UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"
Pós-Graduação em Ciência da Computação

Tiago De Gaspari

Desenvolvimento de um método semiautomático para geração de *ground truths* de vídeos

UNESP

2015

Gaspari, Tiago De.
Desenvolvimento de um método semiautomático para geração de ground truths de vídeos / Tiago De Gaspari. -- São José do Rio Preto, 2015
78 f.: il., tabs.

Orientador: Antonio Carlos Sementille
Coorientador: Silvio Ricardo Rodrigues Sanches
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação. 2. Processamento de imagens – Técnicas digitais. 3. Vídeo digital. 4. Algoritmos de computador. 5. Visão por computador. I. Sementille, Antonio Carlos. II. Sanches, Silvio Ricardo Rodrigues. III. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. IV. Título.
CDU – 518.72:76

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Tiago De Gaspari

Desenvolvimento de um método semiautomático para geração de *ground truths* de vídeos

Orientador: Prof. Dr. Antonio Carlos Sementille

Coorientador: Prof. Dr. Silvio Ricardo Rodrigues Sanches

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Computação Aplicada, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

UNESP

2015

Tiago De Gaspari

Desenvolvimento de um método semiautomático para geração de *ground truths* de vídeos

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de Bauru.

Comissão Examinadora

Prof. Dr. Antonio Carlos Sementille
UNESP – Bauru
Orientador

Prof. Dr. João Paulo Papa
UNESP – Bauru

Prof. Dr. Valdinei Freire da Silva
USP – São Paulo

Bauru
13 de agosto de 2015

Aos meus pais João De Gaspari e Maria Conceição Mello De Gaspari, pelo amor e confiança.

À minha namorada, Paula, por estar sempre ao meu lado, pelo apoio, pelo incentivo e por não me deixar desistir dos meus sonhos.

Agradecimentos

Aos meus pais e minha família pela confiança no meu trabalho, mesmo não entendendo muito bem o que é uma pós-graduação e no que eu estava trabalhando.

À minha namorada, Paula Nascimento Antonio, por ser uma pessoa maravilhosa, por não me deixar desistir nos momentos difíceis, por me convencer que tudo iria dar certo e principalmente por acreditar e confiar em mim. Por me ouvir falar sobre meus problemas, e mesmo muitas vezes sem entender muito sobre o assunto, me ouvir e me ajudar com ideias para resolver eles.

Ao meu orientador, Antonio Carlos Sementille, pelos conselhos não só para o mestrado, mas para a vida, e por nos últimos meses se tornar mais que um orientador, um amigo.

Ao meu coorientador, Silvio Ricardo Rodrigues Sanches, pelas conversas durante o desenvolvimento do projeto e por tirar tantas dúvidas que surgiram ao decorrer do mesmo.

Aos meus amigos Jean, Gabriel, Ricardo e Silas pela força e amizade durante essa trajetória.

Aos colegas do laboratório SACI: Rodrigo, Ivan, Daniel, Everton e Carlos pela companhia e pela ajuda quando eu precisava resolver algum problema.

Ao CNPq pelo auxílio financeiro.

Sumário

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ASCII	*American Standard Code for Information Interchange*
GIMP	*GNU Image Manipulation Program*
HDMI	*High-Definition Multimedia Interface*
KDU	*Kinect Disparity Unit*
LDA	Linhas Divisoras de Águas
OpenCV	*Open Source Computer Vision Library*
RAM	*Random Access Memory*
RGB	*Red Green and Blue*
RGB-D	*Red Green Blue and Depth*
TOF	*Time-of-flight*
XML	*eXtensible Markup Language*
YAML	*YAML Ain't Markup Language*

Resumo

Vários algoritmos de segmentação de vídeo em duas camadas, para a extração de elementos de interesse em primeiro plano (normalmente pessoas) em ambientes não controlados, vem sendo propostos, para diversas aplicações como sistemas de Realidade Aumentada, *video chats*, ou para a compressão de vídeos. Para analisar a qualidade dos vídeos gerados pelos algoritmos de segmentação, diversos métodos os comparam com seus respectivos *ground truths*, que consistem em referências da melhor segmentação possível de um vídeo. Muitas vezes esse *ground truth* é obtido de forma manual, ou seja, o usuário pode ter que segmentar cada quadro (*frame*) do vídeo. Naturalmente este processo é trabalhoso, demorado e muitas vezes não é realizado para todos os quadros que constituem o vídeo. Devido a isto, também, muitas vezes o *ground truth* possui baixa resolução e curta duração. Estes aspectos podem constituir um problema quanto à eficácia da utilização do próprio *ground truth* no processo de avaliação da qualidade da segmentação. Neste contexto, o presente projeto teve como principal enfoque o desenvolvimento de um método semiautomático para a geração de *ground truths* de vídeos, utilizando informações de profundidade, visando a minimização da interação do usuário, o processo de implementação deste método na forma de uma ferramenta que combina etapas automáticas e interativas, assim como os resultados obtidos são comparados a outros trabalhos presentes na literatura.

Palavras-chaves: *Ground Truths* de vídeos, Segmentação de vídeo, Sensores de profundidade.

Abstract

Several bilayer video segmentation algorithms, for the extraction of elements of interest in the foreground (usually people) in uncontrolled environments, have been proposed for various applications such as Augmented Reality systems, video chats or for video compression. To analyze the quality of the videos generated by the segmentation algorithms, different methods compare them with their ground truths, which consist of references of the best possible segmentation of a video. Usually this ground truth is obtained manually, so the user may have to segment each frame of the video. Of course this process is laborious, time consuming and frequently not performed for all frames of the video. Because of this, the ground truth, usually, has low resolution and short duration. These aspects can be an issue to the effectiveness of using these ground truths in the segmentation quality evaluation process. In this context, this project had as its main focus the development of a semi-automatic method for the generation of ground truths of videos, using depth information, in order to minimize the user interaction, the implementation process of this method as a tool that combines automatic and interactive steps, and the results, that are compared to other studies in literature.

Keywords: Video ground truth, Video segmentation, Depth sensor.

1 Introdução

A segmentação de vídeos, com o intuito da extração de um objeto de interesse para substituição do fundo de uma cena, é um processo amplamente utilizado pela indústria cinematográfica e pela televisão por meio de algoritmos de segmentação mais tradicionais (FOSTER, 2010). Estes algoritmos fazem uso de vídeos que são obtidos em ambientes controlados, com iluminação constante e com fundos de cor única (normalmente azul ou verde), chamada de cor chave ou *chroma key*. Simplificando, estes algoritmos isolam o elemento de interesse a partir da eliminação da cor chave de fundo do vídeo original.

No entanto, diversos algoritmos têm sido propostos objetivando a segmentação de vídeos capturados em ambientes não controlados, ou seja, ambientes cujos planos de fundo são aleatórios e sem controle de iluminação. Várias destas segmentações visam, também, a substituição de fundo, como é o caso dos trabalhos de Criminisi et al. (2006), Yin et al. (2011), Stauffer e Grimson (2000).

O interesse no desenvolvimento de algoritmos de segmentação de vídeos com plano de fundos não controlados, deve-se, em grande parte, às aplicações em que os elementos de interesse são pessoas posicionadas em primeiro plano na cena. Como exemplos, podem ser citados os sistemas de Realidade Aumentada (SANCHES et al., 2012b) e sistemas de videoconferências ou *video chats* (CRIMINISI et al., 2006; PAROLIN et al., 2011; YIN et al., 2011). Este tipo de segmentação pode ser realizada para se obter privacidade (uma vez que o plano de fundo do vídeo será alterado), para reduzir-se a banda necessária para o envio e recebimento destes vídeos, ou até mesmo para a comprimi-los.

O fato destes algoritmos funcionarem em ambientes não controlados e com captura realizada por vídeo monocular (sem o auxílio de sensores ou câmeras adicionais calibradas), faz com que diversos fatores e situações no próprio ambiente ocasionem problemas no processo de segmentação (SANCHES et al., 2012a). Utilizar sensores que capturam informações adicionais sobre as imagens, como a distância de cada objeto presente na cena, em relação à câmera, vem se tornando cada vez mais popular nos últimos anos, como o caso do Kinect da Microsoft (ZHANG, 2012). Essas informações adicionais podem ser utilizadas em algoritmos de segmentação de imagem e vídeo, sendo uma alternativa ao uso, apenas, das imagens de câmeras convencionais (WANG et al., 2010; WU; BOULANGER; BISCHOF, 2008). Porém, mesmo nos algoritmos que fazem uso destas informações, os resultados da segmentação podem não atingir a qualidade necessária à aplicação.

Para avaliar a qualidade da segmentação de vídeos com fundo não controlado, alguns trabalhos simplesmente contam os pixels classificados de forma incorreta (YIN et al., 2011), enquanto outros consideram a forma em que o erro se apresenta na imagem, uti-

lizando abordagens objetivas (CORREIA; PEREIRA, 2003; CAVALLARO; GELASCA; EBRAHIMI, 2002). Alguns trabalhos, ainda, propõem abordagens subjetivas (GELASCA; EBRAHIMI, 2009; SANCHES et al., 2012c; SANCHES, 2013). Nestes métodos, são propostas métricas que analisam os diversos tipos de erros gerados pelos diferentes algoritmos de segmentação. Esses erros podem ser divididos em diversos tipos, levando em consideração alguns fatores, como o seu posicionamento em cada quadro do vídeo analisado. A identificação de cada erro é feita ao comparar os quadros de um vídeo segmentado e o seu respectivo *ground truth*.

O *ground truth*, portanto, pode ser entendido como sendo a referência de melhor segmentação possível em um vídeo. Muitas vezes essa segmentação é obtida manualmente, ou seja, seleciona-se em cada quadro de um vídeo, de forma manual, os pixels que pertencem ao objeto de interesse em primeiro plano e os pixels que são pertencentes ao fundo. O processo para a obtenção do *ground truth* de vídeos, atualmente, não pode ser realizado de forma totalmente automática. Uma vez que um *ground truth* representa uma segmentação sem erros, o algoritmo deve realizar uma segmentação perfeita, o que não acontece. Diversos fatores, como os ambientes não controlados, a calibração de equipamentos específicos para a captura do vídeo, ou até mesmo a busca pela excelência na segmentação de um *ground truth*, fazem com que a geração dessa segmentação se torne um desafio que ainda não foi superado totalmente e torne possível pesquisas nessa área.

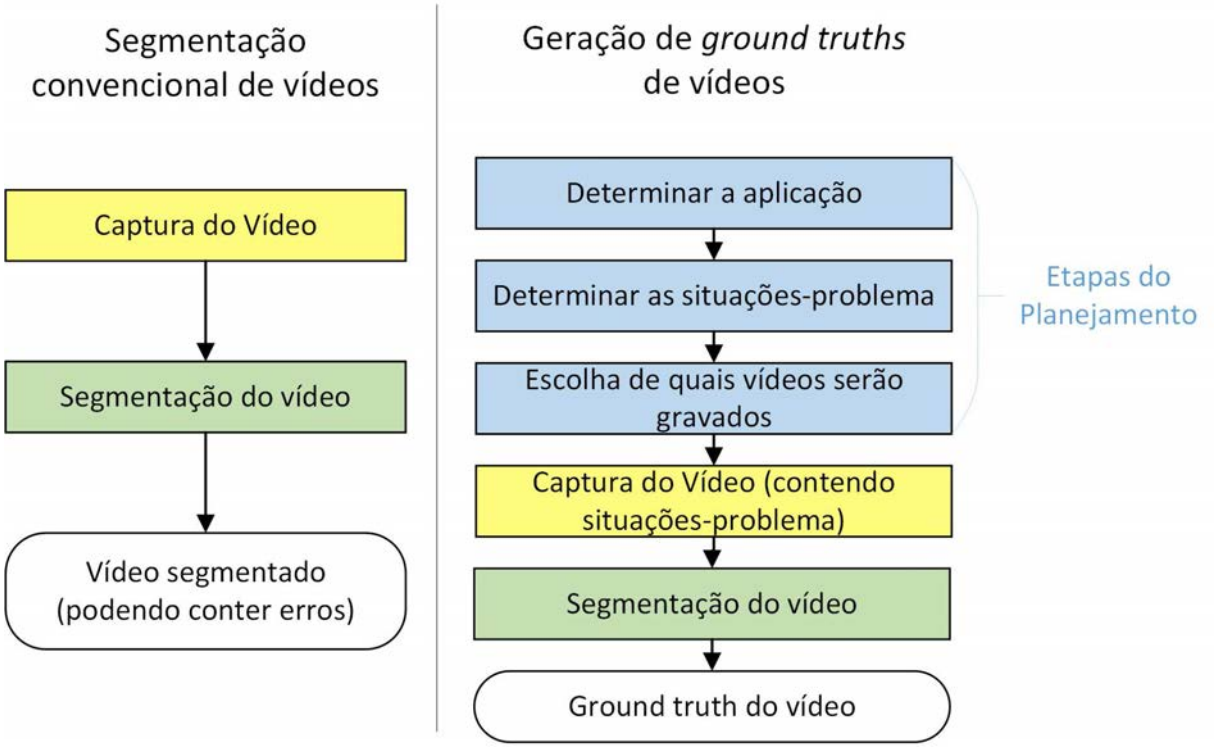
Além de ser a melhor segmentação de um vídeo, o *ground truth* representa um vídeo que pode ser utilizado para validar algoritmos de segmentação em duas camadas. O processo de geração de um *ground truth*, neste caso, é diferente e mais complexo do que uma simples segmentação de vídeo. Para a geração de *ground truths* é necessário planejar desde quais vídeos serão gravados, como a segmentação deverá ser feita e quais situações devem ser consideradas para que o *ground truth* seja útil no processo de validação de qualidade de vídeos.

Na Figura 1 são apresentadas, de forma simplificada, quais as etapas consideradas necessárias no processo de geração de *ground truths*.

As etapas destacadas em azul na Figura 1, representam as etapas que constituem o planejamento das gravações. É importante ressaltar que ao contrário da segmentação de vídeos, onde essa etapa não existe, para a geração de *ground truths* ela é importante, para que a segmentação gerada no processo possa ser utilizada para realizar a validação de outros algoritmos.

A etapa de captura dos vídeos, destacada em amarelo na Figura 1, não possui grandes diferenças entre os dois processos. O único ponto que possui diferença é a forma como as situações-problema devem ser simuladas no momento da gravação do vídeo, o que não é importante no caso de segmentações que não são voltadas à geração de ground truths.

Figura 1 – Etapas necessárias para segmentação de vídeos em duas camadas e para geração de *ground truths* de vídeos



Fonte: elaborada pelo autor.

Por fim, tem-se a etapa de segmentação, destacada em verde, onde os vídeos são segmentados por diferentes algoritmos de segmentação de duas camadas. É importante salientar que no caso da geração do *ground truth*, o resultado não pode conter erros, necessitando para tal, de uma etapa de refinamento da segmentação onde esses são eliminados.

O processo de segmentação de vídeos pode ser comparado ao processo de segmentação de imagens estáticas com objetivo de extração de *foreground*. Métodos interativos de segmentação de imagens estáticas conseguem ótimos resultados, porém precisam de diversas interações do usuário, devido à falta de informação sobre o que deve ser considerado como *foreground* na imagem. Os resultados obtidos por esses algoritmos poderiam ser utilizados como parte do processo de geração de *ground truths* de vídeos, caso não apresentassem erros de classificação de pixel. Para resolver o problema de segmentação de imagens, um grande número de interações, no decorrer do processo, podem não ser um grande problema, porém, caso este tipo de método seja aplicado para resolver o problema de geração de *ground truths* de vídeos, o número de interações necessárias pode tornar-se muito grande, dependendo da quantidade de quadros que constituem os vídeos.

Diversos métodos de segmentação que usam informações de profundidade, além das imagens RGB (*Red*, *Green*, *Blue*) capturadas pelas câmeras convencionais, tem sido propostos ultimamente (WANG et al., 2010; WU; BOULANGER; BISCHOF, 2008). Uma

grande desvantagem da maioria dos equipamentos existentes, independente da tecnologia que é utilizada para a obtenção deste tipo de informação, é a qualidade das imagens RGB geradas por eles, uma vez que o foco destes dispositivos é gerar um mapa de profundidade de qualidade e não imagens RGB. Aplicações onde a interação do usuário é realizada por gestos, identificados por esses dispositivos, ou que não exibem as imagens RGB aos usuários, não necessitam que estas possuam alta resolução, tampouco alta qualidade. A junção deste tipo de sensores às câmeras de vídeo de maior resolução permite que aplicações possam se beneficiar do mapa de profundidade obtido pelos sensores, assim como uma alta qualidade de imagem. A união das informações de câmeras convencionais e destes equipamentos mostrou-se uma alternativa interessante no processo de segmentação para geração de *ground truths* de vídeos.

1.1 Objetivos

Considerando o exposto anteriormente, são objetivos deste trabalho:

- A estruturação de um método para geração de *ground truths* de vídeos em diversas resoluções e sem perda de quadros, utilizando-se, para isto, informações sobre a profundidade da cena. O método, também, deve ser capaz de reduzir a quantidade de interações manuais do usuário, viabilizando a geração de *ground truths* de vídeos de qualquer duração;
- A implementação do método desenvolvido, para fins de teste e avaliação da solução adotada.

1.2 Organização do documento

Este documento está organizado nos seguintes capítulos:

No Capítulo 2 são apresentadas as principais definições e métodos que envolvem a segmentação de imagens e vídeos, especialmente os que se aplicam à segmentação em duas camadas, bem como os principais algoritmos encontrados na literatura.

No Capítulo 3 tem-se uma discussão sobre as dificuldades, os desafios e a motivação para a geração de *ground truths* em vídeos. Além disso, são descritos alguns algoritmos, considerados importantes, de geração de *ground truths* em vídeos, assim como métodos de segmentação de vídeo baseados em informações adicionais.

No Capítulo 4 são descritos os principais métodos de obtenção de informações de profundidade, além da apresentação de um método para calibrar sensores de profundidade, câmeras de vídeo convencionais e unir as informações provenientes desses tipos de equipamentos.

No Capítulo 5 é descrito o método proposto para a geração de *ground truths* de vídeos.

No Capítulo 6 os detalhes referentes à implementação, as dificuldades encontradas e as soluções empregadas em cada uma das etapas implementadas, do método, são descritos.

O Capítulo 7 apresenta os detalhes de implementação do método, bem como as soluções e dificuldades encontradas em cada etapa de seu desenvolvimento.

Por fim, no Capítulo 8, são apresentadas as conclusões do trabalho, assim como seus desdobramentos, no que diz respeito à trabalhos futuros.

2 Segmentação de imagens em duas camadas

Neste capítulo são apresentadas as principais definições e métodos envolvidos no processo de segmentação de imagens em duas camadas, visando a extração de um objeto de interesse em primeiro plano para uma posterior substituição de sua imagem de fundo.

2.1 Processo de *Matting e Trimap*

Uma das formas de extração de um objeto em primeiro plano (*foreground*), de uma imagem digital, para a substituição do plano de fundo original (*background*), é conhecido como *matting*. O processo de *matting* consiste em determinar se as informações de cor contidas em cada pixel, pertencem ao *foreground*, ao *background*, ou em alguns casos, combinando as cores destes dois planos (WANG; COHEN, 2008). O *matting* foi descrito matematicamente por Porter e Duff (1984), assim como o canal alfa que controla o grau de transparência para cada pixel da imagem. Uma imagem $I_z(z = (x, y))$ é calculada como uma combinação entre o *Foreground* F_z e o *Background* B_z usando o mapa de transparências α_z, como apresentado na Equação 2.1

$$I_z = \alpha_z F_z + (1 - \alpha_z) B_z \tag{2.1}$$

onde α_z pode ser qualquer valor real em $[0, 1]$. Se α_z assumir os valores 1 ou 0, pode-se classificar o pixel z efetivamente como *foreground* ou *background*, caso contrário, este pixel é constituído das cores das duas regiões. Isso ocorre nas áreas próximas à borda destes dois planos, ou quando o *foreground* possui partes com algum grau de transparência, como vidro, plástico, ou até mesmo fumaça.

Dada uma simples imagem digital, os valores necessários para o processo de *matting*, ou seja, o mapa de transparência (*alpha map*), não estão disponíveis. Uma vez que estes dados são estimados corretamente, a troca do *background* da imagem original pode ser realizada simplesmente trocando o *background* original B por um novo: B' na Equação 2.1.

Para a produção de uma nova imagem utilizando algumas das técnicas de *matting* (CHUANG et al., 2001; SUN et al., 2004; WANG; COHEN, 2007; BAI; SAPIRO, 2009; HUANG; WANG, 2011), onde é produzido o *alpha map*, são necessárias duas informações: uma imagem digital, a qual pertence o *foreground* que se deseja extrair e uma estrutura de dados chamada *trimap*.

O *trimap* consiste em uma imagem com a mesma resolução em pixels da imagem que se deseja realizar o *matting*, porém, cada pixel é classificado em três regiões: o *foreground*, o *background*, e uma região chamada de área desconhecida. Esta área normalmente é localizada entre o *foreground* e o *background*, onde ainda não está determinado a qual região o pixel pertence. Essa área desconhecida deve ser, portanto, analisada e um valor de transparência deve ser atribuído a cada pixel, gerando-se o *alpha map*. Assim, o processo de *matting* será capaz de substituir o plano de fundo da imagem original.

Este *trimap* pode ser obtido de duas maneiras: manualmente, onde o usuário determina quais são as três regiões na imagem inicial ou através de algoritmos de segmentação de imagens, mais precisamente algoritmos de extração de *foreground*. A Figura 2 apresenta um exemplo de processo de *matting* no qual o usuário seleciona manualmente as regiões do *trimap*.

Figura 2 – Processo de *matting* e *compositing*. (a) Imagem original; (b) *Trimap* definido manualmente; (c) Máscara extraída através do *trimap*; (d) *Foreground* estimado a partir do *trimap*; (e) Uma nova imagem gerada através do processo de *compositing* com um novo *background*. Os resultados foram obtidos ao utilizar o algoritmo de *Robust Matting*

(a) (b) (c) (d) (e)

Fonte: Wang e Cohen (2007).

A segmentação de imagens, ou vídeos, em duas camadas, chamada também de segmentação *bilayer*, pode ser entendida como um caso específico de *matting*, onde o α_z assume valores, exclusivamente, de 0 ou 1, ou seja, cada pixel é classificado como totalmente pertencente ao *background* ou *foreground*, dividindo a imagem nestas duas camadas.

2.2 Segmentação de imagens em duas camadas

Uma vez que a imagem da qual se deseja extrair o *foreground*, ou até mesmo definir um *trimap*, não possui informações sobre onde está, ou o que é o objeto de interesse, nem quais áreas são pertencentes ao plano de fundo, foram propostos algoritmos de segmentação de imagens visando a extração de um objeto de interesse.

A maioria dos algoritmos para extração de um objeto de interesse são interativos, ou seja, necessitam intervenção do usuário. Neste tipo de algoritmo, é preciso que o usuário selecione na imagem, quais são as regiões que são pertencentes ao *foreground* e ao *background*. Em alguns casos, também é possível selecionar até informações mais específicas, como a área desconhecida ou, então, quais áreas são provavelmente pertencentes à uma dessas regiões.

A partir de pequenas marcações feitas pelo usuário por meio de dispositivos como *mouse*, os algoritmos podem levar em consideração diversas características da imagem, com a finalidade de estimarem o *trimap*. A seguir são apresentados alguns métodos utilizados para realizar esta segmentação. É importante frisar que alguns destes métodos obtêm um *trimap* como resultado, enquanto outros já conseguem obter um mapa de transparência completo da imagem.

2.2.1 Baseada em Contraste de bordas - *Intelligent Scissors*

Diversos métodos realizam a segmentação de imagens analisando os contrastes nas bordas. Um destes métodos é o *Intelligent Scissors* (MORTENSEN; BARRETT, 1995), também conhecido como *Magnetic Lasso* (ADOBE SYSTEM INCORPORATED, 2013), o qual permite ao usuário encontrar um contorno com o menor custo, apenas traçando com o mouse, sem muito cuidado, onde pontos chaves são encontrados e o caminho entre esses pontos é traçado para a segmentação. Estes caminhos são obtidos através da análise das bordas existentes entre estes dois pontos, no caso, as bordas do objeto de interesse. Caso os caminhos entre os pontos calculados não sejam perfeitos, ou seja, não obtenham a correta segmentação, o usuário pode selecionar manualmente pontos-chaves entre estes caminhos.

A maior limitação deste algoritmo se dá em casos em que não há um contraste tão grande entre o objeto de interesse e o restante da imagem, ou seja, quando não há uma borda bem definida. Regiões muito texturizadas, ou então que não possuem um contraste significativo nas bordas do objeto de interesse, fazem com que entre dois pontos-chaves, vários caminhos sejam possíveis. Neste caso, é necessário que um ou mais pontos sejam selecionados manualmente, aumentando a interação com o usuário, o que torna o processo de segmentação mais demorado e trabalhoso para o usuário. A Figura 3 apresenta os resultados obtidos após a segmentação utilizando o *Magnetic Lasso* do Adobe Photoshop 2013, utilizando apenas os pontos obtidos com o algoritmo e utilizando alguns pontos selecionados manualmente pelo usuário. Nesta figura, os pontos em amarelo representam pontos-chaves selecionados pelo usuário, enquanto os pontos brancos são gerados automaticamente pelo algoritmo enquanto o usuário passa o cursor do *mouse* em volta do objeto de interesse.

É possível visualizar na Figura 3 que a partir do momento em que existam áreas

Figura 3 – Segmentação através do *Magnetic Lasso* (Adobe Photoshop 2013). Acima: Marcações realizadas apenas pelo algoritmo e resultado obtido; Abaixo: Marcações realizadas pelo algoritmo e selecionadas manualmente pelo usuário e resultado obtido.

Fonte: elaborada pelo autor.

com maiores detalhes (região esquerda da imagem original), é preciso que o usuário selecione manualmente os pontos-chaves do objeto de interesse, para que a segmentação seja executada corretamente. Além disso, é preciso observar que este método realiza uma segmentação binária, ou seja, não gera um *trimap* nem um mapa de transparências: apenas separa o *foreground* do *background*, gerando imagens com serrilhados aparentes na borda dessas regiões.

2.2.2 Baseada em Informações de Cores - *Magic Wand*

O método conhecido como *Magic Wand* (ADOBE SYSTEM INCORPORATED, 2013) ou varinha mágica (em tradução livre), é comum em vários *softwares* de edição de imagem comerciais, como o Adobe Photoshop e o GIMP. O método precisa que o usuário especifique uma ou mais regiões do objeto de interesse, para que assim, a partir das

informações de intensidade de cor dos pixels selecionados, baseado em um limiar, agrupe os pixels vizinhos com cores semelhantes em uma região maior do que a selecionada pelo usuário. Uma vez que o objeto de interesse possua diferentes regiões com diferentes intensidades de cores, é preciso que o usuário selecione várias regiões para a segmentação.

A Figura 4 apresenta o exemplo de uma segmentação realizada utilizando a ferramenta de *Magic Wand* presente no Adobe Photoshop 2013. As marcações foram realizadas por meio de cliques sobre a imagem original e estão representados por círculos pretos.

Figura 4 – Segmentação através do *Magic Wand* (Adobe Photoshop 2013). (a) Marcações realizadas pelo usuário, representadas por círculos pretos e (b) Resultado da segmentação.

(a) (b)

Fonte: elaborada pelo autor.

Como apresentado na Figura 4, o método de *Magic Wand* permite a segmentação interativa e iterativa: à medida que o usuário seleciona mais pontos na imagem original, mais áreas são estimadas como *foreground*. Entretanto, como é um método baseado na comparação de pixels vizinhos, regiões do *foreground*, que possuam cores semelhantes ao *background*, podem ser confundidas pelo algoritmo resultando em uma segmentação incorreta. Além disso, é possível observar que há um grande número de pontos selecionados pelo usuário, ou seja, é um processo que demanda muito tempo do usuário, para a escolha dos melhores pontos.

Como no método *Intelligent Scissors*, o *Magic Wand* não gera um *trimap* ou um mapa de transparências, apenas extrai o *foreground* selecionado pelo usuário, eliminando o *background* original.

2.2.3 Baseada em *Graph Cuts*

O método conhecido como *Interactive Graph Cuts* (BOYKOV; JOLLY, 2001) é um método de segmentação de imagens para extrair o objeto de interesse que utiliza o conceito de *graph cuts*.

Na teoria dos grafos, um *graph cut*, ou apenas *cut*, é uma divisão de um grafo em dois subgrafos, que pode ser realizada seguindo diversas metodologias. Um grafo não orientado $G = (V, A)$ é definido por um conjuntos de vértices V e arestas A que conectam estes vértices. Cada aresta $a \in A$ possui um peso (custo) não negativo p_a. Há também dois ou mais nós especiais, chamados de terminais, que representam como o grafo deve ser dividido. Um *graph cut* é um subconjunto de vértices $C \subset A$ que foram separados a partir dos terminais, no grafo induzido $G(C) = (V, A \backslash C)$. O custo, portanto, do *graph cut* é dado pela Equação 2.2

$$|C| = \sum_{a \in C} p_a \tag{2.2}$$

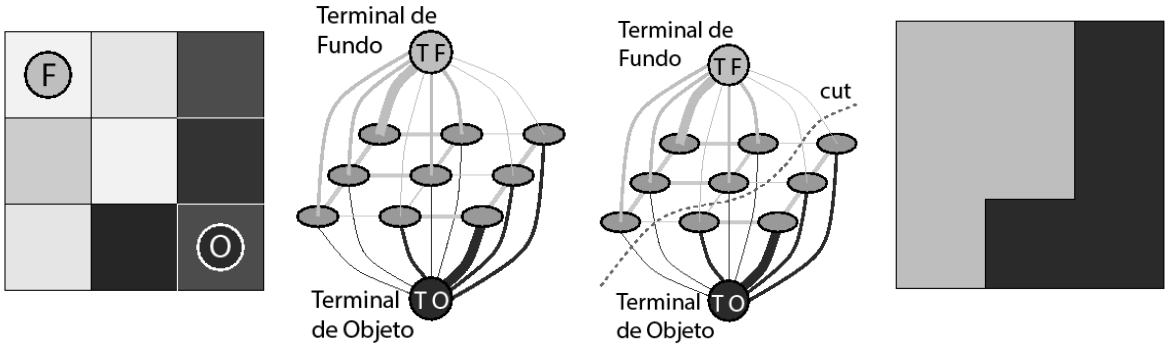
Como cada vértice do grafo pode ser representado por um pixel de uma imagem e as arestas podem representar qualquer relação entre o pixel e seus vizinhos, pode-se realizar a segmentação de imagens utilizando *graph cuts*. Os pesos de cada aresta são analisados e quando é obtido um custo mínimo para cada *graph cut*, a segmentação ótima é obtida, uma vez que a imagem é separada em duas subimagens, com base na característica que foram utilizadas como peso para cada aresta.

A Figura 5 representa um exemplo simples de segmentação de uma imagem de 3x3 pixels. Nos grafos, cada custo é representado pela espessura da aresta, portanto, arestas que possuem menores pesos (mais finas) são fortes candidatas à pertencerem ao corte do *graph cut*.

O trabalho original para a segmentação interativa utilizando *graph cuts* foi produzido por Boykov e Jolly (2001), mas foi sequencialmente estendido por outros pesquisadores para agregar diferentes funcionalidades e interfaces com o usuário, como é o caso do *GrabCut* (ROTHER; KOLMOGOROV; BLAKE, 2004). Uma das vantagens deste algoritmo é a fácil interação com o usuário. À princípio, para realizar uma segmentação, o usuário apenas necessita desenhar um retângulo sobre o objeto de interesse e o algoritmo já realiza a segmentação. A Figura 6 apresenta um exemplo do uso deste sistema para a extração de um objeto de interesse, no caso, uma pessoa.

Por se tratar de um método interativo, é possível que na primeira interação do usuário (desenhando o retângulo), a primeira execução do algoritmo não obtenha um resultado satisfatório, apresentando algumas áreas segmentadas incorretamente. A partir desta primeira segmentação, o algoritmo pode ser executado novamente, para o usuário selecione novamente áreas em que o algoritmo classificou incorretamente para que uma segunda segmentação seja realizada. A Figura 7 apresenta um processo de segmentação, no

Figura 5 – Exemplo de segmentação utilizando *graph cuts*. (a) Imagem de 3x3 pixels sendo um deles selecionado como F (fundo) e outro como O (objeto); (b) Grafo da imagem com os terminais selecionados de Fundo e de Objeto; (c) *Graph cut* passando pelas arestas menos representativas (mais finas); e (d) Resultado da segmentação.



(a) (b) (c) (d)

Fonte: Adaptado de (BOYKOV; JOLLY, 2001).

Figura 6 – Exemplo de aplicação do *GrabCut*. O usuário desenha um retângulo ao redor de um objeto, o qual é extraído da imagem inicial.

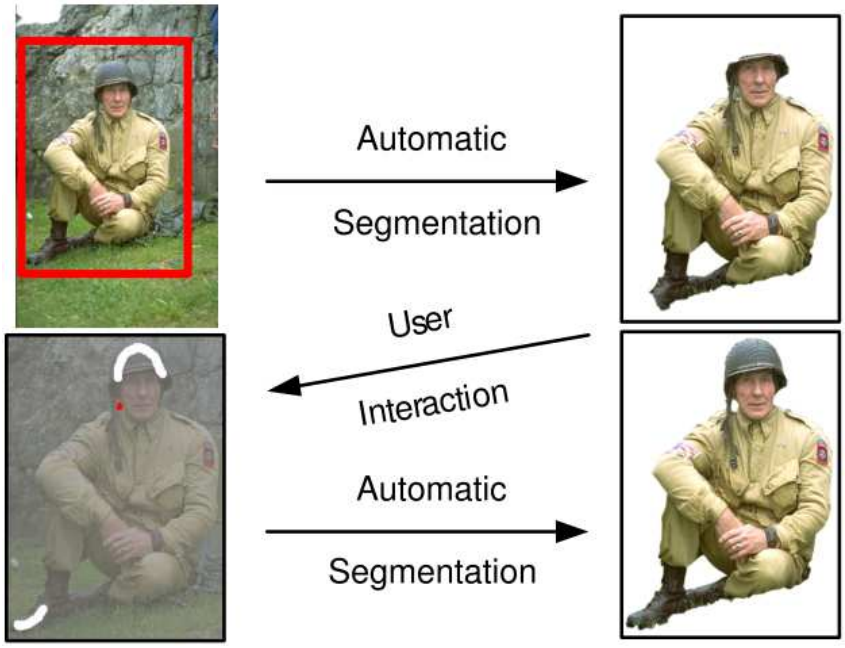
Fonte: Rother, Kolmogorov e Blake (2004).

qual o algoritmo necessita ser executado duas vezes, assim como as marcações realizadas pelo usuário nestas duas execuções.

Outra alternativa ao uso deste algoritmo, é a utilização de uma estrutura chamada de *quadrimap* como informação de entrada, diferentemente de seleções manuais pelo usuário. Este *quadrimap* se assemelha ao *trimap*, porém tem quatro regiões: *Foreground* e *Background* (como no *trimap*) e ao invés de uma região chamada de área desconhecida, possui uma divisão desta, em duas: provável *foreground* e provável *background*. A partir disto, o algoritmo pode dar um peso maior à essas duas regiões, em vez de analisar como apenas uma região desconhecida.

Como resultado desta segmentação, o algoritmo retorna um *quadrimap* com as áreas descritas anteriormente, agrupando as regiões através dos *graph cuts*. Com este

Figura 7 – Após a primeira execução, onde o objeto de interesse é selecionado por um retângulo, é preciso de outra execução do algoritmo, onde seleciona-se o *foreground* (em branco) e o *background* (em vermelho). A segunda execução é suficiente para uma segmentação correta.



Fonte: (ROTHER; KOLMOGOROV; BLAKE, 2004).

quadrimap, algoritmos de *matting* (CHUANG et al., 2001; SUN et al., 2004; WANG; COHEN, 2007; BAI; SAPIRO, 2009; HUANG; WANG, 2011) podem ser usados para a geração do *alpha map* completo, para um *matting* mais preciso.

2.2.4 *Watershed*

Watershed é uma técnica de segmentação de imagens, conhecido também como Linhas Divisoras de Água, ou LDA (VINCENT; SOILLE, 1991). A princípio, a imagem original, é transformada em uma estrutura de três dimensões, onde essa nova dimensão depende do valor do gradiente de cada pixel, ou seja, quanto maior o seu gradiente, maior será o valor atribuído à esta nova dimensão. O valor de gradiente de cada pixel de uma imagem é normalmente utilizado para a detecção de bordas da mesma. Matematicamente, o gradiente de uma imagem é dado pela Equação 2.3

$$\nabla f = \frac{\partial f}{\partial x}\hat{x} + \frac{\partial f}{\partial y}\hat{y} \tag{2.3}$$

onde $\frac{\partial f}{\partial x}$ é o gradiente na direção do eixo das abscissas (x) e $\frac{\partial f}{\partial y}$ é o gradiente na direção do eixo das ordenadas (y). A direção do gradiente pode ser calculada pela Equação 2.4

$$\theta = tan^{-1}(\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}) \tag{2.4}$$

A estrutura tridimensional, obtida dos gradientes de cada pixel, é representada topologicamente e analisada como uma bacia hidrográfica (motivo do nome do método), onde há vales (regiões mais baixas) e picos (regiões mais altas). A análise desta imagem topográfica é dividida em três regiões:

- Pontos pertencentes à uma região mínima local (vales);
- Pontos pertencentes à uma região máxima local (picos);
- Pontos intermediários: pontos que não pertencem à nenhum mínimo ou máximo local.

A partir destas marcações em cada ponto, as regiões das imagens são divididas como se cada vale na imagem fosse enchido com um líquido, até o limite de transbordar para outro vale. Cada vale, é enchido como se fossem pequenos lagos nesta bacia hidrográfica e representa uma região distinta. A imagem é segmentada por meio de cada região formada por esses vales, que se transformaram em lagos.

É importante observar que a segmentação de imagens utilizando *Watersheds* não se trata de um método de segmentação de imagens de duas camadas, ou seja, a imagem original não é dividida em apenas duas ou três regiões (*foreground*, *background* e área desconhecida), mas sim em várias pequenas regiões. Um dos problemas deste método é a utilização de imagens com muito ruído, texturizadas, ou irregulares, causando um efeito chamado de supersegmentação. A Figura 8 apresenta um exemplo de supersegmentação gerada pelo algoritmo de *Watershed*.

Figura 8 – Supersegmentação obtida com o algoritmo de *WaterShed*. (a) Imagem original e (b) imagem segmentada.

(a) (b)

Fonte: elaborada pelo autor.

Para contornar este problema e obter apenas o objeto de interesse da imagem, algoritmos interativos baseados em *Watershed*, ou que combinam a técnica com outros algoritmos como *graph cuts*, *optimal spanning forests* e *random walker* (COUPRIE et al., 2011) são utilizados.

3 Geração de *ground truths* de vídeos

Quando a segmentação em duas camadas é desejada em vídeos com fundos arbitrários e se utiliza apenas uma câmera, diversos algoritmos de segmentação em tempo real podem ser utilizados (CRIMINISI et al., 2006; STAUFFER; GRIMSON, 2000; YIN et al., 2011), porém, os resultados obtidos não conseguem atingir a mesma qualidade na segmentação se comparados aos algoritmos que utilizam vídeos capturados em ambientes controlados e/ou com equipamentos específicos. Diversos fatores contribuem para que os algoritmos não consigam obter um resultado com tanta acurácia, tais como: variações na iluminação, movimentação no fundo, oscilações no dispositivo de captura e cores semelhantes no fundo e no objeto de interesse (SANCHES et al., 2012a). Estas falhas de segmentação são divididos em diversos tipos, de acordo com sua localização na imagem, e são analisados de várias maneiras.

No trabalho de Gelasca (2005), por exemplo, esses erros foram divididos em seis tipos. Exemplos destes são apresentados na Figura 9, assim como o *ground truth*, que não apresenta nenhum erro.

Figura 9 – *Ground truth* e erros comuns, presentes em vídeos segmentados: região adicionada, dois exemplos de buracos na borda do *foreground*, buracos internos, fundo adicionado e o *flickering* (qualquer variação espacial do mesmo erro, em quadros consecutivos no vídeo).

Fonte: Gelasca (2005).

O produto final de um *ground truth* de um vídeo constitui-se de *trimaps* de todos os seus quadros, que representam a melhor segmentação possível daquela imagem. É importante frisar que a melhor estrutura de dados para a substituição do fundo de um vídeo é o mapa de transparência, para efeitos de *anti-aliasing* (GONZALEZ; WOODS, 2002). Por outro lado, utilizar o *trimap* permite a identificação e localização dos erros gerados com os algoritmos de segmentação de vídeo, uma vez que as regiões de *foreground*, *background* e área desconhecida estão bem demarcadas.

O *ground truth* de um vídeo (ou um conjunto deles), deve possuir certas características, dependendo de sua aplicação, para que possa ser utilizado da melhor maneira

possível em um processo de validação. As características, necessárias, são:

- *Simular situações-problema:* É importante que os vídeos que serão analisados contenham situações que possam causar problemas em algoritmos de segmentação de duas camadas, para verificar se estes os levam em consideração e se conseguem contorná-los. Exemplos dessas situações podem ser: movimentação no plano de fundo, alteração na iluminação e cores semelhantes entre o *foreground* e *background*;

- *Vídeos em diferentes resoluções:* Diferentes resoluções podem se tornar um problema ou melhorar o desempenho de determinados algoritmos, portanto, vídeos com diferentes resoluções, são interessantes para uma melhor validação dos algoritmos;

- *Vídeos em diferentes durações:* Caso a aplicação permita, vídeos com diferentes durações são interessantes para comparar diferentes algoritmos de segmentação. Algoritmos que se baseiam nos quadros anteriores para realizar a segmentação dos seguintes podem ter um desempenho melhor com vídeos mais longos.

- *Vídeos capturados com diferentes dispositivos:* Em aplicações nas quais diferentes dispositivos possam ser utilizados, é interessante que os *ground truths* sejam referentes a vídeos provenientes de diferentes câmeras. Com isso, é possível analisar vídeos com diferentes especificidades que podem interferir no desempenho dos algoritmos. Diferentes campos de visão (*field of view)* ou profundidade de campo (*depth of field*) alteram as imagens e podem ser tratadas de forma diferenciada.

Uma das tarefas mais importantes no processo de geração de *ground truths* consiste na segmentação de cada quadro do vídeo. Esse processo pode ser feito manualmente, ou seja, utilizando alguma ferramenta de edição de imagens para identificar em cada quadro do vídeo, quais pixels são pertencentes ao *foreground*, quais pertencem ao *background* e quais pertencem à área desconhecida. Esta tarefa é bastante trabalhosa e demorada, principalmente levando em conta o grande número de quadros de um vídeo (normalmente 30 por segundo, ou mais) e a dificuldade em, dado um quadro do vídeo, identificar corretamente a região correspondente de cada um dos pixels da imagem. Como estatisticamente, a análise dos vídeos para o desenvolvimento de uma métrica precisa de várias amostras, uma vez que cada vídeo possui erros diferentes, o trabalho é ainda maior ao realizar o processo manualmente.

Algumas bases de vídeos disponibilizadas por grupos de pesquisa (CRIMINISI et al., 2006; WANG et al., 2010) possuem *ground truths* de apenas alguns quadros do vídeo, por exemplo, a cada 5 ou 10 quadros, o que diminui a precisão na contagem dos erros de classificação. Além disso, uma forma de identificar a melhor segmentação quadro a quadro pode facilitar a identificação de erros temporais produzidos por algoritmos.

Além da possibilidade de segmentação manual, existem alguns métodos encontrados na literatura para a geração automática e semiautomática de *ground truths*. As características destes métodos, assim como suas vantagens e desvantagens são descritas a seguir.

3.1 Geração de *backgrounds* artificiais

Um dos métodos de segmentação de vídeo em duas camadas que consegue os melhores resultados, utilizando ambientes controlados, é a utilização de uma cor chave de fundo (*chroma key*). Neste método, amplamente utilizado pela indústria cinematográfica e televisiva, o fundo da imagem apresenta uma cor conhecida pelo sistema e iluminado uniformemente por equipamentos específicos. O algoritmo, a partir da informação da cor do fundo, elimina da imagem todos os pixels que possuírem esta cor. A única ressalva na utilização deste método é que os objetos no *foreground* não devem possuir aquela cor, pois, neste caso serão considerados como *background* pelo método.

Ao utilizar um ambiente controlado, melhores resultados são obtidos em relação à algoritmos que utilizam o fundo arbitrário (CRIMINISI et al., 2006; STAUFFER; GRIMSON, 2000; YIN et al., 2011) no processo de segmentação do vídeo. Tiburzi et al. (2008) propuseram um método que gera *ground truths* com base em ambientes controlados utilizando fundo de cor única e produzem novos vídeos com fundos artificiais que são considerados como os fundos originais dos vídeos, para uma posterior análise.

O método consiste nos seguintes passos:

1. Vídeos são gravados em ambientes controlados, com fundos de cor única e conhecida pelo sistema;

2. Uma segmentação ótima é obtida a partir de métodos que utilizam a cor chave para a segmentação. Essa segmentação é considerada o *ground truth* para aquele vídeo;

3. O fundo de cor única do vídeo é substituído por um fundo arbitrário, como, por exemplo, o vídeo de um parque com árvores ao fundo, ou do interior de um escritório; e

4. O novo vídeo constituído com este novo fundo é considerado como o vídeo original, capturado pela câmera.

Ao final destes passos, as informações obtidas são vídeos com fundos arbitrários, como se tivessem sido obtidos em ambientes não controlados, e *ground truths* destes vídeos, obtidos em ambientes controlados.

O grande problema deste método é que nos vídeos com fundos arbitrários, o *foreground* e o *background* não foram obtidos no mesmo ambiente, com isso, qualquer variação na iluminação do *background*, por exemplo, não causa efeitos no *foreground*, ao contrário de um vídeo natural com fundo arbitrário. Além disso, quando se deseja simular situações nos vídeos para analisar como os algoritmos de segmentação de vídeo se comportam, isso não é possível ao utilizar os vídeos gerados artificialmente. Um exemplo de como a iluminação pode ser alterada, é no caso de no vídeo do *background*, uma luz ser apagada, fazendo com que o ambiente fique escuro. Em um vídeo capturado integralmente naquele ambiente, o *foreground* também ficaria escuro, porém nos vídeos artificiais, como o *foreground* foi obtido em outro ambiente, ele não sofreria essa alteração de iluminação.

Além disso, caso haja muito contraste entre o *foreground* e o novo *background*, fica nítido que o vídeo foi gerado artificialmente. Uma vez que as bordas entre o *foreground* e o *background* se tornam evidentes, o desempenho dos algoritmos que serão testados nestes vídeos artificiais será diferente se comparado à um vídeo onde o *foreground* e o *background* foram capturados no mesmo ambiente em um mesmo fluxo de vídeo.

Os autores, ainda, não propõem nenhum tipo de análise em relação ao conteúdo do vídeo. Desse modo, ainda que os erros de classificação de pixels possam ser identificados, as situações-problema as quais os algoritmos de segmentação podem encontrar maior dificuldade não necessariamente serão simuladas.

3.2 Algoritmos de segmentação de imagem

Existem diversos algoritmos que utilizam diferentes abordagens para a realização de segmentação em imagens, como descrito no Capítulo 2. Uma das alternativas para a extração de *foreground* em vídeos, para a geração do *ground truth*, é a utilização direta destes algoritmos. Como o objetivo é a segmentação de vídeos, estes métodos são empregados em todos os quadros do vídeo, como se fossem imagens individuais.

Utilizar esses métodos reduz a duração e o esforço da segmentação totalmente manual, tanto em imagens quanto em vídeos. No caso dos vídeos, esse ganho é ainda maior, uma vez que devido à similaridade de quadros próximos, as marcações que o usuário realizou em um dos quadros podem ser utilizadas em outros, considerando que os objetos do *foreground* estejam no mesmo local ou bem próximos às suas posições nos quadros anteriores. Em *video chats*, por exemplo, o *foreground* não tem tanta movimentação, então para grande parte dos quadros, as mesmas marcações podem ser utilizadas para a segmentação.

Outra alternativa, ao contrário das marcações manuais, é buscar formas de se obtê-las para todos os quadros do vídeo de uma maneira automática. Uma destas formas é a utilização de equipamentos específicos que estimam a profundidade em cada pixel de uma

imagem bidimensional. Estes equipamentos, que utilizam técnicas como *Time-of-Flight* (TOF), luz estruturada ou estereoscopia, conseguem estimar para quadro do vídeo, quais são os objetos que estão mais próximos ou mais distantes da câmera. Alguns trabalhos utilizam desta informação (WANG et al., 2010; WU; BOULANGER; BISCHOF, 2008), em conjunto com algoritmos de segmentação de imagens para extrair o *foreground* em vídeos automaticamente, onde não é preciso que o usuário realize nenhuma marcação, uma vez que os objetos mais próximos à câmera são considerados como *foreground* pelos algoritmos, enquanto o restante da imagem é considerado como *background*. Porém, como a segmentação obtida por esses algoritmos não é perfeita, esta não pode ser considerada o respectivo *ground truth* do vídeo.

Por se tratar de uma segmentação que não é realizada em tempo real e visto que métodos de segmentação de imagens precisam da interação do usuário, utilizar este tipo de equipamento em um primeiro passo de um processo interativo, onde apenas alguns erros devem ser corrigidos manualmente num segundo momento, torna-se uma alternativa viável para a geração de *ground truths*.

Vale ressaltar que, apesar desses métodos facilitarem a produção de uma estrutura capaz de identificar erros de segmentação, para a geração de um *ground truth* ainda é necessária a preocupação com o conteúdo gerado. Uma vez que situações-problema procuram forçar a falha de um algoritmo, a simples aplicação de um método de segmentação não será suficiente para produzir um *ground truth*.

4 Mapas de Profundidade e calibração de câmeras

Como visto no capítulo anterior, uma maneira interessante de realizar a segmentação de imagens e vídeos automaticamente é a utilização de informações extras do ambiente, como o mapa de profundidade. Porém, muitas vezes é preciso que seja realizado um processo de calibração destes tipos de sensores e a junção destas informações às provenientes de câmeras convencionais, um processo semelhante em vários aspectos ao processo de calibração de múltiplas câmeras (ZHANG, 2012).

Este capítulo apresenta os principais métodos e dispositivos utilizados para se obter mapas de profundidade, e um processo de união entre as informações capturadas por uma câmera de vídeo de alta definição e um sensor de profundidade.

4.1 Métodos de obtenção do Mapa de Profundidade

A estrutura conhecida por *depth map*, ou mapa de profundidade, de uma imagem, é representada por uma matriz bidimensional, com as dimensões equivalentes às dimensões da imagem, ou seja, possui um valor para cada pixel da imagem RGB. Muitas vezes a informação deste mapa é incorporado na própria imagem, e armazenado em um canal adicional, chamando de imagem de profundidade ou RGB-D (Depth ou profundidade).

Os valores de cada pixel em um mapa de profundidade representam as distâncias entre os objetos presentes na imagem em relação ao dispositivo que fez a captura. No caso de vídeos, para cada quadro do mesmo, um mapa de profundidade é calculado.

Sensores de profundidade, ou câmeras 3D, são dispositivos que calculam o mapa de profundidade de uma imagem. Exemplos deste tipo de dispositivos que estão sendo utilizados nas duas últimas décadas são o sensor de movimento Kinect da Microsoft (ZHANG, 2012), o sensor D-Imager da Panasonic (Panasonic, 2010), o sensor Xtion-Pro da Asus (ASUS XTION, 2014) e a câmera ZCam da 3DV systems (IDDAN; YAHAV, 2001).

Os três principais métodos de obtenção do mapa de profundidade por estes tipos de sensores são descritos a seguir. O primeiro deles é baseado em informações de câmeras convencionais, enquanto nos outros dois, equipamentos específicos para a captura deste tipo de informação são necessários.

4.1.1 Imageamento estéreo

Existem diversos métodos para a obtenção do mapa de profundidade de um ambiente, utilizando apenas câmeras de vídeo convencionais. Algumas técnicas utilizam imagens capturadas de uma mesma câmera, baseadas em variações na iluminação do ambiente (BASRI; JACOBS; KEMELMACHER, 2007) ou variação na distância focal da própria câmera (ZHANG; NAYAR, 2006). Por outro lado, para a obtenção dessas informações para vídeos, não é possível realizar essas alterações no dispositivo, ou no ambiente. Para isso, outras técnicas, baseadas em imagens estereoscópicas podem ser utilizadas (FAUGERAS, 1993). Cada par de imagem capturado por essas câmeras representa duas perspectivas diferentes de uma mesma cena estática. O maior desafio neste tipo de técnica é a identificação dos pares de pontos, presentes no par de imagens capturadas, que correspondem ao mesmo ponto da cena. Como as imagens, capturadas por uma câmera convencional, não possuem informações adicionais sobre a cena, os pontos devem ser procurados em toda a imagem, o que nem sempre é uma tarefa que obtém bons resultados nos algoritmos deste tipo. Uma maneira de limitar as buscas por esses pontos é utilizar as relações de geometria epipolar (FAUGERAS, 1993).

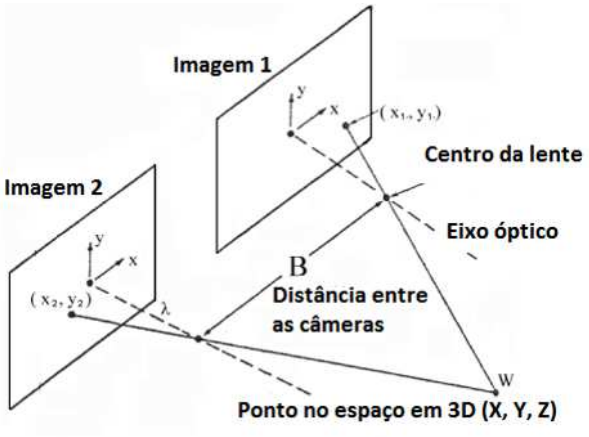
A geometria epipolar para imageamento estéreo, estabelece uma relação geométrica entre duas vistas capturadas por câmeras com centros de projeções não coincidentes. Em termos de algoritmos de visão estéreo destinados à computação de correspondências de pontos, o benefício proporcionado pela geometria epipolar é que, dado um ponto x referente à primeira vista, a busca por pontos correspondentes na segunda vista não precisa cobrir toda a imagem e restringe-se apenas a uma linha na segunda imagem. Esta restrição também é conhecida na literatura como restrição epipolar.

A partir do reconhecimento de um par de pontos correspondentes $x \leftrightarrow x'$ de duas câmeras P e P', é possível a triangulação de um ponto X do espaço 3D, em que X projeta o par de pontos correspondentes nas imagens. A Figura 10 representa um par de imagens capturadas por um par de câmeras, onde o ponto no espaço 3D é calculado através da triangulação. O conjunto de valores de todos os pontos 3D, obtidos com o processo de triangulação, formam o mapa de profundidade das imagens das câmeras.

4.1.2 *Time-of-flight*

A tecnologia conhecida por *Time-of-flight* (TOF) calcula a distância com base na velocidade da luz, calculando o "tempo de voo"(razão do nome da técnica) de um sinal de luz entre a câmera e um objeto. Equipamentos que utilizam desta tecnologia emitem sinais de luz que colidem com o ambiente e retornam ao equipamento. Quando a luz retorna ao dispositivo, a diferença de tempo entre a emissão e a captura do ponto determina a profundidade de um objeto (distância entre o objeto e o sensor), ou seja, quanto menor for o tempo de voo do ponto, menor é sua profundidade (HANSARD et al., 2013). O

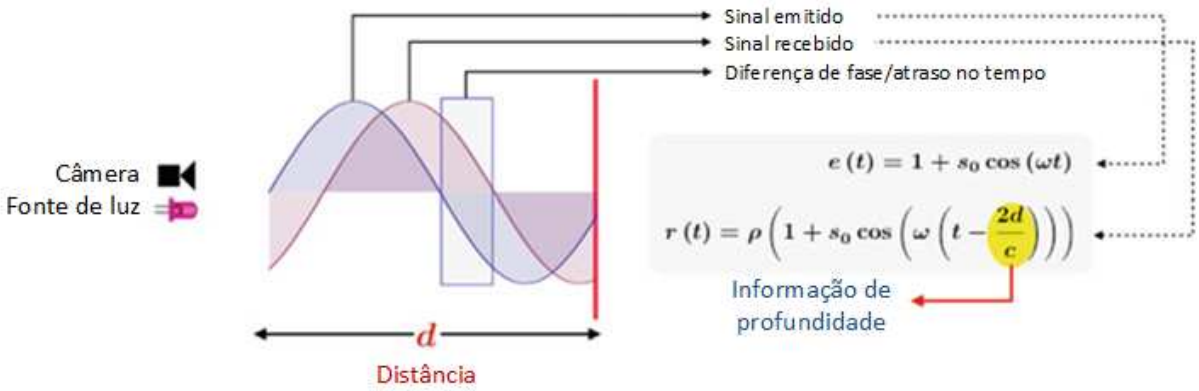
Figura 10 – Princípio do método de triangulação de imagens para a obtenção do mapa de profundidade.



Fonte: Kabayama e Trabasso (2002).

conjunto de valores de profundidade, é chamado de mapa de profundidade. A Figura 11 ilustra este funcionamento.

Figura 11 – Princípio de funcionamento de uma câmera *TOF*. O tempo entre a emissão e a captura dos sinais infravermelhos calculados representam a profundidade de um objeto.



Fonte: Shao et al. (2014).

Na maioria das câmeras deste tipo, para cada pixel da imagem RGB (ou em tons de cinza, dependendo do dispositivo), um ponto de luz é emitido e capturado. Com isso, para cada pixel de cor capturado pela câmera, um valor de profundidade é obtido. Em outros dispositivos, um mesmo valor de profundidade pode representar, mais que um pixel da imagem, de maneira aproximada. Com isso, é possível reduzir o tamanho do mapa de profundidade, porém, isso faz com que a qualidade do mesmo seja afetada com essa aproximação.

Exemplos de dispositivos que utilizam desta tecnologia são a câmera ZCam (IDDAN; YAHAV, 2001), a D-Imager (Panasonic, 2010) e a segunda versão do Kinect, de-

senvolvido para o console Xbox One da Microsoft (SHAO et al., 2014).

Na Figura 12 é apresentada a ZCam, assim como as imagens geradas por esta câmera.

Figura 12 – ZCam e as imagens geradas por essa câmera; (a)ZCam; (b)Imagem em tons de cinza; (c)Mapa de Profundidade.

(a) (b) (c)

Fonte: Ahn et al. (2009).

Entre os problemas que existem em sensores deste tipo tem-se: a baixa resolução no mapa de profundidade e baixa resolução espacial, assim como erros causados por variações radiométricas, geométricas e de iluminação. Por exemplo, a acurácia no cálculo do mapa de profundidade é limitado pela potência da emissão de sinais infravermelhos, o que geralmente é baixo se comparado à luz solar, que por sua vez pode contaminar o sinal emitido pelos sensores. Além disso, a amplitude do sinal infravermelho varia de acordo com o material e a cor da superfície dos objetos (HANSARD et al., 2013).

Outro problema enfrentado por esse tipo de sensor na captura de um vídeo é o *motion blur* (desfoque de movimento em tradução livre), causado por movimentos da câmera ou do objeto, que pode ocasionar discrepâncias entre a imagem RGB e o mapa de profundidade devido a diferença entre os tempos de captura de cada quadro do vídeo (*frame rate*) e o tempo do cálculo da profundidade de cada ponto no respectivo quadro do vídeo (HANSARD et al., 2013).

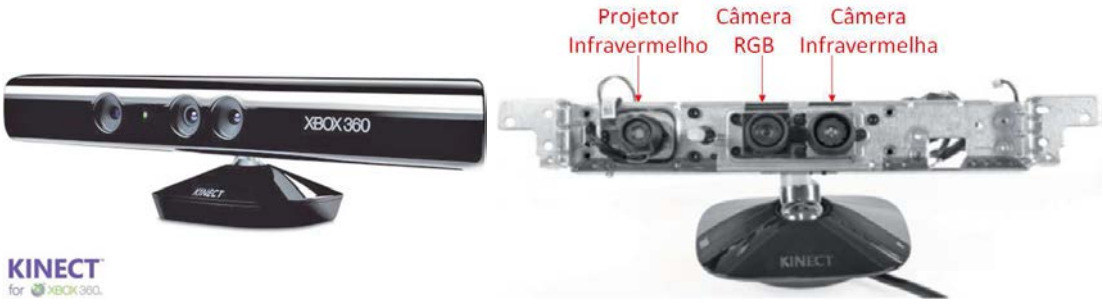
4.1.3 Luz estruturada

Alguns dispositivos, conhecidos como *scanners 3D* de luz estruturada, utilizam a emissão de luz infravermelha de maneira estruturada e um sistema de câmeras infravermelhas para calcular o mapa de profundidade de um ambiente.

Dispositivos baseados nesta técnica, como a primeira versão do Kinect (para o console Xbox 360) da Microsoft, possuem, geralmente, dois principais componentes: um projetor de luz infravermelha e uma câmera infravermelha. A Figura 13 (ZHANG, 2012) apresenta um sensor Kinect, detalhando em seu interior a posição dos componentes neces-

sários para a geração de um mapa de profundidade. O Kinect ainda possui uma câmera RGB convencional, para a captura de imagens RGB respectivas ao mapa de profundidade.

Figura 13 – Microsoft Kinect. (a) Visão externa do sensor; (b) O projetor infravermelho, a câmera infravermelha e a câmera RGB no interior do Kinect.



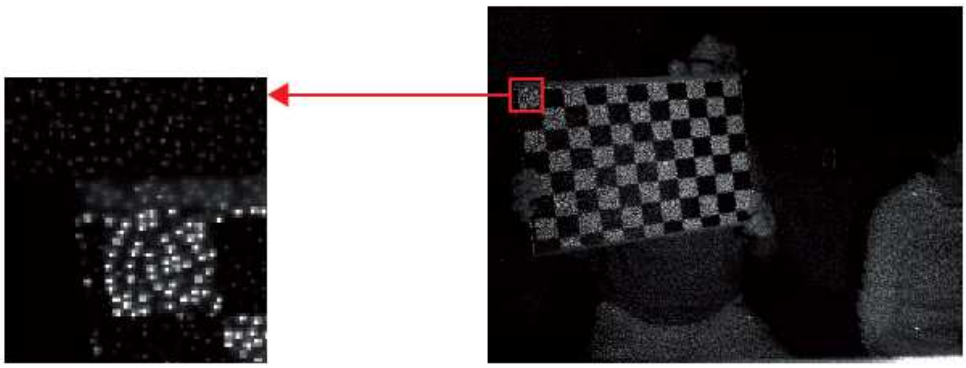
(a) (b)

Fonte: Zhang (2012)

Para a obtenção do mapa de profundidade utilizando esse método, é preciso que a câmera infravermelha identifique os pontos infravermelhos projetados no ambiente e compare com um padrão conhecido. Esse padrão, como no Kinect, corresponde ao padrão de pontos capturados à uma distância conhecida e controlada. Após ser projetado sobre um ambiente e capturado pela câmera infravermelha, o padrão de pontos capturado é comparado ao padrão conhecido pela câmera para a obtenção do mapa de profundidade, ou seja, as distorções que o padrão de pontos sofre ao refletir em um ambiente desconhecido, determina se um objeto está mais próximo ou mais distante que outro (ZHANG, 2012).

A Figura 14 apresenta o padrão de pontos emitidos pelo Kinect refletidos em um ambiente, capturados pela câmera infravermelha do próprio Kinect.

Figura 14 – Padrão de pontos infravermelhos projetados pelo Kinect em um ambiente.



Fonte: Zhang (2012)

A Figura 15 representa o mapa de profundidade gerado pelo Kinect, com base no padrão de pontos infravermelhos capturados (Figura 14). Nessa representação em tons

de cinza, as áreas mais escuras representam os objetos mais próximos, enquanto as áreas mais claras, os mais distantes em relação à câmera. A cor preta, por sua vez, representa as áreas onde a profundidade não pode ser calculada pelo dispositivo.

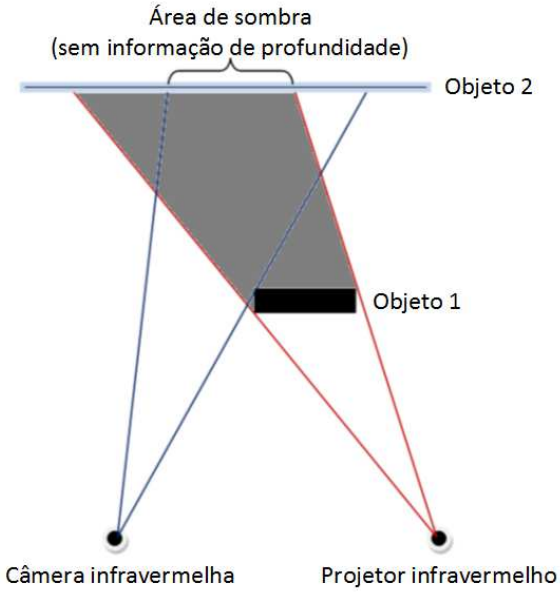
Figura 15 – Mapa de Profundidade obtido com o Kinect.

Fonte: Zhang (2012)

Há duas situações onde são geradas áreas desconhecidas por sensores que utilizam luz estruturada para calcular o mapa de profundidade. Uma delas é o aparecimento de regiões de sombra. Estas sombras são ocasionadas devido a distância entre o emissor e a câmera de luz infravermelha, onde algumas áreas visíveis à câmera não receberam nenhum ponto de luz projetado pelo emissor, sendo impossível calcular sua profundidade (DANCIU; BANU; CALIMAN, 2012). A Figura 16 representa como é formada essa região. É importante salientar que quanto menor for a distância entre os objetos e o sensor, maior será a área da sombra gerada.

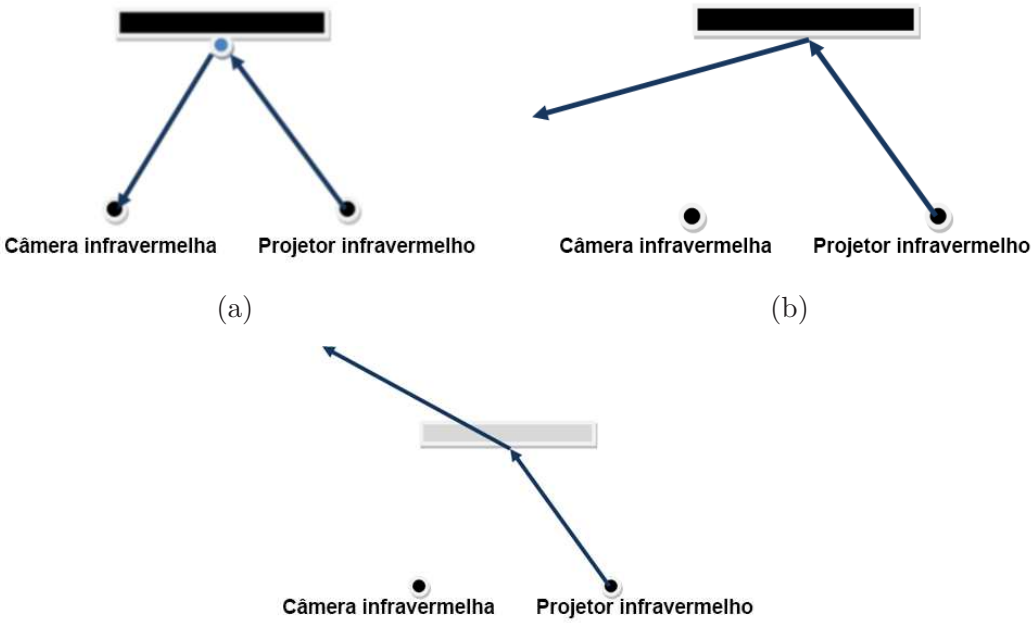
O segundo problema enfrentado por este tipo de sensor, ocorre pelo fenômeno de refração e reflexão das superfícies dos objetos capturados. Superfícies podem ser divididas em três grupos, referentes aos materiais que as compõem: materiais que absorvem a luz infravermelha (papel e madeira), materiais que refletem a luz (espelhos) e materiais que refratam a luz (vidro e plásticos transparentes). Os pontos projetados, podem ser capturados apenas pelos materiais que absorvem a luz. Nos outros dois casos, os pontos não são capturados pela câmera infravermelha do sensor e portanto, são desconsiderados do mapa de profundidade, gerando as áreas desconhecidas (DANCIU; BANU; CALIMAN, 2012). A Figura 17 representa como os três materiais se comportam com relação a luz projetada pelo sensor.

Figura 16 – Representação de como as sombras no mapa de profundidade são geradas.



Fonte: Danciu, Banu e Caliman (2012)

Figura 17 – O comportamento dos feixes de luz infravermelha projetadas em objetos de diferentes materiais; (a) Material que absorve a luz infravermelha; (b) Material que reflete a luz infravermelha; (c) Material que refrata a luz infravermelha.



Fonte: Danciu, Banu e Caliman (2012)

4.2 Calibração de câmeras e projeção do mapa de profundidade

Um dos problemas na utilização de equipamentos que capturam e calculam o mapa de profundidade de um ambiente é a baixa resolução e qualidade das imagens RGB geradas por este tipo de sensor, uma vez que para muitas aplicações ela não é utilizada. Exemplos dessa situação acontecem em aplicações que utilizam apenas o mapa de profundidade para reconhecimento de gestos dos usuários e não exibem as imagens RGB capturadas pelo sensor.

Uma alternativa ao uso das imagens RGB provenientes destes sensores é a utilização apenas do mapa de profundidade em conjunto com imagens de câmeras de alta resolução e definição. Esta junção é realizada baseada na calibração dos sensores e é bastante semelhante aos processos envolvendo câmeras convencionais, onde os parâmetros intrínsecos e extrínsecos de cada dispositivo são calculados. É importante salientar que para realizar a calibração e a junção das informações destes sensores é preciso que os mesmos sejam fixados fisicamente um ao outro, para que os parâmetros extrínsecos (posição e orientação relativos entre os dispositivos) possam ser calculados.

Um método de calibração que vem sendo utilizado amplamente nos últimos anos (TYKKALA et al., 2014; HAN et al., 2013) foi proposto por Herrera C., Kannala e Heikkila (2012). Com este método, através da captura de imagens de um reticulado, comumente utilizado em algoritmos de calibração de câmeras convencionais, e a detecção das bordas deste padrão, é possível calcular os parâmetros intrínsecos de distância focal e coeficiente de distorção da lente de ambas as câmeras (câmera RGB e câmera IR), além das matrizes de rotação e translação que devem ser aplicadas sobre o mapa de profundidade para o alinhamento com a imagem RGB.

Imagens do reticulado em diferentes posições, ângulos e distâncias devem ser capturadas por ambos os dispositivos e fornecidas ao algoritmo para o cálculo dos parâmetros. A detecção dos cantos do tabuleiro nas imagens RGB é feita automaticamente utilizando métodos de visão computacional enquanto que no mapa de profundidade os cantos devem ser selecionados manualmente, pois o reticulado não é visível no mapa de profundidade. A Figura 18 apresenta um exemplo de imagem RGB e o mapa de disparidade usadas pelo algoritmo que calcula os parâmetros.

Para os parâmetros intrínsecos da câmera RGB, é utilizado um modelo similar ao de Heikkila (2000), que consiste de um modelo de uma câmera do tipo *pinhole* (câmera estenopeica) com correções de distorção radial e tangencial. A projeção de um ponto das coordenadas da câmera $\mathrm{x}_c = [x_c, y_c, z_c]^\top$ para as coordenadas de cor da imagem $\mathrm{p}_c = [u_c, v_c]^\top$ é obtido através das Equações 4.1 e 4.2 a seguir. Como primeiro passo, o

Figura 18 – Reticulado utilizado para a calibração. (a) Imagem RGB da câmera convencional; (b) Mapa de profundidade correspondente à imagem RGB capturado pelo sensor de profundidade.

(a) (b)

Fonte: Herrera C., Kannala e Heikkila (2012)

ponto é normalizado por $\mathrm{x}_n = [x_n, y_n]^\top = [x_c/z_c, y_c/z_c]^\top$. A distorção então é realizada:

$$\mathrm{x}_g = \begin{bmatrix} 2k_3 x_n + k_4(r^2 + 2x_n^2) \\ k3(r^2 + 2y_n^2) + 2k_4 x_n y_n \end{bmatrix} \tag{4.1}$$

$$\mathrm{x}_k = (1 + k_1 r^2 + k_2 r^4 + k_5 r^6)\mathrm{x}_n + \mathrm{x}_g \tag{4.2}$$

onde $r^2 = x_n^2 + y_n^2$ e $\mathrm{k}_c = [k_1, ..., k_5]$ é um vetor contendo os coeficientes de distorção. Por fim, as coordenadas da imagem são obtidas pela Equação 4.3 :

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} f_{cx} & 0 \\ 0 & f_{cy} \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} u_{0c} \\ v_{0c} \end{bmatrix} \tag{4.3}$$

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

O modelo de parâmetros intrínsecos depende do sensor de profundidade utilizado. No trabalho onde a calibração foi proposta (HERRERA C.; KANNALA; HEIKKILA, 2012) , o sensor utilizado foi o Kinect. É importante salientar que o Kinect consiste de um sensor que utiliza luz estruturada (seção 4.1.3), no qual um emissor projeta os pontos de luz, enquanto uma câmera infravermelha os captura, comparando a disparidade entre o padrão de pontos capturados em relação à um padrão conhecido previamente capturado à uma distância conhecida. Portanto, a informação de saída do Kinect é um mapa de disparidade, contendo valores em KDUs (*Kinect Disparity Units*).

A transformação entre as coordenadas do sensor de profundidade $\mathrm{x}_d = [x_d, y_d, z_d]^\top$ e as coordenadas da imagem de profundidade $\mathrm{p}_d = [u_d, v_d]^\top$ é bastante similar ao modelo utilizado pela câmera RGB.

Após obtido os parâmetros intrínsecos do sensor de profundidade, é necessário remover a distorção do mapa de disparidade e calcular o valor de profundidade em metros. A Equação 4.4 mostra o cálculo da profundidade de um pixel do mapa de disparidade, com z representando o valor de profundidade, d representando o valor de disparidade e c_0 e c_1 representando os coeficientes de distorção da câmera IR.

$$z = \frac{1}{c_1 d + c_0} \tag{4.4}$$

Com o valor de profundidade do pixel calculado, é possível projetar cada pixel do mapa de disparidade em um espaço 3D, transformando cada pixel em um ponto 3D (com coordenadas x, y e z). A Equação 4.5 e a Equação 4.6 mostram, respectivamente, a projeção das coordenadas x e y, com a dupla (u, v) representando a posição do pixel no mapa de disparidade, (fxd, fyd) representando os parâmetros de distância focal da câmera infravermelha e (cxd, cyd) representando o centro ótico focal da câmera IR. A coordenada z é o próprio valor de profundidade em metros.

$$x = \frac{(u - c_{xd}) \times z}{f_{xd}} \tag{4.5}$$

$$y = \frac{(v - c_{yd}) \times z}{f_{yd}} \tag{4.6}$$

Após obter a coordenada 3D de cada pixel, representada por um ponto no espaço R^3, a rotação e translação para o alinhamento das coordenadas do mapa de profundidade com a imagem RGB são aplicadas. A Equação 4.7 (BURRUS, 2011) demonstra a aplicação da rotação e translação, com $P3D$ representando o ponto 3D, R representando a matriz de rotação, T representando a matriz de translação e $P'3D$ representando o ponto rotacionado e transladado.

$$P'_{3D} = R \times P_{3D} + T \tag{4.7}$$

A projeção dos pixels para o espaço 3D (P_{3D}) é feita utilizando os parâmetros intrínsecos da câmera IR. Com os pontos 3D rotacionados e transladados (P'_{3D}), o processo inverso é realizado: a reprojeção dos pontos 3D para uma imagem 2D. Porém, para este passo, os parâmetros intrínsecos utilizados serão os da câmera RGB externa.

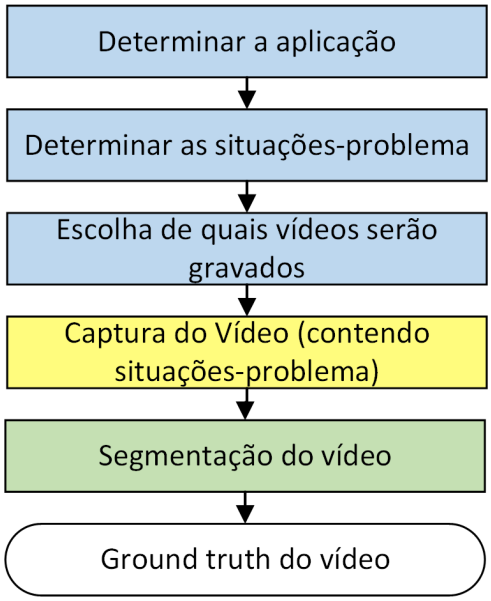
Caso a resolução da câmera externa, calibrada ao sensor de profundidade, seja maior que o mapa de profundidade obtido por este sensor, é preciso realizar uma in-

terpolação do mapa de profundidade. Essa interpolação faz com que pixels vizinhos da imagem RGB possuam o mesmo valor de profundidade, reduzindo a qualidade do mapa de profundidade. Todavia, essa interpolação permite que câmeras com altas resoluções sejam utilizadas com mapas de profundidade menores, como é o caso dos experimentos realizados por Herrera C., Kannala e Heikkila (2012) , onde a resolução da câmera RGB é de 2784x1856 pixels, enquanto o mapa de profundidade possui apenas 1280x1024 pixels (HERRERA C.; KANNALA; HEIKKILA, 2012).

5 Método proposto para a geração de *Ground truths*

O método proposto para gerar *ground truths* de vídeos pode ser dividido em diversas etapas, que são apresentadas na Figura 19.

Figura 19 – Etapas necessárias para a geração de *ground truths* de vídeos



Fonte: elaborada pelo autor.

Um dos maiores problemas no processo de geração de *ground truths* é a etapa de segmentação dos vídeos (a última etapa na estrutura apresentada na Figura 19). Nas seções deste capítulo são descritas as diferenças entre a abordagem de segmentação manual para geração de *ground truths*, e a abordagem semiautomática proposta.

A primeira etapa consiste na identificação da aplicação em que o método de segmentação será utilizado. Essa identificação é necessária, pois cada aplicação possui um cenário específico. Em aplicações de *video chats*, por exemplo, o elemento de interesse trata-se de uma pessoa em que apenas a parte superior do seu corpo é visualizada.

A próxima etapa é a identificação das situações-problema. Uma vez que um *ground truth* tem como objetivo servir como forma de validação de um algoritmo de segmentação, a produção de vídeos que não explorem tais situações pode fazer com que qualquer algoritmo tenha um bom desempenho na extração do elemento de interesse dos vídeos.

A etapa seguinte trata da captura dos vídeos em que as situações-problema são simuladas.

A etapa mais importante é a segmentação dos vídeos capturados. Essa etapa também pode ser considerada a mais trabalhosa, pois produzirá como resultado um vídeo, com o mesmo número de quadros do vídeo de entrada, em que cada pixel é rotulado como primeiro plano, plano de fundo ou região desconhecida.

Nas seções seguintes são discutidos em detalhes os passos necessários nessa fase.

5.1 Abordagens de segmentação de vídeos para geração de *ground truths*

Uma das maneiras de segmentar corretamente um vídeo é realizar o processo manualmente, ou seja, utilizando alguma ferramenta de edição de imagens para identificar em cada quadro do vídeo, quais pixels são pertencentes ao *foreground* ou ao *background*. Esta tarefa é bastante trabalhosa e demorada, principalmente levando em conta o grande número de quadros de um vídeo (normalmente 30 por segundo, ou mais) e a dificuldade em, dado um quadro do vídeo, identificar corretamente a região correspondente de cada um dos pixels da imagem.

Como, estatisticamente, a análise dos vídeos para o desenvolvimento de uma métrica precisa de várias amostras, o trabalho é ainda maior ao realizar o processo manualmente.

Como alternativa para o uso da segmentação manual, neste trabalho é proposta uma abordagem semiautomática para a etapa de segmentação dos vídeos.

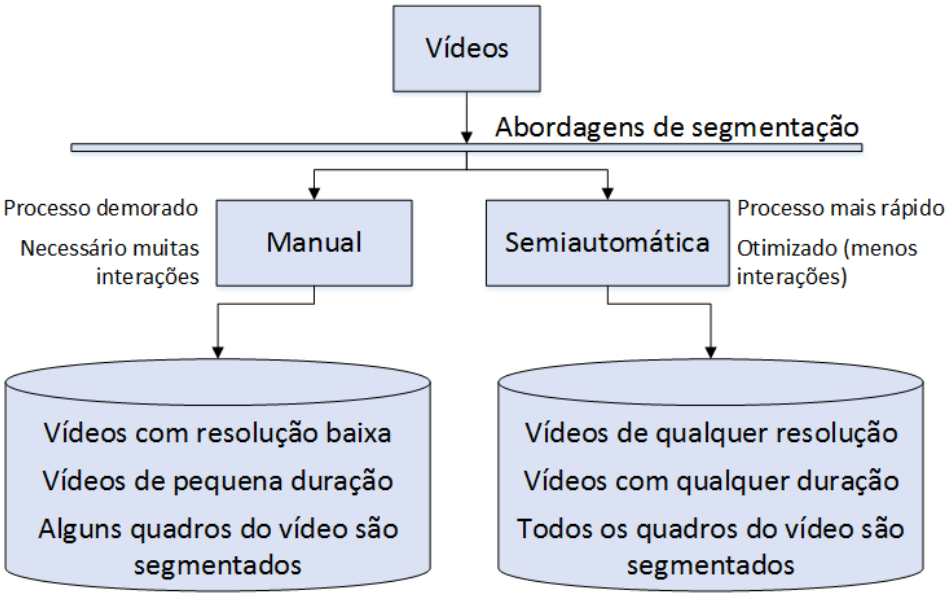
Essa segmentação semiautomática é baseada na utilização de algoritmos de segmentação de imagens, específicos para o tipo de vídeo escolhido nas etapas anteriores, em conjunto com uma etapa de refinamento, a qual permite que os erros possam ser corrigidos de forma interativa por usuários. A segmentação semiautomática permite uma série de benefícios em relação à segmentação manual. As diferenças entre as duas abordagens são apresentadas na Figura 20.

Como é possível ver na Figura 20, há duas grandes diferenças no processo de segmentação: o tempo que é demandado para realizar a segmentação e o número de interações.

Quanto maior a duração de um vídeo, maior o número de quadros que precisam ser segmentados, fazendo com que o processo manual, que é quadro a quadro, se torne demorado e necessite de muitas interações.

A partir dessas limitações, o resultado é diferente ao se utilizar essas duas abordagens. Enquanto a semiautomática não possui limitações em relação à resolução e duração dos vídeos, a forma manual faz com que seja possível apenas segmentar vídeos em baixa

Figura 20 – Diferenças entre as abordagens de segmentação manual e semiautomática



Fonte: elaborada pelo autor.

resolução, de curta duração.

Outro problema é o fato de um simples vídeo possuir muitos quadros, fazendo com que em outros trabalhos na literatura os *ground truths* gerados possuam apenas alguns quadros segmentados. Já na abordagem semiautomática, é possível que todos os quadros do vídeo possuam *ground truths*.

5.2 Etapas principais do método de segmentação semiautomático

O método de segmentação semiautomático criado pode ser dividido em duas etapas principais, e estas, divididas em vários passos que devem ser realizados em determinada ordem, para alcançar o objetivo desejado.

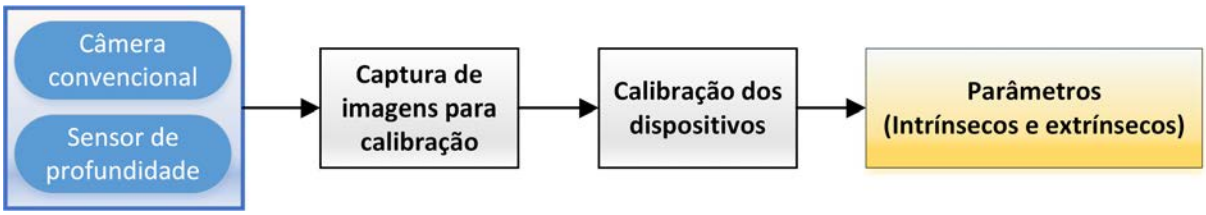
A primeira etapa do método é a obtenção de parâmetros. Essa etapa é composta de passos que devem ser realizados apenas uma vez no início do processo e não participa do fluxo principal do processo de segmentação como um todo. A estrutura desta primeira etapa é ilustrada na Figura 21.

A segunda etapa do método, por sua vez, utiliza as informações obtidas pela primeira etapa e realiza a geração dos *ground truths* de vídeos. Os passos que a constituem, por sua vez, necessitam ser executados para todo vídeo que se deseja obter o *ground truth*.

A estrutura da segunda etapa, dividida em seus passos, é apresentada na Figura 22.

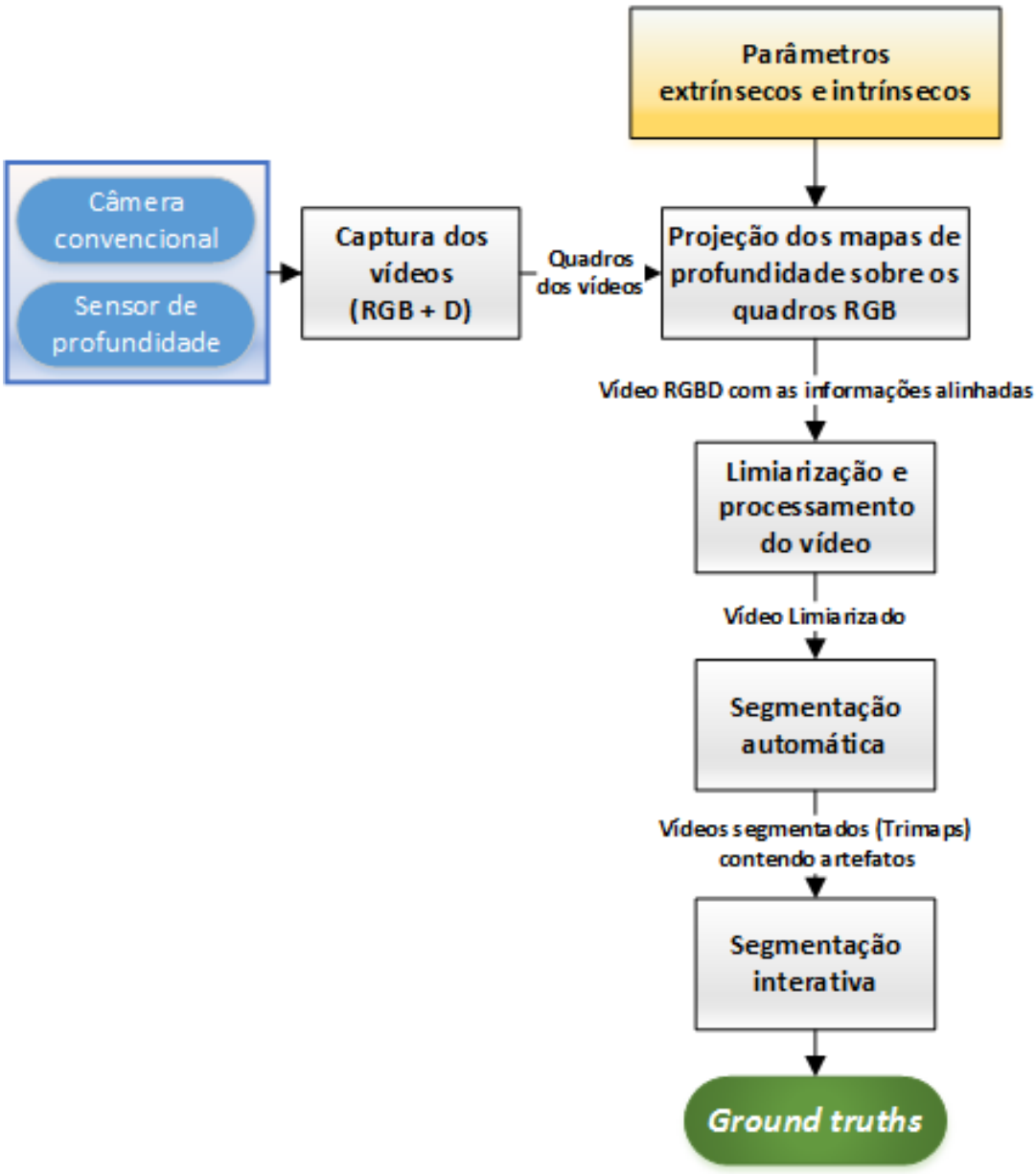
Cada um dos passos que constituem as etapas principais do método, assim como suas características e funções, são apresentados a seguir.

Figura 21 – Estrutura da primeira etapa (obtenção de parâmetros) do método desenvolvido



Fonte: elaborada pelo autor.

Figura 22 – Estrutura da segunda etapa do método desenvolvido



Fonte: elaborada pelo autor.

5.2.1 Calibração dos Dispositivos

O passo de calibração dos dispositivos é necessário ao se utilizar dois equipamentos diferentes, como uma câmera de vídeo convencional e um sensor de profundidade.

Neste passo, são calculados dois tipos de parâmetros dos dispositivos: intrínsecos e extrínsecos. Os parâmetros intrínsecos são referentes às características individuais de cada equipamento, como características das lentes (distância focal e distorção), características do sensor e a geometria de montagem da câmera. Por outro lado, os parâmetros extrínsecos são referentes ao posicionamento e orientação de um dispositivo em relação ao outro.

No final deste passo, esses parâmetros são armazenados para serem utilizados em outras etapas do processo. É importante frisar que esse passo é realizado apenas uma vez, para se obter os parâmetros dos dispositivos e não para todos os vídeos em que se deseja obter o respectivo *ground truth*. Em situações específicas quando um dos dois dispositivos é substituído ou a posição relativa entre os dois é alterada, realizar esse processo de calibração é necessário novamente.

5.2.2 Captura dos Vídeos

A captura dos vídeos, apesar de ser um passo óbvio (pois sem os vídeos não há como obter o *ground truth*), possui certas características a serem consideradas durante o processo.

Uma das preocupações no processo de captura dos vídeos que serão utilizados é em relação à taxa de quadros dos vídeos que devem ser as mesmas para ambos os dispositivos. Isso permite que para quadro da câmera convencional (imagem RGB), exista um quadro do sensor de profundidade (mapa de profundidade). Além disso, ao usar equipamentos distintos, mesmo quando a captura é iniciada no mesmo instante pelos dois dispositivos, é provável que haja um atraso entre o momento da captura do mesmo quadro entre os dois dispositivos. Dependendo das características de cada dispositivo e da maneira que ele é capturado, o atraso entre os quadros pode variar.

5.2.3 Projeção do mapa de profundidade

Com os vídeos capturados anteriormente, um passo importante para continuar com o processo é a união dos dados obtidos pelos dois dispositivos. A partir dos parâmetros obtidos na etapa de obtenção dos mesmos, para cada par de quadros do vídeo, a informação de profundidade é projetada sobre a imagem RGB.

Essa projeção utiliza os parâmetros extrínsecos para transformar os mapas de profundidade, como se eles fossem capturados na posição da câmera RGB. A partir desse momento, cada pixel de cada quadro do vídeo possui uma informação referente à sua

profundidade. Isso é armazenado em um canal adicional D (*depth*) aos canais de cores (RGB), fazendo com que cada pixel da imagem possua quatro canais: RGBD (*Red Green Blue and Depth*).

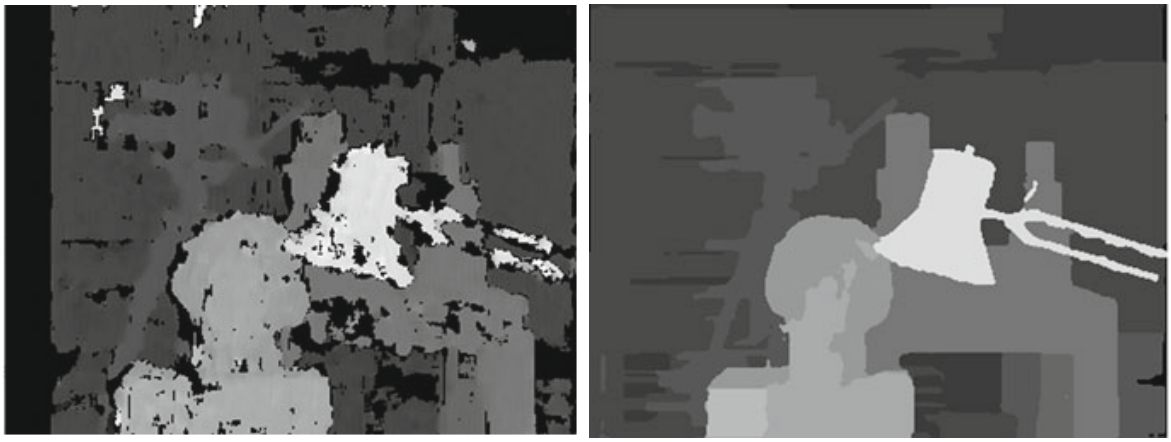
5.2.4 Limiarização e processamento do vídeo

Com o vídeo capturado e as informações de profundidade projetadas sobre eles de maneira correta (utilizando os parâmetros obtidos no processo de calibração) é preciso sinalizar quais regiões devem ser consideradas *foreground* e *background* pelos algoritmos de segmentação. Essa informação é necessária, uma vez que, para determinadas aplicações, apenas o primeiro plano deve ser considerado o *foreground*, enquanto em outros casos, o *foreground* é tudo o que está na cena, com exceção do plano de fundo.

A partir da seleção das áreas de interesse (*foreground*), filtros podem ser aplicados tanto na imagem RGB quanto ao mapa de profundidade associado à ela, para melhorar os resultados dos algoritmos de segmentação utilizados nos próximos passos do processo.

Um dos problemas comumente encontrados em mapas de profundidade, principalmente ao utilizar equipamentos baseados em luz estruturada, como o Kinect V1, por exemplo, são as áreas de sombra, descritas no Capítulo 4 (Seção 4.1.3). Nestes casos, algoritmos de preenchimento de mapas de profundidade, baseados em técnicas de colorização de imagens, podem ser utilizados (SILBERMAN; FERGUS, 2011; LEVIN; LISCHINSKI; WEISS, 2004; LE; JUNG; WON, 2014). Na Figura 23 é apresentado um exemplo de mapa de profundidade capturado por um dispositivo e o mesmo após passar pelo processo de preenchimento (SHAO et al., 2014).

Figura 23 – Exemplo do resultado de preenchimento de mapas de profundidade; (a)Mapa de profundidade original; (b) Mapa de profundidade preenchido



(a) (b)

Fonte: Shao et al. (2014)

5.2.5 Segmentação automática

O passo referente à segmentação automática é responsável por efetuar de fato uma segmentação binária do vídeo. Neste passo, cada quadro do vídeo é classificado em duas regiões (*background* e *foreground*) através de algoritmos interativos de segmentação de imagem. Esse tipo de algoritmo foi escolhido, pois os vídeos que se deseja segmentar não possuem um padrão, como por exemplo: fundos controlados e/ou *foregrounds* conhecidos, como no caso de algoritmos que segmentam apenas pessoas em primeiro plano e realizam buscas de elementos conhecidos na cena, como elementos do rosto de uma pessoa, por exemplo. No caso de vídeos que não possuem estas características, algoritmos interativos podem ser usados, caso as áreas que se deseja segmentar sejam sinalizadas de alguma forma.

Como entrada para este tipo de algoritmo, é utilizada a informação de profundidade devidamente processada e limiarizada pelo passo anterior. Com isso, é possível realizar marcações automáticas em todos os quadros de um vídeo, simulando marcações manuais realizadas por usuários. Exemplos de diferentes tipos de marcações podem ser vistas no Capítulo 2 deste trabalho.

O resultado da segmentação automática utilizando os algoritmos presentes na literatura, mesmo utilizando informações adicionais, como a profundidade, não conseguem resultados perfeitos em todos os quadros de um vídeo, por diversos motivos e situações que podem ocasionar erros (SANCHES et al., 2012a).

Como o objetivo do método é a geração de *ground truths* de vídeos e não apenas uma segmentação de vídeos utilizando informações de profundidade, como em outros trabalhos (CRIMINISI et al., 2006; YIN et al., 2011; STAUFFER; GRIMSON, 2000; WANG et al., 2010; WU; BOULANGER; BISCHOF, 2008), é necessário um último passo para o refinamento da segmentação realizada.

5.2.6 Segmentação interativa

Como visto na seção anterior deste capítulo, o processo de segmentação automática de vídeos com fundos não controlados, mesmo utilizando informações adicionais, não retorna um resultado perfeito. Como neste trabalho busca-se uma segmentação para ser utilizada como *ground truth* de vídeos, é necessário um passo que refina os resultados obtidos nos passos anteriores, para que este possa ser considerado o *ground truth* do vídeo em questão.

Para que esse refinamento não seja realizado manualmente (pixel à pixel), ou utilizando *softwares* de edição de imagem, onde os erros são corrigidos devem ser eliminados, foi desenvolvido um processo de refinamento de segmentação interativo.

Esse processo consiste na utilização de algoritmos interativos de segmentação de

imagens, como no passo anterior. Entretanto, neste caso, interações do usuário, em alguns quadros do vídeo, são necessárias.

O que difere essas interações em relação às interações que podem ser realizadas em *softwares* de edição de imagens, que possuem implementações dos principais algoritmos de segmentação de imagens, é a maneira como elas são feitas.

Por se tratar de vídeos e não imagens estáticas, interagir com todos os quadros, realizando marcações manuais neles, torna-se uma tarefa que demanda tempo e paciência por parte do usuário. Por isso, foi desenvolvido um algoritmo que analisa a informação de profundidade para realizar marcações automáticas nos quadros que possuem erros semelhantes. Assim, o usuário realiza a segmentação interativa em um quadro-chave e nos quadros do vídeo que possuem erros semelhantes, essa segmentação é realizada automaticamente, sem envolver uma nova interação do usuário.

6 Implementação do método proposto

Neste capítulo, a implementação do método de segmentação semiautomático, apresentado no capítulo anterior, é descrita.

Cada passo de cada etapa é descrito em função de suas características, os algoritmos utilizados e as soluções adotadas para os respectivos problemas encontrados durante a sua implementação.

6.1 Calibração dos dispositivos e projeção do mapa de profundidade

O processo de calibração entre uma câmera de alta resolução e um sensor de profundidade foi realizada, utilizando o *toolbox* em Matlab disponibilizado por Herrera C., Kannala e Heikkila (2012) . Maiores detalhes, sobre o processo de calibração utilizando este *toolbox* podem ser encontradas na Seção 4.1.3.

Em relação aos equipamentos, duas câmeras convencionais e um sensor de profundidade, foram utilizadas durante o processo de implementação do método. A princípio, uma *webcam* (Logitech C910) com resolução de 1280 x 720 pixels foi utilizada, porém, para a obtenção de imagens com melhor resolução e qualidade, utilizou-se uma câmera GoPro Hero 3 Black Edition, que permite altas resoluções como 1080p (1920 x 1080 pixels) e 4K (4096 x 2304 pixels), além de controles como o alteração do campo de visão da câmera e taxa de quadros por segundo, diretamente pelo dispositivo.

Algumas câmeras, como a GoPro utilizada, não são projetadas para serem ligadas à um computador diretamente como no caso das *webcams*. Por isso, uma placa de captura de vídeo (Avermedia Gamer HD) foi utilizada para realizar a comunicação entre o computador e a câmera. Esta placa recebe dados provenientes de equipamentos eletrônicos como câmeras fotográficas e filmadoras (como é o caso da câmera utilizada) através de uma interface HDMI e torna o fluxo de vídeo proveniente das câmeras, acessíveis ao computador em tempo real. Por se tratar de um equipamento que se comunica através do barramento PCI, a velocidade de captura e transmissão dos dados para o computador é maior em relação à comunicação USB, por exemplo, reduzindo atrasos que poderiam ocorrer.

Sobre o sensor de profundidade, o escolhido durante a implementação do método foi o Kinect (modelo para o console XBOX 360), da empresa Microsoft. É importante salientar que este dispositivo utiliza a técnica de luz estruturada para obtenção dos mapas

de profundidade.

Em ambos os testes, as câmeras foram unidas fisicamente ao Kinect, para manter a posição e orientação fixas entre os mesmos. A Figura 24 apresenta as estruturas físicas desenvolvidas para a realização dos dois testes no processo de calibração. Como a qualidade da imagem e a resolução são melhores ao utilizar a GoPro, esta câmera foi escolhida para dar continuidade aos experimentos durante o desenvolvimento do método.

Figura 24 – Estruturas físicas elaboradas para unir fisicamente as câmeras aos Kinects

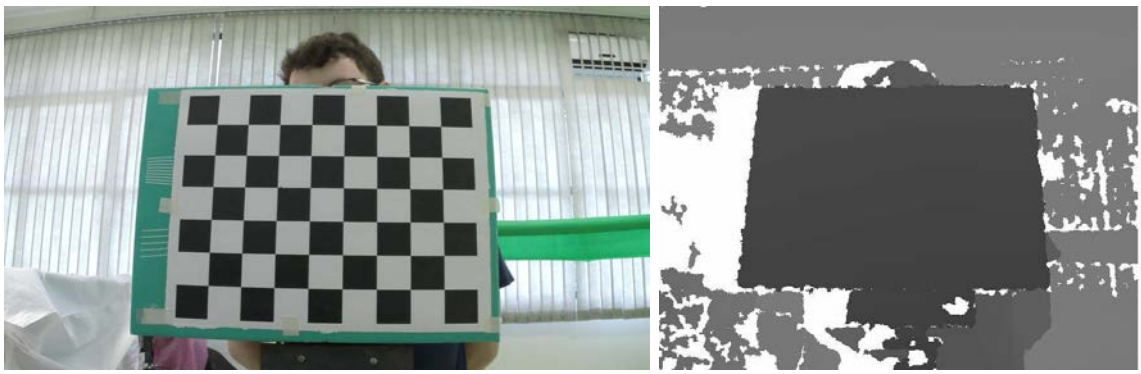
Fonte: elaborada pelo autor.

Segundo Herrera C., Kannala e Heikkila (2012) , para realizar a calibração corretamente, são necessárias pelo menos 30 imagens em diferentes ângulos e distâncias do tabuleiro, divididas em quatro tipos diferentes: com o padrão visto de frente, rotacionado no eixo X, rotacionado no eixo Y e uma imagem de uma superfície plana (sem o padrão) para correção da distorção do sensor de profundidade.

Para realizar o processo de calibração foram capturadas 40 fotos do tabuleiro, em diferentes posições e orientações, pelos dois dispositivos (Kinect e GoPro), além de uma imagem de uma parede plana (para correção da distorção do mapa de profundidade do Kinect). O tabuleiro utilizado é composto de 63 quadrados, divididos em 7 linhas e 9 colunas, brancos e pretos, de 4 centímetros cada. Essas imagens contendo este padrão são necessárias para o cálculo dos parâmetros intrínsecos e extrínsecos dos dois dispositivos. Exemplos das imagens capturadas são apresentados na Figura 25.

A partir do processo de calibração, um arquivo do tipo YAML contendo os parâmetros intrínsecos e extrínsecos dos dispositivos foi gerado e utilizado nos próximos passos durante o desenvolvimento do método. YAML é um acrônimo para "YAML Ain't Markup Language"e consiste em um formato de serialização de dados legíveis por humanos, proposto por Clark Evans em 2001, e não se trata de uma linguagem de marcação, como o XML. É importante deixar claro que o arquivo YAML é representado exclusivamente por caracteres ASCII. Os parâmetros obtidos no processo de calibração, armazenados neste padrão, podem ser lidos em um editor de texto e podem ser distinguidos, como apresen-

Figura 25 – Imagens capturadas e utilizadas para calibração dos dispositivos; (a) Imagem RGB capturada pela câmera externa; (b) Mapa de profundidade gerado pelo Kinect



(a) (b)

Fonte: elaborada pelo autor.

tado na Figura 26, onde há duas matrizes, que representam a dimensão (*rsize1*) e uma transformação linear (*kr1*), respectivamente, de uma das câmeras.

Figura 26 – Trecho do arquivo YAML que possui os parâmetros obtidos no processo de calibração dos dispositivos

```
rsize1: matrix
   rows: 1
   cols: 3
   data: [ 1080.000000, 1920.000000, 3.000000 ]
rK1: matrix
   rows: 3
   cols: 3
   data: [ 1671.678227,    0.000000, 1018.477845,
              0.000000, 1670.878581,  571.003590,
              0.000000,    0.000000,    1.000000 ]
```

Fonte: elaborada pelo autor.

6.2 Captura dos vídeos

Uma das aplicações desenvolvidas, responsável pela captura dos dados dos dispositivos, utiliza as bibliotecas LibFreenect (OPENKINECT, 2015) e OpenCV (BRADSKI, 2000).

A biblioteca OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de código aberto de visão computacional e aprendizado de máquina (BRADSKI, 2000). Ela é amplamente utilizada para resolver problemas de visão computacional, desde pro-

cessamento de imagens, calibração de câmeras, rastreamento de movimento de câmeras, identificar objetos, extrair modelos tridimensionais de objetos, etc. A biblioteca conta com uma comunidade de usuários muito grande, e uma documentação bastante detalhada, o que torna seu uso mais fácil.

A biblioteca conhecida como Libfreenect (OPENKINECT, 2015) também é uma biblioteca de código aberto e realiza a captura de informações do sensor de profundidade Kinect da Microsoft. Essa biblioteca foi escolhida pela sua simplicidade, pois esperava-se obter apenas o mapa de profundidade, e não outras informações como reconhecimento de gestos ou o rastreamento de várias pessoas (funcionalidades disponíveis no kinect), além de ser multiplataforma, funcionando em Windows, Linux e OSX.

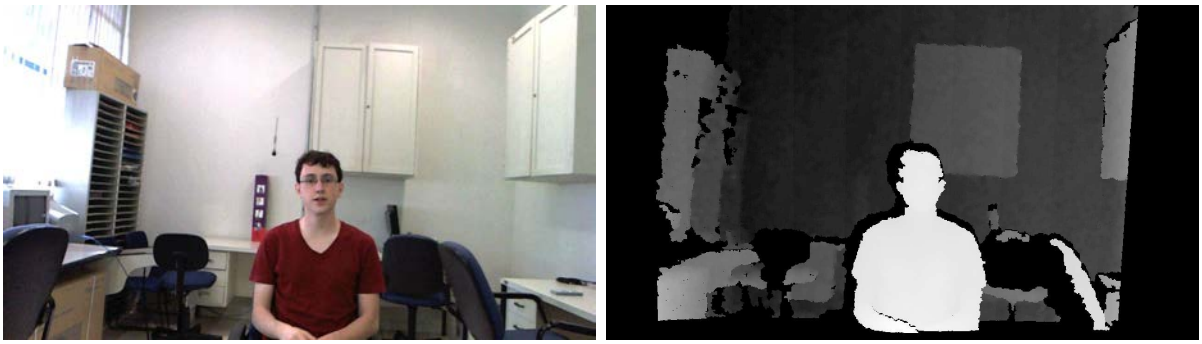
Grande parte das câmeras profissionais, utilizadas pela indústria cinematográfica, possui um mecanismo conhecido como portas para *Genlock* (*Generator Locking*) (MILLERSON; OWENS, 2012). A partir da utilização desta técnica, é possível ligar dois ou mais equipamentos, para que a captura do vídeo seja sincronizada em nível de hardware, entre os equipamentos, resultando em vídeos que possuem quadros capturados no mesmo instante.

Uma vez que a câmera e o Kinect não possuem uma ligação física, em relação aos seus respectivos hardwares, não é possível a utilização de técnicas como o *Genlock*. Portanto, foi preciso desenvolver uma aplicação para realizar a captura simultânea dos vídeos provenientes destes equipamentos.

A partir da calibração dos dois equipamentos, as informações capturadas foram associadas, fazendo com que cada pixel, de cada quadro, tenha além das informações de cor (RGB), um valor de profundidade (D). A Figura 27 apresenta um exemplo de quadro capturado pelos dispositivos, com a informação de profundidade já calibrada utilizando o método de Herrera C., Kannala e Heikkila (2012) . No mapa de profundidade, representado por tons de cinza, os menores valores de profundidade são exibidos em tons mais claros, enquanto as áreas mais escuras representam as áreas desconhecidas pelo Kinect.

O fluxo idealizado para a implementação do algoritmo é apresentado na Figura 28. Nele, todos os quadros que constituem o vídeo são capturados e armazenados na memória de acesso aleatório do computador (RAM), em um *buffer*. Após a captura de todos os quadros e o armazenamento das informações no *buffer* é que se inicia a gravação das imagens em arquivos no computador, em disco. Optou-se por realizar essas duas etapas separadamente, para que não ocorressem atrasos na captura e a taxa de quadros máxima fosse mantida (30 quadros por segundo).

Figura 27 – Quadro de um vídeo capturado pelos dispositivos. (a)Imagem RGB capturada pela câmera de vídeo; (b)Mapa de profundidade gerado pelo Kinect.



(a) (b)

Fonte: elaborada pelo autor.

Figura 28 – Fluxo do algoritmo de captura dos vídeos

Fonte: elaborada pelo autor.

6.3 Projeção do mapa de profundidade

Com os parâmetros, obtidos no processo de calibração, e com os quadros do vídeo, provenientes do processo de captura, o próximo passo do método é a projeção do mapa de profundidade sobre a imagem RGB.

Utilizando os parâmetros do arquivo de calibração é possível gerar mapas de profundidade que se adequem com as imagens RGB, mesmo que as resoluções e o aspecto das imagens não sejam iguais nos dois dispositivos.

A Figura 29 apresenta os mapas de profundidade (por meio de imagens representadas em tons de cinza), antes e após passarem pelo processo de projeção.

Uma observação importante a ser feita sobre a Figura 29 é que há duas áreas desconhecidas nas extremidades horizontais do mapa de profundidade calibrado. Essas áreas estão presentes na imagem, devido à diferença de aspecto (largura:altura) entre os dois dispositivos, uma vez que o mapa de profundidade do Kinect possui a resolução de 640x480, com um aspecto de 4:3, enquanto a imagem da câmera é capturada na resolução de 1920x1080, seguindo o aspecto de 16:9, e o mapa de profundidade calibrado deve seguir a nova resolução, referente a câmera externa.

Figura 29 – Mapas de profundidade. (a) Mapa de profundidade obtido pelo Kinect; (b) Mapa de profundidade após passar pelo processo de projeção

(a) (b)

Fonte: elaborada pelo autor.

6.4 Pré-processamento

Após a captura e processamento do mapa de profundidade com o arquivo de calibração, e apenas com essas informações, é possível determinar quais pixels da imagem estão mais próximos ou mais distantes em relação à câmera. Com esta informação e determinando um limiar de mínimo e máximo para as duas regiões a serem segmentadas (*Foreground* e *Background*), é possível realizar uma segmentação, porém, devido à baixa resolução e qualidade no mapa de profundidade gerado pelo Kinect, o resultado da segmentação não é tão satisfatório. Na Figura 30 é apresentado um quadro do vídeo segmentado apenas utilizando o mapa de profundidade após passar pelo processo de calibração. O *background*, neste caso, foi substituído por um fundo verde para que os erros da segmentação possam ser destacados visualmente.

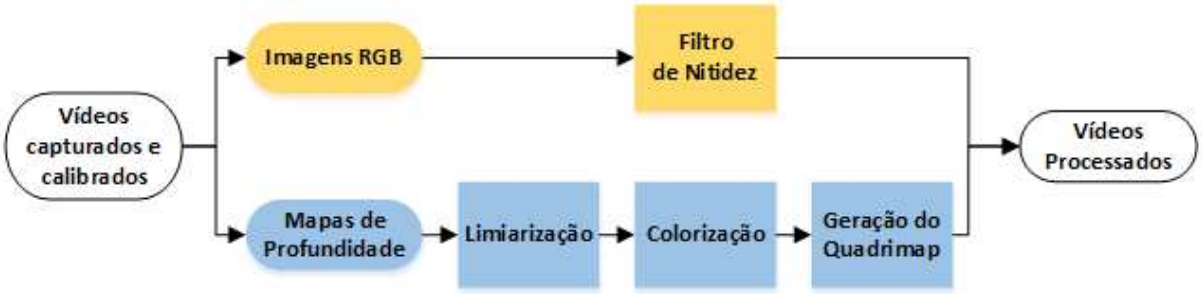
Esse processo de segmentação, mesmo possuindo vários erros em seu resultado, pode ser utilizado em aplicações onde o vídeo é capturado, segmentado e exibido ao usuário em tempo de execução. No caso de aplicações de estúdios virtuais, por exemplo, onde a segmentação do vídeo é utilizada apenas para uma pré-visualização do resultado final, a taxa de quadros do vídeo capturado, e exibido ao usuário, é mais importante do que uma segmentação mais refinada (GASPARI et al., 2014).

Como no presente estudo, busca-se uma segmentação que possua o menor número de erros possíveis (para diminuir ao máximo o número de interações do usuário), um refinamento no processo de segmentação deve ser realizado. Para isso, uma espécie de pré-processamento das informações capturadas é necessária. A Figura 31 apresenta as etapas do pré-processamento de acordo com a ordem em que as operações são efetuadas e nesta seção serão apresentados, brevemente, o que cada etapa realiza.

Figura 30 – Segmentação utilizando apenas a informação do mapa de profundidade.

Fonte: elaborada pelo autor.

Figura 31 – Etapas realizadas no processo de pré-processamento dos quadros do vídeo.



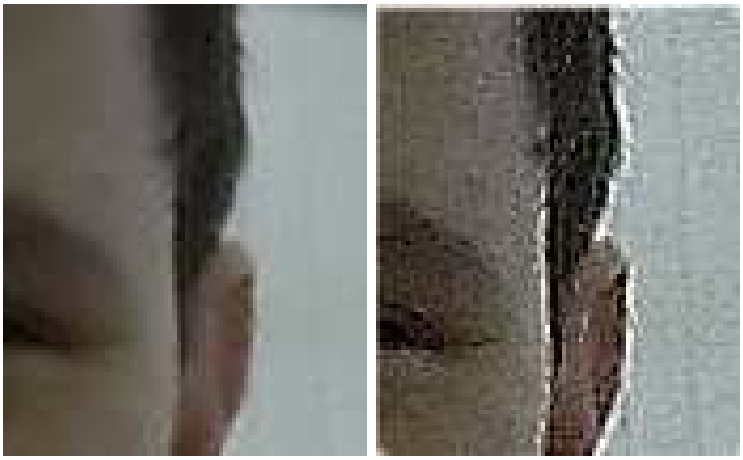
Fonte: elaborada pelo autor.

6.4.1 Filtro de Nitidez (passa-alta)

Em relação à imagem RGB capturada, apenas um filtro simples é executado para proporcionar um melhor desempenho no algoritmo de segmentação. Como o algoritmo, que será apresentado na próxima seção, é baseado no contraste entre a cor dos pixels, a imagem RGB passa por um filtro passa-alta (ou de nitidez) (GONZALEZ; WOODS, 2002) que destaca as bordas dos objetos presentes na cena, aumentando o contraste entre os pixels destas regiões. Com esse contraste acentuado, a transição entre regiões de cores diferentes se torna mais abrupta e notável visualmente.

Na Figura 32 é apresentada uma região de borda de um quadro do vídeo antes e depois de se ter aplicado o filtro passa-alta.

Figura 32 – Resultado após aplicar o filtro passa-alta à imagem RGB; (a) Detalhe nas bordas da imagem original; (b) Resultado obtido



(a) (b)

Fonte: elaborada pelo autor.

6.4.2 Limiarização do mapa de profundidade

O primeiro passo para realizar o processamento das imagens a fim de segmentá-las em dois planos é a escolha do limiar que separa essas duas regiões. Esse limiar é uma informação subjetiva, uma vez que, dependendo do resultado esperado, um valor de limiar diferente deve ser escolhido.

Em determinadas situações, todos os objetos presentes na cena são considerados *foregrounds*, exceto o plano mais distante (como uma parede), que é considerado *background*. Em outras situações, apenas o objeto mais próximo da câmera é considerado *foreground* e o restante da cena capturada é classificado como *background*. O processo de limiarização consiste em determinar qual o intervalo de valores de profundidade se encontram as regiões de *foreground* e *background* da imagem.

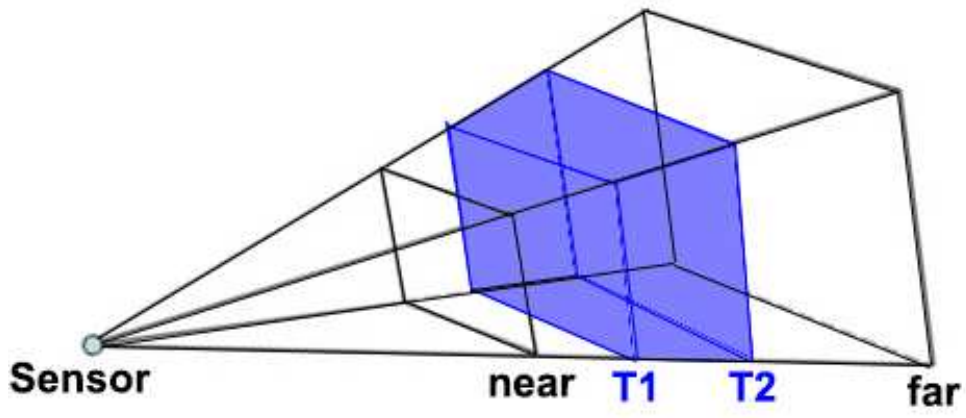
Esse processo, por ser subjetivo, exibe ao usuário um quadro do vídeo e as informações de profundidade deste quadro, para que seja determinado em que intervalo o *foreground* se encontra. Após realizada a coleta desta informação, todos os pixels que possuem valores de profundidade maiores ao intervalo recebem o maior valor possível de profundidade, para que sejam isolados nos próximos passos de processamento das imagens.

Outra forma de se visualizar o processo de limiarização é tratando os limiares como planos de cortes adicionais no *frustum* de visualização. O *frustum* de uma câmera ou sensor de profundidade consiste no volume de captura de informações, que se encontra entre dois planos, *near* e *far*, que representam o início e o final do volume de captura respectivamente.

Na Figura 33 é ilustrado um exemplo de *frustum* do sensor de profundidade uti-

lizado, com os planos *near*, *far* e dois adicionais, T1 e T2, que representam os limiares adicionados no processo de limiarização. A área entre esses planos, destacada em azul, representa a área que será considerada como *foreground*.

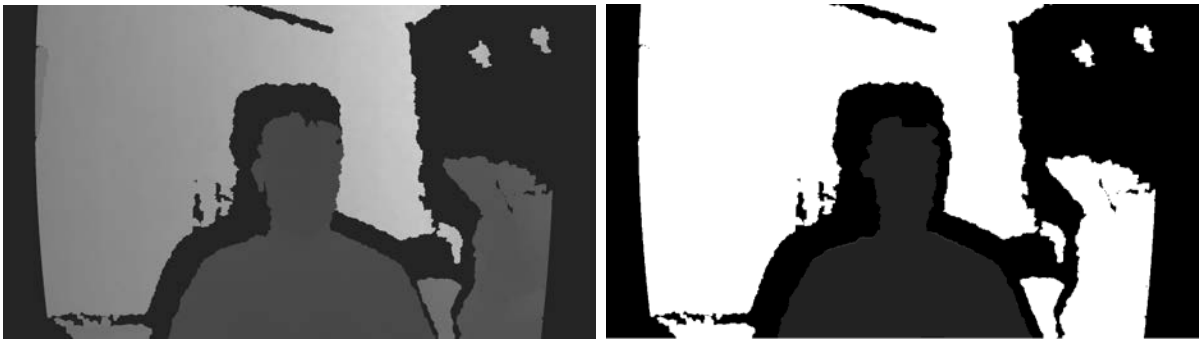
Figura 33 – Exemplo de *frustum* com os planos adicionados no processo de limiarização (T1 e T2)



Fonte: elaborada pelo autor.

Na Figura 34 é apresentado o resultado da limiarização do mapa de profundidade. Na Figura 34(a) encontra-se o mapa de profundidade original, enquanto na Figura 34(b) é apresentado o mesmo mapa de profundidade após passar pelo processo de limiarização. É importante frisar que os mapas de profundidade são apresentados em tons de cinza, nos quais regiões mais escuras representam regiões mais próximas à câmera, enquanto regiões mais claras representam as mais distantes da câmera, com exceção das regiões totalmente pretas, que são regiões que não possuem informações relativas à profundidade, que são tratadas no próximo passo no processo de pré-processamento dos mapas de profundidade.

Figura 34 – Resultado do processo de limiarização do mapa de profundidade. (a)Mapa de profundidade original calibrado; (b)Mapa de profundidade após passar pelo processo de limiarização.



(a) (b)

Fonte: elaborada pelo autor.

Como é possível observar na Figura 34, as bordas da região do foreground são diferentes nas duas imagens. Isso ocorre, pois além da limiarização, a área de borda do foreground passa por um processo de erosão (BOOMGAARD; BALEN, 1992), para que, além da área de sombra, as bordas do *foreground* também sejam analisadas pelo algoritmo de colorização.

6.4.3 Colorização do mapa de profundidade

Um dos problemas em utilizar sensores de profundidade que utilizam a técnica conhecida como luz estruturada, são as regiões de sombra, como apresentado na Seção 4.1.3.

Como a região de sombras está diretamente relacionada à disparidade do projetor de luz infravermelha em relação à câmera infravermelha e à câmera RGB do próprio equipamento, quando uma câmera externa é utilizada, a disparidade é maior, uma vez que a câmera externa está mais distante fisicamente dos outros equipamentos, como pode ser visto na Figura 24.

Esse aumento na disparidade faz com que as regiões de sombra no mapa de profundidade (onde não é possível calcular a profundidade) fiquem relativamente maiores, fazendo com que a informação de grande parte do mapa de profundidade seja perdida.

Para recuperar esses valores, um método de preenchimento de mapas de profundidade, baseado em algoritmos de colorização de imagens, foi utilizado (LEVIN; LISCHINSKI; WEISS, 2004). Este método leva em consideração as áreas desconhecidas do mapa de profundidade e a imagem RGB respectiva do mesmo quadro do vídeo. A partir das informações relativas às cores referentes as áreas desconhecidas, um valor de profundidade é calculado com base nas áreas próximas, que possuem informação de profundidade. Como a aproximação do valor da profundidade é baseado em informações de regiões próximas à elas, quanto maior a região com profundidade desconhecida, menor será a acurácia do resultado.

Como cada quadro do vídeo possui a resolução de 1920x1080 pixels, o desempenho do algoritmo, em relação ao tempo de execução do mesmo, não é satisfatório. Para que o tempo de processamento de todos os quadros do vídeo fosse viável, o mapa de profundidade de cada quadro teve sua resolução reduzida para 320x240 pixels, processado pelo método de preenchimento das áreas desconhecidas e finalmente trazido para sua resolução original. Nos testes realizados, ao serem utilizadas as imagens com menor resolução, reduziu-se o tempo de execução do algoritmo em mais de 95% (de 3600 segundos para 6 segundos para cada frame).

Na Figura 35 é apresentado um mapa de profundidade antes de passar pelo processo de preenchimento das áreas desconhecidas e o mesmo mapa de profundidade após

ser processado pelo algoritmo.

Figura 35 – Resultado do processo de preenchimento do mapa de profundidade. (a)Mapa de profundidade original; (b)Mapa de profundidade preenchido pelo algoritmo de colorização.

(a) (b)

Fonte: elaborada pelo autor.

A mudança de resolução do mapa de profundidade faz com que o resultado do mesmo, após o processo, possua as bordas menos nítidas, como apresentado na Figura 35, porém, como a segmentação, de fato, é realizada no último passo deste processo, não é necessária uma definição exata de borda nesta etapa.

6.4.4 Geração do *quadrimap*

Para a alimentação do algoritmo de segmentação de imagens utilizado no método proposto, é necessário que o mapa de profundidade seja dividido em quatro regiões bem definidas que serão processadas posteriormente: *foreground*, *background*, provável *foreground* e provável *background*.

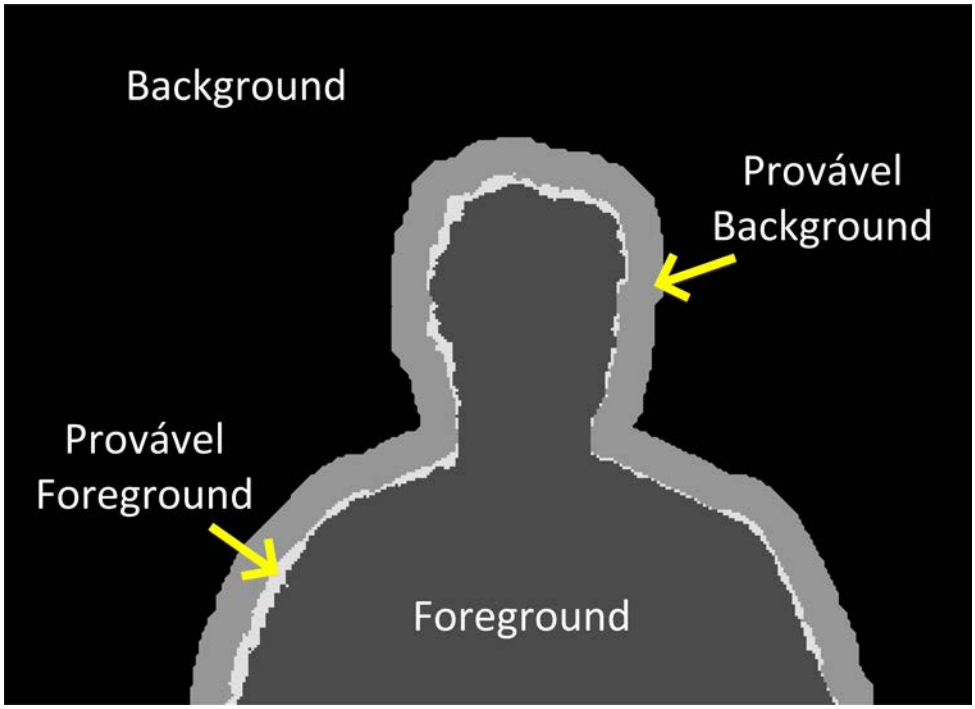
Essa nova imagem, baseada no mapa de profundidade classificado nestas quatro regiões é chamado de *quadrimap*.

Baseado em um limiar para cada região e no histograma calculado para cada mapa de profundidade (considerando este como uma imagem em tons de cinza), o mapa de profundidade é dividido nas quatro regiões do *quadrimap*, sendo que os pixels, são classificados em relação aos seus valores de profundidade, do menor para o maior, na seguinte ordem: *Foreground*, Provável *Foreground*, Provável *Background* e *Background*.

O valor do limiar que divide cada pixel entre as quatro regiões é determinado pelo histograma do mapa de profundidade e seus limites mínimo e máximo. Essa informação é levada em conta uma vez que em determinadas situações as áreas com menor ou maior profundidade podem possuir valores de profundidade diferentes de acordo com o vídeo capturado.

Na Figura 36 é apresentado o *quadrimap* criado nesta etapa do processo.

Figura 36 – *Quadrimap* gerado a partir do mapa de profundidade processado



Fonte: elaborada pelo autor.

É importante observar na Figura 36 que as regiões de *Background* e *Foreground* são divididas completamente das regiões desconhecidas (Provável *Background* e Provável *Foreground*), que são classificadas novamente pelo algoritmo de segmentação automática.

6.5 Segmentação automática baseada em *graph cuts*

A partir do vídeo e mapas de profundidade capturados e pré-processados, um algoritmo de segmentação baseado em *graph cuts* foi utilizado. A segmentação é realizada para cada quadro do vídeo, considerando-os como imagens estáticas individuais e processando-os sequencialmente, de forma automática.

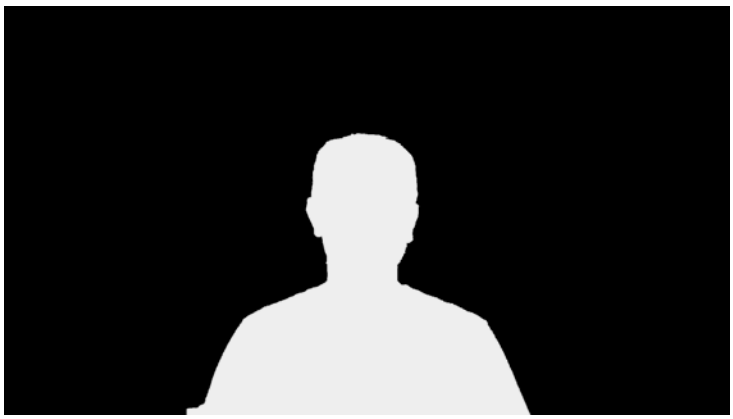
Como entrada para o algoritmo duas informações são necessárias: a imagem RGB que se deseja segmentar e uma classificação prévia de cada pixel dessa imagem em quatro regiões (*foreground*, *background*, provável *background* e provável *foreground*). Essa classificação em quatro regiões é obtida no último passo do pré-processamento, e a estrutura de dados utilizada para tal é chamada de *quadrimap*.

Em algoritmos baseados em *graph cuts*, a imagem que se deseja segmentar é transformada em um grafo, onde cada pixel é representado por um vértice e cada um deles é ligado aos seus pixels vizinhos através de arestas. O peso de cada aresta é definido por qualquer relação entre os pixels; neste caso, o contraste entre os valores de cor entre os pixels e as regiões do quadrimap determinam o peso da aresta.

As regiões definidas como *Foreground* e *Background* são consideradas regiões definitivas, ou seja, que já foram classificadas corretamente e não são alteradas pelo algoritmo de *graph cut*. Os únicos pixels que o algoritmo reclassifica são os identificados como Provável *Background* e Provável *Foreground*. A diferenciação entre essas duas regiões é importante, uma vez que os valores das arestas do grafo utilizado pelo método sofrem alterações decorrentes da região em que se encontram.

Com o grafo criado e com os valores de cada aresta calculados, o algoritmo de *graph cut* realiza a divisão do grafo entre as regiões de *foreground* e *background* por meio de um corte de grafo. Esse corte é definido por uma linha que passa pelas arestas da área desconhecida passando pelas arestas com o menor peso. A partir deste corte, o *quadrimap*, que possuía quatro regiões passa a ter apenas duas, bem definidas: *foreground* e *background*. Na Figura 37 é apresentado o resultado da segmentação utilizando *graph cuts*, apresentado na forma de uma máscara que deve ser aplicada ao respectivo quadro do vídeo para ser segmentado.

Figura 37 – Máscara gerada utilizando *graph cuts* que deve ser aplicada à um quadro do vídeo para realizar a segmentação



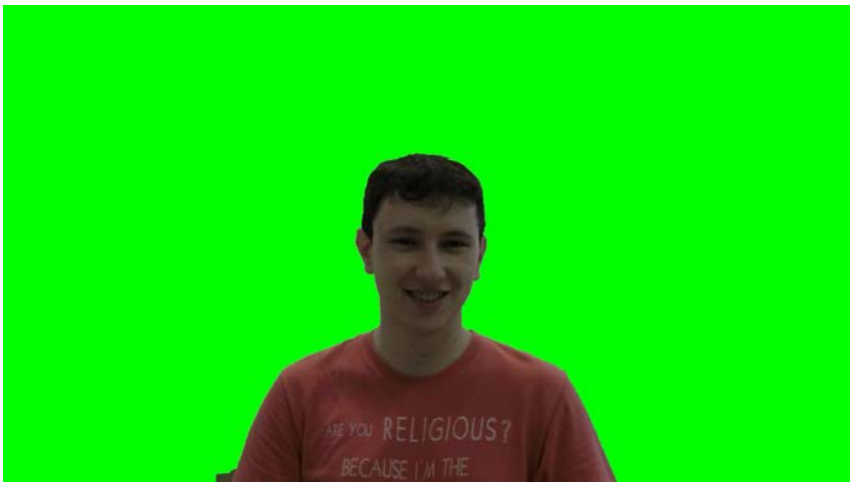
Fonte: elaborada pelo autor.

Na Figura 38 é apresentado um quadro do vídeo segmentado ao aplicar a máscara apresentada na Figura 37. Os pixels classificados como *background* foram substituídos pela cor verde para que os erros de segmentação sejam destacados visualmente.

A imagem apresentada na Figura 38 é o mesmo quadro do vídeo segmentado de maneira mais simples na Figura 30. É possível observar uma melhoria significativa na segmentação utilizando o pré-processamento e o algoritmo de *graph cut*, mesmo utilizando uma abordagem de segmentação automática.

Quando todos os quadros de um vídeo foram considerados, foi possível ver que certos quadros não possuíam erros de segmentação, ao contrário de outros, como no quadro apresentado na Figura 38. A Figura 39 apresenta os resultados do algoritmo em dois

Figura 38 – Resultado da segmentação de um quadro do vídeo utilizando o algoritmo de *graph cut*.



Fonte: elaborada pelo autor.

quadros do mesmo vídeo. Em um dos quadros, o algoritmo não apresentou nenhum erro de segmentação, enquanto no outro, há erros, que estão destacados na figura.

Figura 39 – Exemplo de dois quadros de um vídeo segmentado. (a)Quadro sem erros; (b)Quadro com erros, destacados em azul.

(a) (b)

Fonte: elaborada pelo autor.

Como o objetivo deste trabalho é a geração de *ground truths* de vídeos e não apenas uma segmentação de vídeos utilizando informações de profundidade, como diversos trabalhos presentes na literatura (WANG et al., 2010; WU; BOULANGER; BISCHOF, 2008), um segundo passo para a correção dos erros que foram gerados pelo método automático de segmentação é necessária.

6.6 Segmentação interativa

Como apresentado na seção anterior deste capítulo, utilizando-se imagens RGB, mapas de profundidade, técnicas de processamento de imagens e o algoritmo de corte de grafo, é possível conseguir uma segmentação mais refinada do que ao utilizar apenas o mapa de profundidade proveniente de um sensor como o Kinect e um limiar.

Este método de segmentação pode ser utilizado em diversas aplicações. Métodos semelhantes, utilizando outras tecnologias como câmeras TOF ou até mesmo câmeras infravermelhas, podem ser encontrados na literatura (WANG et al., 2010; WU; BOULANGER; BISCHOF, 2008).

Entretanto, como o objetivo deste estudo não é apenas a segmentação de um vídeo em duas camadas, mas sim a geração de *ground truths* de vídeos, um refinamento da segmentação automática obtida é necessário para a eliminação dos erros gerados.

Devido a diversas situações que podem ocorrer durante a captura do vídeo, algoritmos automáticos de segmentação de imagens e vídeos com fundos não controlados não conseguem um resultado totalmente perfeito. Por esse motivo, optou-se por utilizar uma abordagem interativa de segmentação de imagens para corrigir os erros gerados pelo método automático desenvolvido.

O método interativo desenvolvido baseia-se na exibição dos quadros segmentados do vídeo para o usuário por meio de uma interface, na qual o usuário deve realizar marcações em alguns quadros a fim de refinar o resultado da segmentação automática.

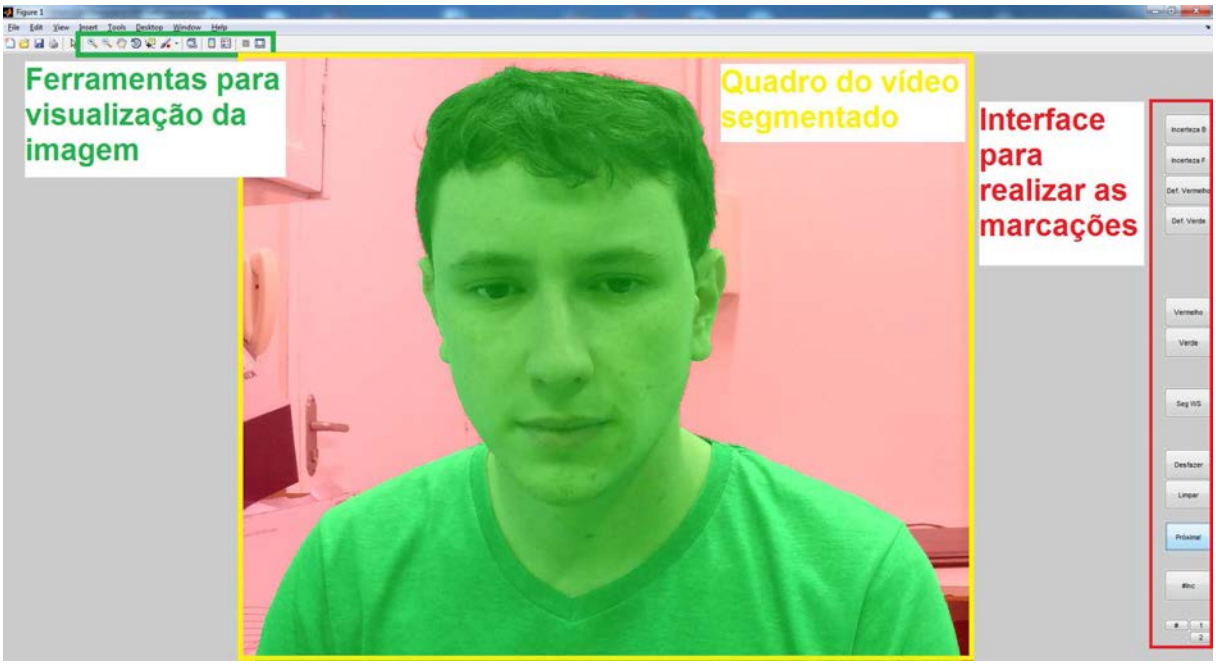
Na Figura 40 é apresentada a interface construída para que o usuário interaja com os quadros do vídeo.

Em vermelho, na Figura 40, estão destacados os comandos utilizados para realizar as marcações, que serão descritas a seguir. Em verde, no canto superior esquerdo da imagem, encontram-se alguns comandos para melhorar a visualização das imagens, como a ferramenta de zoom.

No centro da janela, destacado em amarelo, é apresentado o quadro do vídeo previamente segmentado pelo método automático. Para uma melhor visualização e divisão entre as regiões, optou-se por utilizar uma máscara verde sobre os pixels classificados como *foreground* e uma vermelha para a regiões de *background*.

Para uma melhor visualização, apenas a área de borda, entre o *foreground* e o *background* classificados pelo algoritmo automático, é exibida ao usuário, ou seja, a imagem é enquadrada apenas nas regiões onde o resultado do algoritmo deverá ser corrigido. Isso é realizado para que a imagem seja exibida em um tamanho maior no monitor, destacando detalhes para o usuário. Além disso, a interface permite que o usuário aproxime determinada região da imagem utilizando as ferramentas de zoom presentes na mesma.

Figura 40 – Interface para utilização do algoritmo de segmentação interativo



Fonte: elaborada pelo autor.

Caso determinado quadro do vídeo não tenha que sofrer nenhuma alteração, ou seja, a segmentação automática não gerou nenhum erro, o usuário pode sinalizar que a segmentação já está satisfatória e prosseguir para a inspeção visual do próximo quadro.

Além da inspeção visual, caso a segmentação de um quadro do vídeo possua erros, o usuário pode realizar marcações na imagem para solucionar o problema. As marcações que o usuário pode realizar sobre a imagem, utilizando o mouse, são descritas nas subseções seguintes.

6.6.1 Marcação para segmentação de Bordas usando *Watershed*

O maior número de problemas que ocorrem no método automático de segmentação proposto são áreas que são classificadas erroneamente devido ao contraste da região. Um exemplo de um erro deste tipo é apresentado em destaque na Figura 41. Nesta figura, uma região do *background* foi classificada como *foreground* pelo algoritmo por possuir um contraste maior entre o restante do *background* e o *foreground*. A área classificada erroneamente está destacada em amarelo na imagem.

Para solucionar estes casos, o usuário deve realizar duas marcações sobre a região classificada incorretamente, classificando aquelas regiões como *foreground* e *background*. Na Figura 42 é apresentado um exemplo de marcação feita pelo usuário na região da imagem presente na Figura 41 (região de foreground sinalizada em verde e região do *background* em vermelho). É importante observar que não é preciso muito cuidado ao

Figura 41 – Quadro do vídeo com erro na região de *background*

Fonte: elaborada pelo autor.

realizar as marcações próximo à borda que se deseja segmentar. A região da imagem onde foi realizada as marcações está destacada em amarelo.

Figura 42 – Marcações realizadas para a correção dos erros de segmentação

Fonte: elaborada pelo autor.

Após o usuário selecionar as regiões que devem ser segmentadas novamente, um algoritmo baseado em *watershed* é utilizado, tendo como entrada as marcações realizadas pelo usuário.

Utilizar um método de segmentação baseado em *watershed* foi escolhido, pelo fato de utilizar uma técnica diferente à utilizada na etapa automática de segmentação (*graph cuts*). Com isso, é possível eliminar os erros gerados pelo algoritmo baseado em *graph cut*,

uma vez que o usuário delimita apenas uma região da imagem onde o *watershed* deve ser aplicado. Ao utilizar regiões pequenas e bem delimitadas pelo usuário, o algoritmo de *watershed* se demonstrou bastante eficaz.

Com base nas marcações realizadas, uma subimagem é criada, para que o algoritmo de segmentação baseado em *watershed* seja aplicado apenas na região que contém erros. A partir disso, o algoritmo é executado mais rapidamente e apenas classifica a região de interesse, não alterando as outras regiões da imagem que já foram classificadas pelo algoritmo automático corretamente.

Na Figura 43 é apresentado o resultado da segmentação baseada em *watershed*, na região selecionada pelas marcações apresentadas na Figura 42, assim como a nova segmentação obtida ao refinar essa região com este algoritmo.

Figura 43 – Resultado da segmentação interativa utilizando watershed

Fonte: elaborada pelo autor.

É importante salientar que o usuário pode realizar vários pares de marcações (*foreground* e *background*) em diferentes regiões da imagem para realizar diferentes correções no mesmo quadro.

6.6.2 Marcações Definitivas

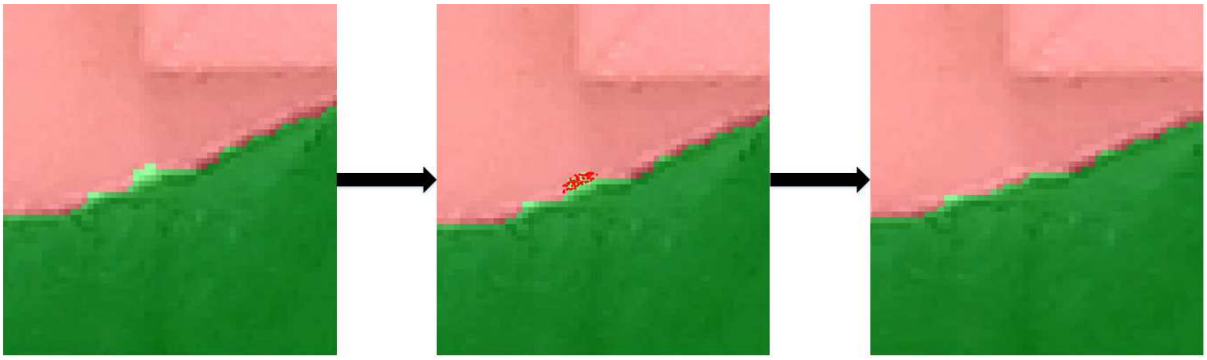
Neste tipo de marcação, o usuário seleciona qual região ele deseja corrigir e manualmente seleciona como os pixels devem ser classificados nessa nova região.

Essa marcação deve ser utilizada pelo usuário em regiões pequenas, onde apenas poucos pixels foram classificados de maneira errada pelo algoritmo automático, uma vez que esta correção é realizada pixel à pixel. Esta tarefa pode ser considerada difícil e

demorada em regiões grandes da imagem e deve ser utilizada apenas em áreas onde não é possível selecionar as duas regiões bem definidas ao utilizar as marcações para segmentação de bordas utilizando *watershed*.

Na Figura 44 é apresentado um exemplo de quadro onde a segmentação foi realizada erroneamente pelo algoritmo de segmentação automática, assim como as marcações realizadas pelo usuário e a nova segmentação obtida.

Figura 44 – Exemplo de marcações definitivas



Fonte: elaborada pelo autor.

6.6.3 Marcações otimizadas

Por se tratar de uma segmentação de vídeo e não de imagens estáticas, foi constatado que em quadros próximos do vídeo, onde as imagens do *foreground* e *background* não sofrem tantas alterações, as regiões classificadas incorretamente também são semelhantes. Para que o usuário não necessite realizar a mesma marcação em vários quadros seguidos do vídeo foi criada uma forma de se otimizar essas marcações.

Este tipo de marcação foi dividido em duas categorias: as marcações pré-programadas e as marcações automáticas.

As marcações pré-programadas são uma variação das marcações para segmentação de bordas e também utiliza o algoritmo de *watershed*, porém podem ser salvas para o uso em outros quadros do vídeo que possuam um erro na mesma região de borda entre o *foreground* e o *background*. Elas foram criadas para que o usuário precise apenas desenhar as marcações uma vez e então salvar esse padrão em um espaço de memória (definidos por números, variando de 1 a 5). Em um próximo quadro, que possua erros semelhantes a um quadro anterior, basta um clique sobre o número em que a marcação foi armazenada, para que o quadro também seja marcado. Isso evita que, caso diversos quadros do vídeo possuam o mesmo erro, o usuário tenha que desenhar o mesmo padrão de marcações em todos eles.

As marcações automáticas são parecidas com as marcações pré-programadas, porém, estas realizam a análise de necessidade de marcações, sem que o usuário necessite analisar se o quadro seguinte possui o mesmo erro que foi corrigido no quadro anterior. Após o usuário realizar uma marcação para corrigir determinado erro de segmentação, ele pode, então, cadastrar esta marcação como sendo uma "marcação inteligente". A partir deste momento, o algoritmo desenvolvido analisa todos os quadros que forem carregados após o cadastro da marcação inteligente, e caso as regiões das imagens do quadro seguinte obedeçam uma série de regras, o quadro também é marcado e segmentado, uma vez que o algoritmo determina que o quadro em questão também possui o erro cadastrado anteriormente.

As regras para que um quadro também seja marcado, automaticamente, utilizando as marcações automáticas dependem dos valores dos pixels marcados pelo usuário. Um quadro também deve ser marcado pelo marcador inteligente, caso os respectivos pixels do novo quadro, em relação aos pixels do quadro marcado, sejam:

- Pertencentes à mesma região (*foreground* ou *background*), exclusivamente; e
- Possuam os mesmos valores de profundidade (ao analisar o mapa de profundidade original capturado pelo dispositivo e pré-processado pelas técnicas descritas na Seção 6.4).

Caso a verificação seja verdadeira para as duas regras descritas acima, o quadro é automaticamente marcado para a segmentação.

6.6.4 Inclusão da área desconhecida

O último passo para a geração dos *ground truths* de vídeos é a composição do *trimap*. Sendo um *trimap*, espera-se que o *ground truth* contenha três regiões bem definidas: *foreground*, *background* (que já foram divididas pelas etapas anteriores) e uma área desconhecida.

Essa área desconhecida, representa os pixels em que não há certeza se podem ser classificados totalmente em uma das outras duas regiões.

Na prática, essas áreas são representadas por pixels que são compostos por uma combinação de cores entre o *foreground* e o *background* e, portanto, não podem ser classificados com total certeza em nenhuma das duas regiões. Exemplos que podem ser listados são objetos translúcidos (como vidro e plástico), fumaça, e objetos muito finos, como fios de cabelo e pelos de animais.

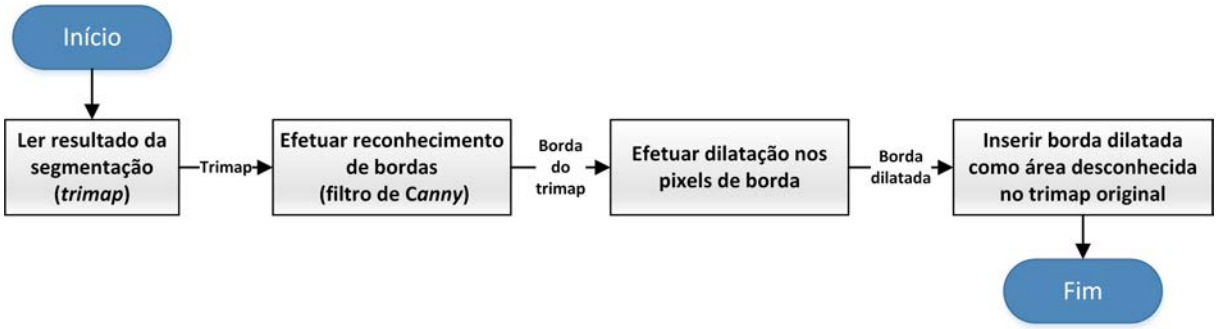
Esses casos, quando aparecem em algum quadro do vídeo, devem ser selecionadas manualmente pelo usuário. Como nas marcações pré-programadas, essas regiões selecio-

nadas como desconhecidas pelo usuário também podem ser salvas para sua utilização em outros quadros do vídeo que possuem a mesma região desconhecida em pixels próximos aos selecionados manualmente. Por esse fato, uma maneira de movimentar as regiões foi criada. A partir dessa função, o usuário pode arrastar a região desenhada anteriormente para qualquer região do quadro e assim, acompanhar a região que deve ser tratada como área desconhecida.

Outra região dos quadros do vídeo, em que não se tem certeza que determinados pixels pertençam as regiões definidas como *foreground* e *background*, é a borda entre essas regiões. Isso acontece pois, muitas vezes, devido à resolução das imagens utilizadas, que fazem com que um pixel represente uma combinação de cores entre o objeto do *foreground* e *background*. Neste caso, os pixels devem ser considerados como pertencentes à área desconhecida, pois, claramente, devem ser desconsiderados em uma futura análise de outros algoritmos.

A inclusão desta área desconhecida nos quadros do vídeo é realizada automaticamente. Uma borda com tamanho pré-definido é inserida entre a região de *foreground* e *background*. O algoritmo que realiza esse procedimento, é descrito em função de suas etapas, na Figura 45.

Figura 45 – Etapas do algoritmo de inclusão da área desconhecida no ground truth

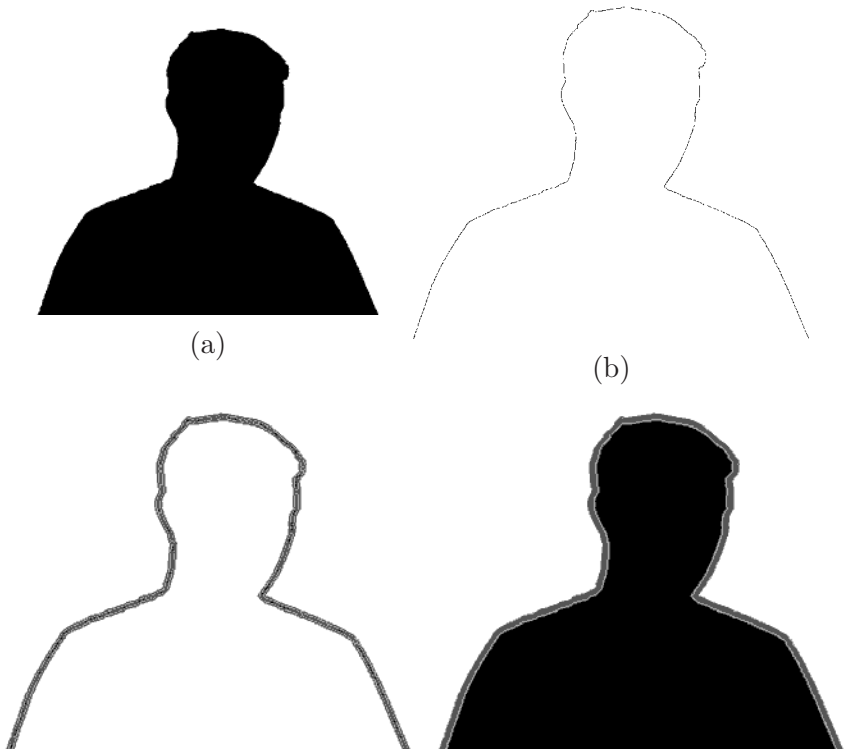


Fonte: elaborada pelo autor.

Na Figura 46 são apresentados os resultados obtidos em cada etapa ao se aplicar o algoritmo acima em um quadro de um vídeo.

Após a inclusão desta informação em todos os quadros do vídeo, as informações são gravadas em disco, ou seja, o *ground truth* de todos os quadros do vídeo são obtidos.

Figura 46 – Resultados obtidos em cada etapa do algoritmo de inclusão da área desconhecida; (a) Segmentação binária obtida pelo algoritmo interativo; (b) Borda extraída utilizando o filtro de detecção de bordas; (c) Borda dilatada; (d) *Trimap* gerado, utilizando a borda dilatada como área desconhecida.



(a) (b) (c) (d)

Fonte: elaborada pelo autor.

7 Testes e Análise dos Resultados

Este capítulo apresenta os resultados obtidos a partir de testes realizados utilizando-se a implementação do método proposto. Este capítulo é dividido em quatro seções, nas quais: o ambiente experimental utilizado é descrito, os detalhes de cada teste são elencados e os resultados obtidos nestes testes são apresentados. Finalmente, uma breve comparação e discussão sobre os *ground truths* gerados, em relação à outras bases de dados presentes na literatura, são apresentadas.

7.1 Ambiente experimental

Alguns testes foram conduzidos durante a implementação do método proposto a fim de analisar o desempenho dos algoritmos desenvolvidos e a usabilidade do sistema.

Para a realização dos testes, foi utilizado um ambiente experimental, composto de um computador, um sensor Kinect (Modelo de Xbox 360) e uma câmera GoPro Hero 3 Black Edition.

O computador utilizado possui a seguinte configuração:

- *Placa Mãe:* Asus P6TD DELUXE;
- *Processador:* Intel Core i7 CPU 930 (2.80GHz);
- *Memória RAM:* 3 pentes de 4 GB Corsair (600 Mhz);
- *Disco rígido:* 1 TB;
- *Placa de vídeo:* NVIDIA GeForce GTX 650;
- *Placa de captura de vídeo:* Avermedia Live Gamer HD;
- *Sistema Operacional:* Microsoft Windows 7 64-bits Service Pack 1

Alguns vídeos foram capturados pelos dispositivos, todos no mesmo ambiente. O ambiente não possui nenhuma característica marcante, a não ser uma sala fechada, possuindo iluminação artificial (por lâmpadas fluorescentes comuns) e iluminação natural, proveniente de algumas janelas, presentes em um dos lados da sala.

Maiores detalhes, específicos de cada vídeo capturado, são descritos na próxima seção.

7.2 Testes realizados

Além dos testes realizados no processo de implementação do algoritmo, durante os quais necessitou-se gerar alguns vídeos para testes de funcionalidades e algoritmos específicos, foram gravados 4 vídeos, com características diferentes, para avaliar-se a eficácia do método implementado.

Todos eles foram gravados no mesmo ambiente experimental, utilizando os mesmos dispositivos. Em relação às semelhanças entre eles, têm-se a resolução do vídeo, de 1920x1080 pixels e a taxa de 30 quadros por segundo. Porém, cada um deles foi gravado simulando uma situação diferente.

É importante deixar claro que em todos os vídeos há uma pessoa em primeiro plano, que corresponde ao *foreground* a ser extraído. Além disso, tem-se que o enquadramento desta pessoa simula um *video chat*, ou seja, trata-se de um enquadramento focando o rosto da pessoa e a parte superior de seu torso.

As diferenças de cada um, são listadas a seguir:

- *Vídeo 1:* Vídeo padrão, com *background* estático, porém, não-controlado e movimentação brusca no *foreground*; Duração: 10 segundos
- *Vídeo 2:* Vídeo contendo fundo não controlado e dinâmico (com movimentação no *background*; Duração: 10 segundos;
- *Vídeo 3:* Vídeo com variação na iluminação do ambiente (algumas luzes do ambiente são desligadas e ligadas durante a gravação do vídeo); Duração: 10 segundos;
- *Vídeo 4:* Vídeo padrão, como o Vídeo 1, porém, com uma duração maior, simulando uma pessoa conversando em uma aplicação de *video chat*; Duração: 20 segundos.

7.3 Resultados obtidos

Nesta seção, os resultados obtidos para cada um dos testes são apresentados e comparados.

Os resultados estão separados nas duas seções a seguir, nas quais são apresentados o desempenho do algoritmo automático e do algoritmo interativo respectivamente.

7.3.1 Desempenho do algoritmo automático

Como apresentado no Capítulo 5, um dos passos do método proposto neste trabalho é um algoritmo de segmentação automática de vídeo, baseado em informações de cores e profundidade.

Para que o desempenho deste algoritmo pudesse ser avaliado, cada pixel de cada quadro segmentado de um vídeo foi classificado em quatro categorias, apresentadas a seguir. Para a classificação dos pixels entre estas categorias foi utilizado o *ground truth* do respectivo vídeo, que foi obtido com o método desenvolvido.

- *Verdadeiros Foregrounds (VF)*: pertencente ao *foreground* e classificado corretamente;
- *Falsos Foregrounds (FF)*: pertencente ao *foreground* e classificado incorretamente;
- *Verdadeiros Backgrounds (VB)*: pertencente ao *background* e classificado corretamente;
- *Falsos Backgrounds (FB)*: pertencente ao *background* e classificado incorretamente;

O desempenho do algoritmo é mensurado pela sua acurácia A, que é definida pela Equação 7.1:

$$A = \frac{VF + VB}{VF + VB + FF + FB} \tag{7.1}$$

onde as variáveis representam os números de pixels que foram classificados nas categorias apresentadas anteriormente.

É importante frisar que, no caso dos pixels que pertencem à área desconhecida do ground truth, os mesmos são ignorados, ou seja, não são contabilizados em nenhuma das quatro categorias acima descritas. Na Tabela 1, a seguir, a acurácia obtida para cada um dos testes efetuados é apresentada.

Tabela 1 – Resultados do algoritmo de segmentação automático aplicado nos vídeos-teste capturados.

Testes	**Pixels classificados corretamente como *background* (VB) em %**	**Pixels classificados corretamente como *foreground* (VF) em %**	**Acurácia em %**
Vídeo 1	21,98	77,17	99,16
Vídeo 2	20,03	76,86	96,89
Vídeo 3	25,98	73,17	99,14
Vídeo 4	23,65	75,20	98,85

Fonte: elaborada pelo autor.

O valor de acurácia encontrado para todos os vídeos foi satisfatório, chegando a mais de 96% em todos os vídeos utilizados.

Um dos fatores que faz com que o resultado do algoritmo automático seja bastante próximo ao *ground truth*, se dá pelo fato de que os erros que são encontrados nessa

segmentação são, em sua maioria, próximos à região de borda entre o *background* e *foreground*. Isso ocorre pois o método utiliza as informações de profundidade capturadas, isola o objeto de interesse do restante da cena, e não permite que ocorram erros em outras regiões da imagem.

O resultado deste algoritmo, utilizado na etapa automática do método proposto, é constituído de uma segmentação *bilayer*, ou seja, que classifica a imagem apenas em *foreground* e *background*. Como a comparação é realizada entre este resultado e seu respectivo *ground truth*, que é constituído de um *trimap*, a área desconhecida presente no *trimap* é desconsiderada. Assim, erros na região da área desconhecida do *ground truth* são desconsiderados no cálculo da acurácia.

Como a maioria dos erros encontrados pertencem à região de borda e esta é desconsiderada no cálculo de acurácia, por fazer parte da região desconhecida do *ground truth*, resultam em ótimos resultados (acima de 96%)

7.3.2 Desempenho da etapa interativa

Em relação à etapa interativa, não há uma maneira de se comparar o seu desempenho em função de pixels classificados corretamente ou não, como no caso do algoritmo automático.

Isso se dá, pelo fato de que esta etapa por ser a última no método proposto, gera *ground truths* de vídeos, os quais não possuem erros de segmentação que possam ser mensurados.

Além disso, por se tratar de um método interativo, mesmo que um quadro de um vídeo seja segmentado por dois usuários diferentes, é possível que a segmentação seja diferente, pois os usuários podem interagir de maneiras diferentes com o algoritmo implementado. Isso não quer dizer que uma das segmentações esteja incorreta, pois é o usuário que deve julgar estas segmentações, lembrando que só depende do usuário identificar os erros e refinar a segmentação, independendo de quantas interações forem necessárias para isso. O importante é que no final dessas interações, todos os quadros de um vídeo sejam segmentados com tamanha precisão a ponto de serem considerados *ground truths*.

Como o objetivo principal do método é reduzir o tempo, o esforço e as interações necessárias para a obtenção do *ground truth* de vídeos, essas características foram mensuradas durante a realização desta etapa do método, na geração dos *ground truths* dos vídeos descritos na Seção 7.2 deste capítulo.

As características analisadas para cada teste, foram:

- *Duração:* Tempo decorrido durante a etapa de segmentação interativa;

- *Quadros alterados:* Número de quadros onde foi preciso realizar alguma alteração;
- *Marcações manuais:* Número de marcações realizadas manualmente (desenhando sobre o quadro);
- *Marcações em áreas desconhecidas:* Número de marcações realizadas manualmente para sinalizar uma região desconhecida.
- *Marcações otimizadas:* Número de marcações realizadas automaticamente (realizadas em apenas um clique);
- *Marcações otimizadas em áreas desconhecidas:* Número de marcações realizadas automaticamente para sinalizar uma região desconhecida; e
- *Média de marcações:* Média de marcações diferentes realizadas em um mesmo quadro de vídeo (incluindo os quadros que não tiveram alterações).

A partir destas características, os vídeos-teste gravados e segmentados previamente pelo algoritmo automático, foram submetidos ao processo de segmentação interativa. Durante o processo de segmentação, cada ocorrência das marcações descritas acima foi anotada de forma manual, individualmente para cada um dos vídeos. Os valores obtidos, para cada característica, de cada vídeo, são apresentados na Tabela 2.

Tabela 2 – Resultados do algoritmo interativo de segmentação aplicado nos vídeos-teste capturados

Testes	Duração*	Quadros alterados	Marcações manuais	Marcações otimizadas	Marcações em áreas desconhecidas (manuais)	Média de marcações por quadro
Vídeo 1	5,5 horas	264/300	134	284	160	1,93
Vídeo 2	6 horas	278/300	138	327	202	2,22
Vídeo 3	5 horas	238/300	98	252	196	1,82
Vídeo 4	10 horas	507/600	227	503	396	1,88

* Duração do processo levando em consideração o tempo para realizar a contagem das marcações efetuadas.

Fonte: elaborada pelo autor.

Analisando os resultados apresentados na Tabela 2 é possível observar que a maioria das marcações realizadas, tanto para corrigir erros de segmentação, quanto para inclusão de áreas desconhecidas no *trimap*, foram do tipo automática.

Isso reforça a ideia de que, por se tratar de um vídeo onde não há grande movimentação do objeto de interesse e por este possuir uma taxa de quadros por segundo que possui continuidade, os erros de segmentação possuem semelhanças em quadros consecutivos do vídeo. A partir disso, as mesmas marcações podem ser realizadas automaticamente,

o que reduz o número de marcações manuais, que precisam de mais tempo e esforço por parte do usuário.

Referente ao Vídeo 1, em alguns quadros do vídeo, devido ao movimento brusco no *foreground*, alguns quadros possuem regiões borradas (movimentação da mão da pessoa filmada). As regiões dos quadros do vídeo que possuem esta característica foram consideradas como áreas desconhecidas. Além de não ser possível classificar os pixels dessa região, pois os mesmos possuem informações de cor tanto do *foreground* quanto do *background*, imagina-se que essa região será tratada pelo processo de *matting*.

No Vídeo 2, para simular uma movimentação no plano de fundo, uma pessoa caminha ao fundo da cena, enquanto outra está sentada, representando o *foregound*. Em alguns quadros do vídeo, nos momentos em que a pessoa que está no *background* se aproxima da região de borda do *foreground*, alguns erros foram notados, mas que puderam ser facilmente eliminados utilizando a etapa interativa do método proposto.

No Vídeo 3, mesmo com alterações na iluminação, realizada ao apagar e acender a principal fonte de luz do ambiente, não houveram alterações em relação a segmentação, pois cada quadro do vídeo é analisado individualmente. A única observação a ser feita é que mais erros apareceram nos quadros menos iluminados, pelo fato de que os quadros capturados possuem menos contraste.

O Vídeo 4, capturado com um número de quadros maior, se comportou como o primeiro vídeo durante o processo de segmentação. Foi possível observar que o número de interações necessárias para se obter o *ground truth* foi proporcional à duração do vídeo.

Em relação ao método como um todo, por possuir uma etapa interativa, foi possível obter todos os *ground truths* de todos os quadros dos vídeos, conforme seu objetivo principal.

Como outra forma de validar os resultados obtidos, mais precisamente a qualidade dos *ground truths*, uma breve comparação à outras bases de dados que contém esta informação foi conduzida e é apresentada a seguir.

7.4 Comparação entre *ground truths*

O resultado ao se utilizar o método criado e a ferramenta desenvolvida, é a obtenção de *ground truths* de vídeos com fundos não controlados.

O método se mostrou bastante interessante, uma vez que permite a geração de *ground truths* de vídeos que possuem melhores características ao se comparar com *ground truths* disponibilizados por alguns grupos de pesquisas.

Uma comparação entre as características dos vídeos gerados neste trabalho e outros da literatura foram conduzidas. Os vídeos de duas bases de dados foram utilizados nesta

comparação. A Tabela 3 apresenta a comparação entre as características entre os *ground truths* gerados a partir do método desenvolvido e bases de vídeos que disponibilizam *ground truths* de vídeos para uso acadêmico.

Tabela 3 – Características dos *ground truths* gerados e das bases de vídeos presentes na literatura

Base de Vídeos com ground truths	**Resolução dos vídeos**	**Número de quadros que possuem ground truths**	**Câmera pode ser movimentada**	**Vídeos com foregrounds diversos (além de pessoas em primeiro plano)**
Ground Truths gerados utilizando o método desenvolvido	1920x1080	Para todos os quadros (100%)	Sim	Sim
Base de vídeos da Microsoft	320x240	a cada 5 ou 10 quadros (10 a 20% do total)	Não	Não
Base de vídeos disponibilizados junto com o algoritmo TOFCut	320x240	a cada 4 quadros (25% do total)	Sim	Não

Fonte: elaborada pelo autor.

Uma das principais características que os vídeos gerados neste trabalho diferem de outros trabalhos encontrados na literatura é que para todos os quadros dos vídeos existe o respectivo *ground truth*; nos outros trabalhos analisados (CRIMINISI et al., 2006; WANG et al., 2010) apenas alguns quadros possuem esta informação.

Segundo alguns trabalhos presentes na literatura, o *ground truth* de apenas alguns quadros é suficiente para analisar a qualidade de vídeos segmentados por diferentes métodos de segmentação, uma vez que utilizando uma amostra dos quadros, estatisticamente, os quadros analisados possuem os erros presentes em todo o vídeo. Porém, em determinadas métricas de qualidade de vídeo, onde as regiões segmentadas de maneira errada não são analisados por um algoritmo de maneira objetiva, mas sim subjetivamente por grupos de pessoas, o *ground truth* de todos os quadros do vídeo é interessante.

Isso se dá pelo fato de que nestas métricas, os vídeos contendo os erros de segmentação são exibidos à usuários para que os mesmos classifiquem quais deles incomodam mais que outros. Como apenas os quadros que possuem o *ground truth* são exibidos para que seja possível realizar a análise da qualidade de vídeo, ao se utilizar as bases de vídeos existentes (CRIMINISI et al., 2006; WANG et al., 2010) o resultado pode não ser satisfatório. Os vídeos exibidos aos usuários por possuírem uma taxa de quadros pequena, (menor que 24 quadros por segundo), não passam a sensação de continuidade à quem está assistindo, o que pode ser até mais incômodo do que os próprios erros gerados pela segmentação.

Como a análise é realizada imaginando-se vídeos e não imagens estáticas, exibir um vídeo em alta resolução e com uma taxa de quadros alta (que possui continuidade), permite que erros temporais sejam analisados, como o caso do *flickering*, que necessita de uma alta taxa de quadros para ser observado por uma pessoa.

É importante frisar que as outras bases de dados que possuem *ground truth*, apresentadas na Tabela 3 apenas disponibilizam esses dados mas não propõe métodos para a obtenção dos mesmos. O interessante do método proposto é que o mesmo pode ser utilizado para criar bases de vídeos, contendo *ground truths*, com diversas resoluções, taxas de quadros e características diferentes, o que não é possível ao se utilizar bases de dados disponibilizadas por outros autores.

8 Conclusão

Uma etapa importante no processo de análise de qualidade de segmentação de vídeos, é a identificação dos erros que ocorrem no processo de segmentação. Diversas métricas identificam esses erros ao se comparar a segmentação obtida com o *ground truth* do vídeo. Portanto, o *ground truth* de um vídeo deve ser uma referência da melhor segmentação possível de um vídeo, além de possuir certas características, para que a análise possa ser realizada da melhor maneira possível (considerando-se os requisitos de determinada aplicação).

Além de características como a resolução e a taxa de quadros do vídeo, é importante que o *ground truth* possa ser utilizado para validar métodos de segmentação de vídeo, por isso, deve simular situações que possam interferir nos resultados destes algoritmos, chamadas de situações-problema. Essas situações muitas vezes são específicas para determinadas aplicações e devem ser levadas em consideração no momento da captura dos vídeos, pois caso estas não sejam simuladas, os vídeos não serão validados corretamente.

Por se tratar de uma informação que geralmente é obtida manualmente, muitas vezes estas características são ignoradas, tornando os *ground truths*, vídeos com baixa resolução e uma taxa de quadros menor que a dos vídeos utilizados realmente na aplicação, além de não simularem as situações-problema.

O presente trabalho apresenta um método semiautomático para a geração de *ground truths*, que consiste em etapas automáticas e interativas. Este método permite que a partir de um vídeo capturado com uma câmera convencional, associado a um sensor de profundidade, e de marcações realizadas por um usuário, seja possível gerar *ground truths* de vídeos de alta resolução e com uma taxa de quadros igual a dos vídeos segmentados que são analisados.

As maiores contribuições deste trabalho incluem o desenvolvimento do método de geração de *ground truths* de vídeo utilizando uma abordagem de segmentação semiautomática, assim como a implementação do mesmo com base em algoritmos de segmentação de imagens e informações de sensores de profundidade.

O método proposto permite que *ground truths* de vídeos com alta resolução, contendo todos os quadros do vídeo e contendo situações-problema, possam ser gerados, diminuindo as interações necessárias ao se utilizar uma abordagem manual de segmentação de vídeo. Isso é possível ao se utilizar informações de profundidade em conjunto com técnicas iterativas e interativas de segmentação de vídeo.

Os possíveis desdobramentos decorrentes deste trabalho são os seguintes:

- A geração de uma base de vídeos e *ground truths* contendo situações-problema de algoritmos de segmentação de vídeo de fundos não-controlados e a disponibilização desta base para sua utilização pela comunidade científica;
- O aperfeiçoamento da etapa automática do método, ao utilizar diferentes algoritmos de segmentação diferentes, ao contrário de um só, para reduzir o número de erros gerados nesta etapa;
- O aperfeiçoamento da etapa interativa do método, ao utilizar algoritmos de rastreamento para que as marcações realizadas pelos usuários sejam posicionadas automaticamente nos quadros, ao decorrer do vídeo;
- A utilização de sensores de profundidade melhores, como o Kinect V2 da Microsoft, para melhorar o mapa de profundidade utilizado; e
- O aperfeiçoamento da interface da ferramenta desenvolvida para a etapa interativa do método, incluindo diferentes opções de visualização das informações disponíveis, tornando-a semelhante a *softwares* comerciais de edição de vídeo.

Referências

ADOBE SYSTEM INCORPORATED. *Adobe Photoshop CS3 User Guide for Windows and Mac OS*. 2013.

AHN, Y.-K. et al. 3d spatial touch system based on time-of-flight camera. *WSEAS Transactions on Information Science and Applications*, World Scientific and Engineering Academy and Society, v. 6, n. 9, p. 1433–1442, 2009.

ASUS XTION. *Xtion motion sensor for pc*. 2014. Disponível em: http://www.asus.com/Multimedia/Xtion PRO/.

BAI, X.; SAPIRO, G. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *International Journal of Computer Vision*, v. 82, n. 2, p. 113–132, 2009.

BASRI, R.; JACOBS, D.; KEMELMACHER, I. Photometric stereo with general, unknown lighting. *International Journal of Computer Vision*, Kluwer Academic Publishers, v. 72, n. 3, p. 239–257, 2007.

BOOMGAARD, R. van den; BALEN, R. van. Methods for fast morphological image transforms using bitmapped binary images. *CVGIP: Graphical Models and Image Processing*, v. 54, n. 3, p. 252–258, maio 1992.

BOYKOV, Y. Y.; JOLLY, M. P. Interactive graph cuts for optimal boundary region segmentation of objects in n-d images. In: *IEEE International Conference on Computer Vision*, 2001, Vancouver, BC: **Proceedings...** IEEE, 2001. v. 1, p. 105–112.

BRADSKI, G. The opencv library. *Doctor Dobbs Journal*, M and T publishing inc, v. 25, n. 11, p. 120–126, 2000.

BURRUS, N. *Kinect calibration*. 2011. Disponível em: http://nicolas.burrus.name/index.php/Research/KinectCalibration.

CAVALLARO, A.; GELASCA, E. D.; EBRAHIMI, T. Objective evaluation of segmentation quality using spatio-temporal context. In: *International Conference on Image Processing*, 2002, Rochester, New York, USA: **Proceedings...** IEEE, 2002. v. 3, p. 301–304.

CHUANG, Y.-Y. et al. A bayesian approach to digital matting. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001, Kauai, Hawaii, USA: **Proceedings...** IEEE, 2001. v. 2, n. 1, p. 264–271.

CORREIA, P.; PEREIRA, F. Objective evaluation of video segmentation quality. *IEEE Transactions on Image Processing*, v. 12, n. 2, p. 186–200, 2003.

COUPRIE, C. et al. Power watershed: A unifying graph-based optimization framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 33, n. 7, p. 1384–1399, 2011.

CRIMINISI, A. et al. Bilayer segmentation of live video. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, New York, USA: **Proceedings...** IEEE, 2006. v. 1, p. 53–60.

DANCIU, G.; BANU, S.; CALIMAN, A. Shadow removal in depth images morphology-based for kinect cameras. In: *16th International Conference on System Theory, Control and Computing*, 2012, Sinaia, Romênia: **Proceedings...** IEEE, 2012. p. 1–6.

FAUGERAS, O. *Three-dimensional computer vision: a geometric viewpoint*: MIT press, 1993.

FOSTER, J. *The green screen handbook: real-world production techniques*: John Wiley & Sons, 2010.

GASPARI, T. D. et al. ARSTUDIO: a virtual studio system with augmented reality features. In: *Proceedings of the 13th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, 2014, Shenzhen, China: **Proceedings...** New York: ACM, 2014. p. 17–25.

GELASCA, E.; EBRAHIMI, T. On evaluating video object segmentation quality: A perceptually driven objective metric. *IEEE Journal of Selected Topics in Signal Processing*, v. 3, n. 2, p. 319–335, 2009.

GELASCA, E. D. *Full-reference objective quality metrics for video watermarking, video segmentation and 3D model watermarking*. Tese (Doutorado) — Universidade de Trieste, 2005.

GONZALEZ, R. C.; WOODS, R. E. *Digital image processing*: Prentice hall Upper Saddle River, 2002.

HAN, J. et al. Enhanced computer vision with microsoft kinect sensor: A review. *Cybernetics, IEEE Transactions on*, v. 43, n. 5, p. 1318–1334, Oct 2013.

HANSARD, M. et al. *Time-of-flight cameras*: Springer, 2013.

HEIKKILA, J. Geometric camera calibration using circular control points. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 22, n. 10, p. 1066–1077, Oct 2000.

HERRERA C., D.; KANNALA, J.; HEIKKILA, J. Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 34, n. 10, p. 2058–2064, Oct 2012.

HUANG, R.; WANG, X. A new alpha matting for nature image. In: *International Conference on Natural Computation*, 2011, Shangai, China: **Proceedings...** IEEE, 2011. v. 3, p. 1790–1794.

IDDAN, G. J.; YAHAV, G. Three-dimensional imaging in the studio and elsewhere. In: *IV Three-Dimensional Image Capture and Applications*, 2001, San Jose, CA, USA: **Proceedings...** SPIE, 2001. v. 4298, p. 48–55.

KABAYAMA, A. M.; TRABASSO, L. G. Performance evaluation of 3d computer vision techniques. *Journal of the Brazilian Society of Mechanical Sciences*, v. 24, p. 234–238, 07 2002.

LE, A. V.; JUNG, S.-W.; WON, C. S. Directional joint bilateral filter for depth images. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 14, n. 7, p. 11362–11378, 2014.

LEVIN, A.; LISCHINSKI, D.; WEISS, Y. Colorization using optimization. In: *ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH*, 2004, Los Angeles, California, USA: **Proceedings...** New York: ACM, 2004. v. 23, n. 3, p. 689–694.

MILLERSON, G.; OWENS, J. *Television production*: CRC Press, 2012.

MORTENSEN, E. N.; BARRETT, W. A. Intelligent scissors for image composition. In: *Annual Conference on Computer Graphics and Interactive Techniques*, 1995, Los Angeles, CA, USA: **Proceedings...** New York: ACM, 1995. p. 191–198.

OPENKINECT. 2015. Disponível em: <http://openkinect.org/wiki/Main_Page>.

Panasonic. *Panasonic 3d image sensor*. 2010. Disponível em: http://www.panasonic-electric-works.net/D-IMager/.

PAROLIN, A. et al. Bilayer video segmentation for videoconferencing applications. In: *IEEE International Conference on Multimedia and Expo (ICME)*, 2011, Barcelona, Spain: **Proceedings...** IEEE, 2011. p. 1–6.

PORTER, T.; DUFF, T. Compositing digital images. In: *Annual Conference on Computer Graphics and Interactive Techniques*, 1984, Minneapolis, Minnesota, USA: **Proceedings...** New York: ACM, 1984. p. 253–259.

ROTHER, C.; KOLMOGOROV, V.; BLAKE, A. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Transactions on Graphics (TOG)*, v. 23, n. 3, p. 309–314, 2004.

SANCHES, S. R. R. *Avaliação Objetiva de qualidade de segmentação*. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais., 2013.

SANCHES, S. R. R. et al. Bilayer segmentation of live video in uncontrolled environments for background substitution: An overview and main challenges. *IEEE Latin America Transactions*, v. 10, n. 5, p. 2138–2149, Sept 2012.

SANCHES, S. R. R. et al. Mutual occlusion between real and virtual elements in augmented reality based on fiducial markers. In: *IEEE Workshop on Applications of Computer Vision*, 2012, Breckenridge, CO, USA: **Proceedings...** IEEE, 2012. p. 49–54.

SANCHES, S. R. R. et al. Subjective video quality assessment in segmentation for augmented reality applications. In: *Symposium on Virtual and Augmented Reality (SVR)*, 2012, Rio de Janeiro, Brasil: **Anais...** IEEE, 2012. p. 46–55.

SHAO, L. et al. *Computer Vision and Machine Learning with RGB-D Sensors*: Springer, 2014.

SILBERMAN, N.; FERGUS, R. Indoor scene segmentation using a structured light sensor. In: *IEEE International Conference on Computer Vision Workshops*, 2011, Barcelona, Espanha: **Workshops...** IEEE, 2011. p. 601–608.

STAUFFER, C.; GRIMSON, W. Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 22, n. 8, p. 747–757, Aug 2000.

SUN, J. et al. Poisson matting. *ACM Trans. Graph.*, v. 23, n. 3, p. 315–321, ago. 2004.

TIBURZI, F. et al. A ground truth for motion-based video-object segmentation. In: *IEEE International Conference on Image Processing*, 2008, San Diego, CA, USA: **Proceedings...** IEEE, 2008. p. 17–20.

TYKKALA, T. et al. Live rgb-d camera tracking for television production studios. *Journal of Visual Communication and Image Representation*, v. 25, n. 1, p. 207–217, 2014.

VINCENT, L.; SOILLE, P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE transactions on pattern analysis and machine intelligence*, v. 13, n. 6, p. 583–598, 1991.

WANG, J.; COHEN, M. Optimized color sampling for robust matting. In: *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, Minneapolis, MN: **Proceedings...** IEEE, 2007. p. 1–8.

WANG, J.; COHEN, M. F. *Image and video matting: a survey*: Now Publishers Inc, 2008.

WANG, L. et al. Tofcut: Towards robust real-time foreground extraction using a time-of-flight camera. In: *3DPVT*, 2010, Paris, França: **Proceedings...**, 2010.

WU, Q.; BOULANGER, P.; BISCHOF, W. Robust real-time bi-layer video segmentation using infrared video. In: *Canadian Conference on Computer and Robot Vision*, 2008, Windsor, Ont.: **Proceedings...** IEEE, 2008. p. 87–94.

YIN, P. et al. Bilayer segmentation of webcam videos using tree-based classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 33, n. 1, p. 30–42, Jan 2011.

ZHANG, L.; NAYAR, S. Projection defocus analysis for scene capture and image display. *ACM Trans. Graph.*, v. 25, n. 3, p. 907–915, jul. 2006.

ZHANG, Z. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, v. 19, n. 2, p. 4–10, Feb 2012.

Autorizo a reprodução xerográfica para fins de pesquisa.

São José do Rio Preto, 13 / 08 / 15

Assinatura