta as notas de um curso dado no 25º Colégio Brasileiro de Matemática. Ele descreve a área de pesquisa conhecida como fotografia 3D, a qual integra diversos métodos para a reconstrução de objetos tridimensionais a partir de imagens. Neste processo estão incluídos a calibração de câmera, a aquisição de informações geométricas e fotométricas e o processamento dos modelos. O vídeo 3D é uma evolução obtida a partir da fotografia 3D.

O sistema descrito por Pekelný & Gotsman [2008] necessita de intervenção manual na etapa de segmentação das partes articuladas de um objeto, etapa esta que poderia ser automatizada. O método apresentado pelos autores é o primeiro capaz de reconstruir um modelo 3D completo de um objeto articulado usando uma única câmera de vídeo de profundidade. Este também é o primeiro algoritmo de captura de movimento sem marcadores que não requer uma malha modelo para o objeto. Conforme a revisão bibliográfica apresentada nesse artigo, há diversos algoritmos na literatura para a captura de movimento sem marcadores a partir de vídeo tradicional. Muitos deles [Cheung *et al.*, 2003; Mündermann *et al.*, 2006; Theobalt *et al.*, 2004] recuperam o movimento de um objeto articulado a partir de sequências de casco visual, adquiridas de silhuetas extraídas deste. Tais algoritmos requerem múltiplas câmeras de vídeo sincronizadas observando o objeto a partir de diferentes posições, o que acarreta complicações relacionadas a calibração multi-câmera temporal e espacial.

Há também diversas abordagens baseadas em modelos [Bray *et al.*, 2004; Grest *et al.*, 2005; Rodgers *et al.*, 2006] capazes de recuperar a pose de um objeto articulado a partir de uma única imagem de profundidade por meio do casamento com um modelo. Isto, no entanto, requer um modelo muito detalhado e calibrado a fim de que uma única imagem de profundidade seja suficiente.

O algoritmo proposto por Anguelov *et al.* [2004] recupera o modelo 3D completo de um objeto juntamente com a detecção da pose. Esse algoritmo também não exige marcadores, mas cada pose de entrada

deve ser uma malha completa da superfície. Segue que esse algoritmo não se aplica a dados oriundos de um vídeo de profundidade.

Também almejamos, em trabalhos futuros, a reconstrução de um modelo 3D completo do objeto. Para registrar múltiplos componentes de um objeto articulado, assume-se uma estrutura esquelética rígida por partes. Uma abordagem diferente foi proposta por Mitra *et al.* [2007]. Nele, o problema do registro é superado pela realização de uma quebra dos dados de entrada em pequenos pedaços quase rígidos. Esse algoritmo assume amostragens espacial e temporal densas e faz o registro usando propriedades cinemáticas da superfície espaço-tempo. Uma versão articulada desse algoritmo poderia ser possível, de modo que tal abordagem poderá ser considerada por nós para a resolução desse problema em tempo oportuno.

Diversos autores têm proposto algoritmos que recuperam tanto a geometria quanto o movimento do objeto. Por exemplo, Ashbrook *et al.* [1999] propõem um algoritmo para a reconstrução de um modelo articulado. Contudo, o algoritmo usa uma abordagem de registro de quadros pareados, tornando-se, portanto, muito ineficiente quando aplicado a longas sequências de dados de entrada. Os autores também admitem que o registro pareado, em oposição a uma abordagem mais global, leva a resultados sub-ótimos.

Os autores Knoop *et al.* [2006] propõem um sistema, chamado VooDoo, de rastreamento de movimento do corpo humano baseado em modelo. Este sistema modela o corpo humano como uma coleção de dez cilindros rígidos móveis (organizados hierarquicamente) e rastreia seu movimento no tempo usando o algoritmo de registro ICP. É preciso redimensionar manualmente o modelo para diferentes pessoas. O VooDoo não fornece bons resultados ao usar somente dados de profundidade como entrada e requer dados adicionais a partir de sensores de rastreamento 2D para conseguir bons resultados.

Um outro trabalho [Wand *et al.*, 2007] trata um problema similar, só que com um objeto deformável. Nele são realizadas reconstrução e otimização iterativas da estrutura 4D assumindo suavidade espaço-temporal. No entanto, o algoritmo é computacionalmente pesado e não é claro se ele é capaz de reconstruir um modelo 3D completo de um objeto deformável a partir de vídeo de profundidade.

Um dos algoritmos mais conhecidos para registro entre duas nuvens de pontos é o ICP. Dentre as diversas implementações que esse algoritmo já recebeu, uma de destaque é o chamado ICP dinâmico, apresentado no artigo de Mitra *et al.* [2007]. O algoritmo apresentado realiza o registro de grandes conjuntos de nuvens de pontos não estruturados, obtidos a partir de objetos móveis e maleáveis, sem, no entanto, calcular as correspondências. Dado como entrada um conjunto de quadros com uma densa amostragem espacial e temporal, como é o caso dos dados adquiridos por um Kinect™, o algoritmo apresentado nesse artigo explora a coerência temporal subjacente aos dados para calcular diretamente o movimento do objeto escaneado e levar todos os quadros para um sistema de coordenadas comum. Em contraste com o método usual, o qual realiza alinhamentos dois a dois entre quadros consecutivos, esse novo algoritmo calcula um movimento consistente global considerando múltiplos quadros. É adicionada, baseando-se na ordenação dos respectivos quadros, uma coordenada de tempo a todos os pontos de entrada, de modo que o problema de calcular o movimento de cada quadro é feito como sendo uma estimação de certas propriedades cinemáticas da superfície espaço-tempo resultante. Executando essa estimação para cada quadro como um todo, é possível calcular movimentos rígidos inter-quadros e, adaptando o método para realizar uma análise local da superfície espaço-tempo, estende-se o algoritmo básico para que ele trate também de objetos deformáveis.

Em se tratando de escaneamento tridimensional, o estado da arte atualmente é o projeto Kinect Fusion, da Microsoft Research [Izadi *et al.*, 2011]. Trata-se de um sistema que permite que um usuário realize a reconstrução tridimensional detalhada de uma cena em ambiente fechado bastando para isso mover um Kinect™ em torno dos objetos alvo. Somente os dados de profundidade capturados pelo Kinect™ são usados para rastrear a pose 3D do sensor e para reconstruir, em tempo real, modelos 3D da cena. O rápido desempenho do sistema deve-se a uma nova sequência de passos que é baseada em processamentos realizados em GPU. Os autores também descrevem o uso do sistema na segmentação de objetos (usada em realidade estendida) e na interação de usuário, com este postando-se diretamente em frente ao sensor, permitindo interações multi-toque em tempo real sobre qualquer tipo de superfície.

Conforme comentado anteriormente, dispositivos sensores de profundidade são geralmente usados para entretenimento e controle de outros dispositivos por meio de interface natural usando gestos. Algoritmos que fazem o reconhecimento de gestos primariamente necessitam de uma estimação e de um rastreamento do esqueleto humano. Aqui, esqueleto é entendido como um conjunto de juntas (pontos) e ossos (segmentos

de reta unindo juntas) que determinam os principais pontos de articulação de uma pessoa. Muitos programas baseiam-se na implementação que é fornecida pelo *middleware* OpenNI, que é de código fechado. Embora sejam frequentes questões da comunidade de programadores à empresa Primesense, detentora do código, sobre detalhes do algoritmo de rastreamento de esqueleto, a empresa sempre afirma que não há pretensões de publicá-lo. O que tem sido amplamente divulgado é o conceito por trás do algoritmo de rastreamento original da Microsoft, usado no Kinect™ [Shotton *et al.*, 2011].

2.4 Vídeo 3D e vídeo 4D

O MÉTODO QUE FOI DESENVOLVIDO também aplica-se a dados oriundos de um sistema de vídeo 3D em tempo real proposto por Vieira *et al.* [2005]. Neste capítulo é dada apenas uma breve explicação sobre o funcionamento do sistema de captura existente, uma vez que há um detalhamento completo que pode ser consultado na dissertação de mestrado [Pires, 2007]. Após esta descrição sucinta, parte-se para a definição do vídeo 4D e para a apresentação da modelagem que já foi feita até então.

2.4.1 Vídeo 3D

O sistema de vídeo 3D [Montenegro *et al.*, 2005; Sá *et al.*, 2002; Velho *et al.*, 2004], constituído por projetor e câmera, obtém imagens de profundidade de objetos por meio da projeção de *slides* com um padrão de faixas coloridas. Tal procedimento permite a obtenção, em tempo real, tanto do modelo $2\frac{1}{2}$ D dos objetos quanto da textura dos mesmos, segundo uma técnica denominada *luz estruturada*. Os dados são capturados a uma taxa de 30 quadros por segundo e possuem alta qualidade: resoluções de 640×480 pixels para a textura e de 90×240 pontos (em média) para a geometria.

O trabalho desenvolvido no mestrado consistiu em um método para detecção, rastreamento e casamento espacial de componentes conexos presentes em um vídeo 3D. A informação de imagem do vídeo (textura) foi combinada com posições tridimensionais (geometria) a fim de alinhar partes de superfícies que são vistas em quadros subsequentes. A abordagem que adotamos consiste na detecção de características salientes no espaço do mundo, provendo um alinhamento de geometria mais completo. O processo de registro é feito segundo a aplicação do algoritmo ICP, introduzido por Besl & McKay [1992]. Resultados experimentais bem sucedidos [Pires *et al.*, 2005] corroboraram nosso método.

A Figura 2.1 fornece uma visão geral do *pipeline* do processamento 3D enquanto que as etapas que constituíram o trabalho desenvolvido no mestrado são ilustradas na Figura 2.2. O Apêndice B detalha todo o processo de captura de vídeo 3D atualmente instalado no IMPA.

2.4.2 Vídeo 4D

O vídeo resultante do sistema de captura do IMPA pode ser, de certa forma, considerado um vídeo $3\frac{1}{2}$ D. Essa conclusão vem do fato de cada quadro do vídeo fornecer um modelo $2\frac{1}{2}$ D da cena gravada, de modo que se tem a ordenada, a abscissa e a cota de cada ponto visível pela câmera, mas não dos pontos oclusos. Pode-se tomar como exemplo de uma imagem $2\frac{1}{2}$ D a apresentada na Figura 2.1 (f). É importante observar que não se tem dados sobre as partes sombreadas do coelho de pelúcia, do cubo e da mão, uma vez que tais partes não estão no campo de visão da câmera. O tempo adiciona uma dimensão ao vídeo, totalizando 3 dimensões e meia.

A evolução para o conceito de vídeo 4D, portanto, almeja o aumento de meia dimensão a partir do vídeo $3\frac{1}{2}$ D. Isso pode ser obtido segundo o casamento das diversas nuvens de pontos de um mesmo objeto, fornecidas a cada quadro do vídeo. Conforme um objeto move-se diante da câmera ou, similarmente, conforme a câmera movimenta-se em relação à cena que é gravada, novos detalhes, tanto referentes à textura quanto à geometria dos objetos, são capturados. Esses novos dados permitem que se vá completando o modelo do objeto em questão pouco a pouco, de acordo com um casamento realizado entre os dados que já faziam parte do modelo e os que são fornecidos a cada quadro do vídeo. Tal casamento fornecerá um modelo tridimensi-

(a)

(b)

(c)

(d)

(e)

(f)

Figura 2.1: Sequência de passos do sistema do IMPA. (a) *Projeção de slides sobre objetos*; (b) *Padrões projetados*; (c) *Fronteiras das faixas identificadas*; (d) *Captura da textura*; (e) *Cálculo da profundidade*; (f) *Modelagem 3D*.

Figura 2.2: Etapas que constituíram o trabalho do mestrado. As etapas consistiram na captura, segmentação, rastreamento, registro e integração.

onal do objeto, modelo esse que, em seguida, seria inserido novamente na cena, compondo o que chamamos de vídeo 4D reconstruído.

De início, portanto, deve-se estudar as diversas formas de se obter esse modelo 3D, composto por geometria e textura, a partir de conjuntos de nuvens de pontos ligeiramente diferentes entre si quando oriundas de quadros consecutivos do vídeo.

Com a inserção do modelo tridimensional completo do objeto na cena e com a implementação de uma interface gráfica, o usuário poderá interagir com o vídeo, assistindo-o a partir de pontos de vista diferentes daqueles em que se encontrava a câmera no momento da gravação da cena. É importante enfatizar que não se trata apenas de uma mudança do ponto de vista em que o vídeo é assistido: toda a geometria e a textura que já foram reconstruídas baseando-se nos quadros gravados estará disponível para visualização.

Há três formas de se obter o vídeo 4D a partir de um vídeo 3 $\frac{1}{2}$ D:

Modo *on-line*: Também podendo ser chamado de modo em tempo real, a reconstrução do modelo é feita concomitante à captura. Conforme cada quadro do vídeo é capturado pela câmera, o processamento já é feito, permitindo que o usuário interaja com o vídeo 4D que está sendo construído naquele mesmo momento.

Modo *off-line*: Semelhante ao modo *on-line*, com a diferença de que os quadros não estão sendo capturados no mesmo momento. Em vez disso, o vídeo 4D é construído a partir dos quadros que vão sendo lidos a partir de um arquivo.

Modo em lote: Neste modo, todo o vídeo 4D é reconstruído e, somente depois, é disponibilizado para interação com o usuário, o qual usufruirá de uma visualização que contenha os dados obtidos de todos os quadros do vídeo.

Há uma gradação na dificuldade dos problemas que podem ser enfrentados de acordo com os elementos presentes na captura de um vídeo 3D. Pode-se citar os seguintes exemplos:

Apenas o cenário, sem objetos presentes. Este é o caso mais simples. Um exemplo de aplicação seria o desenvolvimento de algoritmos que tentam inferir o caminho realizado pela câmera durante a gravação.

Um objeto rígido. Envolve, como principal atividade, a reconstrução tridimensional completa de um modelo do objeto a partir de todas as imagens $2\frac{1}{2}$ D obtidas. Essa reconstrução exige que seja feito o registro de nuvens de pontos de quadros subsequentes. Para a visualização, pode-se aplicar uma malha sobre o volume.

Um objeto rígido mais o fundo da cena. Neste exemplo pode ser trabalhada a tarefa de segmentação e remoção do fundo, comum nos problemas de Visão Computacional.

Vários objetos rígidos, com movimento relativo entre eles. Aqui, passa a ser interessante a identificação e o rastreamento de cada objeto, incluindo casos de sobreposição e toque de um objeto no outro.

Um objeto rígido composto por partes articuladas. Principal caso que pode ser beneficiado com o avanço do método de estimação de movimento desenvolvido.

Um objeto deformável. Há grande interesse em se trabalhar com faces, mãos e corpo humano, os quais constituem um exemplo de objeto deformável. Nesses casos, a adaptação e o casamento entre malhas de geometrias consecutivas no vídeo recebe um tratamento diferenciado, que já vem sendo tratado na literatura.

A sequência de etapas descrita possui diversos elementos em que se pode criar à vontade. Pode-se, por exemplo, alterar o modelo (textura ou geometria) antes de incluí-lo na cena novamente.

Capítulo 3

Metodologia

3.1 Perspectiva geral

PARA O DESENVOLVIMENTO DO MÉTODO APRESENTADO, consideramos que as causas de mudanças em uma cena são fruto do movimento de objetos presentes nesta. Outras opções seriam o movimento da câmera ou modificações na iluminação [Klette *et al.*, 1998].

Szeliski [2011] mostra que a complexidade do tipo de movimento a ser estimado pode variar consideravelmente. Uma vez que uma métrica de erro tenha sido adotada, é preciso escolher uma técnica de *busca* adequada. Pode-se, por exemplo, testar exaustivamente todos os possíveis alinhamentos, ou seja, fazer uma busca completa. Na prática, essa abordagem pode se mostrar muito lenta, de modo que técnicas hierárquicas baseadas em pirâmides de imagens são normalmente usadas [Anandan, 1989; Bergen *et al.*, 1992; Quam, 1984]. Outra alternativa é usar a transformada de Fourier para acelerar os cálculos [Fleet & Jepson, 1990]. Para movimentos mais complexos, modelos de movimento paramétrico por partes usando *splines* podem ser usados [Kybic & Unser, 2003; Szeliski & Coughlan, 1997; Szeliski & Lavallée, 1996].

Segue que a versão mais geral (e desafiadora) de estimação de movimento é a presença de movimentos múltiplos e independentes (e às vezes não-rígidos, como um tecido sendo balançado). Nesses casos, é necessária a estimação do movimento independente de cada pixel, a qual é conhecida como fluxo óptico. Isso normalmente envolve a minimização da diferença entre o brilho ou a cor de pixels correspondentes somadas por toda a imagem [Szeliski, 2011]. Uma das abordagens clássicas para esse problema é realizar o somatório localmente sobre regiões que se interceptam (abordagem baseada em *blocos* ou em *janelas*). É justamente a busca por uma solução para esse problema mais geral que motiva o método que desenvolvemos.

Nem sempre o fluxo óptico pode ser identificado com deslocamentos locais. Como exemplo, considere uma esfera desprovida de textura em sua superfície e que esteja rotacionando em torno de um eixo em frente à câmera. Nenhum fluxo óptico poderá ser observado, embora ocorra um movimento dos pontos do objeto. Por outro lado, para uma esfera estática e mudanças nas condições de iluminação, um fluxo óptico poderá ser observado na superfície da esfera, embora não haja movimento dos pontos em sua superfície [Horn, 1986; Klette *et al.*, 1998].

Pode-se encarar a estimação de movimento como sendo um problema que envolve dois quadros, no qual o objetivo é calcular o campo de movimento que alinha pixels de uma imagem com os de outra. Na prática, a estimação de movimento é usualmente aplicada a um vídeo, em que uma sequência completa de quadros está disponível para realizar esta tarefa.

De acordo com Trucco & Verri [1998], podemos dividir o problema de movimento em dois subproblemas:

Correspondência: Quais elementos de um quadro correspondem a quais elementos do quadro seguinte de uma sequência?

Reconstrução: Dado um número de elementos correspondentes e, possivelmente, o conhecimento dos pa-

râmetros intrínsecos da câmera, o que nós podemos dizer sobre o movimento 3D e a estrutura do mundo observado?

Para resolver o problema da correspondência, pode-se usar duas estratégias:

Métodos diferenciais: aplicam medidas densas, ou seja, calculadas em cada pixel da imagem. Tais métodos usam estimativas de derivadas no tempo e, portanto, requerem que as imagens da sequência sejam amostradas bem próximas (no tempo) umas das outras.

Métodos de casamento: aplicam medidas esparsas, ou seja, calculadas somente em um subconjunto dos pontos das imagens. O filtro de Kalman é exemplo de uma técnica para casamento e rastreamento de características esparsas em imagens no decorrer do tempo.

O método que desenvolvemos busca uma solução para o problema da correspondência e, apesar da coincidência no nome (o método baseia-se no casamento entre grafos), trata-se de um método diferencial (e não de casamento), pois calcula, para cada pixel, a diferença existente entre características que envolvem sua aparência e sua posição no espaço em relação aos outros pixels da imagem. Nossa abordagem usa como base o reconhecimento estrutural de padrões realizado por meio de casamento entre grafos.

3.2 Sequência de etapas

O MÉTODO PROPOSTO é formado por uma sequência de passos:

- aquisição de dados;
- processamento da textura;
- processamento do mapa de profundidade;
- algoritmos de grafos;
- visualização dos dados.

O fluxo de dados é esquematizado na Figura 3.1.

3.3 Dados de entrada

NOSSO ALGORITMO recebe como entrada uma sequência de pares de imagens registradas: textura colorida e mapa de profundidade. A textura é representada por três canais (R, G e B) e o mapa de profundidade é uma imagem em níveis de cinza, sendo também conhecido como canal D (do inglês *depth*, de “profundidade”). Assim, podemos dizer que a entrada de nosso algoritmo constitui-se de uma imagem de 4 canais, a qual vem sendo chamada na literatura como imagem RGBD. Uma vez que o mapa de profundidade capturado já está registrado com a imagem da textura, não há necessidade de conhecer os parâmetros intrínsecos e extrínsecos de calibração entre o sensor de luz infra-vermelha e a câmera.

Além de dados oriundos de um Kinect™, o programa desenvolvido também aceita diversas outras fontes de entrada, tais como arquivos em formato de vídeo (AVI, MPEG, etc.), web-cam, arquivos de vídeo 4D [Vieira *et al.*, 2005] e sequências de arquivos de imagens (BMP, JPEG, etc.). Isso pode facilitar a comparação dos resultados (e eventualmente confirmar as vantagens) no uso da informação adicional do mapa de profundidade com relação a abordagens que usam apenas a informação fornecida pela textura.

A opção de carregar sequências de arquivos de imagens a partir do disco é uma abordagem interessante que nós adotamos para fazer processamento *off-line*. Isso pode ser necessário em situações nas quais os parâmetros para os algoritmos implementados implicam em uma carga excessiva sobre a CPU, aumentando assim o intervalo entre duas capturas consecutivas. Se o intervalo citado é muito grande, a resolução temporal passa a ser muito pequena e uma grande gama de movimentos passa a ser despercebida. Para evitar que

Figura 3.1: Fluxo de dados através do método implementado. A visualização dos dados pode tanto ser feita nos resultados finais quanto a cada passo intermediário.

isso ocorra, nós podemos obter todos os dados desejados de uma só vez, gravando-os em disco sem realizar qualquer tipo de processamento, a fim de acelerar a etapa de aquisição; só então é que, numa fase posterior, podemos rodar o respectivo programa tendo essas imagens como entrada, carregando cada uma somente quando o seu processamento for necessário no decorrer da execução do método.

Quando um dispositivo Kinect™ está disponível, o método desenvolvido também é capaz de lidar com a captura do mapa de disparidade, da textura em níveis de cinza, da imagem da cena iluminada com luz infravermelha e de uma máscara que indica onde o mapa de profundidade é válido. Uma vez que não usamos esses dados para processamento do método desenvolvido, apenas a sua visualização foi implementada.

O mapa de profundidade é definido por um volume de visualização, o qual, por sua vez, é limitado por dois planos: um que é mais próximo ao dispositivo de captura e outro que é mais distante ao mesmo. Estes planos são conhecidos na literatura de Computação Gráfica como plano próximo (z_{near}) e plano distante (z_{far}). Se tomarmos $z_{\text{near}} = 1$, $z_{\text{far}} \rightarrow \infty$ e invertermos o valor da cota dos pontos no mapa de profundidade, então o terceiro canal desta nova imagem processada conterá o inverso da profundidade, ou seja, a disparidade [Okutomi & Kanade, 1993]. Esta medida, também fornecida pelo Kinect™ e passível de visualização (como mapa de disparidade) no sistema desenvolvido, é conveniente em diversos casos, uma vez que, para câmeras que se movem em cenas ao ar livre, o inverso da profundidade até a câmera frequentemente é uma parametrização mais estável, em termos de aproximação numérica, que a distância 3D.

3.3.1 Transformações nos pixels

No decorrer de toda a sequência de etapas que compõem o método desenvolvido, um operador diádico (de duas entradas) representado por c (de *custo*) é aplicado a cada par de pixels de um par de imagens RGBD. No domínio contínuo, isto pode ser denotado como:

$$c(\mathbf{p}_m, \mathbf{p}_i) = d_{\text{RGB}}(\mathbf{p}_m, \mathbf{p}_i) + d_{\text{XYZ}}(\mathbf{p}_m, \mathbf{p}_i), \quad (3.1)$$

em que $\mathbf{p}_m$ e $\mathbf{p}_i$ são vetores que representam pixels em duas imagens RGBD consecutivas na captura do vídeo de entrada, e d_{RGB} e d_{XYZ} são funções que calculam distâncias nos espaços RGB de cores e XYZ de coordenadas do mundo (não confundir com o espaço XYZ de cores).

No caso das imagens com que trabalhamos, podemos considerar que o domínio d -dimensional da função c é $d = 2$, uma vez que as imagens são planares e cada pixel $\mathbf{p}$ pode ser identificado por suas coordenadas (x, y) . Ainda, levando em conta que a imagem de profundidade é registrada com a imagem de textura, temos que, para um pixel de coordenadas (x, y) na imagem de textura, podemos tanto obter o valor de sua cor no espaço RGB quando o valor de sua profundidade z na mesma coordenada (x, y) , só que desta vez na imagem de profundidade. Assim, podemos considerar que, em vez de a entrada serem duas imagens, uma de textura e outra de profundidade, temos apenas uma imagem, porém com quatro canais. Nesta, além dos três canais RGB comuns, que representam sua cor, também teríamos um quarto canal que, em vez de representar a transparência, como normalmente é usado na Computação Gráfica, este canal passa a representar a profundidade (vide esquema na Figura 3.2).

Embora os valores sobre os quais o operador c atua sejam compostos pelos escalares R , G , B e D , todos os cálculos são realizados em espaços vetoriais, sejam eles representativos de espaços de cores ou do espaço físico tridimensional.

Desta forma, o domínio e o contra-domínio do operador c que atua sobre cada pixel das imagens de entrada ficam bem definidos:

$$c: [0, 255]^4 \rightarrow \mathbb{R} \quad (3.2)$$

3.3.2 Pré-processamento dos dados de entrada

O Kinect™ fornece um mapa de profundidade com 11 bits de precisão. Dessa forma, para que possamos visualizar esse mapa como uma imagem em níveis de cinza, é preciso normalizar os valores dos pixels para o intervalo $[0, 255]$. Empiricamente, encontrou-se o valor 0,05 como um fator razoável.

Figura 3.2: Imagem RGBD. O pixel $p = (x, y)$ pertence a uma imagem com quatro canais. Com a mesma coordenada (x, y) é possível acessar os valores dos canais R (red), G (green), B (blue) e D (depth).

3.4 Processamento da textura

Nosso MÉTODO inclui a implementação de diversos processamentos da textura e dois deles são particularmente importantes para obter os resultados desejados:

- reflexão na direção horizontal;
- redução da resolução.

Ambos os processamentos são aplicados nas imagens da textura e do mapa de profundidade.

A transformação de reflexão na direção horizontal é usada por razões pragmáticas: quando um usuário interage com o sistema, é mais fácil olhar para os resultados como se ele estivesse de frente para um espelho. Uma vez que o dispositivo de aquisição comumente é posicionado em um local próximo à tela do computador, de frente para o usuário, uma reflexão na direção horizontal é necessária para fazer com que interações em tempo real tornem-se mais fáceis.

O processamento de redução da resolução, além de acelerar todos os cálculos que são feitos, também é muito relacionado a dois outros fatores:

- a velocidade de cada objeto que se move;
- a distância, até o dispositivo de captura, de cada objeto que se move.

Essas duas quantidades interferem na precisão da detecção de movimento. Resoluções baixas permitem apenas a detecção de movimentos que envolvem a modificação de muitos pixels nas imagens originais, uma vez que muitos pixels são representados por apenas um pixel na imagem de menor resolução. Assim, o movimento será percebido somente se o objeto localiza-se próximo ao dispositivo de captura ou se sua velocidade é suficiente para envolver uma quantidade relativamente considerável de pixels em quadros consecutivos.

A redução da resolução da imagem foi implementada por meio da construção de blocos. Um bloco é definido como sendo uma região retangular na imagem, portanto permitindo blocos quadrados. Todos

Figura 3.3: Volume de visualização ou frustum: somente objetos presentes no interior do tronco de pirâmide (destacado na cor cinza e limitado pelos planos PRÓXIMO e DISTANTE) é que são projetados no plano da imagem.

os pixels que pertencem a essa região retangular têm seus valores (RGB ou de profundidade) alterados para os respectivos valores do bloco, os quais serão explicados na próxima seção. A fim de simplificar esta conversão, nós consideramos uma grade na qual a distância entre linhas verticais é definida como sendo o valor do comprimento do bloco e a distância entre linhas horizontais é o valor da altura do bloco. Nós não lidamos com multi-resolução, de modo que os blocos possuem as mesmas dimensões por toda a imagem. Isto resulta em uma grade regular, igualmente espaçada em cada direção. Por outro lado, o método implementado pode lidar perfeitamente com mudanças no tamanho dos blocos de um quadro a outro.

3.4.1 Cálculo dos blocos médios

Como pode ser inferido do último parágrafo, há uma escolha importante a ser feita quando da construção de um bloco: seus valores RGB e de profundidade. Nosso método permite duas opções:

1. os valores originais do pixel superior esquerdo de um bloco;
2. a média aritmética dos valores de todos os pixels que compõem o bloco.

Dentre os filtros usados, também encontra-se o da mediana, mas ele é aplicado à imagem que é gerada depois de construir um grafo representativo do quadro; portanto, este filtro só será comentado posteriormente.

3.5 Processamento do mapa de profundidade

PROCESSAMENTOS REALIZADOS EM MAPAS DE PROFUNDIDADE são muito parecidos com os realizados em imagens de textura, uma vez que mapas de profundidade também são imagens. Assim, podemos explorar algumas vantagens pelo fato de que mapas de profundidade são imagens em níveis de cinza que podem ser vistas como uma superfície tridimensional.

Portanto, foi implementado um processamento do mapa de profundidade que permite escolher os dados que estão cercados por dois limiares. Este processamento baseia-se somente na determinação destes dois parâmetros, os quais determinam os planos próximo (corte frontal ou anterior) e o distante (corte traseiro ou posterior) do volume de visualização (também conhecido como *frustum*, vide Figura 3.3). Com efeito, estes valores controlam um ajuste fino na definição de um tronco de pirâmide que é restrito à área onde os objetos de interesse estão presentes. Na verdade, nós podemos precisamente remover o fundo e objetos que estão mais próximos ao dispositivo de captura que os objetos-alvo. Com essa limiarização de fundo e de frente, só são incluídos no grafo os pontos que fazem parte da geometria de interesse.

Na Figura 3.4 é mostrada uma imagem da textura de uma cena que contém objetos em três profundidades diferentes: o braço de uma pessoa segurando a moldura de um quadro à frente, um bebê no meio e a parede ao fundo. Podemos remover o fundo aproximando o plano distante (usamos o valor 93). Da mesma forma,

Figura 3.4: Controlando os limiares do volume de visualização: a partir de uma imagem em que todos os elementos pictóricos estão presentes dentro do frustum, é possível remover o fundo (aproximando o plano distante) ou remover objetos que se encontram mais próximos que o de interesse (afastando o plano próximo).

Figura 3.5: Exemplo de mapeamento de textura sobre a disparidade colorizada. Cada imagem participa com 50% no valor RGB. Na imagem de disparidade, os valores das cores vão de vermelho a azul, seguindo a ordem do espectro eletromagnético, conforme a distância ao dispositivo de captura aumenta.

o braço e a moldura podem ser removidos afastando o plano próximo (valor 47). Fazendo o mapeamento da textura apenas nos pontos que pertencem ao volume de visualização, conseguimos isolar o objeto de interesse, causando uma melhora nos resultados e no desempenho de execução. Note que, nas imagens de baixo na mesma figura, o valor 0 da profundidade, que é visto como a cor preta, foi trocado por um padrão quadriculado, comum em programas de processamento de imagens e que facilita a distinção de áreas em que não há conteúdo sendo exibido.

Também relacionado ao mapa de profundidade é o mapeamento da textura. Este procedimento mapeia a textura no mapa de profundidade ou no mapa de disparidade, escolha esta que fica a critério do usuário. Note que, para que isso seja possível, é necessário que uma das imagens esteja registrada com relação à outra. Este problema foi resolvido e maiores detalhes técnicos sobre a solução adotada são descritos no Apêndice A.

O resultado desse processamento da profundidade possui uma implicação muito importante na construção dos grafos, uma vez que, tendo sido o objeto de interesse bem delimitado pelos limiares comentados, elimina-se uma grande quantidade de dados que não são necessários para a estimação do movimento. Nós também podemos colorizar estas duas imagens em níveis de cinza a fim de obter uma melhor visualização dos dados. O mapeamento da textura visa confirmar a acurácia do registro entre as imagens de textura e profundidade ou entre as imagens de textura e disparidade. Embora nós não tenhamos desenvolvido um visualizador que nos permita alterar o ponto de vista da câmera de modo que fosse possível perceber facilmente o correto alinhamento dos dados registrados, nós podemos confirmar esta característica por meio da média dos valores dos pixels das duas imagens, simulando uma transparência, como um canal alfa (veja a Figura 3.5). Com $\alpha = 0$, pixels do mapa de profundidade são totalmente transparentes (ou, equivalentemente, pixels da textura são totalmente opacos). Já com $\alpha = 1$, ocorre o contrário: são os pixels da textura (do mapa de profundidade) que são totalmente transparentes (opacos). Com o controle fornecido pela interface

gráfica, é possível variar suavemente entre os dois extremos.

Alguns dos processamentos da textura e do mapa de profundidade podem ser aplicados em processos paralelos, uma vez que são independentes entre si. Esta característica pode acelerar o fluxo dos dados através da sequência de etapas, principalmente porque uma grande variedade de máquinas modernas possui mais de uma CPU, CPUs com muitos núcleos ou até mesmo GPUs programáveis com muitos núcleos. Apesar disso, ainda não paralelizamos nossos algoritmos e, portanto, atualmente, os dados seguem os passos apresentados como uma sequência serial de etapas.

3.6 Abordagem baseada em grafos

COM O OBJETIVO de encontrar um casamento entre dois grafos, temos que levar em conta características relevantes que descrevam os pontos e temos que ter um modo de comparar tais quantidades. Assim, cada quadro gera um grafo cujos vértices são derivados de propriedades dos blocos. Nós usamos seis valores para cada pixel nos dados de entrada: os dados RGB, extraídos dos canais de cores, e os dados (x, y, z) , com x e y sendo coordenadas da textura e z sendo a distância do objeto ao dispositivo de captura. A representação do quadro é dada por um grafo de atributos relacionais (ARG, da sigla em inglês de *Attributed Relational Graph*), permitindo o armazenamento e a comparação de informação estrutural, temporal e quantitativa. O reconhecimento da direção do movimento é feito através de um casamento inexato entre grafos. Esta abordagem permite diferenças entre o grafo modelo e o de entrada, tanto no conjunto de vértices quanto no conjunto de arestas.

Um ARG é um grafo cujos vértices representam objetos enquanto suas arestas denotam relações entre eles. Objetos podem ser caracterizados por um número finito de atributos (numéricos ou simbólicos), tais como área, perímetro, cor e forma. As relações correspondem, na maioria das vezes, a uma medida de distância ou similaridade [Graciano, 2007]. Com o conteúdo de cada quadro sendo representado por um ARG, a segmentação de movimento se resume a um casamento entre grafos, consistindo na determinação de um mapeamento de um conjunto de vértices de um ARG no conjunto de vértices de um outro.

Cada grafo é tratado como um grafo completo, no sentido de que cada vértice é conectado com todos os outros vértices. De fato, nós calculamos o valor de uma função custo envolvendo a distância entre os valores RGBD de cada dois conjuntos de vértices: (v_m, v_i) , em que v_m e v_i são vértices dos grafos modelo e de entrada, respectivamente. O par que minimiza o valor desta função custo é adicionado ao casamento.

Enquanto os vértices de um grafo armazenam características de pontos, incluindo informações sobre a posição destes pontos, por outro lado relações estruturais são armazenadas nas arestas. No entanto, neste caso, uma vez que tanto a informação estrutural quanto a de aparência podem ser calculadas a partir dos dados armazenados em cada vértice, as posições dos blocos na imagem são usadas para inferir o módulo (tamanho) e a direção dos vetores nos espaços RGB e XYZ. Isso evita a necessidade de se armazenar dados nas arestas da estrutura de dados que representa os grafos.

3.6.1 Construção dos grafos

Os grafos modelo e de entrada são construídos a partir de pares consecutivos de textura e de mapa de profundidade dos quadros de entrada, conforme ilustrado na Figura 3.6. Logo, no início do procedimento de aquisição, nós podemos construir apenas um grafo. Conforme os quadros seguintes são capturados, o grafo de entrada relativo ao quadro imediatamente anterior é atribuído ao grafo modelo e um novo grafo de entrada é construído a partir dos novos dados capturados.

Com o objetivo de construir o grafo, nós consideramos a representação das imagens de entrada (textura e mapa de profundidade) composta por blocos. Dados imagem e os parâmetros dos blocos, nós podemos calcular quantos blocos compõem a nova representação, sendo suficiente para isso fazer a divisão entre o número de linhas e colunas de cada um. Portanto, uma vez que a resolução das imagens de entrada não se altera, as dimensões dos blocos estão diretamente relacionadas com o tamanho dos grafos que são criados, influenciando fortemente no desempenho da aplicação.

Figura 3.6: Construção e casamento de grafos a partir de quadros consecutivos. Cada grafo é formado a partir de uma imagem RGBD. O grafo que serviu de entrada no tempo $t + 1$ torna-se o grafo modelo no tempo $t + 2$.

Cada bloco é candidato a possuir um vértice representativo no grafo, sendo eleito com base no valor de sua profundidade (z). Somente se este valor pertence ao volume de visualização definido anteriormente, na etapa de processamento do mapa de profundidade, um novo vértice é criado, tem seus campos preenchidos com os dados do bloco e é inserido no grafo. Uma consequência desta abordagem é que somente os pixels que possuem valores de profundidade não-nulos são considerados. Assim, pixels que pertencem a áreas de sombra no mapa de profundidade ou que estão muito próximos ou muito longe do dispositivo de captura para ter valores confiáveis estão fora de escopo. Esta perspectiva trata o mapa de profundidade como uma máscara de validação para os pixels da textura.

Uma vez que nós aplicamos a redução da resolução, os valores que compõem os campos dos blocos sempre podem ser obtidos a partir do pixel superior esquerdo, ainda que a opção de calcular os blocos médios esteja ativada.

A estrutura de dados do grafo direcionado é implementada por uma lista de adjacência, na qual há um vetor de vértices e , para cada vértice, há uma lista de arestas de saída. Os dados armazenados nos vértices incluem linha, coluna (ambos relacionados à textura), cota (coordenada z , valor da imagem do mapa de profundidade na mesma linha e coluna), valores dos canais vermelho, verde e azul da imagem de textura e um marcador de tempo, que indica o número de quadros capturados desde que o processo de captura foi iniciado.

3.6.2 Homomorfismo entre grafos

Com o intuito de se relacionar as imagens RGBD de dois quadros consecutivos e inferir o movimento realizado no intervalo entre as duas capturas, realiza-se o casamento entre dois grafos: o modelo e o de entrada. O grafo modelo representa o último par de quadros (imagens de textura e de profundidade), capturados antes dos atuais, os quais são representados pelo grafo de entrada.

Um casamento é um par ordenado de descritores de vértices, com o primeiro se referindo a um vértice do grafo modelo e o segundo relativo a um vértice do grafo de entrada. Veja a Figura 3.7 para um exemplo de casamento entre grafos.

Figura 3.7: Exemplo de casamento entre grafos. Note que este casamento é um homomorfismo: todos os vértices do grafo modelo são casados, há vértices no grafo de entrada que não participam de nenhum casamento e outros que participam mais de uma vez.

3.6.3 Casamento inexato entre grafos: Busca heurística

A abordagem genérica, envolvendo grafos de deformações, permite que o algoritmo adotado reduza as comparações estruturais a cálculos de distâncias entre pontos 3D [Noma *et al.*, 2012].

Para cada vértice pertencente ao grafo modelo, nós encontramos o vértice no grafo de entrada que esteja mais próximo de acordo com uma distância medida por uma função custo. Eventualmente, os vértices casados possuem as mesmas coordenadas (x, y) de textura, indicando que não ocorreu movimento algum naquele local entre os dois pares de quadros considerados.

3.6.4 Função custo

A medida de erro escolhida é uma função custo c dada por uma média ponderada entre duas distâncias: d_{RGB} e d_{XYZ} :

$$c = \alpha \cdot d_{\text{RGB}} + (1 - \alpha) \cdot d_{\text{XYZ}}, \quad (3.3)$$

com $\alpha \in [0, 1]$. O valor d_{RGB} mede a distância entre as cores dos blocos sendo comparados no espaço RGB, enquanto o valor d_{XYZ} mede a distância entre as coordenadas (x, y) da textura e entre os valores z de profundidade. Como pode ser visto, o parâmetro α controla o quanto cada distância é considerada no valor final da função custo. Tomando valores de α de $0 \rightarrow 1$, o valor de cada peso é alterado de acordo. Este arranjo nos permite atribuir importâncias diferentes para cada distância calculada, enquanto a soma dos pesos ainda permanece 1.

A função custo c também pode ser vista como um operador diádico de *mistura linear* de modo que, variando α de $0 \rightarrow 1$, este operador realiza uma *dissolução cruzada temporal* entre as duas imagens. Este efeito é muito visto em apresentações de *slides* e em produções de filmes, ou como um componente de algoritmos de *morphing* de imagens.

Ao calcular a função custo, é preciso decidir sobre qual função de distância usar. Foram implementadas duas delas: distância do tipo quarteirão (*city block* ou *Manhattan*) e distância Euclidiana. Aqui está a fórmula da distância de *Manhattan*:

$$d_{\text{RGB}} = |v_R^M - v_R^I| + |v_G^M - v_G^I| + |v_B^M - v_B^I| \quad (3.4)$$

e esta é a fórmula da distância Euclidiana:

$$\begin{aligned}
 d_{\text{RGB}} &= \|v_{\text{RGB}}^M - v_{\text{RGB}}^I\|_2 \\
 &= \left\| \begin{pmatrix} v_R^M - v_R^I \\ v_G^M - v_G^I \\ v_B^M - v_B^I \end{pmatrix} \right\|_2 \\
 &= \sqrt{(v_R^M - v_R^I)^2 + (v_G^M - v_G^I)^2 + (v_B^M - v_B^I)^2},
 \end{aligned} \tag{3.5}$$

nas quais os supra-índices M e I indicam se o vértice pertence ao grafo modelo ou de entrada, respectivamente, e os sub-índices R , G e B indicam o canal de cor sendo considerado. As mesmas fórmulas são aplicadas aos valores (x, y, z) de profundidade. Funções que decompõem as componentes estruturais em termos de módulo e direção também podem ser exploradas.

Outra opção de definição da função custo é:

$$d = \alpha \cdot d_{\text{RGB}} + \beta \cdot d_{\text{XY}} + \gamma \cdot d_Z. \tag{3.6}$$

Nesta, há uma distinção entre a distância d_{XY} entre as texturas e a profundidade d_Z de cada imagem RGBD. Os parâmetros α , β e γ servem como pesos que intensificam ou diminuem a influência de uma das distâncias no resultado. Apesar de, neste caso, não se fazer necessária a manutenção da soma dos três pesos como uma unidade, este procedimento se faz útil à medida que facilita a percepção da influência percentual de cada uma das distâncias.

Embora muito usado, o espaço de cores RGB não possui muitas propriedades convenientes, de modo que, calcular a distância entre cores usando este espaço não é a melhor forma de se medir a similaridade entre duas cores. Na verdade, o RGB não é um espaço de cores perceptualmente uniforme. Isto significa que duas cores a uma dada distância podem parecer muito similares enquanto duas outras cores separadas pela mesma distância podem parecer muito diferentes [Laganière, 2011].

Muitos trabalhos já foram publicados sobre pesquisa realizada acerca da teoria das cores, a natureza do sistema visual humano e as medidas de distância usadas nesses espaços de cores [Fairchild, 2005; Glassner, 1995; Healey & Shafer, 1992; Livingstone, 2008; Reinhard *et al.*, 2005; Wyszecki & Stiles, 2000]. A página de Fairchild [2012] sobre livros e outras páginas que tratam sobre cores lista diversas outras fontes. Um espaço chamado XYZ, inclui a habilidade de separar a luminância da cromaticidade. Um outro espaço de cores, desenvolvido pela *Commission Internationale d'Eclairage* (CIE) quando em busca de um espaço de cor que representasse melhor a percepção humana, é o $L^*a^*b^*$, também chamado de CIELAB ou CIE $L^*a^*b^*$. Trata-se de um remapeamento não-linear do espaço XYZ, feito de modo que diferenças na luminância ou na cromaticidade são uniformemente percebidas. Por meio da conversão das imagens para este espaço, a distância euclidiana entre dois pixels passa a ser uma medida da similaridade visual entre as duas cores. Ainda que usando o espaço RGB, a adoção das distâncias mencionadas não prejudica os resultados e conta com a vantagem da simplicidade conceitual.

3.7 Visualização dos resultados

CONFORME PODE SER PERCEBIDO, a metodologia desenvolvida envolve o uso de diversos parâmetros. Para alguns destes, a avaliação de qual valor é melhor em determinada situação é facilmente identificada, para outros nem tanto. Além disso, apesar de o interesse principal serem os resultados finais, também é importante visualizar os dados em passos intermediários para se ter uma ideia da interação entre parâmetros diferentes. Com isso em mente, desenvolvemos uma interface gráfica que permite a visualização simultânea dos dados em diferentes passos da sequência de etapas. Parte das diversas opções são descritas a seguir.

3.7.1 Blocos

Para indicar a construção dos blocos, o método dá ao usuário três opções:

Figura 3.8: *Opções de exibição dos blocos. Somente os blocos considerados para a construção do grafo foram desenhados na figura. As opções do desenho da grade e do cálculo dos blocos médios também foram ativadas.*

- desenhar os vértices representantes de cada bloco;
- desenhar uma grade que mostra a interface entre os blocos;
- desenhar cada bloco espalhando a cor do pixel que o representa por toda a área retangular ocupada.

3.7.2 Grafos

Uma vez que cada vértice do grafo representa um bloco em um par processado de imagens, nós adotamos a estratégia de desenhar um grafo simplesmente desenhando esses blocos representados. Dessa forma, um grafo pode ser mostrado seguindo os mesmos três tipos de desenho de blocos, mas enfatizando quais blocos foram considerados para sua construção após a aplicação da eliminação de fundo pelo processamento da profundidade. Refira-se à Figura 3.8 para um exemplo dessas três opções de exibição ativas.

3.7.3 Casamentos entre grafos

O método desenvolvido permite a visualização de casamentos entre grafos de três formas:

- setas;
- blocos coloridos;
- blocos em níveis de cinza.

O primeiro tipo desenha, para cada casamento, uma seta que vai da localização de um vértice no grafo modelo até a posição do vértice casado no grafo de entrada. Uma consequência dessa abordagem é que nós temos, nessa visualização, uma seta iniciando em cada vértice do grafo modelo, mas nós podemos ter, no grafo de entrada, vértices que são pontos de chegada de mais de uma seta e, portanto, outros que não são o destino de nenhuma seta. Isto acontece porque esta relação entre os dois grafos é um homomorfismo, não um isomorfismo.

Se, em vez do desenho de setas o usuário escolhe a opção de desenhar os casamentos como blocos coloridos, então cada direção é representada por um rótulo colorido e o programa pinta toda a área de cada bloco com o rótulo correspondente. Neste tipo de desenho, o usuário ainda pode escolher entre 8 ou 4 direções. A mesma representação é adotada em blocos em níveis de cinza. Para facilitar o uso desse esquema, a interface gráfica inclui um rótulo, no qual as cores indicam as respectivas direções, como pode ser visto na Figura 3.9.

Figura 3.9: Rótulos com 8 ou 4 cores e em níveis de cinza que indicam a direção do movimento. A cor preta, ao centro, indica ausência de movimento.

Cálculo das direções

A fim de calcular a direção para a qual cada bloco está se movendo, nós calculamos o arco-cosseno do produto interno normalizado entre o vetor que representa o deslocamento do bloco e o vetor unitário (1, 0), que aponta para a direção de crescimento do eixo x . Na sequência, nós descobrimos a qual quadrante o vetor pertence levando em conta o sinal de sua coordenada y .

Filtro da mediana

A visualização dos resultados finais mostra que há muitos blocos que corretamente identificam a direção do movimento, mas ainda outros que representam maus casamentos. Com o objetivo de eliminar estes últimos e tratar esta falta de continuidade espacial, o filtro da mediana é aplicado à imagem dos blocos rotulados. A altura da imagem pode ser calculada por meio da divisão da altura da imagem original (entrada) pela altura do bloco; o mesmo procedimento se aplica à largura. O filtro da mediana é parametrizado pelo tamanho do núcleo, o qual deve ser ímpar.

3.8 Desempenho

O MÉTODO DESCRITO foi implementado em um programa escrito em linguagem C++, usando as bibliotecas OpenCV, OpenNI, Boost Graph Library e Qt. Maiores detalhes são dados no Apêndice A. O sistema implementado roda em tempo real quando os parâmetros envolvidos na construção do grafo implicam em poucos vértices, ou seja, quando os blocos possuem dimensões relativamente grandes (acima de 10×10 pixels) e quando os valores de próximo e distante que determinam o volume de visualização implicam em uma considerável eliminação de blocos. Conforme mais dados estão presentes no interior do tronco de pirâmide, mais vértices são criados no grafo. Em casos em que o valor dos parâmetros torna o método lento (e. g., com a área dos blocos abaixo de 100 pixels), faz-se necessária uma outra estratégia para ler a entrada. Nossa abordagem é capturar todas as imagens de entrada (textura e profundidade) de uma sequência de uma só vez, gravá-las como arquivos de imagem em disco rígido e rodar o programa em modo de lote, pós-processando as imagens.

O Kinect™ não é capaz de capturar as imagens de textura e de profundidade (ou nenhum dos outros tipos possíveis de dados de entrada) ao mesmo tempo. Assim, se há algum tipo de processamento de dados entre duas capturas, uma diferença perceptível nos objetos móveis aparece, indicando que os dados não casam entre si. Faz-se, então, um requisito importante capturar todas as imagens necessárias referentes a um mesmo quadro e só então processar os dados. O atraso inerente que existe entre duas capturas consecutivas, sem qualquer processamento no meio, é aceitável para nossos propósitos.

Ao usar o Kinect™ como dispositivo de captura, além das imagens de textura e de profundidade, também é possível capturar o mapa de disparidade, a máscara de profundidade válida e a textura em níveis de cinza (com apenas um canal). Como nós não usamos esses dados em nossos algoritmos e para evitar qualquer atraso adicional, eles não são capturados.

A Figura 3.10 contém uma captura de tela da janela principal do programa desenvolvido. Nela podemos notar que foram implementadas diversas facilidades que uma interface gráfica pode oferecer, tais como menus, barras de botões e barra de estado.

Cada algoritmo implementado seguiu a arquitetura MVC (do inglês *Model-View-Controller*), e segue os padrões de desenvolvimento *strategy*, *controller* e *singleton*. Assim, cada um deles possui uma janela

Figura 3.10: *Janela principal do aplicativo.*

Figura 3.11: Caixa de diálogo do desenho do grafo e do casamento.

de diálogo que faz a interface (serve de visão) com o controlador. Um exemplo é mostrado na Figura 3.11. Trata-se da caixa de diálogo que fornece as opções de desenho do grafo e do casamento ao usuário.

Capítulo 4

Resultados experimentais

ESTE CAPÍTULO ILUSTRA a aplicação do método proposto usando sequências reais. Dados de entrada e saída típicos são mostrados na Figura 4.1, seguidos por versões maiores e comentadas nas figuras seguintes.

A Figura 4.2 mostra a imagem de profundidade capturada de duas pessoas caminhando em direções opostas, com oclusão ocorrendo entre elas e também entre suas respectivas pernas; a imagem da esquerda mostra a cena antes da execução do procedimento de eliminação de fundo enquanto a da direita a mostra após o processamento. A Figura 4.3, à esquerda, mostra a textura para a mesma cena. Note que a profundidade já está registrada com a textura; à direita é exibido o movimento detectado representado por rótulos coloridos. Como se pode ver, o método leva em conta os efeitos de oclusão. Esse experimento mostra a identificação de movimento presente na cena. A pessoa mais à esquerda está mais distante do dispositivo de captura, como indicado pelo nível de cinza no mapa de profundidade; ela está caminhando da esquerda para a direita. A outra pessoa, mais próxima ao dispositivo de captura, executa um movimento da direita para a esquerda. Note a correta classificação de ambos os movimentos, até mesmo na região onde elas se interceptam. Os pixels verdes que aparecem na Figura 4.3 foram identificados como se movendo para cima, um resultado razoável, exceto para a mancha verde que aparece no canto inferior esquerdo: ela aparece devido a um erro na captura da profundidade. Esse mesmo experimento também é um exemplo de como nosso método obtém resultados satisfatórios mesmo na presença de oclusão de objetos móveis. Note como uma perna está parcialmente ocluída por outra e ainda assim seu movimento é corretamente identificado.

Para a entrada mostrada na Figura 4.4 obtivemos a saída exibida na Figura 4.5. Esse experimento mostra o apelo visual que a representação dos casamentos como setas traz. Em vez de rótulos coloridos, setas com pontas em formato de círculos são usadas para indicar o movimento. Esse exemplo foi capturado em tempo real.

O experimento apresentado na Figura 4.6 exemplifica o resultado refinado que obtemos ao trabalhar com blocos pequenos. É exibida a saída quando o método é aplicado ao mesmo vídeo mostrado na Figura 1.3, mas com os blocos de tamanho 2×2 pixels, em vez de 10×10 . O tempo de processamento aumenta substancialmente com tal modificação. Nesses casos é necessário rodar o método em modo de lote. Essa

Figura 4.1: Dados de entrada e saída típicos. Os dados da imagem de profundidade mostrados já foram processados pelo procedimento de eliminação de fundo.

Figura 4.2: Imagem de profundidade capturada de duas pessoas caminhando em direções opostas, antes e após processamento para eliminação de fundo. Esta imagem também exemplifica um caso de oclusão entre objetos móveis.

Figura 4.3: Imagem de textura e movimento detectado, representado por rótulos coloridos, para a mesma cena mostrada na Figura 4.2. Note que a imagem de profundidade já está registrada com a textura.

Figura 4.4: Sequência de vídeo com pessoa movendo seu braço.

Figura 4.5: Visualização dos casamentos como setas, indicando que o braço está se movendo para cima e para a direita.

Figura 4.6: Resultado refinado usando blocos de 2×2 pixels.

Figura 4.7: *Teste de estimação de movimento considerando os valores extremos de α (0 e 1) para blocos de tamanho 10×13 ((b) e (e)) e 5×7 ((c) e (f)). Em (b) e (c) o valor de α é 0; em (e) e (f) o valor de α é 1.*

figura também mostra um efeito colateral do filtro da mediana da OpenCV: uma vez que a filtragem é feita considerando cada canal separadamente, a imagem de saída pode eventualmente ser composta por cores que não estavam presentes na imagem de entrada.

Com todas as características descritas, o sistema desenvolvido também tem se mostrado um arcabouço conveniente para lidar com dados de entrada gerados pelos novos dispositivos sensores de profundidade. A aplicação é capaz de realizar processamento de imagens e executar algoritmos de Visão Computacional, permitindo portanto uma fácil avaliação de resultados a taxas em tempo real.

Na Figura 4.7 é exibido um teste para estudar a influência causada no casamento entre os grafos conforme tomamos $\alpha = 0$ (considerando apenas a geometria) e $\alpha = 1$ (considerando apenas a aparência). Na cena gravada, um bebê dá seus primeiros passos da direita para a esquerda. A primeira coluna da figura mostra a textura e o mapa de profundidade já tendo sofrido eliminação do fundo. Nas outras quatro imagens, com o casamento de cada bloco sendo representado por setas, a linha de cima é o resultado para $\alpha = 0$ e a linha de baixo para $\alpha = 1$. Na coluna do meio, os blocos possuem dimensões 10×13 pixels enquanto que na coluna da direita o tamanho é 5×7 . Note que, com $\alpha = 0$, ocorreu uma tendência de muitos blocos serem casados consigo mesmos. Isso deve-se ao fato de que o deslocamento realizado pelo bebê, de um quadro a outro, foi pequeno, sendo menor que o tamanho dos blocos. Assim, considerando apenas os valores de profundidade, cada bloco acaba sendo casado com ele mesmo, já que ocorre apenas uma pequena mudança em z . Vale a pena notar também que as únicas exceções, ou seja, os únicos blocos que não ficam parados segundo o casamento, são os que ficam nas bordas verticais do mapa de profundidade considerado. Cada um desses blocos acaba sendo casado com o bloco mais próximo. Já com $\alpha = 1$, podemos facilmente perceber que o casamento que leva em conta apenas os valores RGB definitivamente não coincide com o movimento realizado na cena. Também é notável que a diferença entre valores RGB não é uma boa medida de similaridade, já que temos casamento entre blocos com cor de pele e blocos com a cor da camiseta (violeta).

A Figura 4.8 mostra as texturas e os mapas de profundidade que serviram de entrada para que o método desenvolvido gerasse todas as saídas reunidas na Figura 4.9. Nessa sequência, um casal realiza um passo de samba de gafeira no qual o cavalheiro traz a dama em sua direção. O movimento ocorre, portanto, em sua maior parte, da esquerda para a direita, excetuando-se principalmente as mãos dadas e os respectivos braços,

Figura 4.8: Texturas e mapas de profundidade que serviram de entrada para que fossem geradas as imagens mostradas na Figura 4.9.

que sobem um pouco, e a perna de trás da dançarina, que desce um pouco.

A Figura 4.9 mostra uma comparação entre casamentos nos quais se varia os valores do peso α e os tamanhos dos blocos. São exibidos casamentos para $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$, cuja variação ocorre nas colunas da figura. Os testes para os tamanhos dos blocos incluem as dimensões 20×27 , 10×13 , 6×8 , 5×7 , 4×6 , 3×3 , 2×2 e 1×1 . Nas três primeiras linhas de cima para baixo, a representação dos casamentos foi feita por setas; uma vez que para blocos de tamanho 6×8 já torna-se difícil perceber para onde cada seta aponta, nas linhas restantes optou-se pela representação por cores. Nessa figura ficam claras as duas tendências discutidas na figura anterior: quando $\alpha = 0$, grande parte dos blocos são casados consigo mesmos (a cor preta indica “bloco parado”) e quando $\alpha = 1$ ocorrem casamentos que não condizem com o movimento observado na cena. Curiosamente, a maior parte desses casamentos ocorre de baixo para cima, indicando um movimento de subida. Outro fato curioso são os casamentos indicando movimento para a esquerda (cor azul) que só foram notados quando a dimensão dos blocos foi baixada para 1×1 pixel. Visualmente, percebe-se que o melhor casamento parece ocorrer para blocos de 4×6 pixels com $\alpha = 0.25$.

A fim de demonstrar a robustez dos algoritmos implementados, mostramos na Figura 4.10 uma sequência de quadros não-consecutivos de um mesmo vídeo, no qual novamente duas pessoas caminham em direções opostas. Os números dos quadros nesse vídeo são 214, 221 e 230, havendo portanto uma diferença de 7 e 9 quadros, respectivamente. O mapa de profundidade exibido não sofreu nenhum processamento, porém, para a geração dos resultados usou-se o valor 90 para o plano próximo e 217 para o plano distante, o suficiente para remover a parede do fundo e os ventiladores. O tamanho dos blocos testados foram 10×10 , 5×5 , 3×3 e 1×1 . Adotamos $\alpha = 0.3$, o que dá 30% de peso para os valores da textura e 70% de peso para os valores da geometria. A robustez do algoritmo é verificada pela constatação de que a cor que indica a direção do movimento de cada pessoa (vermelho para a direita e azul para a esquerda) se mantém como rótulo majoritário no decorrer do tempo, sem grandes variações. Pode-se notar também que, devido ao fato de o chão ser de madeira encerada, o movimento dos reflexos no chão também é corretamente classificado.

Figura 4.9: Comparação entre casamentos variando os valores do peso α (nas colunas) e os tamanhos dos blocos (nas linhas).

Figura 4.10: *Teste de robustez da classificação no decorrer do tempo.*

Capítulo 5

Conclusão

5.1 Discussão

A METODOLOGIA APRESENTADA nesta tese descreveu todas as etapas envolvidas em uma nova proposta de uso de modelos de grafos com fusão de atributos advindos de vídeo 3D (textura e geometria) para obter a estimação de movimento a partir de imagens RGBD usando homomorfismo entre grafos, incluindo as etapas de aquisição, pré-processamento das imagens de entrada, construção dos grafos e dos vetores de características representativos de cada um de seus vértices, o algoritmo de casamento entre os grafos e as diversas formas de visualização dos resultados finais e intermediários. Como parte do trabalho, foi desenvolvido um *software* que implementa toda a pesquisa realizada.

O principal dispositivo de captura usado foi o Kinect™. Uma vez superadas as dificuldades inerentes ao lançamento de um novo dispositivo no mercado, tais como instalação de *drivers* e curva de aprendizado de uma nova API, o Kinect™ demonstrou ser um sensor de grande valia para a área de Visão Computacional, tendo sido explorado em projetos das mais variadas aplicações. Nossa experiência com esse dispositivo demonstrou as vantagens em se fazer uso do mapa de profundidade gerado em algoritmos que antes só eram usados em imagens de textura.

Algumas dificuldades que surgiram no uso do Kinect™ foram:

Sensibilidade a raios infra-vermelho de outras fontes: o mapa de profundidade gerado pelo Kinect™ depende da reflexão da luz infra-vermelha emitida por ele, de modo que há muita interferência sobre as imagens quando há na cena a presença de raios infra-vermelho emitidos por outras fontes. Isso limita a diversidade de cenas que podem ser capturadas a ambientes noturnos ou fechados. A gravação de uma cena a céu aberto durante o dia sofre muita influência da luz infra-vermelha presente na luz solar, como pode ser verificado na Figura 5.1.

Falhas de captura na imagem de textura: Conforme uma frequência média de 1 quadro a cada 60, ocorrem falhas de captura na imagem de textura. Essas falhas são caracterizadas por translações do quadro em conjunto com a exibição de uma composição de parte da textura com uma matriz de pontos, o que parece revelar um pouco sobre o funcionamento do Kinect™ por meio de luz estruturada. Um exemplo dessas imagens pode ser visto na Figura 5.2.

5.2 Trabalhos futuros

HÁ VÁRIAS CARACTERÍSTICAS em nosso método que são passíveis de melhoria, seja com relação aos resultados ou ao desempenho:

Como mencionado anteriormente, muitos procedimentos que são aplicados nas imagens de textura são independentes dos procedimentos aplicados nas imagens de profundidade. Assim, tais tarefas podem ser

Figura 5.1: Desempenho do Kinect™ na presença de luz solar. É notável a influência da luz infravermelha proveniente do sol na geração do mapa de profundidade.

Figura 5.2: Falha de captura na imagem de textura. A imagem da esquerda representa um quadro anterior na sequência de gravação em relação ao quadro da direita, no qual ocorre a falha.

paralelizadas, de modo a rodarem ao mesmo tempo em diferentes processos, fazendo uso dos múltiplos núcleos de um mesmo processador, por exemplo.

Mais interessante ainda, podemos modelar o casamento entre grafos, que é o gargalo do sistema implementado no método proposto, como um programa paralelo composto por tarefas independentes e que exigem pouca comunicação entre elas. Uma vez que o método não exige uma grande quantidade de memória, acreditamos que uma implementação usando GPUs possa melhorar consideravelmente o desempenho do casamento entre grafos.

Um processamento importante que é aplicado sobre o mapa de profundidade é a remoção do fundo para extração dos objetos de interesse. Um trabalho que vem avançando nesse sentido com imagens capturadas com o Kinect™ é o de Prada *et al.* [2012], intitulado “*Object extraction in RGBD images*”. Já foi demonstrado interesse na integração entre os dois códigos, o que deve ocorrer em breve.

No casamento entre os grafos, blocos que se localizam no interior de objetos móveis que são muito finos têm a tendência de não serem mostrados na visualização dos resultados, ou seja, o casamento indica que tais blocos permaneceram parados de um quadro a outro. Isto acontece porque a textura e a profundidade em tais locais não são muito alteradas. O filtro da mediana minimiza este efeito, mas não é suficiente para extingui-lo. A fim de lidar com esse problema, podemos adotar uma abordagem temporal. Nessa abordagem, cada vértice passa a possuir um marcador de tempo armazenado em seus campos, indicando o número do quadro ao qual aquele vértice pertence. Se o bloco permanecer no mesmo lugar de um quadro a outro, nós podemos usar esses dados para continuar rotulando-o com o mesmo rótulo colorido que foi atribuído a ele, até que um número pré-definido de quadros seja processado.

A direção do movimento de um bloco é calculada no plano que é determinado por suas coordenadas de textura. Fazendo uso do valor z dado pelo mapa de profundidade, podemos determinar o movimento levando em conta direções no espaço 3D. Salientamos que o método permite e já realiza o cálculo do movimento na direção z . No entanto, ainda não foi desenvolvido nenhum tipo de visualização tridimensional, sendo uma das atividades previstas para serem concluídas em um futuro próximo. Os novos tipos de visualização a serem implementados envolveriam o uso do VTK (*Visualization Toolkit*) e passariam a incluir novas formas de se visualizar a estimação de movimento 3D.

Também é possível aplicar esse método para detectar partes rígidas de um objeto articulado. O desenvolvimento de um método para isso, com base na abordagem e código desenvolvidos na presente tese, consiste em uma linha de trabalho de bom potencial.

Uma vez que os sensores da câmera e de luz infra-vermelha estão distanciados de alguns centímetros, naturalmente surgem áreas de sombra na geometria 3D dos objetos gravados. Pode-se explorar a estimação de movimento descrita neste trabalho para fazer o casamento entre diversas nuvens de pontos capturadas a partir de diferentes pontos da cena, de modo a preencher as áreas de sombra por meio do registro de novos dados que surgem com o deslocamento. A fim de otimizar tal registro, é possível limitar o número de pontos rígidos a serem casados, segundo uma pré-seleção realizada no primeiro quadro de um vídeo, como por exemplo os pontos indicados pelo algoritmo “*good features to track*”.

Com relação à função custo, além de tentar outras abordagens que já vem sendo sugeridas e discutidas, também se pode tentar uma otimização da mesma via algoritmos meta-heurísticos.

Por fim, temos ciência de que a avaliação dos resultados foi feita apenas visualmente, envolvendo julgamentos que, apesar de intuitivamente estarem corretos, não deixam de ser subjetivos. Para que seja aplicada uma métrica de erro, pretendemos desenvolver um *ground-truth* para os dados avaliados e os avanços obtidos serão publicados no devido tempo.

Apêndice A

Detalhes técnicos da implementação

A.1 Porte do vídeo 3D para o sistema operacional GNU/Linux

UMA DAS PRINCIPAIS MOTIVAÇÕES no andamento deste trabalho tem sido a reconstrução tridimensional dos objetos presentes em uma cena de vídeo 3D por meio do casamento de diversas imagens $2\frac{1}{2}$ D. Para tanto, a detecção, o rastreamento e o registro dos componentes conexos presentes em um vídeo 3D são passos intermediários importantes.

O projeto envolveu uma etapa intensa de programação. O sistema de vídeo 3D inicial foi desenvolvido no IMPA em plataforma Windows, usando a IDE Visual Studio C++. Durante o mestrado foi desenvolvido código nas linguagens C, C++ e Matlab, além de *scripts* em Bash. Assim, o primeiro passo a ser dado foi a integração de todo o código do sistema de vídeo 3D em uma única linguagem (C++) e, inicialmente, executável apenas para a plataforma GNU/Linux.

A implementação foi realizada na linguagem C++, usando-se a IDE KDevelop e o sistema de construção CMake. A visualização é feita por meio do uso da biblioteca OpenCV. Já a interface gráfica foi desenvolvida com a biblioteca QT, além de experimentos iniciais com o OpenGL e a GLUT.

Aproveitando essa reimplementação, foi alterado também o paradigma de programação: de estruturada (procedural) para orientada a objetos. Essa alteração exigiu um grande esforço no sentido de identificar corretamente as classes relevantes, bem como os seus relacionamentos. Essa modelagem do sistema orientada a objetos permitiu uma melhor compreensão do mesmo e, graças à sua modularização, facilitou seu desenvolvimento, ainda que tenha vindo a se constituir um projeto complexo. Tal planejamento continua sendo acompanhado de diagramas UML, o que serve como uma ótima documentação do projeto.

O código de reconstrução de objetos articulados e captura de movimento sem marcadores tem sido testado tanto em sequências de vídeo 3D adquiridas a partir de cenas reais pelo sistema de aquisição do IMPA quanto em sequências adquiridas com o Kinect™.

A.2 Modelagem do sistema de vídeo 4D

O NOVO SISTEMA IMPLEMENTADO foi desenvolvido com um código completamente novo, segundo uma modelagem orientada a objetos. A fim de fornecer uma documentação sobre essa modelagem, as subseções seguintes apresentam diversos diagramas UML (do inglês *Unified Modeling Language*, Linguagem de Modelagem Unificada), os quais mostram-se extremamente úteis para o entendimento das classes envolvidas no sistema e dos relacionamentos existentes entre elas.

A.2.1 Diagramas de casos de uso

A Figura A.1 mostra uma grande quantidade de casos de uso envolvendo o projeto Vídeo 4D, dos quais participam diversos atores. Nessa figura, os casos de uso são vistos a partir de uma perspectiva de alto nível,

Figura A.1: Casos de uso em alto nível: Diversos atores participam de casos de uso tratados a partir de uma perspectiva de alto nível.

sendo, alguns deles, mais bem detalhados em outras figuras adiante. A seguir, damos uma breve explicação de cada um.

Visualização de textura e geometria: Uma interface gráfica, desenvolvida com o uso de bibliotecas como o GTK ou o Qt, que contém um *widget* dedicado à apresentação da textura e da geometria de um vídeo 4D por meio do VTK (biblioteca de visualização), do OpenGL ou da OpenCV.

Interação com um vídeo 4D: Além de visualizar, um usuário do sistema também deverá ser capaz de interagir com o vídeo, mudando, por exemplo, o ponto de vista a partir do qual a cena é assistida. Isso será feito por meio de controles como aproximação, rotação e translação.

Edição de um vídeo 4D: Envolve alterações aplicadas à textura e à geometria do vídeo.

Salvamento e abertura de um vídeo 4D: Um novo formato de arquivo deverá ser especificado, já que há diversas características que estarão presentes em um vídeo 4D e que não têm lugar no formato do vídeo $3\frac{1}{2}$ D.

Exportação de um vídeo 4D: A fim de realizar troca de dados com outros aplicativos, o sistema deverá ser capaz de exportar tais dados para diversos formatos.

Importação de textura e geometria: Parte da edição de um vídeo 4D envolve alterações na textura e na geometria. Sendo assim, será útil o desenvolvimento de métodos de importação de tais dados, permitindo o uso de ferramentas externas ao sistema.

Conversão de vídeo $3\frac{1}{2}$ D para vídeo 4D: Um conversor poderá preencher os atributos apropriados de um vídeo 4D a partir de um processamento feito sobre os dados originais capturados em um vídeo $3\frac{1}{2}$ D.

Leitura de um vídeo $3\frac{1}{2}$ D: Todo o processamento do sistema inicia-se com a leitura de um arquivo no formato do vídeo $3\frac{1}{2}$ D.

Registro de dois quadros de um vídeo 4D: A reconstrução da geometria e da textura de objetos presentes na cena gravada é feita por meio de registros de quadros consecutivos do vídeo, tomados dois a dois.

Rastreamento de componentes conexas: O rastreamento de componentes conexas em um vídeo 4D é necessário para que sejam isoladas as respectivas nuvens de pontos que servem de entrada para o algoritmo de registro de duas geometrias.

A Figura A.2 mostra que a edição de um vídeo 4D envolve a manipulação de suas texturas e de suas geometrias. Uma textura pode ser adicionada ou removida de um vídeo 4D, além de poder ser editada, permitindo assim sua atualização. Tanto para remover quanto para atualizar uma textura, deve-se fazer uso do caso de uso “Encontrar textura”. Descrição semelhante aplica-se aos casos de uso que envolvem a edição da geometria de um vídeo 4D. Adicionalmente, no caso de uso “Atualizar geometria”, há duas extensões, referentes à atribuição ou desassociação de uma textura a uma determinada geometria. Os casos de uso “Encontrar textura” e “Encontrar geometria” exigirão o armazenamento de dados em um sistema persistente, seja em um banco de dados ou em um sistema de arquivos. A possibilidade de alteração de qualquer um desses elementos permite aplicações como a troca da textura da face de uma pessoa ou o exagero de características da geometria, como aumentar o tamanho de um nariz, por exemplo.

No caso de uso “Salvamento e abertura de um vídeo 4D”, mostrado na Figura A.3, vê-se que o sistema permitirá o uso de dois sistemas persistentes distintos: um banco de dados ou o próprio sistema de arquivos. Para um melhor uso daquele, deverão ser implementados métodos que permitam a consulta dos dados armazenados.

Conforme mostra a Figura A.4, a exportação de um vídeo 4D envolve a exportação separada de seus dois principais componentes: a textura e a geometria. Tal exportação permite que os dados de um vídeo 4D possam ser usados por outros aplicativos, facilitando, por exemplo, a confecção de demonstrações do sistema. O ator que participa desse caso de uso é um sistema persistente. A exportação dos dados pode

Figura A.2: Edição de um vídeo 4D: O ator **EDITOR** participa do caso de uso “Editar um vídeo 4D”.

Figura A.3: Salvamento e abertura de um vídeo 4D: Para a execução deste caso de uso faz-se necessária a definição de um novo formato de arquivo.

Figura A.4: Exportação de um vídeo 4D: O ator SISTEMA PERSISTENTE participa do caso de uso “Exportação de um vídeo 4D”.

ocorrer para diversos formatos de arquivos. No caso da geometria, pretende-se implementar exportadores para o formato PLY, amplamente usado na área de computação gráfica, e para o formato VTCDATA, que será necessário para o desenvolvimento do sistema de visualização usando-se a biblioteca VTK. A exportação para outros formatos deve ser implementada conforme se fizer necessário. Como é de nosso interesse fazer a manutenção de diversas geometrias para um mesmo vídeo 4D, a cada exportação será preciso identificar qual das geometrias armazenadas deve ser exportada. Isso é evidenciado no diagrama pelo caso de uso “Encontra geometria”. A exportação de uma textura de um vídeo 4D ocorre de modo similar. Também aqui surgem diversas opções de formatos, tais como PNG, BMP e JPEG, no caso da exportação de quadros isolados do vídeo. Quando da exportação da textura de um intervalo de quadros consecutivos do vídeo, pode-se realizar a exportação para MPEG e AVI, por exemplo. Nesses casos, diversos tipos de *codecs* podem ser aplicados ao vídeo, a fim de compactá-lo, conforme exemplificado pelo caso de uso “Aplicação de um *codec*”.

A Figura A.5 mostra o caso de uso “Importação de textura e geometria”. Nele, vê-se que a textura e a geometria podem ser importados a partir de diversos formatos de arquivos. No entanto, o único formato que será aceito para a importação de ambos os elementos ao mesmo tempo será o do vídeo $3\frac{1}{2}$ D. Esse caso de uso assemelha-se muito com o de “Exportação de um vídeo 4D”, visto na Figura A.4.

A.2.2 Diagrama de implantação

O diagrama de implantação mostrado na Figura A.6 fornece uma visão geral do sistema de vídeo 4D. Após a captura dos dados usando uma câmera e um projetor, há um processamento na máquina que está diretamente ligada a esse equipamento. Na sequência, a reconstrução do vídeo 3D é feita em uma outra máquina, que é a que aplica o algoritmo BCSL de identificação de faixas para determinação da profundidade de alguns pontos da cena gravada. Quando o sistema de vídeo 4D estiver completo, o fluxo de dados continuará em uma máquina que conterà o executável de importação a partir de um arquivo de vídeo $3\frac{1}{2}$ D, a qual realizará a conversão de vídeo por meio de algoritmos de rastreamento e registro.

A.2.3 Diagrama de classes

A Figura A.7 mostra o diagrama de classes desenvolvido até agora. Segundo ele, um objeto da classe `v4dVideo4D` possui como atributos três objetos das seguintes classes:

v4dV35DFileReader: Após ter o nome do arquivo a ser lido inicializado pelo método `SetFilename`, esse objeto devolve um objeto da classe `v4dVideo4D` quando recebe a mensagem `GetOutput`.

v4dInfo: Contém todas as informações necessárias para que se faça a leitura de cada quadro do vídeo, tais como a resolução da textura, o número de pontos na geometria, as matrizes de calibração entre câmera e projetor, etc.

v4dFileHeader: Classe simples que serve apenas para validar o número mágico do tipo de arquivo e a versão da API que lida com ele.

A.2.4 Diagrama de colaboração

Na Figura A.8, vê-se que um objeto da classe `v4dV35DFileReader` colabora com um usuário do sistema, chamado de Administrador, o qual envia, por meio de uma interface, mensagens para criação, acerto de nome e pedido de saída de dados. Outra colaboração é feita com um objeto da classe `v4dVideo4D`, já que os seus atributos são preenchidos pelo leitor de vídeo $3\frac{1}{2}$ D. Este, por sua vez, realiza uma colaboração consigo mesmo, de modo a obter os dados lidos a partir do arquivo.

A.2.5 Diagramas de estados

Na Figura A.9 está representado um diagrama de estados do sistema de vídeo 4D quando este já possuir uma interface gráfica que fará a integração de todo o sistema. De início, após já ter sido determinado o

Figura A.5: Importação de textura e geometria: Podem ser importadas textura e geometria a partir de diversos tipos de fontes.

Figura A.6: Diagrama de implantação: Fluxo dos dados, desde a captura até o processamento.

nome do arquivo de vídeo 3½D e ter sido lido o seu cabeçalho, inicia-se a leitura dos quadros do vídeo. Depois da leitura de cada quadro, é executado um algoritmo de registro entre a geometria e a textura. Com o registro do novo quadro lido, eventualmente novos dados passarão a fazer parte do modelo, o que exigirá uma atualização dele. Dependendo da interação que o usuário terá com a interface gráfica e do deslocamento dos objetos na cena no momento de gravação do vídeo, um novo ponto de vista da cena deverá ser calculado, o que exigirá uma renovação do quadro que está sendo mostrado pelo módulo de visualização.

A Figura A.10 representa um diagrama de estados de um objeto da classe `v4dV35DFileReader`. De início, após ser criado, esse objeto está em um estado inativo, nada pode ser feito com ele até que lhe seja indicado o nome do arquivo de vídeo 3½D que será lido. Depois de esse nome ser determinado, o objeto encontra-se num estado de prontidão, durante o qual ele pode receber a mensagem `GetOutput`, que iniciará uma verificação sobre a consistência do arquivo. O arquivo somente será validado se realmente existir e puder ser aberto para leitura. Além disso, ele deve conter valores corretos para os campos `magicNumber` e `version`. Se o arquivo não é validado, o objeto lança uma exceção e volta para o estado de inatividade. No caso de ser validado, é criado um objeto da classe `v4dVideo4D` e inicia-se a leitura dos dados. Com isso, o objeto entra num estado que engloba outros estados, chamado de “trabalhando”. Enquanto ele estiver neste estado, haverá um ciclo de leitura de dados e preenchimento de atributos de objetos. Este ciclo se repete até que seja atingido o fim do arquivo, quando então o objeto é destruído.

A.2.6 Diagrama de atividades

A Figura A.11 exibe todas as atividades que são realizadas quando é iniciada a leitura de um vídeo 3½D. Após ser fornecido o nome do arquivo, é criado um objeto da classe `v4dVideo4D` e a leitura do arquivo é iniciada. A princípio são preenchidos os atributos de um objeto da classe `v4dInfo`. Na sequência, para cada quadro do vídeo é feito o preenchimento dos atributos de um objeto da classe `v4dFrame`.

A API `OpenKinect` fornece uma biblioteca dinâmica chamada `libfakenect` a qual, ao ser ligada em conjunto com um programa principal, permite que se leia arquivos gravados em disco após a captura por um dispositivo `Kinect™` por meio de um programa que tenha sido ligado com a biblioteca `libfreenect`. São gravados arquivos de imagem referentes à textura e ao mapa de profundidade de cada quadro, além de diversos arquivos que armazenam dados do acelerômetro. Este últimos permitem que, posteriormente, sem a necessidade de se ter um `Kinect™` conectado ao computador, seja possível simular uma captura, inclusive com perda de quadros devido à uma eventual carga da CPU. Com isso, é possível encontrar bases de dados na internet que contém arquivos que podem ser usados por pessoas que não possuem um `Kinect™`. Além dessa grande utilidade que essa biblioteca fornece, há também uma vantagem prática que vale a pena ser ressaltada: um mesmo programa pode ser ligado com a `libfreenect` ou com a `libfakenect` sem que mudança alguma no código seja feita: a API é a mesma. Essa característica é, sem dúvida, de grande valia para programadores.

Figura A.7: Diagrama de classes do vídeo 4D: Esse diagrama mostra a modelagem orientada a objetos do sistema de vídeo 4D, com suas classes e seus relacionamentos.

Figura A.8: Diagrama de colaboração: São exibidas as colaborações de um objeto da classe v4dV35DFileReader.

Figura A.9: Diagrama de estados I: Os estados referem-se a um objeto de uma classe da interface gráfica que integrará todo o sistema.

Figura A.10: Diagrama de estados II: Estados de um objeto da classe v4dV35DFileReader.

Figura A.11: Diagrama de atividades: Atividades que são executadas quando da leitura de um arquivo de um vídeo 3½D.

A.3 Equipamento e programa da estimação de movimento

O ACESSO AOS DADOS do Kinect™ é feito com o uso da OpenNI™ através de um objeto VideoCapture da OpenCV, cuja versão usada foi a 2.4, além também da biblioteca libfreenect, do projeto OpenKinect. Nosso código está escrito em C++, com a interface gráfica desenvolvida com a API QT. As imagens são capturadas com resolução VGA (640×480) a 30 Hz. Os dados de profundidade possuem uma resolução de 11 bits, representando distâncias com relação à câmera de aproximadamente 40 cm a 6 m. Para construir os grafos, nós usamos a *Boost Graph Library*. O sistema de construção usado foi o CMake. O paradigma de programação adotado foi o orientado a objetos. A versão atual possui 78 classes totalizando 21.547 linhas de código.

Apêndice B

Manual de captura de vídeo 3D

ESTE MANUAL explica como usar o sistema de captura de vídeo 3D desenvolvido sob a coordenação do professor Luiz Velho no IMPA. A primeira seção fornece um início rápido, o qual enfatiza a ordem em que os equipamentos devem ser ligados e os programas executados. Já a Seção B.2 descreve detalhadamente cada fase da captura bem como cada equipamento e programa envolvido na mesma.

B.1 Início rápido

A SEQUÊNCIA DE ETAPAS a ser seguida para se fazer a captura de um vídeo 3D consiste em:

- Ligar a câmera;
- Ligar o projetor;
- Ligar o *genlock*;
- Ligar os monitores;
- *Logar* nos sistemas;
- Executar o programa v4dFaixas no computador conectado ao projetor;
- Projetar quadro branco com as opções *k*, *p* e a barra de espaços;
- Ajustar os objetos na cena a ser gravada;
- Ajustar o enquadramento e a íris da câmera;
- Projetar faixas com as opções *.*, *0*, *p* e a barra de espaços;
- Ajustar o foco do projetor;
- Executar o Microsoft Visual Studio .NET 2003;
- Abrir a solução “VideoCaptura”;
- Executar “Build” → “Build solution”;
- Executar “Debug” → “Start Without Debugging”;
- Ajustar o “Nível Mínimo” de modo que o objeto de interesse seja enfatizado;
- Em “Imagens a Gravar”, determinar a duração da gravação;

- Projetar faixas com as opções $\cdot$ e $\emptyset$;
- Em “Gravação”, clicar em “Grava Vídeo”;
- Abrir a solução “video4d”;
- Construir e executar, abrindo o arquivo “Captura01.cfg”;
- Na aba “Reconstrução”, clicar em “Visualiza 3D”;
- Acertar parâmetro em “Validar vizinhos a uma distância máxima de (cm)”;
- Na aba “Entrada/saída”, clicar em “Gravar vídeo”;

Os arquivos resultantes são Captura01.avi, Captura01.cfg e Captura01.v4d que, atualmente, estão sendo gravados diretamente na raiz do dispositivo D:.

B.2 Detalhamento da captura

O SISTEMA DE CAPTURA de vídeo 3D do IMPA é composto por câmera, projetor, *genlock* e computadores, além dos programas desenvolvidos. A profundidade dos objetos na cena é obtida por meio de luz estruturada, que consiste em um sistema estéreo ativo. Nele, diferentemente de sistemas estéreos comuns, constituídos por duas câmeras, uma delas é substituída por um projetor, o qual projeta faixas na cena. Com isso, a correspondência entre as duas imagens é obtida mais facilmente.

Há dois computadores envolvidos no sistema de captura: um dedicado apenas à geração de faixas e ao envio destas ao projetor e outro dedicado à recepção das imagens capturadas pela câmera e gravadas em disco. Estes computadores permanecem sempre ligados.

A câmera e o projetor são acoplados a um trilho, que por sua vez está preso a um tripé (Figura B.1). O tripé possui rodas e diversos pontos de articulação, os quais podem ser travados ou liberados, fornecendo grande flexibilidade para movimentação do conjunto, ajuste de altura, angulação e distância entre câmera e projetor. Todo o equipamento é semelhante ao de última geração usado em sistemas profissionais de gravação de vídeo e pode ser encontrado facilmente no mercado (também conhecido como “equipamento de prateleira”).

Dada uma determinada configuração do sistema, é preciso realizar dois tipos de calibração: a geométrica e a fotométrica. A calibração geométrica pode ser feita com a gravação de um tabuleiro quadriculado e posterior projeção de um mesmo padrão. Já a calibração fotométrica exige a projeção de *slides* que contenham uma das cores básicas por vez.

B.2.1 Câmera

A câmera usada (Figura B.2) suporta uma resolução máxima de 1920×1080 . Para a captura, a resolução adotada é 1280×720 em modo progressivo (também é suportado o modo entrelaçado). São usadas duas saídas: a HDMI (do inglês *High Definition Multimedia Interface*) e a de vídeo composto. Também há uma saída *firewire*, a qual não é usada. A saída HDMI é conectada à entrada de uma placa gráfica de captura. Já a saída de vídeo composto é conectada a uma entrada de vídeo do dispositivo *genlock*.

A câmera possui uma tampa em suas lentes, servindo de proteção às mesmas. Logo no início do procedimento de captura, essa tampa deve ser removida.

Para auxiliar na visualização e enquadramento da cena, a câmera é equipada com uma tela LCD lateral que, para ser aberta, deve-se pressionar um botão para destravá-la. A câmera é ligada por meio de um botão giratório que também possui uma pequena trava que deve ser pressionada para liberá-lo.

O enquadramento da cena na câmera (Figura B.3) pode exigir a movimentação do tripé que a contém, de modo a afastá-la ou aproximá-la da cena a ser gravada. Efeito semelhante pode ser obtido por meio do controle de *zoom*, que localiza-se na parte de cima da câmera: um botão marcado com os caracteres ‘W’ (do inglês *wide angle*, para afastar a imagem da cena) e ‘T’ (do inglês *telescopic*, para aproximar a imagem

Figura B.1: Câmera e projetor são acoplados a um trilho que por sua vez é preso a um tripé com rodas e vários pontos de articulação.

Figura B.2: Câmera usada no sistema de captura de vídeo 3D.

Figura B.3: Enquadramento da cena na câmera.

Figura B.4: Botão para fazer ajuste manual da íris da câmera.

da cena). Eventualmente, dependendo do brilho refletido pelos objetos, é necessário ajustar a íris da câmera, o que é feito com o uso do botão indicado na Figura B.4.

B.2.2 Projetor

O projetor usado (Figura B.5) é da marca Benq e possui tecnologia DLP, na qual a imagem é criada por espelhos microscópicos organizados em uma matriz.

O projetor recebe o sinal a ser projetado através de sua entrada VGA. Este sinal é enviado por uma placa gráfica NVidia Quadro instalada em um computador dedicado apenas a esta tarefa. Esta mesma placa recebe o sinal de sincronismo gerado pelo *genlock*.

Para ligar o projetor, basta pressionar o botão *POWER*. Ele também possui uma tampa protetora de sua lente, a qual deve ser removida quando da realização de uma captura. O ajuste de seu foco é feito ao se mover um botão giratório próximo à lente de projeção.

Figura B.5: Projetor usado no sistema de captura de vídeo 3D. O botão giratório para ajuste manual de foco encontra-se no canto superior esquerdo da figura da direita.

Figura B.6: Genlock, dispositivo responsável pela sincronização entre câmera e projetor.

B.2.3 Genlock

A captura de vídeo 3D exige que câmera e projetor estejam sincronizados de modo que não haja perda de quadros e que não seja capturado um quadro pela câmera enquanto o projetor está alterando entre a projeção de um *slide* e outro. Tal sincronização é realizada por um dispositivo denominado *genlock*, também conhecido como *black generator* (Figura B.6).

O *genlock* usado é da marca Horita, modelo BG-50. Além do cabo de alimentação, outros dois são ligados a ele. Na entrada de vídeo (**VIDEO IN**) é conectado o cabo que vem da saída de vídeo composto da câmera. O dispositivo gera então o sinal de sincronização e o envia por sua saída **BLK-1**, à qual está conectado um cabo do tipo coaxial com um adaptador BNC. A outra extremidade deste cabo é ligada à placa NVidia Quadro, a mesma que envia o sinal para o projetor. Esta configuração determina o sincronismo entre câmera e projetor no sistema.

Para ligar o *genlock*, basta alternar o botão **POWER** de **OFF** para **ON**.

B.2.4 Monitores

Há dois monitores no sistema (Figura B.7), um conectado à máquina dedicada à projeção das faixas e o outro conectado à máquina dedicada à captura e ao processamento de vídeo 3D. Ambas as máquinas possuem o sistema operacional Windows XP instalado devido à maior facilidade existente para se encontrar *drivers* para o equipamento usado.

É importante seguir a ordem descrita na Seção B.1 para ligar todos os equipamentos, já que isso garante que, ao se ligar o monitor conectado à máquina dedicada à geração dos slides, a imagem seja enviada a ele sem problemas.

B.2.5 O programa v4dFaixas

O programa v4dFaixas (Figura B.8) foi desenvolvido usando a biblioteca GLUT (da sigla em inglês *Open Graphics Library Utilities Toolkit*). Há vários atalhos que permitem alterar parâmetros como, por exemplo, a quantidade de *slides* diferentes a serem projetados ou a quantidade de faixas por *slide*. Os atalhos mais importantes estão listados nas tabelas B.1 e B.2.

B.2.6 Cena

O ajuste dos objetos na cena a ser gravada (Figura B.9) deve levar em conta diversos detalhes. Por exemplo,

Figura B.7: Monitores usados no sistema de captura de vídeo 3D.

Figura B.8: O programa v4dFaixas.

Atalho	1	2	3	4	5	6	7	8	9	0	-	=
Faixas	10	20	30	40	50	60	70	80	90	100	160	200

Tabela B.1: Atalhos do programa v4dFaixas: quantidade de faixas exibidas.

Atalho	Tipo de slides
q	8 slides monocromáticos
,	5 slides coloridos
.	5 slides monocromáticos

Tabela B.2: Atalhos do programa v4dFaixas: tipo de slides exibidos.

Figura B.9: Ajuste dos objetos na cena a ser gravada.

Figura B.10: *Utensílios.*

alguns utensílios favorecem o movimento de alguns objetos de interesse ou evitam que outros objetos sejam capturados (Figura B.10).

Agradecimentos

AOS PROFESSORES Luiz Velho e Roberto Cesar, por me darem a oportunidade da viagem ao IMPA para usar o sistema de captura de vídeo 3D; ao Luiz Fernando Magalhães Cordeiro, programador do Visgraf, por ensinar todo o processo de captura e dar total acesso ao código dos programas envolvidos; ao Djalma Lúcio Soares da Silva, administrador do Visgraf, por fornecer todo o suporte necessário referente ao acesso a máquinas e a salas; e ao Thiago Siqueira Pereira, que testou e criticou a eficácia deste manual como um usuário leigo na captura de vídeo 3D.

Apêndice C

Pôster SIBGRAPI 2012

O PÔSTER DA PÁGINA SEGUINTE foi apresentado em 25 de agosto de 2012 no *Workshop of Works in Progress* do SIBGRAPI 2012 (*Conference on Graphics, Patterns and Images*), realizado na Universidade Federal de Ouro Preto, Minas Gerais. Este mesmo pôster foi exposto e apresentado oralmente no Simpósio de Iniciação Científica e Pós-Graduação do IME-USP em 7 de novembro de 2012 (atas com os artigos no prelo).

MOTION SEGMENTATION FROM TEXTURE AND DEPTH IMAGES USING GRAPH HOMOMORPHISM

DAVID DA SILVA PIRES
IME-USP

ROBERTO M. CESAR-JR
IME-USP

LUIZ VELHO
IMPA

ABSTRACT

We present a method for motion estimation from videos captured by depth-sensing cameras. Our method uses the technique of graph matching to find groups of pixels that move in the same direction in subsequent frames. A cost function, that consider both distances on RGB and XYZ spaces, is minimized in order to choose the best matching for each patch.

CONTRIBUTIONS

The developed method uses graph matching (homomorphism) algorithms on RGBD images to solve a challenging problem: motion estimation on unrestricted scenes, with respect to illumination and to the kind of moving objects. The infrared light allows to obtain shape information even in low-light scenes. The objects can be rigid or deformable, articulated or not. The developed technique generalizes some cases not entirely treated on previous works (which were based only on texture data) to 3D cases, which also make use of depth map information in the scene.

RESULTS

Comparison between matchings: The figure at left shows experiments applied to the two RGBD images displayed on top, which presents a video with a couple of samba dancers. The figure shows a comparison between matchings, where the value of the weight α and the size of the patches are changed.

Method robustness: In order to demonstrate the robustness

of the implemented algorithms, the image at right shows a sequence of non-consecutive frames of a same video, where two subjects walk to opposite sides. Note that the color that indicates the direction of the movement of each subject (red to right and blue to left) is kept along the time, without suffering great variations.

METHOD

Graph matching: In order to find a matching, we use six values for each pixel on input data: RGB and (x, y, z) data. The frame representation is given by an attributed relational graph. The recognition of the direction of the movement is done through an inexact graph matching. This approach has been used to solve many computer vision problems such as object tracking [1], interactive natural image segmentation [2], among others [3].

Graph generation: The model and the input graph are built from consecutive pairs of texture and depth map input frames. In order to build the graph, we consider the representation of the input images (texture and depth map) composed by patches of $m \times n$ pixels.

COST FUNCTION

The cost function c is given by a weighted average between two distances, d_{RGB} and d_{XYZ} :

$$c = \alpha \cdot d_{RGB} + (1 - \alpha) \cdot d_{XYZ},$$

with $\alpha \in [0, 1]$. The d_{RGB} value measures the distance of the color of the patches being compared on RGB space, while the d_{XYZ} value measures the distance between the (x, y) texture coordinates and the Z depth values.

DEPTH MAP FILTERING

Controlling the visualization volume thresholds: from an image where all pictorial elements are present inside the frustum, it is possible to remove the background (approximating the far plane) or remove objects that are closer than the target object (moving away the near plane).

MOTION VISUALIZATION

The developed method allows three types of graph matching visualization:

- arrows;
- color patches;
- gray level patches.

The first one draws, for each matching, an arrow (with a circular tip) that goes from the position of a vertice on model graph to the position of the matched vertice on input graph. Also, the method allows the user to choose drawing the matchings as color or gray level patches. Each direction is represented by a colored label and the program paints the area of each patch with the corresponding label.

SOURCE CODE

The source code, with an interactive graphical user interface, is available at: <http://www.vision.ime.usp.br/~davidsp/4dvideo>

REFERENCES

- [1] GRACIANO, A. B. V. (2012) *Modelagem e reconhecimento de objetos estruturados: Uma abordagem estatístico-estrutural*. Ph. D. thesis, University of São Paulo (IME-USP), São Paulo, Brazil.
- [2] NOMA, A. (2010) *Dois abordagens para casamento de padrões de pontos usando relações espaciais e casamento entre grafos*. Ph. D. thesis, University of São Paulo (IME-USP), São Paulo, Brazil.
- [3] CONTE, D., FOGGIA, P., SANSONE, C. & VENTO, M. (2004) Thirty years of graph matching in pattern recognition. *Intl. Journal of Pattern Recognition and Artificial Intelligence* 18 (3), 265–298.

Apêndice D

Artigo 2013

O ARTIGO APRESENTADO neste apêndice vem sendo preparado com os últimos resultados que vemos obtendo e será submetido oportunamente.

Motion segmentation from texture and depth images using graph homomorphism

Roberto M. Cesar-Jr, David da Silva Pires
 Department of Computer Science
 Institute of Mathematics and Statistics (IME-USP)
 São Paulo, Brazil
www.ime.usp.br/~{cesar,davidsp}

Luiz Velho
 Visgraf Laboratory
 National Institute for Pure and Applied Mathematics (IMPA)
 Rio de Janeiro, Brazil
<http://lvelho.impa.br>

Fig. 1. Detection and segmentation of movement (right) using our method applied to texture (left) and depth (middle) images. Red and blue colors indicate movement to right and to left, respectively.

Abstract—We present an approach for motion segmentation from videos captured by depth-sensing cameras. This is an intermediary and necessary step to identify the rigid components of an articulated object. Our method uses the technique of graph matching to find groups of pixels that move to the same direction in subsequent frames. In order to choose the best matching for each patch, we minimize a cost function that accounts for distances on RGB and XYZ spaces. For parameters that decrease the resolution of the images, our application runs at real-time rates.

Keywords—motion segmentation; graph matching.

I. INTRODUCTION

With the advent of devices like KinectTM (from Microsoft[®]) and XtionTM (from ASUS[®]) that can capture texture and depth images from a scene, there are many new challenges and problems to be faced. The main application of data captured by such equipment is generally concerned with natural interaction. These applications use anthropometric algorithms to estimate pose, skeleton and the number of users in front of the device. Some systems have specialized algorithms to recognize its users, even if there is identical twins among them [1]. It has been attempted to identify gestures to use KinectTM as a control to other devices, aiming an easier or more natural interaction and allowing the use of computers with great accessibility. Examples of works like these include interfaces to use at presentations with slides [2] and the industry of games [3]. Fig. 2 shows the two cited depth sensors.

Besides this kind of application, we can also use the additional information provided by the depth image to improve

Fig. 2. Two depth-sensing devices in use today.

or speed-up algorithms that, until now, were applied only to texture images.

Our approach can be used on situations more general than those with humans making gestures in front of the capture device. The presented method can deal with occlusion and spatial continuity and is independent of frames that are used for initialization. The motion has to be a piecewise parametric one, in order to be possible identify the rigid components that are moving.

Objective: This work aims to show the benefits of using the additional information given by the depth image registered with a texture image, presenting an algorithm that detects the direction of movement at real-time rates. The developed procedure shows, with labels identified by colors, to which direction each group of pixels (called patch here) is moving. Depth data, when added with the traditional RGB values and texture coordinates, eases the delimitation of objects of interest and turns results substantially better when considered as a descriptive feature for each pixel. This characteristic is an advantage over methods that depends on the presence of a well

defined pattern on texture, like checkerboard sequences [4].

Motivation: This technique can be used on more general applications, such as 3D scene reconstruction and augmented reality. On the former, one could reach better registering of data once the parameters of the movement of an object, such as direction and velocity, are known. On the latter, artificial objects of the augmented reality could answer better relative to user's motion. Our approach is also an intermediary step to identify the rigid components of an articulated object [5], [6]. Once we know, for each patch, the direction of the movement, we can cluster similar patches relative to this feature. By accumulating such similarities through a video sequence, it's probable that the rigid components turn clearly identifiable and, by comparing proximity among these components and noticing differences on the directions of their movements, it would be possible to identify the points of articulation too. Other research field that could take advantage over the additional information provided by the depth map is the tracking of 3D human motion. We believe that is possible to improve good works developed on this area [7].

Given a video sequence, such as the one shown at Fig. 3, the application builds a graph for each frame and compare them, between consecutive ones, finding a matching based on distances at RGB and XYZ spaces. These matchings are codified as color labels for visualization purposes.

Fig. 3. Input data is a video sequence of one texture and one depth map images per frame. The texture image provides RGB values while the depth map image gives us triplets (x, y, z) .

A. Related work

The evolving technology inherent to the current depth-sensing devices was initially created to provide natural interaction to video-games but all the research that has been done on areas such as Computer Graphics and Computer Vision has shown a lot of other interesting uses. That is why it has been given so much importance to the development of software that facilitates the use of this equipment.

Although this technology is relatively new, it would not be surprising if, in few years, the same novelty will be present on thin computing clients, like smart-phones and tablets. With the recent growth in display resolutions and graphics processing power of these small devices, it has been possible to leverage these trends and implement software that deals with 3D stereoscopic data, but not without a lag on the processed data [8], hence the urge for efficient and fast algorithms.

The Kinect Identity technology [1] selects a set of three independent identification techniques: face recognition, clothing color tracking and height estimation. These techniques were selected from a major set and demonstrated to be the best ones that, at the same time, were robust, non-CPU and memory intensive and as independent as possible from each other. Such choice indicates the importance of the development of tracking algorithms that uses both kind of data: texture and depth map.

The lack of a better treatment of the depth data in conjunction to the RGB data and, consequently, the use of these on motion segmentation algorithms is felt even by developers of software specialized in gesture recognition, like FFAST, the Flexible Action and Articulated Skeleton Toolkit [9]. They have demonstrated interest in developing real-time head tracking and estimation of the twist of the user's arm. None of these are provided by the middleware OpenNI™ and the solutions to these problems certainly involves computer vision techniques to be applied over both input data.

In the present work, we use the theory of graph matching to find a correspondence between two point sets. This approach has been used to solve many computer vision problems such as interactive natural image segmentation [10], shape matching [11], computer-assisted colorization [12], object tracking [13] and point matching for non-rigid registration [14], among others [15].

II. METHODOLOGY

A. Input data

Our algorithm takes as input a sequence of pairs of registered RGB texture and depth map. Since the acquired input depth map is already registered with the texture image, there is no need for using knowledge about intrinsic or extrinsic calibration parameters between the infra-red light receptor and the camera.

Besides Kinect™'s data, the developed software also accepts input from a variety of other sources, such as video files, web-cam, video 4D files [16] and sequences of image files. Thus, it is easy to compare the results (and eventually confirm the advantages) when using the additional depth map information with approaches that use only texture information.

The option of loading sequences of image files from disk is a very interesting approach that we adopted to make off-line processing. This can be necessary on situations where the parameters to the implemented algorithms imply on CPU-intensive work, increasing the delay between two consecutive captures. In these cases, we can obtain all the desired data at once, recording it on disk without any processing, in order to speed up acquisition; then, we can run the respective program on these images, loading each one only when the system is ready for the next pair of frames.

The developed system is also capable of acquiring the disparity map, the gray-level (instead of the colored RGB) texture, the infrared image and a mask that indicates where the depth map is valid.

B. Pipeline

The data processing occurs according to a specific pipeline that is composed of the following steps:

- data acquisition;
- texture filters;
- depth map filters;
- graph algorithms;
- data visualization.

The data flow is schematized in Fig. 4.

Fig. 4. Data flow through the implemented method. Data visualization can be done at final results as well as at each intermediary step.

Some of the texture and depth map filters can be applied in parallel processes, since they are independent. This feature could accelerate the data flow through the pipeline, mainly because a great variety of modern machines has more than one CPU, CPUs with many cores or even programmable GPUs with many cores. In spite of this, we still have not parallelized our algorithms and thus the data actually follow the presented steps as a serial pipeline.

Although the main interest is on final results, it is also important to visualize the data on intermediary steps. With this in mind, we designed a graphical user interface that allows simultaneous data visualization on the different steps of the pipeline.

C. Texture filters

Our system has various texture filters implemented and two of them are particularly important to obtain the desired results: (i) flip on the horizontal direction and (ii) decrease the resolution. Both filters are applied on texture and depth images.

The flip on the horizontal direction filter is used for pragmatic reasons: when a user interacts with the system, it is easier to look at the results as if it was in front of a mirror.

Since the acquisition device is commonly located on a place next to the screen of the computer, facing to the user, a flip on the horizontal direction is necessary to make real-time interaction easier.

The filter to decrease resolution, besides accelerating all the computations that are done, is closely related to two other factors:

- 1) the velocity of each moving object;
- 2) the distance of each moving object to the capture device.

These two quantities interfere with the precision of movement detection. Small resolutions only allow detection of movements that involve the modification of many pixels at original images, since many pixels are represented by just one pixel at the small resolution image. Then, the movement will be perceived only if the object is close to the capture device or if its velocity is sufficient to traverse a relatively considerable amount of pixels at consecutive frames.

We implement the decreasing of the image resolution by constructing patches. A patch is defined as a rectangular region on the image, thus allowing squared patches. All the pixels that belong to this rectangular region have their values (RGB or depth) changed to the respective values of the patch. In order to simplify this conversion, we consider a grid on which the distance between vertical lines is defined as the value of the patch width and the distance between horizontal lines is the value of the patch height. We don't deal with multi-resolution, so the patches have the same dimensions on the entire image. This results on a regular grid, equally spaced on each direction. On the other hand, the system can deal perfectly with changes on the size of the patches from one frame to another.

Computing the mean patches: As can be inferred from the last paragraph, there is an important choice when constructing a patch: its RGB and depth values. Our method allows two options:

- 1) the original values of the most top-left constituent pixel of the patch;
- 2) the arithmetic average of the values of all the pixels that compose the patch.

We also use a median filter, but it is applied to the image that is generated after building the graph representation, so we will comment on it later.

D. Depth map filters

Filters that deal with depth maps are much like those that deal with texture images, since depth maps are also images. But we can explore some advantages by the fact that depth maps are gray-level images that can be seen as a tridimensional surface.

Therefore, we implemented a depth map filter that allows us to choose data that is enclosed by two thresholds. This filter relies just on the determination on these two parameters, which determine the near (clip front) and far (clip rear or clip back) planes of the visualization volume. Indeed, these values control a fine tuned frustum that is restricted to the area where the objects of interest are present. Actually, we can

precisely remove the background and objects that are closer to the capture device than the target objects.

Also related to the depth map is the texture mapping filter. This filter maps the texture on the depth or disparity map. We can also colorize these two gray-level images in order to get a better visualization of the data (see Fig. 5). The texture mapping aims to confirm the accuracy of the registration between the texture and depth images or between the texture and disparity images. Although we have not developed a viewer that allows us to change the point of view of the camera so that it could be possible to easily perceive the correct alignment on registered data, we can confirm this feature by averaging the values of the pixels from the two images, simulating transparency, like an alpha channel.

Fig. 5. Example of texture mapping over the colorized disparity. Each image participates with 50% on the RGB value. At disparity image, color values go from red to blue as the distance to the capturing device increases.

E. Graph based approach

In order to find a matching, we have to consider relevant features that describe the points and to have a way to compare these quantities. Thus, each frame generates a graph whose vertices are derived from patches properties. We used six values for each pixel on input data: RGB data, extracted from color channels, and (x, y, z) data, with x and y being texture coordinates and z being the distance to the capturing device. The frame representation is given by an attributed relational graph (ARG), allowing storage and comparison of structural, temporal and quantitative information. The recognition of the direction of the movement is done through an inexact graph matching. This approach allows differences between model and input graph.

An ARG is a graph whose vertices represent objects while edges denote relations among them. Objects can be characterized by a finite number of attributes (numerical or symbolic), such as area, perimeter, color and shape. The relations correspond, most times, to a distance or similarity measure [13].

With the contents of each frame being represented by an ARG, motion segmentation resumes to a graph matching, consisting of a determination of a mapping of the vertex set of an ARG to the vertex set of another one.

Each graph is treated as a complete graph, in the sense that every vertex is connected to all the other vertices. In fact, we compute a cost function involving the distance between the RGB and depth values of each two sets of vertices: (v_m, v_i) , where v_m and v_i are vertices from model and input graph, respectively. The pair that minimizes this cost function is added to the matching.

While graph vertices store point patterns, including position information about these points, structural relations are stored at graph edges. However, in this case, since both structural and appearance information can be computed from the data stored on each vertex, the position of the patches on the image are used to infer the modulo (size) and the direction of the vectors on RGB and XYZ spaces.

F. Graph generation

The model and the input graph are built from consecutive pairs of texture and depth map input frames. Thus, at the beginning of the acquisition procedure, we can build just one graph. As the subsequent frames are captured, the input graph relative to the immediate past frame is assigned to the model graph and a new input graph is built from the new data acquired.

In order to build the graph, we consider the representation of the input images (texture and depth map) composed by patches. Given an image and the patches' parameters, we can compute how much patches compose the new representation, being sufficient for this to make a division between the number of rows and columns of each one. Thus, patches' dimensions are directly related with the size of the graphs that are created, highly influencing the performance of the application.

Each patch is a candidate to have a vertex representing it on the graph, being elected based on its Z (depth) value. Only if this value belongs to the viewing volume defined earlier, at the depth filtering stage, a new vertex is created, have its fields filled with the data of the patch and is inserted on the graph. A consequence of this approach is that only pixels that have non-null depth values are considered. So, pixels that belong to shadow areas on depth map or that were too close or too far of the capturing device to have reliable values are out of scope. This perspective treats the depth map as a valid mask to texture pixels.

Since we apply the decreasing resolution filter, the values that compose the patch's fields can always be obtained from the most top-left pixel, even if the choice of computing mean patches is activated.

The directed graph data structure is implemented as an adjacency list, on which there is a vector of vertices and, for each vertex, there is a list of out-edges. Data stored at vertices include row, column (both texture related), Z coordinate (value of depth image at the same row and column), red, green and blue values from texture image, and a time-stamp, that

indicates the number of the captured frame since the capturing process was started.

G. Graph matching

The matching is done between two graphs: model and input. The model graph represents the last pair of frames (texture and depth images), captured before the current one, which is represented by the input graph.

A matching is an ordered pair of vertex descriptors, the first one referring to a model graph vertex and the second one relative to an input graph vertex. See Fig. 6 for a graph matching example.

Fig. 6. Graph matching example. Note that this matching is an homomorphism: all the vertices of the model graph are matched, there are vertices at input graph that do not participate of any matching and others that participate on it more than once.

For each vertex belonging to the model graph, we find the vertex on the input graph that is closest to it according to a distance measured by a cost function. Eventually, the matched vertices have the same (x, y) texture coordinates, indicating that no movements has occurred at that location between the two pairs of frames.

The cost function c is given by a weighted average between two distances: d_{RGB} and d_{XYZ} :

$$c = \alpha \cdot d_{RGB} + (1 - \alpha) \cdot d_{XYZ}.$$

The d_{RGB} value measures the distance of the color of the patches being compared on RGB space, while the d_{XYZ} value measures the distance between the (x, y) texture coordinates and between the Z depth values. As it can be seen, the parameter α controls how much each distance is considered at the final value of the cost function. As the value of α varies from 0 to 1, the value of each weight changes accordingly. This arrangement allows us to assign different importance to each computed distance, while the sum of the weights still remains 1.

When calculating the cost function, we need to decide about which distance function to use. We implemented two of them: city block (Manhattan) and Euclidean distances. Here is the Manhattan distance formula:

$$d_{RGB} = |v_R^M - v_R^I| + |v_G^M - v_G^I| + |v_B^M - v_B^I|$$

and this is the Euclidean distance formula:

$$\begin{aligned} d_{RGB} &= \|v_{RGB}^M - v_{RGB}^I\|_2 \\ &= \|(v_R^M - v_R^I, v_G^M - v_G^I, v_B^M - v_B^I)\|_2, \end{aligned}$$

where the raised indexes M and I indicate if the vertex belongs to the model or input graph, respectively, and the sub-indexes R, G and B indicates the channel color being considered. The same formulas are applied to (x, y, z) depth map values.

H. Results visualization

We developed different kinds of visualization of the results and of the intermediary steps.

Patches: To indicate the construction of patches, the program gives to the user three choices:

- draw the representing vertices of each patch;
- draw a grid that shows the interface among the patches;
- draw each patch by spreading the color of the pixel that represents it over the occupied rectangular area.

Graph: Since each vertex of the graph represents a patch on a filtered pair of images, we adopted the strategy of drawing a graph by just drawing these represented patches. In this way, a graph can be shown by following the same three kinds of patches drawing, but emphasizing which patches were considered to its construction after the application of background elimination by the depth filter. Refer to Fig. 7 for an example of this three drawing choices active.

Fig. 7. Drawing patches options. Only the patches considered for graph generation were drawn on figure. The drawing of the grid and the computation of mean patches options were also activated.

Matchings: Matchings can be drawn in three ways:

- arrows;
- color patches;
- gray-level patches.

The first one draws, for each matching, an arrow that goes from a vertex location on the model graph to the position of the matching vertex on the input graph. A consequence of this approach is that we have, on this visualization, an arrow starting at each vertex of the model graph, but we can have, on the input graph, vertices that are end points of more than one arrow and thus others that are not destination of any arrow. This happens because this relation between the two graphs is an homomorphism, not an isomorphism.

If instead of drawing arrows the user chooses the option of drawing the matchings as color patches, then each direction

is represented by a colored label and the program paints the whole area of each patch with the corresponding label. At this type of drawing, the user can still choose between 8 or 4 directions. The same representation is adopted on gray-level patches. To ease the use of this schema, the graphical interface includes a label, where the colors indicate the respective directions, as can be seen on Fig. 8.

Fig. 8. 8-color, 4-color and 4-gray level labels that indicate the direction of the movement.

Computation of directions: In order to compute the direction to which a patch is moving on, we calculate the arccosine of the normalized dot product between the vector that represents the shift of the patch and the unit vector $(1, 0)$, that points to the direction of the x axis. After that, we discover to which quadrant the vector belongs to by considering the signal of its y coordinate.

I. Median filter

Final results visualization shows that there are many patches that correctly identify the direction of the movement, but still others represent bad matchings. In order to eliminate these ones and treat this lack of spatial continuity, the median filter is applied to the image of the labeled patches. The height of the image can be calculated by dividing the height of the original (input) image by the patch height; the same procedure applies to the width. The median filter is parameterized by the size of the kernel, which has to be odd.

J. Performance

The system runs in real-time when the parameters involved on the construction of the graph imply on few vertices. These parameters include the patches dimensions and the near and far values that determine the visualization volume. As more data is present inside the frustum, more vertices are created at the graph. On cases where the value of the parameters slows down the system, it is possible to run the program on a batch mode, reading images previously recorded on hard disk.

Kinect™ is not able to capture texture and depth images (or any other of the possible types of input data) at the same time. Hence, if there is some kind of data processing between the two captures, a perceptible difference on moving objects appears, indicating that the data do not match. It is then required to capture all the necessary images in a sequence and then process the data. The inherently delay existent between two consecutive captures, without any processing in the middle, is acceptable for our purposes.

Besides texture and depth images, it is also possible to capture the disparity map, the valid depth mask and a one-channel gray-level texture. As we do not use these on our

algorithms and to avoid any additional delay, they are not captured.

All the processing can be done in real-time, but just for patches of relatively big dimensions (above 10×10 pixels). With patches under this area, the processing is slower and it is necessary another strategy for reading the input. Our approach is to take the input images (texture and depth), record them as image files and post-process them off-line as a batch system.

K. Hardware and software

The access to Kinect™ data is done with the use of OpenNI™ through an OpenCV VideoCapture object. Our code is written in C++, with a graphical user interface developed with the Qt API. Images are captured with VGA resolution (640×480) at 30 Hz. The depth data has a resolution of 11 bits, representing distances to camera from approximately 40 cm to 6 m. In order to construct the graphs we used the Boost Graph Library.

III. RESULTS AND DISCUSSION

Fig. 9. Typical input and output data. The shown depth data has already been processed by background elimination filter.

Typical input and output data are shown at Fig. 9, followed by isolated frames on the subsequent figures.

Fig. 10 shows the captured depth of two subjects walking at opposite directions, with occlusion occurring between them and also between their respective legs, while Fig. 11 shows this depth after background elimination filtering. Fig. 12 shows the texture for the same scene. Note that the depth is already registered with the texture. Finally, Fig. 13 shows the detected motion represented as color labels. As we can see, the method accounts for the effects of occlusion. This experiment shows the identification of motion present on the scene. The leftmost subject is more distant to the capturing device, as indicated by the gray levels in the depth map; it is walking from left to right. The other subject, closer to the capturing device, executes a movement from right to left. Note the correct classification of both movements, even on the region where they intercept each other. The green pixels that arise on Fig. 13 were identified as moving up, a reasonable result, except for the green blob that appears at left-bottom corner: it appears due to error on depth capturing. This same experiment is also an example of how our method gets successful results even in the presence of occlusion of moving objects. Note how a leg is partially occluded by another one and still has its motion correctly identified.

For the input shown on Fig. 14, we obtained the output on Fig. 15. This experiment shows the nice visual appealing that

Fig. 10. Captured depth of two subjects walking at opposite directions. This also exemplifies a case of occlusion between moving objects.

Fig. 12. Texture image for the same scene shown at Fig. 10. Note that the depth image is already registered with the texture.

Fig. 11. Depth shown at Fig. 10 after filtering for background elimination.

Fig. 13. Detected motion represented as color labels.

the representation of matchings as arrows brings. Instead of color labels, arrows with circled tips are used to indicate the motion. This example was captured on real-time.

The last presented experiment exemplifies the fine grained result that we obtain when working with small patches. Fig. 16 shows the output when the method is applied to the same video shown at Fig. 1, but with patches of size 2×2 pixels, instead of 10×10 . The processing time increases substantially with such modification. In these cases it is necessary to run the method on off-line mode. This figure also shows a collateral effect of OpenCV median filter: once the filtering is done considering each channel separately, the output image can eventually be composed of colors that were not present at input image.

With all the described features, the developed system has also shown to be a convenient framework to deal with input

data generated by the new depth-sensing devices. The application is capable of doing image processing and execute computer vision algorithms, thus allowing easy evaluation of results at real-time rates.

A. Future work

There are many features on our method on which we can improve the results or the performance:

- As mentioned earlier, most filters that are applied on the texture images are independent of filters that are applied on the depth map images. Thus, these filters can be parallelized to run on different processes.
- Patches that are located on the interior of moving objects that are too thick have the tendency to not be shown on results visualization. This happens because the texture and the depth on such locations do not alter too much

Fig. 14. Another video sequence, where a subject is moving his arm.

Fig. 15. Visualization of the matchings as arrows, indicating that the arm is moving up and to the right direction.

from one frame to another. The median filter minimizes this effect, but it is not sufficient. In order to deal with this problem, we could take a temporal approach. Each vertex has a time-stamp stored on its fields, indicating the number of the frame to which that vertex belongs. Once a patch stays at the same location from one frame to another, we could use this data to continue labeling it with the last color label that was assigned to it, until a predefined number of frames is processed.

- The direction of the movement of the patches is calculated in the plane that is determined by texture coordinates. By using the Z value given by the depth map, we could determine the motion by considering directions on 3D space.
- It is possible to apply this method to detect the rigid parts of an articulated object.

ACKNOWLEDGMENT

REFERENCES

- [1] T. Leyvand, C. Meekhof, Y. C. Wei, J. Sun, and B. Guo, "Kinect identity: Technology and experience," *Computer*, vol. 44, pp. 94-96, 2011.
- [2] M. A. da Silva, "Técnica de navegação em documentos utilizando Microsoft® Kinect™," December 2011, undergraduate work.
- [3] Microsoft®. (2012, May) Xbox 360 - official site. [Online]. Available: <http://www.xbox.com/en-US/>

Fig. 16. Fine grained result using patches of 2×2 pixels.

- [4] R. Tron and R. Vidal, "A benchmark for the comparison of 3-D motion segmentation algorithms," in *IEEE International Conference on Computer Vision and Pattern Recognition*, June 2007.
- [5] M. P. Kumar, P. H. S. Torr, and A. Zisserman, "Learning layered motion segmentations of video," in *Proceedings of the International Conference on Computer Vision*, 2005.
- [6] P. H. S. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *International Workshop on Vision Algorithms*, W. Triggs, A. Zisserman, and R. Szeliski, Eds., 1999, pp. 278-295.
- [7] X. Zhao and Y. Liu, "Generative tracking of 3D human motion by hierarchical annealed genetic algorithm," *Pattern Recognition*, vol. 41, pp. 2470-2483, 2008.
- [8] J. L. Olson, D. M. Krum, E. A. Suma, and M. Bolas, "A design for a smartphone-based head mounted display," *IEEE Virtual Reality*, pp. 231-232, March 2011.
- [9] E. A. Suma, B. Lange, A. S. Rizzo, D. M. Krum, and M. Bolas, "FAAST: The flexible action and articulated skeleton toolkit," *IEEE Virtual Reality*, pp. 245-246, March 2011.
- [10] A. Noma. (2012, May) Interactive natural image segmentation. [Online]. Available: <http://structuralsegm.sourceforge.net/>
- [11] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509-522, 2002.
- [12] A. Noma, "Duas abordagens para casamento de padrões de pontos usando relações espaciais e casamento entre grafos," Ph.D. dissertation, University of São Paulo, São Paulo, Brazil, July 2010.
- [13] A. B. V. Graciano, "Rastreamento de objetos baseado em reconhecimento estrutural de padrões," Master's thesis, University of São Paulo, São Paulo, Brazil, March 2007.
- [14] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Comput. Vis. Image Underst.*, vol. 89, no. 2-3, pp. 114-141, 2003.
- [15] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *Intl. Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 3, pp. 265-298, 2004.
- [16] M. B. Vieira, L. Velho, A. Sá, and P. C. Carvalho, "A camera-projector system for real-time 3D video," in *Proceedings of IEEE International Workshop on Projector-Camera Systems (PROCAMS)*, San Diego, California, USA, June 2005, jointly with CVPR 2005.

Referências Bibliográficas

- 3DVSYSTEMS (2009). URL <http://www.3dvsystems.com>.
- AGIN, G. J. & BINFORD, T. O. (1976). Computer description of curved objects. *IEEE Transactions on Computers* **C-25**(4), 439–449.
- ALLEN, B., CURLESS, B. & POPOVIĆ, Z. (2002). Articulated body deformation from range scan data. In: *Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*. University of Washington, Association for Computing Machinery (ACM).
- ANANDAN, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision* **2**(3), 283–310.
- ANGUELOV, D., KOLLER, D., PANG, H.-C., SRINIVASAN, P. & THRUN, S. (2004). Recovering articulated object models from 3D range data. In: *20th Conference on Uncertainty in Artificial Intelligence (UAI)*, vol. 70 of *ACM International Conference Proceeding Series*. Computer Science Dept., Stanford University, Arlington, Virginia, United States: AUAI Press. URL <http://ai.stanford.edu/~drago/>.
- ASHBROOK, A. P., FISHER, R. B., WERGH, N. & ROBERTSON, C. (1999). Construction of articulated models from range data. In: *Proceedings of BMVC*.
- BAJCSY, R. & KOVACIC, S. (1989). Multiresolution elastic matching. *Computer Vision, Graphics and Image Processing* **46**(1), 1–21.
- BAKER, S., BLACK, M., LEWIS, J. P., ROTH, S., SCHARSTEIN, D. & SZELISKI, R. (2007). A database and evaluation methodology for optical flow. In: *Eleventh International Conference on Computer Vision*. Rio de Janeiro, Brazil.
- BAKER, S. & MATTHEWS, I. (2004). Lucas-Kanade 20 years on: A unifying framework: Part 1: The quantity approximated, the warp update rule, and the gradient descent approximation. *International Journal of Computer Vision* **56**(3), 221–255.
- BANNO, A., MASUDA, T., OISHI, T. & IKEUCHI, K. (2008). Flying laser range sensor for large-scale site-modeling and its applications in Bayon digital archival project. *International Journal of Computer Vision* **78**(2-3), 207–222.
- BARRON, J. L., FLEET, D. J. & BEAUCHEMIN, S. S. (1994). Performance of optical flow techniques. *International Journal of Computer Vision* **12**(1), 43–77.
- BELONGIE, S., MALIK, J. & PUZICHA, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(4), 509–522.
- BERGEN, J. R., ANANDAN, P., HANNA, K. J. & HINGORANI, R. (1992). Hierarchical model-based motion estimation. In: *Second European Conference on Computer Vision*. Santa Margherita, Liguere, Italy.
- BESL, P. J. & JAIN, R. C. (1985). Three-dimensional object recognition. *Computing Surveys* **17**(1), 75–145.

- BESL, P. J. & MCKAY, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(2), 239–256.
- BLACK, M. J. & ANANDAN, P. (1996). The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding* **63**(1), 75–104.
- BOLLES, R. C., BAKER, H. H. & MARIMONT, D. H. (1987). Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision* **1**, 7–55.
- BRAY, M., KOLLER-MEIER, E., MÜLLER, P., VAN GOOL, L. & SCHRAUDOLPH, N. N. (2004). 3D hand tracking by rapid stochastic gradient descent using a skinning model. In: *Proceedings in CVMP*.
- BREGLER, C. & MALIK, J. (1998). Tracking people with twists and exponential maps. In: *Computer Vision and Pattern Recognition (CVPR)*. Computer Science Division of U. C. Berkeley.
- BRUHN, A., WEICKERT, J. & SCHNÖRR, C. (2005). Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision* **61**(3), 211–231.
- CANESTA (2009). URL <http://www.canesta.com>.
- CHEUNG, G., BAKER, S. & KANADE, T. (2003). Shape-from-silhouette for articulated objects and its use for human body kinematics estimation and motion capture. In: *Proceedings of CVPR*.
- CHRISTENSEN, G., JOSHI, S. & MILLER, M. (1997). Volumetric transformation of brain anatomy. *IEEE Transactions on Medical Imaging* **16**(6), 864–877.
- CHUI, H. & RANGARAJAN, A. (2003). A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding* **89**(2-3), 114–141.
- CONTE, D., FOGGIA, P., SANSONE, C. & VENTO, M. (2004). Thirty years of graph matching in pattern recognition. *Intl. Journal of Pattern Recognition and Artificial Intelligence* **18**(3), 265–298.
- CURLESS, B. & LEVOY, M. (1996). A volumetric method for building complex models from range images. In: *Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*. Stanford University.
- DA SILVEIRA, M. A. (2011). Técnica de navegação em documentos utilizando Microsoft® Kinect™. Undergraduate work.
- DAI, S., BAKER, S. & KANG, S. B. (2009). An MRF-based deinterlacing algorithm with exemplar-based refinement. *IEEE Transactions on Image Processing* **18**(5), 956–968.
- DE HAAN, G. & BELLERS, E. B. (1998). Deinterlacing—an overview. In: *Proceedings of IEEE*, vol. 86.
- FAIRCHILD, M. (2012). URL http://www.cis.rit.edu/fairchild/WhyIsColor/books_links.html.
- FAIRCHILD, M. D. (2005). *Color Appearance Models*. Wiley, 2nd ed.
- FAUGERAS, O. D. & HEBERT, M. (1987). *Three-Dimensional Machine Vision*, chap. XII - The representation, recognition and positioning of 3-D shapes from range data. Boston: Kluwer Academic Publishers, pp. 301–353.
- FLEET, D. & JEPSON, A. (1990). Computation of component image velocity from local phase information. *International Journal of Computer Vision* **5**(1), 77–104.
- FUH, C.-S. & MARAGOS, P. (1991). Motion displacement estimation using an affine model for image matching. *Optical Engineering* **30**(7), 881–887.
- GAI, J. & KANG, S. B. (2009). Matte-based restoration of vintage video. *IEEE Transactions on Image Processing* **18**, 2185–2197.

- GALL, D. L. (1991). MPEG: A video compression standard for multimedia applications. *Communications of the ACM* **34**(4), 46–58.
- GAVRILA, D. M. & DAVIS, L. S. (1996). Tracking of humans in action: A 3-D model-based approach. In: *ARPA Image Understanding Workshop*. Computer Vision Laboratory of University of Maryland.
- GLASSNER, A. S. (1995). *Principles of Digital Image Synthesis*. San Francisco: Morgan Kaufmann Publishers.
- GOOL, L. V., TUYTELAARS, T., FERRARI, V., STRECHA, C., WYNGAERD, J. V. & VERGAUWEN, M. (2002). 3D modeling and registration under wide baseline conditions. In: *Proceedings of ISPRS Commission III*, vol. 34. Part 3A, Photogrammetric Computer Vision.
- GRACIANO, A. B. V. (2007). *Rastreamento de objetos baseado em reconhecimento estrutural de padrões*. Master's thesis, University of São Paulo, São Paulo, Brazil.
- GRACIANO, A. B. V. (2012). *Modelagem e reconhecimento de objetos estruturados: uma abordagem estatístico-estrutural*. Ph.D. thesis, Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP), São Paulo, SP.
- GREST, D., WOETZEL, J. & KOCH, R. (2005). Nonlinear body pose estimation from depth images. In: *Proceedings of DAGM*.
- HANSEN, M., ANANDAN, P., DANA, K., VAN DER WAL, G. & BURT, P. (1994). Real-time scene stabilization and mosaic construction. In: *IEEE Workshop on Applications of Computer Vision*. Sarasota.
- HEALEY, G. E. & SHAFER, S. A. (1992). *Color. Physics-Based Vision: Principles and Practice*. Cambridge, MA: Jones & Bartlett.
- HOGG, D. (1983). Model-based vision: A program to see a walking person. *Image and Vision Computing* **1**(1), 5–20.
- HORN, B. K. P. (1986). *Robot Vision*. Cambridge, Massachusetts: MIT Press.
- HORN, B. K. P. & SCHUNCK, B. G. (1981). Determining optical flow. *Artificial Intelligence* **17**, 185–203.
- HORN, B. K. P. & WELDON JR., E. J. (1988). Direct methods for recovering motion. *International Journal of Computer Vision* **2**(1), 51–76.
- HUANG, T. S. (1981). *Image Sequence Analysis*. Berlin, Heidelberg: Springer-Verlag.
- IRANI, M., ROUSSO, B. & PELEG, S. (1997). Recovery of ego-motion using image stabilization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(3), 268–272.
- IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A. & FITZGIBBON, A. (2011). KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In: *24th annual ACM symposium on User interface software and technology*. St. Andrews, UK: ACM.
- KANADE, T. (ed.) (1987). *Three-Dimensional Machine Vision*. Boston: Kluwer Academic Publishers.
- KLETTE, R., SCHLÜNS, K. & KOSCHAN, A. (1998). *Computer Vision: Three-Dimensional Data from Images*. Singapore: Springer-Verlag.
- KNOOP, S., VACEK, S. & DILLMANN, R. (2006). Sensor fusion for 3D human body tracking with an articulated 3D body model. In: *Proceedings of IEEE Conference on Robotics and Automation*.
- KOKARAM, A. (2004). On missing data treatment for degraded video and film archives: a survey and a new bayesian approach. *IEEE Transactions on Image Processing* **13**(3), 397–415.

- KUMAR, M. P., TORR, P. H. S. & ZISSERMAN, A. (2005). Learning layered motion segmentations of video. In: *Proceedings of the International Conference on Computer Vision*.
- KYBIC, J. & UNSER, M. (2003). Fast parametric elastic image registration. *IEEE Transactions on Image Processing* **12**(11), 1427–1442.
- LAGANIÈRE, R. (2011). *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 recipes to master this library of programming functions for real-time computer vision*. Packt Publishing.
- LANITIS, A., TAYLOR, C. J. & COOTES, T. F. (1997). Automatic interpretation and coding of face images using flexible models. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19.
- LEE, M.-C., GE CHEN, W., LIN, C. L. B., GU, C., MARKOC, T., ZABINSKY, S. I. & SZELISKI, R. (1997). A layered video object coding system using sprite and affine motion model. *IEEE Transactions on Circuits and Systems for Video Technology* **7**(1), 130–145.
- LEYVAND, T., MEEKHOF, C., WEI, Y. C., SUN, J. & GUO, B. (2011). Kinect identity: Technology and experience. *Computer* **44**, 94–96.
- LIN, M. H. (1999). Tracking articulated objects in real-time range image sequences. In: *IEEE International Conference on Computer Vision (ICCV)*, vol. 1. Interval Research Corporation, Palo Alto, CA.
- LIVINGSTONE, M. (2008). *Vision and Art: The Biology of Seeing*. New York: Abrams.
- LUCAS, B. D. & KANADE, T. (1981). An iterative image registration technique with an application to stereo vision. In: *Seventh International Joint Conference on Artificial Intelligence*. Vancouver.
- LÉZORAY, O. & GRADY, L. (eds.) (2012). *Image Processing and Analysis with Graphs*. CRC Press, Taylor & Francis.
- MATSUSHITA, Y., OFEK, E., GE, W., TANG, X. & SHUM, H.-Y. (2006). Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**(7), 1150–1163.
- MATTHEWS, I. & BAKER, S. (2004). Active appearance models revisited. *International Journal of Computer Vision* **60**(2), 135–164.
- MATTHEWS, I., XIAO, J. & BAKER, S. (2007). 2D vs 3D deformable face models: Representational power, construction and real-time fitting. *International Journal of Computer Vision* **75**(1), 93–113.
- MICROSOFT® (2012). Xbox 360 - official site. URL <http://www.xbox.com/en-US/>.
- MIKIĆ, I., TRIVEDI, M., HUNTER, E. & COSMAN, P. (2001). Articulated body posture estimation from multi-camera voxel data. In: *Computer Vision and Pattern Recognition (CVPR)*. Computer Vision and Robotics Research Lab of Department of Electrical and Computer Engineering of University of California, San Diego, La Jolla, CA.
- MITRA, N. J., FLÖRY, S., OVSJANIKOV, M., GELFAND, N., GUIBAS, L. & POTTMANN, H. (2007). Dynamic geometry registration. In: *Eurographics Symposium on Geometry Processing*.
- MONTENEGRO, A. A., PEIXOTO, A., SÁ, A., MEDEIROS, E. & ESCRIBA, L. A. R. (2005). *Fotografia 3D*. Luiz Velho e Paulo Cezar Carvalho (eds.).
- MORIMOTO, C. & CHELLAPPA, R. (1997). Fast 3D stabilization and mosaic construction. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. San Juan, Puerto Rico.
- MÜNDERMANN, L., CORAZZA, S. & ANDRIACCHI, T. (2006). Markerless human motion capture through visual hull and articulated ICP. In: *Proceedings of NIPS Workshop*.

- NAGEL, H. H. (1986). Image sequences—ten (octal) years—from phenomenology towards a theoretical foundation. In: *Eighth International Conference on Pattern Recognition*. Paris.
- NAGEL, H.-H. & ENKELMANN, W. (1986). An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8**(5), 565–593.
- NOMA, A. (2010). *Duas abordagens para casamento de padrões de pontos usando relações espaciais e casamento entre grafos*. Ph.D. thesis, University of São Paulo, São Paulo, Brazil.
- NOMA, A. (2012). Interactive natural image segmentation. URL <http://structuralsegm.sourceforge.net/>.
- NOMA, A., GRACIANO, A. B. V., CESAR-JR, R. M., CONSULARO, L. A. & BLOCH, I. (2012). Interactive image segmentation by matching attributed relational graphs. *Pattern Recognition* **45**(3), 1159–1179.
- OKUTOMI, M. & KANADE, T. (1993). A multiple baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(4), 353–363.
- OLSON, J. L., KRUM, D. M., SUMA, E. A. & BOLAS, M. (2011). A design for a smartphone-based head mounted display. *IEEE Virtual Reality*, 231–232.
- PAPENBERG, N., BRUHN, A., BROX, T., DIDAS, S. & WEICKERT, J. (2006). Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision* **67**(2), 141–158.
- PEKELNY, Y. & GOTSMAN, C. (2008). Articulated object reconstruction and markerless motion capture from depth video. *Computer Graphics Forum* **27**(2), 399–408. Center for Graphics and Geometric Computing Technion, Israel.
- PIRES, D. S. (2007). *Rastreamento de componentes conexas em vídeo 3D para obtenção de estruturas tridimensionais*. Master's thesis, Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, Brasil. URL <http://www.vision.ime.usp.br/~davidsp/mestrado/dissertacao>.
- PIRES, D. S., CESAR-JR, R. M., VIEIRA, M. B. & VELHO, L. (2005). Tracking and matching connected components from 3D video. In: *Proceedings of 18th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*. Natal, Rio Grande do Norte, Brasil: IEEE Computer Society Press.
- PRADA, F., CRUZ, L. & VELHO, L. (2012). Object extraction in RGBD images. In: *Conference on Graphics, Patterns and Images*. Ouro Preto, Brazil.
- PRIMESENSE (2009). URL <http://www.primesense.com>.
- QUAM, L. H. (1984). Hierarchical warp stereo. In: *Image Understanding Workshop*. New Orleans.
- REHG, J. & KANADE, T. (1994). Visual tracking of high DOF articulated structures: an application to human hand tracking. In: *Third European Conference on Computer Vision*. Stockholm, Sweden.
- REHG, J. & WITKIN, A. (1981). Visual tracking with deformation models. Sacramento.
- REINHARD, E., WARD, G., PATTANAİK, S. & DEBEVEC, P. (2005). *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann Publishers.
- RODGERS, J., ANGUELOV, D., PANG, H.-C. & KOLLER, D. (2006). Object pose detection in range scan data. In: *Proceedings of CVPR*.
- SHASHUA, A. & TOELG, S. (1997). The quadric reference surface: Theory and applications. *International Journal of Computer Vision* **23**(2), 185–198.

- SHASHUA, A. & WEXLER, Y. (2001). Q-warping: Direct computation of quadratic reference surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(8), 920–925.
- SHI, J. & TOMASI, C. (1994). Good features to track. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Seattle.
- SHOTTON, J., FITZGIBBON, A., COOK, M., SHARP, T., FINOCCHIO, M., MOORE, R., KIPMAN, A. & BLAKE, A. (2011). Real-time human pose recognition in parts from single depth images. *CVPR*.
- SIGAL, L., ISARD, M., SIGELMAN, B. H. & BLACK, M. J. (2003). Attractive people: Assembling loose-limbed models using non-parametric belief propagation. In: *Neural Information Processing Systems (NIPS)*. Department of Computer Science of Brown University (Providence, RI) and Microsoft Research Silicon Valley (Mountain View, CA).
- SRINIVASAN, S., CHELLAPPA, R., VEERARAGHAVAN, A. & AGGARWAL, G. (2005). *Handbook of Image and Video Processing*, chap. 3.13 - Electronic image stabilization and mosaicking algorithms. Academic Press, 2nd ed., pp. 309–322.
- SUMA, E. A., LANGE, B., RIZZO, A. S., KRUM, D. M. & BOLAS, M. (2011). FFAST: The flexible action and articulated skeleton toolkit. *IEEE Virtual Reality*, 245–246.
- SZELISKI, R. (2011). *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer, 1st ed.
- SZELISKI, R. & COUGHLAN, J. (1997). Spline-based image registration. *International Journal of Computer Vision* **22**(3), 199–218.
- SZELISKI, R. & LAVALLÉE, S. (1996). Matching 3-D anatomical surfaces with non-rigid deformations using octree-splines. *International Journal of Computer Vision* **18**(2), 171–186.
- SÁ, A., CARVALHO, P. C. & VELHO, L. (2002). (b, s)-BCSL: Structured light color boundary coding for 3D photography. In: *Proceedings of 7th International Fall Workshop on Vision, Modeling, and Visualization*.
- THEOBALT, C., DE AGUILAR, E., MAGNOR, M., THEISEL, H. & SEIDEL, H.-P. (2004). Marker-free kinematic skeleton estimation from sequences of volume data. In: *Proceedings of ACM VRST*.
- TORR, P. H. S. & ZISSERMAN, A. (1999). Feature based methods for structure and motion estimation. In: *International Workshop on Vision Algorithms* (TRIGGS, W., ZISSERMAN, A. & SZELISKI, R., eds.).
- TRON, R. & VIDAL, R. (2007). A benchmark for the comparison of 3-D motion segmentation algorithms. In: *IEEE International Conference on Computer Vision and Pattern Recognition*.
- TRUCCO, E. & VERRI, A. (1998). *Introductory Techniques for 3-D Computer Vision*. Upper Saddle River, NJ: Prentice Hall.
- VELHO, L., VIEIRA, M. B., LUZ, J. L. S., SÁ, A. & CARVALHO, P. C. (2004). Real time 3D video. URL <http://w3.impa.br/~lvelho/v3d-s04/3dv-poster.pdf>.
- VIALUX (2009). URL <http://www.vialux.de>.
- VIEIRA, M. B., VELHO, L., SÁ, A. & CARVALHO, P. C. (2005). A camera-projector system for real-time 3D video. In: *Proceedings of IEEE International Workshop on Projector-Camera Systems (PROCAMS)*. San Diego, California, USA. Em conjunto com CVPR 2005.
- WAND, M., ADAMS, B., OVSIANIKOV, M., BERNER, A., BOKELOH, M., JENKE, P., GUIBAS, L., SEIDEL, H.-P. & SCHILLING, A. (2009). Efficient reconstruction of nonrigid shape and motion from real-time 3D scanner data. *ACM Transactions on Graphics* **28**(2), 1–15. ISSN: 0730-0301.

-
- WAND, M., JENKE, P., HUANG, Q., BOKELOH, M., GUIBAS, L. & SCHILLING, A. (2007). Reconstruction of deforming geometry from time-varying point clouds. In: *Proceedings of Eurographics Symposium on Geometry Processing*.
- WEDEL, A., CREMERS, D., POCK, T. & BISCHOF, H. (2009). Structure- and motion-adaptive regularization for high accuracy optic flow. In: *International Conference on Computer Vision*. Kyoto, Japan.
- WYSZECKI, G. & STILES, W. S. (2000). *Color Science: Concepts and Methods, Quantitative Data and Formulae*. New York: John Wiley & Sons, 2nd ed.
- YU, S. X., GROSS, R. & SHI, J. (2002). Concurrent object recognition and segmentation by graph partitioning. In: *Neural Information Processing Systems (NIPS)*. Robotics Institute of Carnegie Mellon University and Center for the Neural Basis of Cognition (Pittsburg, PA).
- ZHAO, X. & LIU, Y. (2008). Generative tracking of 3D human motion by hierarchical annealed genetic algorithm. *Pattern Recognition* **41**, 2470-2483.