UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS,
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
APLICADA - PIPCA
NÍVEL MESTRADO

ALESSANDRO PAROLIN

Segmentação de Imagens de Pessoas em Tempo Real para Videoconferências

SÃO LEOPOLDO
2011

ALESSANDRO PAROLIN

Segmentação de Imagens de Pessoas em Tempo Real para Videoconferências

Dissertação submetida à avaliação como requisito parcial para a obtenção do grau de Mestre em Computação Aplicada

Orientador: Prof Dr. Luiz Paulo Luna de Oliveira

SÃO LEOPOLDO
2011

P257s Parolin, Alessandro

Segmentação de imagens de pessoas em tempo real para videoconferências / por Alessandro Parolin. – São Leopoldo, 2011.

76 f. : il. color. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2011.

Orientação: Prof. Dr. Luiz Paulo Luna de Oliveira, Ciências Exatas e Tecnológicas.

1. Videoconferências. 2.Processamento de Imagens. 3.Detecção de faces. 4.Segmentação de objetos. 5.Visão computacional. 6. Interação homem-máquina. I.Oliveira, Luiz Paulo Luna de. II.Título.

CDU 004.773.5
004.932
004.92

Catalogação na publicação:
Bibliotecária Carla Maria Goulart de Moraes – CRB 10/1252

Dedico este trabalho a meus pais,
Nelson e Angela.

AGRADECIMENTOS

Agradeço:

primeiramente aos meus pais, Nelson e Angela, por sempre terem me apoiado, e me incentivado a estudar e perseguir meus sonhos. Foi graças a eles que consegui grandes conquistas da minha vida;

ao meu orientador do primeiro ano do mestrado, Cláudio, que infelizmente não pode continuar me orientando. No entanto, sempre me auxiliou com sábias dicas;

ao meu segundo e atual orientador, Luna, que aceitou me orientar no segundo ano do mestrado, tornando a conclusão deste trabalho possível;

a todos os meus colegas do mestrado, em especial Fábio, Vicente e Anderson, pelos auxílios no decorrer do curso e pelas horas de descontração;

à HP, pelo financiamento de meus estudos no primeiro ano do mestrado;

à Unisinos, por ter me concedido a bolsa Milton Valente.

Quero agradecer também a todos que, de alguma forma, contribuíram com a realização deste trabalho.

"Failure is simply the opportunity to begin again, this time more intelligently."

- Henry Ford

"Imagination is more important than knowledge."

- Albert Einstein

RESUMO

Segmentação de objetos em imagens e vídeos é uma área relativamente antiga na área de processamento de imagens e visão computacional. De fato, recentemente, devido à grande evolução dos sistemas computacionais em termos de *hardware* e à popularização da internet, uma aplicação de segmentação de imagens de pessoas que vem ganhando grande destaque na área acadêmica e comercial são as videoconferências. Esse tipo de aplicação traz benefícios a diferentes áreas, como telemedicina, educação à distância, e principalmente empresarial. Diversas empresas utilizam esse tipo de recurso para realizar reuniões/conferências a nível global economizando quantias consideráveis de recursos. No entanto, videoconferências ainda não proporcionam a mesma experiência que as pessoas têm quando estão num mesmo ambiente. Portanto, esse trabalho propõe o desenvolvimento de um sistema de segmentação da imagem do locutor, específico para videoconferências, a fim de permitir futuros processamentos que aumentem a sensação de imersão dos participantes, como por exemplo, a substituição do fundo da imagem por um fundo padrão em todos ambientes. O sistema proposto utiliza basicamente um algoritmo de programação dinâmica guiado por energias extraídas da imagem, envolvendo informações de borda, movimento e probabilidade. Através de diversos testes realizados, observou-se que o sistema apresenta resultados equiparáveis aos do estado da arte do tema, sendo capaz de ser executado em tempo real a uma taxa de 8 FPS, mesmo com um código não otimizado. O grande diferencial do sistema proposto é que nenhum tipo de treinamento prévio é necessário para efetuar a segmentação.

Palavras-chave: Videoconferências, Processamento de Imagens, Detecção de Faces, Segmentação de objetos, Visão Computacional, Interação homem-máquina.

TITLE: "Real-Time Human Image Segmentation for Videoconferences"

ABSTRACT

Object segmentation has been discussed on Computer Vision and Image processing fields for quite some time. Recently, given the hardware evolution and popularization of the World Wide Web, videoconferences have been the main discussion in this area. This technique brings advantages to many fields, such as telemedicine, education (distance learning), and mainly to the business world. Many companies use videoconferences for worldwide meetings, in order to save a substantial amount of resources. However, videoconferences still do not provide the same experience as people have when they are in the same room. Therefore, in this paper we propose the development of a system to segment the image of a person who is attending the videoconference, in order to allow future processing that may increase the experience of being in the same room. For instance, the background of the scene could be replaced by a standard one for all participants. The proposed system uses a dynamic programming algorithm guided by energies, such as image edges, motion and probabilistic information. After extensive tests, we could conclude that the results obtained are comparable to other state of the art works and the system is able to execute in real time at 8 FPS. The advantage of the proposed system when compared to others is that no previous training is required in order to perform the segmentation.

Keywords: Human Segmentation, Image Processing, Computer Vision, Face Detection.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

FDP - Função Densidade de Probabilidade

FPS - Quadros por segundo (*Frames per second*)

RGB - Sistema de cores aditivas formado por Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*)

ROI - Região de interesse (*Region of Interest*)

TD - Transformada Distância

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A detecção e segmentação de objetos em vídeos ou imagens estáticas já é um tema que vem sendo discutido há algum tempo na área de visão computacional. Muitas pesquisas já foram e ainda vêm sendo feitas propondo novas soluções para esse tipo de problema, devido à grande gama de aplicações, como em imagens médicas (localização de tumores, cirurgia assistida por computador, dermatologia, etc.), localização de objetos em imagens de satélite, inspeção industrial de produtos, localização de pessoas (vídeos de vigilância, sistemas de apoio ao motorista, controle de acesso e navegação de robô, etc.), entre outras. Recentemente, graças à grande evolução em termos de *hardware*, técnicas de detecção/segmentação de objetos/pessoas vêm sendo aplicadas em tempo real.

Uma outra possível aplicação para técnicas de segmentação de imagens de pessoas são as videoconferências. Nesse tipo de evento, pessoas em diferentes lugares podem participar de uma conversa ou reunião, sem precisarem estar no mesmo local, simplesmente utilizando uma câmera conectada a um computador com acesso à internet. De fato, com o crescente avanço e popularização da rede mundial de computadores nos últimos anos, videoconferências vêm se tornando cada vez mais populares, trazendo grandes benefícios para diferentes áreas. A telemedicina, por exemplo, permite que um paciente possa ser examinado por um médico sem que este tenha acesso direto àquele (HARRISON et al., 1996). Professores podem ministrar aulas para seus alunos, mesmo quando estejam viajando para participar de uma conferência. Outro exemplo pode ser encontrado no mundo empresarial. Conforme citado em Davis e Weinstein (2005), videoconferências são uma ótima forma de organizar reuniões entre diferentes empresas a nível global, economizando quantias consideráveis de recursos que seriam gastos em viagens. Ainda, videoconferências já são utilizadas para uso doméstico, para conversas com amigos, parentes, etc.

No entanto, as videoconferências ainda não proporcionam a mesma experiência que as pessoas experimentam quando estão interagindo em um mesmo local. Isso acontece por fatores como dificuldade de manter contato do tipo "olhos nos olhos" entre os participantes, e também pelo fato de que a sensação de imersão dos participantes nem sempre é adequada. Para contornar esse tipo de problema, existem até mesmo soluções comerciais, como por exemplo o *HP Halo*[1]. Esse produto consiste em uma sala com câmeras, telas e microfones estrategicamente posicionados, de forma que os participantes tenham a sensação de estar olhando diretamente para as pessoas enquanto falam. Além disso, o fundo da sala é padrão em todos os ambientes e o sistema de som é do tipo *surround*, aumentando ainda mais a sensação presencial, isto é, de que as pessoas estão no mesmo

[1]Mais informações disponíveis em: http://h71028.www7.hp.com/enterprise/us/en/halo/products.html

local.

Dentro do contexto acima estabelecido, tem-se em vista a proposição de um sistema para delinear locutores em tempo real, a fim de permitir futuros pós-processamentos que proporcionem uma maior sensação de imersão dos participantes no mesmo ambiente. Um exemplo prático é a substituição do fundo das salas de todos os participantes por um fundo padrão. Na área de processamento de imagem, tal operação de delinear exatamente a região de um objeto em uma imagem é chamada de *segmentação*. Portanto, ao longo do trabalho, esse termo será utilizado para denotar a ação de delinear a região da imagem que representa o participante de uma videoconferência.

O grande desafio atual é que a grande parte dos sistemas de segmentação de locutores ainda necessita de algum tipo de treinamento prévio para efetuar a segmentação. Além do mais, muitas vezes também é necessário algum tipo de controle do ambiente para que o sistema seja capaz de identificar as silhuetas das pessoas com eficiência. Ainda, sistemas que apresentam alta precisão geralmente utilizam diversos tipos de sensores, como câmeras estéreo ou infravermelho, o que nem sempre é possível ou viável.

1.2 OBJETIVOS

Tendo em vista a motivação apresentada, a contribuição do presente trabalho é o desenvolvimento de um sistema de segmentação de pessoas, focado para videoconferências, que seja capaz de identificar a silhueta do locutor sem qualquer tipo de treinamento, e em ambientes genéricos, utilizando apenas uma câmera.

Dado que esse trabalho é focado para aplicações em videoconferências, parte-se de algumas premissas, como: (i) somente uma pessoa estará participando de cada vídeo; (ii) o locutor estará de frente para a câmera; (iii) somente a parte superior do corpo da pessoa será levada em consideração; (iv) o fundo da imagem deverá ser estático.

O sistema proposto será desenvolvido na linguagem C++ com auxílio da biblioteca OpenCV [2] (BRADSKI e KAEHLER, 2008). A segmentação será feita utilizando uma técnica de programação dinâmica e diferentes energias, como informação de borda, movimento e probabilidade são utilizadas para determinar a silhueta da pessoa.

O presente trabalho está estruturado da seguinte forma: no capítulo 2 são apresentados os conceitos básicos referentes a processamento de imagens e outras teorias envolvidas no desenvolvimento do trabalho; no capítulo 3 é discorrido o estado da arte do problema em questão; no capítulo 4 o desenvolvimento do sistema proposto é demonstrado com detalhes; no capítulo 5 os resultados obtidos são discutidos; e finalmente, no capítulo 6 é apresentada a conclusão e perspectivas da continuação deste trabalho.

[2] Mais informações disponíveis em: http://opencv.willowgarage.com/wiki/

2 CONCEITOS BÁSICOS

Técnicas de processamento de imagens são implementadas através de algoritmos matemáticos, portanto, é necessário definir matematicamente uma imagem. Uma imagem colorida, por exemplo, pode ser definida por uma função $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, onde o valor $f(x, y)$ retorna três componentes, correspondendo aos canais azul, vermelho e verde do sistema de cores RGB, respectivamente. No caso de imagens monocromáticas (escala de cinza), o valor $f(x, y)$ retorna um escalar que representa a intensidade de luz no ponto (x, y). Em termos computacionais, a função $f(x, y)$ é mapeada para uma matriz $M \times N$ e os valores das coordenadas (x, y) tornam-se discretos.

Outro conceito básico em processamento de imagens e extremamente importante é o histograma. Esse último, de acordo com Gonzalez e Woods (2002), pode ser expressado por uma função discreta $h(r_k) = n_k$, onde r_k é o *k-ésimo* nível de cinza da imagem em uma faixa $[0, L - 1]$, e n_k é a quantidade de píxeis da imagem contendo o nível de cinza r_k. Através de histogramas, diversas operações espaciais podem ser executadas, como a melhoria de imagens, por exemplo. Além disso, ele fornece estatísticas importantes sobre a imagem. Uma operação comum com histogramas é a normalização, dado por $p(r_k) = n_k/n$, para $k = 0, 1, ..., L - 1$.

Como mencionado no capítulo 1, o objetivo deste trabalho é desenvolver um sistema capaz de segmentar imagens de pessoas, ou seja, que saiba determinar quais píxeis de uma imagem pertencem à pessoa e quais pertencem ao fundo da imagem. De acordo com Forsyth e Ponce (2002), não é possível determinar se um *pixel* pertence ao primeiro ou segundo plano através de sua simples análise, o que cria a necessidade do desenvolvimento de técnicas mais complexas. Ainda, segundo o autor, para aplicações de segmentação de pessoas, em especial, não existe uma técnica consolidada na literatura, porém, algumas regras gerais são claras. Deve-se procurar por regiões na imagem que representam a pessoa e a partir de então, juntá-las. A fim de ter uma estimativa inicial de onde a pessoa está, detecta-se a face da pessoa e a seguir, efetua-se a segmentação propriamente dita.

2.1 DETECÇÃO E RASTREAMENTO DE FACE

Dentre as diversas técnicas de detecção de face, a mais utilizada e já consolidada na literatura é a proposta por Viola e Jones (2001). Com essa técnica, a detecção é feita através da análise de características do tipo *Haar* extraídas da imagem. Inicialmente, a imagem em questão é convertida para escala de cinza e várias subjanelas, com tamanho variável, são deslocadas sobre ela, dentro das quais diversas características *Haar* são extraídas.

Alguns exemplos desse tipo de característica podem ser vistos na Figura 2.1. Para características com dois retângulos, o valor extraído é a diferença entre a soma dos píxeis

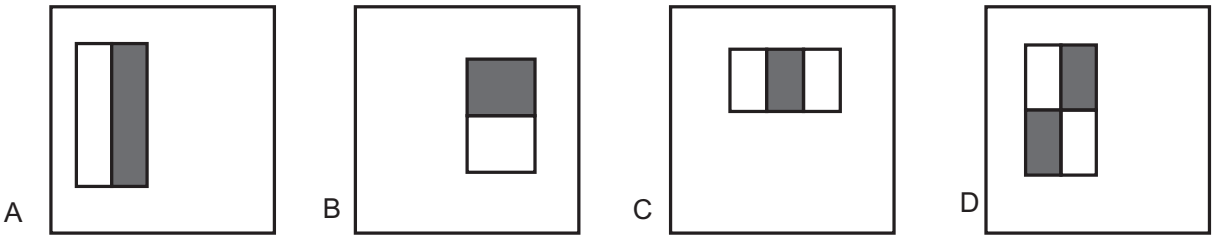


Figura 2.1: Exemplo de características do tipo *Haar*. (A) e (B) apresentam características com dois retângulos. (C) apresenta uma característica com três retângulos e (D) apresenta uma com quatro retângulos.

sob as duas regiões retangulares. As regiões possuem mesmo tamanho e formato e são adjacentes vertical ou horizontalmente. Para uma característica de três retângulos, o resultado é a soma dos píxeis sob as duas regiões retangulares externas subtraída da soma dos píxeis da região retangular central. Finalmente, para uma característica com quatro retângulos, o resultado é a diferença entre os pares de retângulos diagonais.

Após a etapa do cálculo das características, as subjanelas são submetidas a uma cascata de classificadores. Essa cascata consiste em diversos nodos, sendo que cada um deles possui um classificador treinado pelo algoritmo *Adaboost* (FREUND e SCHAPIRE, 1995). A idéia básica dessa técnica é que os primeiros nodos da cascata eliminem grande parte das subjanelas que não representam uma face, o que torna o algoritmo mais rápido. Se a subjanela for aprovada, ela passa para o próximo nodo. Os nodos que estão próximos do final da cascata possuem uma taxa de falsos positivos mais baixa. Um esquema desse sistema pode ser visto na Figura 2.2. De acordo com Bradski e Kaehler (2008), de 70% a 80% das janelas que não contém faces são eliminadas logo nos dois primeiros nodos do classificador.

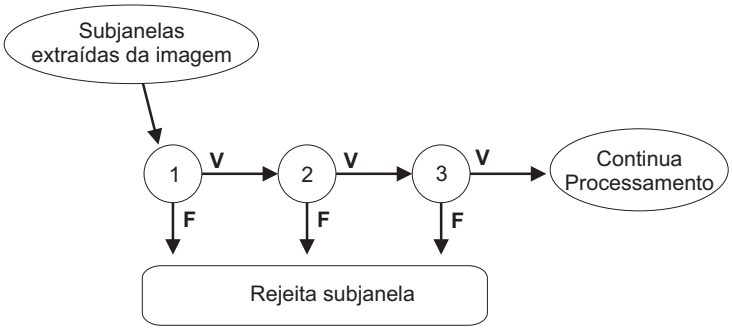


Figura 2.2: Exemplo do funcionamento da cascata de classificadores no algoritmo *Adaboost*. Adaptado de Viola e Jones (2001).

Além da cascata de classificadores mencionada acima, o conceito de imagem integral foi introduzido, o que aumentou consideravelmente o desempenho do detector de faces.

A imagem integral é uma representação intermediária do quadro em questão, dada por:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \tag{2.1}$$

onde $i(x', y')$ é a imagem original.

Um exemplo da imagem integral pode ser visto na Figura 2.3, na qual o valor do *pixel* no ponto 1 é a soma das intensidades do retângulo A. O valor no ponto 2 é a dado por A+B. O valor no ponto 3 é dado por A+C, e no ponto 4 é obtido através de A+B+C+D. A soma do retângulo D pode ser computada através de 4+1-(2+3). Portanto, qualquer característica *Haar* pode ser computada em tempo constante.

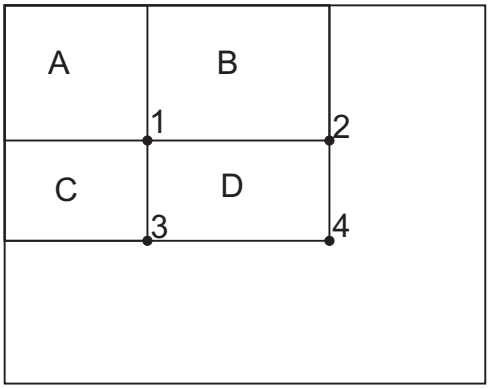


Figura 2.3: Representação da Imagem Integral. Adaptado de Viola e Jones (2001).

No entanto, uma das desvantagens do modelo proposto por Viola e Jones (2001) está no fato de que não há uma memória que armazene a posição da face nos quadros anteriores, portanto a face deve ser novamente detectada a cada quadro. Devido a ruídos, ocasionalmente a face detectada em um determinado quadro pode não ser a mesma detectada no quadro anterior ou ser detectada com um tamanho não condizente com o anterior, o que compromete a coerência temporal do sistema. Ainda, essa técnica pode apresentar falsos positivos ao longo do vídeo. Dependendo da aplicação, os fatores citados não são um problema. Porém, no caso deste trabalho, a segmentação final da imagem da pessoa depende fortemente do resultado do detector de face nos quadros inicias, como será explicado mais adiante.

Para sobrepor o problema descrito anteriormente, um rastreador de face robusto foi proposto em Bins et al. (2009), o qual se baseia na técnica mostrada anteriormente. Em um primeiro momento, a face é detectada utilizando a solução de Viola e Jones (2001), e após a detecção inicial, características KLT (SHI e TOMASI, 1993) são distribuídas sobre a região da face detectada, as quais serão rastreadas ao longo do vídeo. Neste modelo, a face é rastreada utilizando um retângulo centrado no ponto central da face. Uma comparação entre as técnicas propostas por Viola e Jones (2001) e Bins et al. (2009) pode ser vista na Figura 2.4. Na primeira linha da imagem pode-se ver o resultado do detector de face

Figura 2.4: Comparação entre o detector e rastreador de face. Primeira linha: Detector de face proposto por Viola e Jones (2001). Segunda linha: Rastreador de face proposto por Bins et al. (2009).

em diferentes quadros ao longo de um vídeo. É possível perceber que o tamanho da face detectado varia, mesmo quando a pessoa se mantém na mesma distância em relação a câmera. Além disso, quando a face está muito inclinada, o detector não é capaz de encontrá-la. Já na segunda linha, pode-se ver o resultado do rastreador de faces, o qual foi capaz de obter a posição e tamanho da face do locutor, mesmo quando essa está inclinada.

O algoritmo de rastreamento das características KLT é capaz de rastrear diversas características com um custo computacional baixo. O rastreamento é feito utilizando um filtro WVMF (*Weighted Vector Median Filter*), o qual faz parte de uma classe de filtros baseados em médias ponderadas. Sua vantagem está no fato de que seus pesos podem ser facilmente selecionados de forma que *outliers* (características KLT que estão fora da face) sejam praticamente descartadas. Ainda, esse rastreador é capaz de lidar com mudanças na escala, que corresponde a uma translação 3D, de forma satisfatória.

2.2 DETECÇÃO DE BORDAS

Dado que a face foi detectada, tem-se agora uma estimativa inicial de onde a pessoa está. Portanto, é necessário obter mais informações da imagem a fim de determinar sua silhueta. Uma informação relevante e direta para ser obtida são as bordas da imagem, já que, conforme apresentado em Gonzalez e Woods (2002), detecção de bordas é o método mais comum para obter descontinuidades que tenham algum significado em imagens em tons de cinza. De acordo com o autor, uma borda é um conjunto de píxeis situados entre duas regiões de tonalidades diferentes. Na prática, nem sempre é possível obter uma distinção satisfatória entre os objetos devido a diversos fatores, como baixa qualidade do sistema de aquisição da imagem, iluminação da cena, taxa de amostragem, etc. Isso faz com que a borda se torne borrada ou, muitas vezes, até de difícil percepção.

Diferentes técnicas de detecção de bordas já existem e estão consolidadas na lite-

ratura. Dentre as mais utilizadas, pode-se citar o filtro de Sobel (GONZALEZ e WOODS, 2002), utilizada em diversos sistemas, como em Jabri et al. (2000) e Wu e Nevatia (2005).

O filtro de Sobel utiliza derivadas primeiras para obter os contornos dos objetos contidos em uma imagem. Tendo em vista que uma borda é uma variação brusca de intensidade de cinza na imagem, a derivada primeira é uma técnica adequada já que produz uma resposta forte nesses casos, como pode ser visto na Figura 2.5. Conforme já foi citado, devido a diversos fatores, uma borda dificilmente é bem definida como no primeiro caso da figura. Geralmente, uma borda apresenta um perfil de uma rampa inclinada, como no segundo caso. A inclinação da borda é inversamente proporcional ao seu grau de suavidade.

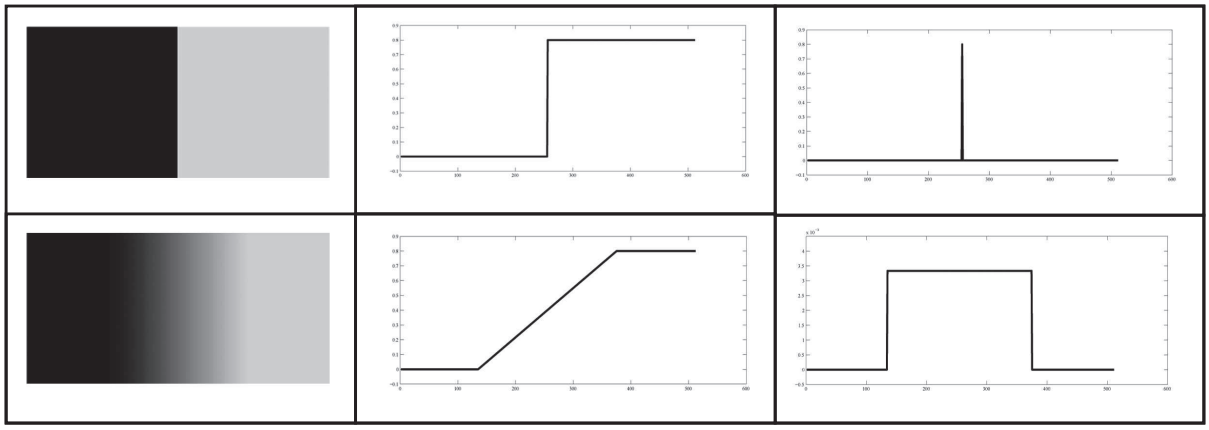


Figura 2.5: Exemplo de bordas. Primeira coluna: borda em si; Segunda coluna: perfil de uma linha qualquer da imagem; Terceira coluna: Derivada do perfil da borda.

A derivada é obtida através da magnitude do gradiente em cada *pixel*. Dada uma imagem definida por uma função $f(x, y)$, o gradiente de f na coordenada (x, y) é dado por um vetor de duas dimensões:

$$\nabla \boldsymbol{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}, \tag{2.2}$$

e a magnitude do vetor é dada por:

$$\nabla \boldsymbol{f} = \sqrt{{G_x}^2 + {G_y}^2}. \tag{2.3}$$

Em termos computacionais, os gradientes são obtidos filtrando a imagem com duas matrizes 3×3, apresentadas na Figura 2.6.

A filtragem é feita no domínio espacial, posicionando o centro da máscara sobre

cada *pixel* da imagem original, sendo que a resposta do filtro é dada por:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t), \tag{2.4}$$

onde $a = b = 1$ (no caso do filtro de Sobel), w representa a máscara, f é a imagem original e g é a imagem filtrada. Um exemplo do funcionamento da filtragem pode ser visto na Figura 2.7.

-1	0	1
-2	0	2
-1	0	1

(a) Componente horizontal

-1	-2	-1
0	0	0
1	2	1

(b) Componente vertical

Figura 2.6: Matrizes 3×3 utilizadas para a aplicação da técnica de Sobel.

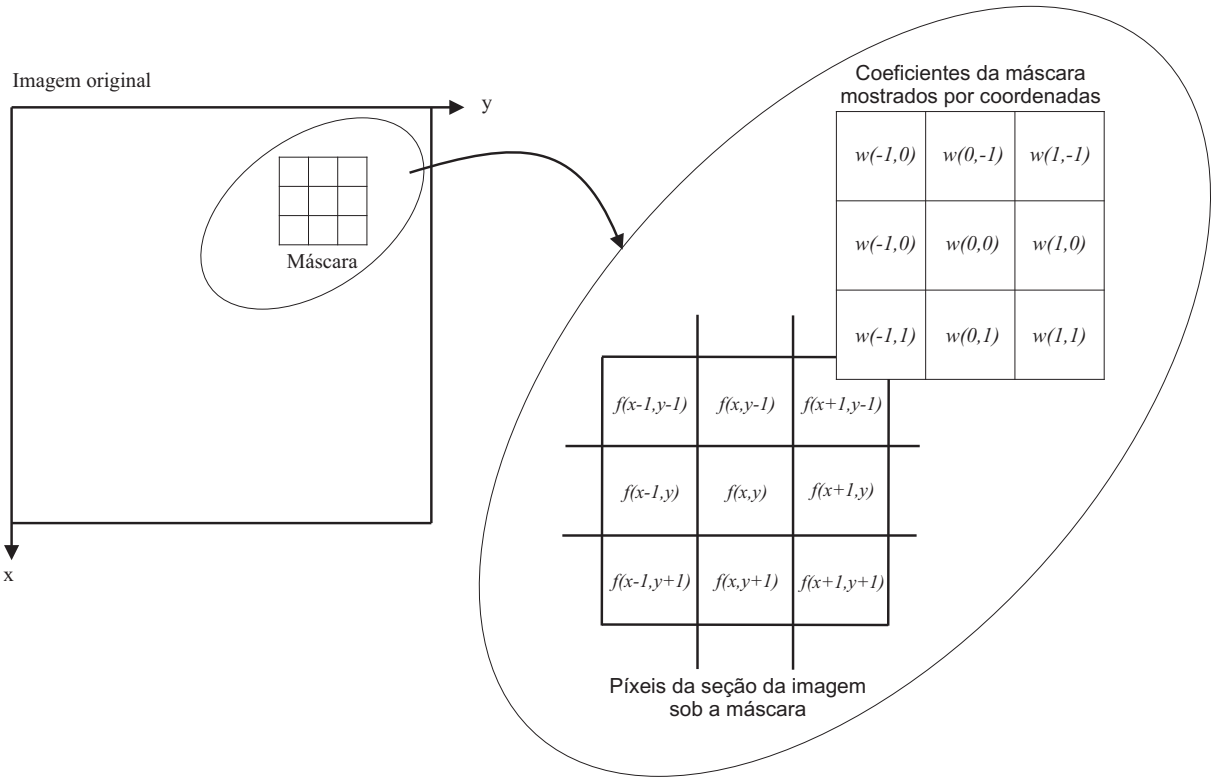
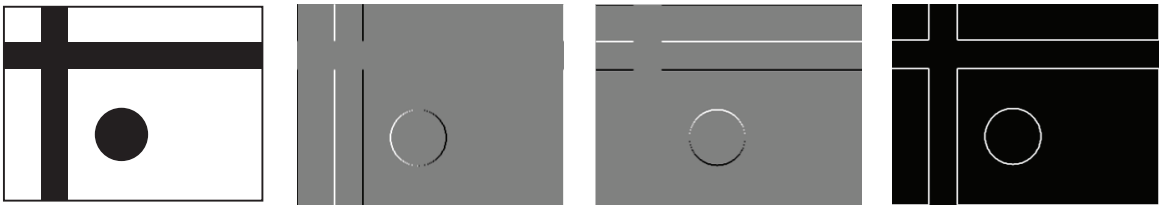


Figura 2.7: Exemplo do funcionamento da filtragem no domínio espacial. Adaptado de Gonzalez e Woods (2002).

As matrizes apresentadas nas Figuras 2.6(a) e 2.6(b) são utilizadas para calcular os

componentes x e y do gradiente, respectivamente. As duas novas imagens geradas são, então, utilizadas para a determinação do módulo dos gradientes, resultando no mapa de bordas, como apresentado na Figura 2.8.



(a) Imagem original. (b) Componente x. (c) Componente y. (d) Módulo (mapa de bordas).

Figura 2.8: Componentes do gradiente calculado utilizando o operador Sobel.

Tomando a matriz apresentada na Figura 2.6(a) como exemplo, pode-se perceber que a primeira coluna da matriz apresenta coeficientes negativos e a terceira, positivos. Dessa forma, regiões que apresentem uma transição brusca retornam valores altos, enquanto que regiões homogêneas resultam em zero, em um caso ideal. Ainda, pode-se notar que os coeficientes da primeira e última linha também são levados em consideração, o que faz com que o filtro de Sobel também funcione como um filtro de média, suavizando a imagem para eliminar possíveis ruídos. O gradiente gerado sempre aponta perpendicularmente à borda detectada.

A filtragem no domínio espacial apresenta um problema quando o filtro é posicionado nas bordas da imagem, pois alguns de seus elementos estarão posicionados fora da imagem. Isso pode ser tratado de diferentes formas, como multiplicando esses índices por 0 ou duplicando a borda da imagem, que é o caso do OpenCV. Píxeis virtuais são criados fora da borda da imagem, de tal forma que $f(-dx, y) = f(0, y)$, $f(w + dx, y) = f(w - 1, y)$, onde d é um escalar, e assim por diante.

(a) Imagem original. (b) Operador Sobel. (c) Operador DiZenzo

Figura 2.9: Comparação entre o resultado dos operadores Sobel e DiZenzo.

Outras técnicas mais robustas, como operador DiZenzo (GONZALEZ e WOODS, 2002),

também são utilizadas para a detecção de bordas. Esse último leva em consideração a informação de cor da imagem para a determinação das bordas. Porém, vários testes foram feitos e comprovou-se que o resultado não apresenta melhoras significativas em relação ao operador Sobel. Um dos resultados pode ser visto na Figura 2.9. Portanto, já que a técnica de Sobel possui um custo computacional menor, optou-se por ela.

2.3 TEORIA DE DECISÃO BAYESIANA

Além da informação de borda, optou-se por utilizar informações probabilísticas provenientes da imagem a fim de segmentar a pessoa. Conforme demonstrado em trabalhos similares, como em Lin et al. (2007), Zhao e Davis (2005) e Criminisi et al. (2006), abordagens probabilísticas tendem a aumentar a precisão do sistema. A idéia é utilizar a teoria de decisão Bayesiana para determinar qual a probabilidade de um *pixel* pertencer ao primeiro ou segundo plano. No decorrer deste trabalho, primeiro plano será utilizado para se referir à pessoa, enquanto que segundo plano é referente ao fundo da imagem.

A teoria de Decisão Bayesiana é uma abordagem estatística para o problema de classificação de padrões. Essa abordagem é baseada na quantificação do compromisso entre várias decisões de classificação e custos associados a elas (DUDA et al., 2000).

Na teoria de Bayes existem dois fatores principais: probabilidade *a priori* e função densidade de probabilidade. A primeira representa o conhecimento sobre um determinado domínio. Por exemplo: se um dado fosse jogado ao ar, inicialmente assume-se que a probabilidade de qualquer um dos números ser sorteado é a mesma, então $P(x) = 1/6$. No entanto, se o dado fosse "viciado", a probabilidade *a priori* favoreceria um determinado número. Nesse caso, tem-se um conhecimento sobre o domínio da aplicação e a probabilidade *a priori* deve ser adaptada. O segundo fator da teoria de Bayes é a função densidade de probabilidade, a qual descreve o comportamento das variáveis do sistema. Uma distribuição muito utilizada é a distribuição normal, já que, assim como descrito em Duda et al. (2000), diversos elementos na natureza seguem essa distribuição, além de que ela é analiticamente mais fácil de ser tratada.

No caso deste trabalho, o teorema de Bayes é utilizado para calcular a probabilidade de um pixel x pertencer a uma determinada classe ω_j através de:

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)}, \tag{2.5}$$

onde $p(x|\omega_j)$ é a função densidade de probabilidade, $P(\omega_j)$ é a probabilidade a priori da classe ω_j, e $p(x)$ é a FDP para toda distribuição de amostras, dada por:

$$p(x) = \sum_{j=1}^{c} p(x|\omega_j)P(\omega_j), \tag{2.6}$$

onde c é a quantidade de classes presente no problema. No caso deste trabalho, são duas: primeiro e segundo plano.

2.4 FILTRO GAUSSIANO

Sabe-se que as informações provenientes de dispositivos de captura de imagem, como uma *webcam* no caso deste trabalho, são afetadas por ruídos devido à iluminação, qualidade do *hardware*, entre outros fatores. Portanto, em diferentes etapas do processamento, filtros de suavização são aplicados para atenuar esses ruídos. Um filtro simples que permite obter resultados satisfatórios é o filtro Gaussiano, o qual também pode ser implementando utilizando uma matriz discreta, como as apresentadas na Figura 2.6. No entanto, no caso do filtro Gaussiano, os pesos da máscara são baseados na distribuição Gaussiana e seu tamanho é variável. No caso 1D, a Gaussiana pode ser calculada através de:

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-x^{2/(2\sigma^2)}}, \tag{2.7}$$

onde σ é o desvio padrão. Estendendo a Equação 2.7 para o caso 2D simetricamente, tem-se que:

$$g(x, y) = \frac{1}{2\pi\sigma^2} \exp^{-(x^2+y^2)/(2\sigma^2)}. \tag{2.8}$$

Alguns exemplos de suavização utilizando o filtro Gaussiano podem ser vistos na Figura 2.10, onde foi definido que $M = 3\sigma$, onde M é o tamanho da máscara simétrica. O filtro é aplicado independentemente nos 3 canais de cores RGB.

(a) Imagem original (b) $\sigma = 1$ (c) $\sigma = 3$ (d) $\sigma = 5$

Figura 2.10: Exemplo da aplicação de um filtro Gaussiano 2D no domínio espacial.

Obviamente, quanto maior o tamanho da janela e o desvio padrão, maior será o nível de suavização da imagem. No caso da biblioteca OpenCV, o desvio padrão é calculado automaticamente baseado no tamanho da janela, de tal forma que:

$$\sigma = \left(\frac{M}{2} - 1\right) \times 0,3 + 0,8. \tag{2.9}$$

2.5 CURVAS DE BÉZIER

Dado que as energias (informações) foram obtidas, pode-se determinar a silhueta da pessoa na imagem. No entanto, uma busca exaustiva por toda imagem possui um custo computacional alto. Dado que a posição da face é conhecida, pode-se utilizar isso como um ponto de partida e definir um espaço de busca em torno dela, onde sabe-se que a silhueta deve estar contida. Nesse trabalho, essa região de busca é descrita por curvas de Bézier.

De acordo com Farin (1988), a curva de Bézier de ordem n pode ser expressa da seguinte forma:

$$\boldsymbol{b}(t) = \sum_{n}^{i=0} \boldsymbol{b}_i B_i^n(t), \tag{2.10}$$

onde $\boldsymbol{b}_i$ é o ponto de controle i, n é a ordem da curva, $0 \leq t \leq 1$ descreve a posição do ponto sobre a curva, e $B_i^n(t)$ é um polinômio de Bernstein, dado explicitamente por:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \tag{2.11}$$

sendo que os coeficientes do binômio são dados por:

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!}, & se\ 0 \leq i \leq n; \\ 0, & caso\ contrário. \end{cases} \tag{2.12}$$

O cálculo da curva de Bézier também pode ser feito de forma iterativa através do algoritmo de Casteljau (FARIN, 1988). No caso deste trabalho, cada curva de Bézier é criada utilizando quatro pontos de controle $\{\boldsymbol{b_0}, \boldsymbol{b_1}, \boldsymbol{b_2}, \boldsymbol{b_3}\}$. Um exemplo da criação da curva pode ser visto na Figura 2.11.

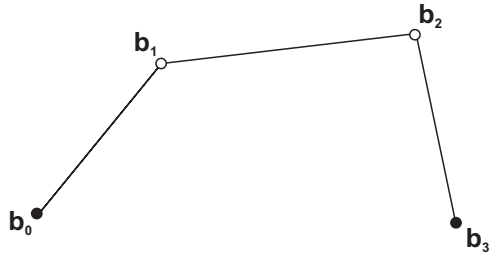


Figura 2.11: Pontos de controle da curva de Bézier.

Uma das propriedades da curva é que ela intercepta o primeiro e último ponto de controle e o intervalo entre eles é definido por $t = [0, 1]$. Portanto, dado um valor $0 \leq t \leq 1$, inicialmente encontra-se o valor para o ponto $\boldsymbol{b_0^1}$ no primeiro segmento de reta, como demonstrado na Figura 2.12. Nesse caso:

$$\boldsymbol{b_0^1} = \boldsymbol{b_0} + t(\boldsymbol{b_1} - \boldsymbol{b_0}), \tag{2.13}$$

que também pode ser escrita da seguinte forma:

$$\boldsymbol{b}_{\mathbf{0}}^{\mathbf{1}} = (1-t)\boldsymbol{b}_{\mathbf{0}} + t\boldsymbol{b}_{\mathbf{1}}. \tag{2.14}$$

Da forma análoga, isso é feito para o segundo e terceiro segmentos de reta, obtendo-se os pontos:

$$\boldsymbol{b}_{\mathbf{1}}^{\mathbf{1}} = (1-t)\boldsymbol{b}_{\mathbf{1}} + t\boldsymbol{b}_{\mathbf{2}}; \tag{2.15}$$

$$\boldsymbol{b}_{\mathbf{2}}^{\mathbf{1}} = (1-t)\boldsymbol{b}_{\mathbf{2}} + t\boldsymbol{b}_{\mathbf{3}}, \tag{2.16}$$

como apresentado na Figura 2.13.

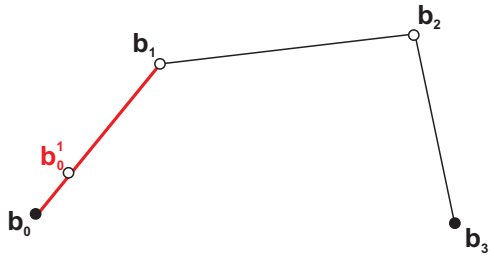


Figura 2.12: Ponto correspondente a t no primeiro segmento de reta.

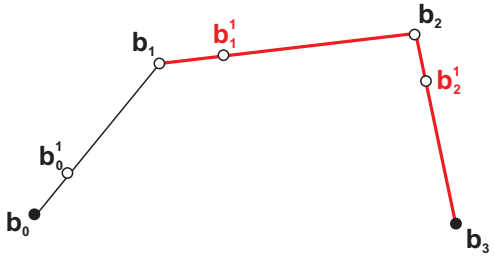
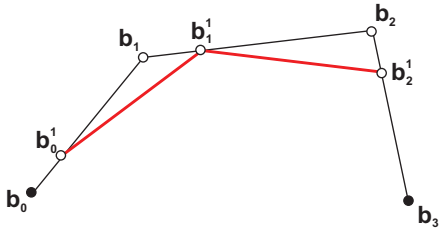
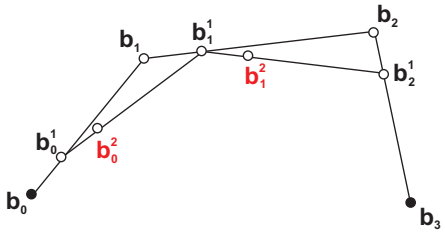


Figura 2.13: Pontos correspondente a t no segundo e terceiro segmentos de reta.

Portanto, tem-se os três primeiros pontos respectivos ao parâmetro t em cada um dos segmentos de reta. A partir de então, novas retas são traçadas entre os pontos obtidos, como mostra a Figura 2.14.



(a) Segmentos de reta traçados entre os pontos obtidos na iteração anterior.



(b) Pontos referentes ao parâmetro t nas novas retas.

Figura 2.14: Segunda iteração do algoritmo de Casteljau.

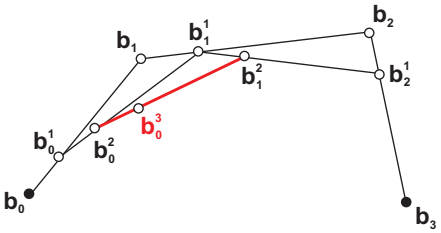
Os novos pontos, analogamente aos anteriores, também são obtidos em função de t, de modo que:

$$\boldsymbol{b_0^2} = (1-t)\boldsymbol{b_0^1} + t\boldsymbol{b_1^1}; \tag{2.17}$$

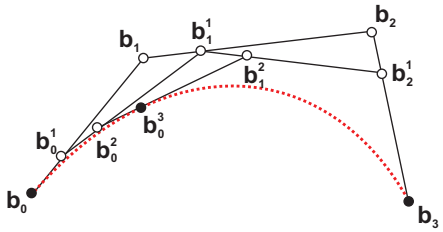
$$\boldsymbol{b_1^2} = (1-t)\boldsymbol{b_1^1} + t\boldsymbol{b_2^1}. \tag{2.18}$$

Finalmente, tem-se a última reta, neste caso entre os pontos $\boldsymbol{b_0^2}$ e $\boldsymbol{b_1^2}$, obtendo-se o último ponto $\boldsymbol{b_0^3}$, como mostra a Figura 2.15, dado por:

$$\boldsymbol{b_0^3} = (1-t)\boldsymbol{b_0^2} + t\boldsymbol{b_1^2}. \tag{2.19}$$



(a) Última reta obtida.



(b) Ponto sobre a curva de Bézier referente a t.

Figura 2.15: Última iteração do algoritmo de Casteljau.

Efetuando o procedimento anterior iterativamente para o intervalo $t = [0, 1]$, pode-se obter todos os pontos sobre a curva de Bézier. Como utiliza-se apenas quatro pontos de controle, pode-se obter uma fórmula fechada para o cálculo de $\boldsymbol{b_0^3}$, de forma que:

$$\begin{aligned}
\boldsymbol{b_0^3} &= (1-t)\boldsymbol{b_0^2} + t\boldsymbol{b_1^2} \\
&= (1-t)[(1-t)\boldsymbol{b_0^1} + t\boldsymbol{b_1^1}] + t\boldsymbol{b_1^2} \\
&= (1-t)^2\boldsymbol{b_0^1} + t(1-t)\boldsymbol{b_1^1} + t\boldsymbol{b_1^2} \\
&= (1-t)^2\boldsymbol{b_0^1} + t(1-t)\boldsymbol{b_1^1} + t[(1-t)\boldsymbol{b_1^1} + t\boldsymbol{b_2^1}] \\
&= (1-t)^2\boldsymbol{b_0^1} + 2t(1-t)\boldsymbol{b_1^1} + t^2\boldsymbol{b_2^1} \\
&= (1-t)^2\boldsymbol{b_0^1} + 2t(1-t)\boldsymbol{b_1^1} + t^2[(1-t)\boldsymbol{b_2} + t\boldsymbol{b_3}] \\
&= (1-t)^2\boldsymbol{b_0^1} + 2t(1-t)\boldsymbol{b_1^1} + t^2(1-t)\boldsymbol{b_2} + t^3\boldsymbol{b_3} \\
&= (1-t)^2\boldsymbol{b_0^1} + 2t(1-t)[(1-t)\boldsymbol{b_1} + t\boldsymbol{b_2}] + t^2(1-t)\boldsymbol{b_2} + t^3\boldsymbol{b_3} \\
&= (1-t)^2\boldsymbol{b_0^1} + 2t(1-t)^2\boldsymbol{b_1} + 3t^2(1-t)\boldsymbol{b_2} + t^3\boldsymbol{b_3} \\
&= (1-t)^2[(1-t)\boldsymbol{b_0} + t\boldsymbol{b_1}] + 2t(1-t)^2\boldsymbol{b_1} + 3t^2(1-t)\boldsymbol{b_2} + t^3\boldsymbol{b_3} \\
&= (1-t)^3\boldsymbol{b_0} + t(1-t)^2\boldsymbol{b_1} + 2t(1-t)^2\boldsymbol{b_1} + 3t^2(1-t)\boldsymbol{b_2} + t^3\boldsymbol{b_3} \\
&= (1-t)^3\boldsymbol{b_0} + 3t(1-t)^2\boldsymbol{b_1} + 3t^2(1-t)\boldsymbol{b_2} + t^3\boldsymbol{b_3}.
\end{aligned} \tag{2.20}$$

2.6 ALGORITMO DE DIJKSTRA

Conforme apresentando anteriormente, graças à detecção de face, tem-se uma estimativa inicial da posição da pessoa. Utilizando essa estimativa como ponto de partida, uma curva em formato de Ω é posicionada sobre a pessoa. Logo após, retas perpendiculares são dispostas ao longo dessa curva e diversos nodos são dispostos sobre essas retas. Essa configuração pode ser interpretada como um grafo (ver Figura 4.7).

Dado que as informações da imagem são conhecidas, é feita então, a segmentação utilizando o algoritmo de Dijkstra (DIJKSTRA, 1959), o qual determinará o caminho ótimo baseado nessas informações. Esse caminho representará uma aproximação da silhueta real da pessoa. O processo detalhado de como esse algoritmo é aplicado no presente trabalho é demonstrado na Seção 4.4.

De acordo com Cormen et al. (2002), o algoritmo de Dijkstra resolve o problema do caminho mais curto partindo de uma única origem em um grafo orientado ponderado $G = (N, E)$, onde N é o conjunto de nodos (vértices) e E é o conjunto de arestas.

Conforme descrito em Goldbarg e Luna (2000), considere F o conjunto dos nós fechados (vértices para o qual já se conhece o caminho mínimo); A é o conjunto dos nós abertos (vértices para os quais ainda não se conhece o caminho mínimo); V é o conjunto dos nós rotulados e abertos em G (conhece-se o caminho mínimo, mas talvez não seja o menor), Γ^+ é o conjunto de vizinhos do vértice r; r é o vértice a ser fechado na iteração t; t é um contador de iterações; $rot(i)$ é um vetor que armazena os vértices que deram origem à distância calculada para o nodo de índice i; d_{ij} é a distância entre o vértice x_i e x_j; e d_{ij}^t é a distância entre dois vértices na iteração t.

As variáveis são iniciadas de forma que: $d_{11} \leftarrow 0$; $\{d_{1i} \leftarrow \infty \forall i \in N \backslash \{x_1\}\}$; $V \leftarrow \{x_1\}$; $A \leftarrow \{N\}$; $F \leftarrow \oslash$; $\{rot(i) \leftarrow 0 \forall i \in N\}$ e o algoritmo pode ser descrito de acordo com o seguinte pseudocódigo:

for $t = 1$ to n **do**
  $r \leftarrow x_i$ tal que $d_{1i} \leftarrow \min\{d_{1i}\}$, $\forall x_i \in A$
  $F \leftarrow F \cup \{r\}$
  $A \leftarrow A \backslash \{r\}$
  $V \leftarrow A \cap \Gamma^+(r)$
  for $i \in V$ **do**
    $p \leftarrow \min\left\{d_{1i}^{t-1}, (d_{1r} + d_{ri})\right\}$
    if $p < d_{1i}^{t-1}$ **then**
      $d_{1i}^t \leftarrow p$
      $rot(i) \leftarrow r$
    end if
  end for
end for

Um exemplo gráfico do funcionamento do algoritmo pode ser visto na Figura 2.16. Inicialmente, um nodo é escolhido (no caso da Figura 2.16(a), o nodo 1) e determina-se com quais nodos ele está conectado, neste caso: 2, 3 e 4. O vetor de duas posições associado a cada vértice armazena o vértice origem e a distância entre eles, respectivamente (Figura 2.16(b)).

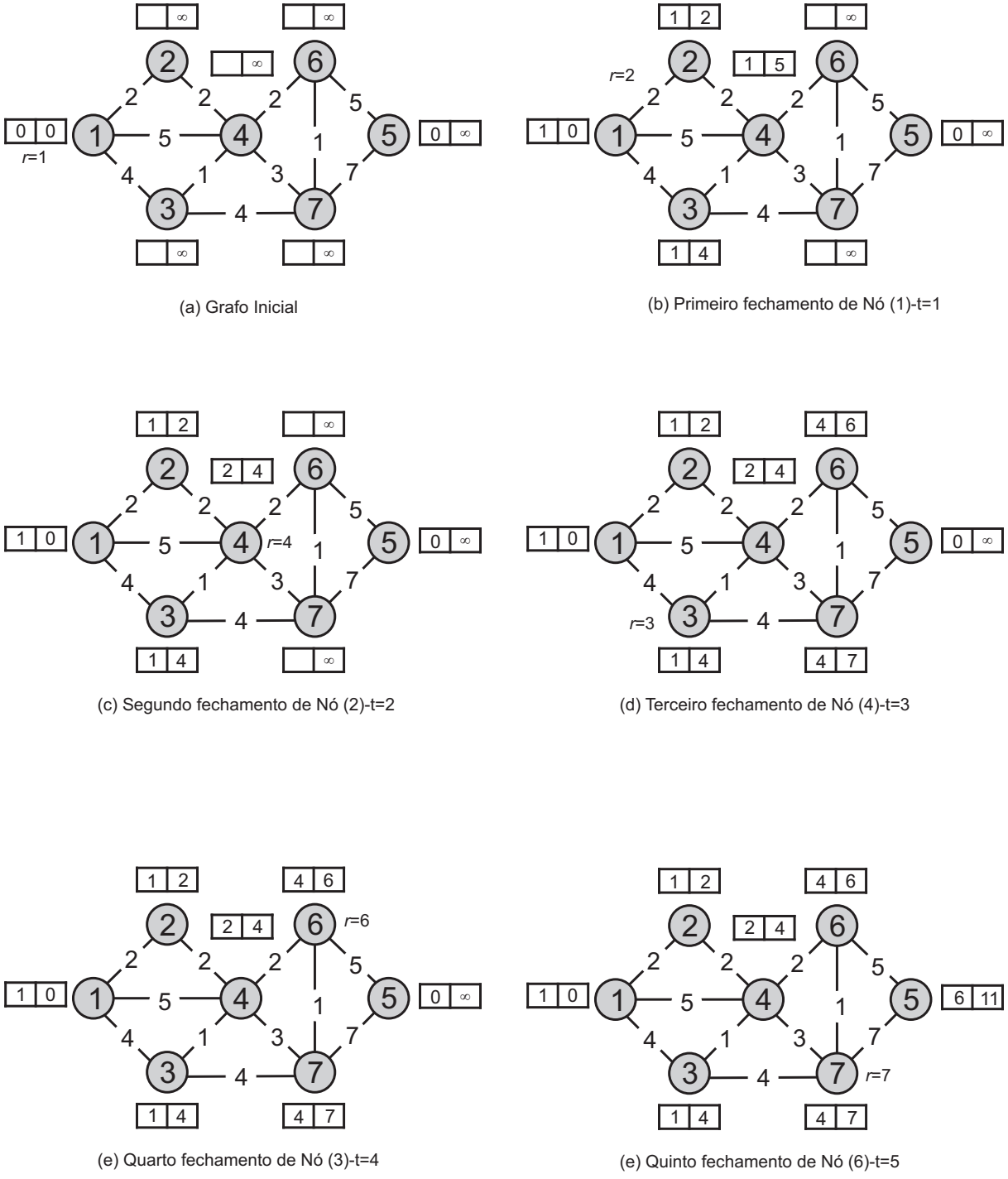


Figura 2.16: Exemplo da evolução do algoritmo de Dijsktra. Adaptado de (GOLDBARG e LUNA, 2000).

O nodo que acabou de ser avaliado (nodo 1) é descartado. A seguir, seleciona-se um

dos nodos com o qual o último vértice em questão estava conectado (neste caso, o nodo 2). Avalia-se os nodos que ele possui ligação (somente nodo 4). Novamente, calcula-se qual a distância até ele. A distância até o nodo 4 é calculada através da distância até o nodo 2 mais a distância entre nodo 2 e 4. Pode-se ver que ela é menor do que a distância previamente calculada (entre nodo 1 e 4). Portanto, o vetor de controle associado ao nodo 4 é atualizado (Figura 2.16(c)).

Isso é feito sucessivamente até que todos os nós tenham sido avaliados. Assim que essa etapa de verificação é concluída, escolhe-se o nodo final (neste caso, nodo 5) e calcula-se o caminho inverso até o nodo de origem.

Pode-se perceber que o algoritmo utiliza uma estratégia "gulosa", já que quando um vértice x_i já foi analisado, ele é desconsiderado e somente os outros vértices são levados em consideração. Apesar de que, geralmente, os algoritmos gulosos não apresentam um resultado ótimo, conforme provado em Cormen et al. (2002), o algoritmo de Dijkstra sempre retorna o menor caminho possível. A complexidade do algoritmo é $O(n^2)$ e é capaz de calcular a menor distância de um vértice inicial a todos os outros.

3 REVISÃO BIBLIOGRÁFICA

O cérebro humano é capaz de detectar e segmentar objetos, extrair suas características e reconhecê-los facilmente em qualquer ambiente. No entanto, desenvolver uma técnica computacional que seja capaz de fazer o mesmo é extremamente complexo. Isso se deve ao fato de que o desempenho humano para efetuar tais atividades se dá principalmente a um nível subconsciente, de acordo com Plataniotis e Venetsanopoulos (2000). Ainda, de acordo com o autor, a segmentação de objetos é uma etapa de extrema importância na área de visão computacional, já que processamentos futuros, como extração de informação a partir de imagens, é altamente dependente de uma segmentação precisa.

Atualmente, não existe uma maneira genérica de efetuar detecção/segmentação de objetos de forma eficaz. Diversas soluções foram e ainda vêm sendo propostas buscando resolver problemas relacionados ao tema, sendo que muitas delas são altamente focadas para uma determinada aplicação, levando em consideração informação já conhecida sobre o ambiente onde será detectada a pessoa/objeto, enquanto que outras procuram criar soluções que funcionem em diversos ambientes. Além disso, algumas delas somente são capazes de obter a localização do objeto, sem maiores informações sobre ele (apenas detecção), enquanto que outras obtém detalhes sobre a silhueta do mesmo (segmentação).

A grande parte das técnicas propostas podem ser divididas em categorias, como: Casamento de máscaras baseadas em contornos (*Contour-based template Matching*), como por exemplo nos trabalhos de Gavrila e Philomin (1999), Tuzel et al. (2007) e Zhao e Davis (2005); baseadas em "manchas" (*blob-based*), como apresentado por Piccardi (2004); e baseados em modelos (*Model-Based*), como a solução de Segen (1996). O primeiro tipo de abordagem lida bem com variação de cor do objeto em si e também com variação de iluminação. No entanto, não são eficientes em ambientes com diversos objetos, resultando em um grande número de falsos positivos. A segunda abordagem geralmente requer uma subtração do fundo da imagem para determinar o objeto em questão. Obviamente, é necessário um treinamento prévio à segmentação em si, o que nem sempre é viável dependendo da aplicação. Além disso, esse tipo de técnica é muito suscetível a variações no ambiente, como movimentos da câmera (por exemplo: uma *webcam* colocada em uma tela de um *laptop*) ou mudança de iluminação. O terceiro tipo utiliza modelos estocásticos, como Modelos Aleatórios de Markov (*Markov Random Fields*), o que permite que segmentações mais precisas possam ser feitas, até mesmo em ambientes complexos. Porém, novamente, esse modelo exige treinamento e sua implementação é complexa.

Além dos três tipos de abordagens citadas anteriormente, recentemente, soluções utilizando algoritmos de programação dinâmica vêm sendo propostas, as quais são eficientes do ponto de vista computacional. A seguir, são apresentados com detalhes alguns trabalhos relacionados ao estado-da-arte, demonstrando as respectivas metodologias, resultados

e limitações. Talvez pelo fato do tema abordado neste trabalho ser muito dinâmico, a maioria dos trabalhos estejam publicados em congressos.

A solução proposta por Gavrila e Philomin (1999) apresenta um modelo de segmentação de sinais de trânsito e pedestres em tempo real, utilizando uma câmera em movimento. No modelo, são utilizadas máscaras, previamente criadas, que representam os objetos que são esperados durante o vídeo. A máscara varre a imagem inteira procurando o melhor local para ser posicionada. O critério utilizado para determinar se a máscara representa ou não o objeto foi o valor da distância *chamfer*, computada sobre a transformada distância da imagem original. Como já era esperado, um conjunto muito grande de máscaras resulta em perda de desempenho. Portanto, alguns procedimentos foram feitos para aprimorar a performance do sistema, como efetuar uma subamostragem dos pontos da máscara e alguns aprimoramentos específicos do *hardware* utilizado. Foram efetuados testes numa base de dados de 1000 imagens de sinais de trânsito, utilizando 36 máscaras, e foi observado que o sistema apresentou uma taxa de acerto acima de 95%, sendo capaz de executar a uma taxa de 10-15 Hz. Outro teste mais complexo foi feito utilizando uma base de dados de pedestres, com aproximadamente 700 imagens, e utilizando, desta vez, 5500 máscaras. Os testes demonstraram que a taxa de acerto caiu para 75-85% quando fixado o número de falsos positivos para ser menor que dois por imagem. Desta vez, o sistema executou a uma taxa de 1-5 Hz. Alguns resultados visuais podem ser observados na Figura 3.1.

Figura 3.1: Exemplo de resultados da solução proposta por Gavrila e Philomin (1999).

O sistema proposto por Lin et al. (2007) é capaz de segmentar pessoas em imagens

e vídeos. A segmentação é feita utilizando um *framework* Bayesiano, onde são utilizadas informações sobre o contorno da pessoa e cor (quando utilizada a subtração de fundo). O modelo foi baseado no artigo apresentado anteriormente, porém, desta vez as diferentes máscaras criadas representam silhuetas locais de um objeto, sendo que elas foram dispostas em uma hierarquia, portanto, a junção delas representa o objeto completo. Desta forma, é possível segmentar objetos altamente articulados de forma mais fácil. Após o posicionamento das máscaras, é gerado um conjunto de hipóteses iniciais representando as pessoas na imagem e suas respectivas silhuetas. Além disso, essas hipóteses passam por um processo de otimização utilizando um *framework* Bayesiano do tipo MAP (*Maximum a Posteriori Probability*). O sistema atingiu resultados satisfatórios mesmo em imagens com grande oclusão, conforme pode ser visto nos resultados apresentados na Figura 3.2. O algoritmo foi implementado em C++ e executa a 2 quadros por segundo (FPS) quando não é utilizada a subtração de fundo e, caso contrário, a 5 FPS, para vídeos com resolução de 384×288.

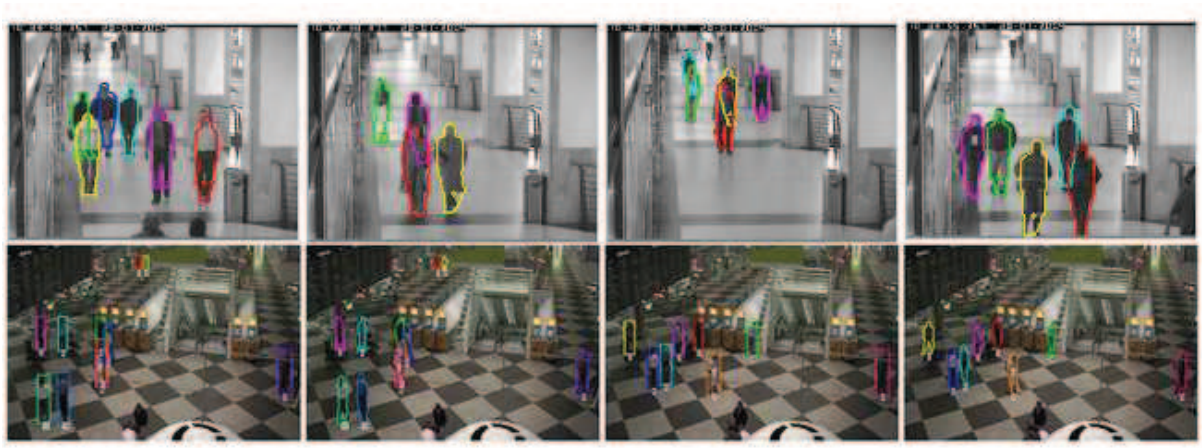


Figura 3.2: Exemplo de resultados do sistema proposto por Lin et al. (2007).

Em Tuzel et al. (2007) é apresentado um modelo para detecção de pessoas em imagens estáticas. Para a detecção das pessoas na imagem, inicialmente é definida uma janela de tamanho R, e janelas menores $r_i; i = 1, 2, ...$ são criadas dentro dessa em diferentes posições e tamanhos. Para cada janela r_i, calcula-se uma matriz de covariância, as quais funcionam como descritores de objetos. As janelas r_i são redimensionadas ao passo de $1/10$ ao longo do eixo horizontal e vertical até que $r_i = R$. Para a classificação das matrizes, o artigo introduz uma nova abordagem, que é a classificação de pontos (representados por cada matriz de covariância) numa Variedade de Riemann. Essa etapa é feita utilizando uma seqüência de classificadores dispostos na topologia cascata-de-rejeição, a qual foi criada com o algoritmo *LogitBoost*. A vantagem dessa abordagem é que se um classificador detecta que uma determinada região não representa uma pessoa, ela já é descartada e não precisa passar para os estágios seguintes da cascata. No entanto, quanto mais es-

tágios forem utilizados, maior é a certeza de que aquela determinada região representa uma pessoa, sendo que a confirmação só é obtida após a aprovação do último estágio. O sistema foi testado na base de dados INRIA, contendo 1774 imagens com pessoas e 1671 sem pessoas. Considerando uma taxa de falsos positivos por janela de 10^{-4}, o sistema obteve uma taxa de erros de 6,8%. A abordagem proposta, no entanto, não é capaz de detectar a silhueta da pessoa de forma fina, somente a região onde ela se encontra, como pode ser visto na Figura 3.3.

Figura 3.3: Resultados da detecção de pessoas do sistema desenvolvido por Tuzel et al. (2007).

Figura 3.4: Exemplo de segmentação utilizando *snakes*.

No modelo proposto por Ballerini (2003), a autora parte da premissa que somente uma pessoa estará na cena e que ela estará se movendo. Uma pessoa é modelada utilizando três *snakes* (cabeça, torso e pernas) conectadas entre si. A posição inicial da *snake* é obtida através da diferença entre quadros consecutivos (informação de movimento) e sua energia é baseada em Algoritmos Genéticos (AGs). Os testes empíricos foram efetuados utilizando imagens obtidas através de uma *webcam* com baixa resolução (160×120). O sistema proposto foi capaz de realizar segmentações em tempo real, no entanto, os resultados não foram muito satisfatórios, tendo em vista que para manter o sistema em tempo real é necessário restringir alguns aspectos dos AGs, como por exemplo o número de indivíduos iniciais e o número de gerações. A autora acredita que se os valores desses parâmetros forem aumentados, os resultados serão aprimorados, porém, o custo computacional do

sistema também aumentará. Alguns resultados do sistema apresentado podem ser vistos na Figura 3.4.

O trabalho proposto por Dalal e Triggs (2005) também apresenta um sistema de detecção de pessoas em imagens estáticas. Nesse trabalho, a idéia principal é representar um objeto como uma distribuição local de gradientes (HOG). Um HOG é obtido dividindo a imagem em pequenas regiões (células) e acumulando um histograma 1-D de direções de gradientes de todos os píxeis dessa célula. Para garantir que os atributos sejam invariantes a mudanças na iluminação, os histogramas são normalizados, criando então um descritor HOG. Os descritores são utilizados em um classificador do tipo SVM (*Support Vector Machine*). O modelo proposto foi testado com dois conjuntos de dados: MIT e INRIA. De acordo com os autores, o sistema foi capaz de obter uma taxa de detecção quase perfeita na base MIT e, na outra, diminuiu o número de falsos positivos por janela em uma ordem de grandeza, quando comparado a outros sistemas. Porém, mais uma vez, o modelo não apresenta informação de silhueta, apenas obtém a região da imagem onde a pessoa se encontra.

No modelo apresentado por Jia e Zhang (2007), também são utilizados HOGs, porém, desta vez, eles são integrados com um classificador do tipo cascata-de-rejeição, composto por CARTs (*Classification and Regression Trees*), o que faz com que o desempenho do sistema seja muito maior que o modelo proposto anteriormente, tornando o sistema capaz de executar em tempo real. Para a criação dos classificadores, foi utilizado o algoritmo *Adaboost*. De acordo com os autores, o sistema obteve uma alta taxa de detecção com uma taxa de falsos positivos baixa. A Figura 3.5 apresenta alguns exemplos de detecções na base de dados INRIA. Novamente, tem-se somente a informação da localização da pessoa. Nenhuma informação sobre a silhueta é obtida.

Figura 3.5: Alguns exemplos de detecção na base de dados INRIA. Modelo proposto por (ZHU et al., 2006).

O artigo de Zhao e Davis (2005) apresenta um sistema de segmentação de pessoas focado para pequenas reuniões, portanto somente a parte superior da pessoa é considerada. O modelo proposto utiliza informação de cor e contorno das imagens. Inicialmente, é feito um casamento de máscaras de forma hierárquica para detectar a pessoa. Após essa etapa, é feita uma segmentação do tipo "Figure-Ground" para aprimorar o resultado do

casamento de máscaras. Essa etapa consiste basicamente em determinar a probabilidade de um *pixel* pertencer ao primeiro plano ou ao fundo. O resultado final da segmentação é determinado através da soma ponderada das estimativas calculadas anteriormente utilizando informação de cor e formato. Os pesos da soma e o tamanho da janela utilizada para obter informação de cor são ajustados dinamicamente durante a segmentação. O sistema proposto atingiu uma taxa de detecção de 92% e uma taxa de falsos positivos de 7%. Alguns resultados do modelo proposto podem ser vistos na Figura 3.6.

Figura 3.6: Exemplos de segmentação utilizando o sistema proposto por Zhao e Davis (2005)

No trabalho de Kolmogorov et al. (2005), é apresentado um sistema de separação das camadas de primeiro e segundo plano da imagem. O foco da aplicação é a substituição do segundo plano em tempo real para videoconferências. Os autores exploram duas abordagens: *Layered Dynamic Programming* (LDP) e *Layared Graph Cut* (LGC). Essas duas abordagens são utilizadas para fazer a junção das informações probabilísticas extraídas da imagem, as quais se baseiam em informação de cor, estéreo e contraste. Na solução proposta, a probabilidade de cor para o primeiro e segundo plano foram modeladas utilizando misturas Gaussianas do espaço de cor RGB criadas a partir de *frames* manualmente segmentados. Além disso, o modelo do segundo plano é melhorado juntando probabilidades individuais de cada *pixel*. O resultado do modelo proposto foi feito através da amostragem do resultado a cada 5 *frames* e comparando-os com um *ground-truth*. O erro é determinado pela quantidade de píxeis classificados erroneamente. Os autores concluem que os benefícios da utilização da informação estéreo juntamente com cor/contraste pode ser notada facilmente. Em alguns testes, algumas pessoas se moviam atrás do locutor e, utilizando informação estéreo, o algoritmo tornou-se muito mais robusto, concentrando-se apenas no locutor. Além disso, os autores concluem que ambas abordagens são válidas para aplicações em tempo real, já que os algoritmos de programação dinâmica são capazes de serem executados de forma rápida. A Figura 3.7 apresenta alguns resultados obtidos no trabalho.

Figura 3.7: Resultados do trabalho proposto por Kolmogorov et al. (2005).

A aplicação do trabalho de Criminisi et al. (2006) estende o trabalho anterior adicionando uma cadeia de Markov de segunda ordem. O modelo de probabilidade a posteriori é definido por um Campo de Condições Aleatórias (*Conditional Random Field*) que utiliza quatro tipo de energias: a cadeia de Markov de segunda ordem (que garante coerência temporal da área segmentada), um termo de coerência espacial, probabilidade de cor das distribuições de primeiro e segundo plano, e probabilidade de movimento. As probabilidades de cor foram modeladas através dos histogramas do espaço de cor YUV, enquanto que a probabilidade de movimento é calculada utilizando as derivadas espaciais e temporais. Os resultados foram avaliados com relação a um *ground-truth*. Os autores concluem que os resultados não são melhores que a segmentação que utiliza informação estéreo, no entanto, são equiparáveis. Também concluem que a mistura das informações de cor e movimento reduzem a taxa de erro. Alguns resultados desse modelo podem ser vistos na Figura 3.8. No artigo, o autor cita que algumas variáveis são obtidas através do uso de *ground-truths*, o que implica em um treinamento do sistema. Porém, em nenhum momento é destacado de que forma é feito esse treinamento.

Figura 3.8: Exemplo de resultados. Adaptado de (CRIMINISI et al., 2006)

Em Yin et al. (2007), novamente é proposto um modelo de segmentação utilizando imagem monocular, que sejam comparáveis a um sistema estéreo. Ele se baseia em informação de movimento e seu contexto espacial, introduzindo um novo tipo de característica, chamada de *moton*. O sistema foi treinado utilizando dados que continham informação de profundidade, como em soluções que utilizam informação estéreo, forçando-o a combinar outras informações disponíveis para induzir a informação estéreo que não está presente. A segmentação final da imagem da pessoa é feita através da técnica *binary min-cut*, a qual busca minimizar a energia em um grafo, dada por um *Conditional Random Field*

(CRF). Testes experimentais foram feitos utilizando vinte e oito seqüências de vídeos monoculares. Os píxeis de cada quinto ou décimo quadro dos vídeos foram manualmente classificados em primeiro ou segundo plano, ou indeciso. Nos experimentos, quarenta e seis quadros de sete vídeos foram utilizados para treino, dois para validação e os quatrocentos e vinte e seis quadros restante para teste. O sistema obteve uma taxa média de erro abaixo de 1%, quando comparada ao *ground-truth*, executando a 1,2 FPS. Alguns exemplos de resultados do algoritmo podem ser vistos na Figura 3.9.

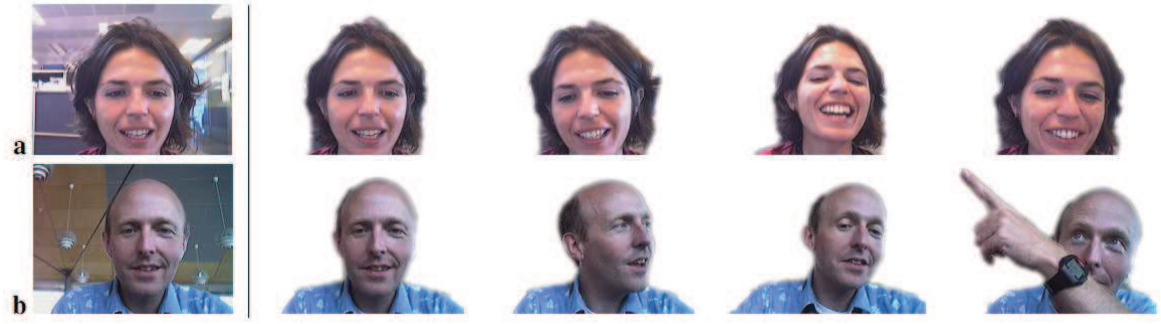


Figura 3.9: Alguns exemplos dos resultados obtidos no modelo de Yin et al. (2007). (a) e (b) apresentam um quadro original de duas seqüências de vídeo para teste e alguns dos quadros segmentados.

A solução proposta por Ross e Kaelbling (2005) utiliza movimento dos objetos para encontrá-los. Dado que um objeto se moveu, é possível obter informações de seu interior e bordas, utilizados então para treinar o modelo de segmentação. O algoritmo é treinado utilizando vídeos previamente segmentados usando subtração de *background*. O algoritmo divide a imagem em *patches* de tamanho 5×5 píxeis, de forma que não se sobreponham e características são extraídas dos pontos sob eles, como cor, brilho e gradiente. Elas são então utilizadas em um *Markov Random Field*. Um pós-processamento é efetuado para garantir a continuidade das bordas e sua consistência. O sistema foi testado em dois conjuntos de dados: um vídeo de carros passando por uma auto-estrada e uma pessoa caminhando em frente a um quadro branco. As medições foram feitas através de métricas como *precision* e *recall*. No primeiro conjunto de dados, o sistema atingiu uma *precision* de 79% e *recall* de 81%, enquanto que no segundo caso, o algoritmo atingiu uma *precision* de 68% e *recall* de 81%. Esse trabalho é interessante no aspecto de que demonstra uma técnica de "aprender" as características do objeto, o que poderia ser utilizado para rastreá-lo.

No trabalho de Zhao e Lee (2006), a silhueta humana é modelada através de Modelos de Markov Ocultos (*Hidden Markov Models* ou HMM). Inicialmente, a silhueta da pessoa é convertida para um sinal 1D. As direções das bordas (calculadas através de gradientes) e os comprimentos dos cortes transversais, de uma ponta a outra da silhueta, são utilizados como característica. Esse tipo de característica fornece informações sobre o formato da

silhueta da pessoa, como gorda ou magra, por exemplo. Na verdade, dois modelos de Markov são treinados: um deles é responsável por identificar se a silhueta pertence a uma pessoa, enquanto que o outro faz o oposto. As silhuetas iniciais são detectadas através de movimento na imagem. Essas são submetidas aos modelos de Markov, já que objetos pertencentes ao fundo podem aparecer juntamente com a pessoa, como por exemplo: a pessoa está abrindo uma porta ou se movimentando muito rápido. Os classificadores HMM são treinados utilizando o algoritmo *Expectation Maximization*. Os testes foram efetuados utilizando 273 vídeos, nos quais 80 silhuetas humanas e 312 não humanas foram manualmente segmentadas para efetuar o treinamento dos classificadores. O algoritmo atingiu uma taxa de acerto de 89,7% para *frames* que não possuíam silhuetas humanas, e 63,4% para silhuetas humanas. Os autores consideram que a taxa de acerto não é muito alta, mas é melhor do que sistemas que simplesmente levam em consideração mudanças nos *frames* para detectar silhuetas humanas. Alguns resultados da solução proposta pelo autor são apresentados na Figura 3.10.

Figura 3.10: Exemplo de resultados do modelo proposto por Zhao e Lee (2006).

Em Kim et al. (2005) é apresentado um modelo de subtração de fundo utilizando *codebooks*, os quais são representados por um ou mais *codewords*. Esse último consiste em um vetor RGB e uma tupla contendo informações de intensidades mínimas e máximas de brilho de todos os píxeis relacionados ao *codeword*, a freqüência com que ele ocorre, o maior intervalo durante o treinamento em que ele não foi recorrente, e a primeira e última vez que ele ocorreu. Cada *pixel* possui um *codebook* correspondente. As amostras de cada *pixel* são agrupadas em conjuntos de *codewords* baseando-se em uma métrica de distorção de cor e limites de brilho. A detecção do primeiro plano é feita simplesmente subtraindo a imagem atual do modelo do plano de fundo criado durante o treinamento. Um *pixel* é classificado como pertencente ao fundo da imagem se: (i) a distorção de cor de algum *codeword* for menor que o limiar de detecção; (ii) o nível de brilho estiver no intervalo desse *codeword*. Os autores comparam o modelo proposto com outros sistemas de subtração de fundo, como MOG e Kernel. A primeira técnica busca modelar o plano de fundo utilizando distribuições Gaussianas multimodais, enquanto que a segunda busca encontrar a função densidade de probabilidade para cada *pixel* baseada em amostras. Através da comparação os autores concluem que o sistema proposto apresenta desempenho tão bom quanto os similares, além de que utiliza menos memória e é mais eficiente computacionalmente

pelo fato de não utilizar componente probabilístico. O sistema também foi estendido para que fosse capaz de lidar com objetos adicionados ao fundo após o treinamento e os resultados foram satisfatórios. Além disso, o efetuando-se algumas modificações no algoritmo, ele é capaz de lidar com mudanças globais da iluminação de forma eficaz. O sistema apresentou uma taxa de falsos positivos constante através de todos os níveis de brilho e alguns resultados podem ser vistos na Figura 3.11.

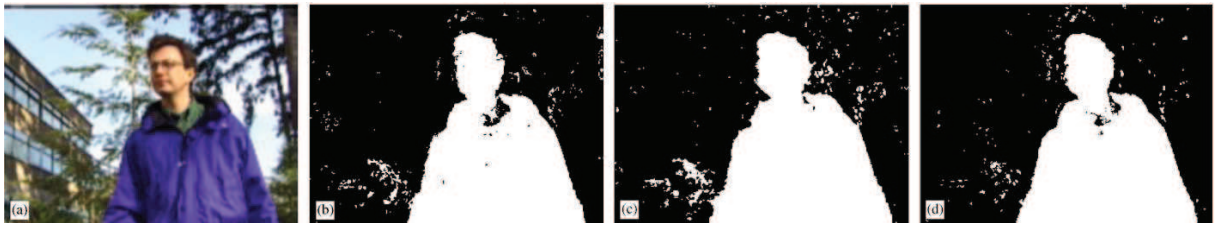


Figura 3.11: Resultados da detecção em fundos dinâmicos: (a) Imagem original, (b) MOG, (c) Kernel, (d) Sistema proposto por Kim et al. (2005).

Conforme pôde ser visto anteriormente, diversas soluções foram propostas para o problema de detecção/segmentação de pessoas para diferentes aplicações. No entanto, muitas não são capazes de encontrar a silhueta com precisão, apenas detectar a pessoa. Além disso, as que detectam as silhuetas requerem que o sistema seja submetido a um treinamento e/ou possuem um desempenho baixo em termos de FPS. O sistema proposto neste trabalho deve ser capaz de ser executado em tempo real e não necessitará de nenhum tipo de treinamento. Os detalhes de cada etapa do desenvolvimento são apresentadas no próximo capítulo.

4 DESENVOLVIMENTO

Neste capítulo são apresentadas as etapas do desenvolvimento do sistema proposto. O sistema foi desenvolvido na linguagem C++ com auxílio da biblioteca OpenCV e o seu funcionamento pode ser visto no fluxograma apresentado na Figura 4.1.

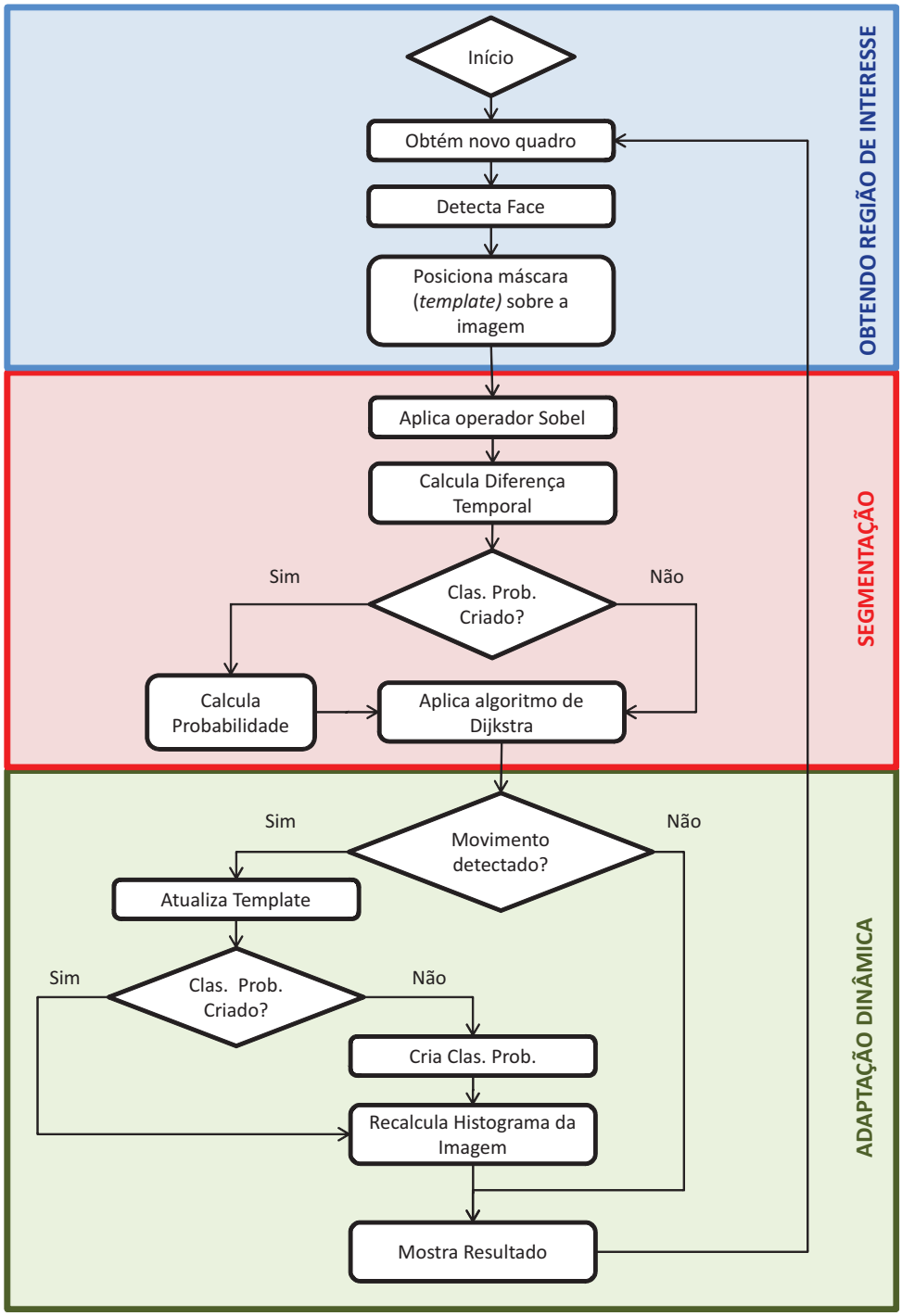


Figura 4.1: Fluxograma do funcionamento básico do sistema.

Conforme pode ser visto no fluxograma, o funcionamento do sistema é dividido em três etapas principais: obtendo região de interesse, segmentação e adaptação dinâmica.

Na primeira etapa, a face do locutor é detectada e, baseando-se no resultado, uma máscara pré-definida em formato de Ω é posicionada sobre a pessoa. Diversas retas são dispostas perpendicularmente ao longo dessa máscara, e um conjunto de nodos é disposto sobre cada reta. Esse conjunto de retas e pontos pode ser interpretado como um grafo orientado.

Na segunda etapa, diferentes energias são extraídas da imagem, como informação de bordas, movimento e, quando disponível, probabilística. Baseado nessas informações, o algoritmo de Dijkstra é utilizado para determinar o caminho ótimo do grafo, o qual se constitui em uma aproximação da silhueta final da pessoa.

A terceira e última etapa do sistema se constitui no ajuste dinâmico, executado quando existe movimento no vídeo. Como, por premissa, o locutor é a única pessoa na cena e o fundo da imagem é estático, assume-se que nesse momento a silhueta encontrada pelo algoritmo é mais confiável. Portanto, a máscara é atualizada de acordo com a silhueta retornada pelo algoritmo de Dijkstra. Além disso, a função densidade de probabilidade e probabilidade a priori são obtidas nessa etapa, permitindo que o classificador probabilístico seja utilizado.

A seguir, as etapas acima são descritas com detalhes.

4.1 DETECÇÃO DE FACE

A primeira etapa do algoritmo, conforme visto na Figura 4.1 é a detecção de face. A sua posição e tamanho são conhecidos pelo uso da técnica proposta em (BINS et al., 2009). Dessa forma, pode-se estimar a posição e tamanho do corpo da pessoa também, obtendo uma região de interesse onde a silhueta deve estar contida. Isso reduz significativamente o espaço de busca, o que reduz o custo computacional e aumenta a precisão do sistema.

Figura 4.2: Exemplo da detecção de face em diferentes quadros da seqüência de vídeo.

Um exemplo dos resultados desse método pode ser visto na Figura 4.2. Pode-se perceber que o algoritmo é capaz de manter a proporção da face detectada durante o vídeo mesmo quando há translação 3D (aproximação e afastamento da pessoa em relação à câmera), como no quarto e quinto caso da figura.

4.2 OBTENDO A REGIÃO DE INTERESSE

A partir do ponto central da face detectada do interlocutor, uma máscara (*template*) pré-definida em formato de Ω, criada a partir de dados antropométricos, é posicionada sobre a imagem, de tal forma que o centro da face da máscara coincida com o centro da face da pessoa. Como essa máscara foi criada manualmente, ela tem tamanho fixo. Então, para que ela seja posicionada de forma satisfatória, ela deve ser escalada para adequar-se ao tamanho da pessoa. Isso é feito utilizando um fator de escala λ de acordo com o tamanho da face detectada, de forma que:

$$\lambda = s_f / s_m * c, \tag{4.1}$$

onde s_f e s_m são os tamanhos da face do interlocutor e da máscara, respectivamente. A constante $c = 1,7$ é um fator de correção definido empiricamente. A partir de então, pode-se efetuar a translação dos pontos p_i da máscara original para os novos pontos p'_i, de forma que:

$$\boldsymbol{p'_i} = \boldsymbol{c_f} + ((\boldsymbol{p_i} - \boldsymbol{c_m})\lambda), \tag{4.2}$$

onde $\boldsymbol{p_i}$ é o ponto i da máscara, $\boldsymbol{c_f}$ é o ponto central da face do interlocutor, e $\boldsymbol{c_m}$ é o ponto central da face da máscara.

Os pontos escalados e transladados da máscara $\boldsymbol{p'_i}$ são utilizados como pontos de controle para a criação de curvas de Bézier. Isso é feito para tornar a curva da máscara mais suave, o que auxiliará na construção do espaço de busca do algoritmo de Dijkstra, mostrado mais adiante. Como visto anteriormente, a curva de Bézier é dada pela Equação (2.20), a qual utiliza quatro pontos de controle. Portanto, a cada quatro pontos da máscara, tem-se uma curva.

(a) Detecção da face (b) Posicionamento da máscara

Figura 4.3: Exemplo do posicionamento da máscara sobre a pessoa.

Retas perpendiculares são posicionadas sobre as curvas criadas, sendo que o tamanho dessas retas é proporcional ao tamanho da face do locutor (mais especificamente, o tamanho é metade do raio da face encontrada). O uso das curvas de Bézier define linhas

suaves, ou seja, sem transições bruscas, não prejudicando o posicionamento das retas perpendiculares. Isso permite com que seja criado uma região de interesse adequada e bem definida.

Sobre cada reta perpendicular i, n nodos são distribuídos. Cada nodo $n_j(i)$ pode conectar-se aos nodos da reta perpendicular adjacente $i+1$. O conjunto de retas perpendiculares e seus nodos criam um grafo $G = (N, E)$. Para cada aresta e, um peso $w(e)$ é calculado baseado na média da energia dos píxeis sob e. Um exemplo do posicionamento da máscara pode ser visto na Figura 4.3(b). A região em amarelo claro representa a região de interesse (ROI), definida pela máscara (curva laranja) e retas perpendiculares (em azul), onde será feita a busca pelo caminho do grafo que maximiza a energia.

4.3 ENERGIAS

Neste trabalho, uma técnica de programação dinâmica (algoritmo de Dijkstra) será utilizada para a segmentação da pessoa. Um dos maiores desafios deste trabalho é determinar as informações que serão extraídas da imagem para formar uma energia que guiará o algoritmo da programação dinâmica. Uma abordagem direta é a utilização de detectores de bordas, já que assim como descrito em Gonzalez e Woods (2002), é uma das técnicas mais simples de identificação de objetos em uma cena. Para isso, será utilizado o filtro de Sobel.

Ainda, como trata-se da segmentação da pessoa em um vídeo, algum tipo de informação temporal também está presente. Neste caso, pode-se detectar movimentos da pessoa, já que parte-se do princípio que é o único movimento que estará na cena. Além disso, conforme pode ser visto em trabalhos similares, o componente probabilístico demonstra-se ser uma informação relevante para a segmentação, portanto, também será utilizado. Nesta seção será explicado, em detalhes, como as energias citadas foram obtidas da imagem.

4.3.1 Operador Sobel

A aplicação do filtro de Sobel pode ser feita na direção horizontal e vertical, ou seja, obtém-se as coordenadas do vetor gradiente para cada *pixel* da imagem. Para obter o mapa de bordas, portanto, calcula-se o módulo desse gradiente através das Equações (2.2) e (2.3). Assim como descrito na seção 2.2, esse operador utiliza somente a informação de luminância da imagem, ou seja, elas são convertidas para escala de cinza antes do filtro de Sobel ser aplicado.

Ainda, antes da aplicações do operador, um filtro Gaussiano de tamanho 11×11 é aplicado sobre as imagens para eliminar possíveis ruídos, já que, conforme apresentado por Gonzalez e Woods (2002), derivadas são muito suscetíveis a ruídos. Além disso, a aplicação do filtro Gaussiano cria uma suavização na imagem, o que acaba por gerar bordas mais fortes no resultado do filtro de Sobel. Alguns exemplos podem ser vistos na

Figura 4.4

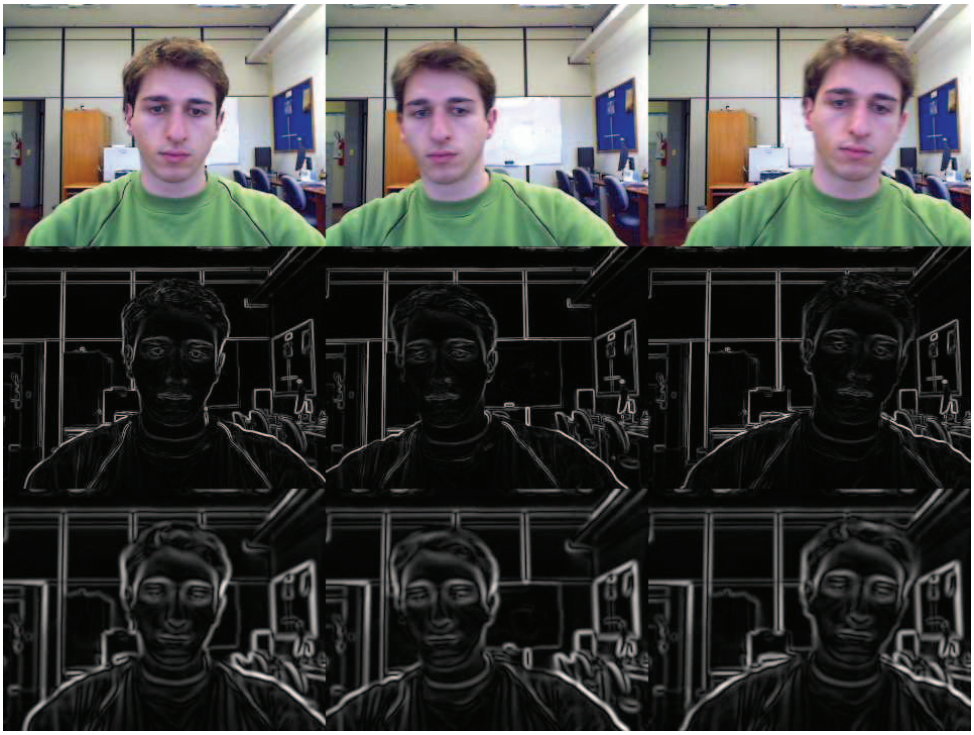


Figura 4.4: Exemplo da aplicação do filtro de Sobel. Primeira linha: quadro original; Segunda linha: resultado do filtro de Sobel sem o filtro Gaussiano. Terceira linha: filtro de Sobel após a aplicação do filtro Gaussiano.

4.3.2 Movimento

Outra informação extraída do vídeo é a informação de movimento. Isso é feito simplesmente calculando a diferença entre dois *frames*, $f(x, y, t)$ e $f(x, y, t+1)$, nos tempos t e $t+1$. No entanto, essa detecção não é precisa devido a ruídos e variação de iluminação na captura do vídeo. Para reduzir influencias externas, antes da subtração, novamente um filtro Gaussiano (de tamanho 11×11) é aplicado sobre os quadros. Logo após, é feita uma limiarização da imagem, de tal forma que o resultado levado em consideração é dado por:

$$d_{ij}(x, y) = \begin{cases} |f(x, y, t) - f(x, y, t+1)|, & se \ \ |f(x, y, t) - f(x, y, t+1)| > T; \\ 0, & caso \ contrário; \end{cases} \tag{4.3}$$

onde através de testes empíricos definiu-se $T = 10$. Um exemplo visual dessa abordagem pode ser visto na Figura 4.5.

Considera-se que houve movimento somente se a quantidade de píxeis não nulos resultantes for superior a 30% do total de píxeis da imagem. Esse valor demonstrou-se adequado nos diferentes testes que foram feitos.

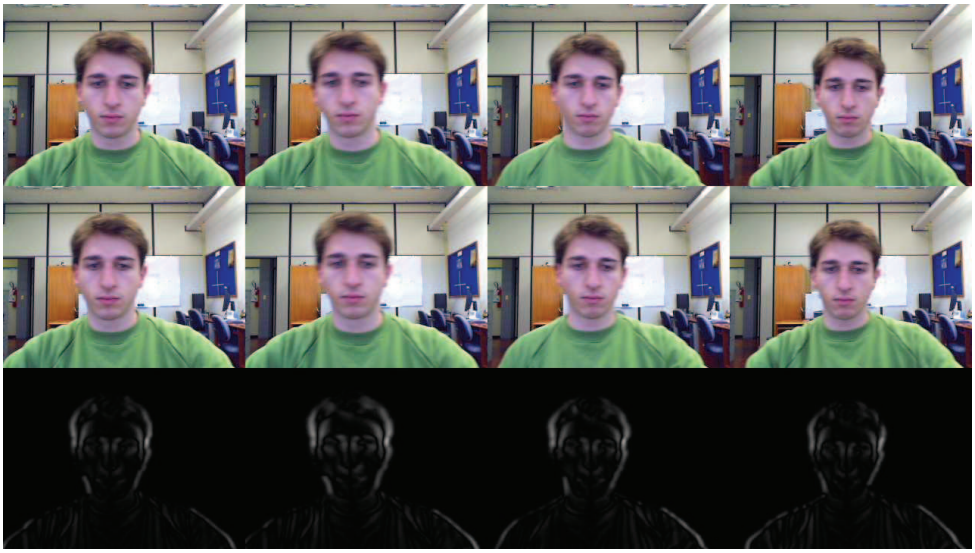


Figura 4.5: Exemplo do resultado ao longo de diversos quadros. Primeira linha: quadro $f(x, y, t)$; Segunda linha: quadro $f(x, y, t + 1)$; Terceira linha:$|f(x, y, t) - f(x, y, t + 1)|$.

Partindo-se do princípio que somente uma pessoa estará na cena, pode-se pressupor que a informação de movimento é mais confiável do que a informação das bordas. Portanto, quando ocorre movimento na imagem, o resultado do algoritmo de Dijkstra tende a ser mais preciso. Pode-se aproveitar essa característica, então, para atualizar a máscara na qual as retas perpendiculares são posicionadas, de forma que a nova máscara passe a ter o formato da silhueta resultante do quadro anterior. Assim, o sistema pode adaptar-se a diferentes formatos que a região que representa o locutor possui.

4.3.3 Probabilidade

Informação de movimento e bordas são relevantes para a detecção da pessoa na imagem. Porém, se não há movimento, a informação de borda por si só não é capaz de retornar resultados muito precisos, principalmente em ambientes com diversos objetos ao fundo. Tendo em vista que em trabalhos similares, como em Criminisi et al. (2006) e Yin et al. (2007), a informação probabilística mostrou-se eficaz, optou-se por adicionar essa componente na solução proposta.

Conforme apresentado anteriormente na seção 2.3, a teoria de decisão Bayesiana foi utilizada, a qual possui dois elementos principais: função densidade de probabilidade e probabilidade *a priori*.

De acordo com Jain (1989) e conforme apresentado na seção 2, o histograma de uma imagem representa a freqüência relativa de ocorrência de diferentes níveis de cinza na imagem. Dessa forma, pode-se utilizar um histograma 3D, contendo os componentes R,

G e B da imagem, como a função densidade de probabilidade, assim como também é feito em Ross e Kaelbling (2005). Porém, deve-se levar em consideração somente os píxeis que pertencem à classe em questão. Para isso, é necessário conhecer previamente as regiões pertencentes ao primeiro e segundo plano, o que não é possível determinar com precisão em um primeiro momento. Portanto, optou-se por criar o classificador probabilístico somente quando há movimento na imagem, já que ela é mais confiável que a informação de borda. Então, a adaptação do classificador é feito com base na segmentação resultante daquele quadro. Dessa forma, o sistema "aprende" a cor da pessoa e do fundo, o que, teoricamente, deve contribuir para o aumento de precisão do algoritmo. Além disso, toda vez que o movimento é detectado em algum quadro, o histograma é atualizado para adequar o sistema a possíveis mudanças que ocorreram na cena. A fim de tornar essa adaptação mais suave, ao invés de simplesmente substituir o histograma antigo pelo novo, é feita uma soma ponderada dos histogramas $p_{t1}(r_k)$ e $p_{t2}(r_k)$:

$$p(r_k) = (1 - \zeta)p_{t1}(r_k) + \zeta p_{t2}(r_k), \tag{4.4}$$

onde $p_{t1}(r_k)$ é o histograma antigo, $p_{t2}(r_k)$ é o novo, e $0 \leq \zeta \leq 1$ é a taxa de atualização.

Quanto à probabilidade *a priori*, ela geralmente é definida pela razão entre a quantidade de amostras da classe ω_j e o total de amostras. Porém, essa informação não está disponível previamente já que nenhum tipo de treinamento é efetuado antes do funcionamento do sistema. No entanto, dado que o sistema lida com imagens, pode-se explorar algum tipo de informação espacial na obtenção da probabilidade *a priori*. Portanto, a Transformada Distância da imagem foi utilizada para determinar o quão perto do locutor um determinado ponto está.

Conforme apresentado em Bradski e Kaehler (2008), a Transformada Distância recebe como entrada uma imagem binária e o resultado é uma nova imagem em que cada *pixel* representa a distância entre ele e o *pixel* preto mais próximo. Para obter a imagem binária que servirá de base para o cálculo da TD, utiliza-se o resultado da segmentação baseada em informação de movimento e bordas. Mais especificamente, quando ocorre o primeiro movimento, espera-se que o resultado seja preciso o suficiente para criar a imagem binária separando o locutor do fundo. Portanto, são criadas duas imagens TD_{pp} e TD_{sp}, representando as Transformadas Distâncias do primeiro e segundo plano normalizadas, respectivamente, apresentadas na segunda coluna da Figura 4.6.

Para os píxeis pertencentes ao primeiro plano do resultado da segmentação, a probabilidade *a priori* é dada por:

$$P_{pp}(x, y) = 0,5 + 0,5TD_{pp}(x, y), \tag{4.5}$$

$$P_{sp}(x, y) = 1 - P_{pp}. \tag{4.6}$$

Analogamente, tem-se que para os píxeis pertencentes ao segundo plano, a probabilidade *a priori* é calculada através de:

$$P_{sp}(x, y) = 0,5 + 0,5TD_{sp}(x, y), \tag{4.7}$$

$$P_{pp}(x, y) = 1 - P_{sp}. \tag{4.8}$$

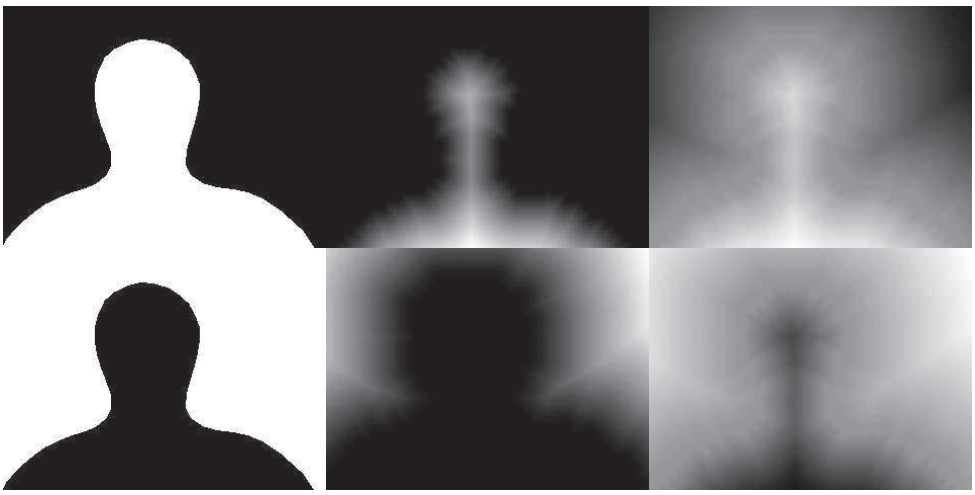


Figura 4.6: Primeira linha refere-se ao cálculo da probabilidade *a priori* do primeiro plano, e a segunda, do segundo plano. Primeira coluna: resultado da segmentação utilizando informação de movimento; Segunda coluna: Transformada Distância; Terceira coluna: Probabilidade *a priori*.

Portanto, pontos que se encontram exatamente no centro da pessoa possuem $P_{pp} = 1$, enquanto que quanto mais afastado do centro, menor será sua probabilidade. Da mesma forma, $P_{sp} = 1$ para píxeis afastados do centro da imagem, e quanto mais próximos da pessoa estiverem, mais próximos de 0 será seu valor. O valor $0,5$ é somado nas Equações (4.6) e (4.6) pelo fato de que o valor da Transformada Distância exatamente na borda é 0. Assim, a regra de Bayes é respeitada, sendo que para qualquer ponto $P_{pp} + P_{sp} = 1$. Um exemplo do resultado obtido pode ser visto na terceira coluna da Figura 4.6.

Dado que tem-se os dois elementos principais do teorema de Bayes, pode-se calcular a probabilidade de um *pixel* pertencer ao primeiro plano utilizando a Equação 2.5.

4.4 SEGMENTAÇÃO UTILIZANDO O ALGORITMO DE DIJKSTRA

Conforme foi mostrado na Figura 4.3, a região de interesse define um grafo, no qual os pesos das arestas são dados pelas energias extraídas da imagem, mais precisamente, nos pontos abaixo da própria aresta, de forma que:

$$peso(e) = \frac{1}{n}\sum_{i=1}^{n} E_{f_i}, \tag{4.9}$$

onde n é o total de píxeis sob a aresta e, e E_{f_i} é a energia final do *pixel* i, dado por:

$$E_{f_i} = w_e E_{ei} + w_m E_{mi} + w_p E_{p_i}, \tag{4.10}$$

onde E_e representa a informação de borda, E_m representa a informação de movimento, e E_p representa a informação probabilística. Os pesos de cada energia foram obtidos empiricamente, sendo $w_e = 0.1$, $w_m = 0.4$ e $w_p = 0.5$. Optou-se por dar um peso maior para as informações de movimento e probabilidade, por serem mais confiáveis.

A informação de borda utilizada poderia ser simplesmente o valor do módulo do gradiente. Porém, dessa forma, a orientação do gradiente não estaria sendo considerada. Além disso, comprovou-se através de testes preliminares que a utilização do módulo somente não era precisa, devido ao fato de que em ambientes com muitos objetos ao fundo, poderiam existir bordas mais fortes do que as da própria pessoa. Portanto, optou-se por uma nova abordagem onde a informação de borda é dada por:

$$E_e = \boldsymbol{N} \cdot \boldsymbol{\nabla f} = |N|\,|\nabla f| \cos\theta, \tag{4.11}$$

onde $\boldsymbol{N_i}$ é o vetor normal ao vetor candidato $\boldsymbol{v_i}$ entre dois vértices do grafo, e θ é o ângulo entre os vetores $\boldsymbol{N}$ e $\boldsymbol{\nabla f}$. Dessa forma, espera-se que bordas que não estejam muito alinhadas com os vetores candidatos $\boldsymbol{v_i}$ sejam desconsideradas, já que provavelmente não fazem parte da silhueta da pessoa. Dado que o cálculo é baseado no produto interno entre os vetores, a Equação (4.11) retorna valores mais altos quando o vetor candidato $\boldsymbol{v_i}$ estiver completamente alinhado com a borda, já que $\theta = 0$. Um exemplo dessa abordagem pode ser visto na Figura 4.7.

Quanto à informação de movimento, percebeu-se que utilizando apenas a diferença entre os quadros não obteve-se melhoras significativas. Inclusive, muitas vezes o resultado ficava pior. Isso ocorria porque existia informação de movimento não somente nas bordas, mas também na parte interior da região correspondente à pessoa. Dessa forma, a silhueta tinha uma tendência a passar por dentro da imagem da pessoa, o que não é visualmente agradável e comprometia seriamente a precisão do sistema. Portanto, de forma a evitar esse problema, a informação de movimento é mais forte quando tende às bordas, de forma que:

$$E_m = \frac{|f(x,y,t) - f(x,y,t+1)|}{l}, \tag{4.12}$$

onde l representa o nível do vértice no grafo.

A última informação levada em consideração é a de borda. Para incorporar essa componente na informação geral, o filtro de Sobel foi aplicado sobre o resultado da imagem probabilística, já que píxeis com grande probabilidade são mais claros. Portanto, de forma

análoga à Equação (4.11), tem-se que:

$$E_p = \boldsymbol{N} \cdot \nabla f_p = |N|\,|\nabla f_p| \cos\theta, \tag{4.13}$$

onde ∇f_p é o vetor gradiente da imagem resultante do operador Sobel, tendo como entrada a imagem probabilística.

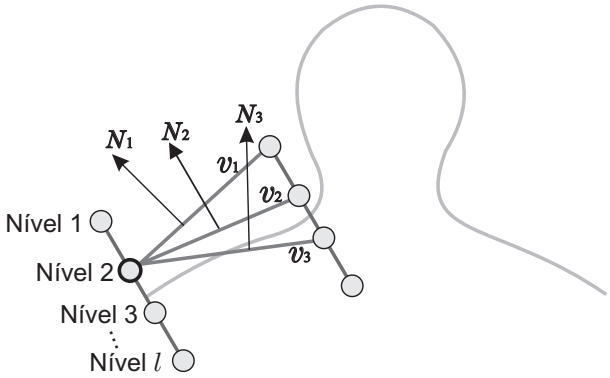


Figura 4.7: Exemplo de vetores candidatos v_i a partir de um vértice.

O algoritmo de Dijkstra procura encontrar o caminho com o menor custo por todo o grafo. No entanto, pelo fato de que as energias utilizadas neste trabalho retornam valores altos nos pontos onde a silhueta é mais adequada, modificou-se o algoritmo para que ele encontre o caminho de maior custo. Porém, o funcionamento geral do algoritmo continua igual.

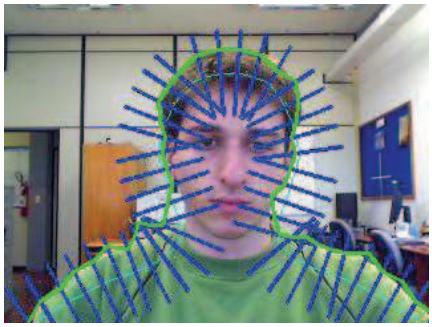


Figura 4.8: Exemplo do resultado do algoritmo de Dijkstra (em verde).

Para encontrar o caminho de maior custo, calcula-se os caminhos possíveis de qualquer um dos vértices da linha perpendicular da esquerda até qualquer um dos vértices da linha perpendicular da direita. A fim de diminuir ainda mais o custo computacional do sistema, cada vértice de um determinado segmento perpendicular pode se conectar somente a uma parte dos vértices do próximo segmento. Um exemplo disso pode ser visto na Figura 4.7, na qual o algoritmo analisa o caminho entre um vértice de um segmento perpendicular e somente três do segmento seguinte.

Um exemplo de silhueta encontrada pelo algoritmo pode ser visto na Figura 4.8. Pode-se notar que o resultado é bem ruidoso, o que não é visualmente atraente, dado que a silhueta da pessoa é uma curva suave.

4.5 FILTRO PASSA-BAIXA

Após a execução do algoritmo de Dijkstra, tem-se o resultado da silhueta detectada. Porém, como pôde ser visto, esse resultado tende a ser ruidoso no sentido de que a silhueta possui descontinuações bruscas em seu decorrer. Isso prejudica a qualidade visual do resultado. Além disso, como a máscara é atualizada ao longo do tempo, ela seria modificada e se tornaria brusca, o que faria com que os segmentos perpendiculares não fossem posicionados de forma satisfatória, fazendo com que as próximas silhuetas detectadas sejam imprecisas e isso tende a piorar no decorrer da execução do algoritmo.

Figura 4.9: Exemplo de suavização da silhueta através de um filtro passa-baixa.

Para evitar esse tipo de problema, após a detecção da silhueta, aplica-se um filtro passa-baixa no resultado obtido, a fim de obter um resultado mais suave, demonstrado na Figura 4.9. O filtro utilizado foi o filtro de Chebyshev (OPPENHEIM et al., 1999) de ordem 2, utilizando uma frequência de corte $0,5\pi$.

4.6 SUBSTITUIÇÃO DO SEGUNDO PLANO

Após a segmentação bem sucedida, tem-se uma curva que descreve a parte da imagem que representa a pessoa e a parte que representa o fundo. Assim, é possível substituir o fundo da imagem por outro fundo qualquer.

A simples substituição dos pontos que estão fora da silhueta detectada como sendo pertencente ao locutor não causa um efeito visual interessante, devido a transição brusca entre a pessoa e o fundo. Portanto, a fim de criar um efeito visual mais agradável, é feito um esmaecimento das imagem na região da silhueta detectada. Isso é feito utilizando a Transformada Distância da imagem, de forma que a cor do *pixel* na borda é dada por:

$$c(x,y) = f(x,y)w_{pp} + f'(x,y)w_{sp}, \tag{4.14}$$

onde $f(x, y)$ é a imagem original, $f'(x, y)$ é a imagem que será colocada em segundo plano, e w_{pp} e w_{sp} são os pesos respectivos de cada imagem, sendo que $w_{pp} + w_{sp} = 1$. Para os píxeis que se encontram no primeiro plano, tem-se que:

$$w_{pp} = \begin{cases} 1 & se\ DT_{pp} > L, \\ 0,5 + 0,5DT_{pp}/L, & caso\ contrário; \end{cases} \tag{4.15}$$

$$w_{sp} = 1 - w_{pp}, \tag{4.16}$$

onde $L = 3$ é um limiar que define a distância máxima que os píxeis devem estar da silhueta para que sofram o esmaecimento. De forma análoga, tem-se para os píxeis do segundo plano que:

$$w_{sp} = \begin{cases} 1 & se\ DT_{sp} > L, \\ 0,5 + 0,5DT_{sp}/L, & caso\ contrário; \end{cases} \tag{4.17}$$

$$w_{pp} = 1 - w_{sp}. \tag{4.18}$$

Na Figura 4.10 pode-se ver a diferença de alguns quadros com e sem o efeito de esmaecimento.

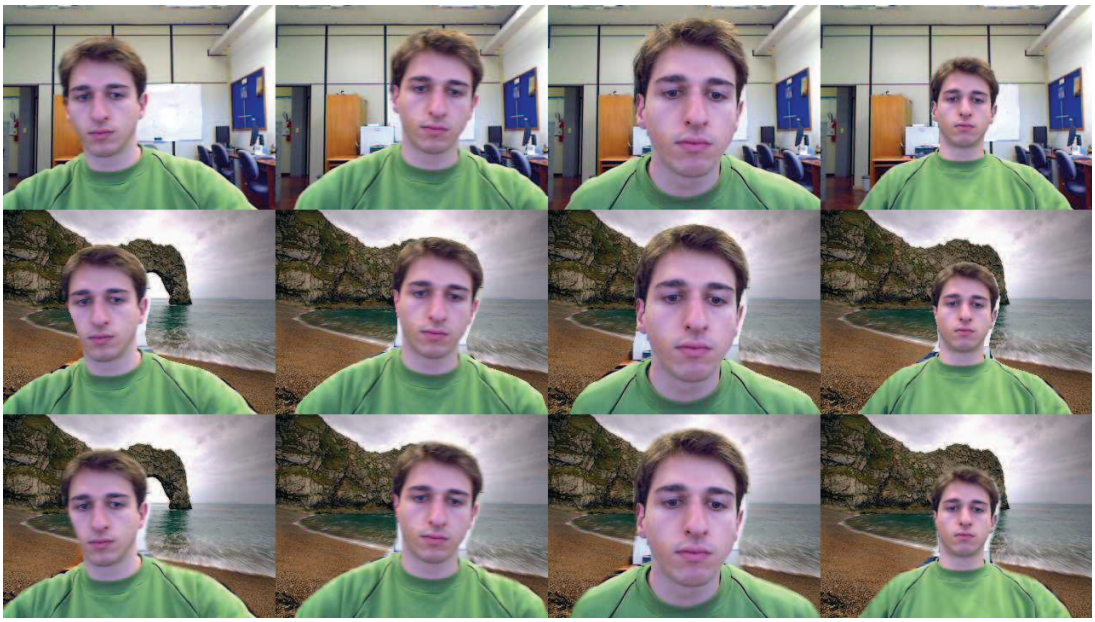


Figura 4.10: Exemplo de substituição do segundo plano. Primeira linha: quadros originais; Segunda linha: Substituição do fundo sem o efeito de esmaecimento; Terceira linha: Substituição do fundo com efeito.

5 RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os resultados obtidos com o sistema desenvolvido. Um grande desafio deste trabalho foi determinar como seriam feitas as avaliações. Uma métrica bastante utilizada são os *ground-truths*, como pode ser visto em Kolmogorov et al. (2005), Criminisi et al. (2006) e Yin et al. (2007). Através dessa métrica, os píxeis de cada quadro de um vídeo são previamente classificados como pertencentes ao primeiro ou segundo plano e o resultado do algoritmo é comparado a eles. Essa técnica é interessante para se ter uma noção da precisão do sistema em situações reais, portanto ela foi escolhida como uma das métricas para avaliar os resultados do sistema proposto.

No entanto, a geração de *ground-truths* demanda um grande esforço, já que a segmentação da pessoa nos vídeos é feita de forma manual. Além disso, ela não é 100% precisa, mesmo que a tarefa de determinar os píxeis que fazem parte da pessoa seja relativamente intuitiva. Portanto, um modelo humanóide 2D simples foi criado para a geração de vídeos em ambientes controlados, permitindo que os resultados fossem analisados de forma precisa.

5.1 DESENVOLVIMENTO DO HUMANÓIDE

Conforme citado anteriormente, apesar de a segmentação manual da imagem ser um processo relativamente intuitivo, não é possível garantir que todos os píxeis foram classificados corretamente, como por exemplo os que se encontram exatamente na fronteira entre a pessoa e o fundo. Um exemplo disso pode ser visto na Figura 5.1, a qual apresenta a imagem de uma pessoa se movimentando. Tomando por exemplo a região do cabelo, não é possível determinar com clareza quais pontos pertencem à pessoa ou ao fundo.

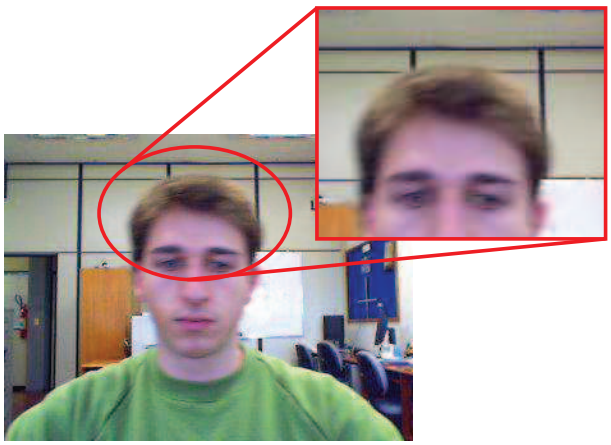


Figura 5.1: Exemplo de situação em que não é possível determinar facilmente se um determinado *pixel* pertence ao primeiro ou segundo plano.

Tendo em vista o exemplo citado anteriormente, para obter resultados analíticos

precisos, vídeos artificiais foram criados e métricas mais precisas foram utilizadas, como a métrica do supremo (RUDIN, 1976), a qual é utilizada para calcular a maior distância entre duas funções.

Os vídeos artificiais foram criados utilizando um modelo humanóide 2D simples, composto somente pelo tronco, cabeça e pescoço, como pode ser visto na Figura 5.2. O objetivo desse humanóide é fornecer vídeos com os quais possa-se obter resultados precisos sobre a segmentação, já que como ele é gerado através de funções matemáticas conhecidas, pode-se calcular com exatidão o erro do sistema. Na Figura, pode-se ver os parâmetros que podem ser alterados, como: a altura e largura dos ombros h_t e l, respectivamente; a altura do pescoço h_p; o ângulo de abertura dos braços esquerdo e direito θ_1 e θ_2; o ângulo de inclinação do pescoço θ_3; e posição e tamanho da face c_f e r, respectivamente.

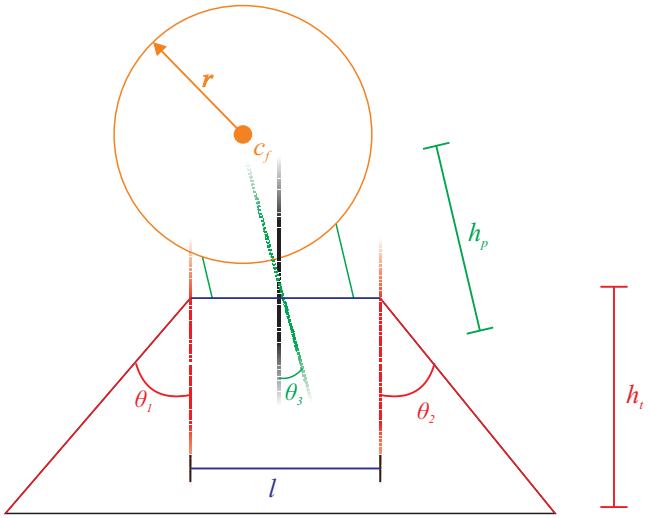


Figura 5.2: Representação do modelo humanóide criado.

Sendo assim, pode-se simular o movimento de abertura de braços, movimentação do pescoço e corpo do humanóide. Além disso, é possível alterar o fundo da imagem, bem como o preenchimento do tronco, simulando a utilização de uma roupa com textura, por exemplo.

A utilização do humanóide permite que os resultados analisados sejam precisos, já que cada ponto é conhecido através de funções matemáticas. Neste trabalho, a análise dos resultados dos vídeos gerados com o humanóide foi dada pela métrica do supremo, calculando as distâncias entre os pontos do resultado suavizado do algoritmo de Dijkstra e do corpo do humanóide. Sendo assim, o erro final de cada seqüência de vídeo é dado pelo erro máximo levando em consideração todos os quadros, de forma que:

$$E = \max_i \max_x ||s_i(x) - h_i(x)||, \tag{5.1}$$

onde $s_i(x)$ representa a silhueta gerada pelo algoritmo de Dijkstra já suavizada pelo filtro passa-baixa no quadro i e $h_i(x)$ representa a função que descreve o humanóide, como pode

ser visto na Figura 5.3. Na prática, calcula-se as distâncias entre o ponto de intersecção da reta perpendicular com o humanóide, representado por um ponto amarelo na Figura 5.3, e o ponto de intersecção da reta perpendicular com a curva da silhueta final, representado por um ponto laranja. Esse processo é repetido para todas as retas perpendiculares em cada quadro.

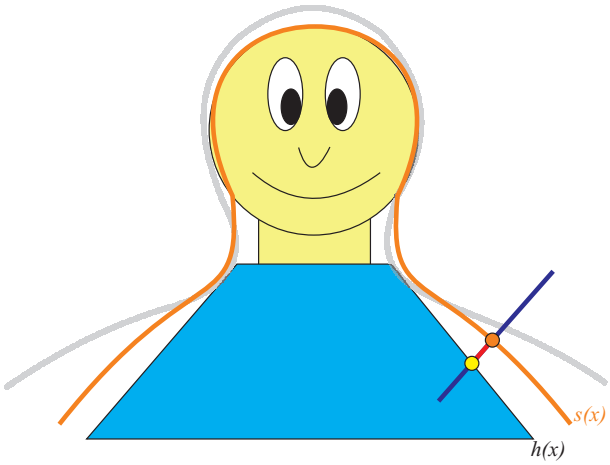


Figura 5.3: Exemplo do cálculo da métrica do supremo. A curva em laranja representa o resultado do algoritmo de Dijkstra suavizado $s_i(x)$. Os pontos relativos à silhueta real do humanóide são dados pela função $h_i(x)$. O erro da métrica do supremo nesse caso está representado pelo segmento de reta vermelho.

5.2 TESTES COM O HUMANÓIDE

O sistema desenvolvido é altamente parametrizável, como por exemplo: quantidade de retas perpendiculares sobre a máscara, número de pontos sobre cada reta, quantidade de nodos que o algoritmo de Dijkstra pode levar em consideração para calcular o custo entre uma reta perpendicular e a próxima, pesos de cada energia, parâmetros do filtro passa-baixa, taxa de aprendizagem do classificador probabilístico, entre outros. É muito difícil determinar uma combinação ótima desses parâmetros, já que isso depende fortemente do ambiente em que o algoritmo está inserido. Em um primeiro momento, optou-se por realizar testes variando a quantidade de segmentos perpendiculares Ns sobre o *template* e o número de pontos Np sobre cada uma delas. Além disso, conforme apresentado na seção 4.4, cada nodo de uma reta perpendicular pode se ligar a somente três nodos da próxima. Porém, foram feitos testes diminuindo essa restrição de forma que cada nodo pudesse levar em consideração até 30% dos nodos do próximo segmento perpendicular.

Para o primeiro teste, foi gerado um vídeo de cem quadros com o humanóide efetuando um movimento lateral e movimentando a cabeça em um fundo de cor homogênea. A Figura 5.4 apresenta alguns quadros da seqüência de vídeo, e a Figura 5.5 apresenta alguns resultados obtidos.

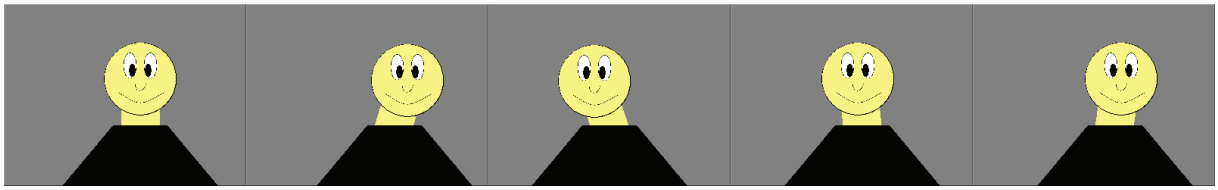
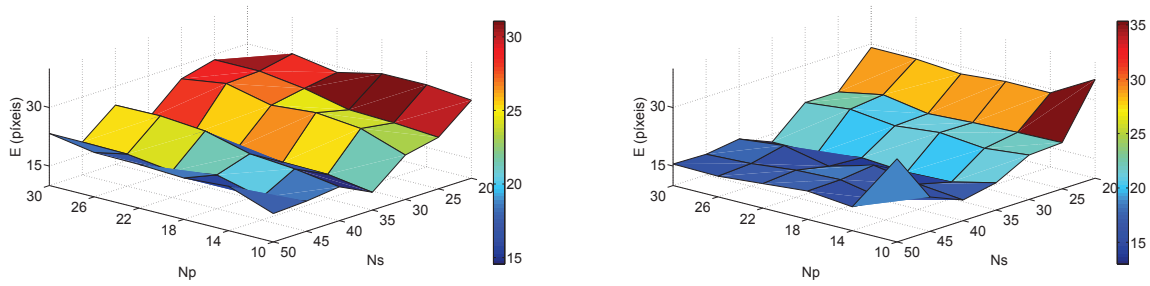


Figura 5.4: Alguns quadros da primeira seqüência de teste gerada.



(a) Restrição de apenas três nodos.

(b) Restrição de 30%.

Figura 5.5: Superfícies representando o erro obtido através da métrica do supremo para a primeira seqüência de teste.

Como pode-se perceber nos gráficos apresentados, o erro não muda consideravelmente com relação ao número de pontos sobre o segmento perpendicular, mas sim com relação à quantidade de retas sobre a máscara. Ainda, nota-se que o erro começa elevado quando $N_s = 20$, atingindo um ponto ótimo em torno $N_s = 35$ e após esse ponto o erro tende a aumentar novamente. Um fato interessante é que na Figura 5.5(b) o erro, em geral, é menor no que na Figura 5.5(a), mostrando que para esse caso em específico restringir a ligação de um nodo de uma reta perpendicular com apenas três da próxima reta causa uma perda de precisão.

O teste apresentado foi razoavelmente simples, já que muitos fatores que podem influenciar no desempenho do sistema foram eliminados, tais como ruídos, texturas, etc. Em um segundo momento, foram gerados vídeos procurando simular a ondulação que as roupas possuem quando vestidas por uma pessoa para testar a precisão do algoritmo quando se tem diversas curvas ao longo do corpo da pessoa. Para isso, uma flutuação periódica foi adicionada na criação das retas do humanóide que representam os braços esquerdos e direito. De fato, cada braço foi criado utilizando a equação paramétrica da reta, portanto, cada ponto $P(x, y)$ sobre a reta é dado por:

$$P = P_0 + t\boldsymbol{v}, \tag{5.2}$$

onde P_0 é o ponto inicial do segmento de reta, e $\boldsymbol{v}$ é o vetor diretor. Adicionando-se o

sinal periódico, tem-se que a coordenada x do ponto é dada por:

$$P_x = P_{0x} + tv_x + Asin(\theta)sin(\omega_x t), \tag{5.3}$$

onde A é a amplitude do sinal, θ é o ângulo da reta que representa o braço, e ω_x é a freqüência do sinal na direção do eixo horizontal. Analogamente, tem-se que:

$$P_y = P_{0y} + tv_y + Acos(\theta)sin(\omega_y t). \tag{5.4}$$

Alguns quadros dos vídeos gerados podem ser vistos na Figura 5.6.

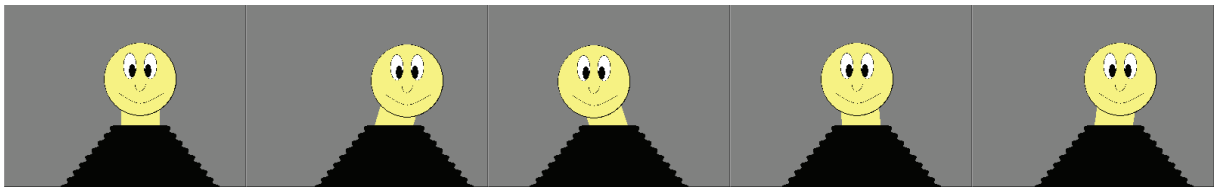
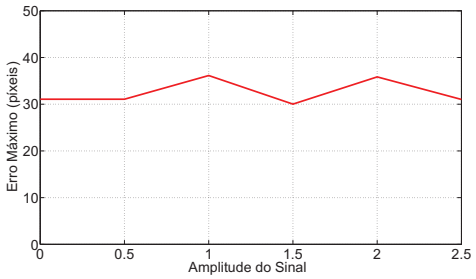
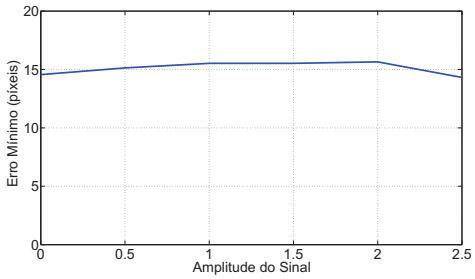


Figura 5.6: Alguns quadros para o segundo teste do sistema. Neste caso, o vídeo artificial foi gerado utilizando $A = 2.5$.

Portanto, o primeiro teste foi expandido para diferentes cenários com diferentes amplitudes do sinal periódico. Novamente, executou-se o teste com e sem a restrição de que cada nodo pode conectar-se a somente outros três nodos do próximo segmentos perpendicular e os respectivos resultados podem ser vistos nas Figuras 5.9 e 5.10. Mais uma vez, a remoção da restrição ocasionou uma melhora na precisão do sistema.



(a) Erro máximo de cada teste apresentado na Figura 5.9

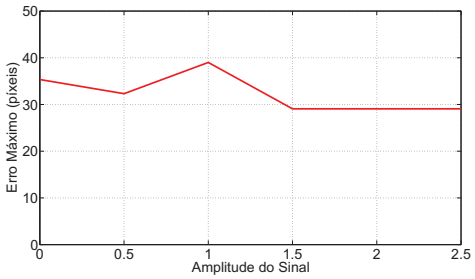


(b) Erro mínimo de cada teste apresentado na Figura 5.9

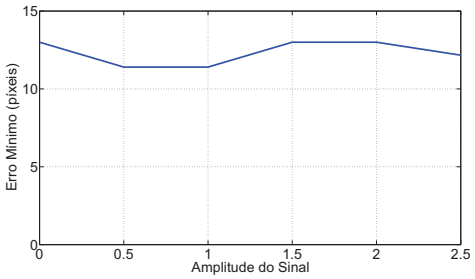
Figura 5.7: Tendência do erro à medida que a amplitude do sinal periódico é aumentada utilizando a restrição de três nodos.

Para se ter uma idéia da tendência do erro em função do aumento da amplitude do erro, os erros máximos de cada um dos gráficos das Figuras 5.9 e 5.10 foram selecionados e mostrados nas Figuras 5.7(a) e 5.8(a). Analogamente, os erros mínimos também foram selecionados e são apresentados nas Figuras 5.7(b) e 5.8(b). Nos gráficos apresentados pode-se ver que o erro não varia significativamente em função da amplitude do sinal

periódico, o que indica que a precisão do algoritmo não é afetada significativamente por pequenas variações na superfície do humanóide, ou em um caso real, por ondulações na roupa, por exemplo. Também, como já era esperado, pode-se notar que o erro mínimo é menor quando a restrição da ligação entre os nodos de cada segmento perpendicular é eliminada.

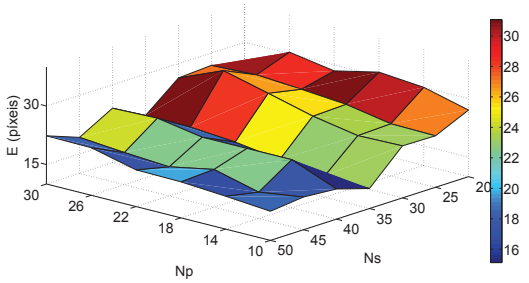


(a) Erro máximo de cada teste apresentado na Figura 5.10

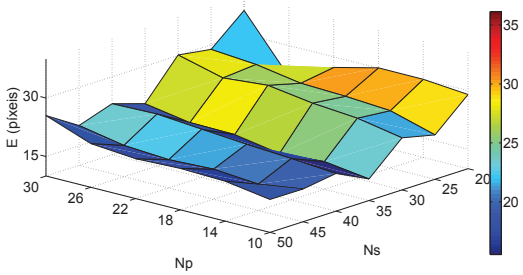


(b) Erro mínimo de cada teste apresentado na Figura 5.10

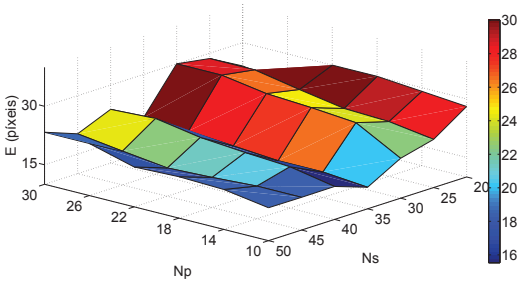
Figura 5.8: Tendência do erro à medida que a amplitude do sinal periódico é aumentada, removendo-se a restrição.



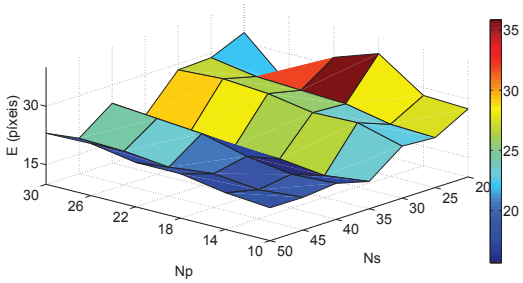
(a) $A = 0.5$



(b) $A = 1$

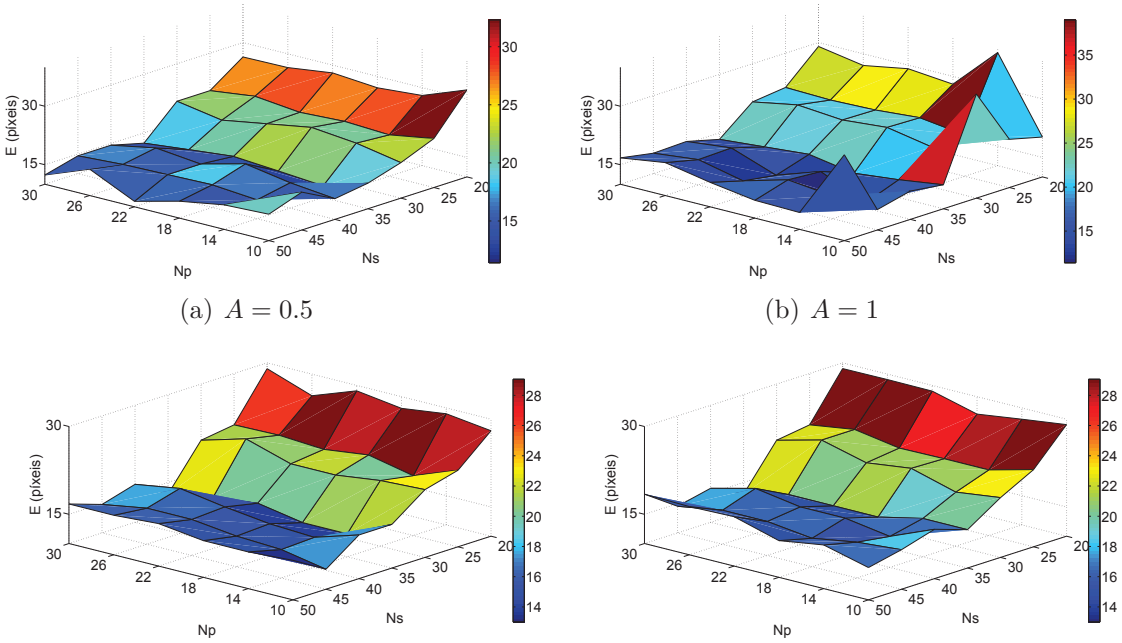


(c) $A = 1.5$



(d) $A = 2$

Figura 5.9: Superfícies do erro obtido no segundo teste com a restrição de que cada nodo pode conectar-se a somente três da próxima reta perpendicular.



(a) $A = 0.5$ (b) $A = 1$

(c) $A = 1.5$ (d) $A = 2$

Figura 5.10: Superfícies do erro obtido no segundo teste, permitindo que até 30% dos nodos da próxima reta perpendicular fossem levados em consideração.

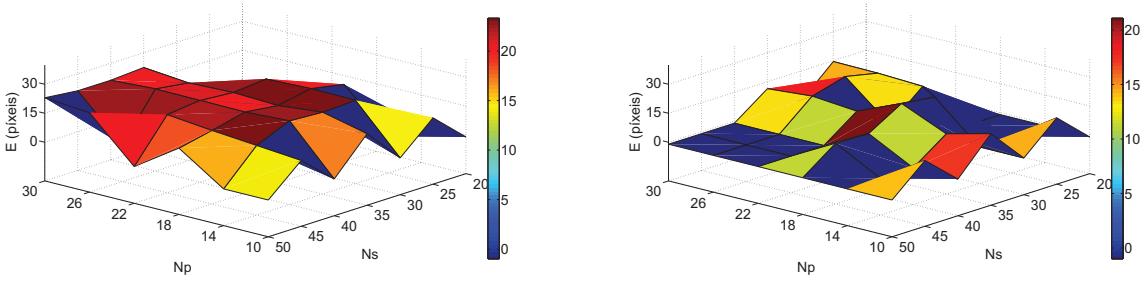
O segundo teste demonstrou que o desempenho do algoritmo não é afetado de forma significativa por pequenas ondulações sobre o humanóide, portanto, para os próximos testes será utilizado $A = 1.5$, já que foi considerado o valor de amplitude que mais se assemelha com uma ondulação de uma roupa real.

A fim de testar o desempenho do algoritmo em fundos não homogêneos, um terceiro teste foi feito com um vídeo contendo retas verticais pretas no segundo plano. Alguns quadros podem ser vistos na Figura 5.11 e os resultados podem ser vistos na Figura 5.12.

Figura 5.11: Quadros da terceira seqüência de teste.

Nesse caso pode-se ver que o desempenho do algoritmo não foi satisfatório. Os valores abaixo de zero nos gráficos da Figura 5.12 representam o fato de que em todos os quadros do vídeo um ou mais segmentos perpendiculares não intersectaram com o humanóide, não sendo possível calcular o erro através da métrica do supremo. Além disso, o fato de um ou mais segmentos perpendiculares não terem intersecção com a pessoa indica que o resultado visual do algoritmo também não foi satisfatório. Um exemplo disso

pode ser visto na Figura 5.12(b), para $N_s = 30$ e $Np = 14$, e a causa desse problema é apresentada na Figura 5.13.



(a) Resultado com restrição.

(b) Resultado sem restrição.

Figura 5.12: Superfícies do erro para o teste com vídeo contendo retas verticais ao fundo.

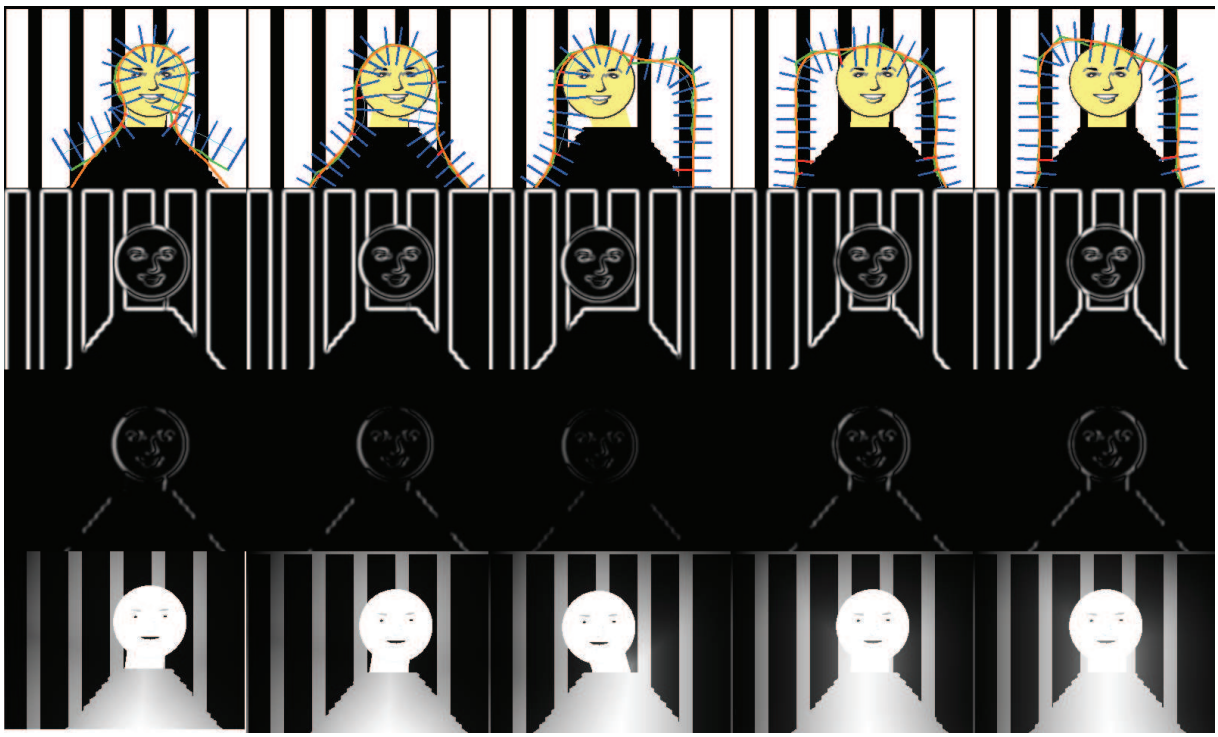


Figura 5.13: Resultado visual para $N_s = 30$ e $Np = 14$. Primeira linha: Resultado do algoritmo de Dijkstra; Segunda linha: Módulo do operador Sobel; Terceira linha: Informação de movimento; Quarta linha: Informação probabilística.

Na primeira linha da Figura 5.13, as curvas em verde representam a silhueta gerada pelo algoritmo de Dijkstra e a curva em laranja, a curva suavizada pelo filtro passa-baixa. É possível perceber que o algoritmo acaba sendo atraído para uma das retas perpendiculares ao fundo, visto que a energia da borda naquela região é forte. Além disso, como elas possuem a mesma cor que o corpo do humanóide, a informação probabilística também acaba sendo forte, o que prejudica a precisão do sistema. Como o algoritmo

também atualiza a região de busca ao longo do vídeo quando há movimento, ela acaba sendo atualizada erroneamente e diversos segmentos perpendiculares não se intersectam mais com o humanóide, fazendo com que o erro aumente cada vez mais ao longo do vídeo. O principal problema encontrado nesse caso é que, como o sistema se baseia em três energias para efetuar a segmentação e duas delas estavam fortemente sugerindo que as retas verticais fossem parte da pessoa, o algoritmo de Dijkstra acabou não sendo preciso. No entanto, um fato interessante é que nesse caso, o resultado utilizando a restrição da ligação entre os nodos, apresentado na Figura 5.12(a), foi melhor, já que não permitiu que o algoritmo criasse silhuetas que fossem muito distintas da máscara.

Um quarto teste, desta vez utilizando retas verticais mais claras ao fundo, foi executado, utilizando a mesma configuração do teste anterior, e os resultados podem ser vistos na Figura 5.14. Novamente, os parâmetros N_p e N_s foram variados e a nova superfície de erros é apresentada na Figura 5.15.

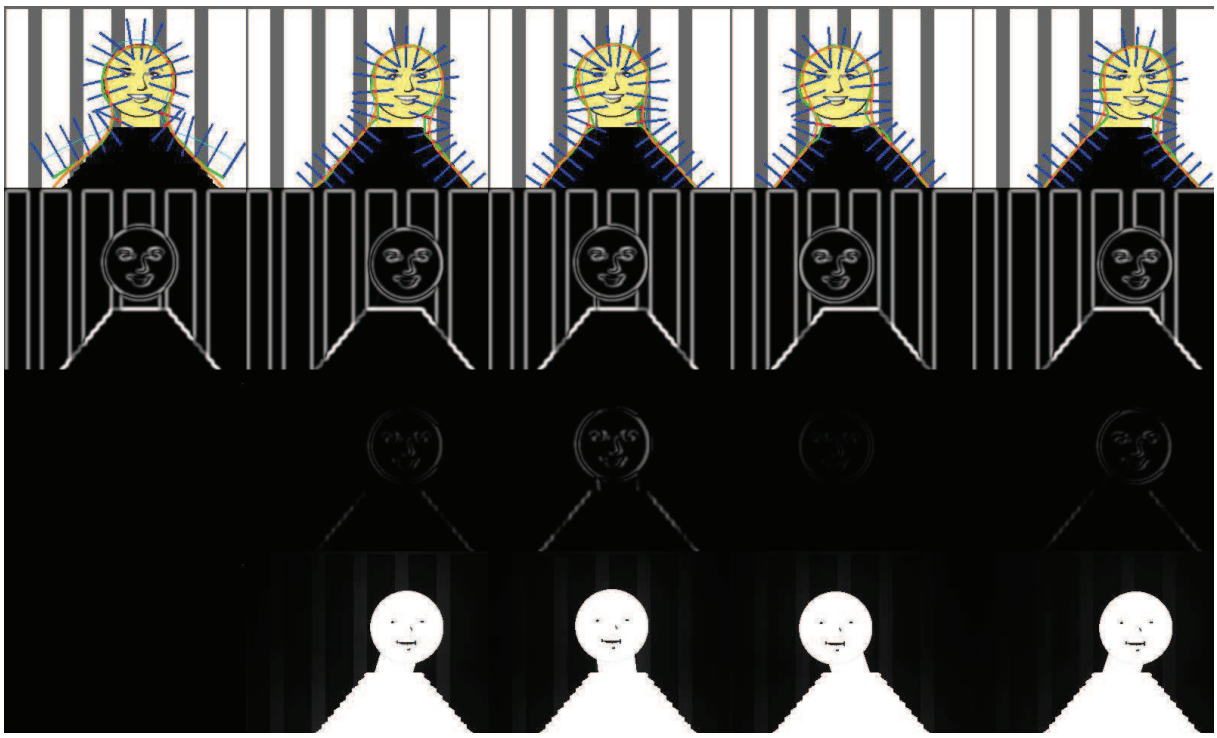


Figura 5.14: Resultado visual do quarto teste, utilizando-se $N_s = 30$ e $Np = 14$.

Desta vez o algoritmo foi capaz de separar de forma satisfatória o humanóide do fundo, graças a informação probabilística que está muito mais coerente, já que a cor do fundo é diferente da cor do humanóide. Vale notar que novamente a informação de borda retornou um resultado forte nas retas verticais ao fundo. Porém, como a informação probabilística possui um peso cinco vezes maior que a informação de borda, o algoritmo foi capaz de contornar esse problema.

Para o quinto teste, foi utilizado uma foto de um escritório como fundo do vídeo. Os resultados visuais podem ser vistos na Figura 5.16 (utilizando $N_s = 35$ e $N_p = 18$) e

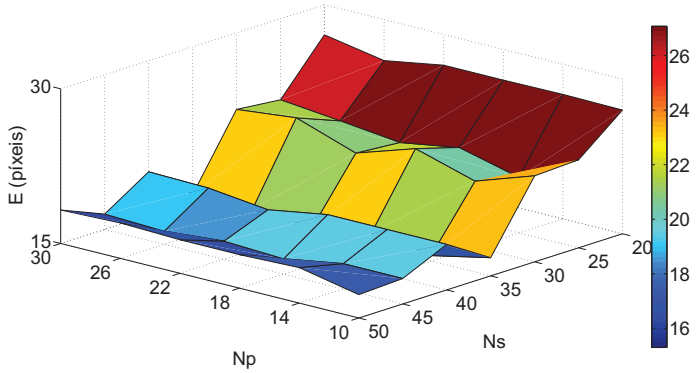


Figura 5.15: Superfície do erro referente ao quarto teste.

os resultados numéricos podem ser vistos na Figura 5.17. Como é possível perceber nesse caso, apesar do fundo da imagem conter diversas bordas, a informação de probabilidade do humanóide foi altamente precisa, fazendo com que o algoritmo de Dijkstra fosse capaz de gerar uma segmentação satisfatória.

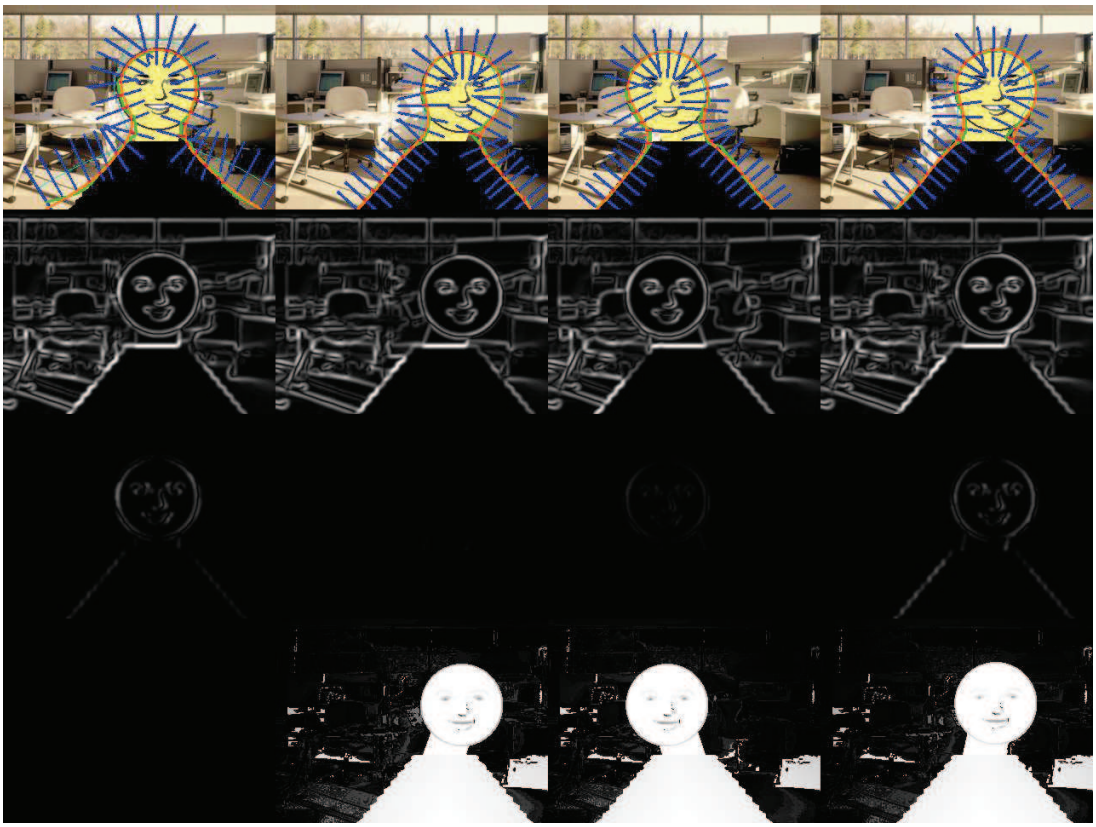


Figura 5.16: Exemplo visual do resultado ao longo da quinta seqüência de vídeo.

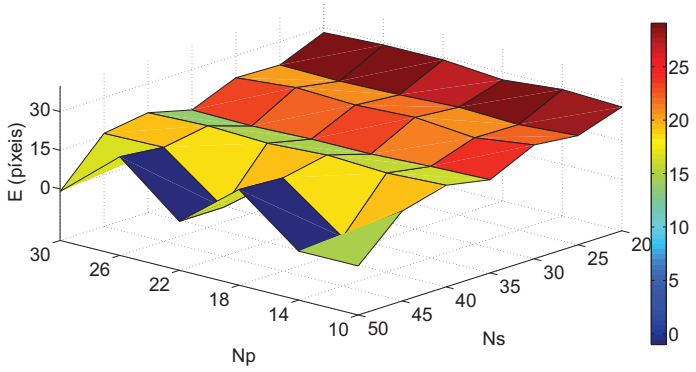


Figura 5.17: Superfície do erro para o teste com um vídeo com a foto de um escritório ao fundo.

Nos testes anteriores, parâmetros como o número de segmentos perpendiculares sobre a máscara, quantidade de pontos sobre cada um deles e a quantidade de nodos que são analisados pelo algoritmo de Dijkstra entre as retas perpendiculares foram variados. Ainda, funções periódicas com diferentes amplitudes foram adicionadas ao humanóide e diferentes fundos foram testados nas imagens. Porém, além desses parâmetros, muitos outros podem ser alterados. Dado que a silhueta gerada pelo algoritmo baseia-se em três energias, é interessante variar os pesos w_e, w_m e w_p, conforme visto na Equação 4.10, para encontrar uma combinação que retorne o melhor resultado, sendo que $w_e + w_m + w_p = 1$. Portanto, utilizando o melhor resultado do teste anterior ($N_p = 26$ e $N_s = 35$), cada um dos pesos assumiu valores dentro dos seguintes intervalos: $w_e = [0, 1]$, $w_p = [0, 1 - w_e]$ e $w_m = [0, 1 - (w_e + w_p)]$ e alguns dos resultados podem ser vistos na Tabela 5.1:

Neste caso, o menor erro foi obtido com a configuração de número 10, na qual $w_e = 0.7$, $w_m = 0.0$ e $w_p = 0.3$, e o maior erro, na configuração 2, onde $w_e = 0.2$, $w_m = 0.3$ e $w_p = 0.5$. Apesar de que o erro foi menor quando a informação de movimento não estava presente, não se pode concluir que essa informação não seja relevante, dado que o erro não teve uma variação significativa ao longo da tabela, exceto quando a informação probabilística não é utilizada, como nas configurações 4, 6, 8, 11, 13 e 14. Nesses casos, obteve-se um erro de -1, que corresponde ao fato de que segmentos perpendiculares não estavam intersectando com o humanóide, implicando em um resultado visual não satisfatório. Ainda, efetuando-se uma análise qualitativa nos resultados obtidos com as configurações 2 e 10, não é possível perceber diferenças significativas entre os resultados, como pode-se ver na Figura 5.18.

Através da simples análise da Tabela 5.1, não é possível determinar uma configuração ótima baseando-se no resultado anterior, já que isso é dependente dos fatores que influenciam a segmentação no vídeo, como iluminação, texturas, etc. Tendo em vista que a configuração utilizada até o momento $w_e = 0.1$, $w_m = 0.4$ e $w_p = 0.5$ vem se mostrando satisfatória, de forma geral, para os testes executados, considera-se que é uma

configuração adequada.

Conf.	w_e	w_m	w_p	Erro (píxeis)
1	0.1	0.4	0.5	14.7648
2	**0.2**	**0.3**	**0.5**	**17.8885**
3	0.3	0.3	0.4	13.6015
4	0.3	0.7	0.0	-1
5	0.4	0.3	0.3	14.1421
6	0.4	0.6	0.0	-1
7	0.5	0.1	0.4	13.4164
8	0.5	0.5	0.0	-1
9	0.6	0.2	0.2	14.5602
10	**0.7**	**0.0**	**0.3**	**13.3417**
11	0.7	0.3	0.0	-1
12	0.8	0.0	0.2	14.5602
13	0.9	0.0	0.1	-1
14	1.0	0.0	0.0	-1

Tabela 5.1: Tabela demonstrando alguns resultados baseado na variação dos pesos das energias.

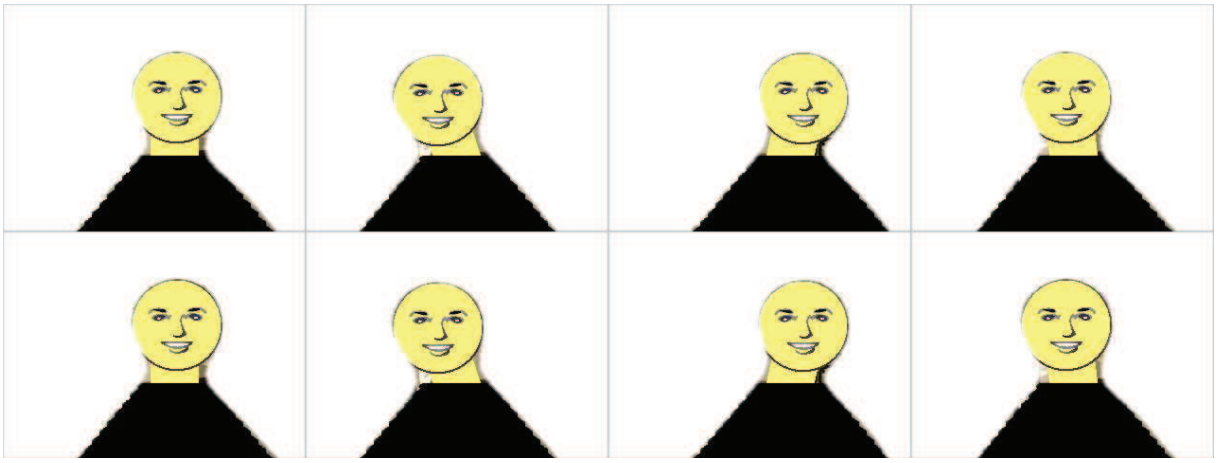


Figura 5.18: Comparação entre os resultados obtidos com a configuração 2 (primeira linha) e 10 (segunda linha).

5.3 VÍDEOS REAIS

Apesar de os testes com o humanóide retornarem resultados precisos do ponto de vista analítico, eles não refletem fielmente como o algoritmo se comporta em ambientes reais, dado que não é possível modelar completamente todos os elementos que afetam a segmentação da pessoa de forma eficaz. Portanto, testes com vídeos reais também foram feitos e os resultados foram analisados qualitativa e quantitativamente.

Os vídeos foram capturados utilizando uma *webcam* com resolução de 320×240 píxeis e 15 FPS. Tendo em vista que a geração de *ground-truths* é uma tarefa trabalhosa e que demanda uma quantidade de tempo considerável, foi feita uma subamostragem dos vídeos e os *ground-truths* foram gerados a cada cinco quadros. O erro final é dado em termos da porcentagem média de píxeis incorretamente classificados a cada quadro do vídeo.

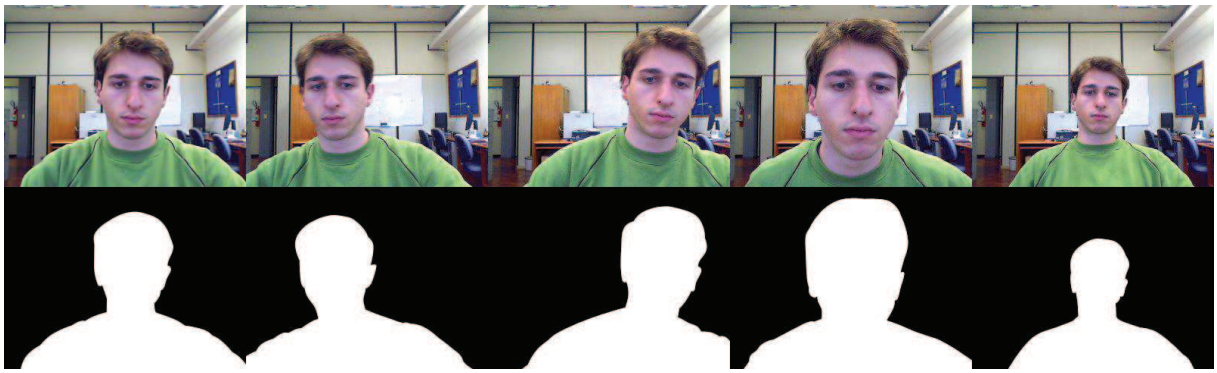
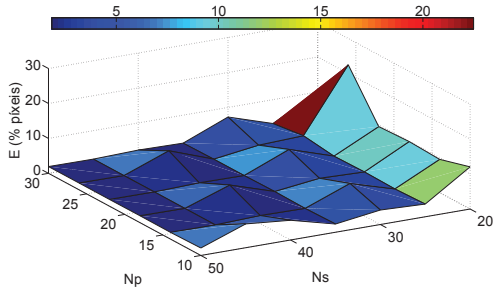
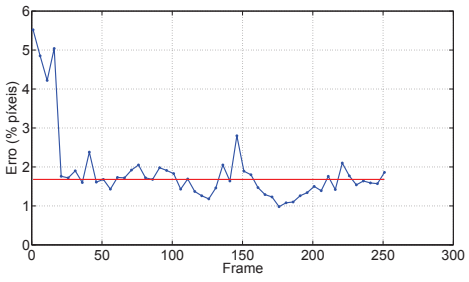


Figura 5.19: Exemplo de alguns quadros da sequência do primeiro vídeo e seus respectivos *ground-truths*.

O primeiro vídeo avaliado é de uma pessoa em um escritório. Nesse caso, pode-se perceber diversos objetos ao fundo e a iluminação do ambiente é homogênea. Alguns quadros do vídeo com seus respectivos *ground-truths* podem ser vistos na Figura 5.19.



(a) Superfície do erro para o primeiro teste real.



(b) Erro a cada quadro do vídeo utilizando $N_s = 45$ e $N_p = 14$. A reta vermelha representa a mediana do erro.

Figura 5.20: Resultado do teste com vídeo contendo uma pessoa na frente de um fundo com diversos objetos.

Mais uma vez, analisou-se os resultados alterando parâmetros como N_p e N_s. Assim como no teste com o humanóide, o algoritmo foi capaz de segmentar com precisão a pessoa, sendo que o menor erro obtido foi de $1,9\%$, utilizando $N_s = 45$ e $N_p = 14$, como demonstrado na Figura 5.20(a). Até mesmo na região dos cabelos da pessoa, que geralmente é complicado de se obter um resultado visual satisfatório, o algoritmo se comportou bem, como pode ser visto na Figura 5.21. Pode-se ver que nos primeiros quadros

a segmentação não era satisfatória e vários píxeis em torno do pescoço da pessoa não estavam sendo corretamente classificados. Esse problema aconteceu porque nos primeiros quadros a pessoa estava parada, portanto não existia informação de movimento, logo o classificador probabilístico ainda não havia sido criado. Isso fez com que o algoritmo utilizasse apenas informação de borda para gerar a silhueta final. Assim que houve um pequeno movimento (segunda coluna), o classificador probabilístico foi criado e seu resultado foi preciso, aumentando drasticamente a qualidade do resultado visual. O gráfico apresentado na Figura 5.20(b) mostra o erro obtido a cada quadro do vídeo.

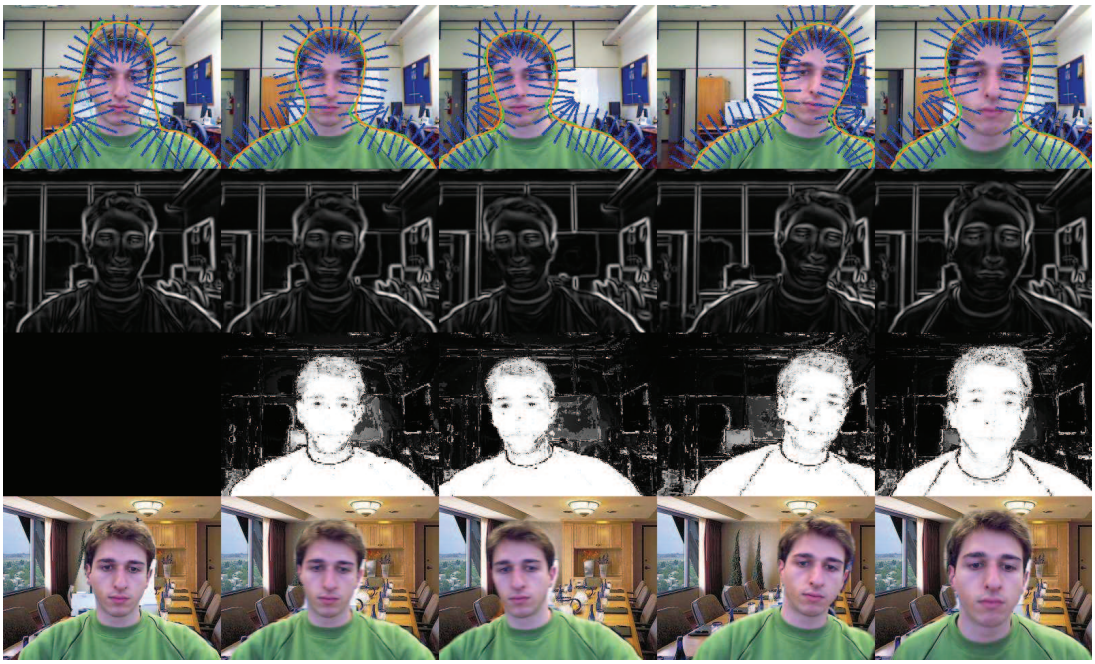


Figura 5.21: Resultado visual do teste para $N_s = 45$ e $N_p = 14$.

O segundo vídeo avaliado é de uma pessoa na frente de uma parede branca com poucos objetos ao fundo. Nesse vídeo a iluminação é mais intensa do lado direito da cena. A Figura 5.22 apresenta alguns quadros desse vídeo juntamente com os respectivos *ground-truths*.

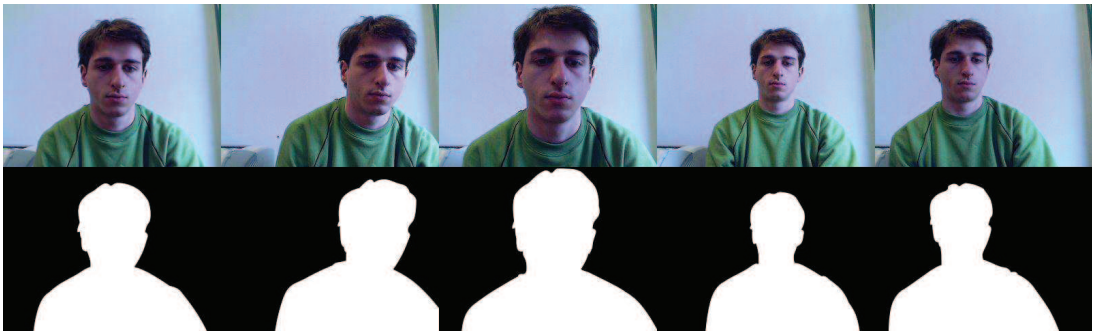
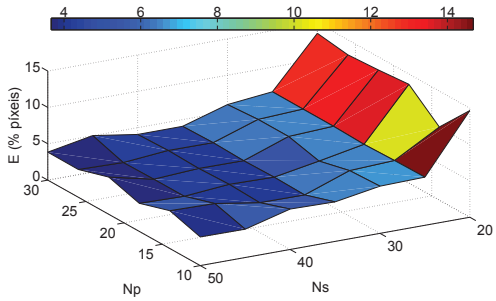


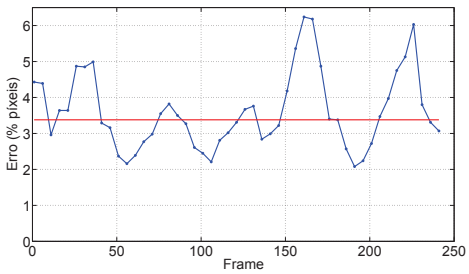
Figura 5.22: Exemplo de alguns quadros da sequência do segundo vídeo e seus respectivos *ground-truths*.

O resultado visual foi relativamente satisfatório, como pode ser observado na Figura 5.23 (utilizando $N_s = 45$ e $N_p = 26$). A informação de borda foi afetada do lado direito da imagem devido à iluminação desigual na cena. No entanto, a informação probabilística manteve-se coerente, não prejudicando a segmentação final nessa região.

Figura 5.23: Resultado visual do teste.



(a) Superfície do segundo para o primeiro teste real.



(b) Erro a cada quadro do vídeo utilizando $N_s = 45$ e $N_p = 26$. A reta vermelha representa a mediana do erro.

Figura 5.24: Resultado do teste com vídeo contendo uma pessoa na frente de um fundo sem muitos objetos.

Um problema que ocorreu nesse último caso foi que no lado esquerdo da figura havia um objeto que inicialmente estava sendo classificado como parte do primeiro plano. Quando o locutor movimentou-se, como as retas perpendiculares da extremidade esquerda da cena não estavam intersectando com ele, o objeto foi levado em consideração no cálculo da probabilidade, sendo classificado fortemente pelo classificador probabilístico como

parte do primeiro plano ao longo do vídeo. Devido a esse fato, o erro final foi maior que no teste anterior, atingindo 3,66%, utilizando $N_s = 45$ e $N_p = 26$, como mostrado na Figura 5.24(a). No decorrer do vídeo, o erro oscila bastante, como pode ser visto na Figura 5.24(b). Essa oscilação foi causada por causa do objeto que foi segmentado erroneamente no início do vídeo. Portanto, conforme a pessoa se afastava do objeto, o erro aumentava e, chegando mais próximo, o erro diminuía, como pode ser visto na terceira e quarta coluna da Figura 5.24(a), respectivamente.

5.4 DISCUSSÃO

Nesta seção foram feitos diversos testes com o sistema desenvolvido em diferentes cenários e alterando os parâmetros do sistema. Obviamente, efetuar testes exaustivos utilizando todas possíveis combinações de parâmetros é virtualmente impossível, dada a grande quantidade de parâmetros.

Tendo em vista os resultados apresentados, pode-se concluir que o resultado do sistema varia de acordo com a quantidade de retas perpendiculares posicionadas sobre a máscara, porém, não são afetados de forma significativa alterando-se o número de pontos sobre cada reta. Ainda, a restrição de que um nodo pode conectar-se a somente três nodos da reta perpendicular seguinte mostrou-se ineficaz, sendo que acabava por prejudicar a precisão do sistema na maior parte dos casos.

Com relação aos pesos de cada energia, apesar de não ser possível determinar uma combinação ótima, a escolha $w_e = 0.1$, $w_m = 0.4$ e $w_p = 0.5$ demonstrou-se adequada para os testes executados. Além disso, foi possível perceber na Tabela 5.1 que a informação probabilística é extremamente relevante para tornar o sistema mais robusto em ambientes contendo diversas bordas ao fundo da imagem.

O sistema desenvolvido foi comparado a outro sistema similar do estado da arte de segmentação de pessoas. A solução escolhida foi a apresentada em Yin et al. (2007) pelo fato de possuir um foco muito similar com o deste trabalho e também pelo fato que os autores tornaram sua base de dados de teste disponível na internet[1]. No entanto, muitos dos vídeos de teste não puderam ser utilizados por não satisfazerem as premissas deste trabalho, descritas na seção 1.2.

A primeira comparação do sistema proposto com o sistema de Yin et al. (2007) pode ser vista na Figura 5.25, utilizando-se a seqüência de teste GTTS54 (assim nomeada pelos autores). O teste foi efetuado utilizando os parâmetros $Np = 14$ e $N_s = 45$, já que essa configuração apresentou bons resultados anteriormente. A comparação dos resultados numéricos obtidos pode ser visto na Figura 5.26.

[1]Disponível em: `http://research.microsoft.com/en-us/um/people/antcrim/data_i2i/treebasedclassifiers_cvpr07_data.zip`

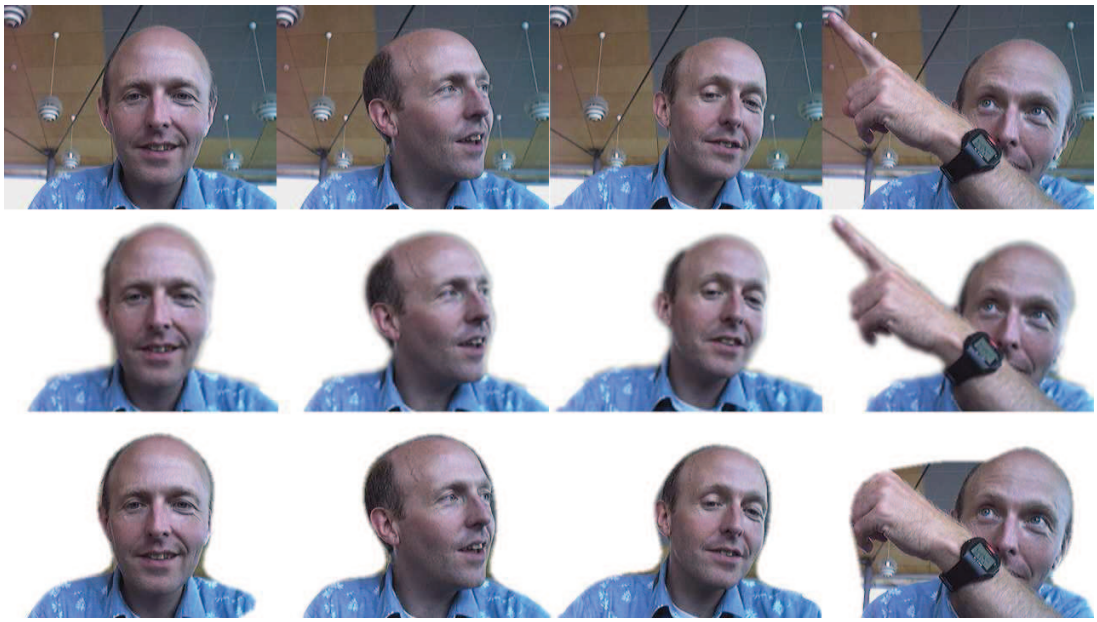
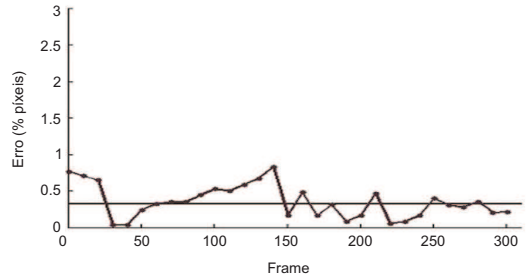
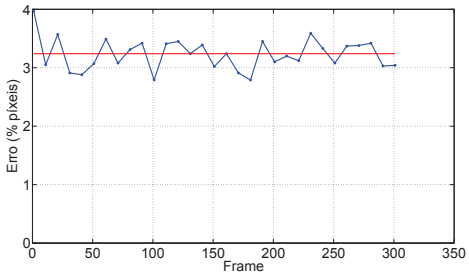


Figura 5.25: Seqüência de vídeo GTTS54. Primeira linha: alguns quadros originais do vídeo; Segunda linha: Resultado do sistema proposto por Yin et al. (2007); Terceira linha: Resultado do sistema aqui proposto.



(a) Resultado do sistema proposto por Yin et al. (2007). A reta horizontal no gráfico apresenta a mediana do erro.



(b) Resultado do sistema proposto. A reta em vermelho representa a mediana do erro.

Figura 5.26: Erro ao longo da seqüência GTTS54 com relação ao *ground-truth*.

Pode ser visto que, com exceção do último quadro da Figura 5.25, no qual parte da mão do locutor foi cortada, o sistema aqui proposto apresenta resultados equiparáveis com o sistema de Yin et al. (2007). Apesar de o erro apresentado na Figura 5.26(b) ser maior que o apresentado na Figura 5.26(a), ele é aceitável, dado os resultados visuais apresentados.

O fato de que a mão da pessoa não apareceu na cena pode ser explicada pela restrição utilizada na ligação entre os nodos e também pela quantidade de retas perpendiculares posicionadas sobre a pessoa. No entanto, mesmo removendo-se a restrição, ou seja, permitindo que cada nodo possa se ligar a qualquer outro nodo da próxima reta, esse problema não foi resolvido, como apresentado na Figura 5.27. Como pode-se perceber na terceira

coluna da Figura, as retas perpendiculares não circulam totalmente a mão do locutor. Além disso, deve-se notar que o algoritmo de Dijkstra leva em consideração o custo global da curva, o que explica o resultado. Na última coluna, nota-se que não há intersecção das retas perpendiculares com o dedo da pessoa, portanto, não sendo levado em consideração na silhueta final.

A remoção da restrição não corrigiu o problema ocorrido com a mão do locutor e ainda criou um novo problema, apresentado em todos os quadros da Figura 5.27. No canto inferior direito da imagem, a silhueta gerada ignorou o ombro do locutor, prejudicando fortemente o resultado visual final. Conclui-se então que, apesar das limitações que a restrição impõe ao sistema, ela previne resultados incoerentes na maior parte dos casos.

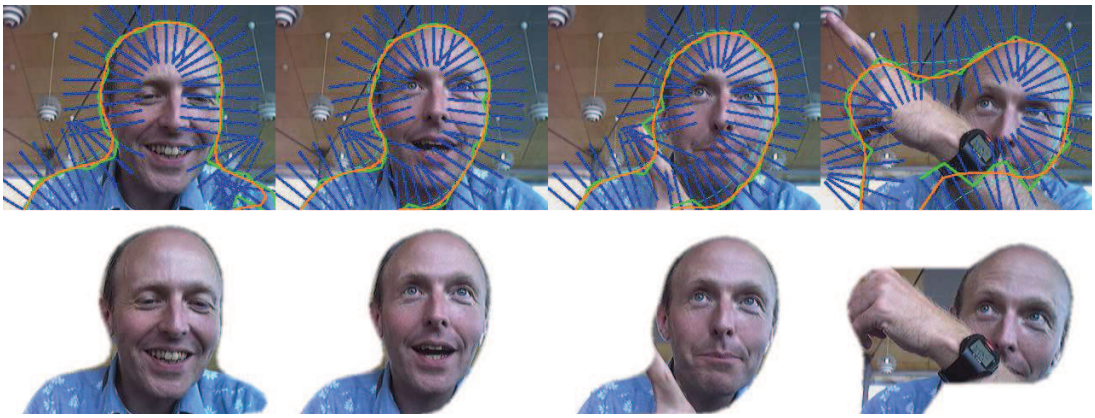


Figura 5.27: Seqüência de vídeo GTTS54 eliminando todas as restrições.

Outra seqüência de vídeo utilizada para teste foi a GTTS56. No entanto, os resultados explícitos desse teste não estão descritos no artigo dos autores. O resultado visual utilizando o sistema proposto é apresentado na Figura 5.28 (utilizando a configuração $N_s = 45$ e $N_p = 14$).

Nesse teste é possível observar que a segmentação do lado esquerdo da pessoa não foi bem sucedida, mais especificamente na região do cabelo. Esse problema pode ser explicado pelo fato de que a informação de probabilidade nesses pontos não era forte. O principal motivo é que a maior parte é incidida por uma luz, fazendo com que sua cor seja mais clara. Porém, naquela região, havia pouca luz incidindo, o que modificou a cor e fez com que a probabilidade daqueles píxeis pertencerem ao primeiro plano fosse menor. Esse problema não ocorreu do lado direito da imagem.

Outro fato que deve ser notado é que como os ombros do locutor não estão aparecendo na imagem, a máscara que foi posicionada sobre a pessoa conteve alguns segmentos perpendiculares não intersectando com a pessoa. Isso fez com que a silhueta final acabasse se estendendo para as bordas da imagem, causando um efeito visual indesejado e diminuindo a precisão do sistema, como pode ser visto na Figura 5.29. Inicialmente o erro era mais baixo, sendo que aumentou a medida que a silhueta se estendeu para as bordas

da imagem, estabilizando-se no final.

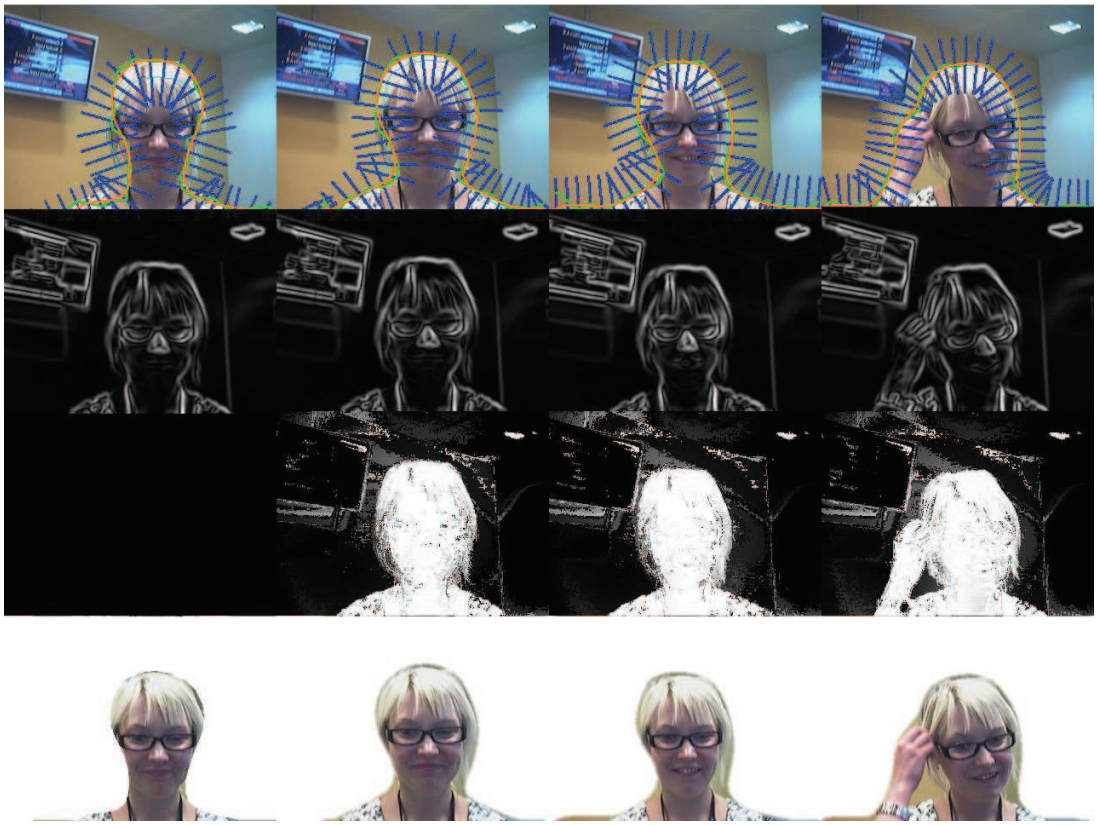


Figura 5.28: Seqüência de vídeo GTTS56.

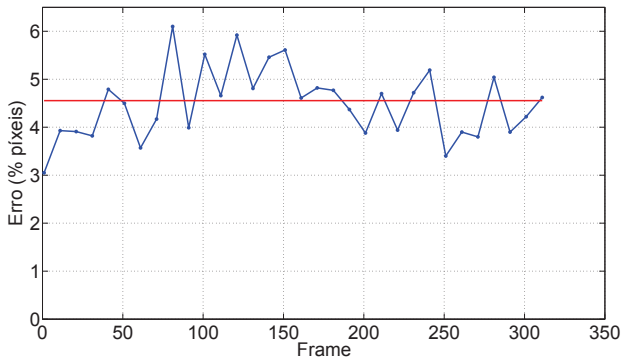


Figura 5.29: Teste executado com a seqüência de teste GTTS56. A reta em vermelho representa a mediana do erro.

De forma geral, considera-se que o sistema, apesar de não superar os resultados apresentados em Yin et al. (2007), apresenta resultados satisfatórios e equiparáveis. Além disso, considera-se que ele atende os objetivos propostos no início deste trabalho. As Tabelas 5.2 e 5.3 apresentam os melhores resultados obtidos quando os resultados foram analisados com a métrica do supremo e comparados aos *ground-truths*, respectivamente. Com a métrica do supremo, observa-se que o maior erro foi de 15,29 píxeis de distância

entre a silhueta gerada e a silhueta real do humanóide. Tendo em vista que a resolução da imagem é de 320×240 pontos, pode-se considerar esse erro como sendo baixo. Além disso, analisando-se os resultados comparados com os *ground-truths*, a média dos erros levando em consideração todos os vídeos apresentados foi de 3,32%, também considerado baixo.

	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
E (píxeis)	13,0	13,0	11,0	15,29	13,89
FPS	8,01	7,65	8,97	8,42	7,43

Tabela 5.2: Resultados obtidos utilizando a métrica do supremo.

	Fundo c/ objetos	Fundo s/ objetos	GTTS54	GTTS56
E (% píxeis)	3,66	1,9	3,23	4,49
FPS	7,20	8,60	7,76	7,90

Tabela 5.3: Resultados obtidos comparando com *ground-truths*.

Ainda, vale notar que o sistema de Yin et al. (2007) requer um treinamento e executa em torno de 1,2 FPS, enquanto que o sistema proposto não necessita de qualquer treinamento e executa a uma taxa média de 8 FPS utilizando o código desenvolvido em C++ não otimizado.

6 CONCLUSÃO E TRABALHOS FUTUROS

Segmentação de objetos é uma das principais atividades em visão computacional, já que a obtenção de qualquer informação referente ao objeto depende de uma segmentação satisfatória. Dentre a grande gama de aplicações para segmentação, uma que vem ganhando grande espaço tanto no mundo comercial quanto acadêmico é a segmentação de pessoas. Dada a crescente evolução em termos de *hardware* e a popularização da internet, videoconferências vêm sendo alvo desse tipo de aplicação. Isso se deve pelo fato de que videoconferências são muito utilizadas, principalmente por empresas, para organizar uma reunião economizando recursos que seriam gastos com viagens, por exemplo. Nesse caso, a segmentação é útil para permitir que futuros processamentos sejam efetuados com as imagens, como por exemplo substituir o fundo da imagem de cada participante por um fundo padrão, aumentando a sensação de imersão do participante. Tendo em vista esse cenário, este trabalho apresentou o desenvolvimento de um sistema de segmentação de locutores em tempo real focado para videoconferências.

O sistema foi desenvolvido na linguagem C++ e utilizando a biblioteca OpenCV e a segmentação final foi dada pelo algoritmo de Dijkstra. Inicialmente, a face do locutor foi detectada e rastreada ao longo do vídeo. Baseando-se na sua posição e tamanho, uma região de busca foi criada utilizando uma máscara (*template*) posicionada sobre o locutor. Essa região de busca continha diversos pontos chaves sobre os quais a silhueta deveria ser formada, sendo interpretada como um grafo. Logo após, um mapa de bordas, informação de movimento e uma componente probabilística foram extraídas da imagem e combinadas linearmente para formar uma energia final que guiou o algoritmo de Dijkstra para determinar qual era a silhueta da pessoa.

Para testar o sistema, vídeos reais foram comparados com *ground-truths* e, ainda, um humanóide 2D foi desenvolvido para permitir que os resultados fossem analisados de forma mais precisa. Por fim, o sistema também foi testado com vídeos disponibilizados por autores de trabalhos similares. Conforme mencionado no decorrer do trabalho, devido à grande quantidade de parâmetros, não foi possível testar todas combinações possíveis e determinar qual é a ótima. No entanto, dado os resultados obtidos, considera-se que a utilização de $N_p = 18$, $N_s = 35$, $w_e = 0.1$, $w_m = 0.4$ e $w_p = 0.5$ é adequada.

Além disso, comparou-se os resultados obtidos com outras soluções. Dentre estas, a mais adequada a comparações com este trabalho é a de Yin et al. (2007). Comparando-se essa solução com a aqui proposta, pôde-se notar que ela apresentou resultados visuais mais precisos, sendo capaz de detectar diversos tipos de movimento, como pôde ser visto no caso da Figura 5.26. No entanto, globalmente os resultados do sistema proposto são visualmente equiparáveis. Enquanto que a solução de Yin et al. (2007) foi executada a uma taxa de 1,2 FPS, necessitando de treinamento prévio, o sistema proposto executou a

uma média de 8 FPS mesmo com o código não otimizado, não sendo necessário nenhum tipo de treinamento. Portanto, pode-se concluir que o sistema proposto atinge os objetivos descritos no início deste trabalho, sendo capaz de segmentar a imagem de uma pessoa em tempo real para videoconferências.

O trabalho apresentado pode ser estendido das seguintes formas: utilizar um método adaptativo para os pesos das energias, de modo que possam se adequar ao ambiente em que o sistema está inserido; utilizar técnicas de aprendizagem do plano de fundo, minimizando a influência das energias de objetos que não pertençam ao locutor; a combinação das energias pode ser modificada, explorando-se combinações mais complexas do que uma combinação linear; restrições podem ser impostas à curva final, procurando garantir que seu formato final assemelhe-se ao de uma pessoa; em termos de desempenho, programação em GPU certamente implicará em grandes ganhos.

BIBLIOGRAFIA

BALLERINI, L. Multiple genetic snakes for people segmentation in video sequences. In: *SCIA'03: Proceedings of the 13th Scandinavian conference on Image analysis*. Berlin, Heidelberg: Springer-Verlag, 2003. p. 275–282.

BINS, J.; JUNG, C. R.; DIHL, L. L.; SAID, A. Feature-based face tracking for videoconferencing applications. In: *ISM '09: Proceedings of the 2009 11th IEEE International Symposium on Multimedia*. Washington, DC, USA: IEEE Computer Society, 2009. p. 227–234.

BRADSKI, G.; KAEHLER, A. *Learning OpenCV: Computer Vision with the OpenCV Library*. Cambridge, MA: O'Reilly, 2008.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Algoritmos: Teoria e Prática*. : Editora Campus, 2002.

CRIMINISI, A.; CROSS, G.; BLAKE, A.; KOLMOGOROV, V. Bilayer segmentation of live video. In: *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2006. p. 53–60.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*. Washington, DC, USA: IEEE Computer Society, 2005. p. 886–893.

DAVIS, A. W.; WEINSTEIN, I. M. The business case for videoconferencing. March 2005.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, Springer Berlin / Heidelberg, v. 1, p. 269–271, 1959.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification (2nd Edition)*. : Wiley-Interscience, 2000.

FARIN, G. *Curves and surfaces for computer aided geometric design: a practical guide*. San Diego, CA, USA: Academic Press Professional, Inc., 1988.

FORSYTH, D. A.; PONCE, J. *Computer Vision: A Modern Approach*. : Prentice Hall, 2002.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. In: *Proceedings of the Second European Conference on Computational Learning Theory*. London, UK: Springer-Verlag, 1995. p. 23–37.

GAVRILA, D.; PHILOMIN, V. Real-time object detection for "smart" vehicles. *IEEE International Conference on Computer Vision*, IEEE Computer Society, v. 1, p. 87, 1999.

GOLDBARG, M. C.; LUNA, H. P. L. *Otimização combinatória e programação linear : modelos e algoritmos*. : Editora Campus, 2000.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing (2nd Edition)*. : Prentice Hall, 2002.

HARRISON, R.; CLAYTON, W.; WALLACE, P. Can telemedicine be used to improve communication between primary and secondary care? *BMJ*, v. 313, n. 7069, p. 1377–1380, 1996.

JABRI, S.; DURIC, Z.; WECHSLER, H. Detection and location of people in video images using adaptive fusion of color and edge information. In: *in Proc. 15th International Conference on Pattern Recognition*. 2000. p. 627–630.

JAIN, A. K. *Fundamentals of digital image processing*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

JIA, H.-X.; ZHANG, Y.-J. Fast human detection by boosting histograms of oriented gradients. In: *Proceedings of the Fourth International Conference on Image and Graphics*. : IEEE Computer Society, 2007. (ICIG '07), p. 683–688.

KIM, K.; CHALIDABHONGSE, T. H.; HARWOOD, D.; DAVIS, L. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, v. 11, n. 3, p. 172 – 185, 2005. ISSN 1077-2014. Special Issue on Video Object Processing.

KOLMOGOROV, V.; CRIMINISI, A.; BLAKE, A.; CROSS, G.; ROTHER, C. Bi-layer segmentation of binocular stereo video. In: *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*. Washington, DC, USA: IEEE Computer Society, 2005. p. 407–414.

LIN, Z.; DAVIS, L.; DOERMANN, D.; DEMENTHON, D. Hierarchical part-template matching for human detection and segmentation. In: *IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007*. 2007. p. 1–8.

OPPENHEIM, A. V.; SCHAFER, R. W.; BUCK, J. R. *Discrete-time signal processing (2nd ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1999.

PICCARDI, M. Background subtraction techniques: a review. In: *Proc. of IEEE SMC 2004 International Conference on Systems, Man and Cybernetics*. The Hague, The Netherlands: , 2004. v. 4, p. 3099–3104.

PLATANIOTIS, K. N.; VENETSANOPOULOS, A. N. *Color Image Processing and Applications (Digital Signal Processing)*. 1. ed. : Springer, 2000. Hardcover.

ROSS, M. G.; KAELBLING, L. P. Learning static object segmentation from motion segmentation. In: *AAAI'05: Proceedings of the 20th national conference on Artificial intelligence*. : AAAI Press, 2005. p. 956–961.

RUDIN, W. *Principles of Mathematical Analysis, Third Edition*. 3rd. ed. : McGraw-Hill Science/Engineering/Math, 1976.

SEGEN, J. A camera-based system for tracking people in real time. In: *Proceedings of the 13th International Conference on Pattern Recognition*. 1996. v. 3, p. 63–67.

SHI, J.; TOMASI, C. *Good Features to Track*. Ithaca, NY, USA, 1993.

TUZEL, O.; PORIKLI, F.; MEER, P. Human detection via classification on riemannian manifolds. In: *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR 2007*. 2007. p. 1–8.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 1, p. 511–I–518 vol.1, April 2001.

WU, B.; NEVATIA, R. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In: *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Washington, DC, USA: IEEE Computer Society, 2005. p. 90–97.

YIN, P.; CRIMINISI, A.; WINN, J.; ESSA, I. Tree-based classifiers for bilayer video segmentation. In: *In CVPR*. 2007.

ZHAO, L.; DAVIS, L. Closely coupled object detection and segmentation. In: *10th IEEE International Conference on Computer Vision, 2005. ICCV 2005*. 2005. v. 1, p. 454–461 Vol. 1.

ZHAO, S.-L.; LEE, H.-J. Human silhouette extraction based on hmm. In: *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2006. p. 994–997.

ZHU, Q.; YEH, M.-C.; CHENG, K.-T.; AVIDAN, S. Fast human detection using a cascade of histograms of oriented gradients. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2006. v. 2, p. 1491–1498.